Redbooks

ibm.com/redbooks

In-Place Analytics with Live Enterprise Data with IBM DB2 Query Management Facility

Doug Anderson

Mike Biere

Linus Olson

Shawn Sullivan

IBM

Redbooks

International Technical Support Organization

In-Place Analytics with Live Enterprise Data with IBM QMF

October 2016

SG24-8370-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (October 2016)

This edition applies IBM DB2 Query Management Facility for z/OS V11.2.1.

Contents

Notices

Trademarks

Preface

IBM® DB2® Query Management Facility™ for z/OS® provides a zero-footprint, mobile-enabled, highly secure business analytics solution. IBM QMF™ V11.2.1 offers many significant new features and functions in keeping with the ongoing effort to broaden its usage and value to a wider set of users and business areas. In this IBM Redbooks® publication, we explore several of the new features and options that are available within this new release.

This publication introduces TSO enhancements for QMF Analytics for TSO and QMF Enhanced Editor. A chapter describes how the QMF Data Service component connects to multiple mainframe data sources to accomplish the consolidation and delivery of data.

This publication describes how self-service business intelligence can be achieved by using QMF Vision to enable self-service dashboards and data exploration. A chapter is dedicated to JavaScript support, demonstrating how application developers can use JavaScript to extend the capabilities of QMF. Additionally, this book describes methods to take advantage of caching for reduced CPU consumption, wider access to information, and faster performance.

This publication is of interest to anyone who wants to better understand how QMF can enable in-place analytics with live enterprise data.

Authors

This book was produced by a team of specialists from around the world.

Doug Anderson is a Senior Product Specialist for QMF based in the US. He has 14 years of experience in Business Intelligence and Performance Management. He holds an Applied Science degree in Network Administration. His areas of expertise include analytics, user education, and product support.

Mike Biere is a Senior Specialist in QMF located in Cincinnati, Ohio, US. He is a long-standing Business Analytics professional. He has 35 years of experience in Business Analytics and 39 years of IT experience starting as a Systems Engineer with IBM in 1978. He holds a B.S. from San Francisco State University and an M.S. from the University of Cincinnati in Terrestrial Ecology. His areas of expertise include database, analytics, and other aspects of user computing. He has written extensively about Business Analytics and published two books and numerous journal articles about the subject.

Linus Olson is a Product Manager in the US. He has 15 years of experience in the Business Intelligence field. He holds a degree in Computer Science/Business Management/Cultural Studies from the University of Minnesota, Twin Cities. His areas of expertise include QMF and general Business Intelligence.

Shawn Sullivan is a Senior Consultant for Rocket Software in the US. He has 18 years of experience with IBM QMF and DB2 Tools. He holds degrees in Engineering from MIT and Education from Harvard. His areas of expertise include delineating and solving problems, installing, configuring, and getting the most out of software applications. He has written extensively about QMF in two previous IBM Redbooks and many custom solution documents.

Special thanks to the following people for their contributions in leading this project:

- Blanca Borden
- Deanna Ziemba
- Martin Keen

Thanks to the following people for their contributions to this project:

- Andy Seuffert
- Andrea Reid
- Mark Flores
- Shoki Ansari
- Jay Bruce
- Claudia Franco
- Joe Sacco
- Samantha Buhler
- Mike Carlson
- George Smyth

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introducing IBM DB2 Query Management Facility for z/OS

IBM DB2 Query Management Facility for z/OS V11.2.1 offers many significant new features and functions in accordance with the ongoing effort to broaden its usage and value to a wider set of users and business areas. In this book, we explore several of the new features and options that are available in this new release.

As you scan the table of contents, you might see a particular topic that holds more interest than others. Therefore, each section can be read as a stand-alone module and does not depend on another section. Look at Query Management Facility (QMF) as a multi-platform, multi-function solution that shares objects and function as much as possible across platforms and systems. For example, a typical SQL query object that is stored in the traditional QMF catalog can be used by workstation, web, and mobile users. More sophisticated objects, such as a graphical dashboard, are not available to a TSO user but can be used within the other components.

QMF radically changed since its inception as a 3270-based query and reporting tool for TSO and IBM CICS®. Its core functions remain in those environments but so much more is available to an enterprise that is seeking a business intelligence and business analytics solution for all its users. Additional resources are available beyond the scope of this book, such as videos and more, that illustrate many of the enhancements that make QMF a complete analytics solution, providing dashboarding, analytics reports, and visualizations to all users.

This chapter includes the following topics:

- What is new in IBM Query Management facility for z/OS and why do I care?
- Setting the stage for data access
- Self-service business analytics, what and why?
- Caching in QMF and the use of JavaScript
- TSO enhancements

Now, we delve into several of the new and exciting features of V11.2.1.

1.1 What is new in IBM Query Management facility for z/OS and why do I care?

Many additions and changes occurred to the QMF solution, far too many to describe in this book. The QMF solution continues to be enhanced and extended to keep pace with its position as a world-class, enterprise business intelligence and analytics suite. However, three significant new features were added based on many requests and current market trends:

- Easy and high-performing access to enterprise data on IBM z Systems®: IBM Information Management System (IBM IMS™), Virtual Storage Access Method (VSAM), Sequential, System Management Facilities (SMF), and Adaptable Database System (Adabas).
- Enterprise data federation. Cross-data sources, such as DB2 and ADABAS, need to be joined but setting up this environment can be daunting and highly inefficient.
- Self-service Business Analytics for users. Users want to create their own analytical objects (dashboards and charts) and communicate their findings without requiring technical skills.

The current thinking among most businesses today is that they cannot afford to continue to move data and restructure data, which flood the enterprise. The new mind-set is to keep the data where it is captured and access it in place. Much of the data within any enterprise resides on the mainframe in sources, such as IMS or VSAM. Point solutions are available to access these sources but many are insufficient.

QMF has a new feature, which is called *Data Service*, that provides the access method and the federation capabilities that are in great demand. This new feature uses server-side processing and heavily uses z Systems Integrated Information Processors (zIIP) processors (up to 99%) and parallelism to deliver information from its sources with extreme efficiency.

By performing powerful data access and federation on System z and zIIP engines, enterprises can create analytics queries across different z Systems resources without affecting their CPU costs.

For the user who wants to access enterprise data and quickly create a chart or dashboard and collaborate with others about their results easily, QMF has a new feature, which is called *QMF Vision*. QMF Vision is a simple point and click, drag and drop builder interface that is designed for the rapid development of charts and dashboards with minimal required skill.

Many enhancements are available for the TSO user, too. In this book, we describe two enhancements that are interesting to the existing QMF community:

- QMF Analytics for TSO is an easy to use, menu-driven extension to QMF that provides enhanced graphics and statistical analysis for users.
- QMF Enhanced Editor is an editor that is based on Interactive System Productivity Facility (ISPF) and is tightly integrated with both QMF and ISPF. This new capability is for users who do not use the standard QMF query and procedure edit panels and functions.

1.2 Setting the stage for data access

In today's business environment, we see the emergence of a seemingly endless variety of new data sources, such as Hadoop and other unstructured information. The rate at which this data is accumulated tends to make the act of collecting it and then moving it to another data source prohibitive in cost and time. And, many systems that are run on the mainframe use more traditional non-relational data stores, such as VSAM and IMS. Sometimes, it feels as though we are drowning in data.

Like it or not, much of this information can be critical to individuals who want to perform various analyses to measure, check, verify, and predict business information. So, we see a vast array of data and data types, which are spread across the enterprise and housed in external sites, that we want to include in our analytics.

Data that is contained in sources, such as IMS and VSAM, is often copied to a relational model, such as DB2 or others, to provide easier access to the data from analytics tools, such as QMF. In many cases, this data needs to be joined to other existing data. If the information is not copied to a relational source or if data sources are mixed, you are required to join it across data sources and cross-platform, which is defined as *data federation*.

In many cases, combining data can be inefficient and costly from a performance perspective. We use IMS as an example. IMS DBAs are diligent in ensuring that data is available and the access performance is optimized. They tried to use other business analytics data access solutions only to see degraded IMS performance or a slow access method.

What if you leave the data where it resides and access it from QMF in a cost-efficient and high-performing manner? What if you can combine data sources, such as IMS, VSAM, or Adabas, with other data sources efficiently? We describe this capability in Chapter 2, "Virtualizing data by using QMF Data Service" on page 7.

As you delve into the data access material, think of your own environment and what data requires access for your business analysis. Do you have IMS or VSAM data that is beneficial to users without needing to move it or perform extract, transform, and load (ETL) processes? Are you required to join (federate) data across data sources and platforms efficiently?

1.3 Self-service business analytics, what and why?

The arena of business analytics is a difficult environment to navigate. Over the years, the key players and offerings changed. The supported platforms supported changed and, in particular, the user interfaces changed. If you spend enough time and acquire enough technical skill, you can eventually emerge with the answers to the business problems that you are addressing with any tool that is available. Constant shuffling and repositioning by business intelligence and analytics vendors occur to win your favor and establish their ability to solve your problems.

If you have any experience with these tools, you know that certain features and functions are easier than others to learn and apply. For years, an effort existed to create tools with a massive range of features and functions that also offered ease of use and simplicity. This task is daunting for both the vendor and prospective users.

Reality set in where everyone realized that a wide range of users and user types existed, including a large population that wanted to use an analytics tool to create and disseminate their findings but that did not require a vast array of skills to accomplish this analysis. Many of them realized that they did not have time to learn a full-function tool but that a simplified user interface with options to access key data and place it on a simple dashboard met their requirements. They might want one element (chart) in a dashboard that linked to another element so they drilled through their data for additional detail.

In today's terms, this capability is described as *self-service business intelligence (BI)/analytics*. Users want an easy, uncomplicated means of creating their own analysis. The QMF Vision component is designed to provide these capabilities and the capability to communicate the user's results with others. QMF Vision can be applied against various data sources, including QMF queries, and guide the user through a set of simple steps to produce the required results.

So what about your environment? You might already have in-house tools that perform business intelligence and analytical functions. We suggest that you look at QMF Vision and its capabilities. It is included in IBM DB2 Query Management Facility for z/OS V11.2.1 and available to all users.

1.4 Caching in QMF and the use of JavaScript

Hidden capabilities are in QMF for Workstation and QMF for IBM WebSphere®, including caching in QMF and the use of JavaScript. Caching provides a means for executing a query and storing the results for repetitive use so that you do not need to execute the same query over and over.

We use the example of a shared dashboard where one of the presentation elements, a chart, is viewed by several users. The data does not change often or it might be historical and static in content. Instead of multiple users that run the same query, the first user or perhaps a scheduled task initiates the query and the results are stored in a cache. Subsequent users take advantage of the cached results, and no addition load is placed on the source database.

We emphasize the expanded number of sources of data for QMF beyond its original access to DB2 only. With the ever burgeoning amount of data that is available on various web sites, you might want to incorporate information from one of these sites in your business analyses. If you or someone in your organization is skilled in JavaScript, you can access this information and add it to one of the many available data sources. Therefore, we continue to extend the reach of QMF to new, modern data sources as they emerge.

1.5 TSO enhancements

This section describes two TSO enhancements: QMF Analytics for TSO and QMF Enhanced Editor.

1.5.1 QMF Analytics for TSO: New business charts and analytics

QMF Analytics for TSO extends the capabilities of the more traditional mainframe user with a 3270 terminal interface. QMF evolves with new, graphical user interfaces and options, and IBM adds value and function to improve the experience for existing users who continue to work within the more traditional terminal interface.

This new feature offers two significant functional areas:

- Enhanced business charts and graphs within a simple, menu-driven interface with prompts for the user and options that facilitate the creation of powerful analytics with minimal required skill.
- Statistical analysis methods that use the same approach as the new charts and graphs. The same menu-driven, “fill in the blanks with user prompts” approach is used here to extend the analytic capabilities of QMF so that you do not need to learn special syntax to perform complex analyses.

To see the available options and how to interact with data that is fetched by QMF, see 6.1, “QMF Analytics for TSO” on page 174. Now, a new analytics object can be saved in the QMF catalog for subsequent sharing and repetitive use.

1.5.2 QMF Enhanced Editor

QMF contains an edit command panel that is simple to use but, conversely, limited in function. In a text-based 3270 environment, you do not have many of the features that are available in a graphical user interface (GUI) that you see with a workstation or web interface.

Because of the complexity and length of many QMF queries and procedures, the default editor can be a bit cumbersome to perform tasks, such as splitting lines or duplicating text and other options someone with an ISPF background prefers. In previous releases, you can always invoke ISPF as an edit option and gain benefits that way. However, in so doing, you did not interact with QMF. QMF merely gave you the ability to pass text back and forth.

The enhanced editor offers traditional ISPF interactions and several QMF interactions, such as the ability to execute QMF objects from within the editor, listing QMF objects, copying QMF objects, and previewing a small subset of the tables that are being accessed to test the data before you run the entire query.

Colorization of query and procedure syntax, such as highlighting an error in red or any of the available colors in the palette, might be customized by the user to their specifications. The enhanced editor is a tremendous productivity improvement for anyone who is creating or maintaining QMF queries and procedures. The more complex the number of lines in a query or procedure, the greater the ability of the user to interact with it.

We think that your usage of QMF and the many benefits that it offers are enhanced by this book and the new features that we describe.

2

Virtualizing data by using QMF Data Service

This chapter includes the following topics:

- Overview of QMF Data Service
- Configuring access to the data
- Using QMF Data Service Studio to create virtual tables
- Accessing the QMF Data Service data
- Additional considerations
- Summary

2.1 Overview of QMF Data Service

QMF Data Service (QDS) is a component of the IBM DB2 Query Management Facility for z/OS V11.2.1 product that connects to several mainframe data sources: DB2, IBM Information Management System (IMS), Virtual Storage Access Method (VSAM) files, Adaptable Database System (Adabas), System Management Facilities (SMF) records, system logs, and operation logs. QDS provides SQL access to the data in these sources. The files and tables in these sources are available to other QMF family members as tables that are all in a single SQL-compliant relational database.

Therefore, the data from these back-end data sources can be combined in a single SQL request (JOIN, UNION, or SUBQUERY). In addition, this connection, mapping, and federation can be extended to include data from DB2 on other platforms: IBM i, Windows, Linux, and AIX, including the DB2 that is used with IBM InfoSphere® Federation Server. All of the data sources that are supported by IBM InfoSphere Federation Server are available to the QDS.

To accomplish this consolidation and delivery of data, QDS stores only metadata about these data sources as objects, which are called *virtual tables*, with no actual migration, movement, or duplication of the bulk data. Instead, the source data remains in place and only the subset of the data that is specified by the SQL request is pulled and delivered on demand (Figure 2-1). A virtual table seems similar to the view in DB2 or other relational databases, which, for an introduction, is not a bad way of thinking about a virtual table.

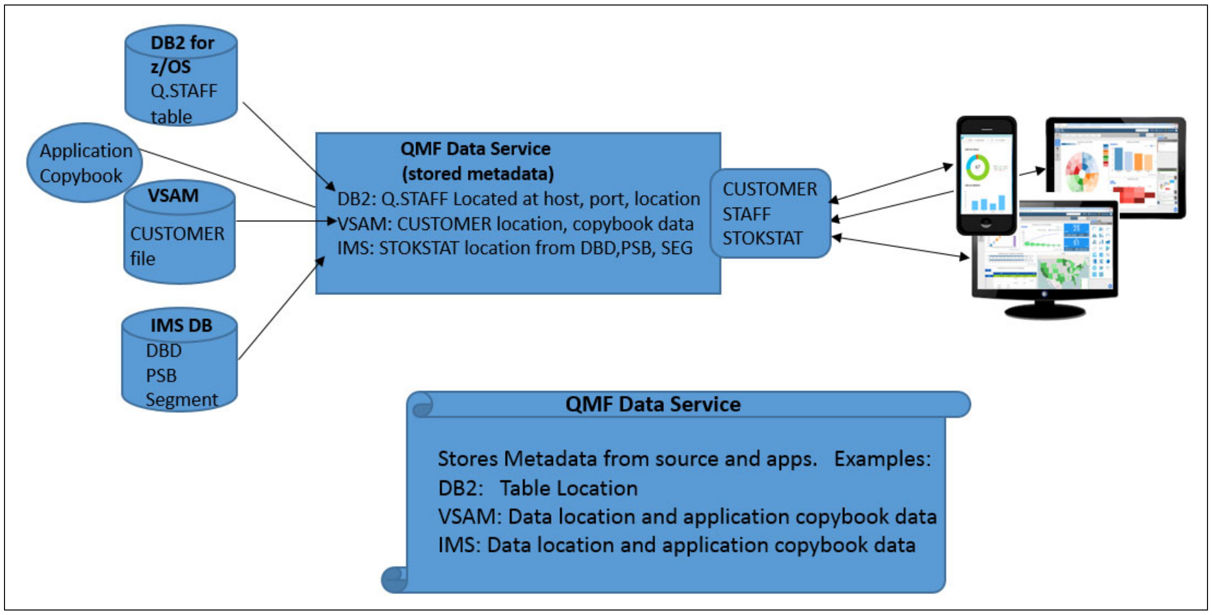

DB2 for z/OS
Q.STAFF table
Application Copybook
VSAM
CUSTOMER file
IMS DB
DBD
PSB
Segment
QMF Data Service
(stored metadata)
DB2: Q.STAFF Located at host, port, location
VSAM: CUSTOMER location, copybook data
IMS: STOKSTAT location from DBD,PSB, SEG
CUSTOMER
STAFF
STOKSTAT
QMF Data Service
Stores Metadata from source and apps. Examples:
DB2: Table Location
VSAM: Data location and application copybook data
IMS: Data location and application copybook data

Figure 2-1 QMF Data Service

2.1.1 Why is QMF Data Service useful?

QMF Data Service (QDS) makes certain data pulls easier, cheaper, and faster than they were before. Also, certain data pulls are possible that were not possible before.

For example, QDS offers the following benefits:

- Easier: QDS pulls data from multiple DB2 subsystems. Before, a client application made multiple connections in turn to one DB2 subsystem after another and pulled the data in separate query requests. Or, the user used three-part names if they were set up but then faced certain restrictions on the SQL that was supported. With QDS, the tables from many DB2 subsystems are presented as though they are in a single relational database and are available for JOINS, APPENDS, and so on.
- Cheaper: QDS requires less CPU time and offloads more workload to the z Systems Integrated Information Processor (zIIP) engine. QDS was developed to off-load up to 99% of its workload to the zIIP engine.
- Faster: QDS can perform by using parallel processing and implementing Map Reduce.
- Now possible: QDS makes it possible to federate (JOIN, UNION, SUB QUERY, and so on) tables on different DB2 subsystems in a single SQL statement. Furthermore, QDS makes it possible for SQL to access VSAM data, SMF data, SYSTEM log data, and other sequential files. All of this file-based data can also be federated with DB2 data in a single SQL statement. Similarly, any of the other supported data sources, such as IMS and Adabas, can be queried individually or in federation with any of the supported data sources.

2.2 Configuring access to the data

This section addresses the following topics:

- Time guidance
- Configuring access to DB2
- Configuring access to IMS data
- Configuring access to VSAM data

2.2.1 Time guidance

Time guidance starts after QMF Data Service (QDS) is installed and customized, as described in the documentation at the DB2 Query Management Facility Library web page at this website:

`http://www.ibm.com/support/docview.wss?uid=swg27048346`

For more information, see the following documents:

- *DB2 QMF Data Service Getting Started Guide*, GC27-8806-01
- *DB2 QMF Data Service Customization Guide*, GC27-8807-01
- *DB2 QMF Data Service Solutions Guide*, GC27-8808-01
- *DB2 QMF Data Service Studio User's Guide*, SC27-8816-01

After the installation of QDS and the configuration of the supporting data sets and JCL members is complete, you must take two remaining steps for each data source. You must configure access to that source, and you must go through a process, which is called *mapping*, to create the virtual tables. In mapping, you select the subsets of tables, views, and files from the data source to be available to users through QDS.

The term *subset* indicates that not all tables, views, and files are mapped and that of those tables, views, and files that are mapped, not all of the columns are necessarily included. Again, this description might sound similar to the properties of a view in a relational database, and for an introduction, that is not a bad way of looking at a virtual table and mapping. In fact, QDS can create its own view objects so the rows that are exposed can also be limited.

Configuring access is quick. It involves editing the server initialization member, a REXX program, and editing the JCL member that is run for the started task. In the server initialization member, you add the connection information for the data source and set a flag to enable the connection. Collect all of the connection information for your sources ahead of time. With all of the connection information that is gathered in advance, this process might take 5 - 10 minutes for each data source. If you use DB2, an additional requirement exists to bind packages at DB2.

Creating a virtual table is a quick process. A GUI desktop wizard creates a virtual table in 1- 2 minutes. However, you must map the tables, views, and files one at a time. (SMF and system log are the exceptions, and they are mapped in bulk automatically.) The time that you need depends on how many tables, views, and files exist. The following pages guide you through the process of mapping. Go through the process a few times to gain familiarity with the process. Then, plan how much time your deployment project requires.

2.2.2 Configuring access to DB2

You must configure three files to set up access to DB2:

- The server initialization member REXX program CQDSIN00 from the SCQDEXEC data set contains the configuration information for connecting to DB2 and the other data sources.
- The CQD1PROC from the SCQDCNTL data set is launched as the started task. It requires references to the DB2 runtime libraries.
- The CQDBINDD job in the SCQDCNTL data set is used to perform the package binds at each DB2 location that is required by QDS. Unlike QMF for Workstation and QMF for WebSphere, these binds are a one-time action and typically need to be rebound only when DB2 is upgraded from one version to another version.

Editing CQDSIN00

Each type of data source has a section of the program with a flag at the top that is set to either `DoThis` or `DontDoThis`. The default is `DontDoThis`, which means that the configuration for this data source is disabled and ignored.

The section from CQDSIN00 for DB2 is shown in Example 2-1. It contains a preconfigured set of parameters with a default setting and a description of the parameter's role. Unless a specific need exists, leave the default values. The values in Example 2-1 are from a configured system. So, the flag is set to `DoThis`.

Example 2-1 CQDSIN00 for DB2

```
/* Enable DRDA access to DB2 database subsystems                   */
 /*-----------------------------------------------------------------*/
 if DoThis then do
MODIFY PARM NAME(TRACEOEDRDARW)       VALUE(YES)"
/*-------------------------------------------------------------*/
/* To require individual clients to opt in to DRDA usage, set    */
/* the following parameter to YES.  Opt-in processing requires  */
/* that the client host user parm (USERPARM) is set to "DRDA".  */
```

```
/*--------------------------------------------------------------*/
MODIFY PARM NAME(CLIENTMUSTELECTDRDA)  VALUE(NO)"
/*--------------------------------------------------------------*/
/* Set the following parameter to YES to bypass sending WLM-     */
/* related information to DRDA when each connection is          */
/* first established.  The WLM information includes client name, */
/* workstation name, application name, and accounting info.      */
/*--------------------------------------------------------------*/
MODIFY PARM NAME(DRDASKIPWLMSETUP)     VALUE(NO)"
/* Set the following parameter to YES to use DRDA access         */
/* for CQD's logging task.  If you enable DB2 logging via       */
/* DRDA you must configure and specify passticket values for     */
/* each DRDA endpoint by including the APPLNAME keyword.         */
/*--------------------------------------------------------------*/
"MODIFY PARM NAME(DRDAFORLOGGINGTASK)   VALUE(NO)"
/* The following parameter sets the package name prefix used for */
/* DRDA package selection.                                       */
/* RS - is the normal DRDA package prefix                        */
/*--------------------------------------------------------------*/
"MODIFY PARM NAME(DRDAPACKAGEPREFIX)    VALUE(DS)"
/*--------------------------------------------------------------*/
/* The following parameter sets the DRDA package collection      */
/* name.  A 1-to-18 character value should be used.  The         */
/* default value is "NULLID".  If you change the name here you   */
/* must also rebind the DRDA packages with the alternate         */
/* collection name (see jobs BINDDDF and BINDDXO in the Server   */
/* CNTL data set).                                               */
/*--------------------------------------------------------------*/
"MODIFY PARM NAME(DRDACOLLECTIONID)     VALUE(NULLID)"
/* Each DRDA endpoint must be defined using DEFINE DATABASE     */
 /* statements.  Repeat the DEFINE DATABASE statement for each    */
 /* DB2 subsystem for which DRDA processing is to be enabled.    */
```

Example 2-2 shows the key parameters that identify the connection to the DB2 system. The connection parameters of location, port number, and IP address are used in the same way for many clients of DB2, such as QMF for Workstation.

Example 2-2 Connect to the DB2 system

```
/*--------------------------------------------------------------*/
"DEFINE DATABASE TYPE(MEMBER)"                   ,
                "NAME(SA1A)"                     ,
                "LOCATION(RS50SA1A)"             ,
                "DDFSTATUS(ENABLE)"              ,
                "PORT(3900)"                      ,
                "IPADDR(127.0.0.1)"              ,
                "CCSID(37)"
  /*            "APPLNAME(DSN1LU)"    */
  /*            "IDLETIME(110)"
  /*--------------------------------------------------------------*/
  /* If DB2 ZPARM parameter IDTHTOIN is set to a non-zero value    */
  /* then add comma after APPLNAME parm, uncomment and set         */
  /* IDLETIME to a value slightly lower (10 secs.) than IDTHTOIN.  */
  /* this will also allow product DRDA threads to become inactive. */
 /*--------------------------------------------------------------*/
```

In addition to the standard IP address, port, and location, you must specify a TYPE and a NAME. The name is the four-character subsystem identifier (SSID). The TYPE is from the following list:

- `GROUP` if the distributed data facility (DDF) endpoint is a DB2 group director
- `MEMBER` if the DDF endpoint is a DB2 group member for z/OS
- `LUW` if the DDF endpoint is a DB2 instance or group member for Linux, UNIX, or Microsoft Windows (LUW)

Because QDS is a Distributed Relational Database Architecture (DRDA) application requestor, it can be configured to connect to all of your DB2 for z/OS and DB2 i, and DB2 LUW instances if network connectivity exists. If you are configuring for a DB2 for z/OS subsystem on the same logical partition (LPAR) as QDS, use the “loopback” address of `127.0.0.1` for the IP address.

To configure more than one DB2 subsystem, repeat the block of values for the first DB2 subsystem and then modify the NAME, LOCATION, and so on, as shown in Example 2-3.

Example 2-3 Configuring more than one DB2 subsystem

```
"DEFINE DATABASE TYPE(MEMBER)"                    ,
                   "NAME(SA1A)"                     ,
                   "LOCATION(RS50SA1A)"             ,
                   "DDFSTATUS(ENABLE)"              ,
                   "PORT(3900)"                     ,
                   "IPADDR(127.0.0.1)"              ,
                   "CCSID(37)"
   /*              "APPLNAME(DSN1LU)" */
  /*               "IDLETIME(110)"                               */
   "DEFINE DATABASE TYPE(MEMBER)"                   ,
                   "NAME(RA1A)"                     ,
                   "LOCATION(RS52RA1A)"             ,
                   "DDFSTATUS(ENABLE)"              ,
                   "PORT(3895)"                     ,
                   "IPADDR(rs52.mycomp.com)"              ,
                   "CCSID(37)"
   /*              "APPLNAME(DSN1LU)" */
  /*               "IDLETIME(110)"                               */   .
```

Note: The IPADDR for the second entry is not `127.0.0.1`. This subsystem is at a remote LPAR. With the connection information in place, the next step is to bind the packages for QDS at each of the DB2 locations.

Editing CQDBINDD

The CQDBINDD is shipped in the SCQDCNTL data set. It contains the commands to run the binds from the SCQDDBRM member. It contains instructions about the edits that are needed (Example 2-4).

Example 2-4 CQDBINDD execution instructions

```
//*      EXECUTING THIS JOB:
//*
//*      0) SPECIFY VALID JOBCARD
//*         SEE "== 0 ==>" ABOVE.
//*
//*      1) CHANGE HLQ TO THE HIGH LEVEL QUALIFIER FOR CQD
//*
//*      2) CHANGE DSN!!! TO THE DB2 HIGH LEVEL QUALIFIER
//*
//*      3) CHANGE DSN? TO THE DB2 SUBSYSTEM WHERE YOU WISH TO
//*         BIND THE PLANS.
//*
//*      4) REVIEW PLAN(DSNTIAD) IN STEP GRNT0001 TO ENSURE IT POINTS
//*         TO THE CORRECT DB2 VERSION OF THE DSNTIAD PLANNAME.
//*
//*      5) IF YOU DESIRE, CHANGE THE VALUE "NULLID" TO ANY OTHER
//*         1-TO-18 CHARACTER COLLECTION NAME.  YOU MUST ALSO UPDATE
//*         THE SERVER STARTUP PARAMETER DRDACOLLECTIONID.
```

This last comment, number 5, about the NULLID collection cross-references the CQDSIN00 setting for `"MODIFY PARM NAME(DRDACOLLECTIONID)     VALUE(NULLID)"`.

Also, back in CQDSIN00, the following PARM exists:

```
"MODIFY PARM NAME(DRDAPACKAGEPREFIX)    VALUE(DS)"
```

If your internal naming standards call for a different package prefix or collection name and you change it in CQDSIN00, you must make the corresponding changes in the actual package names and collection that are called out in CQDBINDD.

For example, two pieces from CQDBINDD show the default DS prefix and the NULLID collection (Example 2-5 and Example 2-6).

Example 2-5 CQDBINDD part 1

```
BROWSE    DVS.CQD.SV110201.SCQDCNTL(CQDBINDD)
 Command ===>
   SELECT MEMBER=((CQDR510A,DSOR510A))
   COPY INDD=DBRMLIB,OUTDD=DBRMTMP
   SELECT MEMBER=((CQDR510B,DSOR510B))
   COPY INDD=DBRMLIB,OUTDD=DBRMTMP
   SELECT MEMBER=((CQDR510C,DSOR510C))
```

Example 2-6 CQDBINDD part 2

```
FREE PACKAGE(NULLID.DSOC510A.(*))
FREE PACKAGE(NULLID.DSOC510B.(*))
FREE PACKAGE(NULLID.DSOC510C.(*))
```

Note: These FREE statements occur *before* the QDS BIND statements. So, if this QDS BIND is your first QDS BIND, you get a high MAX CC in the feedback for the job. Check the job status, but the message likely shows that the FREE step is the cause and that no actual problem exists.

More inspection of CQDBINDD shows that although with CQDSIN00, you specified connection information for a remote DB2 subsystem, the CQDBINDD job refers only to a system on the local LPAR (comment number 3 in Example 2-4 on page 13). To perform the binds at remote DB2 subsystems (different LPARs), you do not need to install the entire QDS application at each LPAR. You can copy CQDBINDD and the SCQDDBRM data set, which contains the members that are referenced in CQDBINDD, to the other LPARs and edit CQDBINDD and submit it for each DB2 subsystem that is local to the specific LPAR.

Editing CQD1PROC

You need to edit CQD1PROC to establish the DB2LIB parameter, which holds the value of the DB2 library that contains the DB2 interface modules, such as DSNALI and DSNHLI.

Locate this line in CQD1PROC:

```
DB2LIB='DSNX10'
```

Uncomment this line and change it to your site-specific value. It must be all uppercase characters, and it must be in single quotation marks.

The configuration of IMS connectivity is similar, but it does not require the binds. So, it consists of editing the CQDSIN00 and the CQD1PROC only.

The configuration of VSAM connectivity is even simpler. It does not require any changes to these data sets. The connection is handled during the mapping process that creates the virtual table.

Before you continue to configure other data sources, look at a case study that shows why setting up QDS can be an improvement over traditional QMF workflow and the workflow of other applications.

A DB2 use case

This case is based on a known client’s use of QMF for Workstation WebSphere.

An organization has 20 DB2 for z/OS subsystems: DB2A, DB2B, DB2C, … DB2T. Each DB2 subsystem has its own copy of a particular table, which is called FINANCE.COSTS, but different data is in each subsystem. A user needs to generate a report or spreadsheet that contains the data from all 20 versions of the table from the different subsystems that are concatenated.

Currently, this user runs a QMF PROC, as shown in Example 2-7.

Example 2-7 Current QMF PROC

```
RUN QUERY GETDATA
CONNECT TO DB2A
SAVE DATA AS FINANCE.SUMMARY(ACTION = REPLACE
CONNECT TO DB2B
RUN QUERY
CONNECT TO DB2A
SAVE DATA AS FINANCE.SUMMARY(ACTION = APPEND
CONNECT TO DB2C
```

```
RUN QUERY
CONNECT TO DB2A
SAVE DATA AS FINANCE.SUMMARY(ACTION = APPEND
Etc
CONNECT TO DB2T
RUN QUERY
CONNECT TO DB2A
SAVE DATA AS FINANCE.SUMMARY(ACTION = APPEND
DISPLAY FINANCE.SUMMARY
EXPORT DATA TO C:\DOCS\FINANCE.XLSX (DATAFORMAT=XLSX
```

This procedure connects to each subsystem, queries the table, reconnects to the first subsystem, and inserts the results into the same temporary table. This temporary table is queried and the data is downloaded to a spreadsheet.

QDS virtualizes the 20 versions of the table into a single relational database, and the PROC is similar to this example:

```
RUN QUERY GETDATA
EXPORT DATA TO C:\DOCS\FINANCE.XLSX (DATAFORMAT=XLSX
```

The query GETDATA now contains 20 SELECT statements that are concatenated with UNION ALL clauses.

This approach simplifies the logic and reduces the I/O during the SAVE DATA. QDS can retrieve the data in parallel while the traditional QMF PROC performed the work in a series. With QMF for Workstation or QMF for WebSphere, the connection is a DRDA connection and so up to 60% of the DB2 workload (retrieving the data and performing the INSERTS) is zIIP eligible. With QDS, the connection to DB2 is still DRDA and the DB2 workload of retrieving the data is the same. But the work of concatenating the data is shifted to QDS, and the work that is performed in QDS is up to 99% eligible. DB2 does not create a temp table and import the data into it.

So, the parallel processing provides a better response time for QMF. The workload shift to QDS provides a higher portion of zIIP offload. And, the DB2 workload at DB2A, where all of the SAVE DATAs were taking place, is eliminated.

Although this example is focused on DB2, the query in the PROC that is shown can combine IMS and DB2 and VSAM, which is a workflow that QMF was unable to perform before the QDS technology.

2.2.3 Configuring access to IMS data

The back-end configuration for IMS consists of editing CQDSIN00 and CQD1PROC.

Editing CQDSIN00

Each type of data source has a section of the program with a flag at the top that is set to either `DoThis` or `DontDoThis`. The default is `DontDoThis`, which means that the configuration for this data source is disabled and ignored.

Locate the IMS section and perform these steps:

1. Change the `DontDoThis` to `DoThis`.
2. Enter the IMSID.
3. Enter the IMSDSNAME.

The result is shown in Example 2-8.

Example 2-8 Editing CQDSIN00

```
/* Enable IMS CCTL/DBCTL support                                 */
 /*-----------------------------------------------------------------*/
 if DoThis then
   do
    "MODIFY PARM NAME(DBCTL)              VALUE(YES)"
    "MODIFY PARM NAME(IMSID)              VALUE(IXA1)"
    "MODIFY PARM NAME(IMSDSNAME)          VALUE(IMS.IXA1.SDFSRESL)"
    "MODIFY PARM NAME(IMSMINTHREADS)      VALUE(5)"
    "MODIFY PARM NAME(IMSMAXTHREADS)      VALUE(10)"
```

Editing CQD1PROC

You must perform a minor edit in CQD1PROC to establish the IMSLIB, which contains the IMS RESLIB/SDFSRESL data sets.

Follow these steps:

1. Locate this line in CQD1PROC:

   ```
   IMSLIB='IMS.RESLIB'
   ```

2. Uncomment this line and change it to your site-specific value. It must be all uppercase characters and it must be in single quotation marks, as shown in Example 2-9.

Example 2-9 Editing CQD1PROC

```
/*********************************************************************
/*      THE FOLLOWING LINE MUST BE ADDED TO THE PROC STATEMENT       *
/*      FOR IMS INTERFACE SUPPORT AND UNCOMMENTED ON THE STEPLIB     *
/*      STATEMENT.                                                   *
/*                                                                   *
/           IMSLIB='IMS.IXA1.SDSFRESL'                                     *
/*********************************************************************
```

3. Also, if you activated DB2LIB above this entry, go back and place a comma after it. If you previously activated any of the data sources after IMS, follow this IMSLIB entry with a comma.
4. Then, on the STEPLIB, uncomment the line for &IMSLIB (Example 2-10).

Example 2-10 STEPLIB

```
//STEPLIB      DD   DISP=SHR,DSN=&HLQ..SCQDLOAD
//             DD   DISP=SHR,DSN=&DB2LIB..SDSNEXIT
//             DD   DISP=SHR,DSN=&DB2LIB..SDSNLOAD
//*            DD   DISP=SHR,DSN=&ADALOAD
//             DD   DISP=SHR,DSN=&IMSLIB
```

5. Restart the CQD1PROC started task.

The rest of the work for the IMS setup occurs during the virtual table creation. To create the virtual tables, you need the location of the source files for the IMS database descriptions (DBDs) and program specification blocks (PSBs).

2.2.4 Configuring access to VSAM data

Unlike DB2 and IMS, VSAM does not have management software that gives access to the data and handles communications. No configuration settings in the QDS data sets are needed to enable access to VSAM data. However, each VSAM file that QDS connects to needs an associated copybook to tell QDS about the lengths of fields and data types. So, the only preparation is to make the copybooks available.

2.3 Using QMF Data Service Studio to create virtual tables

QMF Data Service Studio is an Eclipse desktop application for Windows. It is used to perform administrative tasks on a running QDS server. Because QMF Data Service Studio is an Eclipse application, it adopts certain common Eclipse terminologies, such as views, perspectives, and editors. The Help contains tutorials about this terminology.

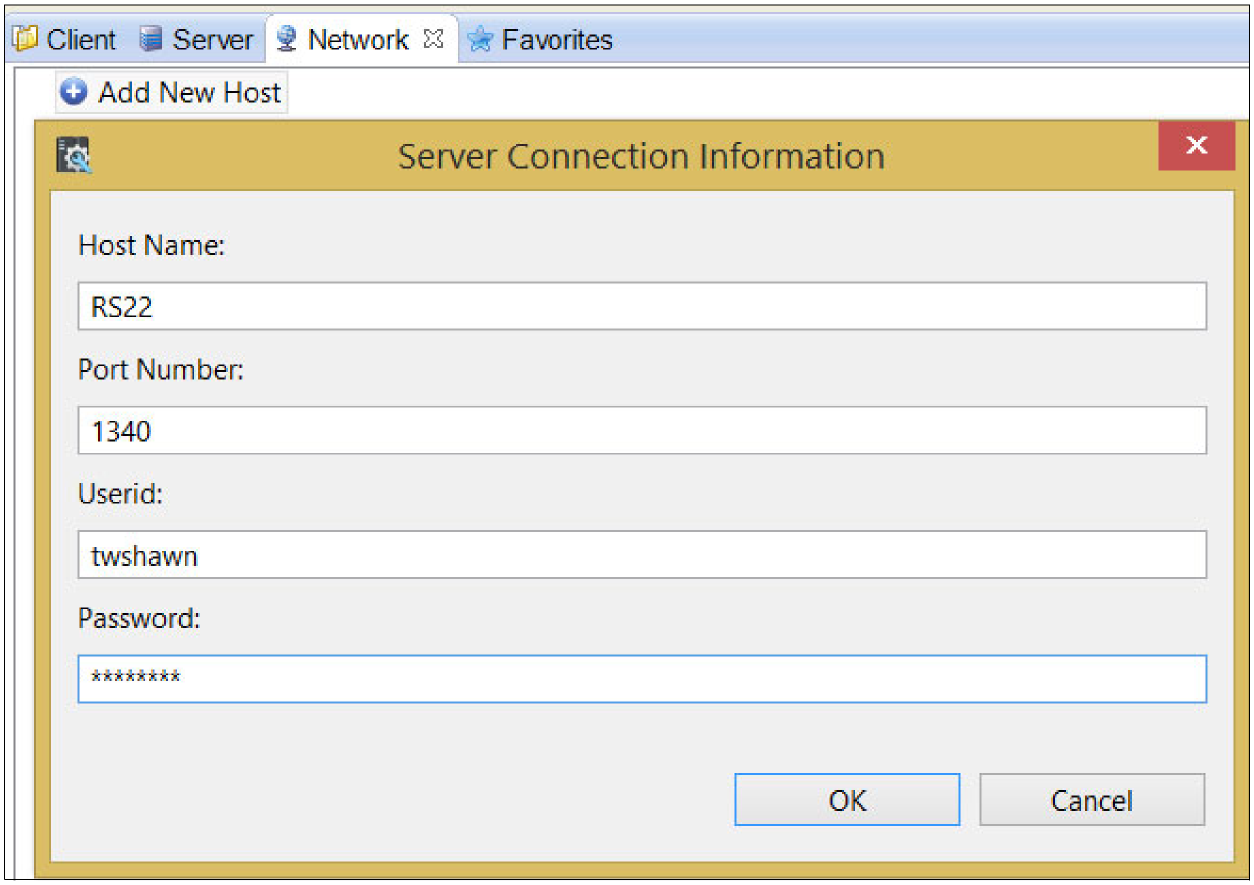
These terms are explained:

- A *view* is a tabbed window with an explorer tree for drilling into settings, viewing them, and manipulating them through the right-click menu.
- An *editor* is a tabbed window where you edit or manipulate text and objects. The editor is often launched from a view right-click menu.
- A *perspective* is a collection of views, editors, menus, and buttons that are grouped because they help with a particular workflow.

2.3.1 Initial setup

When the QMF Data Service Studio is started for the first time, you need to configure a connection to the mainframe systems where QDS is running:

1. Select **Window** → **Open Perspective** → **Data Service**.
2. In the central work area, click **Network**, as shown in Figure 2-2 on page 18.
3. Double-click the **Add New Host** icon and fill out the connection information, as shown in Figure 2-2 on page 18. The Port number is OEPORTNUMBER from the CQD1IN00 program.



Host Name: RS22

Port Number: 1340

Userid: twshawn

Password: ********

Figure 2-2   Server Connection Information

4. The server is in place, as shown in Figure 2-3. Repeat this process for each LPAR where QDS is running.

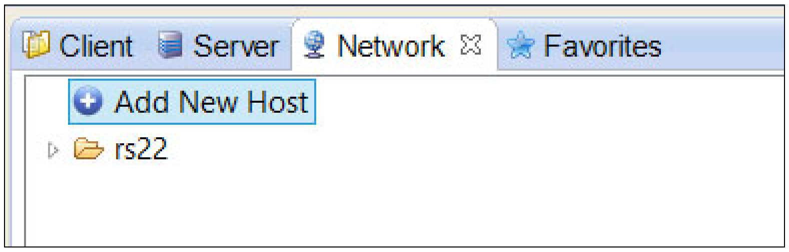
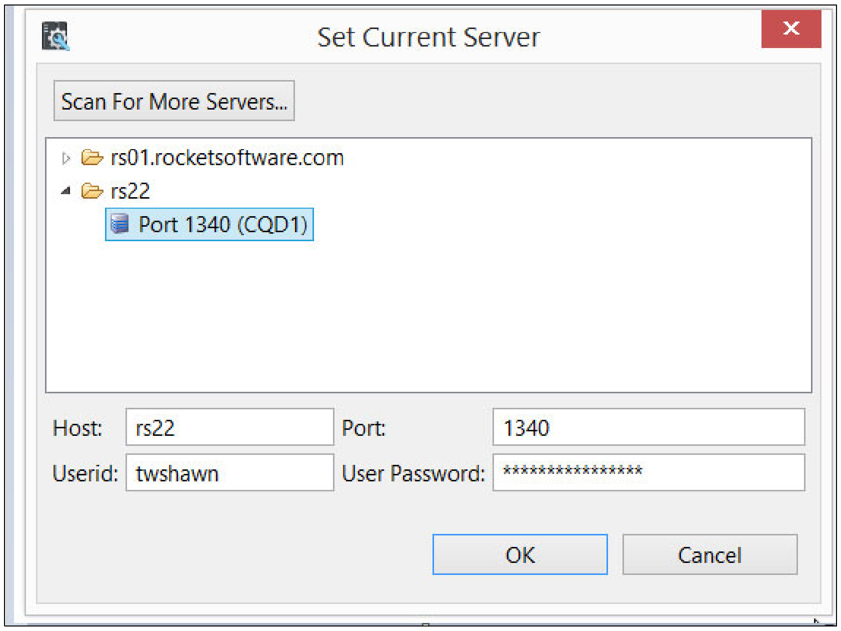


Figure 2-3   First server in place

5. Switch to **Server** in the central work area, and select **Set Server**. Choose one of the servers that you put in place on the Network tab, as shown in Figure 2-4. Click **OK**.



Host: rs22
Port: 1340
Userid: twshawn
User Password: ****************

Figure 2-4 Set Current Server

Note: The CQD1 next to Port 1340 in Figure 2-4 is the SSID parameter from the CQD1IN00 program.

6. You are ready to create virtual tables for DB2, IMS, and so on. Expand the tree structure, as shown in Figure 2-5. Right-click **CQD1** and select **Create Virtual Table**.

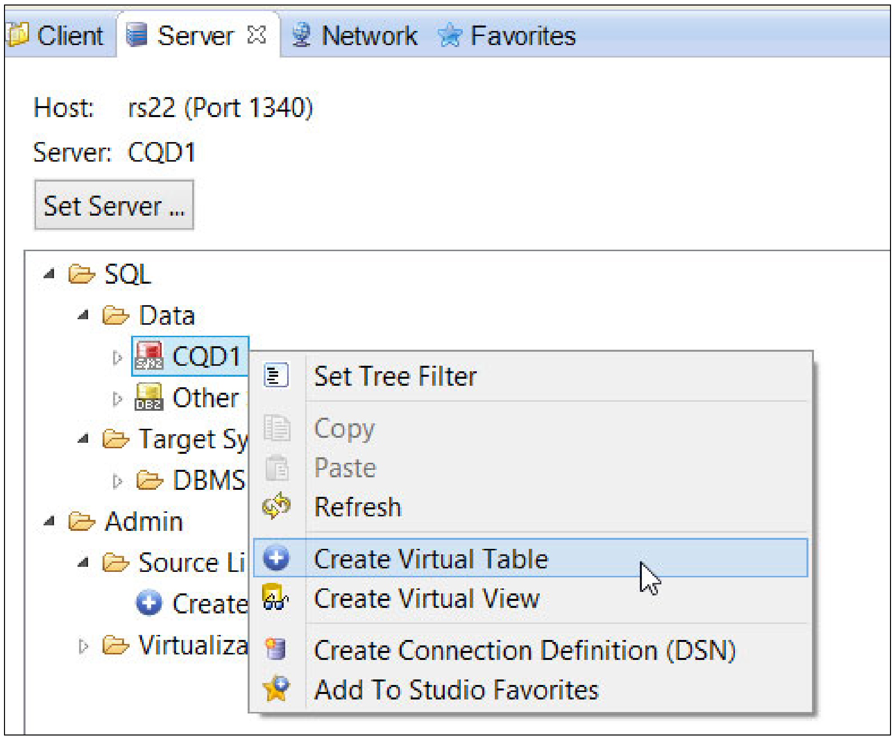


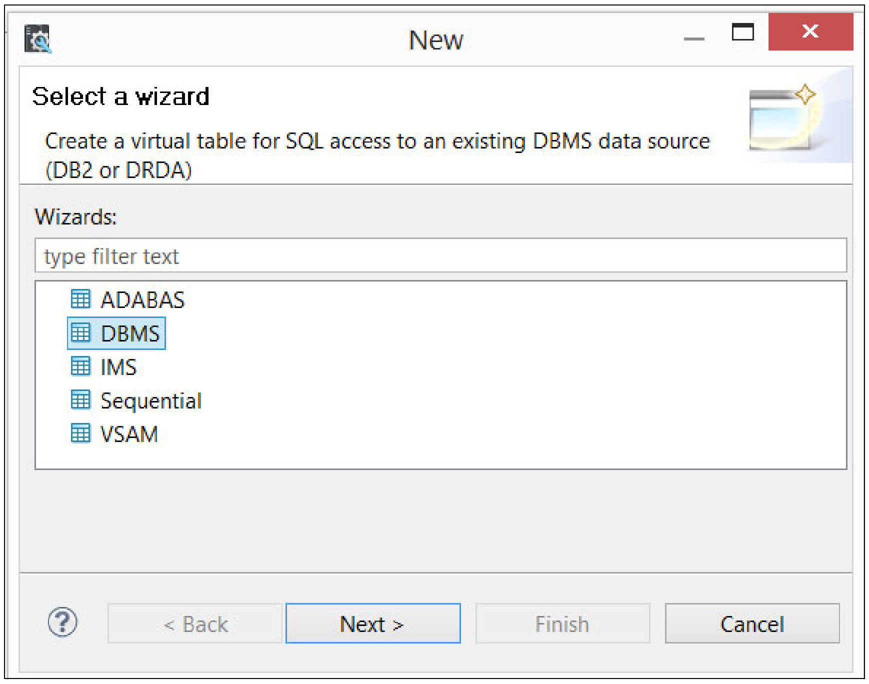
Figure 2-5 Explorer tree

Note: A Create Virtual View option exists. As with a DB2 view, you can create a view that is a subset of the columns and rows in an existing virtual table.

2.3.2 Virtualize the DB2 data

Perform the following steps to virtualize the DB2 data:

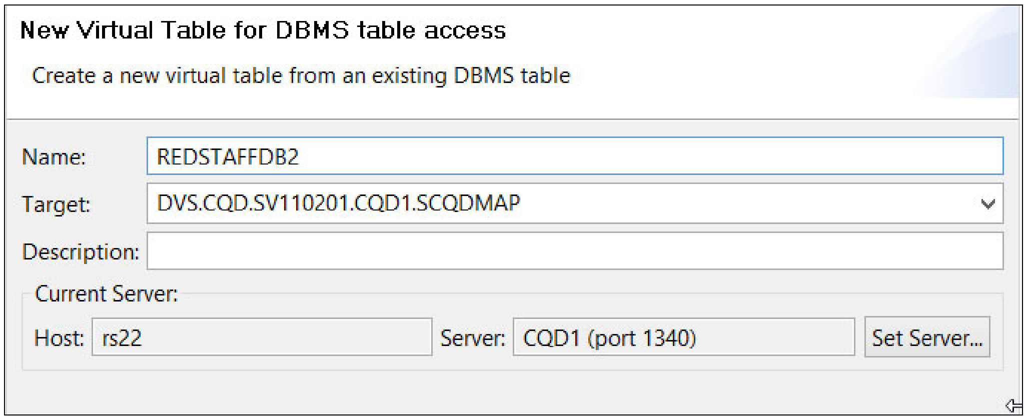
1. Right-click **CQD1** under Data, as shown in Figure 2-5 on page 20, and select **Create Virtual Table**.
2. DB2 is a database management system (DBMS), so select **DBMS**, as shown in Figure 2-6.



Create a virtual table for SQL access to an existing DBMS data source (DB2 or DRDA)

Figure 2-6 Select DBMS for the relational database management system

3. For the Name field in Figure 2-7, enter the name that the users see for the table and use in their SQL statements. No schema name shows for the virtual table.



New Virtual Table for DBMS table access

Create a new virtual table from an existing DBMS table

Name: REDSTAFFDB2

Target: DVS.CQD.SV110201.CQD1.SCQDMAP

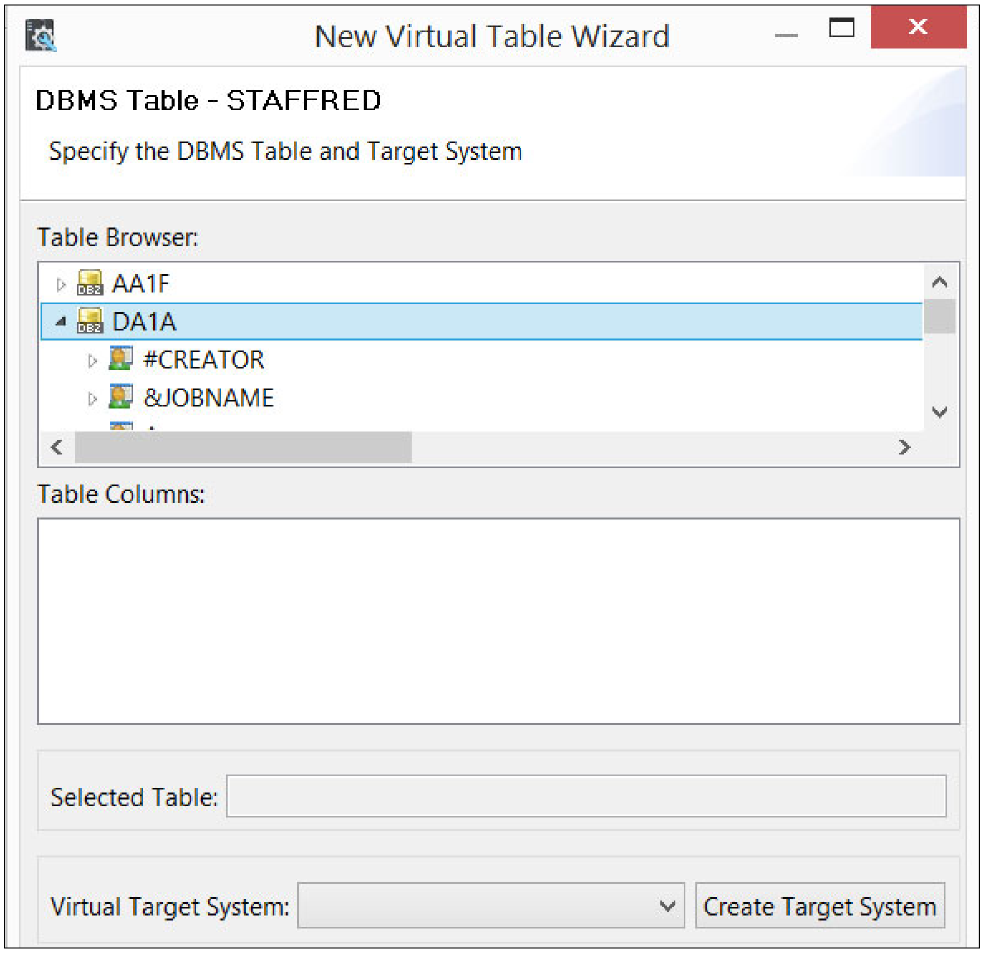
Description:

Current Server:

Host: rs22 Server: CQD1 (port 1340) Set Server...

Figure 2-7 Enter the name of the virtual table

4. Select the subsystem, as shown in Figure 2-8, that you configured. (QDS discovers all of the subsystems, whether they are configured for QDS or not.)



DBMS Table - STAFFRED

Specify the DBMS Table and Target System

Table Browser:

- AA1F
- DA1A
  - #CREATOR
  - &JOBNAME

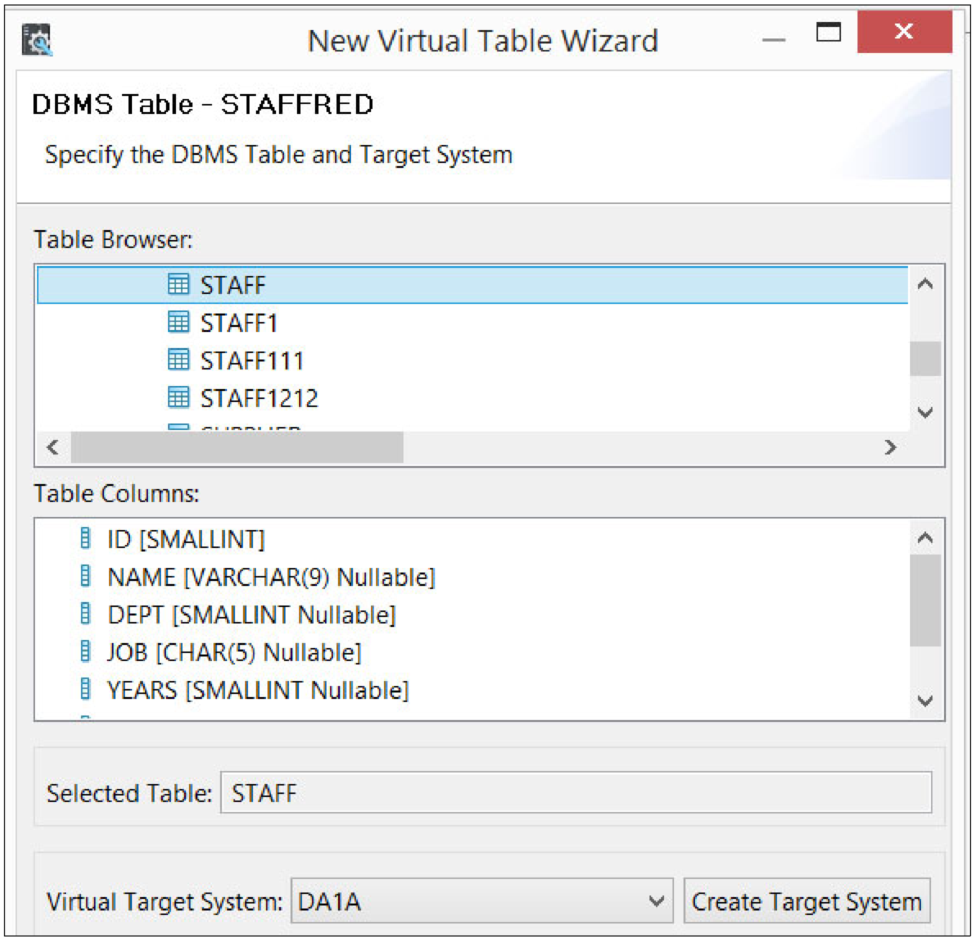
Table Columns:

Selected Table:

Virtual Target System:

Figure 2-8 Select the DB2 subsystem

5. Scroll down to the schema name and select the table to virtualize, as shown in Figure 2-9.


New Virtual Table Wizard

DBMS Table - STAFFRED

Specify the DBMS Table and Target System

Table Browser:

STAFF
STAFF1
STAFF111
STAFF1212

Table Columns:

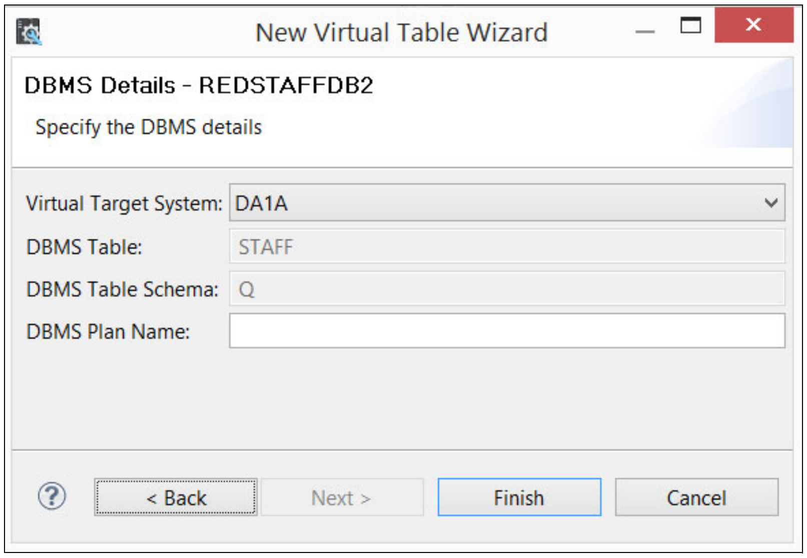
ID [SMALLINT]
NAME [VARCHAR(9) Nullable]
DEPT [SMALLINT Nullable]
JOB [CHAR(5) Nullable]
YEARS [SMALLINT Nullable]

Selected Table: STAFF

Virtual Target System: DA1A

Figure 2-9 The source table and its columns

6. Click **Finish**. The virtual table is ready for QMF for Workstation and QMF for WebSphere to query it, as shown in Figure 2-10.


New Virtual Table Wizard

DBMS Details - REDSTAFFDB2

Specify the DBMS details

Virtual Target System: DA1A
DBMS Table: STAFF
DBMS Table Schema: Q
DBMS Plan Name:

Figure 2-10 Click Finish

2.3.3 Virtualize IMS data

Before you create a virtual table for IMS, you must create a source library under the Admin branch. Setup steps are required for IMS during the creation process for the first virtual table. Follow these steps:

1. Drill down into the Admin branch to the Source Libraries. Double-click the plus sign (**+**) icon for **Create Virtual Source Library**, as shown in Figure 2-11.

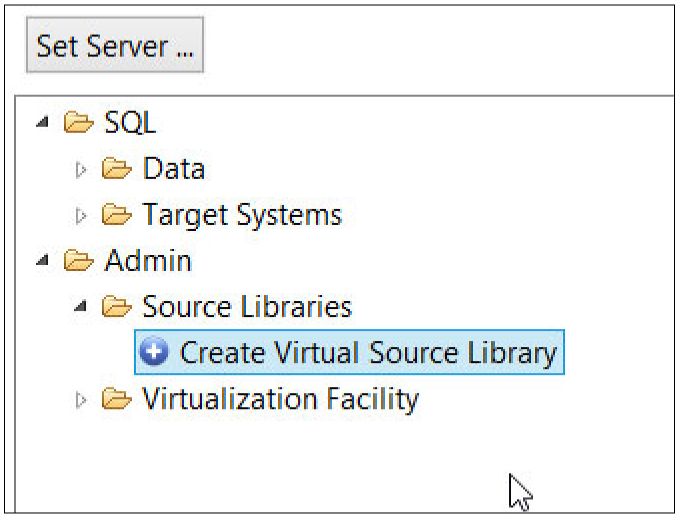
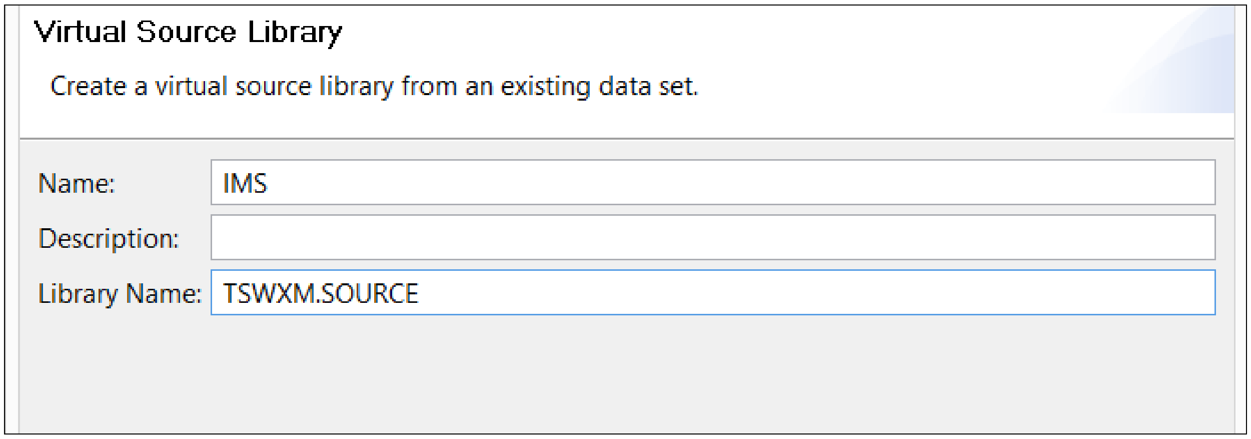


Figure 2-11 Adding a source Library for IMS

2. Enter a name for the library. Enter the IMS data set that contains the relevant DBDs, PSBs, and source copybooks (Figure 2-12).



Virtual Source Library

Create a virtual source library from an existing data set.

Name:	IMS
Description:	
Library Name:	TSWXM.SOURCE

Figure 2-12 Enter the library information

3. Create a virtual table for IMS, which starts in the same way as for DB2 and all other sources, as shown in Figure 2-13.

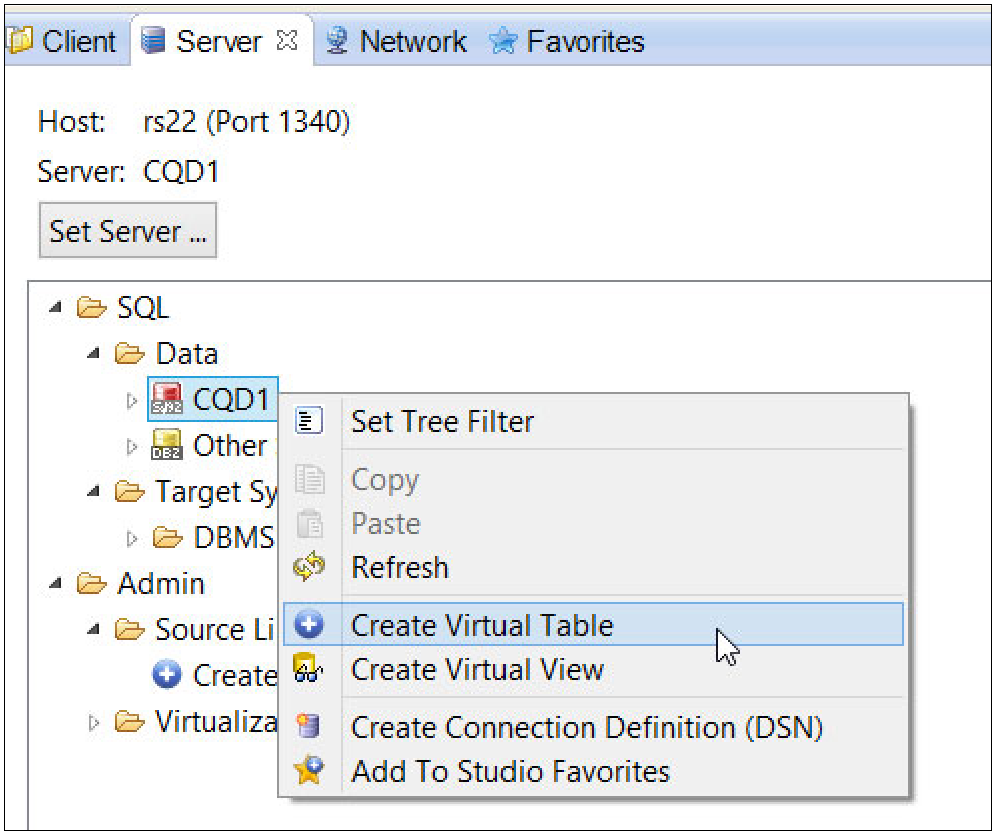


Figure 2-13 Create an IMS virtual table

4. Select **IMS** from the list of data sources, as shown in Figure 2-14, and click **Next**.

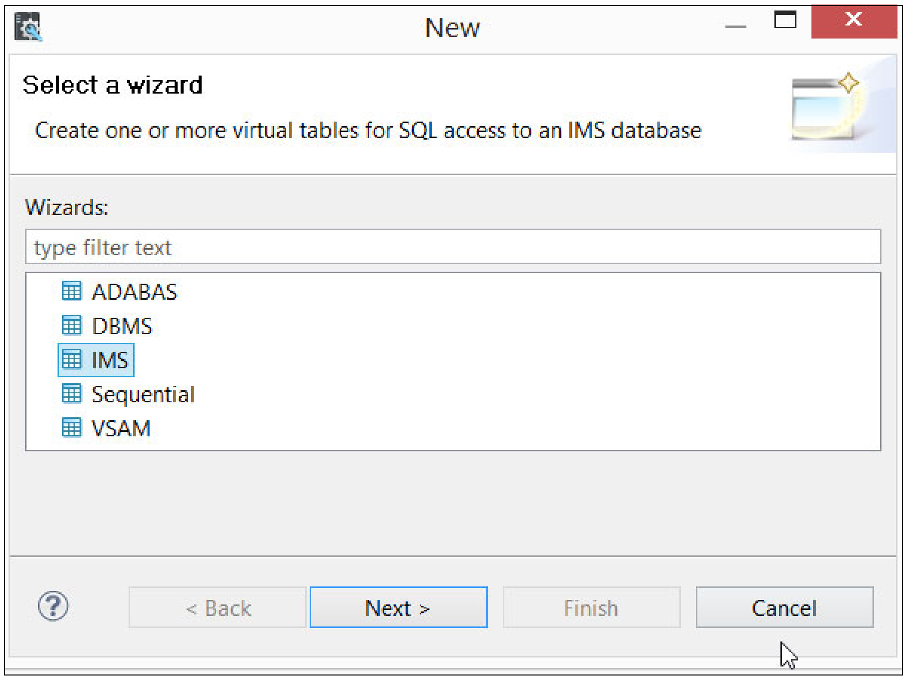


Figure 2-14 Selecting IMS

5. Extract the DBD and the PSB source definitions and copybook, as shown in Figure 2-15.

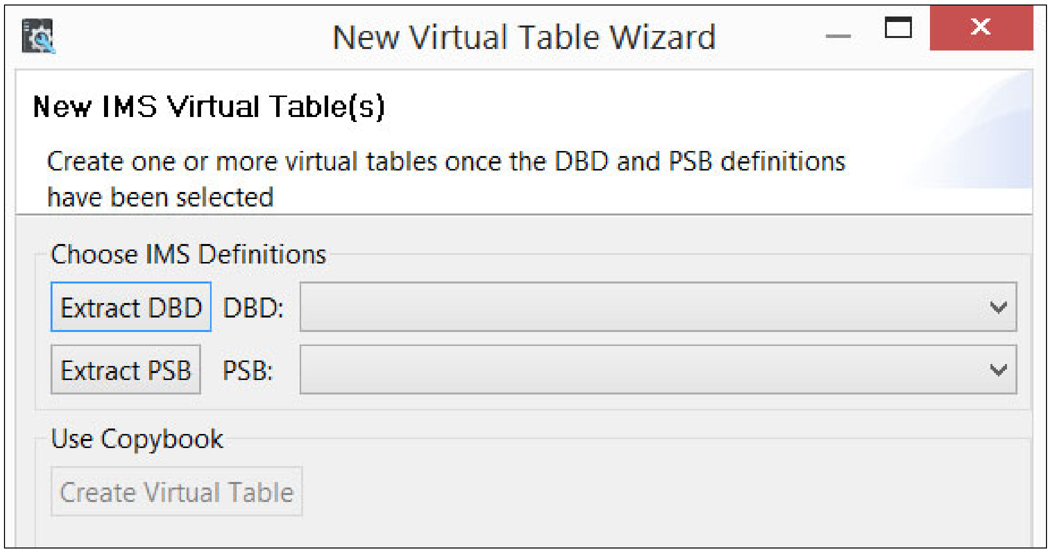
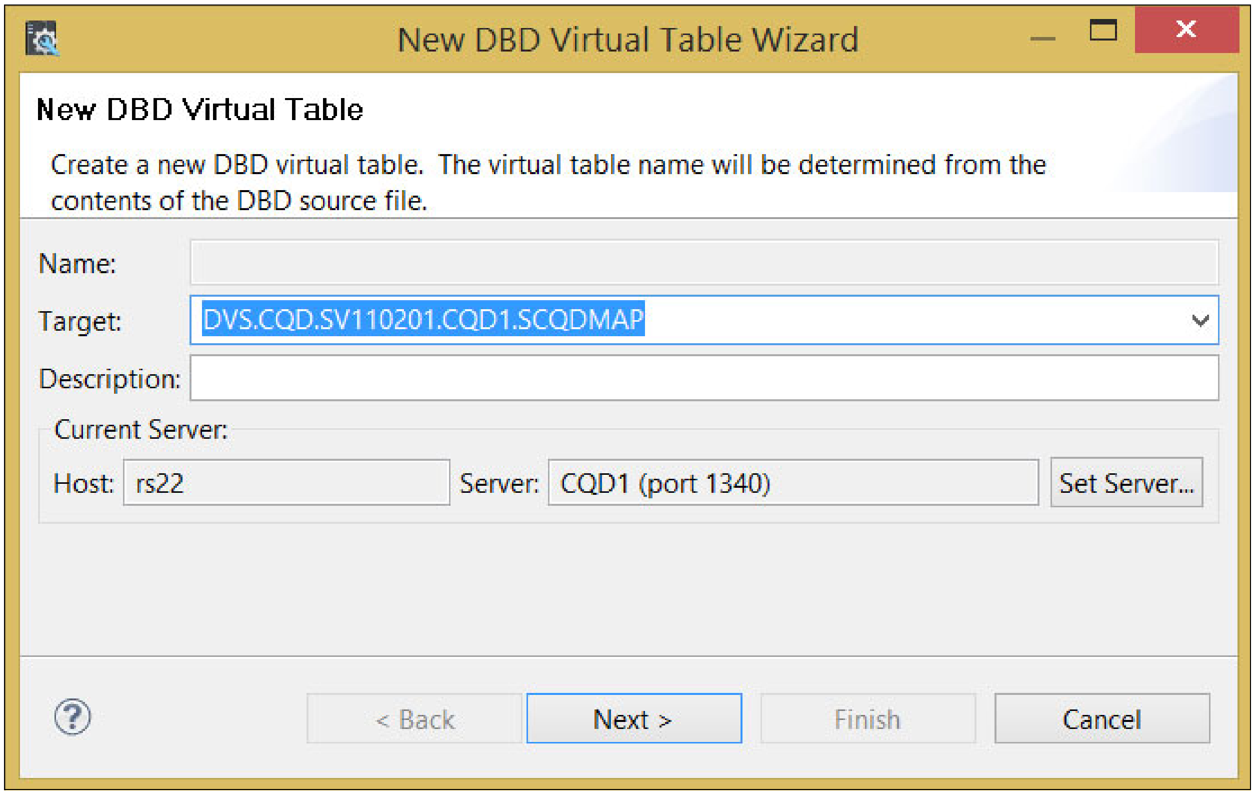


Figure 2-15 Extract DBD, PSB, and copybook

6. This multistep wizard uses these copybook definitions and file locations to create the virtual table definition. When QDS receives an SQL query for this virtual table, it can locate the IMS data and format it as a relational result set. Click **Extract DBD**, as shown in Figure 2-15.

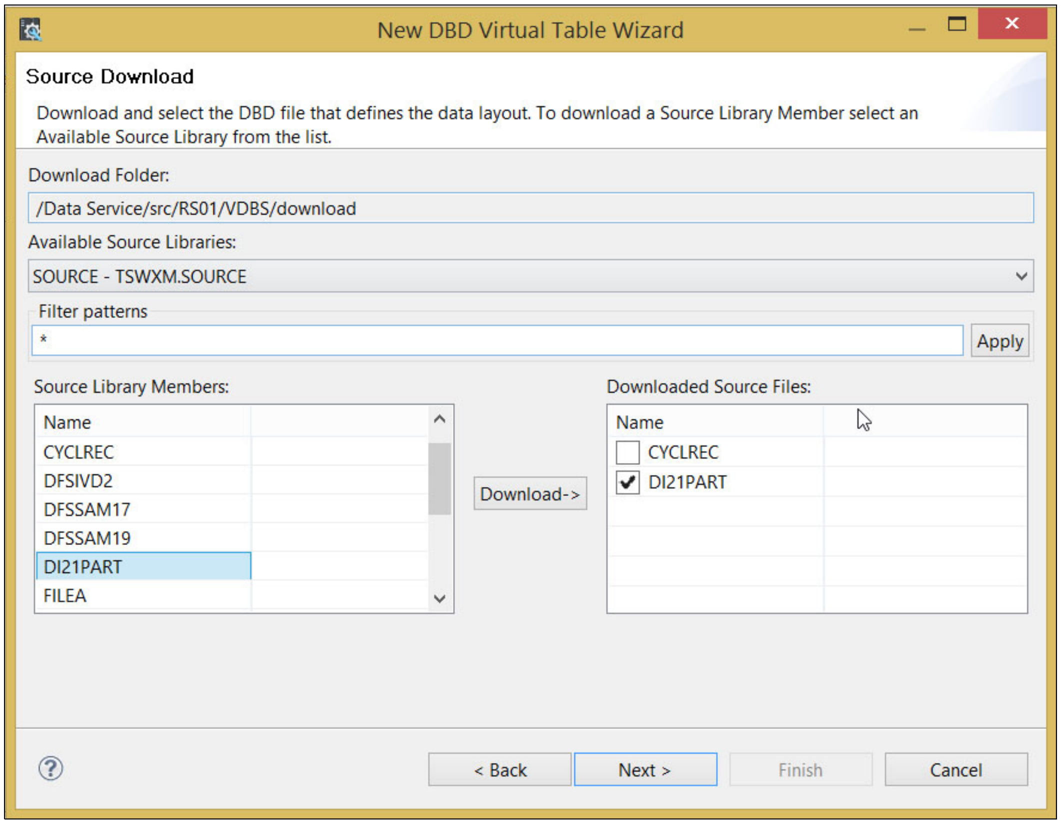
7. The wizard that is shown in Figure 2-16 shows the data set on the mainframe where the mapping information is sent. Click **Next**.


New DBD Virtual Table Wizard
New DBD Virtual Table
Create a new DBD virtual table. The virtual table name will be determined from the contents of the DBD source file.
Current Server:
Set Server...
< Back
Next >
Finish
Cancel

Name:
Target: DVS.CQD.SV110201.CQD1.SCQDMAP
Description:
Host: rs22
Server: CQD1 (port 1340)

Figure 2-16 First step of the DBD extraction

8. Choose the DBD from the source library, as shown in Figure 2-17, and click **Next**.



Download and select the DBD file that defines the data layout. To download a Source Library Member select an Available Source Library from the list.

Download Folder: /Data Service/src/RS01/VDBS/download

Available Source Libraries: SOURCE - TSWXM.SOURCE

Filter patterns: *

Source Library Members:

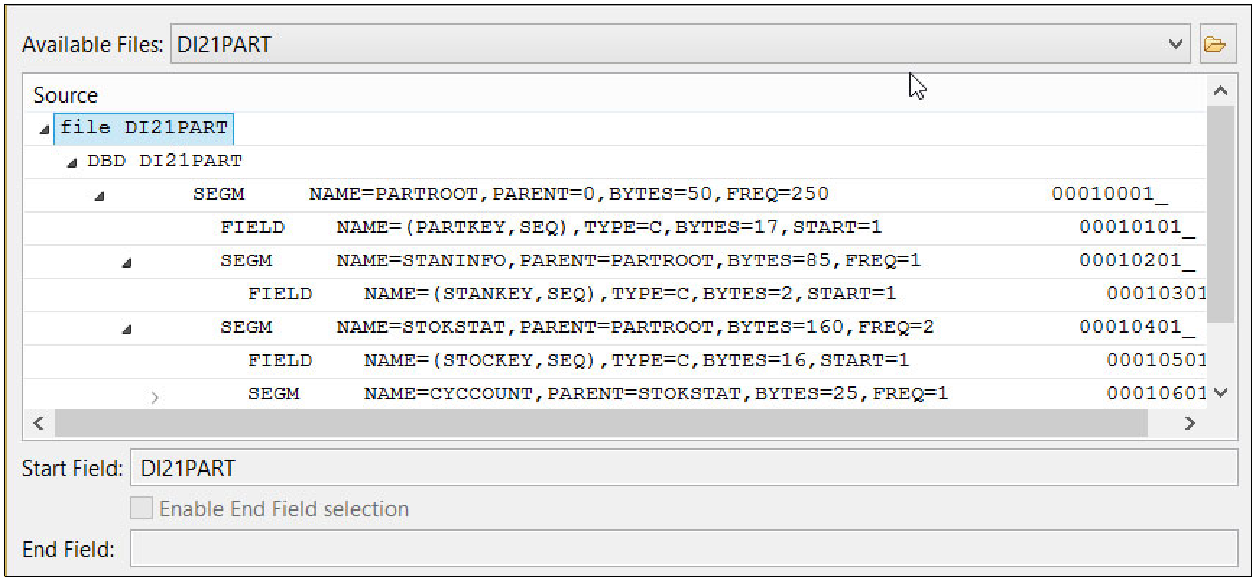
Name
CYCLREC
DFSIVD2
DFSSAM17
DFSSAM19
DI21PART
FILEA

Downloaded Source Files:

Name
☐ CYCLREC
✓ DI21PART

Figure 2-17 Choose the DBD

9. Examine the DBD detail and click **Finish** (Figure 2-18).



Available Files: DI21PART

Source

```
file DI21PART
  DBD DI21PART
      SEGM     NAME=PARTROOT,PARENT=0,BYTES=50,FREQ=250              00010001_
        FIELD    NAME=(PARTKEY,SEQ),TYPE=C,BYTES=17,START=1           00010101_
        SEGM     NAME=STANINFO,PARENT=PARTROOT,BYTES=85,FREQ=1        00010201_
          FIELD    NAME=(STANKEY,SEQ),TYPE=C,BYTES=2,START=1          00010301
        SEGM     NAME=STOKSTAT,PARENT=PARTROOT,BYTES=160,FREQ=2       00010401_
          FIELD    NAME=(STOCKEY,SEQ),TYPE=C,BYTES=16,START=1         00010501
          SEGM     NAME=CYCCOUNT,PARENT=STOKSTAT,BYTES=25,FREQ=1      00010601
```

Start Field: DI21PART

☐ Enable End Field selection

End Field:

Figure 2-18 The DBD details

10.Repeat this process for the PSB, as shown in Figure 2-19.

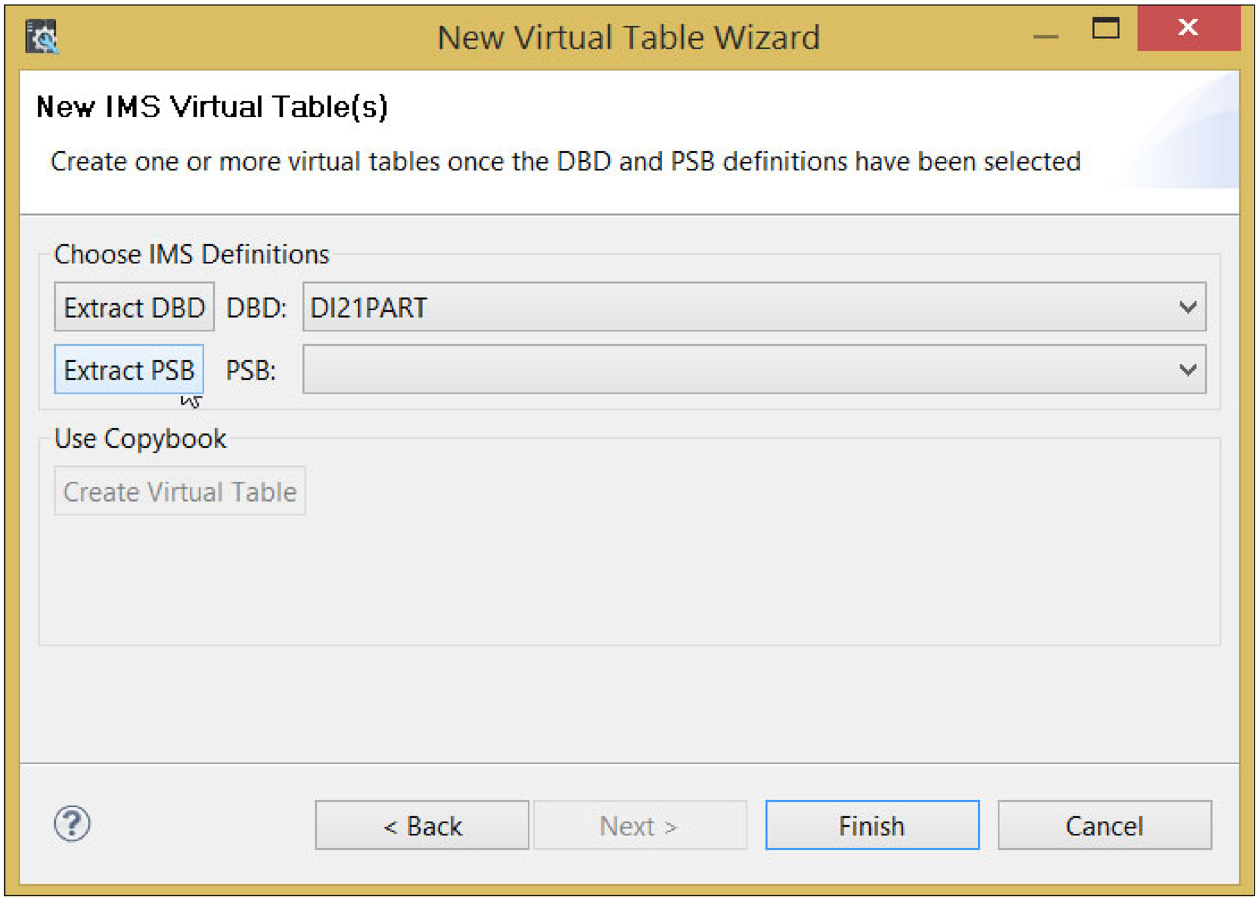
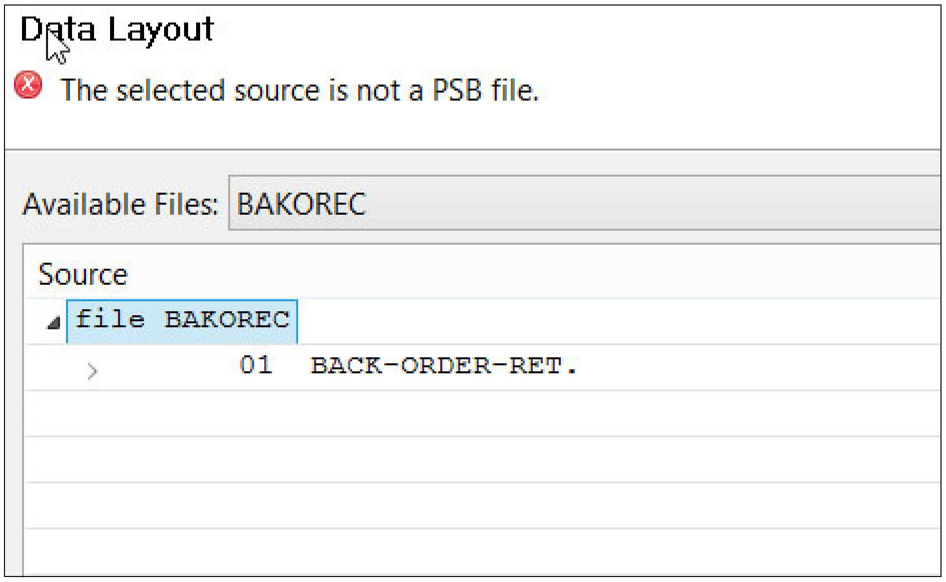


Figure 2-19 Extract the PSB

11.If you choose the wrong file, a message is displayed at top of the dialog, as shown in Figure 2-20.

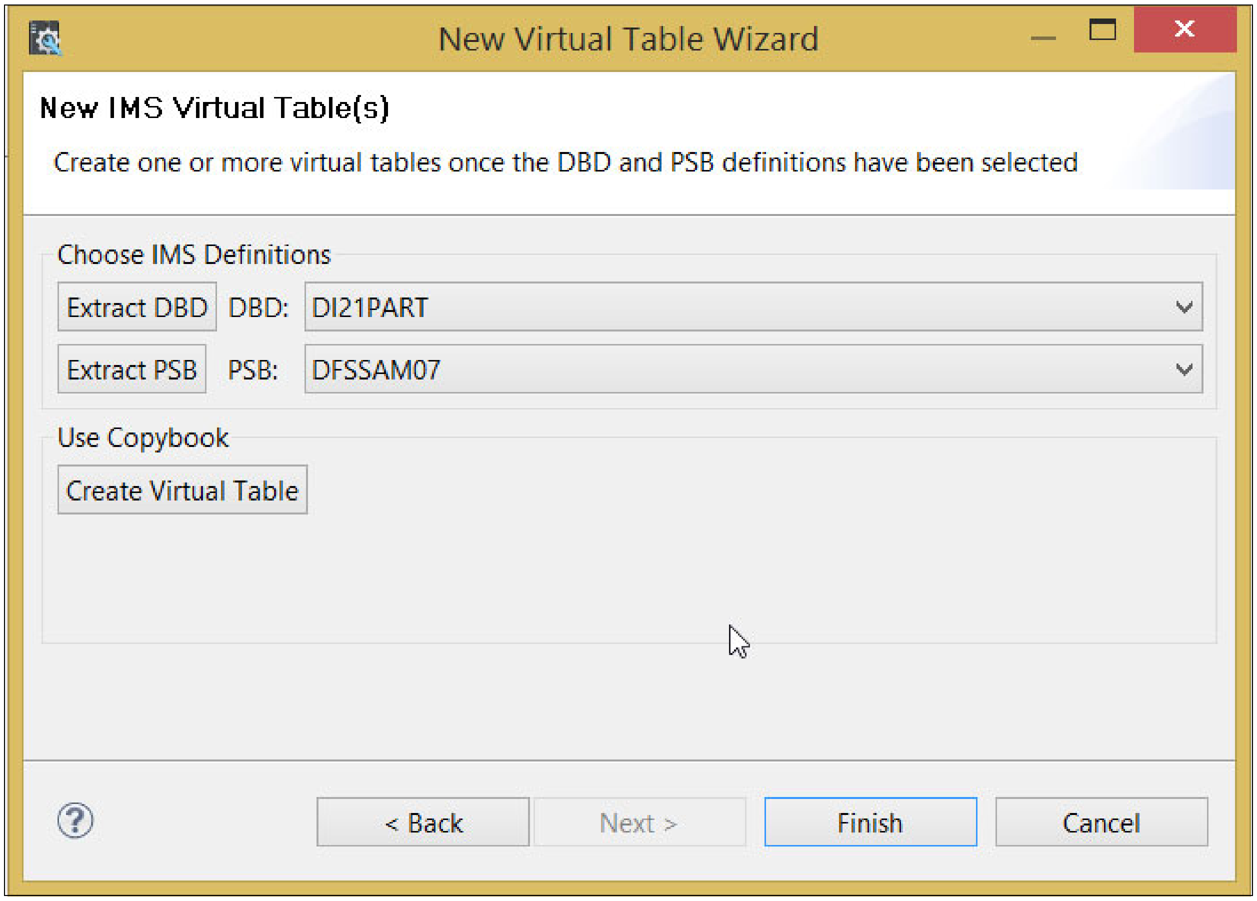

Data Layout

The selected source is not a PSB file.

Available Files: BAKOREC

Figure 2-20 Message for the wrong file type

12. After you select the DBD and the PSB, click **Create Virtual Table**. Navigate to the windows where you choose the copybook, as shown in Figure 2-21.


New Virtual Table Wizard

New IMS Virtual Table(s)

Create one or more virtual tables once the DBD and PSB definitions have been selected

Choose IMS Definitions

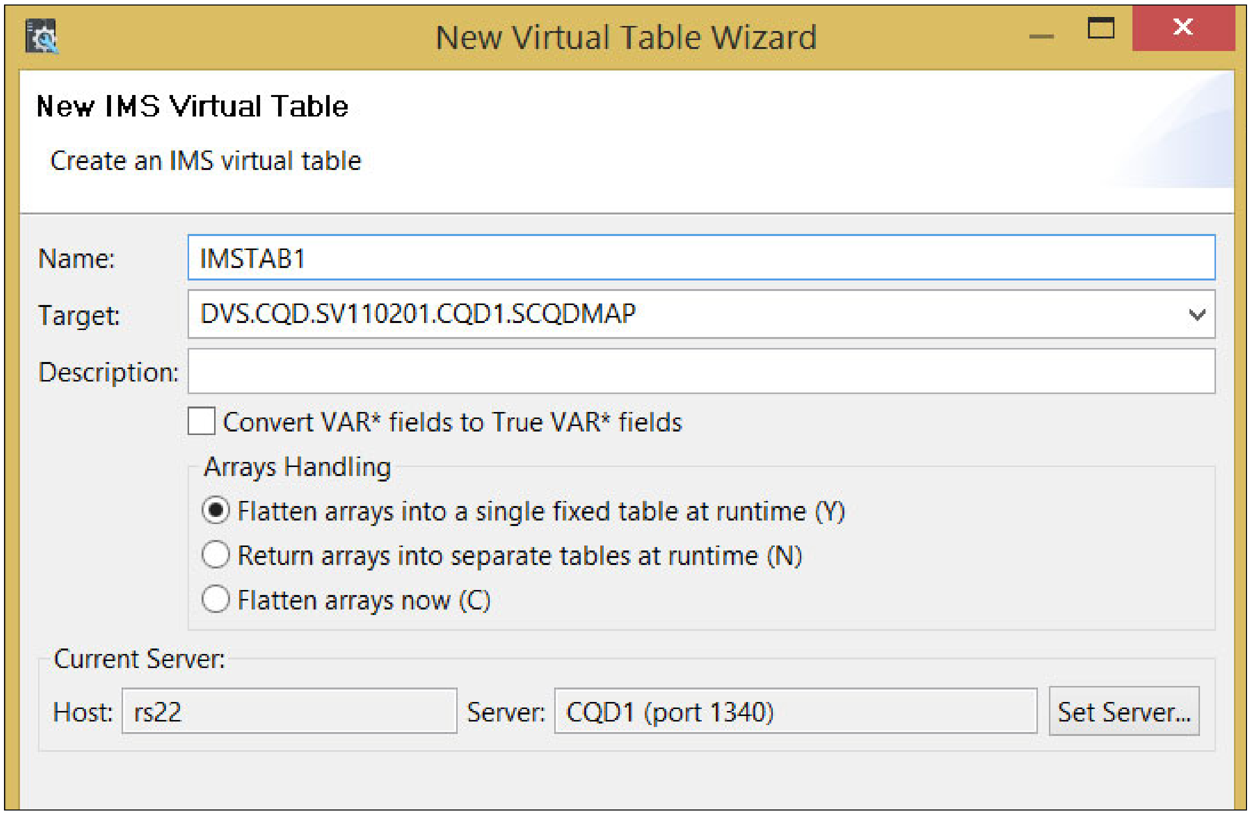
DBD: DI21PART

PSB: DFSSAM07

Use Copybook

Figure 2-21 Create virtual table

13. Enter a name for the virtual table, as shown in Figure 2-22.



New IMS Virtual Table

Create an IMS virtual table

Name:	IMSTAB1
Target:	DVS.CQD.SV110201.CQD1.SCQDMAP
Description:	

☐ Convert VAR* fields to True VAR* fields

Arrays Handling

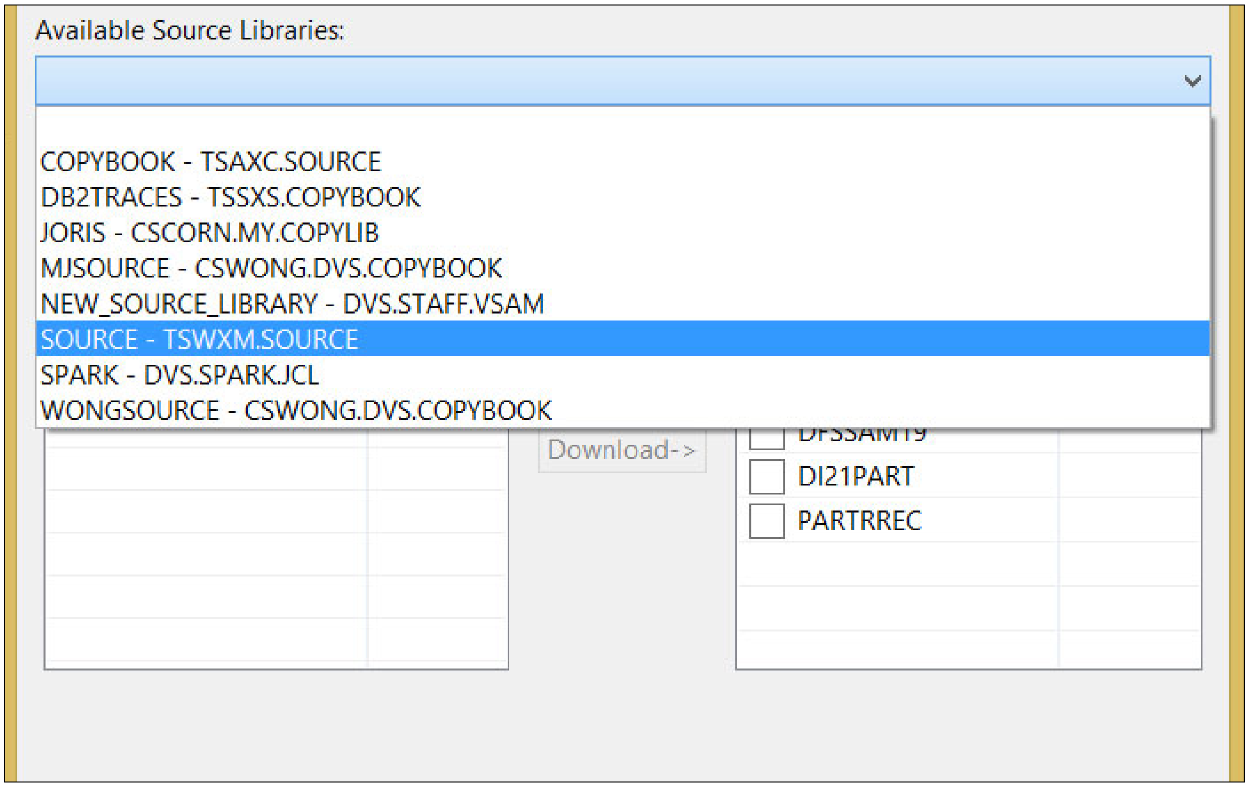
- ● Flatten arrays into a single fixed table at runtime (Y)
- ○ Return arrays into separate tables at runtime (N)
- ○ Flatten arrays now (C)

Current Server:

Host: rs22 Server: CQD1 (port 1340)

Figure 2-22 Naming the IMS virtual table

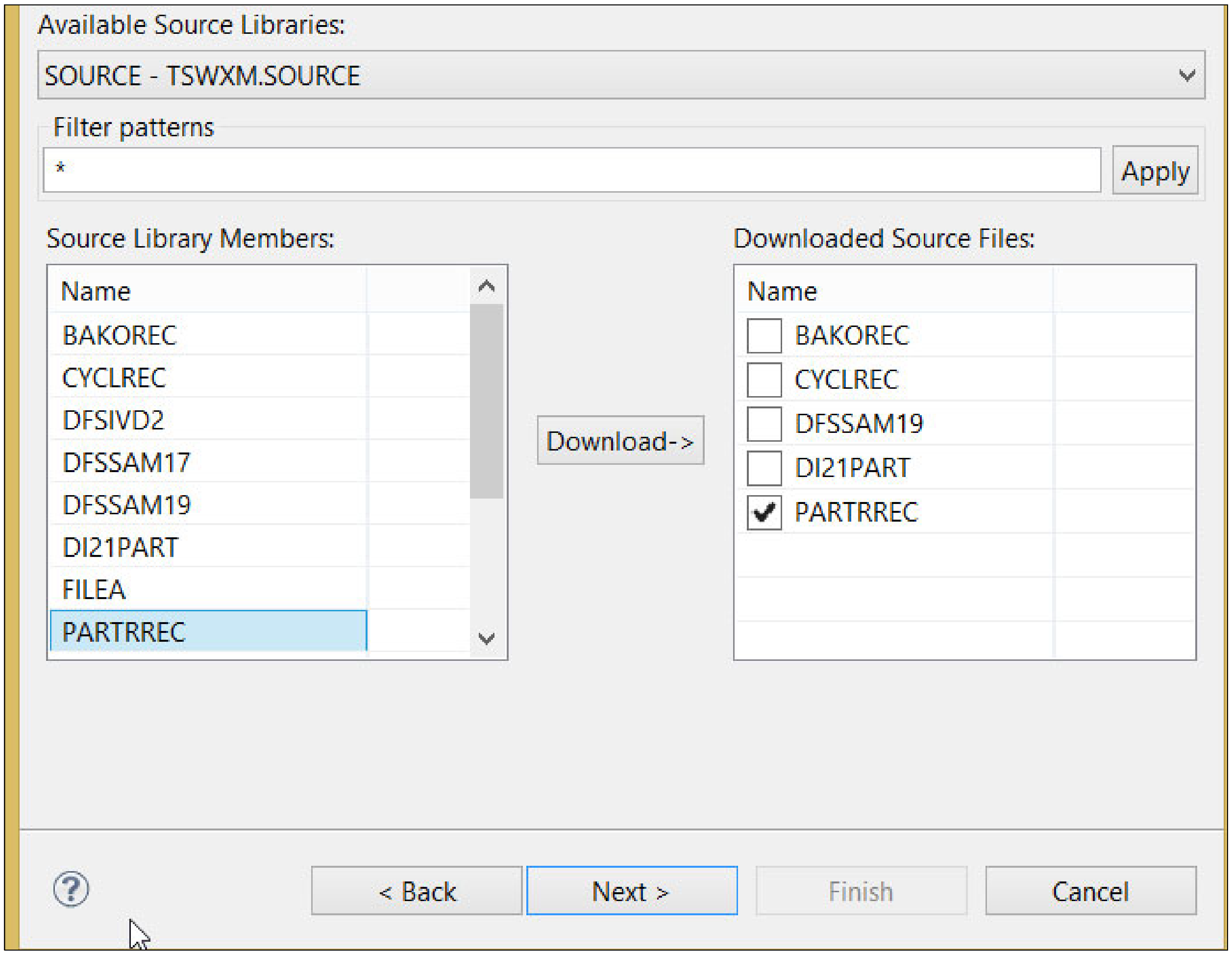
14. Select the source library with the copybook, as shown in Figure 2-23. In general, this library might not be the same library as the library that holds the DBD or PSB source. In the example that is shown here, the source files and the copybooks were all assembled in one data set ahead of time.


Available Source Libraries:

COPYBOOK - TSAXC.SOURCE
DB2TRACES - TSSXS.COPYBOOK
JORIS - CSCORN.MY.COPYLIB
MJSOURCE - CSWONG.DVS.COPYBOOK
NEW_SOURCE_LIBRARY - DVS.STAFF.VSAM
SOURCE - TSWXM.SOURCE
SPARK - DVS.SPARK.JCL
WONGSOURCE - CSWONG.DVS.COPYBOOK

Figure 2-23 Select the copybook source library

15. Select the copybook, as shown in Figure 2-24.


Available Source Libraries:

SOURCE - TSWXM.SOURCE

Filter patterns

*

Apply

Source Library Members:

Name
BAKOREC
CYCLREC
DFSIVD2
DFSSAM17
DFSSAM19
DI21PART
FILEA
PARTRREC

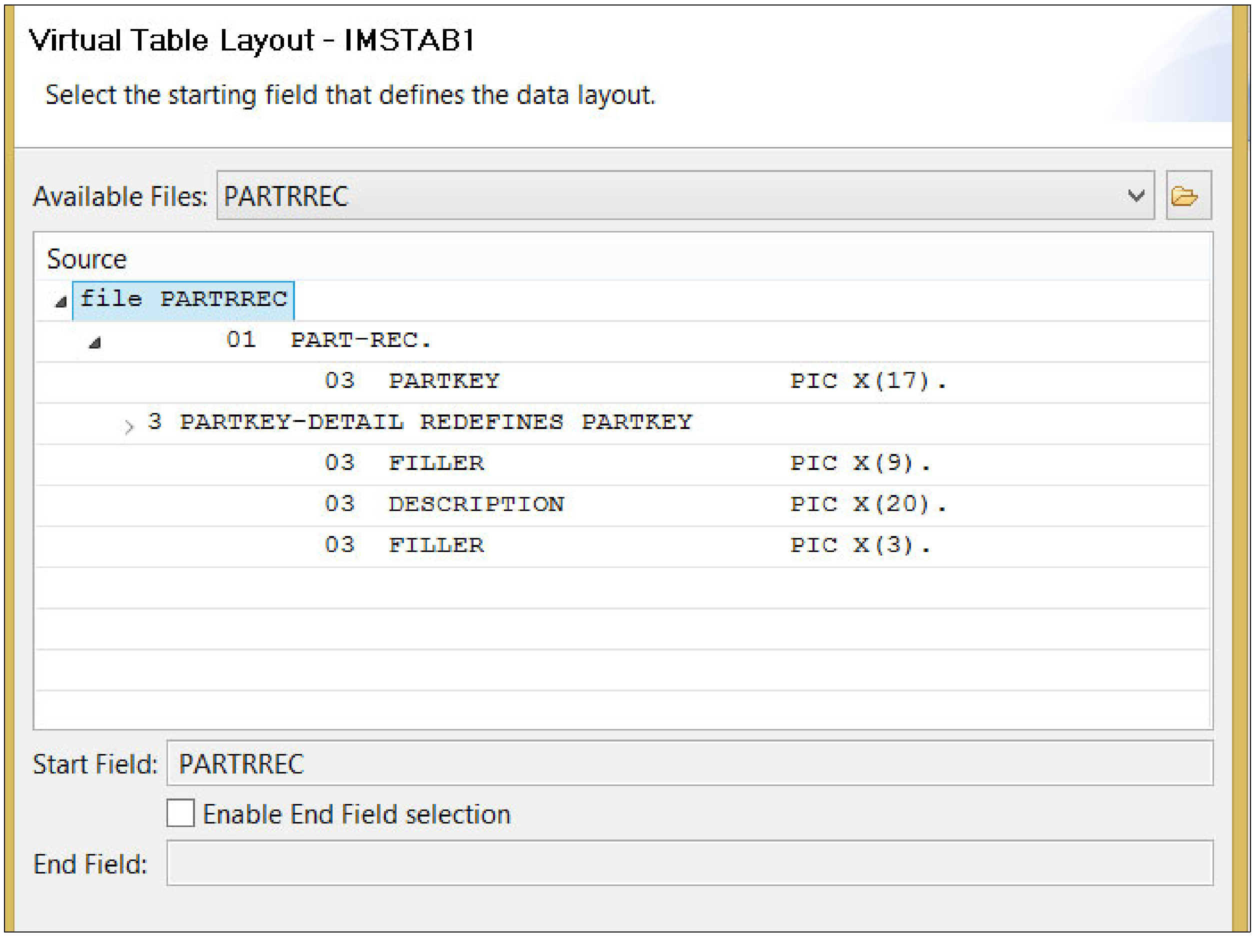
Download->

Downloaded Source Files:

	Name
☐	BAKOREC
☐	CYCLREC
☐	DFSSAM19
☐	DI21PART
✓	PARTRREC

Figure 2-24   Select the copybook

16. Examine the copybook, as shown in Figure 2-25.


Virtual Table Layout - IMSTAB1

Select the starting field that defines the data layout.

Available Files: PARTRREC

Source

```
file PARTRREC
        01   PART-REC.
             03   PARTKEY                   PIC X(17).
  3 PARTKEY-DETAIL REDEFINES PARTKEY
             03   FILLER                    PIC X(9).
             03   DESCRIPTION               PIC X(20).
             03   FILLER                    PIC X(3).
```

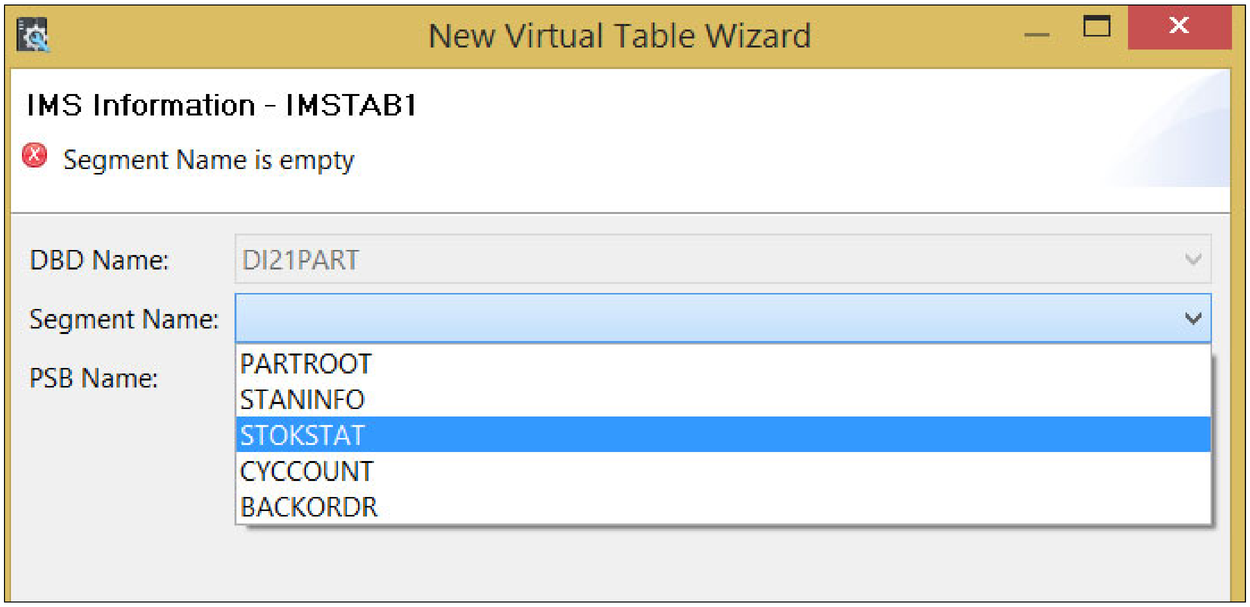
Start Field: PARTRREC

Enable End Field selection

End Field:

Figure 2-25   Examine the copybook

17. Select the segment, as shown in Figure 2-26.


New Virtual Table Wizard

IMS Information - IMSTAB1

Segment Name is empty

DBD Name: DI21PART

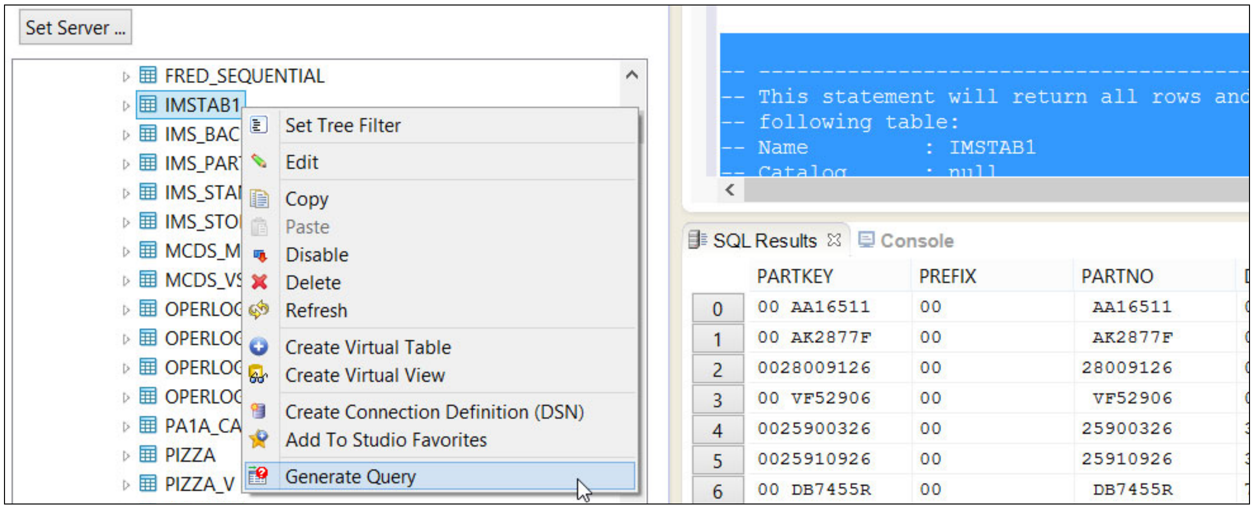
Segment Name:

PSB Name:

PARTROOT
STANINFO
STOKSTAT
CYCCOUNT
BACKORDR

Figure 2-26   Selecting the segment

18. Run a test query on the new virtual table, as shown in Figure 2-27.



```
-- ------------------------------------
-- This statement will return all rows and
-- following table:
-- Name          : IMSTAB1
-- Catalog       : null
```

	PARTKEY	PREFIX	PARTNO
0	00 AA16511	00	AA16511
1	00 AK2877F	00	AK2877F
2	0028009126	00	28009126
3	00 VF52906	00	VF52906
4	0025900326	00	25900326
5	0025910926	00	25910926
6	00 DB7455R	00	DB7455R

Figure 2-27 Generate a test query

2.3.4 Virtualize the VSAM data

VSAM data is one of the simpler data sources to create virtual tables. Example 2-11 shows a sample of a VSAM file to virtualize.

Example 2-11 VSAM file to virtualize

```
Browse           DVS.CUSTOMER.VSAM                              Top of 159
 Command ===>
Scroll PAGE
                             Type KSDS     RBA <RLS FILE>          Format CHAR
 Key                                          Col 1
 <==>+----10---+----2----+----3----+----4----+----5----+----6----+----7----+----
 ****  Top of data  ****
 ....USA           Daredevil Skiing              276 North Street         ......
 ....Finland       Flip-It Frisbee               Rattipolku 3             ......
 ....USA           Hoops Croquet Co.             Suite 415                40 Gro
 ....United Kingdom Gone Fishing Ltd              Unit 2                   83 Pon
 ....USA           Match Point Tennis            66 Homer Ave             ......
 ....United KingdomFanatical Athletes            20 Bicep Bridge Rd       Leyton
 ....Finland       Body Fit Sports               Peltolantie 2            ......
```

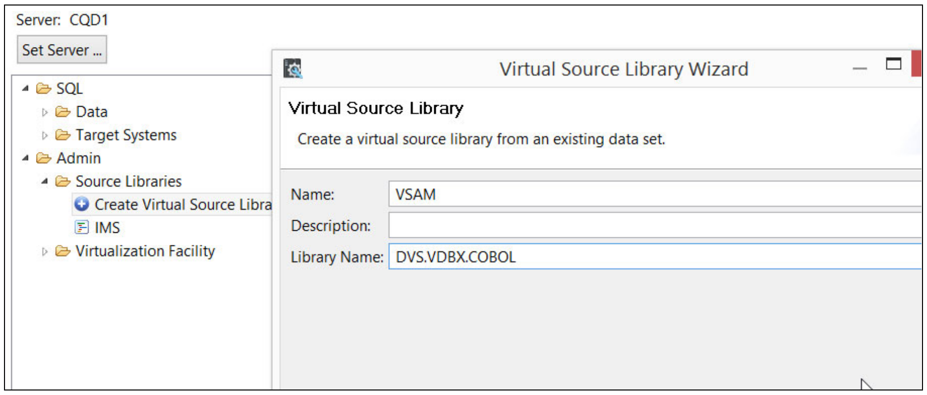
Example 2-12 shows a sample of the COBOL copybook for this file.

Example 2-12 COBOL copybook

```
BROWSE    DVS.VDBX.COBOL(CUSTOMER) - 01.02          Line 00
Command ===>
******************************** Top of Data ***************
     *******************************************************
     * COBOL DECLARATION FOR TABLE REDBOOK.CUSTOMERS
     *******************************************************
      01  DCLCUSTOMERS.
          10 CUSTNUM              PIC S9(9) USAGE COMP.
          10 COUNTRY              PIC X(14).
          10 NAME                 PIC X(30).
          10 ADDRESS                PIC X(25).
          10 ADDRESS2               PIC X(19).
          10 CITY                   PIC X(15).
          10 STATE                  PIC X(17).
          10 POSTALCODE             PIC X(9).
          10 CONTACT                PIC X(19).
          10 PHONE                  PIC X(14).
```

With the two files, we can create the virtual table. Follow these steps:

1. In the QMF Data Service Studio, on the Set Server tab, expand the tree to **Admin → Source Libraries** and double-click the plus sign (+) icon for the **Create Virtual Source Library**, as shown in Figure 2-28.



Create a virtual source library from an existing data set.

Name: VSAM
Description:
Library Name: DVS.VDBX.COBOL

Figure 2-28 Creating a VSAM source library

2. Enter the data set that holds the copybook. Click **Finish**.
3. Create a virtual table by right-clicking your server and selecting **Create Virtual Table**, as shown in Figure 2-29.

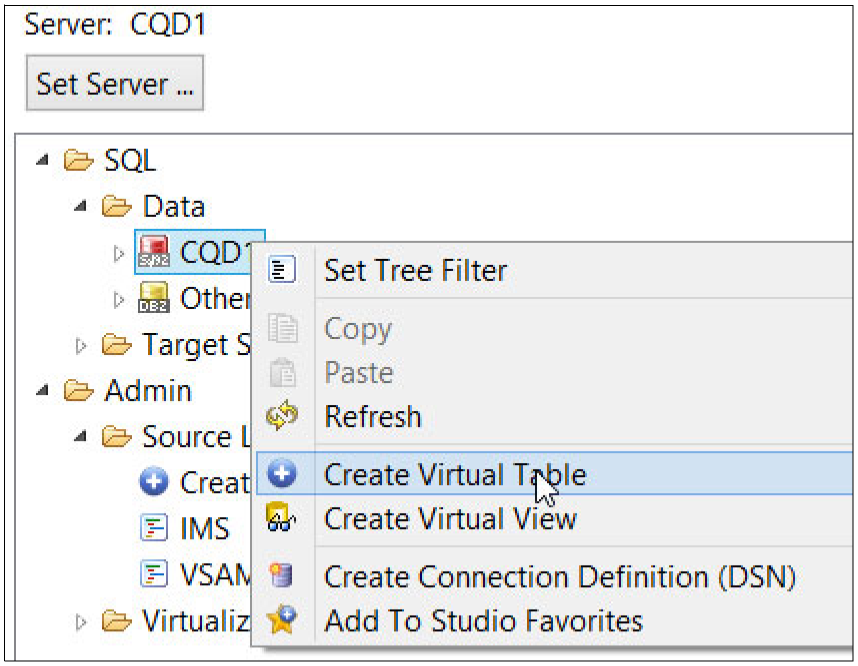
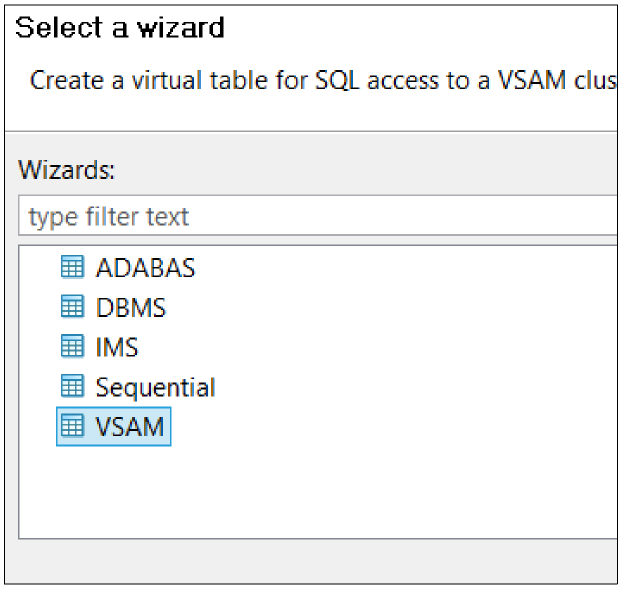


Figure 2-29 Create a virtual table

4. Select **VSAM** from the list, as shown in Figure 2-30.



Select a wizard

Create a virtual table for SQL access to a VSAM clus

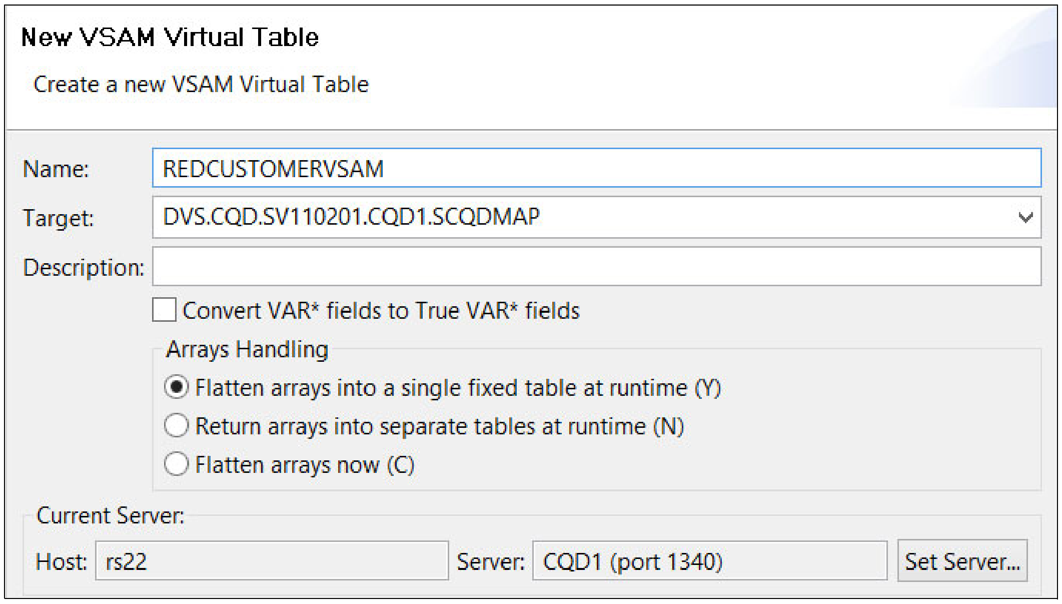
Wizards:

type filter text

ADABAS
DBMS
IMS
Sequential
VSAM

Figure 2-30 Select VSAM

5. Enter a name for the table, as shown in Figure 2-31.



New VSAM Virtual Table

Create a new VSAM Virtual Table

Name: REDCUSTOMERVSAM

Target: DVS.CQD.SV110201.CQD1.SCQDMAP

Description:

☐ Convert VAR* fields to True VAR* fields

Arrays Handling

● Flatten arrays into a single fixed table at runtime (Y)

○ Return arrays into separate tables at runtime (N)

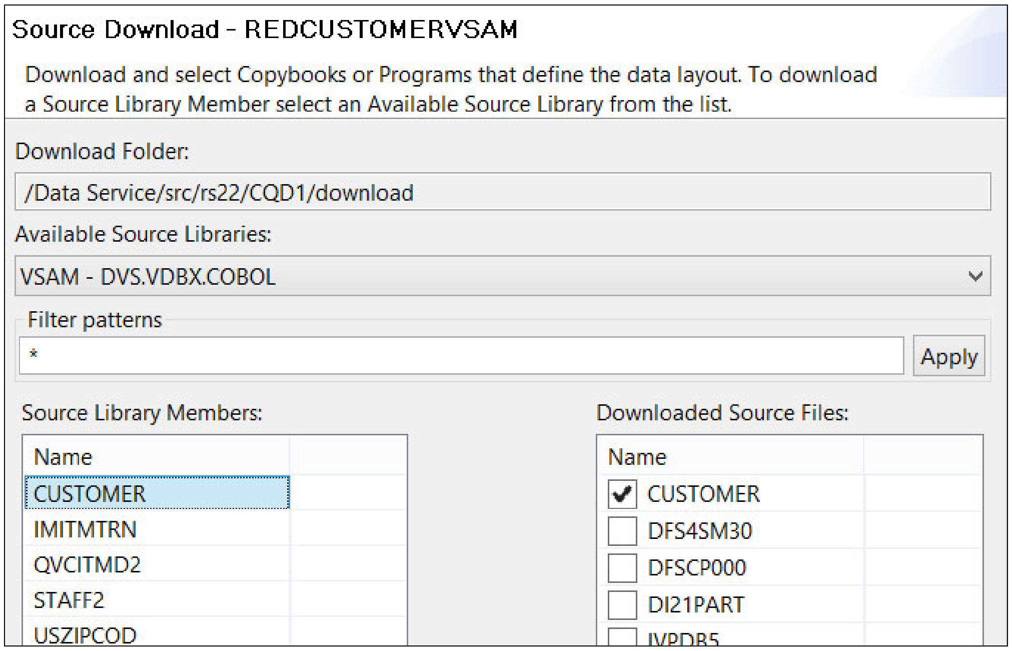
○ Flatten arrays now (C)

Current Server:

Host: rs22 Server: CQD1 (port 1340)

Figure 2-31 Name the table

6. Select the copybook to download, as shown in Figure 2-32.


Source Download - REDCUSTOMERVSAM

Download and select Copybooks or Programs that define the data layout. To download a Source Library Member select an Available Source Library from the list.

Download Folder: /Data Service/src/rs22/CQD1/download

Available Source Libraries: VSAM - DVS.VDBX.COBOL

Filter patterns: *

Source Library Members:

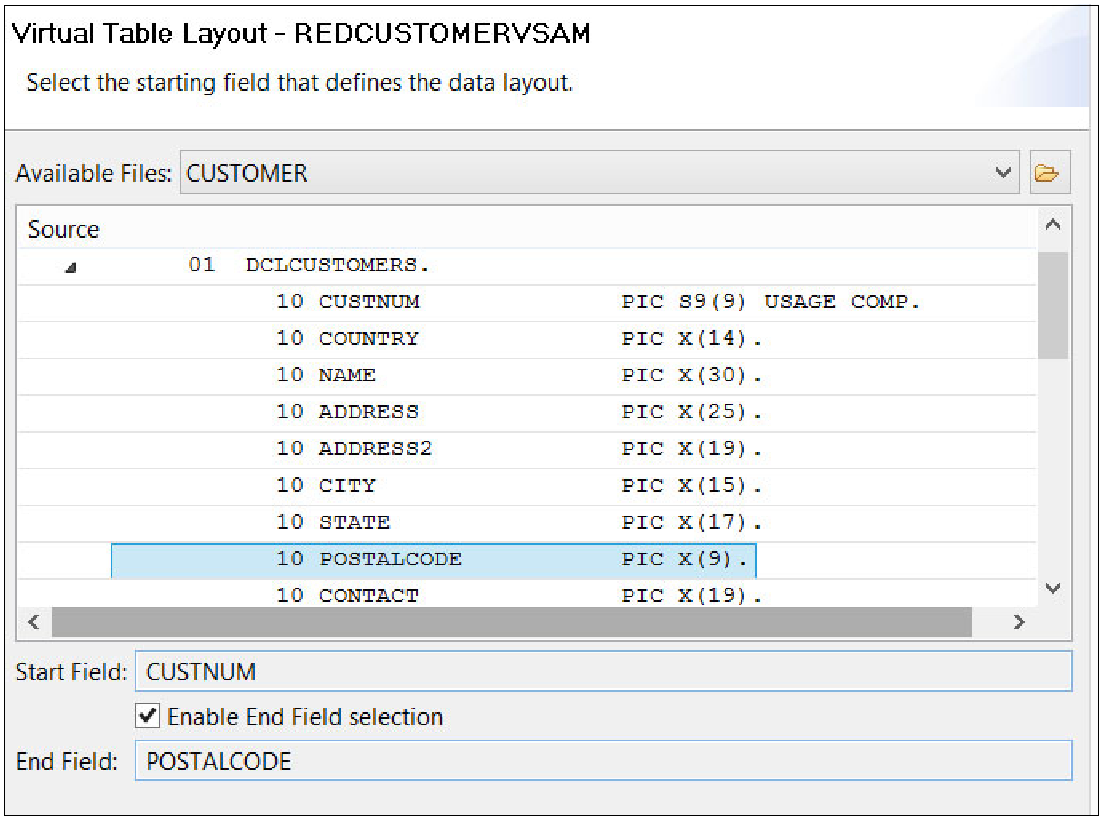
Name
CUSTOMER
IMITMTRN
QVCITMD2
STAFF2
USZIPCOD

Downloaded Source Files:

Name
✓ CUSTOMER
DFS4SM30
DFSCP000
DI21PART

Figure 2-32 Select the copybook for the VSAM file

7. By default, all columns are selected for the virtual table, but you can specify an end field by selecting **Enable End Field selection**. We stop at `POSTALCODE`, as shown in Figure 2-33.


Virtual Table Layout - REDCUSTOMERVSAM

Select the starting field that defines the data layout.

Available Files: CUSTOMER

Source
```
01  DCLCUSTOMERS.
    10 CUSTNUM                PIC S9(9) USAGE COMP.
    10 COUNTRY                PIC X(14).
    10 NAME                   PIC X(30).
    10 ADDRESS                PIC X(25).
    10 ADDRESS2               PIC X(19).
    10 CITY                   PIC X(15).
    10 STATE                  PIC X(17).
    10 POSTALCODE             PIC X(9).
    10 CONTACT                PIC X(19).
```

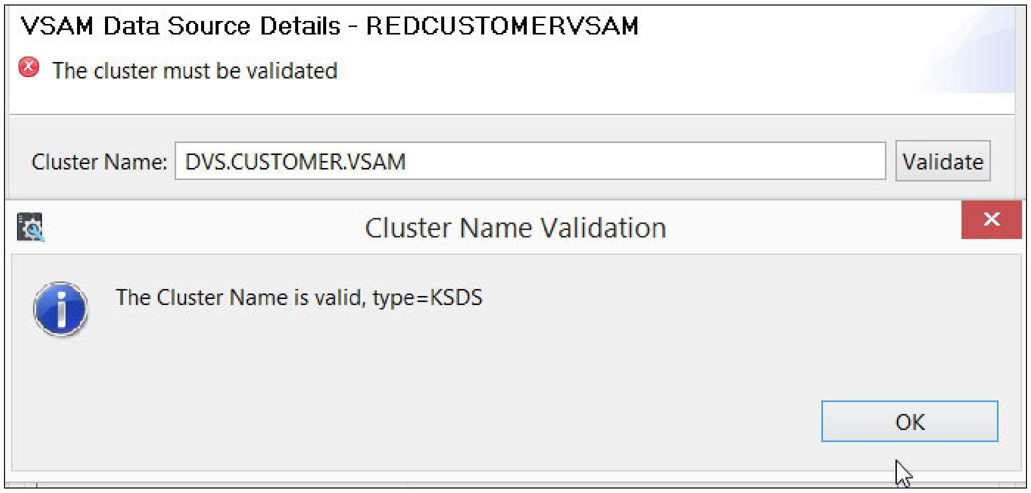
Start Field: CUSTNUM

✓ Enable End Field selection

End Field: POSTALCODE

Figure 2-33 Choosing the end field

8. Enter the location of the VSAM file for the cluster name and click **Validate**, as shown in Figure 2-34.


VSAM Data Source Details - REDCUSTOMERVSAM

The cluster must be validated

Cluster Name: DVS.CUSTOMER.VSAM

The Cluster Name is valid, type=KSDS

Figure 2-34 Validating the cluster

9. Click **Finish**. You can test the VSAM data access by right-clicking and selecting **Generate Query**, as shown in Figure 2-35.

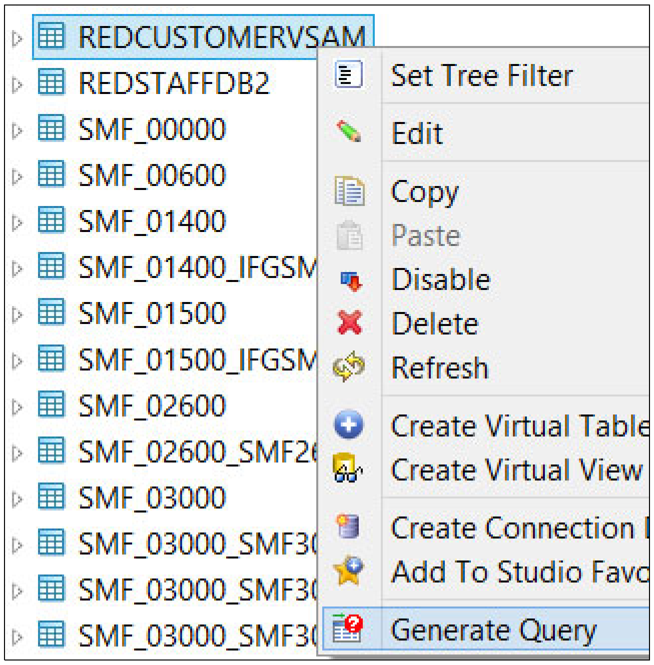
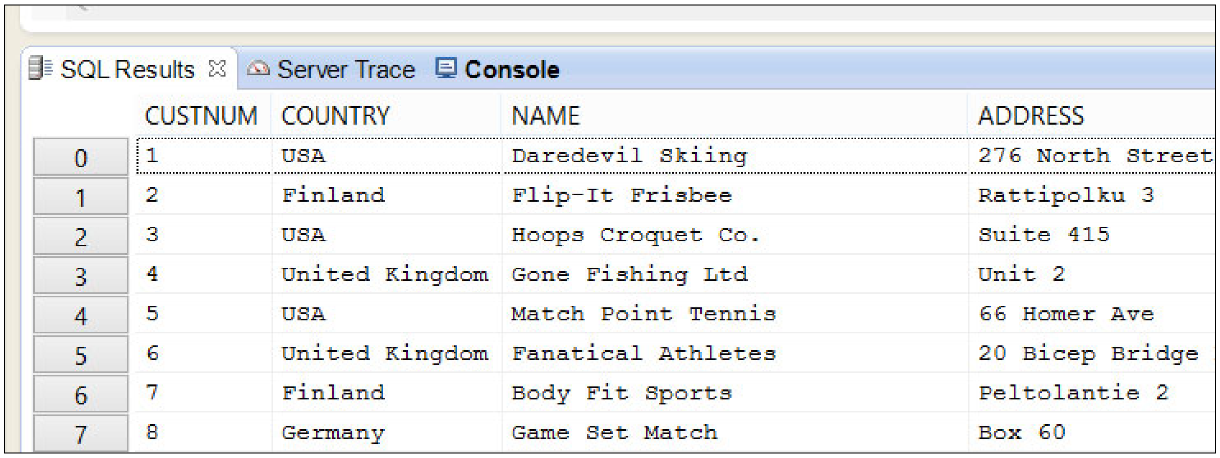


Figure 2-35 Generating a query

The data is returned in Figure 2-36.



	CUSTNUM	COUNTRY	NAME	ADDRESS
0	1	USA	Daredevil Skiing	276 North Street
1	2	Finland	Flip-It Frisbee	Rattipolku 3
2	3	USA	Hoops Croquet Co.	Suite 415
3	4	United Kingdom	Gone Fishing Ltd	Unit 2
4	5	USA	Match Point Tennis	66 Homer Ave
5	6	United Kingdom	Fanatical Athletes	20 Bicep Bridge
6	7	Finland	Body Fit Sports	Peltolantie 2
7	8	Germany	Game Set Match	Box 60

Figure 2-36   VSAM data that is returned by using SQL

2.4 Accessing the QMF Data Service data

The QMF Data Server Studio itself can run SQL and return data in a rudimentary fashion. QMF for Workstation and QMF for WebSphere are robust front ends for QDS.

2.4.1 Creating a QMF Data Service data source in QMF

Follow these steps:

1. With QMF for z/OS V11.2.1, QMF for Workstation has a new branch in the explorer tree for QMF Data Service connections, as shown in Figure 2-37.

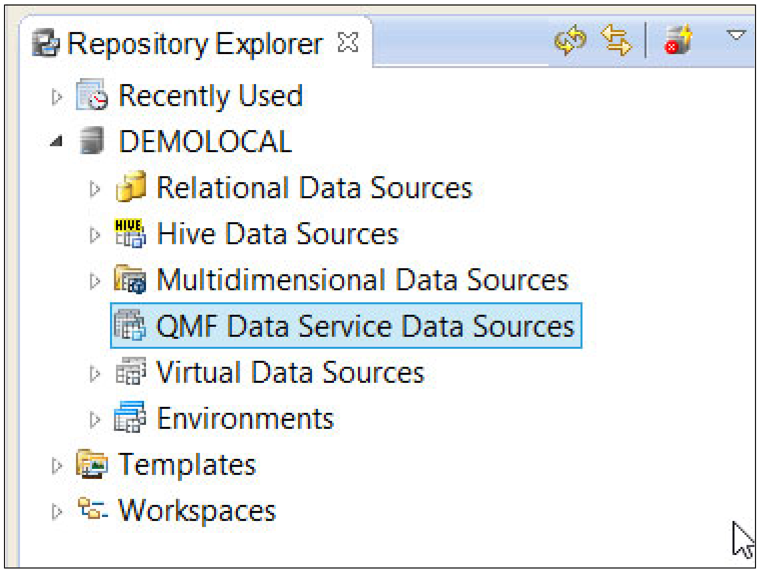


Figure 2-37   QMF Data Service branch

2. Right-click **QMF Data Service** and select **New** → **QMF Data Service Data Source**, as shown in Figure 2-38 to start the process.

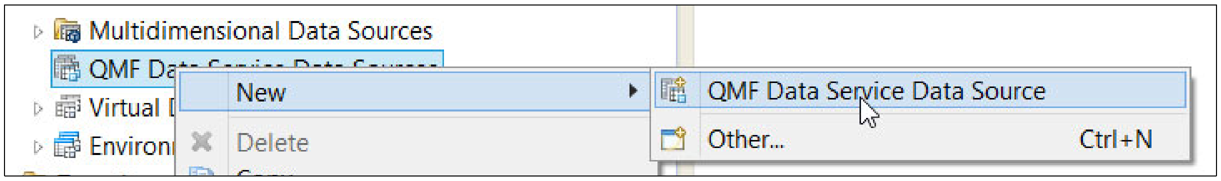
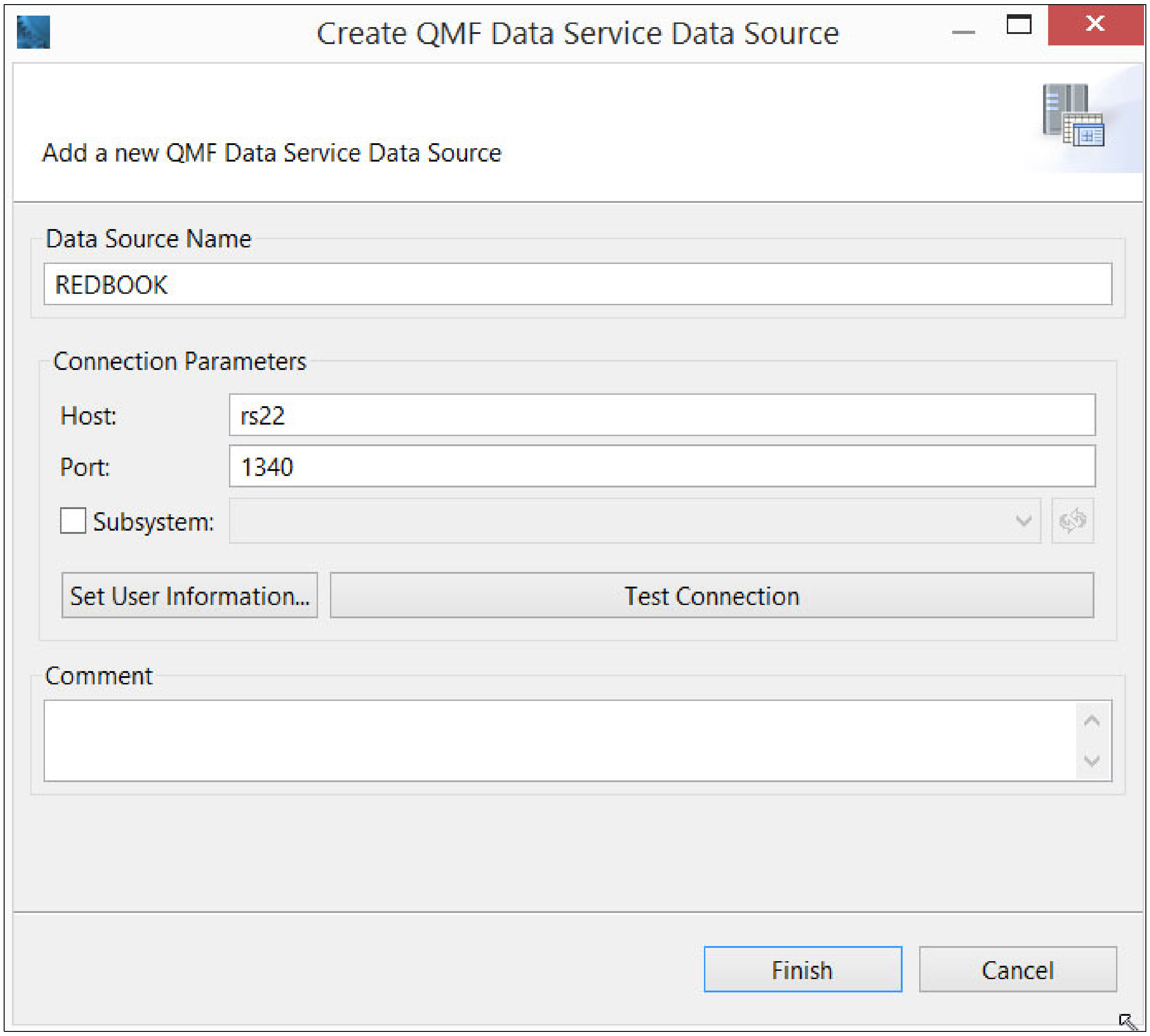

Multidimensional Data Sources
QMF Data Service Data Sources
Virtual
Environ
New
Delete
QMF Data Service Data Source
Other... Ctrl+N

Figure 2-38 Creating a QDS entry

3. Enter the connection information and click **Finish**, as shown in Figure 2-39.


Create QMF Data Service Data Source
Add a new QMF Data Service Data Source

Data Source Name: REDBOOK

Connection Parameters

Host: rs22

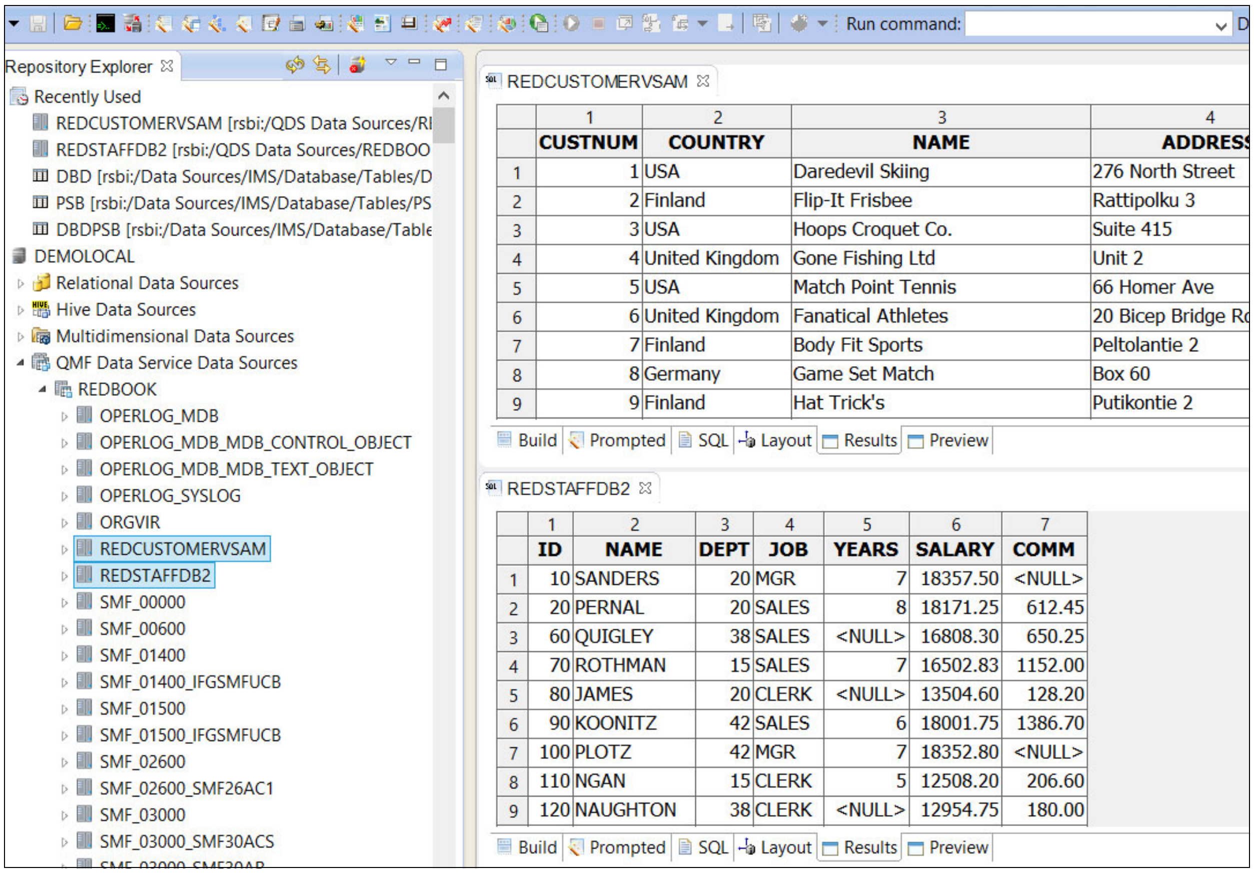
Port: 1340

Subsystem:

Set User Information...
Test Connection
Comment
Finish
Cancel

Figure 2-39 Enter the QDS connection information

4. Now, all users of this QMF repository can access these now-federated data sources. Figure 2-40 shows the DB2 data and the VSAM data side by side.



Recently Used
- REDCUSTOMERVSAM [rsbi:/QDS Data Sources/RI
- REDSTAFFDB2 [rsbi:/QDS Data Sources/REDBOO
- DBD [rsbi:/Data Sources/IMS/Database/Tables/D
- PSB [rsbi:/Data Sources/IMS/Database/Tables/PS
- DBDPSB [rsbi:/Data Sources/IMS/Database/Table

DEMOLOCAL
- Relational Data Sources
- Hive Data Sources
- Multidimensional Data Sources
- QMF Data Service Data Sources
  - REDBOOK
    - OPERLOG_MDB
    - OPERLOG_MDB_MDB_CONTROL_OBJECT
    - OPERLOG_MDB_MDB_TEXT_OBJECT
    - OPERLOG_SYSLOG
    - ORGVIR
    - REDCUSTOMERVSAM
    - REDSTAFFDB2
    - SMF_00000
    - SMF_00600
    - SMF_01400
    - SMF_01400_IFGSMFUCB
    - SMF_01500
    - SMF_01500_IFGSMFUCB
    - SMF_02600
    - SMF_02600_SMF26AC1
    - SMF_03000
    - SMF_03000_SMF30ACS

REDCUSTOMERVSAM

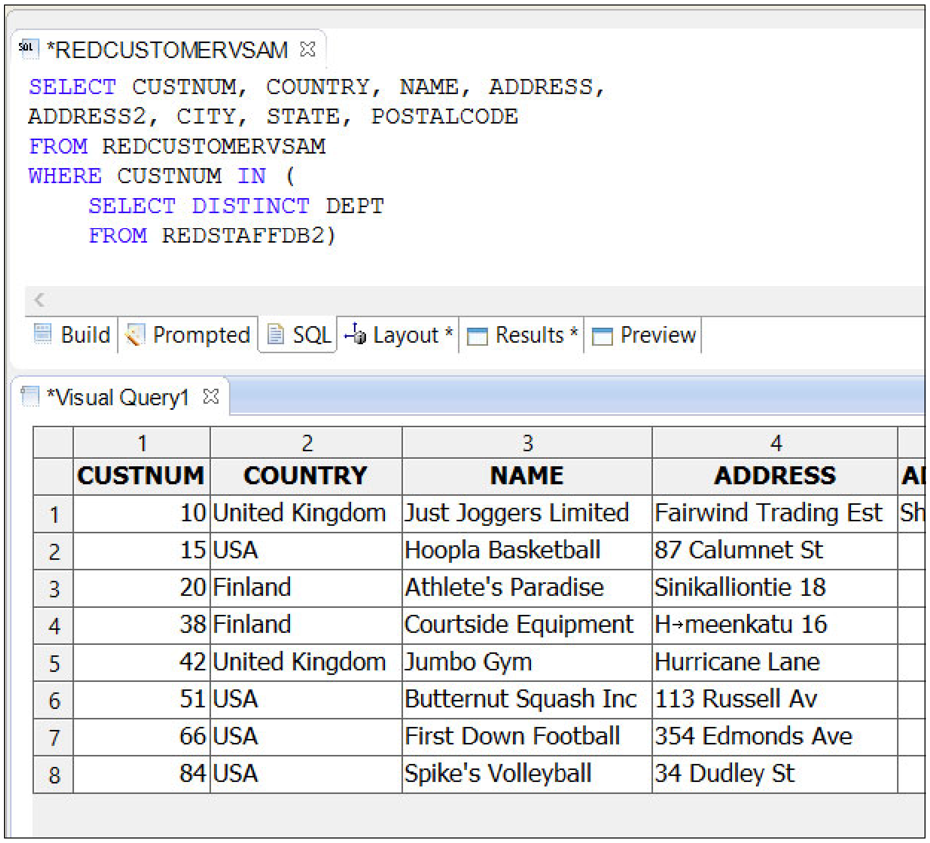
	1	2	3	4
	CUSTNUM	COUNTRY	NAME	ADDRESS
1	1	USA	Daredevil Skiing	276 North Street
2	2	Finland	Flip-It Frisbee	Rattipolku 3
3	3	USA	Hoops Croquet Co.	Suite 415
4	4	United Kingdom	Gone Fishing Ltd	Unit 2
5	5	USA	Match Point Tennis	66 Homer Ave
6	6	United Kingdom	Fanatical Athletes	20 Bicep Bridge Rd
7	7	Finland	Body Fit Sports	Peltolantie 2
8	8	Germany	Game Set Match	Box 60
9	9	Finland	Hat Trick's	Putikontie 2

REDSTAFFDB2

	1	2	3	4	5	6	7
	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	10	SANDERS	20	MGR	7	18357.50	<NULL>
2	20	PERNAL	20	SALES	8	18171.25	612.45
3	60	QUIGLEY	38	SALES	<NULL>	16808.30	650.25
4	70	ROTHMAN	15	SALES	7	16502.83	1152.00
5	80	JAMES	20	CLERK	<NULL>	13504.60	128.20
6	90	KOONITZ	42	SALES	6	18001.75	1386.70
7	100	PLOTZ	42	MGR	7	18352.80	<NULL>
8	110	NGAN	15	CLERK	5	12508.20	206.60
9	120	NAUGHTON	38	CLERK	<NULL>	12954.75	180.00

Figure 2-40 DB2 and VSAM side by side

5. Figure 2-41 shows the DB2 and VSAM data that is federated in a single result set.


*REDCUSTOMERVSAM

```
SELECT CUSTNUM, COUNTRY, NAME, ADDRESS,
ADDRESS2, CITY, STATE, POSTALCODE
FROM REDCUSTOMERVSAM
WHERE CUSTNUM IN (
    SELECT DISTINCT DEPT
    FROM REDSTAFFDB2)
```

	1	2	3	4	
	CUSTNUM	COUNTRY	NAME	ADDRESS	AI
1	10	United Kingdom	Just Joggers Limited	Fairwind Trading Est	Sh
2	15	USA	Hoopla Basketball	87 Calumnet St	
3	20	Finland	Athlete's Paradise	Sinikalliontie 18	
4	38	Finland	Courtside Equipment	H→meenkatu 16	
5	42	United Kingdom	Jumbo Gym	Hurricane Lane	
6	51	USA	Butternut Squash Inc	113 Russell Av	
7	66	USA	First Down Football	354 Edmonds Ave	
8	84	USA	Spike's Volleyball	34 Dudley St	

Figure 2-41 Joined DB2 and VSAM data

2.5 Additional considerations

Consider the following information:

- ADABAS is a supported data source. Its setup is similar to DB2 and IMS.
- SMF records and the system log are also data sources. The steps to set up access are brief, and they are described in detail in *DB2 QMF Data Service Solutions Guide*, GC27-8808-01.
- The QDS is a DRDA client, and it can connect to remote DB2 instances. So, one instance of QDS on an LPAR can map DB2 tables from all of the other locations on your network as well as the VSAM files, IMS, and ADABAS on the local LPAR.

- If you need to federate data that is not DB2 data across different LPARs, for example, IMS on several different LPARs or on different physical machines, you need to set up QDS on each LPAR. Then, on one system, set up connections to the other QDS instances as though they were DB2. Then, the virtual tables at those remote QDS instances can be mapped to a single QDS.
- The following statement is in the prerequisites for QMF for WebSphere in *Installing and Managing DB2 QMF for Workstation and DB2 QMF for WebSphere,* GC27-6742-02:

  > *"QMF for WebSphere requires Java Runtime Environment (JRE) V7 or later to access QMF Data Service data sources."*

  As of this writing, when IBM WebSphere Application Server is installed, it comes with and uses IBM Java runtime environment (JRE) V6. As a result, QMF for WebSphere, when it is deployed with a default configuration of IBM WebSphere Application Server, does not display a QMF Data Service node in the Repository Explorer tree and does not display the built-in QMF Data Service Java Database Connectivity (JDBC) drivers.

  Even if the QMF Data Service connection was already configured in QMF for Workstation and queries were written, you cannot run them in QMF for WebSphere if WebSphere Application Server (or any other Web Application Server, such as Apache Tomcat) uses the Java 6 JRE. The documentation for WebSphere Application Server describes the steps to upgrade the JRE that is in use by WebSphere Application Server.

2.6 Summary

After the QMF Data Service is installed and configured, you need to perform a few steps to set up connections to data sources:

1. Edit the CDQ1IN00 server initialization member to provide the connection information and to enable the DoThis flag.
2. For DB2, edit CQDBINDD and run it to bind the QDS packages at each DB2 subsystem that you need to access.
3. Edit the CDQ1PROC that is run as a started task to start the QDS server. You need to enter specific resource fields, depending on the data source.
4. After the customizations, use the desktop QMF Data Service Studio to set up the virtual tables.
5. Use QMF for Workstation, QMF for WebSphere, or QMF Vision to query, report, visualize, and build applications from these virtual tables.

3

Using self-service business intelligence

"*Self-service business intelligence*" applies to user tools that are less complex for the non-technical user. They offer an intuitive interface to access data from various sources and provide easy-to-use menus and options to create dashboards, charts, and other business output. Many powerful business intelligence tools are available, but many have so much function and offer so many options that a non-skilled individual cannot effectively use them.

Now, in IBM DB2 Query Management Facility for z/OS V11.2.1 (QMF), IBM addresses the self-service user community with a new member. QMF Vision is a key enhancement to the QMF product for self-service dashboards and data exploration. Empowering users is a goal of many organizations to reduce decision-making time, reduce IT requests and involvement, increase accessibility to data, and improve the organization's agility.

Users typically know the data that they want and the application that it is in, but they do not know the database that stores that data, what the structure of the database is, or even how to gain access to it. The need for this information creates a requirement for other people, typically, the IT department and database administrators (DBAs), to create the data sets and objects to be analyzed by users. With the addition of QMF Vision, the user can build their own visualizations from provided data sets.

The user still does not need to know the database or how to build queries, but they now can quickly view the data in the manner that they need, plus all of the variations of it. Now, IT and the DBAs can produce and publish reliable data sets and tables one time. They can ensure that users are using the same clean, consistent data across the company. Users can then focus on collaborating on insights that are gained from the data without worrying whether the insights are based on the correct data.

This chapter includes the following topics:

- QMF Vision installation summary
- Interface
- Accessing data
- Creating your visualizations
- Collaboration

3.1 QMF Vision installation summary

This section outlines the installation process for QMF Vision.

3.1.1 Time guidance

When the installation process is correctly prepared, the installation process can be completed quickly. During a full QMF installation or an upgrade, a QMF Vision executable is created that can now be run on the server to which QMF Vision is installed. The executable provides an installation wizard that guides the administrator through the installation process. The preparation portion includes identifying your servers and establishing accessibility. See server requirements in 3.1.4, “Architecture” on page 46.

3.1.2 Recommended personnel

Few people are involved in the installation. You need an administrator for the server that you are installing on, or you need “Run as administrator” permission when you launch the executable. A QMF Workstation/WebSphere administrator is needed for subsequent connection into the QMF environment. QMF WebSphere is required to connect QMF Vision to a QMF repository, making the queries and tables that are published in QMF Workstation/WebSphere available to the QMF Vision users. If you use Lightweight Directory Access Protocol (LDAP) security, an LDAP administrator might be needed.

3.1.3 Back-end activities

The QMF Vision installation requires a MongoDB instance for the storage of the metadata. During the QMF Vision installation, the installer auto-detects an existing instance of MongoDB. If an existing instance of MongoDB is not found, the directory can be manually set to an existing installation or a new instance of MongoDB can be downloaded and installed as part of the installation process. The MongoDB instance does not need to be on the same server as QMF Vision, but it can be, depending on the architecture that you want.

Data can be accessed by using either QMF WebSphere or QMF Data Service. You need to install either QMF WebSphere or QMF Data Service and configure it independently of QMF Vision. During the setup and configuration of QMF Vision, a connection is made into these environments to open published data so that the users can build their visualizations.

3.1.4 Architecture

Before you begin installation, ensure that your system is configured. Figure 3-1 on page 47 shows the overall architecture of QMF Vision.

The server requirements are listed:

- QMF Vision supports the following Windows versions:
  - Microsoft Windows 7 64-bit
  - Windows Server 2008 R2 Service Pack 1
  - Windows Server 2012 with .NET Framework 3.5 enabled

Ensure that your production environment has the following minimum configurations:

- 4-core microprocessor
- 8 GB RAM
- 250 GB free disk space

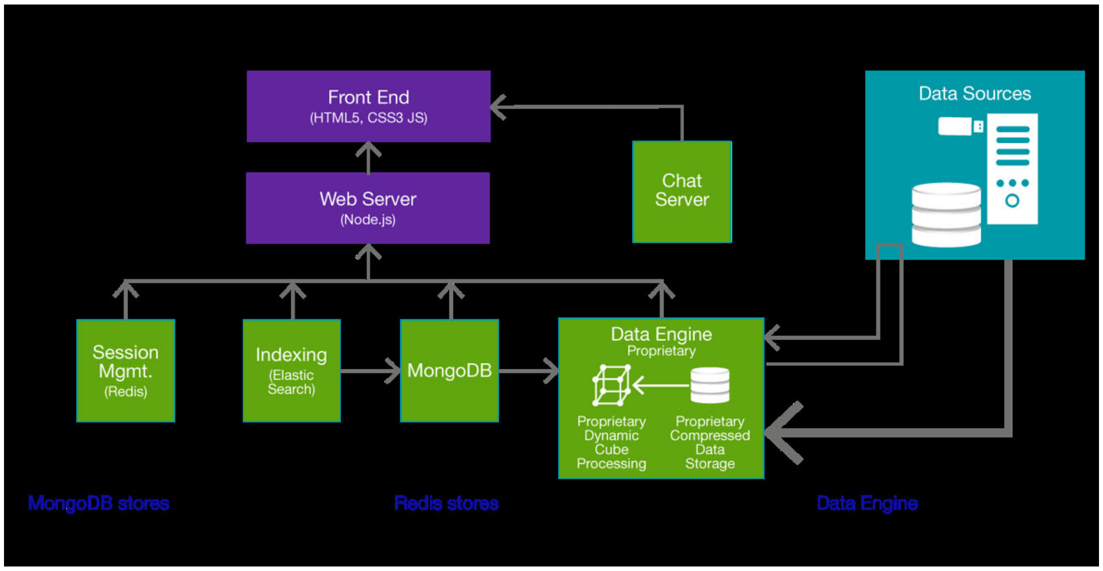


Figure 3-1 Architecture

3.2 Interface

The interface of QMF Vision is described.

3.2.1 User access

A web browser is used to access QMF Vision. A URL is generated during the installation process in this form:

```
http://<servername>:3000/qmf#/login
```

This URL is provided to users who can access QMF Vision by using their preferred web browser. This URL does not change unless the server name or IP is changed by IT, so it can be bookmarked, saved as a favorite, or made accessible as a link from another page. Nothing is downloaded to the client machine, and no plug-ins are necessary for operation. All users access the same installation to allow object sharing and instant messaging.

Separate installations of the QMF Vision component cannot interoperate. Files cannot be transferred between separate installations of the QMF component.

Note: QMF incorporates HTML 5 responsive design, which enables mobile usage.

A user can request an account by accessing the provided URL and selecting **Create an account** within the login window, as shown in Figure 3-2.

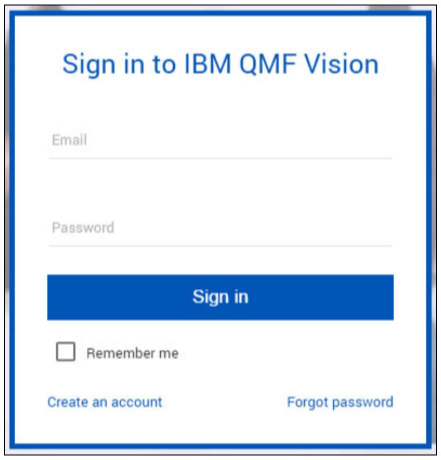


Figure 3-2 Create an account

After the information is submitted, an administrator must approve the account before the user can access the interface.

A user can be invited to use QMF Vision by using an automated email. The email address is entered into QMF Vision, which triggers an email to the user with an invitation to finish the creation of their account and to use the software.

Other options include connecting to LDAP and uploading multiple email addresses that are listed in a Microsoft Excel file. For more information, see *Getting Started with DB2 QMF Vision*, GC27-8805-01.

3.2.2 User administration

By default, the first user to create an account and log in has "administrator" permissions. All administration is handled in the same interface that users work within. The administration area is accessed by clicking your ***username*** and then selecting **Settings** (Figure 3-3).

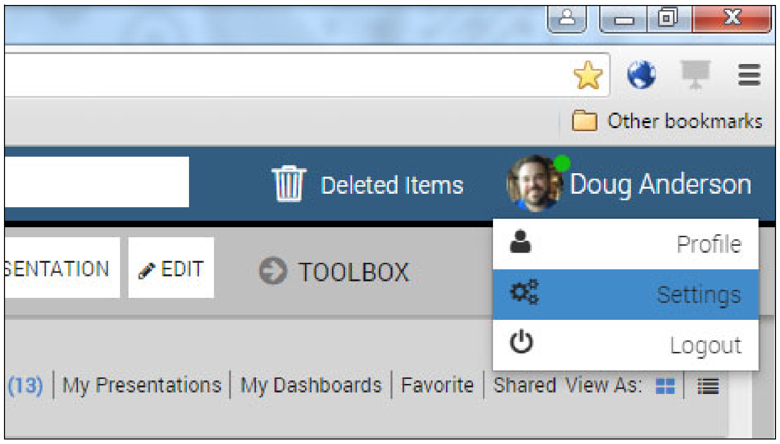


Figure 3-3 Accessing the administration area

The administrator establishes the company profile, manages users, and works with the default display settings.

Note: Other users can be designated as administrators by the initial administrator.

3.3 Accessing data

This section describes the use of QMF WebSphere and connecting to QMF Data Service.

3.3.1 Using QMF WebSphere

QMF WebSphere/Workstation interfaces enable connectivity to multiple data sources. Use QMF WebSphere/Workstation interfaces for the preparation of data, which can then be available to users of the QMF Vision interface. QMF Vision is most efficiently used when you need to visualize summary-level data.

A connection to QMF WebSphere can be added within the Data area. Enter the URL that links to your installed QMF WebSphere environment by using this format:

```
http(s)://[server]:[port]/[ContextRoot]
```

For example, use this format:

```
http://www.ibm.com:9080/QMFWebSphere112
```

After you enter the URL, perform the following steps:

1. Click **Connect** (Figure 3-4).

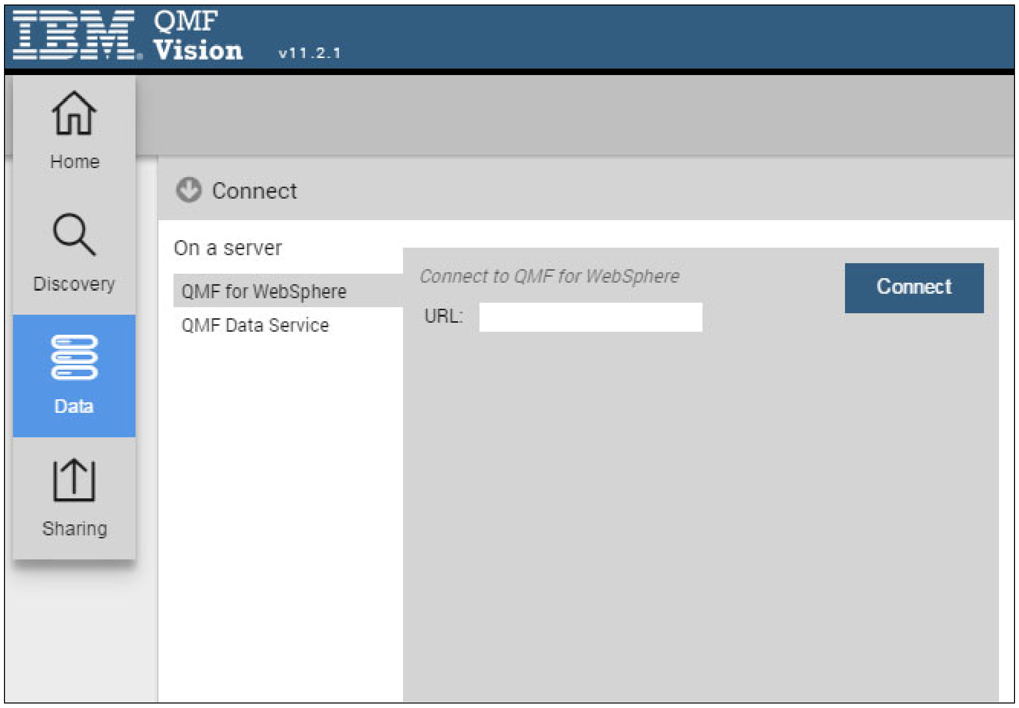


Figure 3-4 Connecting to QMF for WebSphere

2. The available Repository Connections are displayed, as shown in Figure 3-5. If the URL is entered incorrectly or a connectivity problem exists, a message box alerts you to the issue and provides the context of the problem.

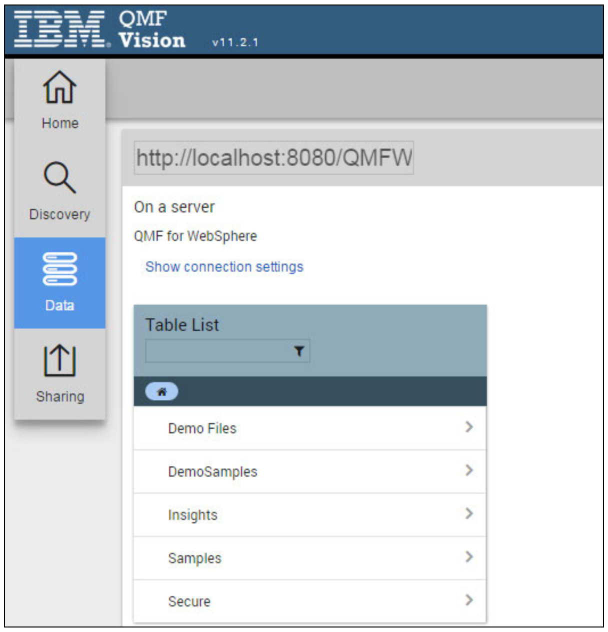
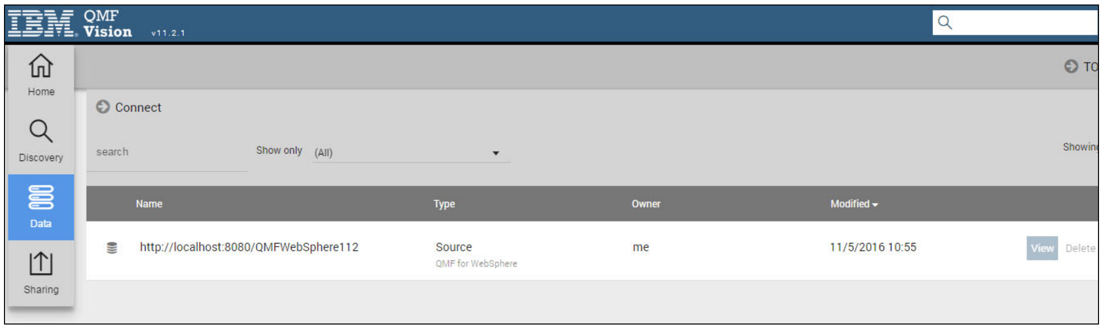


Figure 3-5 Connection settings

3. By default, the name of the data source is the URL that is used for the connection. This name can be changed to make it more easily identifiable. In Figure 3-5, the URL that is shown in the gray bar is the name of the connection. By clicking within this area, you can edit the name of the data source. After the connection is completed, you can browse the repository for accessible data by clicking **View**.

4. After you create the data source connections, they show in a list view, as shown in Figure 3-6.



Name	Type	Owner	Modified
http://localhost:8080/QMFWebSphere112	Source QMF for WebSphere	me	11/5/2016 10:55

Figure 3-6 Connection list

5. Data needs to be explicitly available to be accessible within the QMF Vision interface. A query or a table can be published to QMF Vision from which users can create their visualizations. You can publish a query or table from either the QMF Workstation or QMF WebSphere User or Administrator interface by right-clicking the object and selecting **Publish to QMF Vision** from the menu that is shown in Figure 3-7.

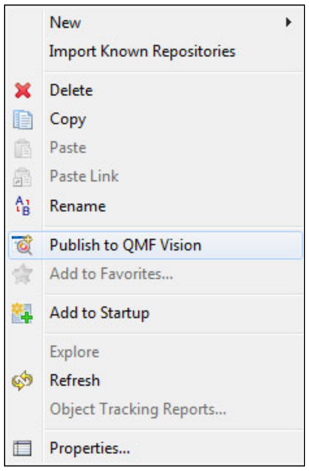
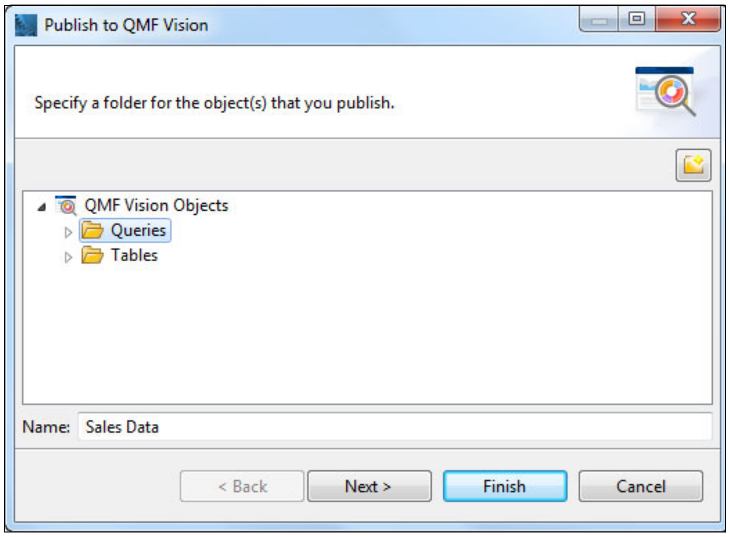


Figure 3-7 Publish to QMF Vision

6. Set the name of the object and folder location to use, as shown in Figure 3-8. You can create folders to better organize the objects. These folders are displayed in QMF Vision when a user browses through the available queries or tables.


Publish to QMF Vision

Specify a folder for the object(s) that you publish.

QMF Vision Objects
Queries
Tables

Name: Sales Data

Figure 3-8 Publishing

7. Click **Next** to enter the credentials, if needed, or click **Finish** to complete the publishing process.

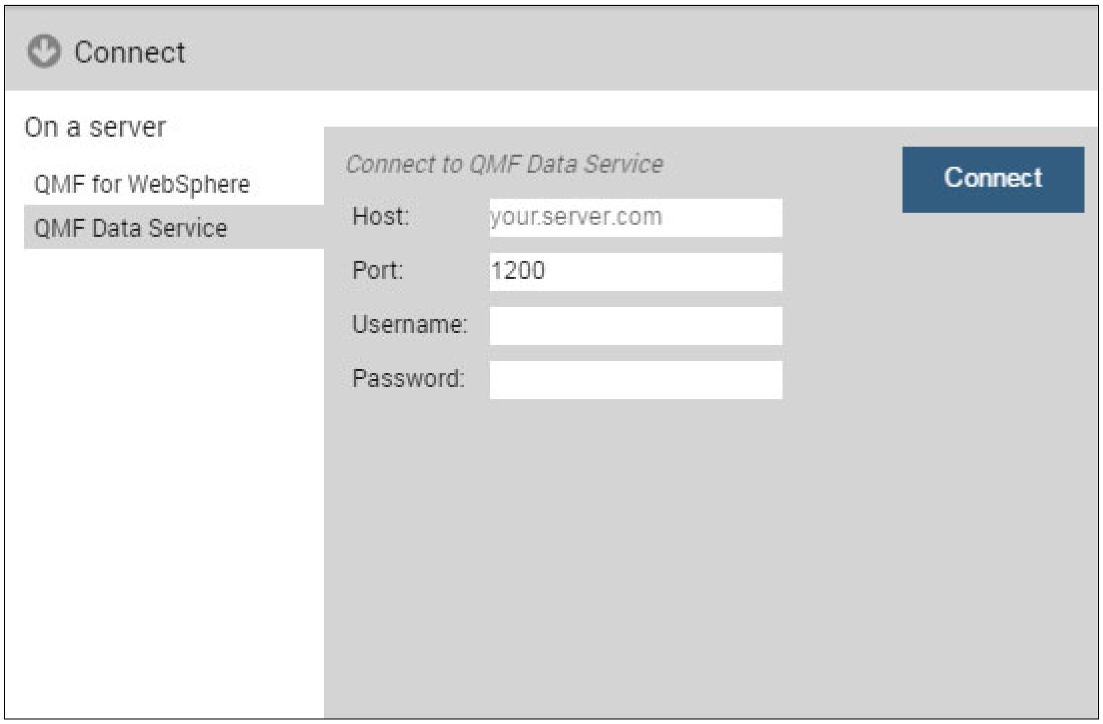
Note: Only objects that were published are visible to users.

3.3.2 Connecting to QMF Data Service

The most common method of accessing data from QMF Data Service is through a connection to the WebSphere/Workstation environment. This connection to the WebSphere/Workstation environment is where most power users create queries against the available mainframe data, prepare the queries, and then publish the queries to the QMF Vision environment.

However, sometimes, a direct connection to QMF Data Service files is needed. This connection is available within QMF Vision. Direct access also allows the use of a data pack where the QMF Vision user can create manual joins between mainframe files with common data. Follow these steps:

1. You can connect directly from the Data area by selecting **QMF Data Service** and entering the correct connection information. After you enter the connection information, click **CONNECT** to complete the process. See Figure 3-9.



Host: your.server.com
Port: 1200
Username:
Password:

Figure 3-9 QMF Data Service connection

2. If you connect successfully, an option is presented to select the QMF Data Service database that you want, as shown in Figure 3-10 and Figure 3-11.

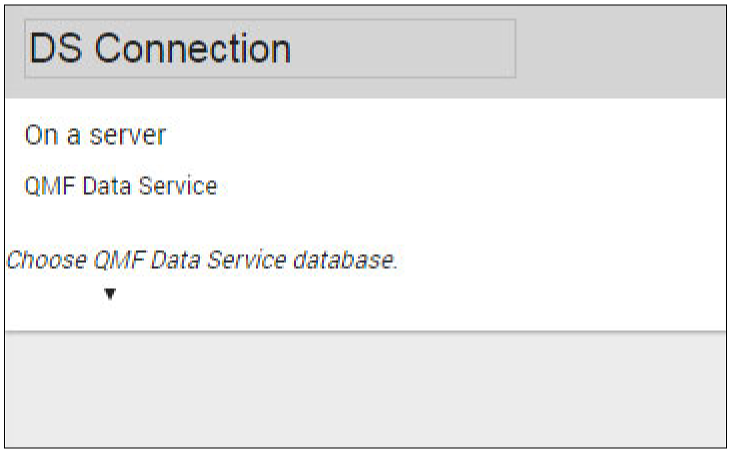


Figure 3-10 Connection made

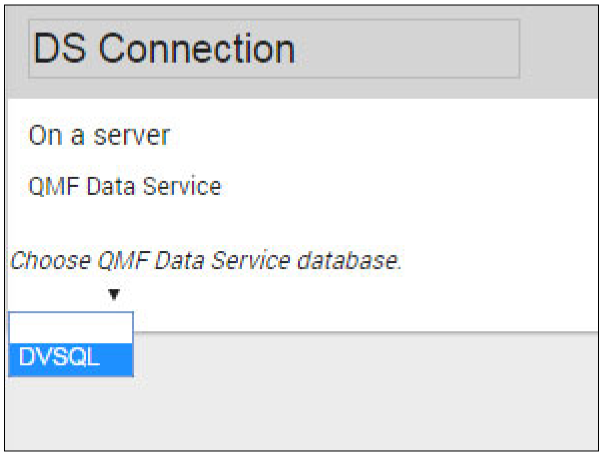
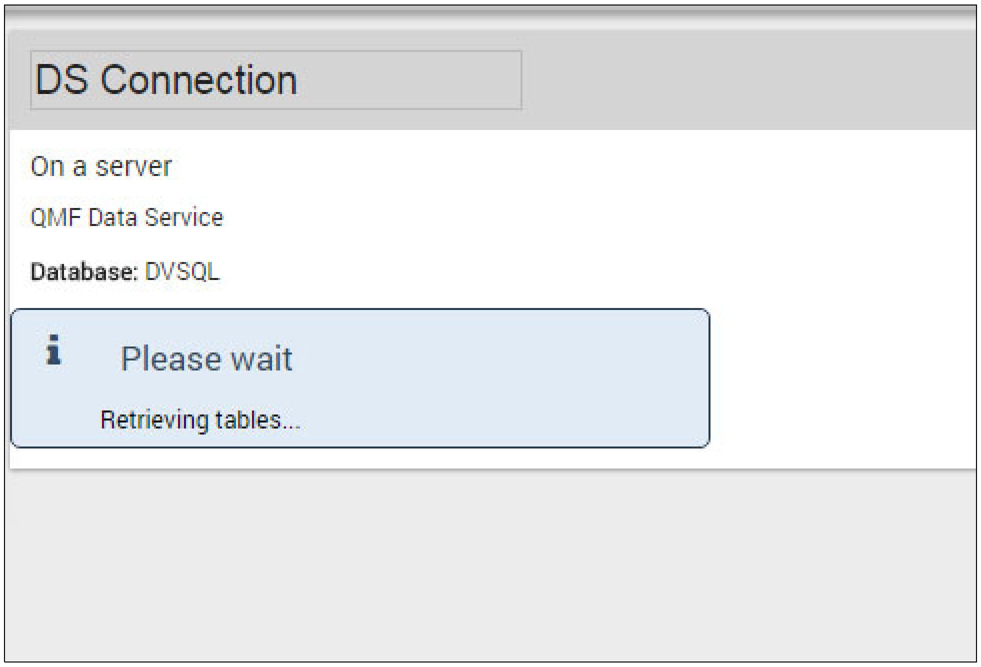


Figure 3-11 Select the QMF Data Service database

3. After you select a QMF Data Service database, a blue box informs you to wait while QMF Data Service retrieves tables (Figure 3-12). This process can take time based on how many files are virtualized and available with QMF Data Service.



DS Connection

On a server

QMF Data Service

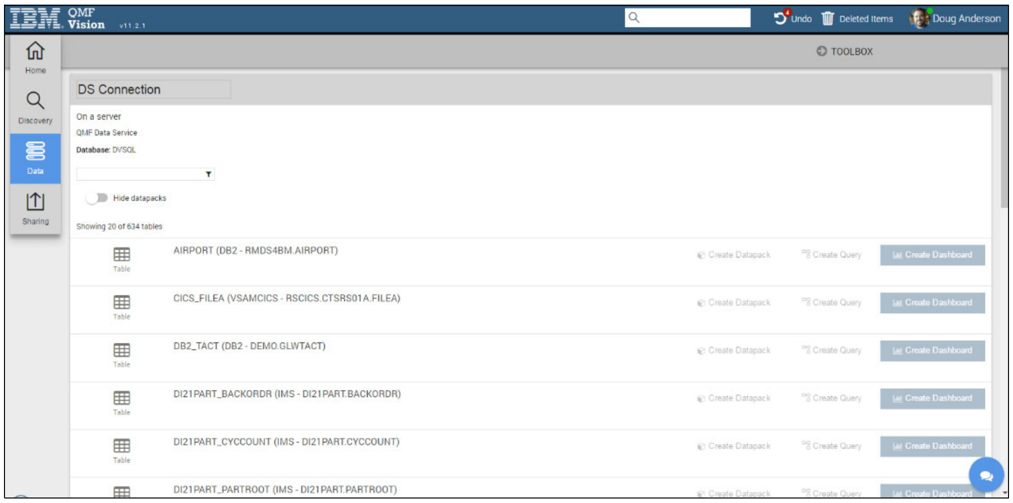
Database: DVSQL

Please wait

Retrieving tables...

Figure 3-12 Retrieving tables

4. After the tables are retrieved, a list displays and the capability to build queries and dashboards is available. After the initial connection and retrieval, the table list is shown on the subsequent access to create objects (Figure 3-13). A specific file can be selected to create a query or dashboard. Also, the data pack can be created. This process is covered in *Getting Started with DB2 QMF Vision*, GC27-8805-01.



DS Connection

On a server

QMF Data Service

Database: DVSQL

Showing 20 of 634 tables

AIRPORT (DB2 - RMDS4BM.AIRPORT)

CICS_FILEA (VSAMCICS - RSCICS.CTSRS01A.FILEA)

DB2_TACT (DB2 - DEMO.GLWTACT)

DI21PART_BACKORDR (IMS - DI21PART.BACKORDR)

DI21PART_CYCCOUNT (IMS - DI21PART.CYCCOUNT)

DI21PART_PARTROOT (IMS - DI21PART.PARTROOT)

Figure 3-13 Table list for a QMF Data Service connection

5. If the QMF Data Service option under CONNECT appears in red letters and shows “`Unavailable`” when you click it (Figure 3-14), you must install the IBM Data Service Open Database Connectivity (ODBC) driver. The installation process is documented in *Installing and Managing DB2 QMF for Workstation and DB2 QMF for WebSphere*, GC27-6742-02.

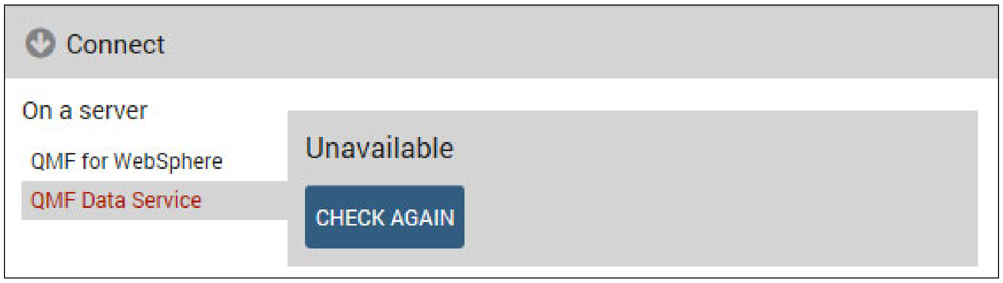


Figure 3-14 Data Service connection is unavailable

3.4 Creating your visualizations

This section contains the following considerations when you create visualizations:

- ▶ Understanding the initial home window
- ▶ Building a dashboard
- ▶ Reports and working with data
- ▶ Editing or replacing a query
- ▶ Coordinated highlighting and associations

3.4.1 Understanding the initial home window

Each user has their own home window that opens after the user logs in (Figure 3-15). If a user is new, their home window is empty until they build dashboards or until dashboards are shared with them by other users.

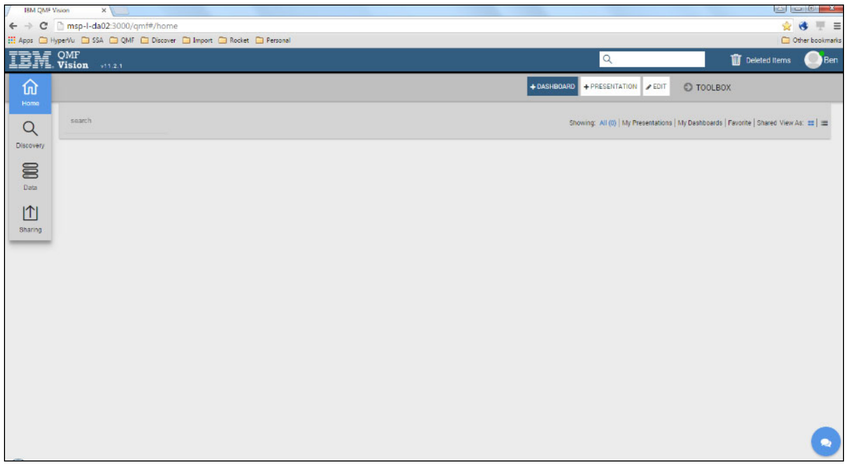

IBM QMF Vision
msp-l-da02:3000/qmf#/home
Deleted Items
Ben
+ DASHBOARD
+ PRESENTATION
EDIT
TOOLBOX
Home
Discovery
Data
Sharing
search
Showing: All (0) | My Presentations | My Dashboards | Favorite | Shared View As:

Figure 3-15 New user home window

Figure 3-16 shows the home window of an active user with a tile view of available dashboards to both view and interact. The dashboards can be shown in this tile form or in a list fashion. Other filter options are available in the upper-right area to help a user quickly access the dashboard that is needed for analysis. The user can select additional filter options, such as My Dashboard and Favorite.

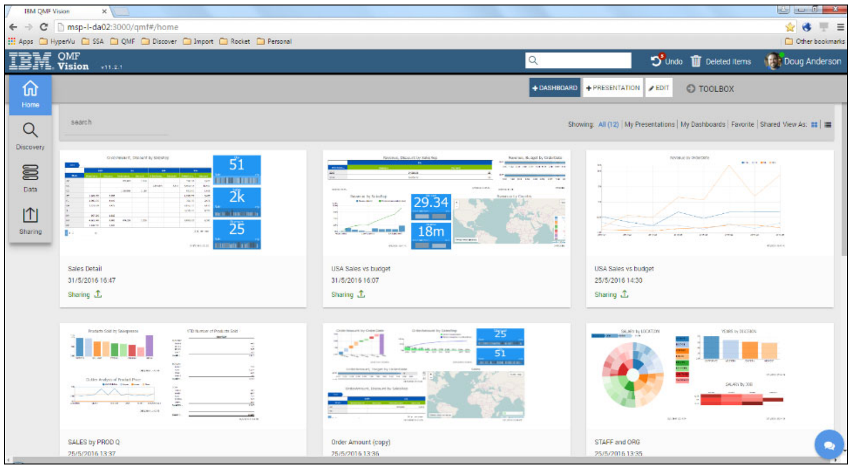


Figure 3-16   Active user home window

3.4.2 Building a dashboard

Many ways are available to build a dashboard. The most common method is to start from the Data window, select your data source or existing query, and build your dashboard.

Note: A query is created within QMF Vision when a dashboard object is created. This process differs from a query that is published to QMF Vision. The query that is published can be looked at more as a data set in this instance. The use of the query that is created in QMF Vision allows data type overrides, calculated columns, and so on. The query that is created in QMF Vision is not stored or available in the QMF Workstation or WebSphere environment.

Follow these steps:

1. Select the data source that you need to work with. Click **View** to browse through the available data by using the connection that is shown in Figure 3-17. The key information is shown to explain the source of data. In the dark section that displays four labeled bubbles, the top bubble displays the name of the connection that is accessed.

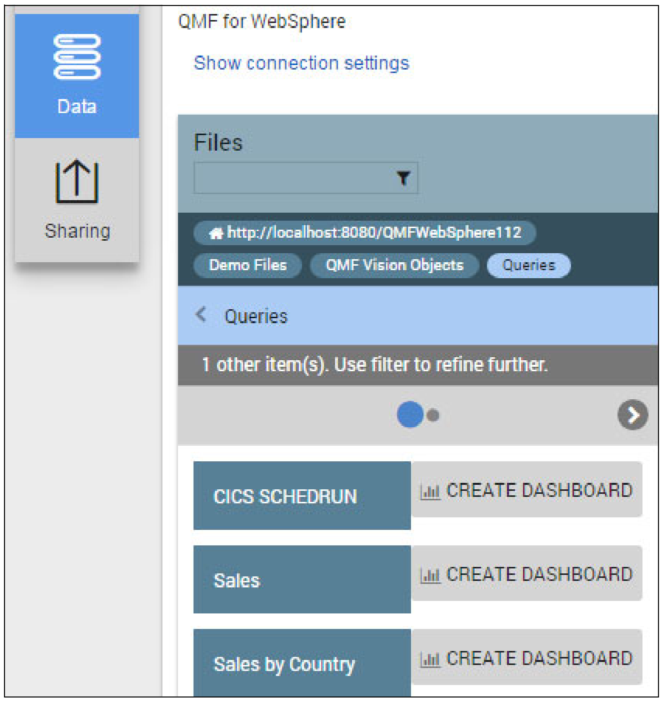


Figure 3-17 Data that is available for dashboards

2. In Figure 3-18, the URL is the name, but this name varies based on the user who created the data source connection and the name that they assigned to the connection.

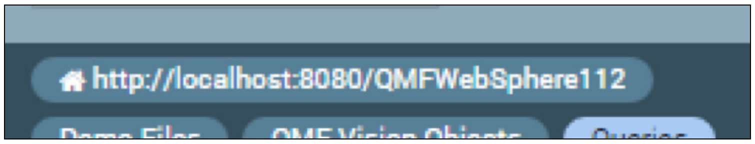


Figure 3-18 Data source connection

3. Immediately under the data source connection is the repository that is accessed. In our example, the repository name is Demo Files, as shown in Figure 3-19.

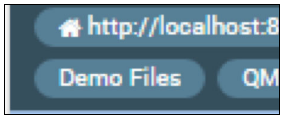


Figure 3-19 Repository connection

4. The label of QMF Vision Objects is the default level for files that are available to the QMF Vision interface. This name is constant through all QMF WebSphere connections. Any remaining bubbles are the folder structure that is created to better organize the accessible queries or tables (Figure 3-20). The folders are created in the Workstation/WebSphere environment and automatically carry over into QMF Vision, as shown in Figure 3-21.

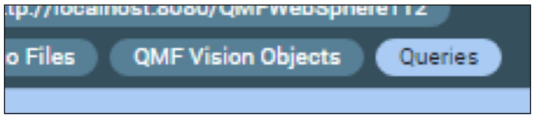
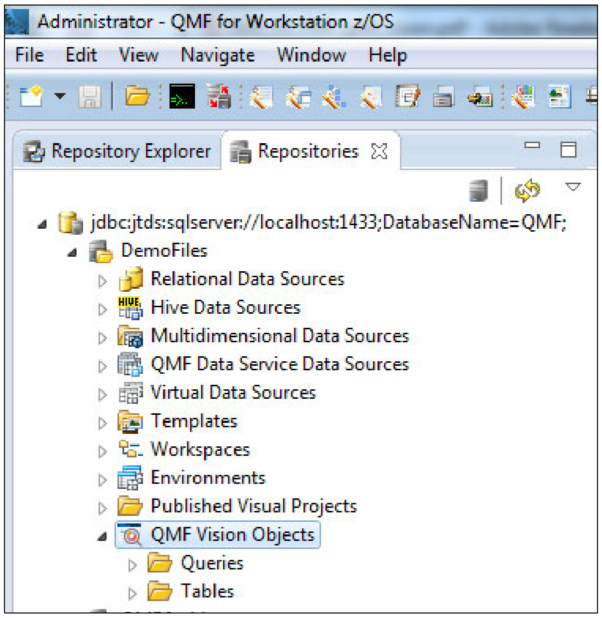


Figure 3-20 Folder structure that shows the selected Queries folder



Administrator - QMF for Workstation z/OS
File Edit View Navigate Window Help
Repository Explorer
Repositories

jdbc:jtds:sqlserver://localhost:1433;DatabaseName=QMF;
- DemoFiles
  - Relational Data Sources
  - Hive Data Sources
  - Multidimensional Data Sources
  - QMF Data Service Data Sources
  - Virtual Data Sources
  - Templates
  - Workspaces
  - Environments
  - Published Visual Projects
  - QMF Vision Objects
    - Queries
    - Tables

Figure 3-21 Folder structure as shown within QMF for Workstation/WebSphere

5. The sample structure is defined in the following manner:
   - Name of connection: `http://localhost:8080/QMFWebSphere112`
   - Repository name: `Demo Files`
   - Default folder level: `QMF Vision Objects`
   - User-created folder: `Queries`

6. After you identify the necessary data, click **Create Dashboard** next to the object to start the dashboard creation process. The window changes to display a column chart with a single data element that is displayed in multiple columns, as shown in Figure 3-22.

Note: An algorithm quickly analyzes the data and selects a data element that allows a column chart to display and present a few columns. You do not need to use this data. This data is offered only to create a visual start position.

Data is analyzed and identified as either a dimension/context column or a measure column. The dimension or context columns are character and date columns. The measure columns are numeric values.

Note: Color labels around the column names are used to quickly identify the column data type that is assigned. Blue is used for dimension/context columns and green is a measure column.

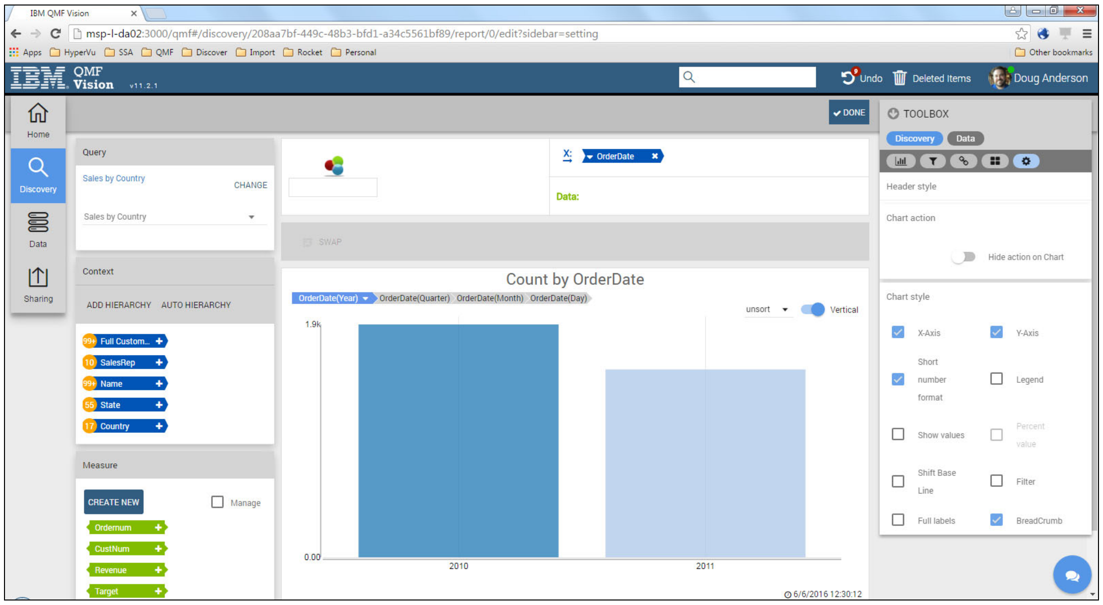


Figure 3-22 Initial view of the dashboard after creation

7. Columns can be added to the chart by clicking the plus sign (+) to the right of the column name. Columns can be removed from the chart by clicking the multiplication sign (x) to the right of the column name if it is used. When a column is added or removed, it automatically moves to the correct area.

   By adding multiple dimension/context columns, you create a drill-down capability. By adding multiple measure columns, you create a drill-across capability.

   Also, you can create a drill-down hierarchy manually or automatically to allow the single-click addition of the hierarchy to the chart. A *hierarchy* is a predefined hierarchical structure of context columns that can be reused. The hierarchies are saved with the underlying query of the chart. If the automatically generated hierarchies are selected, they are created, identifying multiple relationships that can be used in charts. These hierarchies can be removed and edited.

   The manual hierarchy can be created by clicking **ADD HIERARCHY**. They display as shown in Figure 3-23. Click the label of the column to set as the top level of the hierarchy.

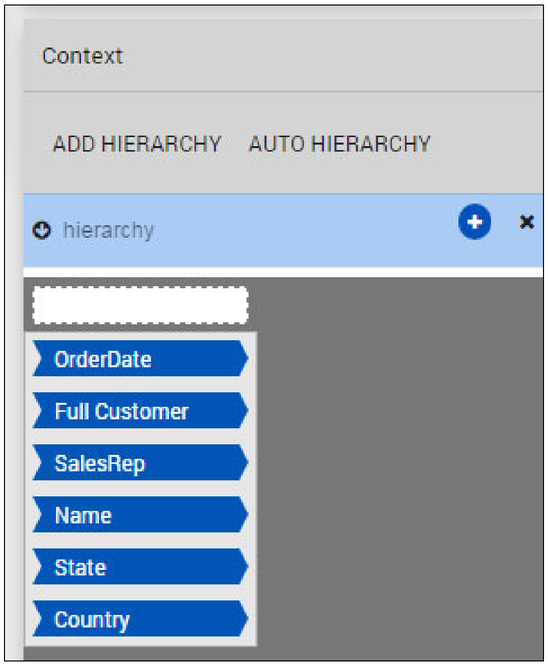


Figure 3-23 Add hierarchy option

8. After you select the top-level column, click in the plain white box to add the next level to the hierarchy and repeat for any additional levels. The name for the hierarchy can also be edited to better identify data. To edit the name, click in the box that is labeled hierarchy, as shown in Figure 3-24. After the hierarchy is completed, you can add the hierarchy to the chart by clicking the blue circle that contains a plus sign (+).

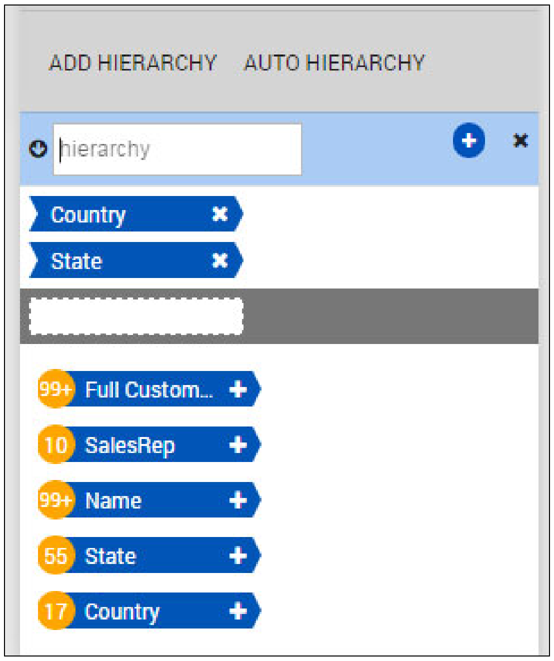


Figure 3-24 Modifying the hierarchy and adding it to the chart

Note: A date column automatically generates a calendar hierarchy to enable views of the data by common periods. If this calendar hierarchy is not completed, the column is not set as a date format. This hierarchy is contained within the column itself and not shown in the hierarchy list.

9. You can alter settings in each chart to customize how the chart looks and behaves. These settings are in the TOOLBOX area and vary based on the chart type. The TOOLBOX is on the right side of the window. The TOOLBOX can be collapsed by clicking the down arrow next to the word TOOLBOX. Figure 3-25 shows the chart settings that are specific to the Column Chart.

Note: The TOOLBOX is a multiple-purpose area for you to use for several interactions in QMF Vision.

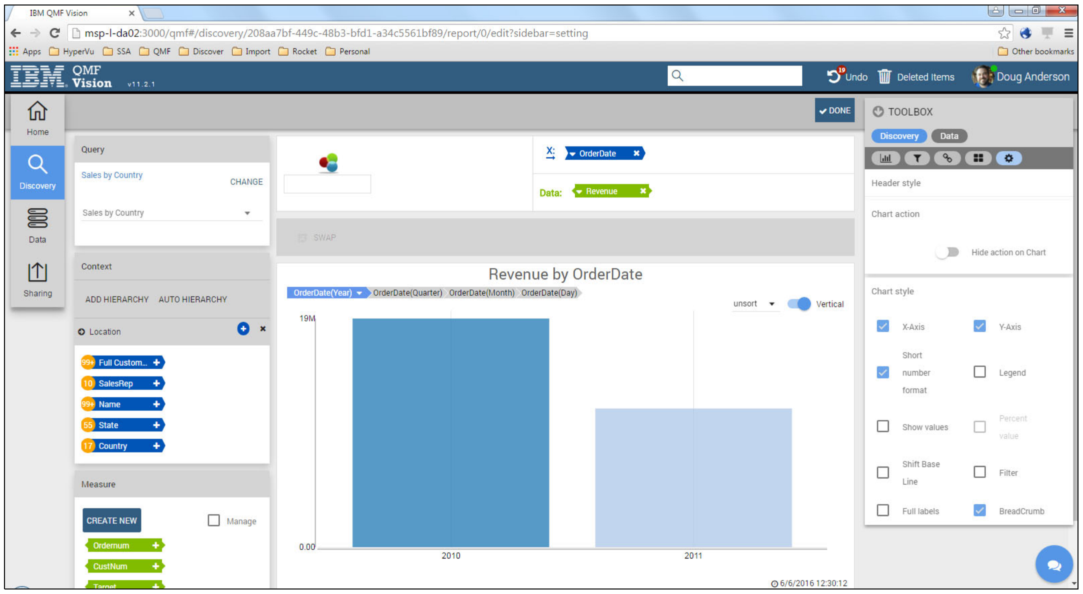


Figure 3-25 Chart setup

10. After the chart is complete, click **DONE** to move to the dashboard Discovery area (Figure 3-26) where the dashboard can be accessed, shared, or modified.

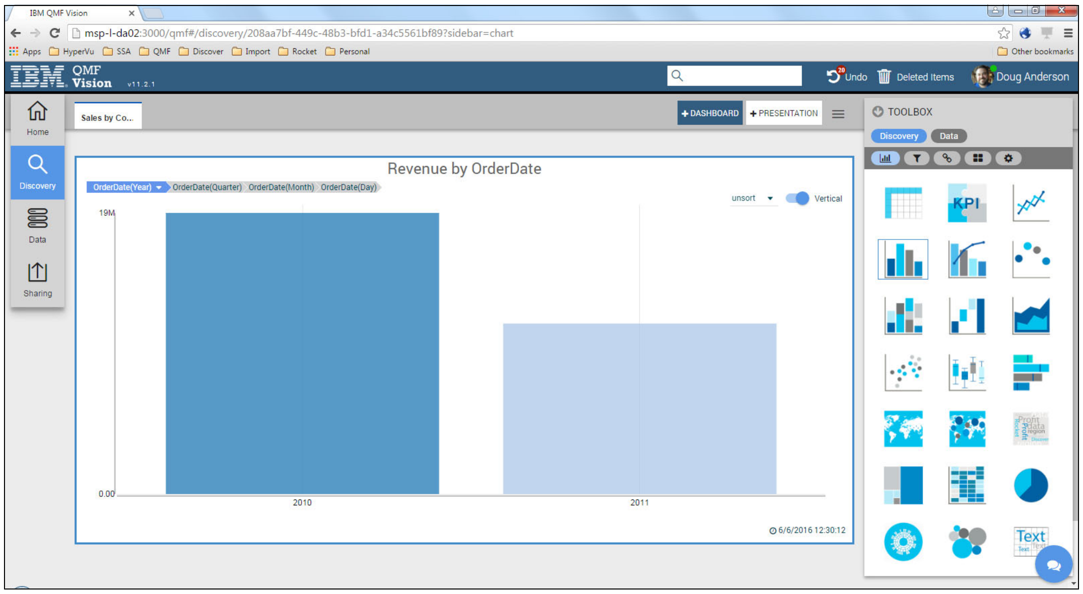


Figure 3-26 Completed chart for dashboard

11. The dashboard can be left as-is or added to by dragging a chart from the TOOLBOX area onto the dashboard, as shown in Figure 3-27. The position of the new chart is shown by the large blue box on the right side, and the existing chart will resize and move location based on the addition of the new chart.

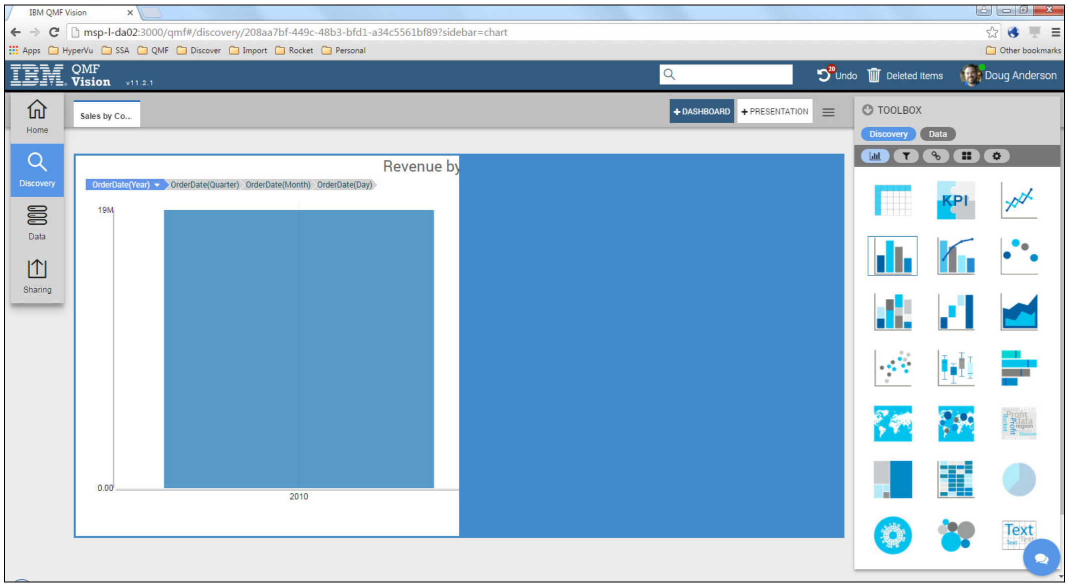


Figure 3-27 Adding another chart to the dashboard

12. After the new chart is added, a data source needs to be selected. Click the cylinders icon in the middle of the new chart view and choose the data source from the TOOLBOX area, as shown in Figure 3-28. Data sources and queries are listed in the TOOLBOX, and the same query can be used multiple times or a new source can be used.

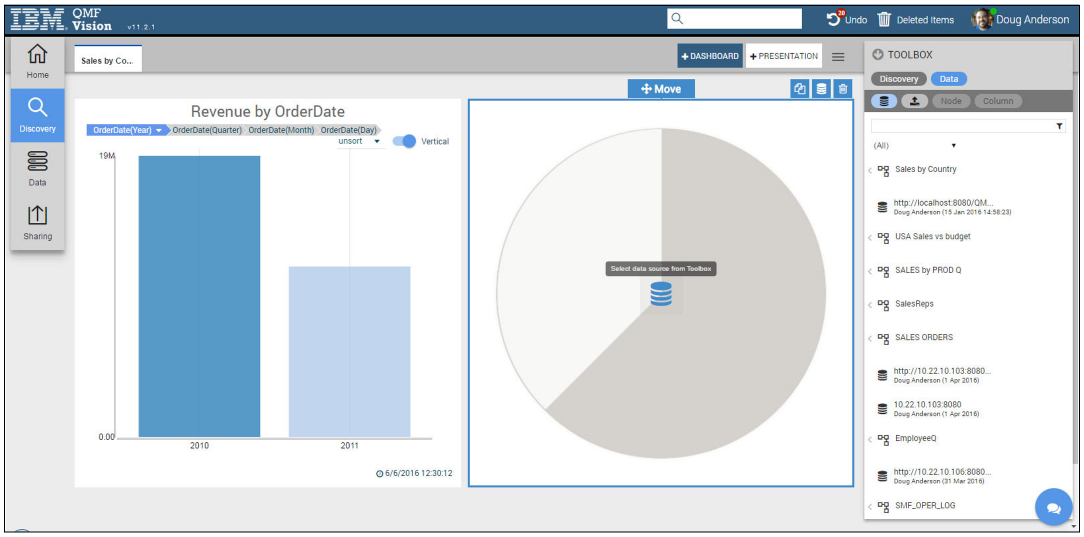


Figure 3-28 Selecting the data source for the new chart

13. By following the creation steps, another chart is added to the dashboard. Additional charts can be added at any time. The sizing and location of each chart can be manually positioned, as needed.

14. You can edit a chart after you create it by hovering over the chart that needs modification and selecting the **EDIT** icon for that chart. See Figure 3-29 and Figure 3-30.

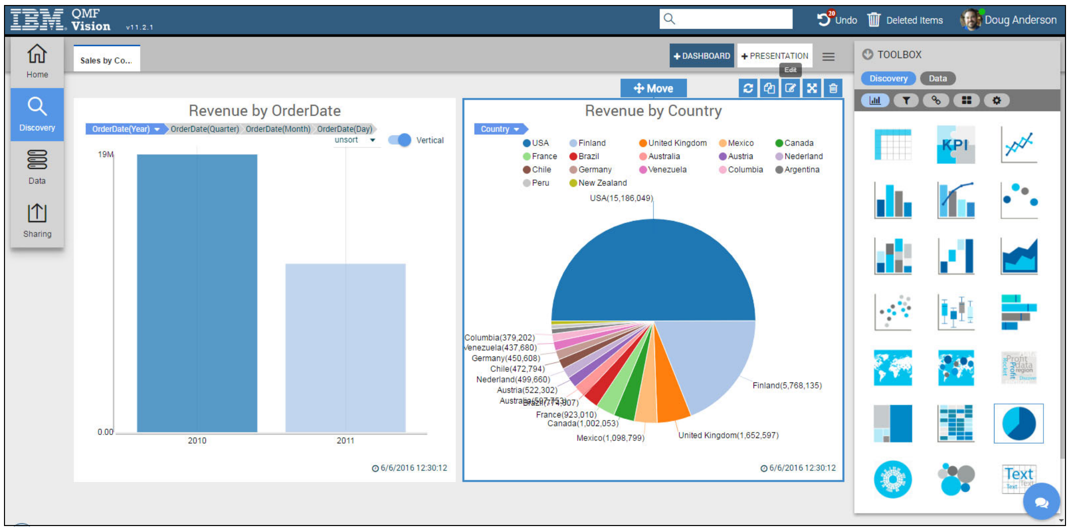


Figure 3-29 Editing a chart on a dashboard

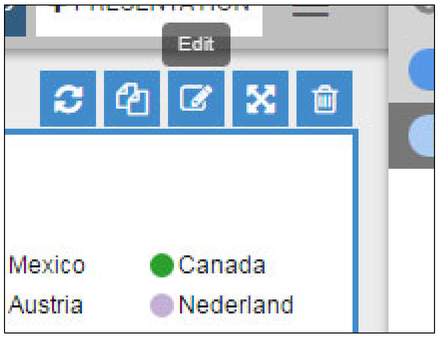


Figure 3-30 Edit icon

15. The TOOLBOX displays automatically so that you can adjust chart settings or change the chart type. Click the new chart type to switch the view. The same data columns are displayed, but they can be modified, if needed. See Figure 3-31, Figure 3-32, and Figure 3-33 on page 66. If an existing chart is not selected when a new chart is selected, the new chart is added to the dashboard as another object.

Note: Certain chart types require different types of data to display correctly, so certain settings and data columns must be changed.

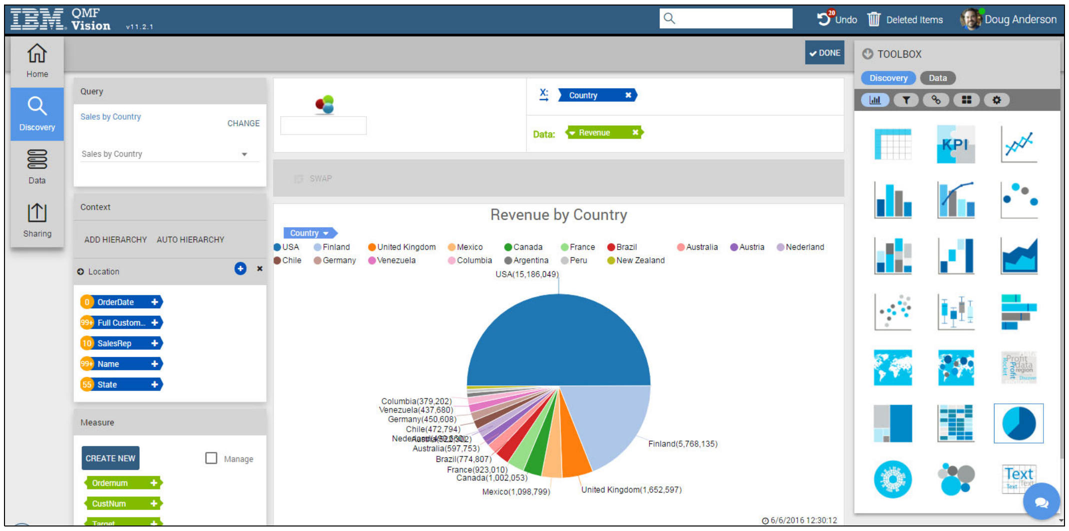


Figure 3-31 Changing the chart

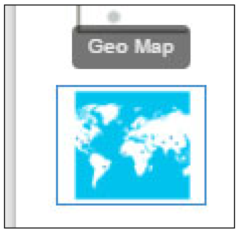


Figure 3-32 Geo Map icon

16. Figure 3-33 shows the Geo Map view.

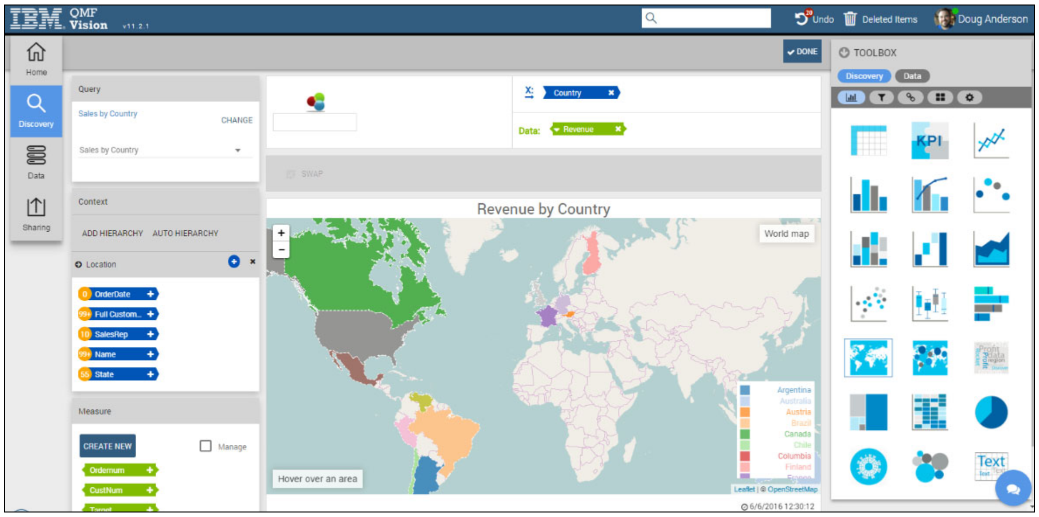


Figure 3-33 Geo Map view

17. The use of a hierarchy creates a drill-down structure that you can add with a single click and reuse in different charts. You do not need to use a hierarchy to create a drill-down scenario. Add multiple context/dimension columns (blue label columns) to create the drill-down capability from a summary level to a detailed level. You can use this drill-down capability for two or more levels. Figure 3-34 shows an example of a GEO MAP chart that highlights summary data at a country level with the ability to drill down to a state level.

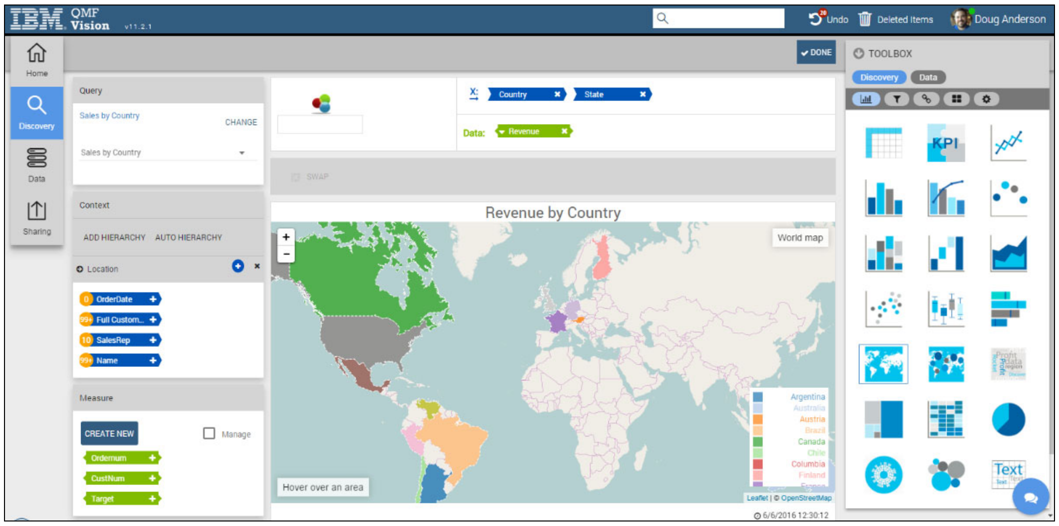


Figure 3-34 Adding drill-down function

18. After a second level is added to the x-axis, the drill-down capability is automatically enabled. You can test the drill capability by double-clicking an area on the chart, as shown in Figure 3-35.

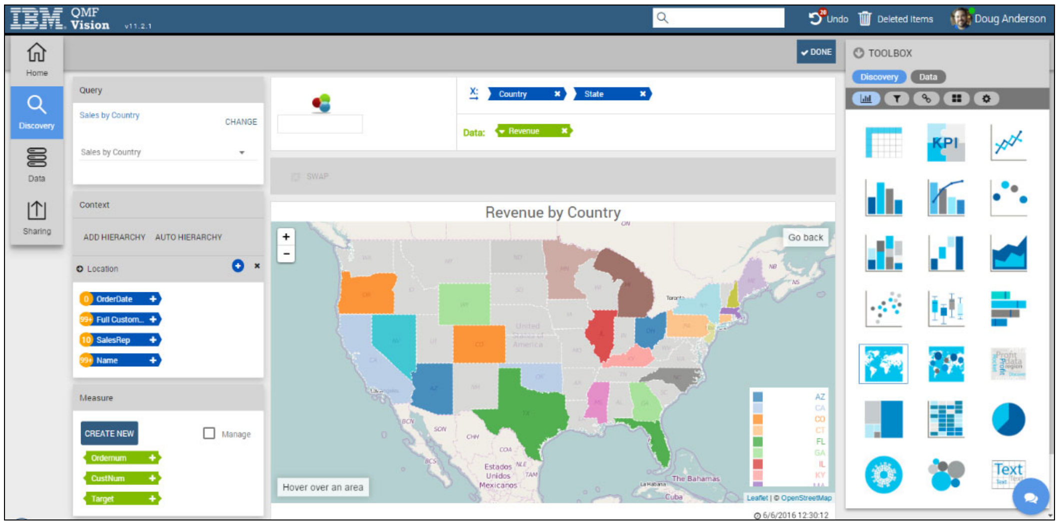


Figure 3-35 Drill-down test

19. After all edits are complete, accept the changes by clicking the blue **DONE** check box in the upper-right corner. This selection returns you to the full dashboard view, as shown in Figure 3-36.

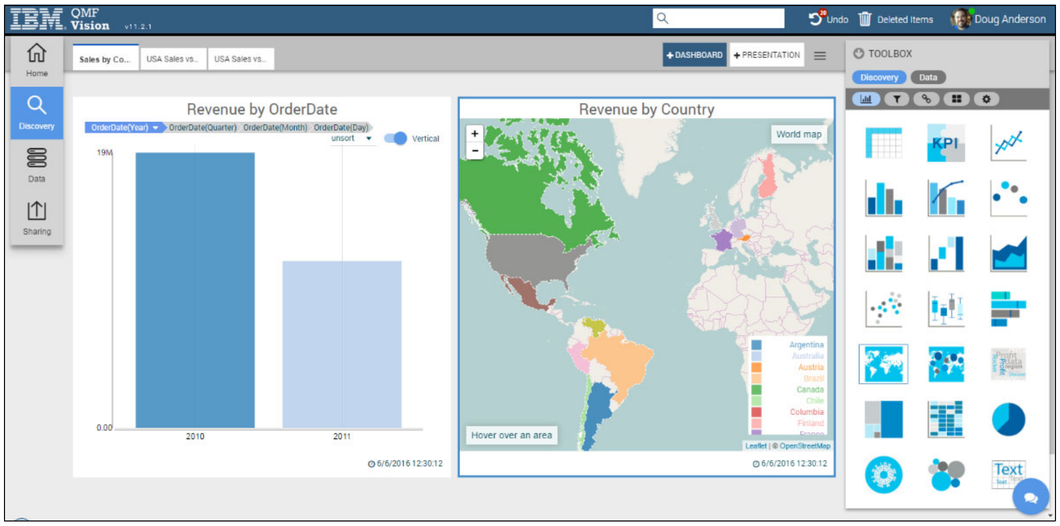


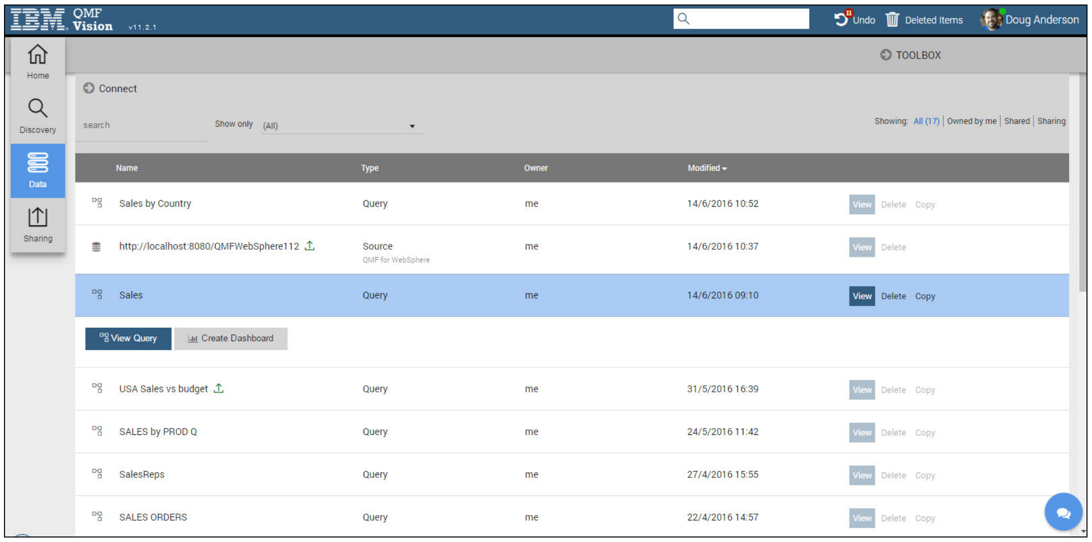
Figure 3-36 Click Done and go back to the full dashboard view

Note: Geo maps do not require any shape files or latitude/longitude coordinates. A column of data is needed with an appropriate label that identifies the corresponding area.

3.4.3 Reports and working with data

A common use question for QMF Vision is reporting in addition to using the dashboards. While several traditional reports can be built within the QMF Vision environment, the reports are still created as a chart on the dashboard. The TABLE chart can be used for report-like views of data. This chart includes several options for customizing the look and feel with the type of interactions that can be incorporated for the users. The process of building a dashboard in 3.4.2, "Building a dashboard" on page 56 started with a data source and the selection of a query that was published to QMF Vision. Follow these steps:

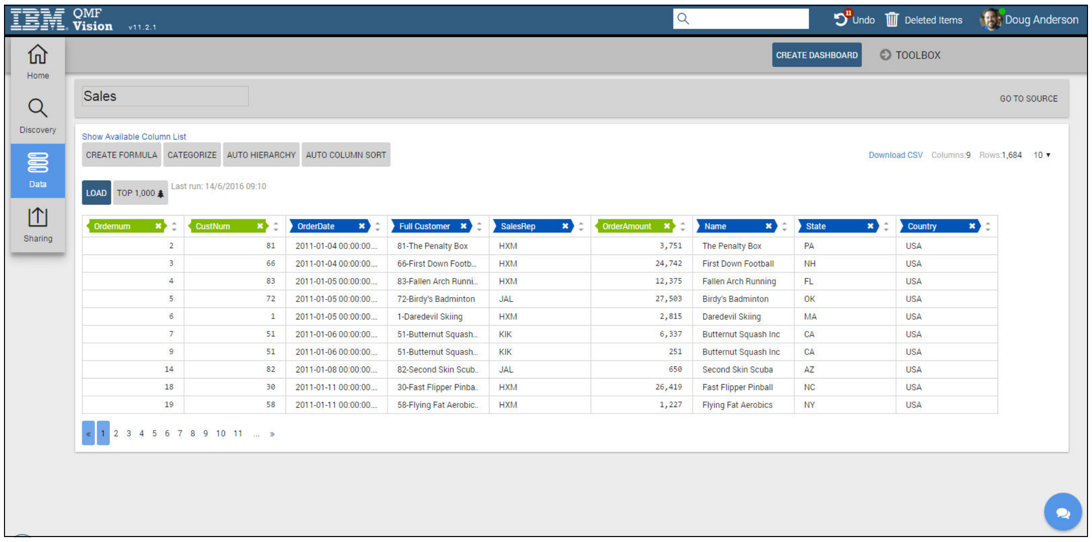
1. A QMF Vision query is automatically created when a new dashboard object is generated. These queries can be selected and reused in multiple charts. See Figure 3-37.



Name	Type	Owner	Modified	
Sales by Country	Query	me	14/6/2016 10:52	View Delete Copy
http://localhost:8080/QMFWebSphere112	Source QMF for WebSphere	me	14/6/2016 10:37	View Delete
Sales	Query	me	14/6/2016 09:10	View Delete Copy
USA Sales vs budget	Query	me	31/5/2016 16:39	View Delete Copy
SALES by PROD Q	Query	me	24/5/2016 11:42	View Delete Copy
SalesReps	Query	me	27/4/2016 15:55	View Delete Copy
SALES ORDERS	Query	me	22/4/2016 14:57	View Delete Copy

Figure 3-37 Accessing a query

2. Figure 3-38 shows the two available methods. The blue VIEW option is available when you hover over the query. The VIEW QUERY and CREATE DASHBOARD options are displayed if you click an area within the query that is not a button. The area expands to show the two additional options. When you click VIEW or VIEW QUERY, the actual data of the query appears.



Ordernum	CustNum	OrderDate	Full Customer	SalesRep	OrderAmount	Name	State	Country
2	81	2011-01-04 00:00:00....	81-The Penalty Box	HXM	3,751	The Penalty Box	PA	USA
3	66	2011-01-04 00:00:00....	66-First Down Footb...	HXM	24,742	First Down Football	NH	USA
4	83	2011-01-05 00:00:00....	83-Fallen Arch Runni...	HXM	12,375	Fallen Arch Running	FL	USA
5	72	2011-01-05 00:00:00....	72-Birdy's Badminton	JAL	27,503	Birdy's Badminton	OK	USA
6	1	2011-01-05 00:00:00....	1-Daredevil Skiing	HXM	2,815	Daredevil Skiing	MA	USA
7	51	2011-01-06 00:00:00....	51-Butternut Squash...	KIK	6,337	Butternut Squash Inc	CA	USA
9	51	2011-01-06 00:00:00....	51-Butternut Squash...	KIK	251	Butternut Squash Inc	CA	USA
14	82	2011-01-08 00:00:00....	82-Second Skin Scub...	JAL	650	Second Skin Scuba	AZ	USA
18	30	2011-01-11 00:00:00....	30-Fast Flipper Pinba...	HXM	26,419	Fast Flipper Pinball	NC	USA
19	58	2011-01-11 00:00:00....	58-Flying Fat Aerobic...	HXM	1,227	Flying Fat Aerobics	NY	USA

Figure 3-38 Query view

3. By default, the first 10 rows of data are shown, but you can change the number of rows. Near the upper-right corner of the Data area (Figure 3-39), a drop-down arrow is displayed to change the number of rows. A summary shows the total number of columns and rows. You can use the remaining option to download the full data set into a comma-separated values (CSV) file.

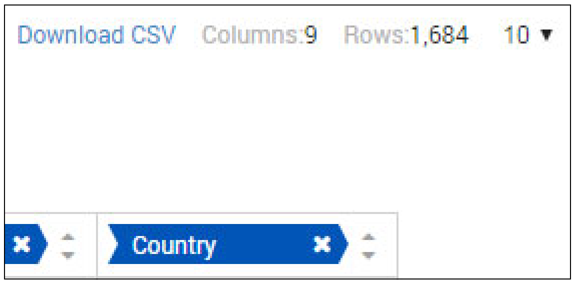
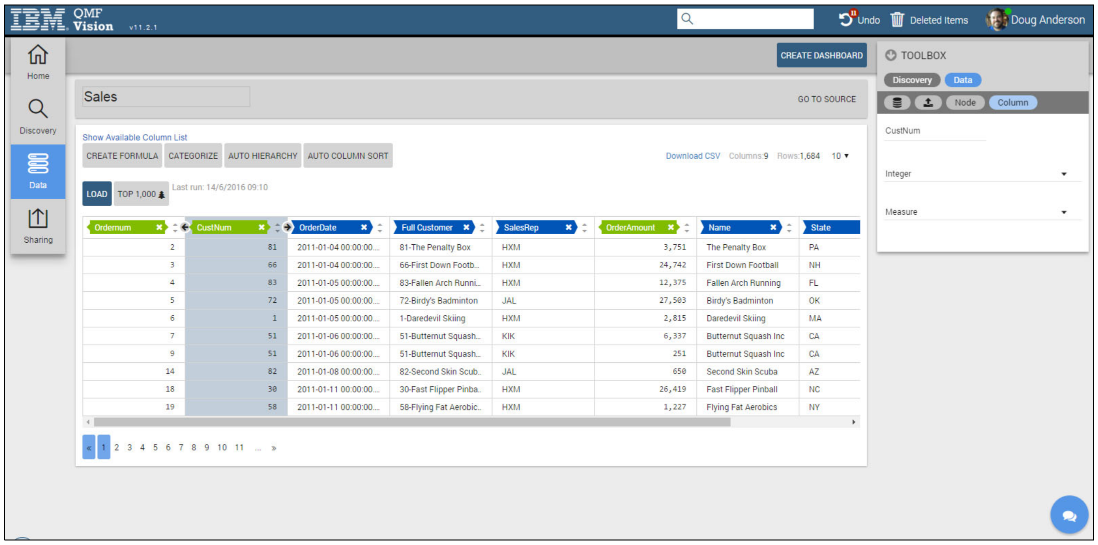


Figure 3-39 Query view

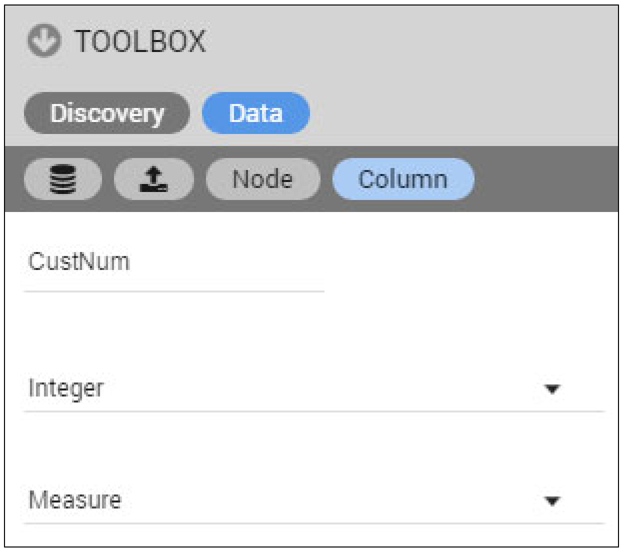
4. Data is analyzed and assigned as either a dimension/context or a measure type. The dimension or context columns are character and date columns. The measure columns are numeric values. You can override the assignments so that data can be used in a different manner than the original interpretation. Selecting a column both highlights it and shows the column properties in the TOOLBOX, as shown in Figure 3-40.



Ordernum	CustNum	OrderDate	Full Customer	SalesRep	OrderAmount	Name	State
2	81	2011-01-04 00:00:00....	81-The Penalty Box	HXM	3,751	The Penalty Box	PA
3	66	2011-01-04 00:00:00....	66-First Down Footb...	HXM	24,742	First Down Football	NH
4	83	2011-01-05 00:00:00....	83-Fallen Arch Runni...	HXM	12,375	Fallen Arch Running	FL
5	72	2011-01-05 00:00:00....	72-Birdy's Badminton	JAL	27,503	Birdy's Badminton	OK
6	1	2011-01-05 00:00:00....	1-Daredevil Skiing	HXM	2,815	Daredevil Skiing	MA
7	51	2011-01-06 00:00:00....	51-Butternut Squash...	KIK	6,337	Butternut Squash Inc	CA
9	51	2011-01-06 00:00:00....	51-Butternut Squash...	KIK	251	Butternut Squash Inc	CA
14	82	2011-01-08 00:00:00....	82-Second Skin Scub...	JAL	650	Second Skin Scuba	AZ
18	30	2011-01-11 00:00:00....	30-Fast Flipper Pinba...	HXM	26,419	Fast Flipper Pinball	NC
19	58	2011-01-11 00:00:00....	58-Flying Fat Aerobic...	HXM	1,227	Flying Fat Aerobics	NY

Figure 3-40 Changing the data type

Figure 3-41 shows the data type identifiers.



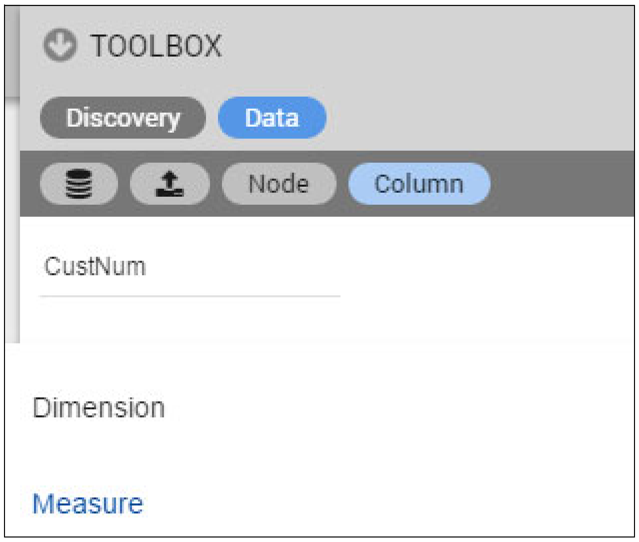
CustNum

Integer

Measure

Figure 3-41 Data type identifiers

5. For this data set, the CustNum column needs to be changed from a measure to a dimension so that the data can be used differently in charts. After the column is selected, you can change the value of Measure to Dimension. Selecting the value that you want populates the change in the data automatically and the column heading changes colors. Figure 3-42 shows this process.



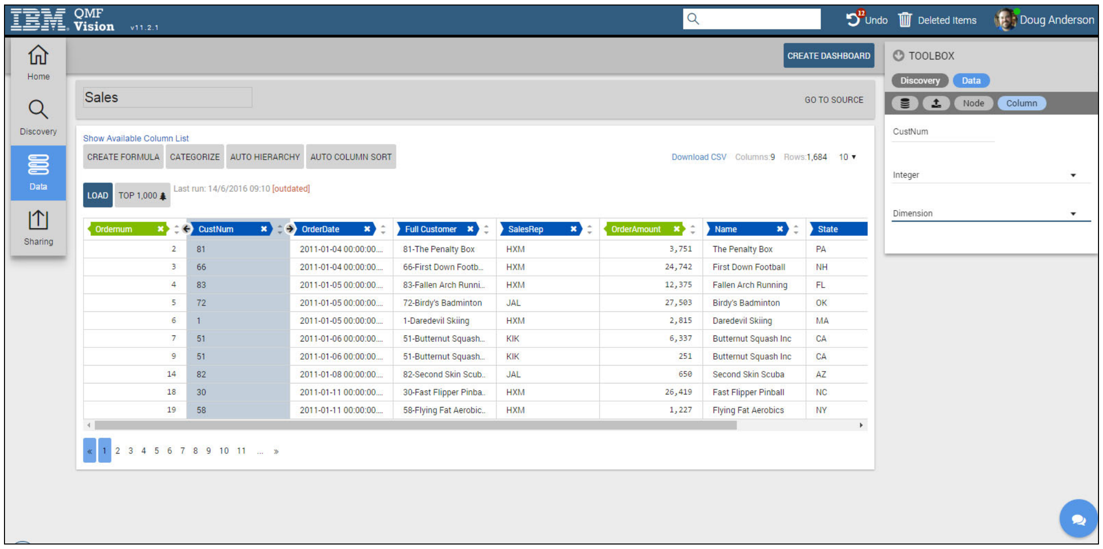
CustNum

Dimension

Measure

Figure 3-42 Drop-down list to change the data type

Figure 3-43 shows the changed column.



Ordernum	CustNum	OrderDate	Full Customer	SalesRep	OrderAmount	Name	State
2	81	2011-01-04 00:00:00....	81-The Penalty Box	HXM	3,751	The Penalty Box	PA
3	66	2011-01-04 00:00:00....	66-First Down Footb...	HXM	24,742	First Down Football	NH
4	83	2011-01-05 00:00:00....	83-Fallen Arch Runni...	HXM	12,375	Fallen Arch Running	FL
5	72	2011-01-05 00:00:00....	72-Birdy's Badminton	JAL	27,503	Birdy's Badminton	OK
6	1	2011-01-05 00:00:00....	1-Daredevil Skiing	HXM	2,815	Daredevil Skiing	MA
7	51	2011-01-06 00:00:00....	51-Butternut Squash...	KIK	6,337	Butternut Squash Inc	CA
9	51	2011-01-06 00:00:00....	51-Butternut Squash...	KIK	251	Butternut Squash Inc	CA
14	82	2011-01-08 00:00:00....	82-Second Skin Scub..	JAL	650	Second Skin Scuba	AZ
18	30	2011-01-11 00:00:00....	30-Fast Flipper Pinba..	HXM	26,419	Fast Flipper Pinball	NC
19	58	2011-01-11 00:00:00....	58-Flying Fat Aerobic..	HXM	1,227	Flying Fat Aerobics	NY

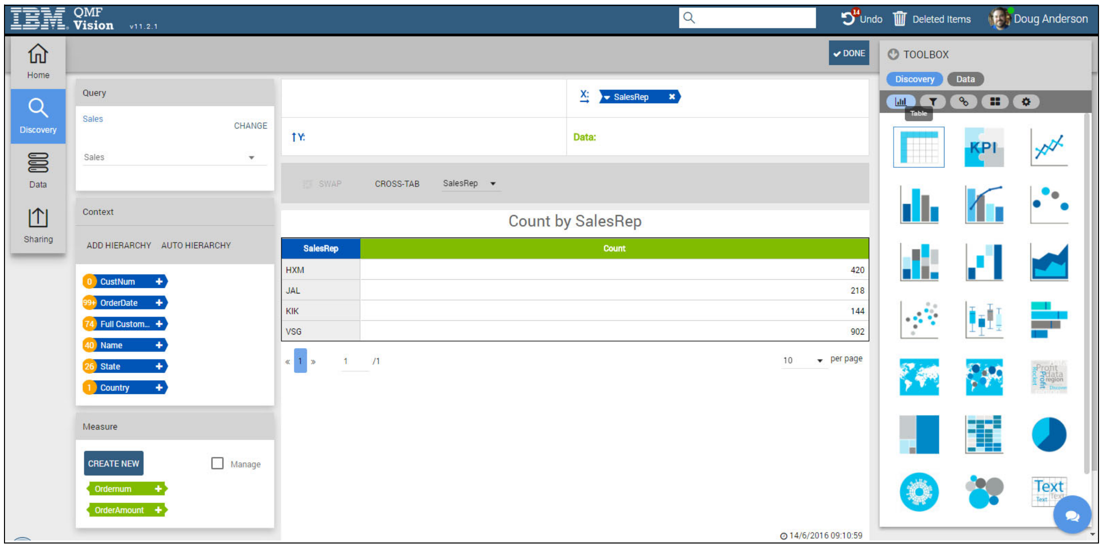
Last run: 14/6/2016 09:10 [outdated]

Download CSV Columns:9 Rows:1,684

Figure 3-43   Data type is changed

6. You can add formula columns and categorizations while you view the query. For more information, see *Getting Started with DB2 QMF Vision*, GC27-8805-01.

   After you set the data to the format that you need, start the build process by clicking **CREATE DASHBOARD** in the upper-right part of the window. You can change the initial display of a column chart to TABLE CHART for the report view that is shown in Figure 3-44.

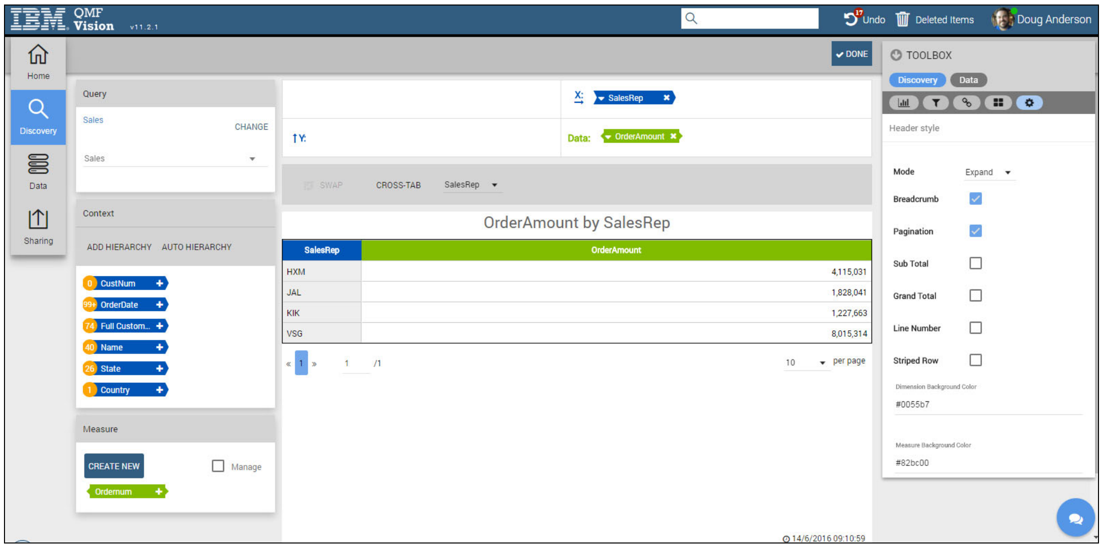


Count by SalesRep

SalesRep	Count
HXM	420
JAL	218
KIK	144
VSG	902

Figure 3-44   Table chart to start the report view

7. After you change the chart type, add the measure (Figure 3-45) to modify the output. You can add multiple measures to display additional columns of values. If a single dimension is used, it is shown as the first column of data. The corresponding values are displayed to the right.



OrderAmount by SalesRep

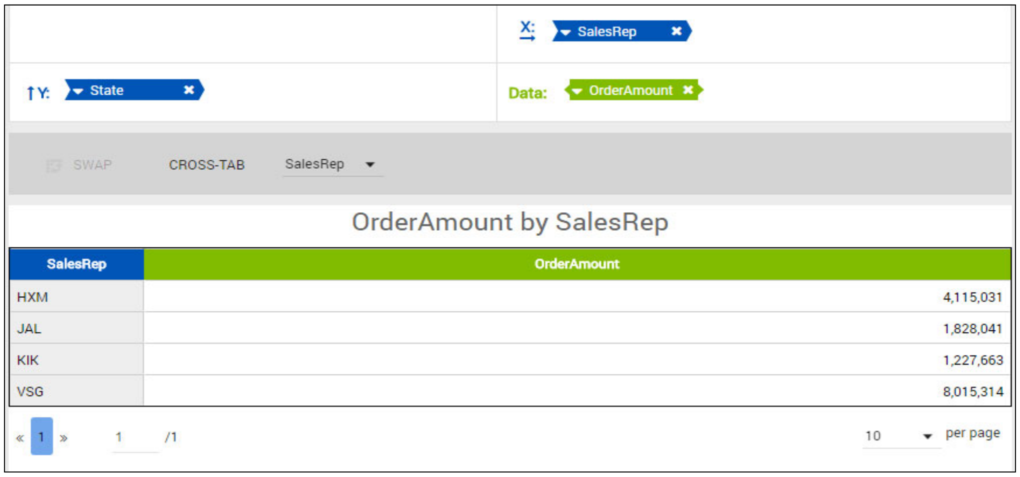
SalesRep	OrderAmount
HXM	4,115,031
JAL	1,828,041
KIK	1,227,663
VSG	8,015,314

« 1 » 1 /1 10 per page

Mode	Expand
Breadcrumb	✓
Pagination	✓
Sub Total	☐
Grand Total	☐
Line Number	☐
Striped Row	☐
Dimension Background Color	#0055b7
Measure Background Color	#82bc00

Figure 3-45 Adding a measure

8. Adding a dimension value to the y-axis does not automatically change the data that is displayed, but it enables additional functionality. Figure 3-46 shows how the addition of State to the y-axis does not change the data that is shown.



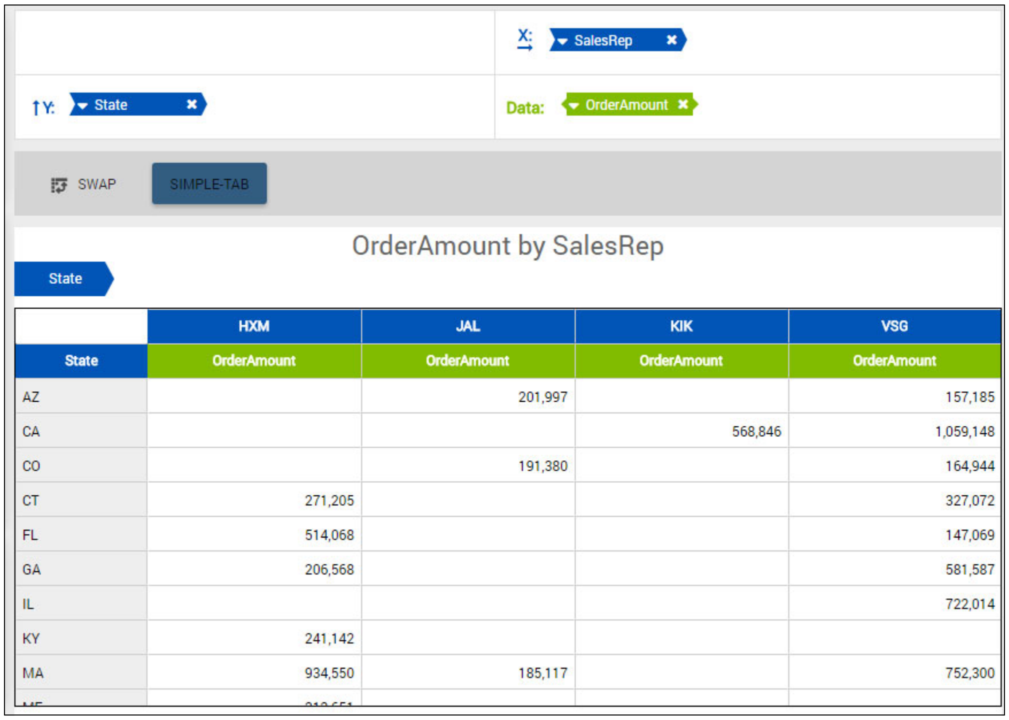
OrderAmount by SalesRep

SalesRep	OrderAmount
HXM	4,115,031
JAL	1,828,041
KIK	1,227,663
VSG	8,015,314

« 1 » 1 /1 10 per page

Figure 3-46 Setting the y-axis

9. After you add the y-axis, Figure 3-47 shows the cross tabulation (crosstab) feature, which is in the gray bar. Click **CROSS-TAB** to highlight the function, which changes to SIMPLE-TAB. Use this crosstab function to revert to the original view, if needed. The cross-tab moved the x-axis column data to the columns and summarized the measures according to the y-axis (rows). You can use multiple columns for the y-axis but only by using a hierarchy. This capability adds another level of measure breakdown, which is useful when you add subtotals to a report.

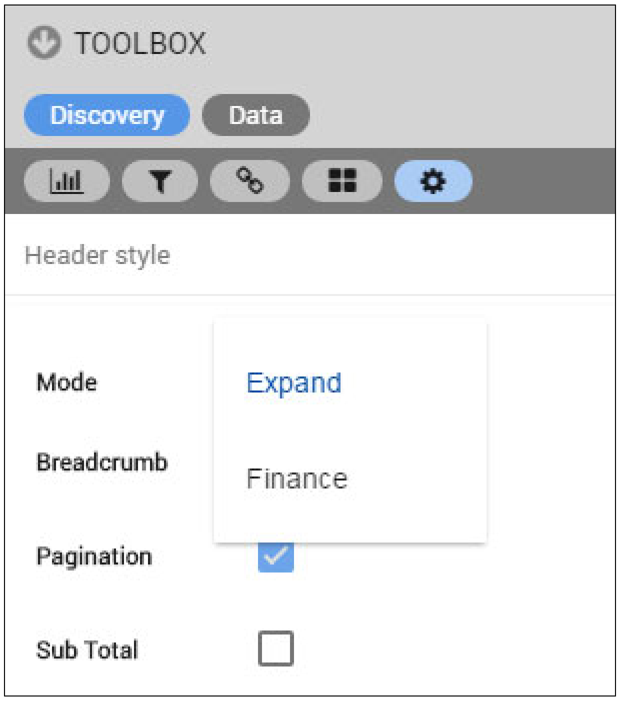


OrderAmount by SalesRep

	HXM	JAL	KIK	VSG
State	OrderAmount	OrderAmount	OrderAmount	OrderAmount
AZ		201,997		157,185
CA			568,846	1,059,148
CO		191,380		164,944
CT	271,205			327,072
FL	514,068			147,069
GA	206,568			581,587
IL				722,014
KY	241,142			
MA	934,550	185,117		752,300

Figure 3-47 Crosstab

10. Within the TOOLBOX, you can use the available options (Figure 3-48) to set the report type as static or expanding, set subtotals, enable striped rows, and more.



Mode: Expand / Finance

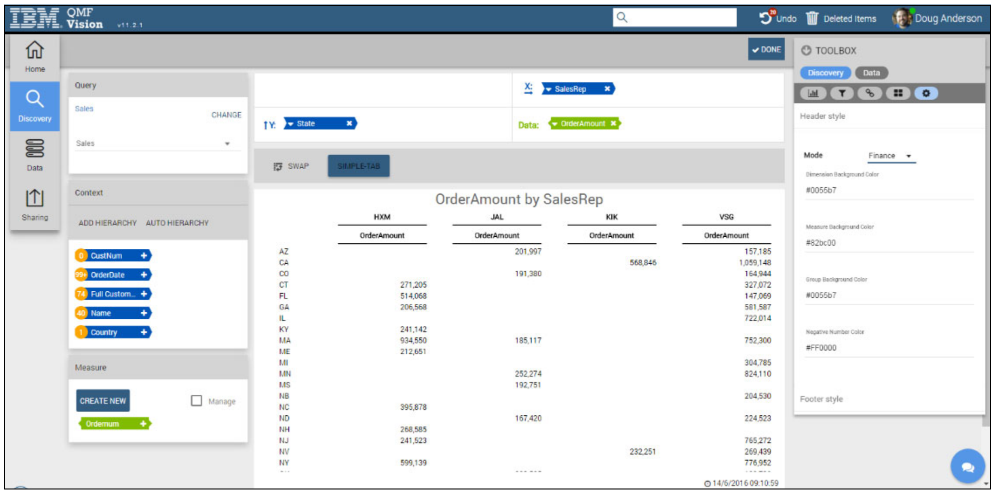
Breadcrumb

Pagination ✓

Sub Total ☐

Figure 3-48   Report format

11. By changing the mode of the report from Expand to Finance, you change the look of the view to a financial report, as shown in Figure 3-49.



Mode: Finance

Dimension Background Color: #0055b7

Measure Background Color: #82bc00

Group Background Color: #0055b7

Negative Number Color: #FF0000

OrderAmount by SalesRep

	HXM	JAL	KIK	VSG
	OrderAmount	OrderAmount	OrderAmount	OrderAmount
AZ		201,997		157,185
CA			568,846	1,059,148
CO		191,380		164,944
CT	271,205			327,072
FL	514,068			147,069
GA	206,568			581,587
IL				722,014
KY	241,142			
MA	934,550	185,117		752,300
ME	212,651			
MI				304,785
MN		252,274		824,110
MS		192,751		
NB				204,530
NC	395,878			
ND		167,420		224,523
NH	268,585			
NJ	241,523			765,272
NV			232,251	269,439
NY	599,139			776,952

14/6/2016 09:10:59

Figure 3-49   Finance report

3.4.4 Editing or replacing a query

After you create a chart, you can modify it. Another common question is whether a query that is in use with a chart can be edited or replaced? The answer is yes.

After the chart is in edit mode, the name of the query shows in the upper-right area of the workspace. Figure 3-50 shows a query that is named Sales. By clicking the name, you can edit the query.

Important: Any other charts that use the same query can be affected by your changes when you edit a query. For independence from other objects, create a duplicate query to replace the current query and then perform the edits.

Clicking **CHANGE** opens other sources and queries in the TOOLBOX. Use this function to select a replacement query. However, you must reestablish the data columns that are used in the chart.

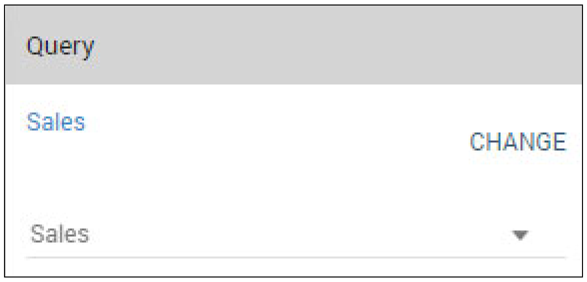


Figure 3-50 Edit query

3.4.5 Coordinated highlighting and associations

A common request for dashboards is for the capability to control secondary objects when you click a specific chart. You can pass a value from one object to another object. If a dashboard object was built by using the same underlying query, this functionality is enabled, by default.

If you do not want to use this function, you can turn off the function by disabling **Coordinate Highlighting** in the settings in the TOOLBOX for a specific dashboard, as shown in Figure 3-51.

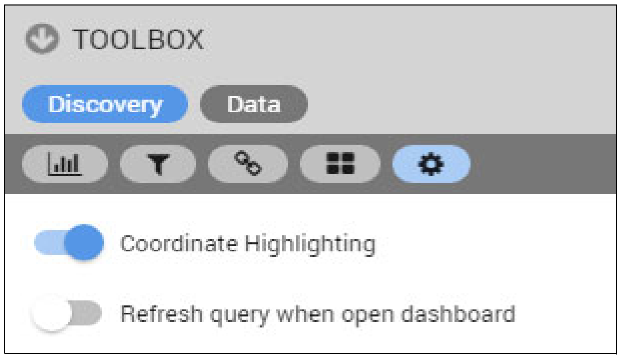


Figure 3-51 Coordinated Highlighting function

A change to this setting is applied only to the dashboard that is open and under construction. This capability provides individual settings for each dashboard so that you use the functionality that you want on a case-by-case basis.

On a dashboard with enabled coordinated highlighting, a single click to a column or pie wedge captures the value that is selected and passes that value to other objects with the associated value and column name. The value is highlighted in the lower-left corner of the object that is coordinated (Figure 3-52). If this setting is disabled, each object is independent of each other and unable to pass values between them.

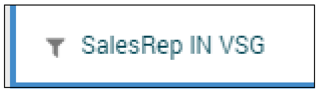


Figure 3-52 Coordinated value

Most dashboards consist of different underlying queries and therefore are not automatically able to pass values from one object to another. You can pass values from one object to another by using the ASSOCIATION (Figure 3-53) feature from the TOOLBOX for a specific dashboard.

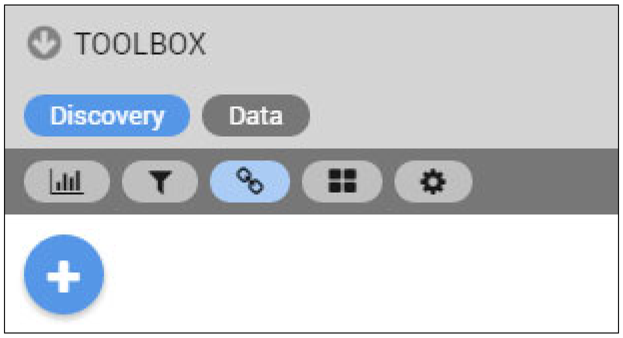


Figure 3-53 Association

To create an association, click the plus sign (**+**) in the blue circle, as shown in Figure 3-54.

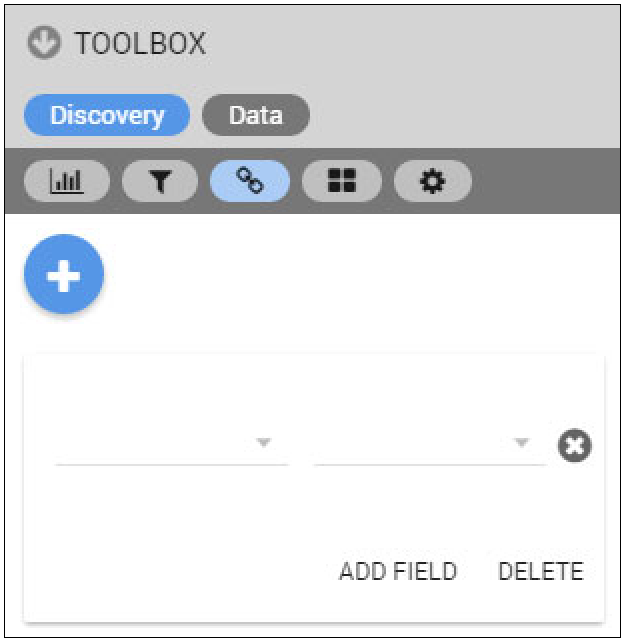


Figure 3-54 Create an association

Use the first drop-down box to select the query. Use the second drop-down box to the right of the query to select the column to be part of the association. After you complete the first two selections, click **ADD FIELD** to create the next set of drop-down selections. Choose the next query and column with data in common with the first column selection (Figure 3-55). You can repeat this process, if needed, for additional queries in the dashboard that contain the same data columns.

If another association exists, you can repeat the process by clicking the plus sign (**+**) again to create another set of associations and leave the previous associations in place.

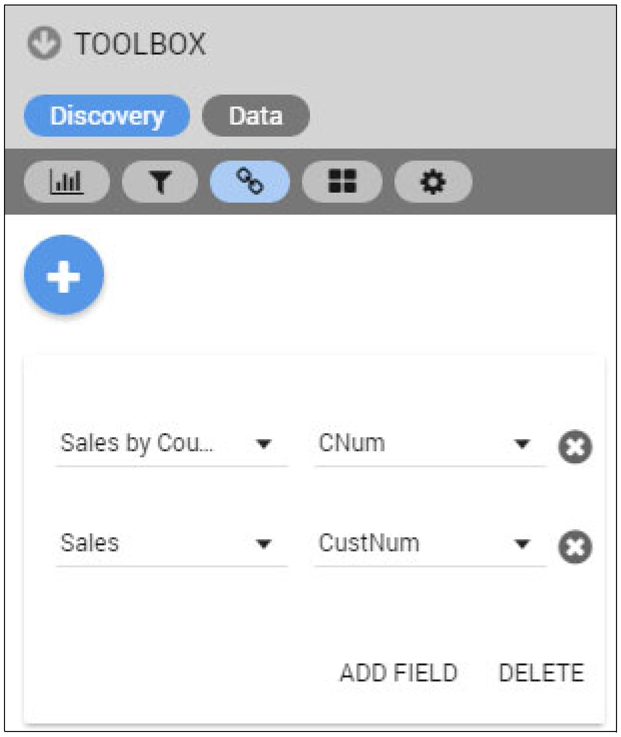


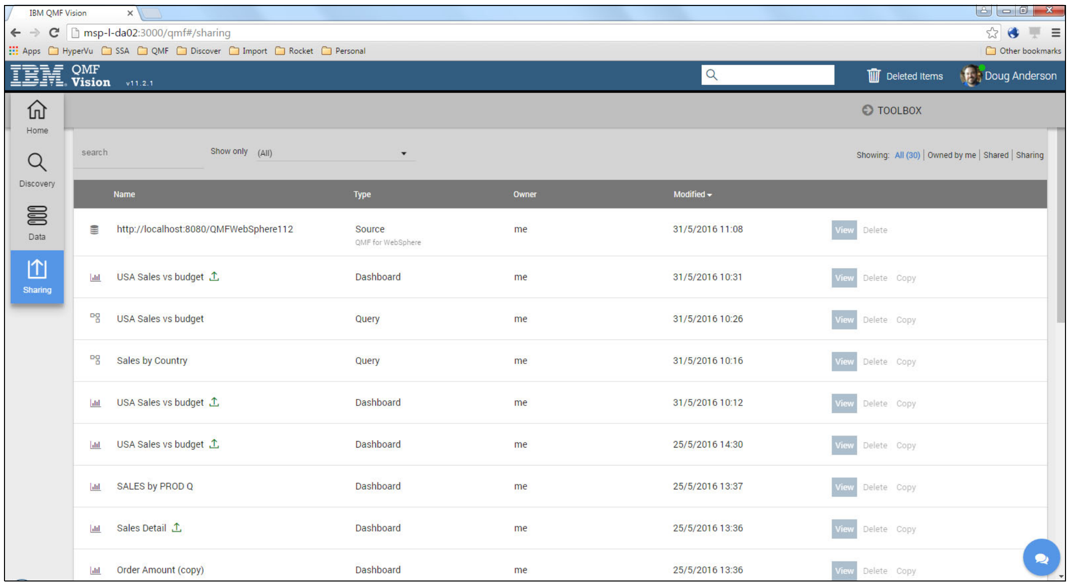
Figure 3-55 Completed association

3.5 Collaboration

This section describes collaborative capabilities by using shared data sources and built-in messaging.

3.5.1 Sharing data sources, queries, and dashboards

Each user has their own objects within QMF Vision, and, by default, no one else can see or access them. An object can be shared with individual users and groups of users. After an object is shared, it is available to other users. A shared dashboard shows on their home window. A shared data source and query are listed in their Data area, as shown in Figure 3-56.


IBM QMF Vision
msp-l-da02:3000/qmf#/sharing
Apps HyperVu SSA QMF Discover Import Rocket Personal Other bookmarks
IBM QMF Vision v11.2.1
Deleted Items
Doug Anderson
Home
Discovery
Data
Sharing
TOOLBOX
search
Show only (All)
Showing: All (30) | Owned by me | Shared | Sharing

Name	Type	Owner	Modified	
http://localhost:8080/QMFWebSphere112	Source QMF for WebSphere	me	31/5/2016 11:08	View Delete
USA Sales vs budget	Dashboard	me	31/5/2016 10:31	View Delete Copy
USA Sales vs budget	Query	me	31/5/2016 10:26	View Delete Copy
Sales by Country	Query	me	31/5/2016 10:16	View Delete Copy
USA Sales vs budget	Dashboard	me	31/5/2016 10:12	View Delete Copy
USA Sales vs budget	Dashboard	me	25/5/2016 14:30	View Delete Copy
SALES by PROD Q	Dashboard	me	25/5/2016 13:37	View Delete Copy
Sales Detail	Dashboard	me	25/5/2016 13:36	View Delete Copy
Order Amount (copy)	Dashboard	me	25/5/2016 13:36	View Delete Copy

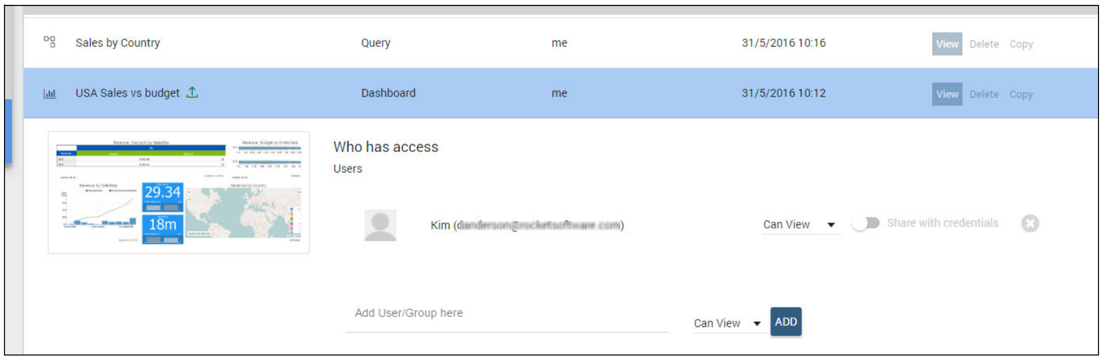
Figure 3-56 Sharing

The following icons are described:

- ↓ indicates that this object is shared with me by other users.
- ↑ indicates that this object is shared by me with other users.

Different permissions are available, depending on the object that is shared. All objects need to be designated if the user or group can view or edit.

Additionally, with secured data sources, you can elect to share with the object with credentials or the user can enter their own credentials, as shown in Figure 3-57. Sharing can also be removed by clicking the circle with an x next to the user or group that you are sharing with.



Sales by Country	Query	me	31/5/2016 10:16
USA Sales vs budget	Dashboard	me	31/5/2016 10:12

Who has access

Users

Kim ([redacted])

Figure 3-57 Sharing permissions

Dashboards can be shared from your home window, as well. Hovering over the lower-right corner of a dashboard tile shows four icons. Each icon identifies the action while you hover over the icon. Figure 3-58 shows the icon for sharing the dashboard with other users.

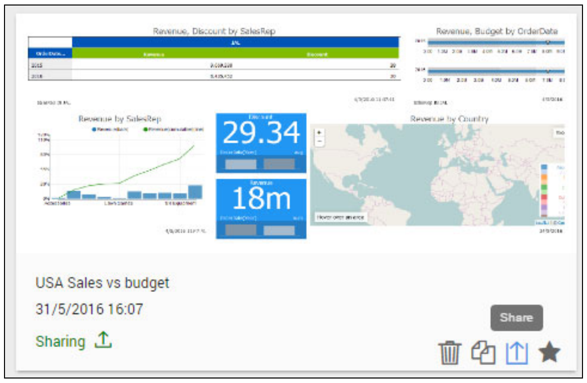


Figure 3-58 Sharing from the dashboard tile

3.5.2 Built-in messaging

A chat capability is built into QMF Vision so that users can communicate with each other directly in the software without requiring the integration of a third-party tool. Messages can be sent back and forth to individuals or groups. A blue circle icon displays in the lower-right corner of the window (Figure 3-59). Click the icon to start the chat process.

Figure 3-59 Chat icon

If a message is sent to a user who is not online, a message indicator displays on their next login to alert the user to new unread messages, as shown in Figure 3-60.

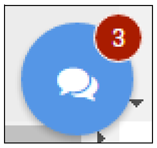
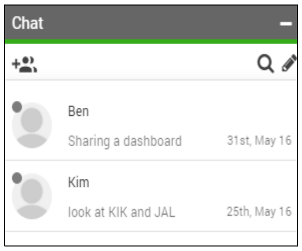


Figure 3-60 Unread message indicator

Clicking the chat icon displays a dialog box (Figure 3-61) so that you can select a message recipient. Any previous conversations display so that you can reengage where the previous message left off.

Start a new conversation by clicking the icon that shows two people with a plus sign (+) sign. The dialog box changes to show a list of users to select from or you can use the search option to find a user based on the characters that are entered.



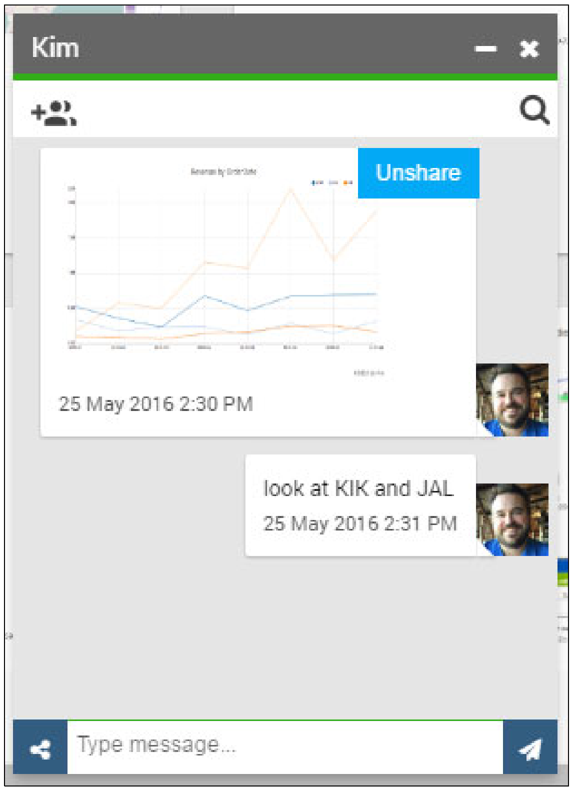
Chat

Ben
Sharing a dashboard 31st, May 16

Kim
look at KIK and JAL 25th, May 16

Figure 3-61 Dialog box to find a message recipient

If you open a previous conversation, the messages and any shared dashboard tiles display, as shown in Figure 3-62.



Kim

Unshare

25 May 2016 2:30 PM

look at KIK and JAL
25 May 2016 2:31 PM

Type message...

Figure 3-62 Open conversation

You can type messages in the dialog box and send them to the recipients. A dashboard can be dragged into the conversation from the home window, as shown in Figure 3-63.

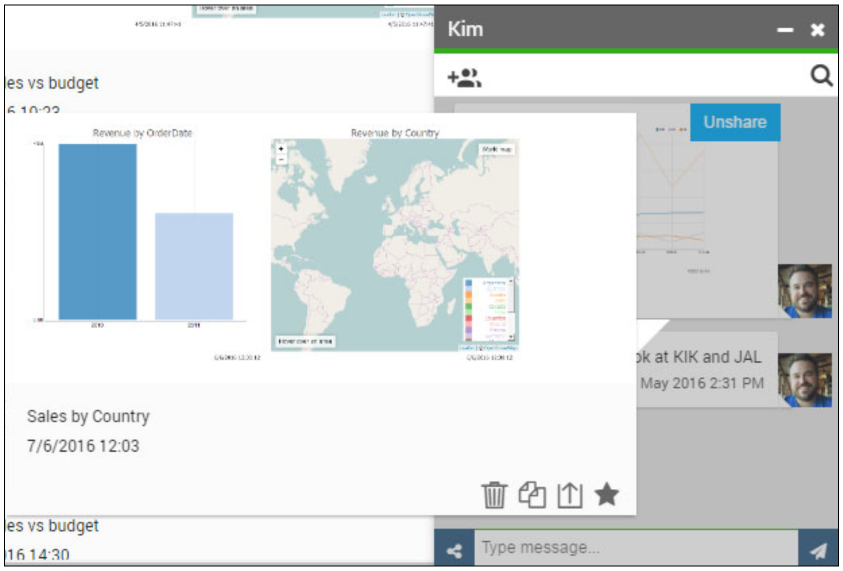
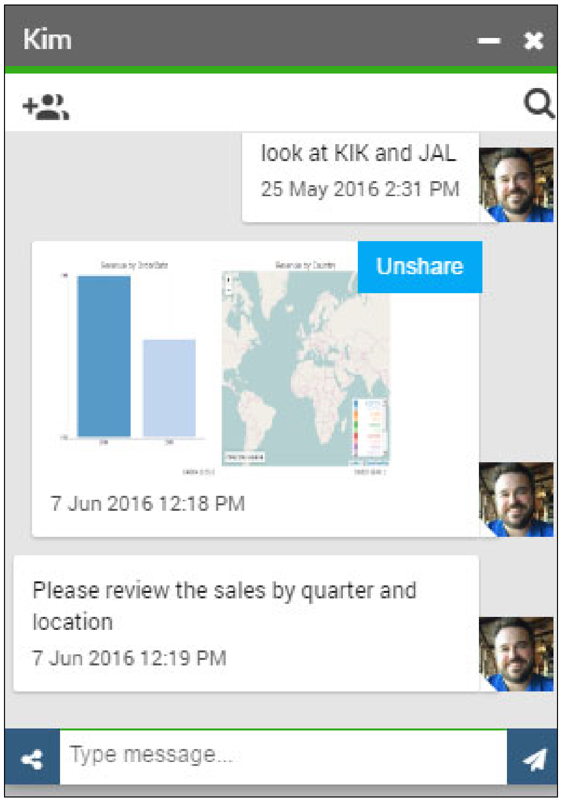


Figure 3-63 Sharing from the dashboard tile

The dashboard shows in the conversation, as shown in Figure 3-64, so that you can share the dashboard with the recipients. The dashboard shows both in the conversation and in their home window.



look at KIK and JAL
25 May 2016 2:31 PM

7 Jun 2016 12:18 PM

Please review the sales by quarter and location
7 Jun 2016 12:19 PM

Figure 3-64 Sharing from the dashboard tile

Conversations are retained and available to open and review. A conversation can be removed by clicking the pencil icon in the upper-right corner of the chat window, as shown in Figure 3-65.

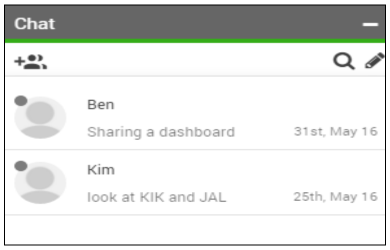


Figure 3-65 Removing a conversation

That chat window changes and a red circle with a minus sign (-) appears next to each conversation. Clicking this red circle with a minus sign (-) icon removes the conversation from the list (Figure 3-66). If necessary, you can start a new conversation with the persons that were part of the deleted chat. Any objects that are shared during a chat are still in shared mode even after you delete the conversation.

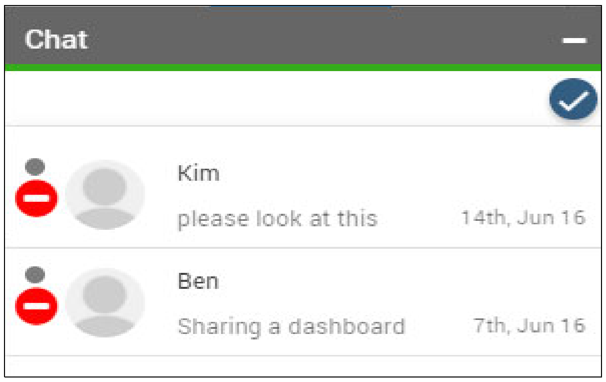


Figure 3-66 Editing chats

4

Using JavaScript in QMF

IBM DB2 Query Management Facility for z/OS V11.2.1 (QMF) contains an embedded JavaScript engine. This engine allows application developers to use JavaScript to further extend the capabilities of QMF. JavaScript can be used to create virtual tables that are based on web service data, in calculated columns in a result set, to create new functions in Visual applications, or to extend the logic that can be used in QMF procedures.

This chapter describes several examples of how to use JavaScript in QMF to expand the capabilities of the product.

This chapter includes the following topics:

- Using JavaScript tables to access data on the web
- Using JavaScript functions in calculations, visual dashboards, and visual reports
- JavaScript procedures

Note: The implementation of JavaScript in QMF contains the core language only. It does not contain objects or methods for manipulating HTML documents.

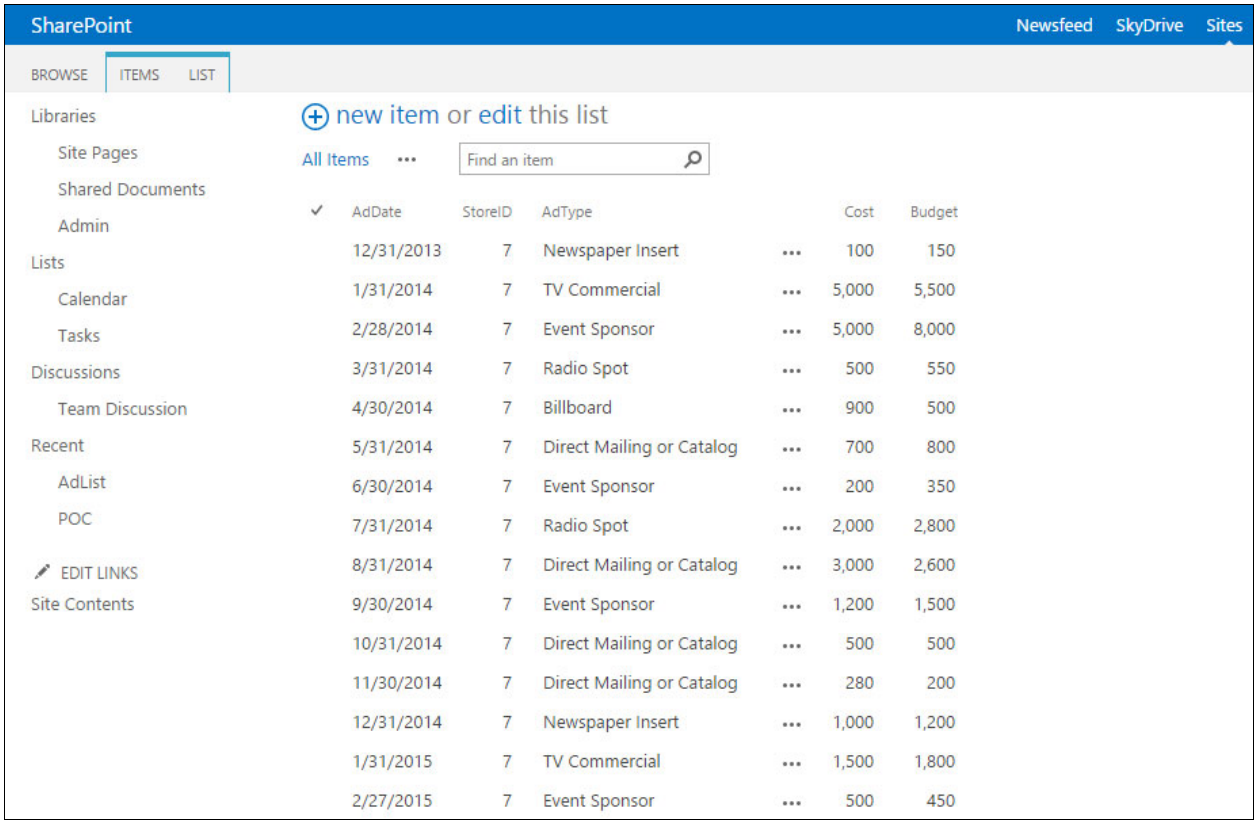
4.1 Using JavaScript tables to access data on the web

This section describes the creation, population, and use of JavaScript tables.

4.1.1 Overview of JavaScript tables

Sometimes, data that you need to access might be available through web services only. You can use JavaScript tables to collect the data from these services and process it in the form of relational tables.

In this example, information is stored in a list on a Microsoft SharePoint server. Figure 4-1 shows a sample of the list in SharePoint.



AdDate	StoreID	AdType		Cost	Budget
12/31/2013	7	Newspaper Insert	•••	100	150
1/31/2014	7	TV Commercial	•••	5,000	5,500
2/28/2014	7	Event Sponsor	•••	5,000	8,000
3/31/2014	7	Radio Spot	•••	500	550
4/30/2014	7	Billboard	•••	900	500
5/31/2014	7	Direct Mailing or Catalog	•••	700	800
6/30/2014	7	Event Sponsor	•••	200	350
7/31/2014	7	Radio Spot	•••	2,000	2,800
8/31/2014	7	Direct Mailing or Catalog	•••	3,000	2,600
9/30/2014	7	Event Sponsor	•••	1,200	1,500
10/31/2014	7	Direct Mailing or Catalog	•••	500	500
11/30/2014	7	Direct Mailing or Catalog	•••	280	200
12/31/2014	7	Newspaper Insert	•••	1,000	1,200
1/31/2015	7	TV Commercial	•••	1,500	1,800
2/27/2015	7	Event Sponsor	•••	500	450

Figure 4-1 Sample AdCostBudget list that is stored in SharePoint

4.1.2 Creating a JavaScript table

Important: You must add a JavaScript table to a virtual data source. If no virtual data source exists, you must create a virtual data source. For more information, see “Creating a virtual data source” in the online help.

In this example, we created a virtual data source that is called Sharepoint Data.

Follow these steps:

1. Create a JavaScript table to be the container for the data that is pulled in from the web service. In the Repository Explorer tab in either QMF Workstation or QMF for WebSphere, expand **Virtual Data Sources** → **Sharepoint Data** → **Database** → **Tables**, as shown in Figure 4-2.

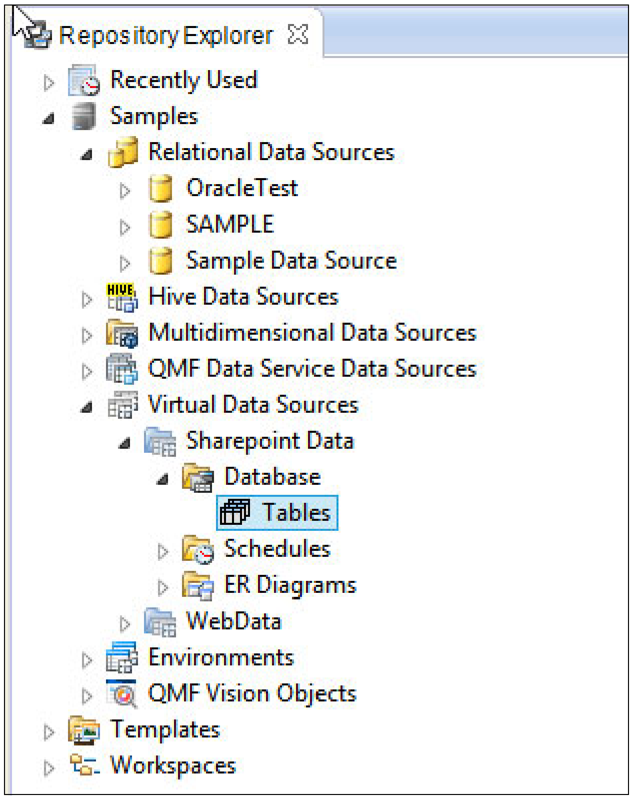


Figure 4-2 Selecting Tables under the Sharepoint Data Virtual Data Source

2. Right-click **Tables**. Select **New** → **JavaScript Table**, as shown in Figure 4-3, to open the JavaScript Table wizard.

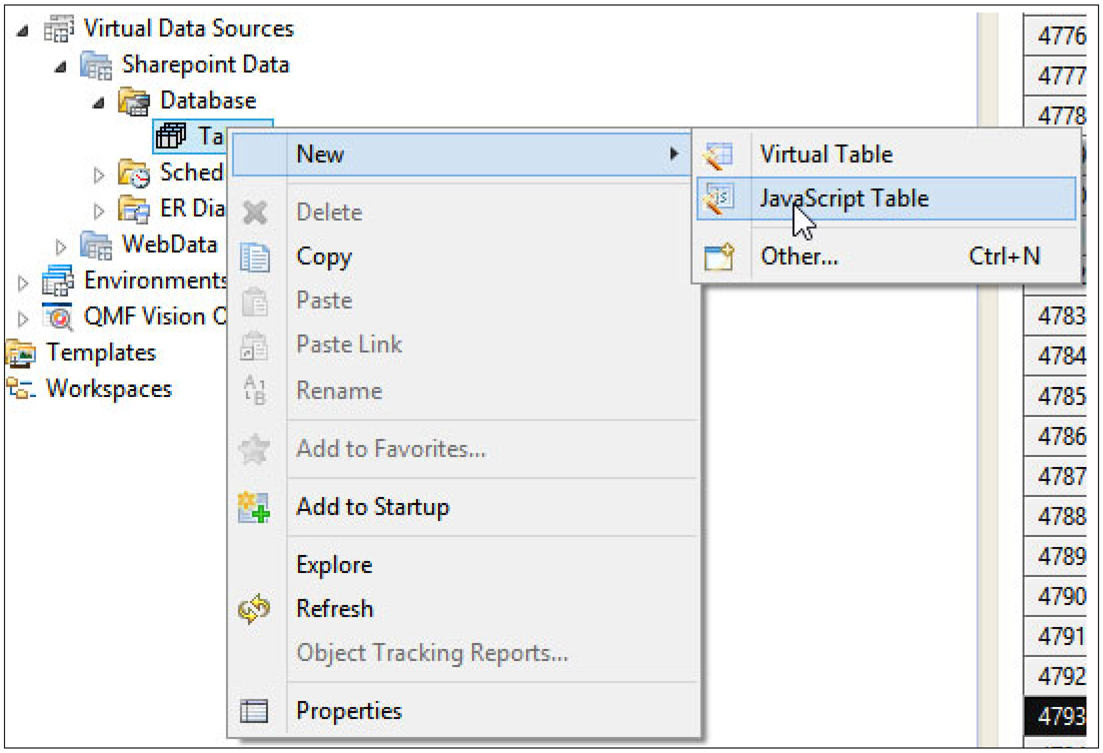
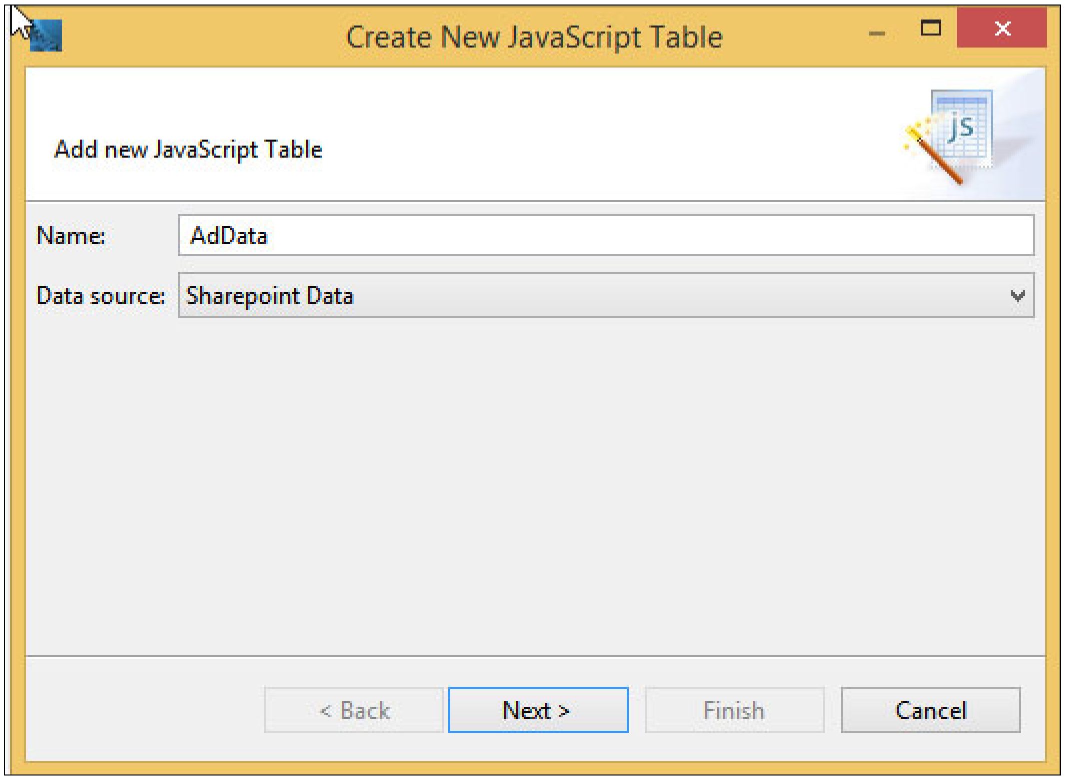


Figure 4-3 Launching the JavaScript Table wizard

3. Enter a logical name for the JavaScript Table and verify that the correct Virtual Data Source is selected from the drop-down list. In this example, we get information about various advertising campaigns. After you enter the name and select the data source, click **Next** (Figure 4-4).


Create New JavaScript Table

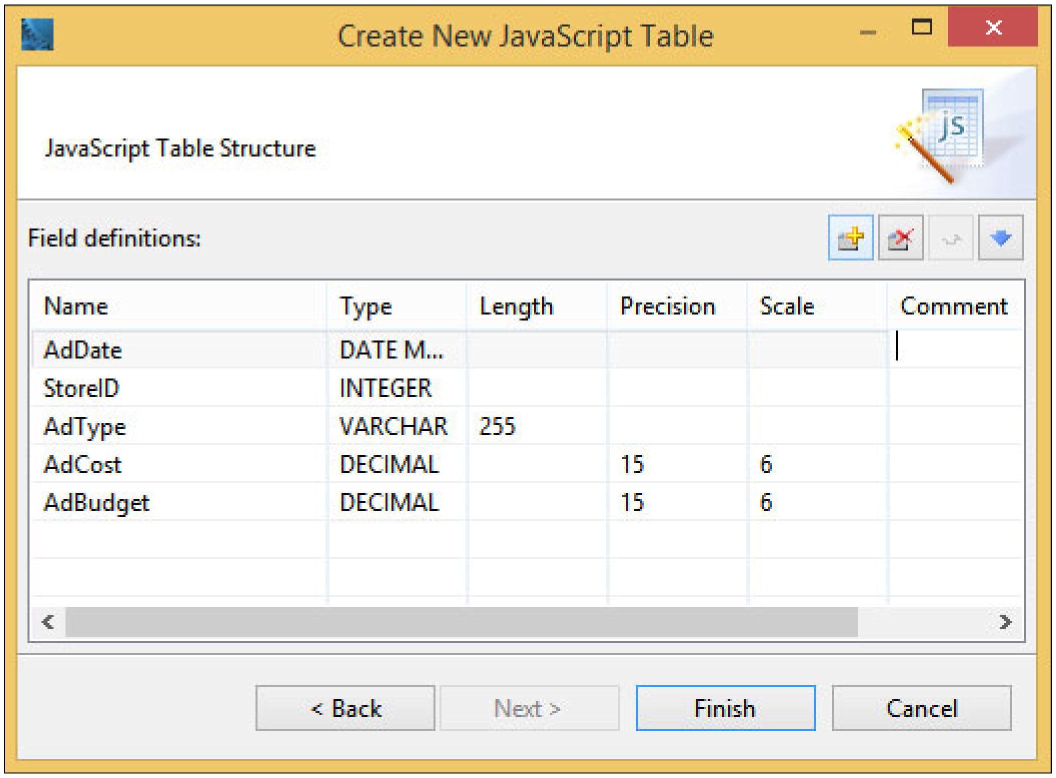
Add new JavaScript Table

Name: AdData

Data source: Sharepoint Data

Figure 4-4 The JavaScript Table wizard

4. Add columns that correlate to the fields that are returned from the web service. A wide variety of character, numeric, date, and time data types are available (Figure 4-5).



Name	Type	Length	Precision	Scale	Comment
AdDate	DATE M...				
StoreID	INTEGER				
AdType	VARCHAR	255			
AdCost	DECIMAL		15	6	
AdBudget	DECIMAL		15	6	

Figure 4-5 Setting the JavaScript table structure

After you create the table, you can add, delete, and modify columns.

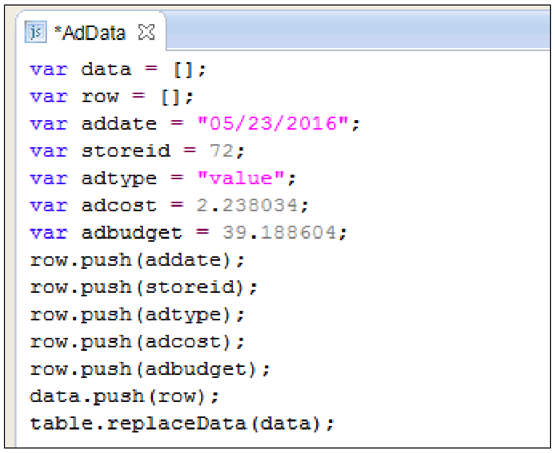
Video: If documentation does not exist about the data that is returned by the web service, it is helpful to launch the URL in a browser to see the data that is returned.

See the example at this website:

`https://youtu.be/QPumjy5sWpc`

The code that is provided in this example does not handle the Sharepoint Data if it returns rows with a different number of columns than the number that is defined for the JavaScript table structure. SharePoint does not include a column that has a null value. If you plan to include a column in the JavaScript table, you must set the column in the SharePoint list so that it does not allow null values.

5. After you add all of the columns, click **Finish**. The script editor for the JavaScript table opens, containing basic code that is based on the columns that were added (Figure 4-6).


*AdData

```
var data = [];
var row = [];
var addate = "05/23/2016";
var storeid = 72;
var adtype = "value";
var adcost = 2.238034;
var adbudget = 39.188604;
row.push(addate);
row.push(storeid);
row.push(adtype);
row.push(adcost);
row.push(adbudget);
data.push(row);
table.replaceData(data);
```

Figure 4-6 Basic auto-generated code for a new JavaScript table

6. This example uses the SharePoint application programming interface (API) to pull values from a SharePoint list into a JavaScript table. You can use the code that is shown in Example 4-1 with most SharePoint lists. To use this code, you must change the three items in bold (`s`, `dt`, and `url`) to fit your environment.

Note: First, you must obtain the list ID for the SharePoint list. See the video:

`https://nickgrattan.wordpress.com/2008/04/29/finding-the-id-guid-for-a-SharePoint-list/`

After you obtain the list ID for the SharePoint list, retrieve the `s` and `dt` values by viewing the XML data URL in a browser. To see how to formulate this URL, see the article at this website:

`https://blogs.msdn.microsoft.com/kaevans/2009/05/01/getting-xml-data-from-a-SharePoint-list-the-easy-way/`

Example 4-1 Using the populateJSTable function

```
var namedata = populateJSTable();
function populateJSTable(){
   table.eraseData();
   var xmlData = obtainXMLData();
   var totalRows = xmlData["totalRows"];
   for (var i = 0; i<= totalRows; i++){
      table.appendData(populateRow(xmlData, i));
   }
}
function populateRow(innerXMLData, rowIndex){
   var data = [];
   row = [];
   var columnNames = innerXMLData["columnNames"];
   var columnLabels = innerXMLData["columnLabels"];
   var rows = innerXMLData["rows"];
   for (var columnIndex = 0; columnIndex<columnNames.length; columnIndex++){
      var columnValue = rows[rowIndex]["@"+columnNames[columnIndex]].toString();
      row.push( columnValue );
      log(columnValue);
   }
    data.push(row);

    return data;
}
function obtainXMLData(){
   var s = new Namespace("uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882");
   var dt = new Namespace("uuid:C2F41010-65B3-11d1-A29F-00AA00C14882");
   var rs = new Namespace("urn:schemas-microsoft-com:rowset");
   var z = new Namespace("#RowsetSchema");
   var str = getXMLData();
   var xml = XML(str);
   var schema = xml.s::Schema.s::ElementType.s::AttributeType;
   var rows = xml.rs::data.z::row;
   var columnNames = [];
   var columnLabels = [];
   for (var key in schema) {
      log("key: " + key);
      columnNames[key] = schema[key].@name;
      log("col_name "+columnNames[key]);
      columnLabels[key] = schema[key].@rs::name;
      log("col_label "+columnLabels[key]);
   };
   var maxRowIndex = -1; for (var key in rows) {maxRowIndex = key;}
   var outputData = {};
   outputData["totalRows"] = maxRowIndex;
   outputData["rows"] = rows;
   outputData["columnNames"] = columnNames;
   outputData["columnLabels"] = columnLabels;
   return outputData;
}
function getXMLData()
{
    var url =
'https://www.xxxxxxxx.com/US/_vti_bin/owssvr.dll?Cmd=Display&List=%7Be7ebba2a-e0ae
-41bf-a809-29fcf441744a%7D&XMLDATA=TRUE';
   var request = new RSBIHttpRequest();
   request.ignoreCertificates();
   request.open("GET", url, false, &DSQQW_DQ&USERNAME&DSQQW_DQ,
&DSQQW_DQ&PASS&DSQQW_DQ, &DSQQW_DQ&DOMAIN&DSQQW_DQ);
   var response = '';
   request.onreadystatechange = function()
   {
      if(request.readyState === 4)
      {
         response = request.responseText;
      }
   }
   request.send(null);
   return response;
}
```

Note: In this example, the user is prompted for the SharePoint username, password, and domain when a query is run. This information is specified in the following line:

```
&DSQQW_DQ&USERNAME&DSQQW_DQ, &DSQQW_DQ&PASS&DSQQW_DQ, &DSQQW_DQ&DOMAIN&DSQQW_DQ
```

The username, password, and domain are wrapped in the double-quotation mark global variable, *&DSQQW_DQ*. You can hardcode these values by using a service account, also.

4.1.3 Populating the JavaScript table

The JavaScript table populates by using the Source Script when a query is run against the JavaScript table.

Depending on how much data is returned and how often the data changes, an administrator might want to set up an expiration schedule to cache the data that is stored in the JavaScript table. This expiration schedule prevents QMF from going to the web service and pulling the data. Instead, the data is stored on the system for as long as the cache is valid. By caching the data, potentially large performance gains are possible.

Video: The following video shows an example of setting up and applying a schedule to a virtual data source:

`https://youtu.be/C_QaVQXyisw`

4.1.4 Using the JavaScript table

You can use JavaScript tables like any other relational database tables. One of the easiest ways to start to use a JavaScript table, such as the AdData table that is created in the example, is to follow these steps:

1. Navigate to the JavaScript table in the Repository Explorer tab, as shown in Figure 4-7.

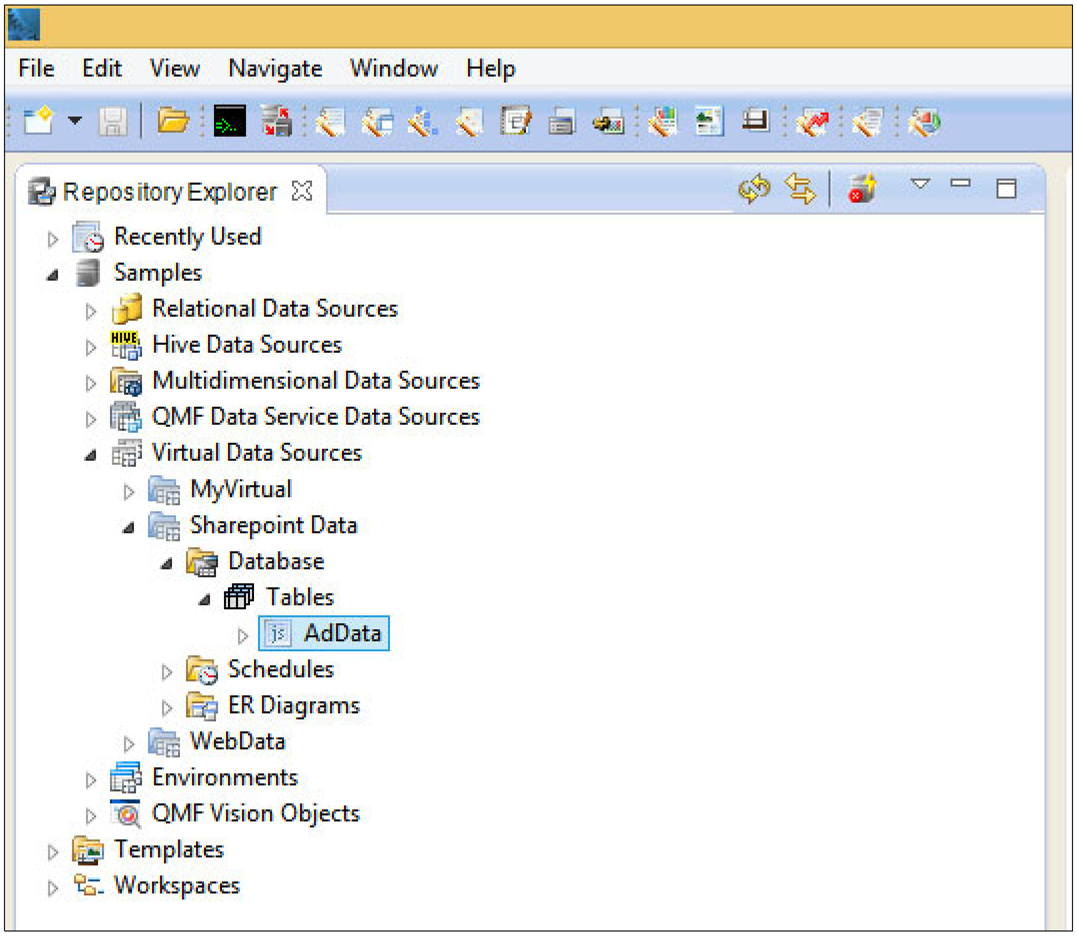
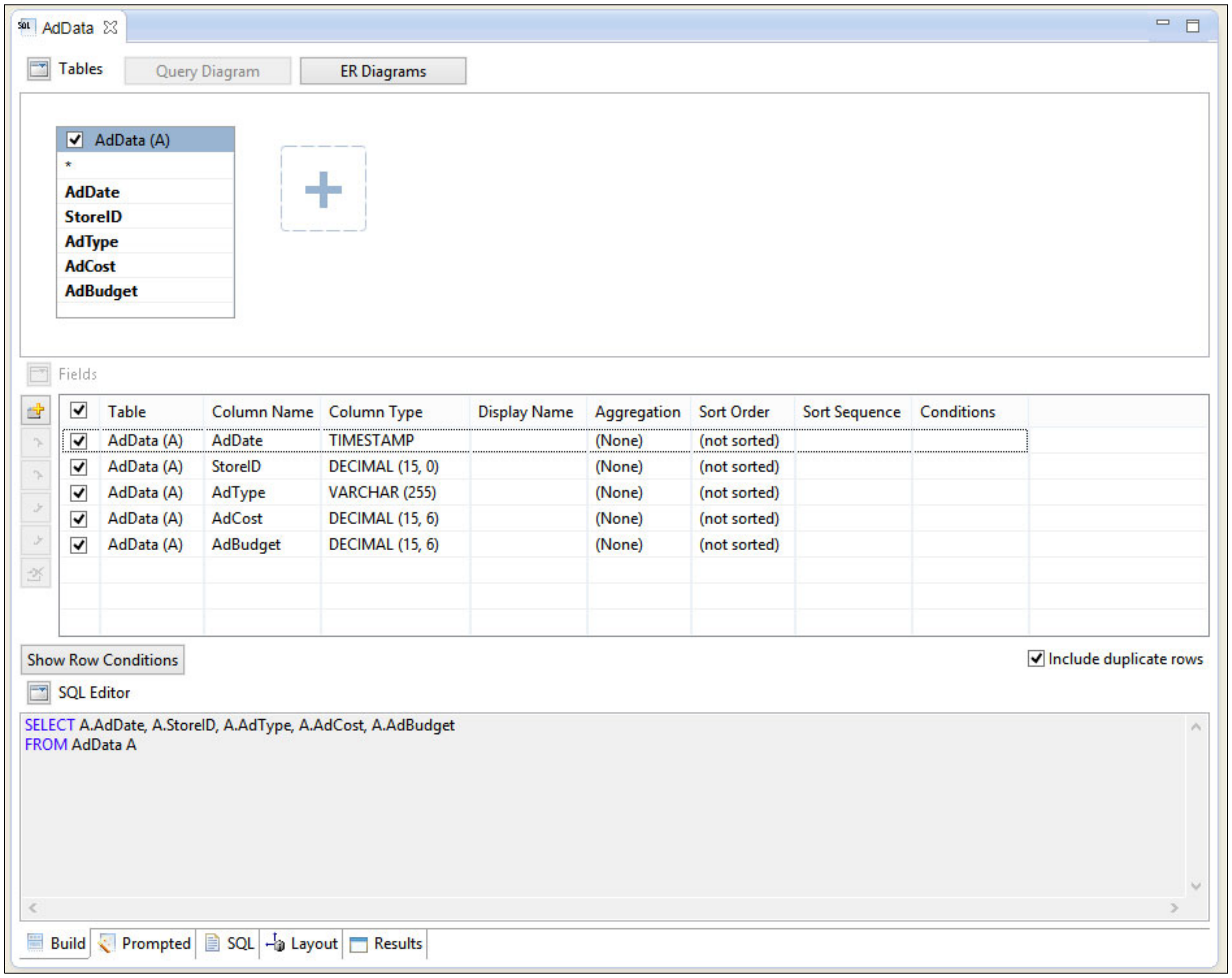


Figure 4-7 JavaScript table in a virtual data source

2. Double-click the JavaScript table, which is AdData in this example, to open the table in the SQL Query Editor (Figure 4-8).



✓	Table	Column Name	Column Type	Display Name	Aggregation	Sort Order	Sort Sequence	Conditions
✓	AdData (A)	AdDate	TIMESTAMP		(None)	(not sorted)		
✓	AdData (A)	StoreID	DECIMAL (15, 0)		(None)	(not sorted)		
✓	AdData (A)	AdType	VARCHAR (255)		(None)	(not sorted)		
✓	AdData (A)	AdCost	DECIMAL (15, 6)		(None)	(not sorted)		
✓	AdData (A)	AdBudget	DECIMAL (15, 6)		(None)	(not sorted)		

```
SELECT A.AdDate, A.StoreID, A.AdType, A.AdCost, A.AdBudget
FROM AdData A
```

Figure 4-8 SQL Query Editor

3. Use the SQL Query Editor to modify the query, as needed.

Note: The JavaScript tables store the results in the embedded Apache Derby database. If you modify the SQL, ensure that you follow the standards that are implemented in Apache Derby. For more information, see this website:

`https://db.apache.org/derby/`

4. When the query is ready, click **Run query** to view the query results (Figure 4-9).

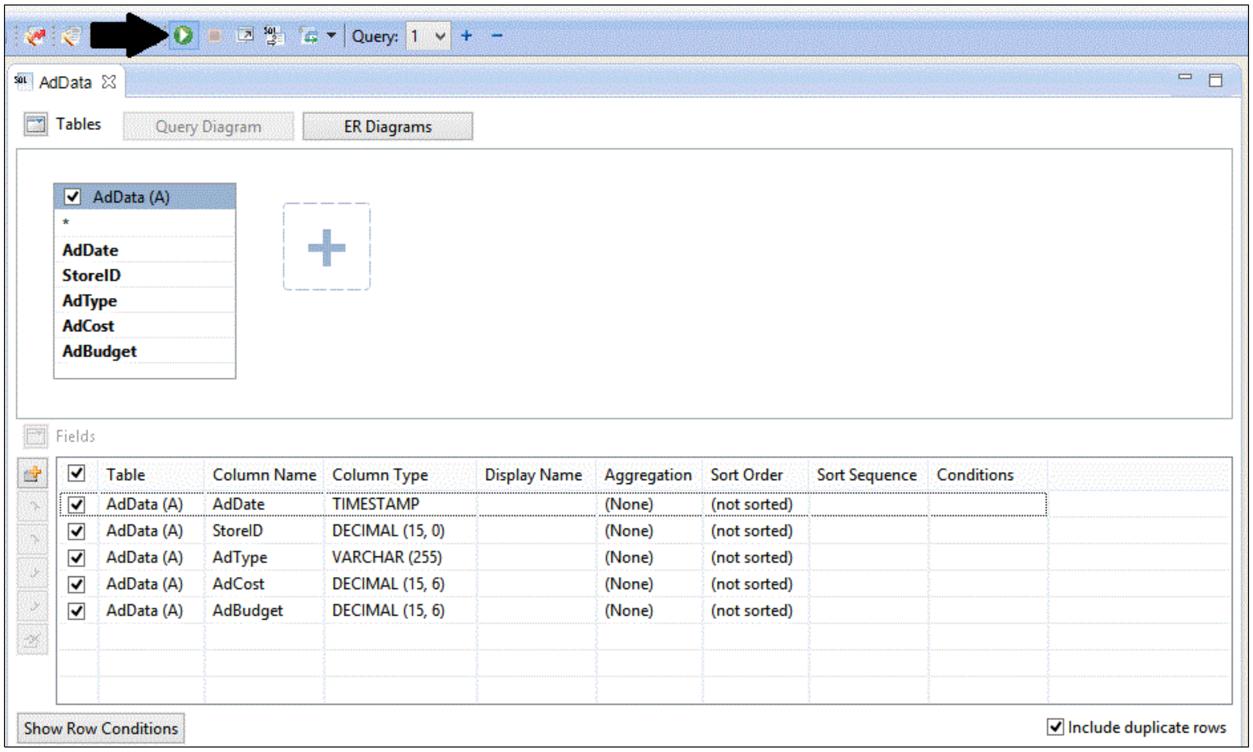
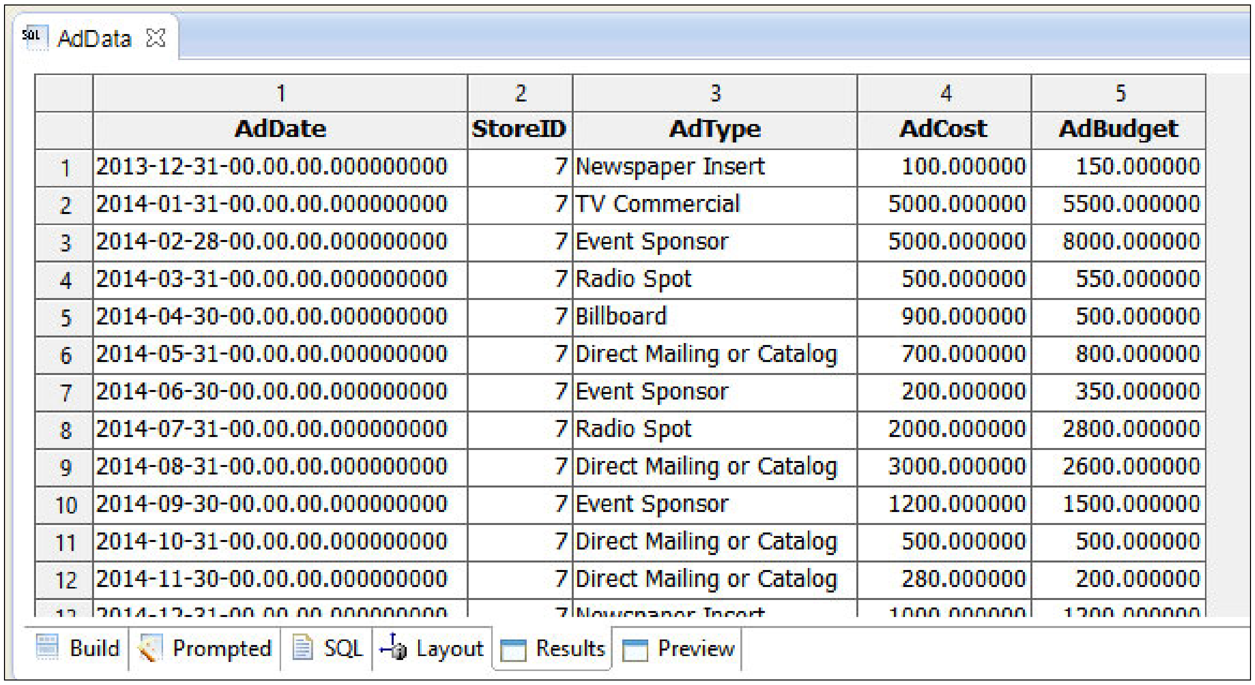


	Table	Column Name	Column Type	Display Name	Aggregation	Sort Order	Sort Sequence	Conditions
✓	AdData (A)	AdDate	TIMESTAMP		(None)	(not sorted)		
✓	AdData (A)	StoreID	DECIMAL (15, 0)		(None)	(not sorted)		
✓	AdData (A)	AdType	VARCHAR (255)		(None)	(not sorted)		
✓	AdData (A)	AdCost	DECIMAL (15, 6)		(None)	(not sorted)		
✓	AdData (A)	AdBudget	DECIMAL (15, 6)		(None)	(not sorted)		

Figure 4-9 The run query icon in the SQL Query Editor

Figure 4-10 shows the results.



	1	2	3	4	5
	AdDate	StoreID	AdType	AdCost	AdBudget
1	2013-12-31-00.00.00.000000000	7	Newspaper Insert	100.000000	150.000000
2	2014-01-31-00.00.00.000000000	7	TV Commercial	5000.000000	5500.000000
3	2014-02-28-00.00.00.000000000	7	Event Sponsor	5000.000000	8000.000000
4	2014-03-31-00.00.00.000000000	7	Radio Spot	500.000000	550.000000
5	2014-04-30-00.00.00.000000000	7	Billboard	900.000000	500.000000
6	2014-05-31-00.00.00.000000000	7	Direct Mailing or Catalog	700.000000	800.000000
7	2014-06-30-00.00.00.000000000	7	Event Sponsor	200.000000	350.000000
8	2014-07-31-00.00.00.000000000	7	Radio Spot	2000.000000	2800.000000
9	2014-08-31-00.00.00.000000000	7	Direct Mailing or Catalog	3000.000000	2600.000000
10	2014-09-30-00.00.00.000000000	7	Event Sponsor	1200.000000	1500.000000
11	2014-10-31-00.00.00.000000000	7	Direct Mailing or Catalog	500.000000	500.000000
12	2014-11-30-00.00.00.000000000	7	Direct Mailing or Catalog	280.000000	200.000000
13	2014-12-31-00.00.00.000000000	7	Newspaper Insert	1000.000000	1200.000000

Figure 4-10 Results of running a query against the AdData JavaScript table

5. You can save the query and use it in the same way that you use any other relational database query in QMF.

4.2 Using JavaScript functions in calculations, visual dashboards, and visual reports

JavaScript is a powerful, cross-platform, object-oriented scripting language. It is one of the most common programming languages in the world, with many tutorials that are readily available online. QMF users use the standard library of objects that are included with JavaScript, such as Array, Date, and Math, and the core set of language elements, such as operators, control structures, and statements, to build their own powerful functions in QMF. JavaScript popularity also provides many existing examples and samples that you can use in QMF.

In this example, we create a JavaScript function to produce the latitude and longitude for a specific address so that the address can be marked on a map.

4.2.1 Creating a JavaScript module

Follow these steps to create a JavaScript module:

1. Click **File** → **New** → **Other** (Figure 4-11).

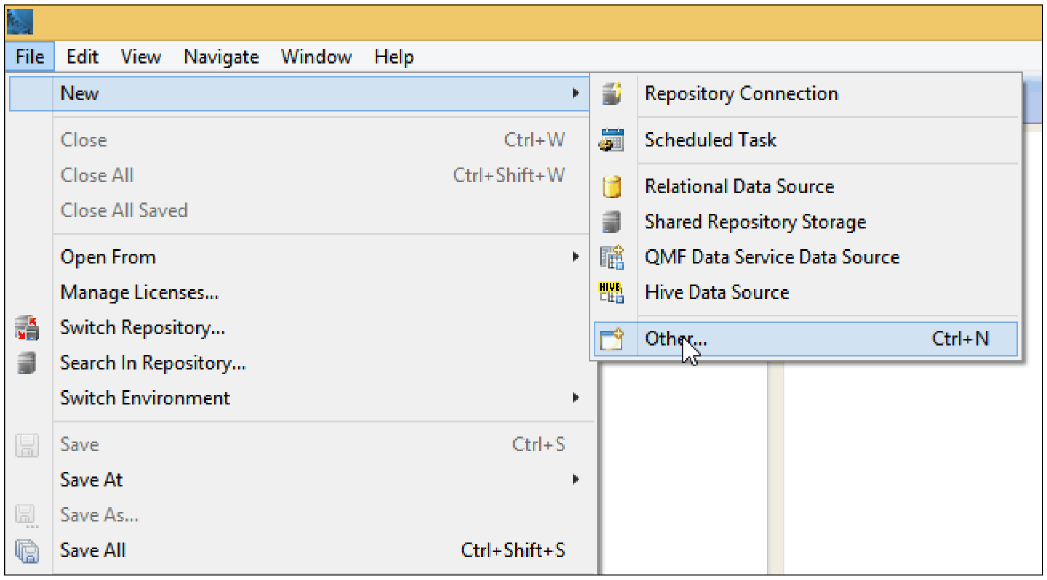


Figure 4-11 Launching the Select a wizard window

2. In the Select a wizard window, expand **QMF Objects** (Figure 4-12).

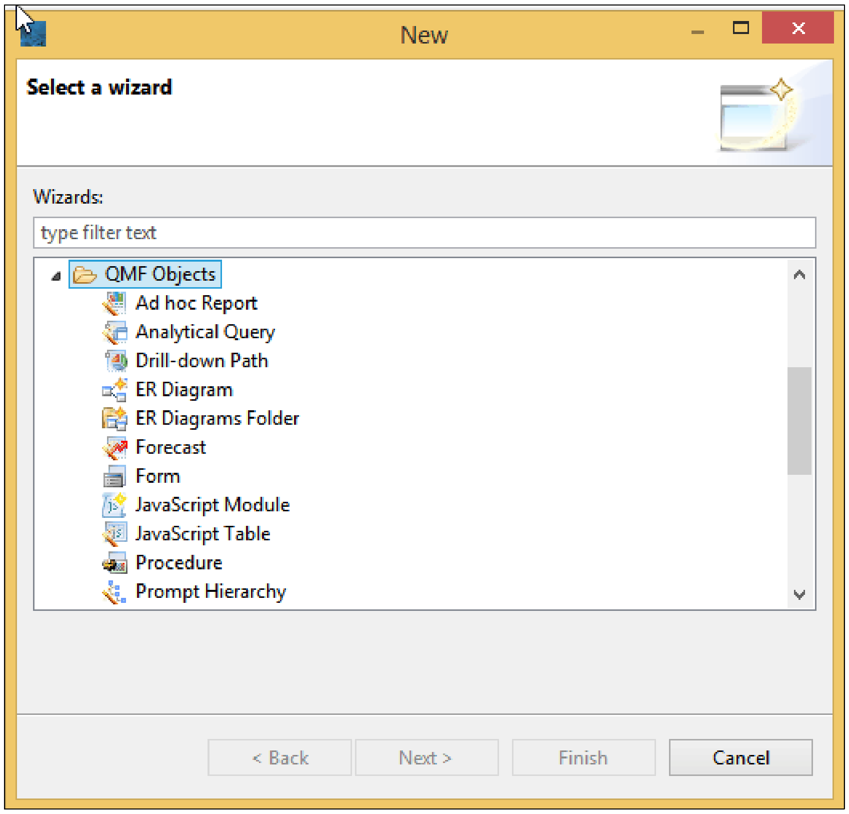
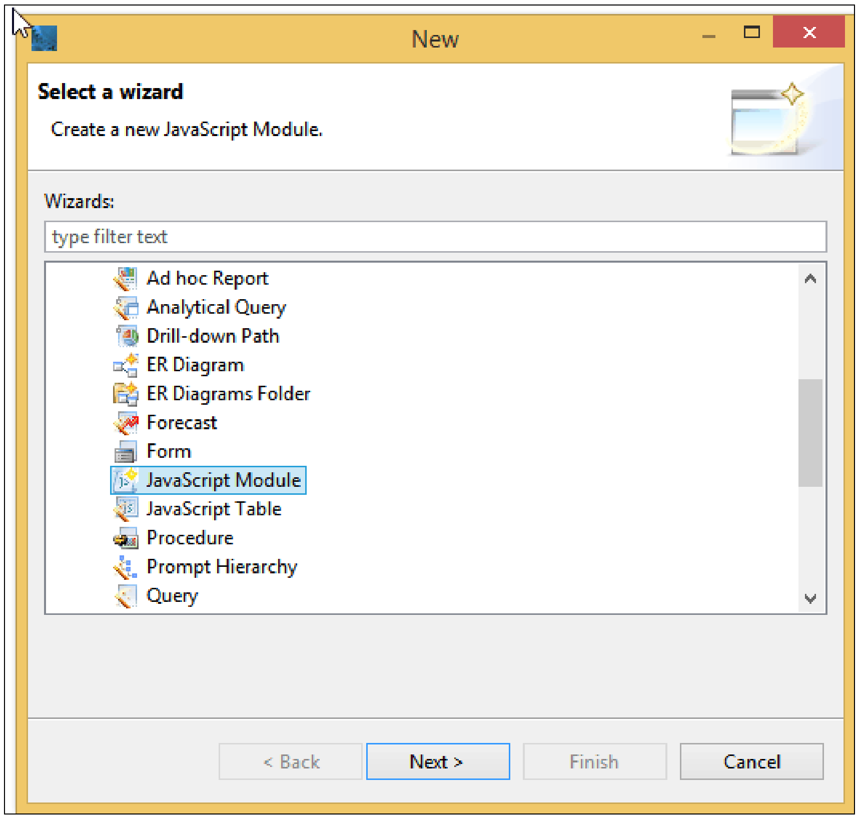


Figure 4-12 List of QMF object wizards

3. Select **JavaScript Module** (Figure 4-13), and click **Next**.


New

Select a wizard

Create a new JavaScript Module.

Wizards:

type filter text

- Ad hoc Report
- Analytical Query
- Drill-down Path
- ER Diagram
- ER Diagrams Folder
- Forecast
- Form
- JavaScript Module
- JavaScript Table
- Procedure
- Prompt Hierarchy
- Query

Figure 4-13 Selecting the JavaScript Module wizard

4. Enter a name for the JavaScript module and click **Finish** (Figure 4-14).

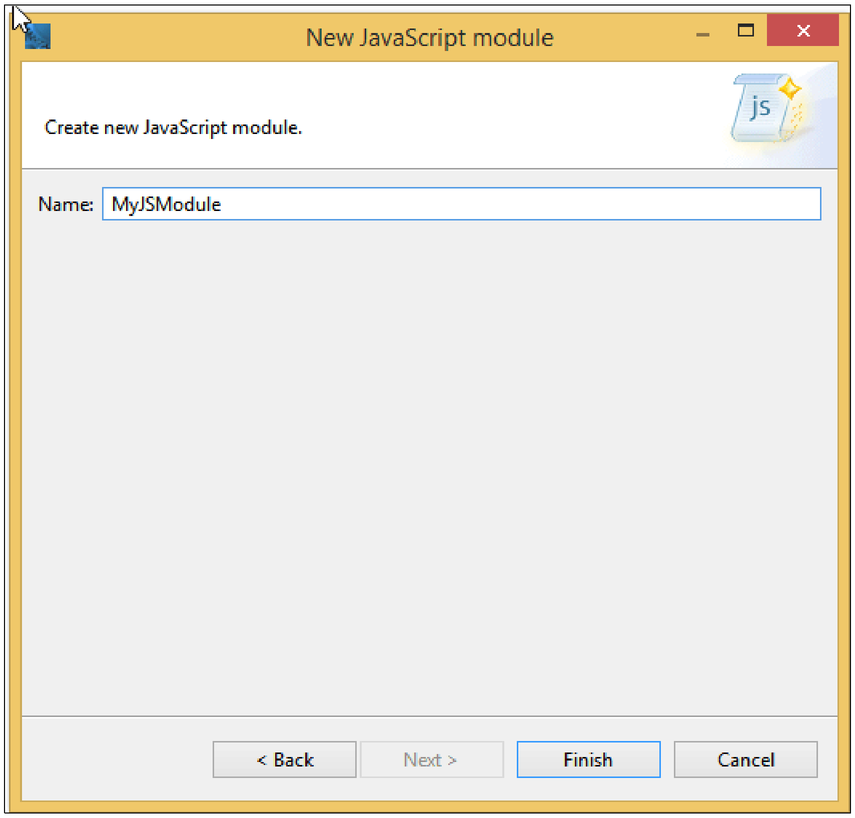


Figure 4-14 The JavaScript module wizard

5. A blank JavaScript Editor window opens (Figure 4-15).

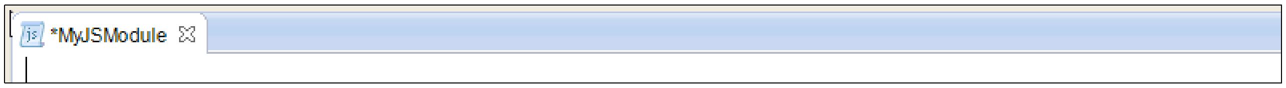
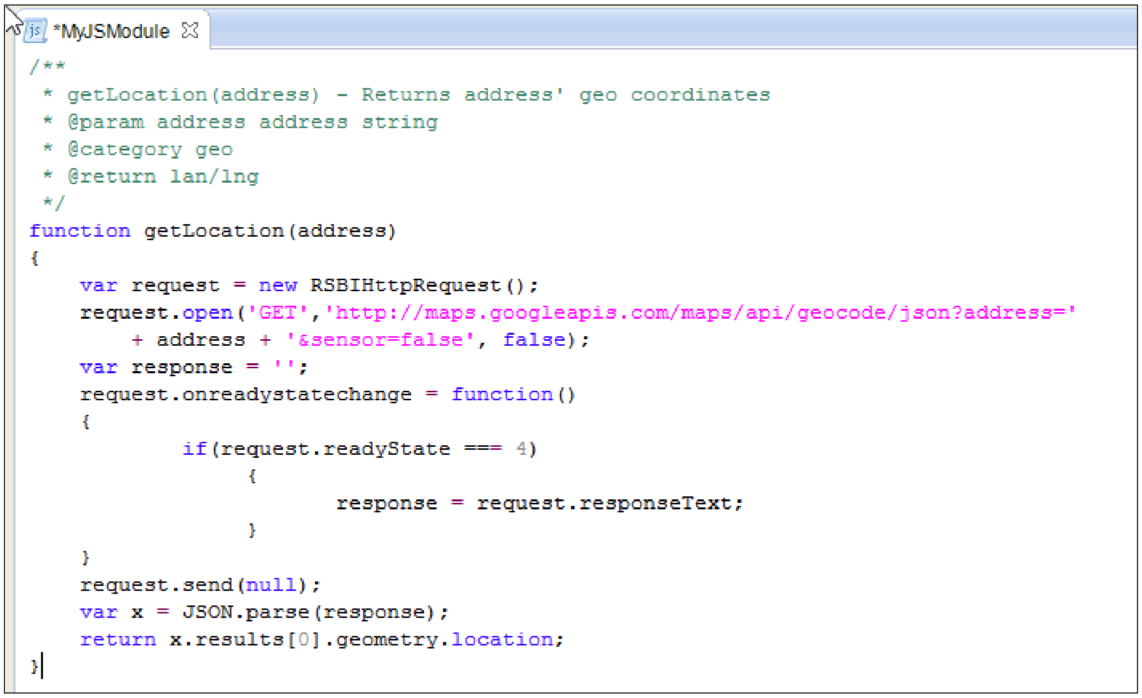


Figure 4-15 Empty JavaScript Editor for a new JavaScript module

6. QMF users can use this editor to create their JavaScript functions (Figure 4-16).



```
/**
 * getLocation(address) - Returns address' geo coordinates
 * @param address address string
 * @category geo
 * @return lan/lng
 */
function getLocation(address)
{
    var request = new RSBIHttpRequest();
    request.open('GET','http://maps.googleapis.com/maps/api/geocode/json?address='
        + address + '&sensor=false', false);
    var response = '';
    request.onreadystatechange = function()
    {
            if(request.readyState === 4)
                 {
                        response = request.responseText;
                 }
    }
    request.send(null);
    var x = JSON.parse(response);
    return x.results[0].geometry.location;
}
```

Figure 4-16 JavaScript function that produces geo coordinates that are based on a specified address

7. Example 4-2 provides this code.

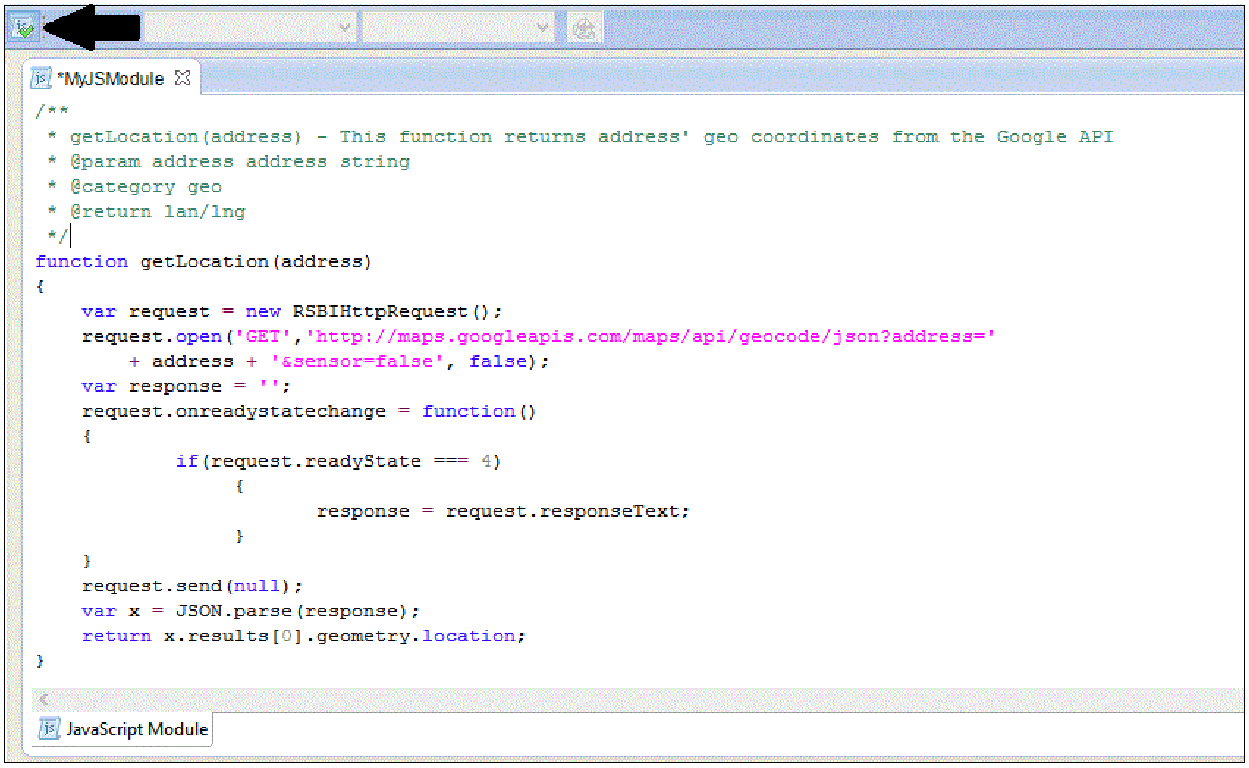
Example 4-2 JavaScript function that produces geo coordinates that are based on a specified address

```
/**
 * getLocation(address) - This function returns address' geo coordinates from the
Google API
 * @category Geo
 * @param address address string
 * @return lat/lng
 */
function getLocation(address)
{
      var request = new RSBIHttpRequest();
        var address_en = encodeURIComponent(address.trim());
   request.open('GET','http://maps.googleapis.com/maps/api/geocode/json?address='
+ address_en + '&sensor=false', false);
   var response = '';
   request.onreadystatechange = function()
   {
         if(request.readyState === 4)
                 {
                 response = request.responseText;
                 }
   }
   request.send(null);
   var x = JSON.parse(response);
   return x.results[0].geometry.location;
}
```

Note: If a comment is added before the function, the application interprets this comment as the description of the function. This comment is shown when the function is selected in the Expression Designer in a Visual Application or the Calculated Column Expression builder for a query.

You can use the @category tag to add the function to an existing function category or to create a category for the function. If no @category is specified in the comment, the function is added to the JavaScript Modules category.

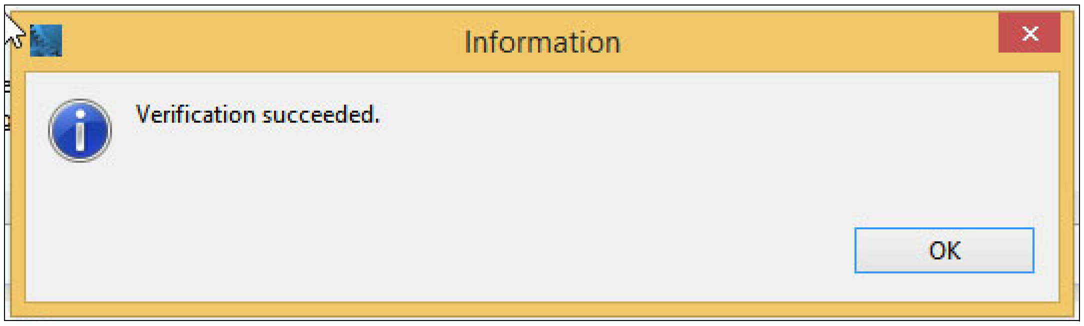
8. After the function is finished, click the (highlighted) **Check JavaScript Syntax and Structure** icon to validate the JavaScript function (Figure 4-17).



```
/**
 * getLocation(address) - This function returns address' geo coordinates from the Google API
 * @param address address string
 * @category geo
 * @return lan/lng
 */
function getLocation(address)
{
    var request = new RSBIHttpRequest();
    request.open('GET','http://maps.googleapis.com/maps/api/geocode/json?address='
        + address + '&sensor=false', false);
    var response = '';
    request.onreadystatechange = function()
    {
            if(request.readyState === 4)
                 {
                         response = request.responseText;
                 }
    }
    request.send(null);
    var x = JSON.parse(response);
    return x.results[0].geometry.location;
}
```

Figure 4-17 Check JavaScript Syntax in the JavaScript Editor

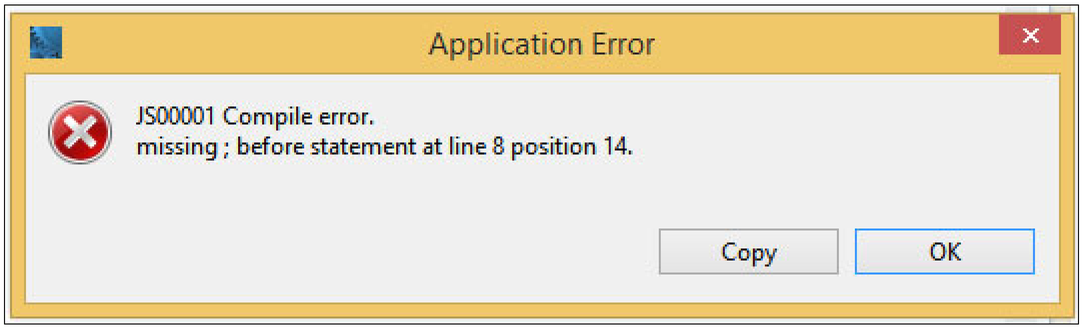
9. If the JavaScript in the module is valid, the message that is shown in Figure 4-18 is returned.



Verification succeeded.

Figure 4-18 Successful verification message

If the validation fails, an Application Error message shows with additional information about where the error occurred in the code (Figure 4-19).

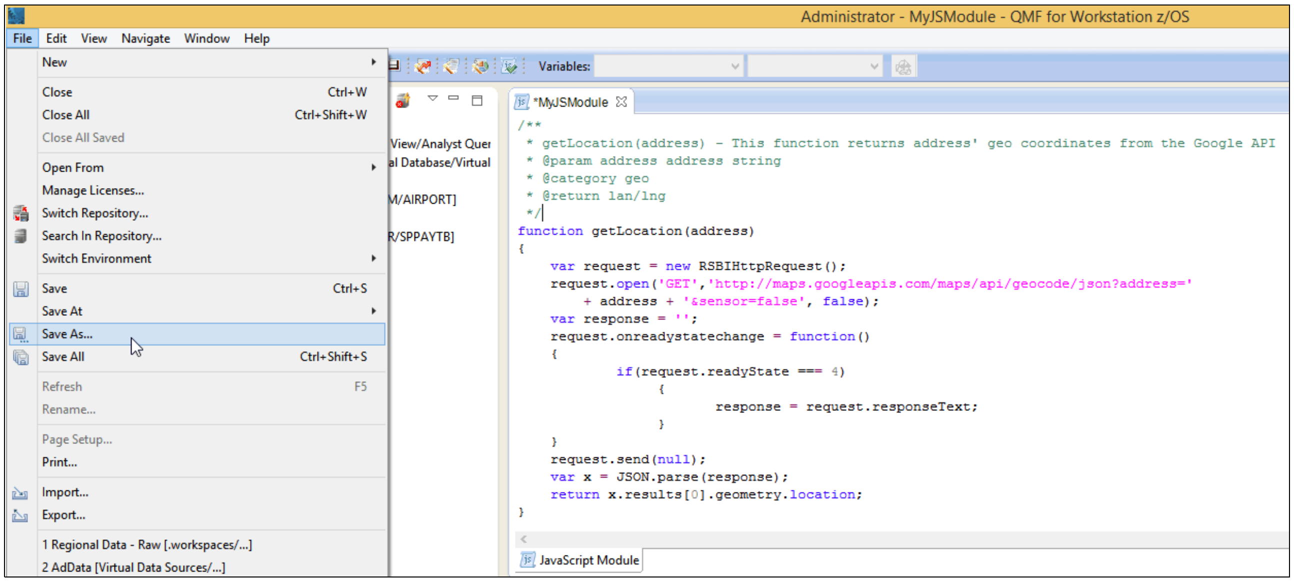


JS00001 Compile error.
missing ; before statement at line 8 position 14.

Figure 4-19 Validation failure

10. If the code is valid, save the JavaScript module to the repository so that users can load the module for use with their queries or Visual Applications. Click **File** → **Save as** (Figure 4-20).

:



```
/**
 * getLocation(address) - This function returns address' geo coordinates from the Google API
 * @param address address string
 * @category geo
 * @return lan/lng
 */
function getLocation(address)
{
    var request = new RSBIHttpRequest();
    request.open('GET','http://maps.googleapis.com/maps/api/geocode/json?address='
        + address + '&sensor=false', false);
    var response = '';
    request.onreadystatechange = function()
    {
            if(request.readyState === 4)
                {
                        response = request.responseText;
                }
    }
    request.send(null);
    var x = JSON.parse(response);
    return x.results[0].geometry.location;
}
```

Figure 4-20 Save the JavaScript module

11.Select **Save to Repository** (Figure 4-21).

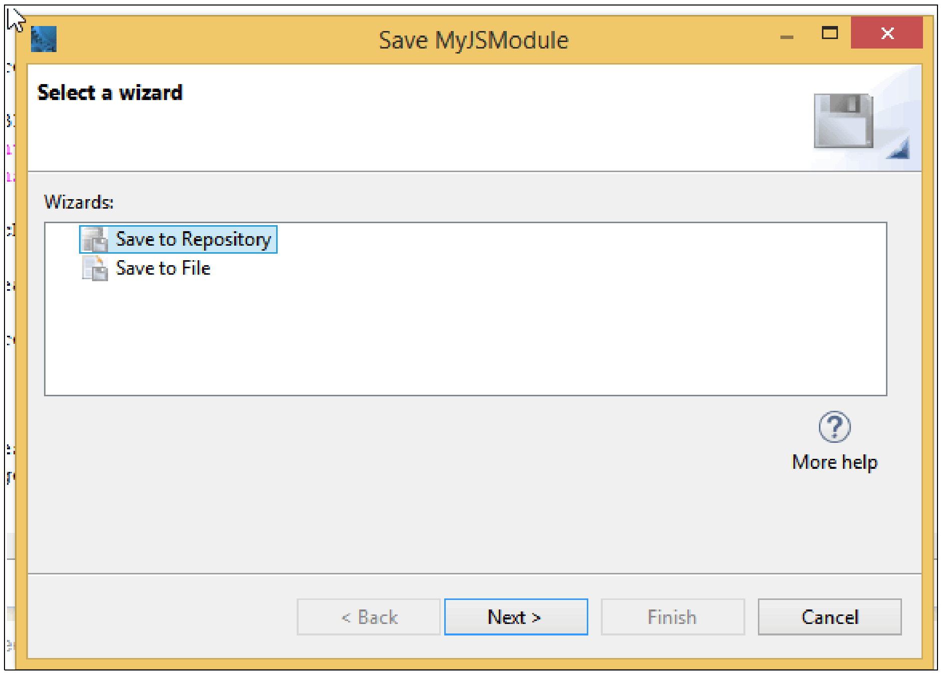
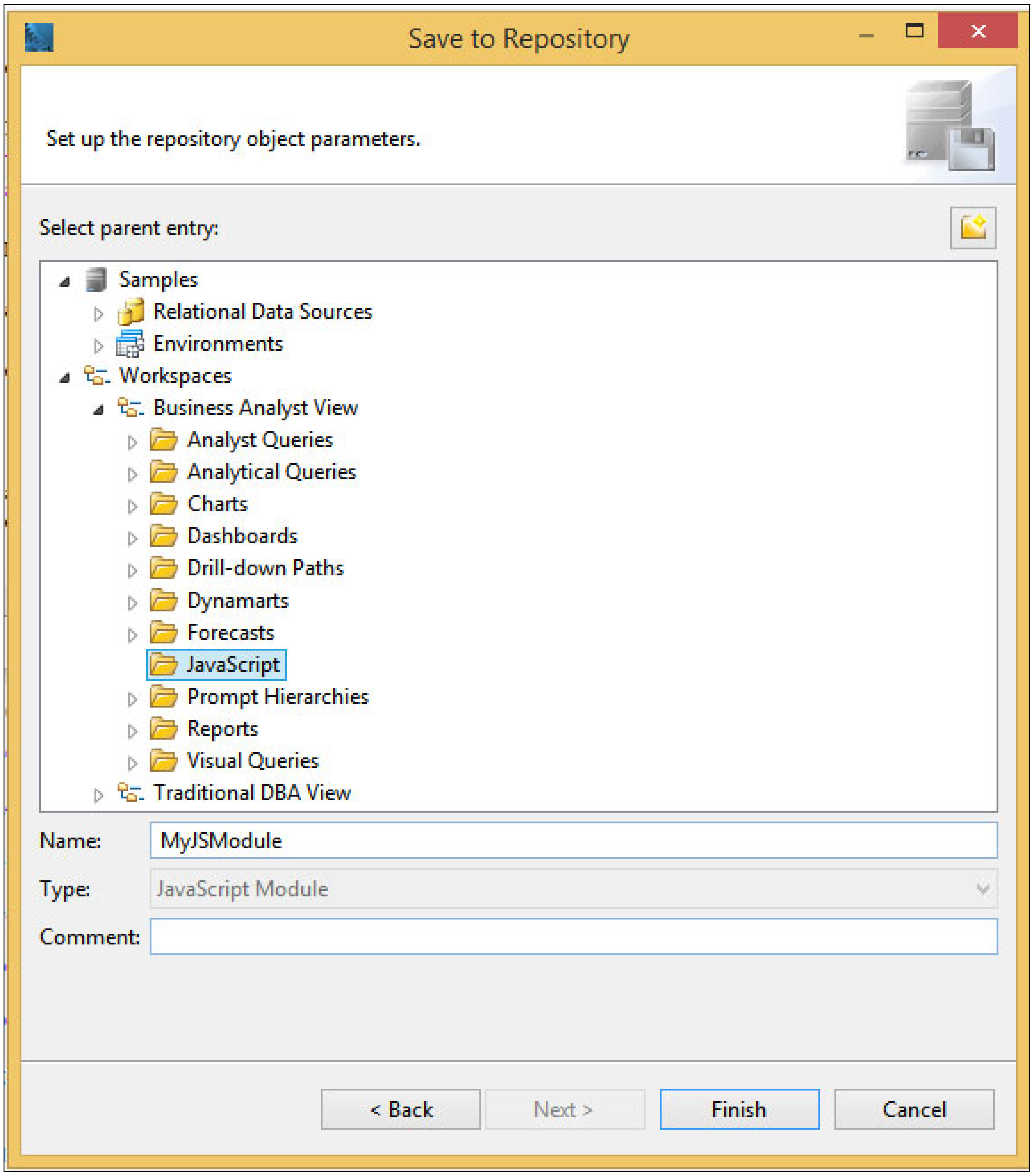


Figure 4-21 Launching the Save to Repository wizard to save the JavaScript module

12. In the Save to Repository window, select the workspace to save the JavaScript module in and click **Finish** (Figure 4-22).

    In this example, we created a JavaScript folder in the Business Analyst View workspace. We named the JavaScript module MyJSModule and saved it to the JavaScript folder.



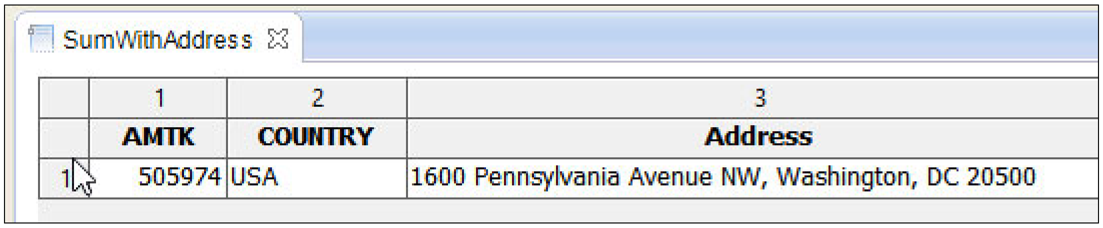
Name: MyJSModule
Type: JavaScript Module
Comment:

Figure 4-22 Saving the JavaScript module to the repository

4.2.2 Using a JavaScript module in a calculated column

You can use JavaScript functions in the Expression Designer when you add calculated columns to queries. Users can reuse existing JavaScript by loading the module in the query. Follow these steps:

1. Open the query to which you want to add a JavaScript module with the functions in the query editor. The following example is a visual query (Figure 4-23).

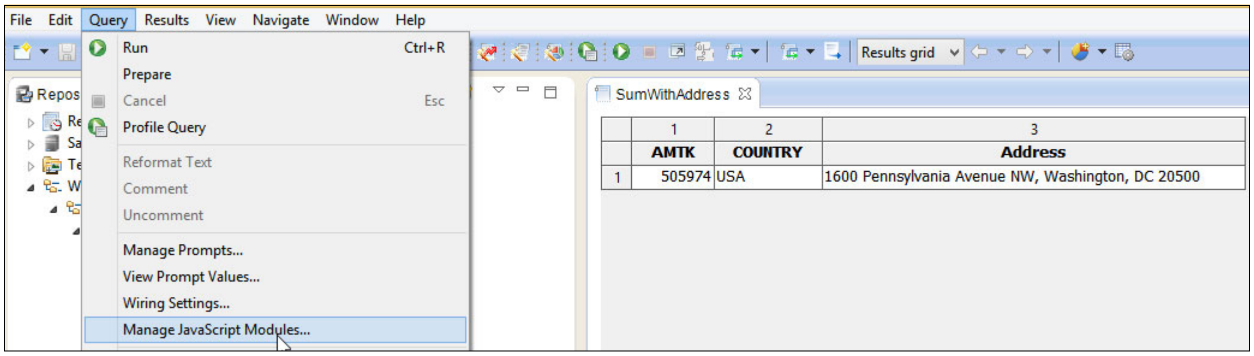


	1	2	3
	AMTK	COUNTRY	Address
1	505974	USA	1600 Pennsylvania Avenue NW, Washington, DC 20500

Figure 4-23 Sample result set

This query contains a total, country, and full address. This example uses the function that we created in the JavaScript module to produce calculated columns that provide the latitude and longitude for the address.

2. From the main menu, select **Query** → **Manage JavaScript Modules** (Figure 4-24).



	1	2	3
	AMTK	COUNTRY	Address
1	505974	USA	1600 Pennsylvania Avenue NW, Washington, DC 20500

Figure 4-24 Selecting Manage JavaScript Modules from the Query menu

3. The JavaScript Modules window opens (Figure 4-25).

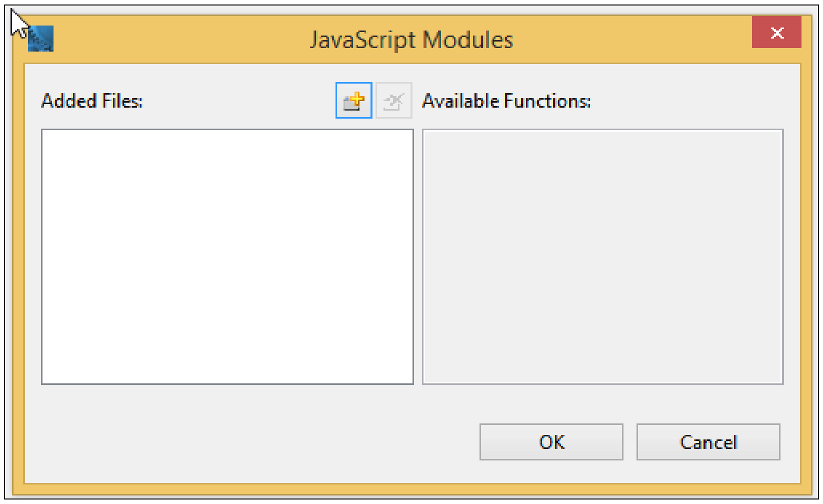
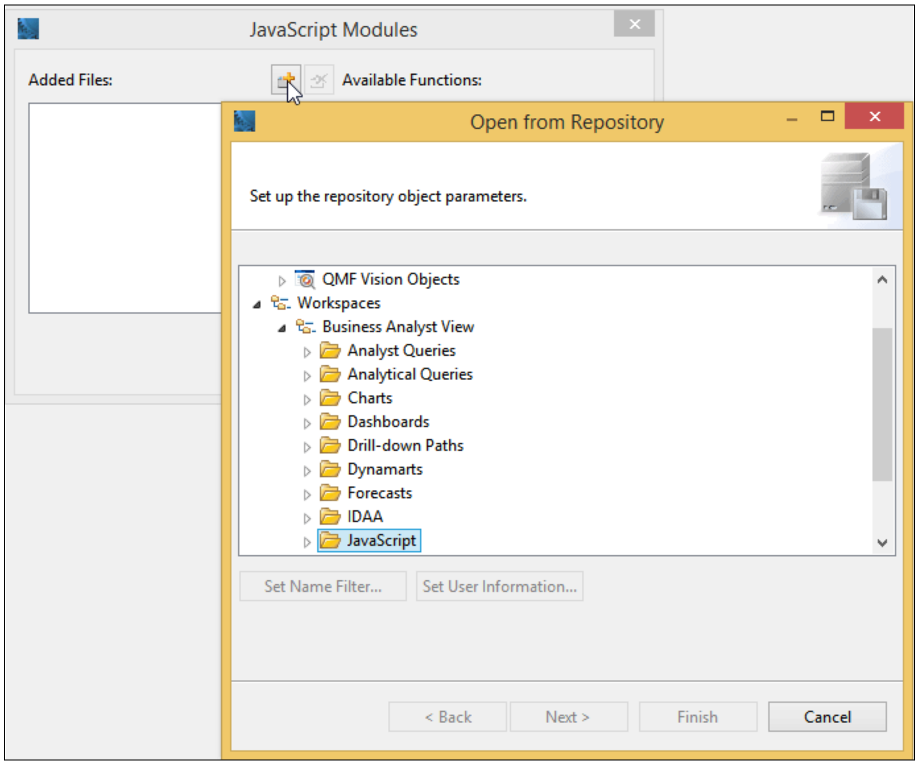


Figure 4-25 JavaScript Modules window

4. Click the Add JavaScript Module plus sign (**+**) icon. The Open from Repository window opens (Figure 4-26).



Set up the repository object parameters.

- QMF Vision Objects
- Workspaces
  - Business Analyst View
    - Analyst Queries
    - Analytical Queries
    - Charts
    - Dashboards
    - Drill-down Paths
    - Dynamarts
    - Forecasts
    - IDAA
    - JavaScript

Figure 4-26 Opening a JavaScript module from the repository

5. Select the JavaScript module that you want to add, and click **Finish**. In this example, we navigated to the MyJSModule JavaScript module in the JavaScript folder that we previously created (Figure 4-27).

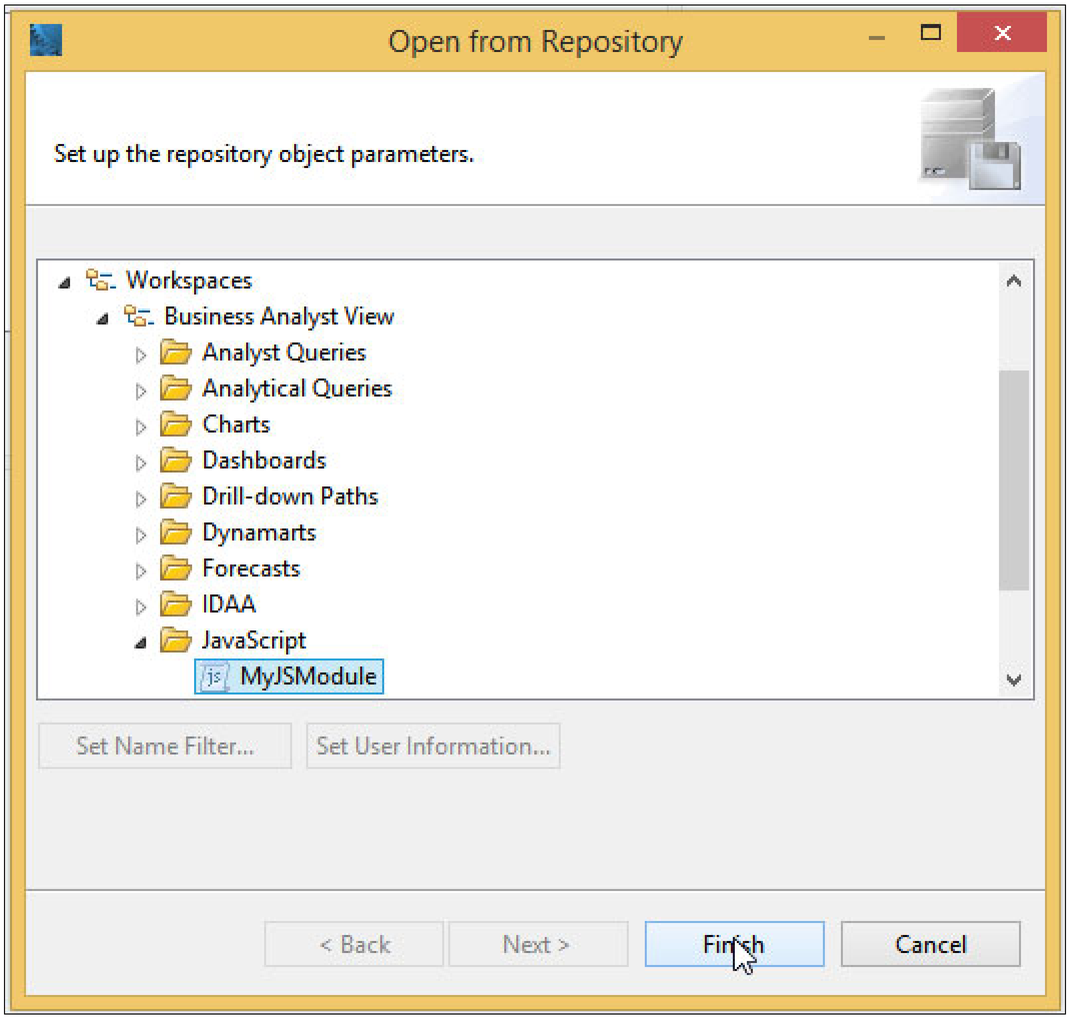
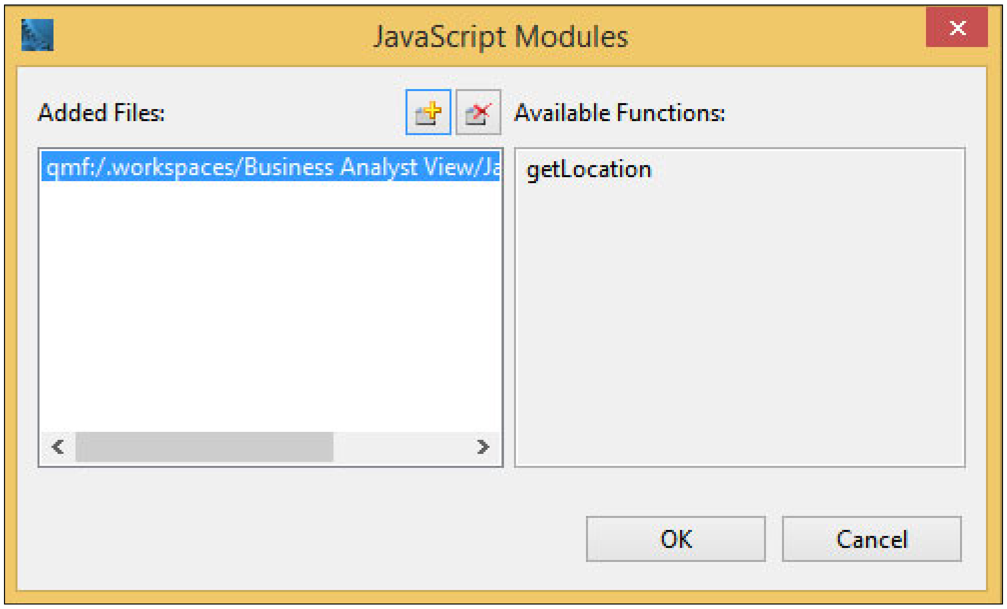


Figure 4-27 Selecting the JavaScript module from the repository

6. After the module is added, the available functions are listed (Figure 4-28).



qmf:/.workspaces/Business Analyst View/Ja

getLocation

Figure 4-28 JavaScript module pane after the module is added

7. Click **OK** to return to the query that the JavaScript module was loaded into.

8. Right-click a column in the query. (We selected the Address column.) Select **Add Calculated Column After** (or Before). See Figure 4-29.

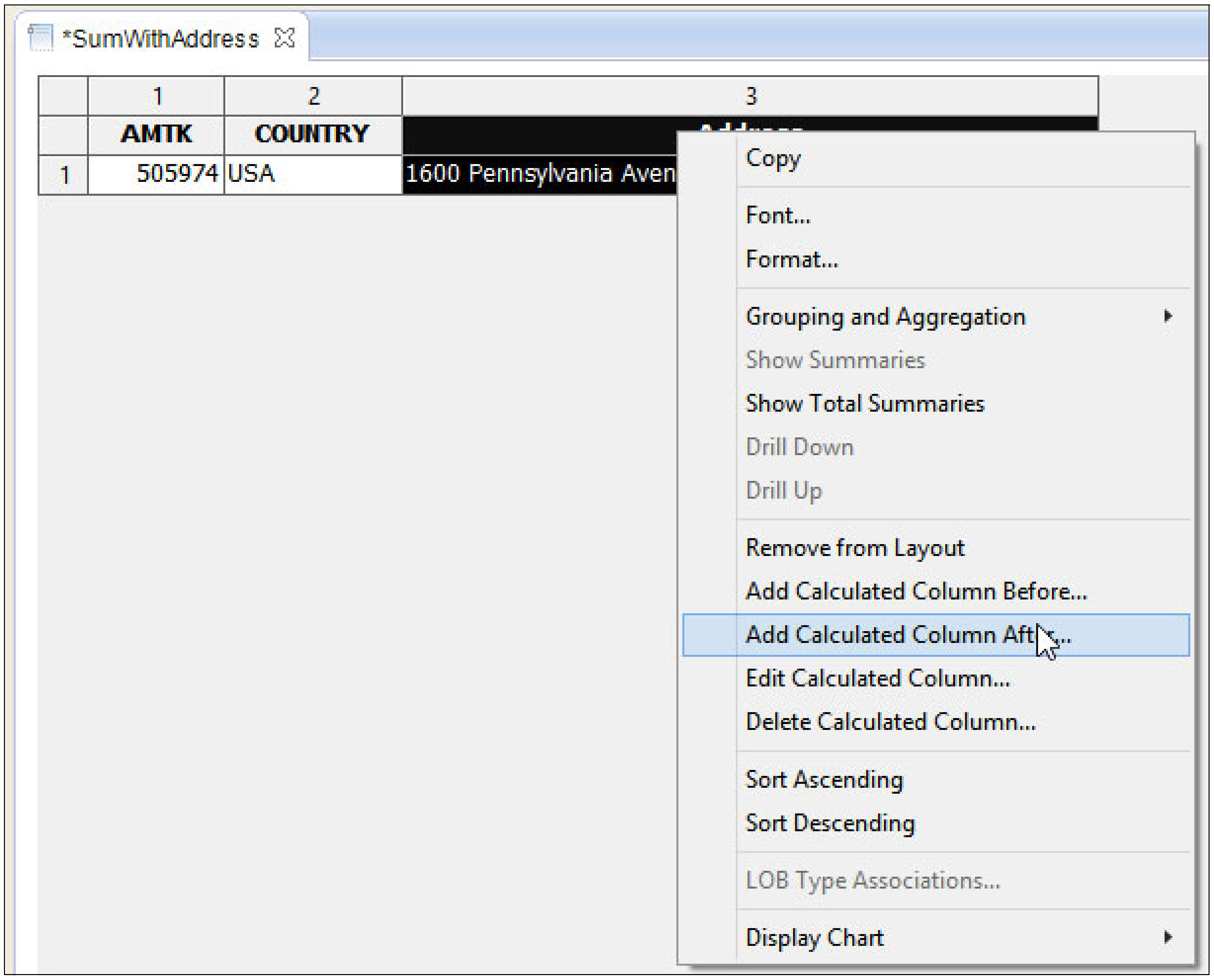
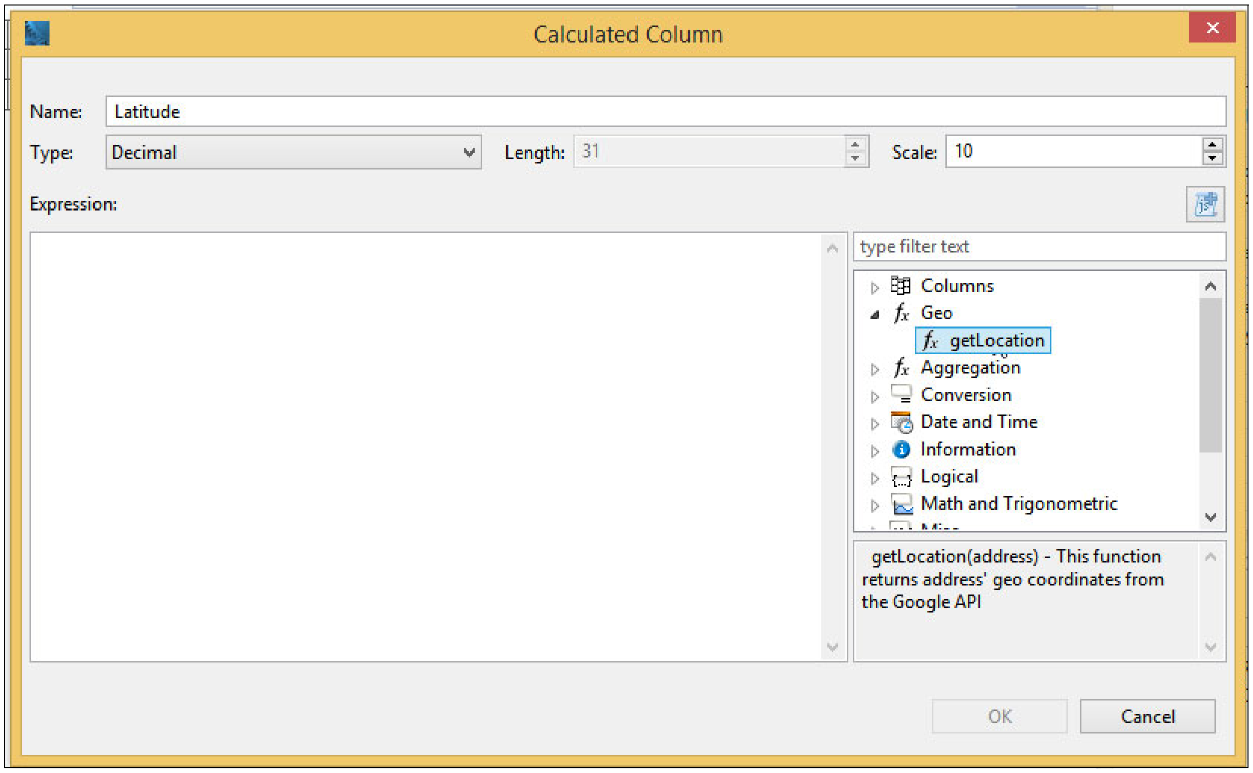


Figure 4-29 Selecting the Add Calculated Column After option from the column menu

9. The Expression Designer opens for a new calculated column. We named the column Latitude because we pull the latitude from the location object that is returned from the API. Latitude is a Decimal data type with a scale of 10. We expanded the Geo category to see the new getLocation function that we added (Figure 4-30).

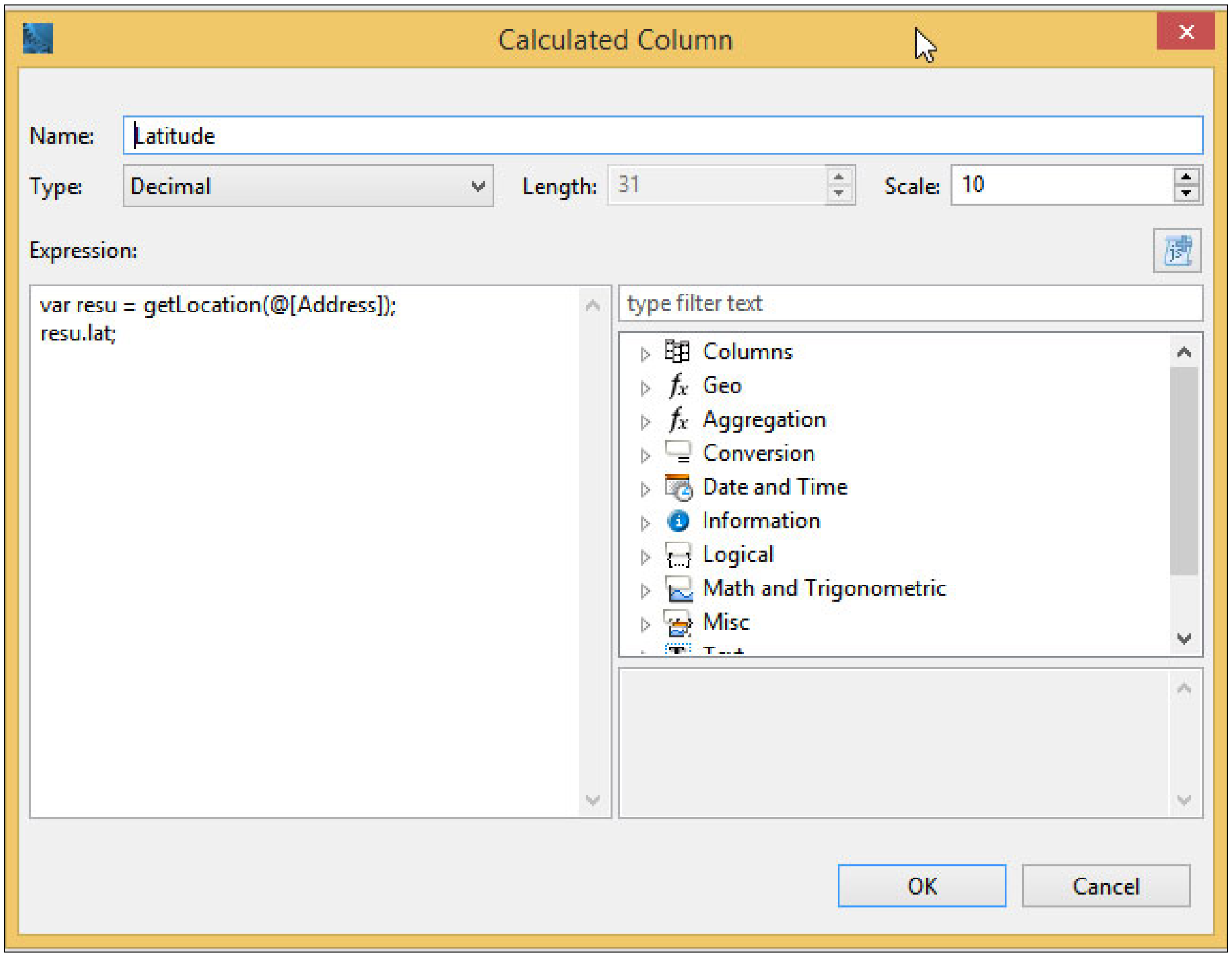


getLocation(address) - This function returns address' geo coordinates from the Google API

Figure 4-30 Calculated Column window

10. We entered the expression, as shown in Figure 4-31, and clicked **OK**.

:



Name: Latitude

Type: Decimal

Length: 31

Scale: 10

Expression:

```
var resu = getLocation(@[Address]);
resu.lat;
```

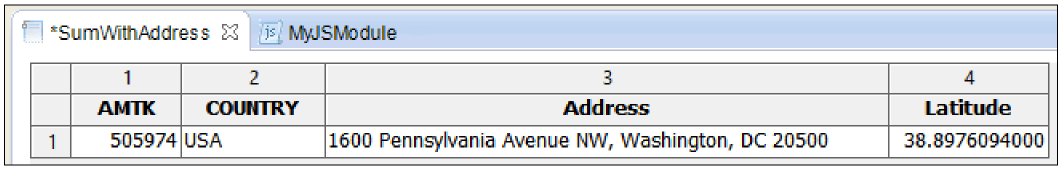
Figure 4-31   Calculated Column window with the code for the Latitude column

Example 4-3 shows the code.

Example 4-3   Expression

```
var resu = getLocation(@[Address]);
resu.lat;
```

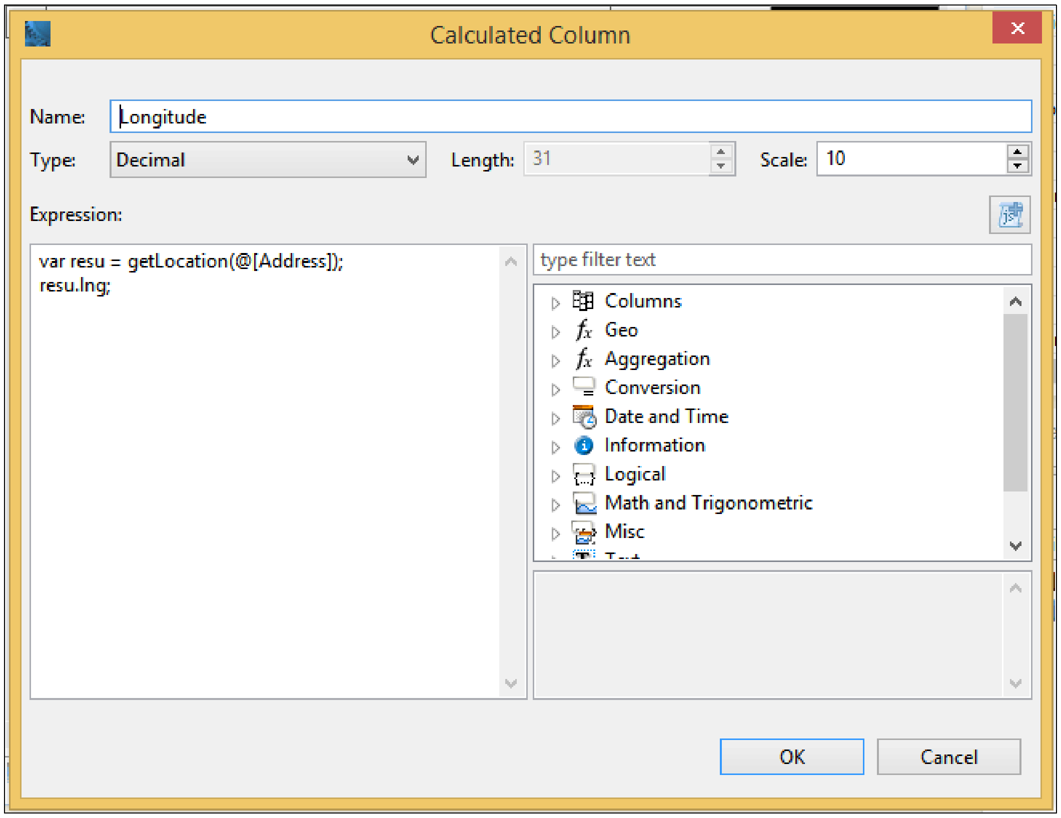
11. The Latitude calculated column for the address column is shown in the results (Figure 4-32).



	1	2	3	4
	AMTK	**COUNTRY**	**Address**	**Latitude**
1	505974	USA	1600 Pennsylvania Avenue NW, Washington, DC 20500	38.8976094000

Figure 4-32   Query results with the Latitude calculated column

12. Add another calculated column by following steps 8 - 10. Name the new calculated column `Longitude` and modify the formula to provide the longitude instead of the latitude. See Figure 4-33.



Name: Longitude
Type: Decimal
Length: 31
Scale: 10

```
var resu = getLocation(@[Address]);
resu.lng;
```

Figure 4-33 Calculated Column window with the code for the Longitude column

Example 4-4 shows the code for the Longitude calculated column.

Example 4-4 Expression for the longitude

```
var resu = getLocation(@[Address]);
resu.lng;
```

Video: You can add this query to a Visual Project and the Longitude and Latitude columns can be used to map the point on a map object. See this video:

https://youtu.be/2CTCLAenshw

4.2.3 Using a JavaScript module in a visual application

You can use JavaScript functions to further extend the expressions that are used in visual applications. In this example, a JavaScript module is loaded into a visual dashboard. This module provides two additional JavaScript functions in the Expression Designer: getLocationLat() and getLocationLng(). Then, we can use these new functions to plot an address on a map.

The example assumes that a JavaScript module exists that is called MyJSModule, which contains the code that is shown in Example 4-5.

Example 4-5 MYJSModule

```
/**
 * getLocationLat(address) - This function returns address' geo coordinates from
the Google API
 * @category Geo
 * @param address address string
 * @return lat
 */
function getLocationLat(address)
{
      var request = new RSBIHttpRequest();
        var address_en = encodeURIComponent(address.trim());
   request.open('GET','http://maps.googleapis.com/maps/api/geocode/json?address='
+ address_en + '&sensor=false', false);
   var response = '';
   request.onreadystatechange = function()
   {
         if(request.readyState === 4)
                 {
                 response = request.responseText;
                 }
   }
   request.send(null);
   var x = JSON.parse(response);
   //log(x.results[0].geometry.location);
   return x.results[0].geometry.location.lat;
}
/**
 * getLocationLng(address) - This function returns address' geo coordinates from
the Google API
 * @category Geo
 * @param address address string
 * @return lng
 */
function getLocationLng(address)
{
      var request = new RSBIHttpRequest();
        var address_en = encodeURIComponent(address.trim());
   request.open('GET','http://maps.googleapis.com/maps/api/geocode/json?address='
+ address_en + '&sensor=false', false);
   var response = '';
   request.onreadystatechange = function()
   {
         if(request.readyState === 4)
```

```
                 {
                 response = request.responseText;
                 }
   }
   request.send(null);
   var x = JSON.parse(response);
   //log(x.results[0].geometry.location);
   return x.results[0].geometry.location.lng;
}
```

This code differs slightly from the JavaScript that we used in the last example. Instead of returning the location object (and needing to parse the latitude and longitude from this object), this code uses separate functions. The getLocationLat() function returns the latitude for an address and the getLocationLng() returns the longitude for an address.

Perform the following steps:

1. Create a visual dashboard by clicking the **New Visual Dashboard** icon on the toolbar (Figure 4-34).

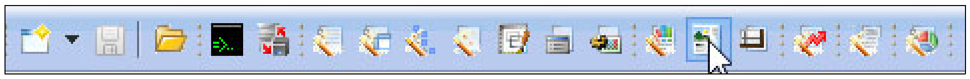


Figure 4-34 New Visual Dashboard icon in the toolbar

2. Enter the name (`MapUsingJS` in this example) for the new dashboard, and click **Finish** (Figure 4-35).

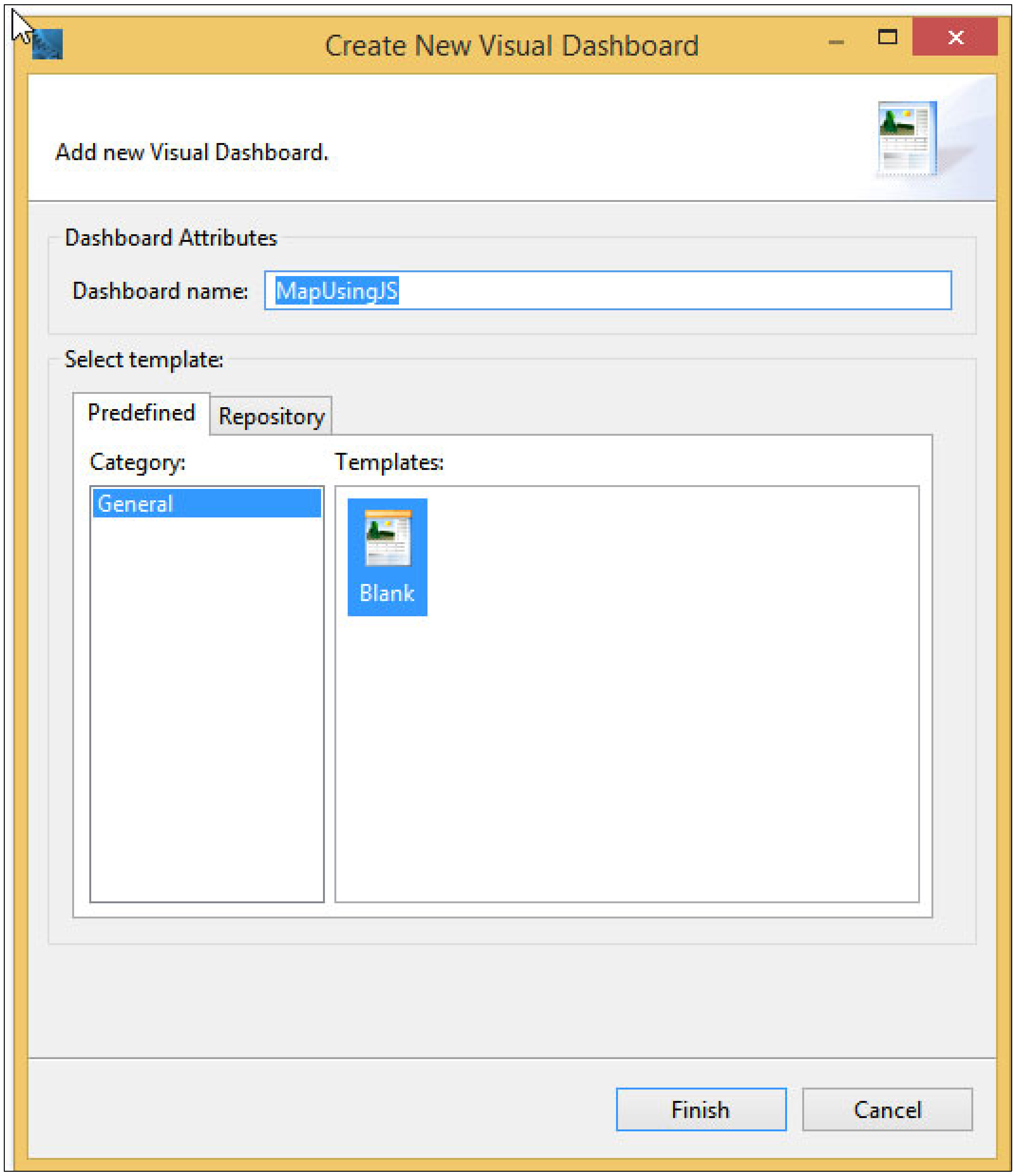


Figure 4-35 Create a visual dashboard

3. An empty visual dashboard opens for you to work with. You need to load the JavaScript module.

4. In the Project Explorer, navigate to JavaScript Modules under **Globals** in the new visual dashboard project. Right-click **JavaScript Modules** and select **Insert JavaScript Module** (Figure 4-36).

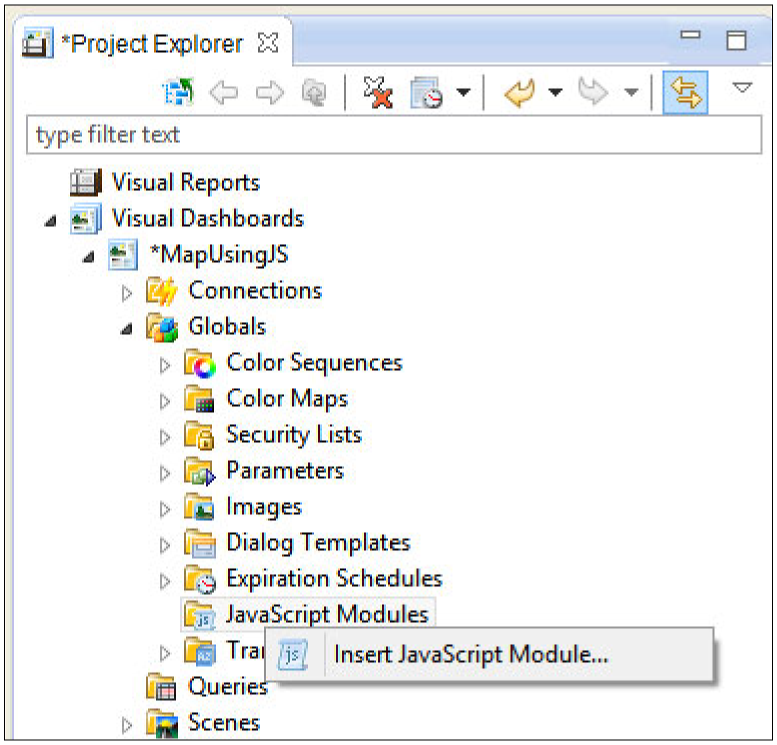
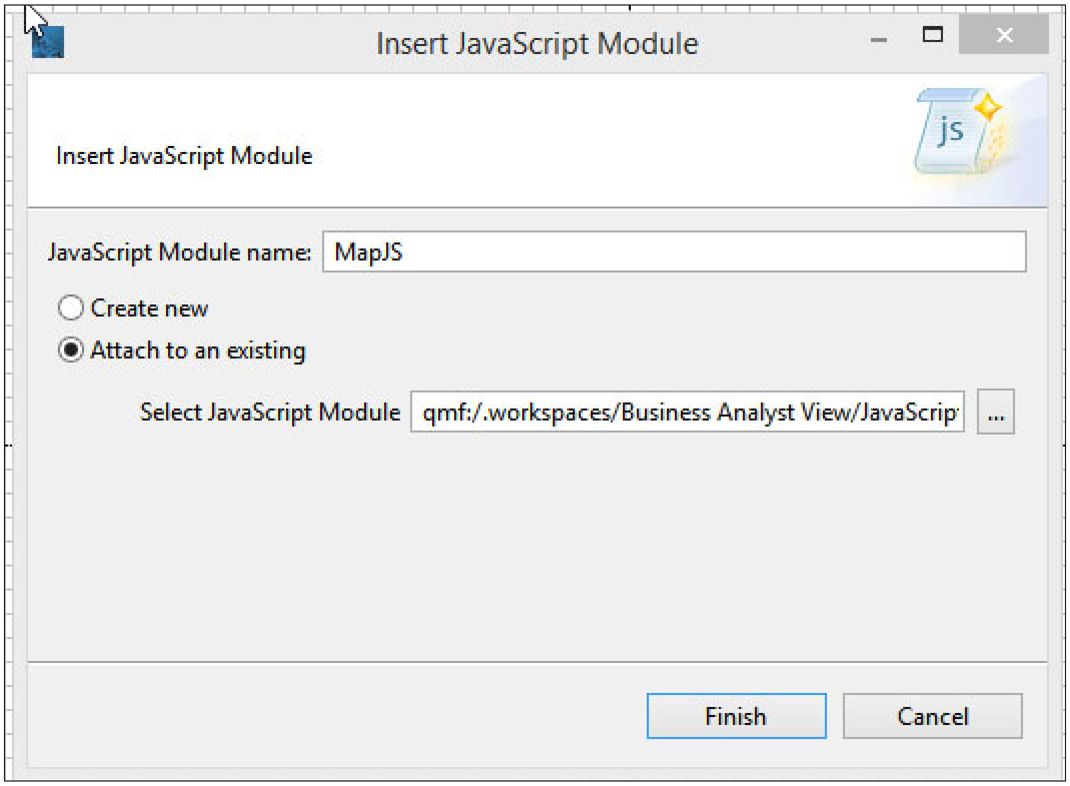


Figure 4-36 Inserting a JavaScript module into the Project Explorer

5. In the Insert JavaScript Module wizard, enter a name for the module. In this example, the JavaScript module name is MapJS. Then, click the ellipses (**...**) to browse to the JavaScript module in your repository. If needed, you can create a JavaScript module to attach to the dashboard. After the module is selected, click **Finish** (Figure 4-37).


Insert JavaScript Module

Insert JavaScript Module

JavaScript Module name: MapJS

Create new

[✓ Attach to an existing]

Select JavaScript Module qmf:/.workspaces/Business Analyst View/JavaScrip

Figure 4-37 Using the Insert JavaScript Module window with an existing module

6. The new module is loaded into the project. The new module shows in the Project Explorer under JavaScript Modules (Figure 4-38).

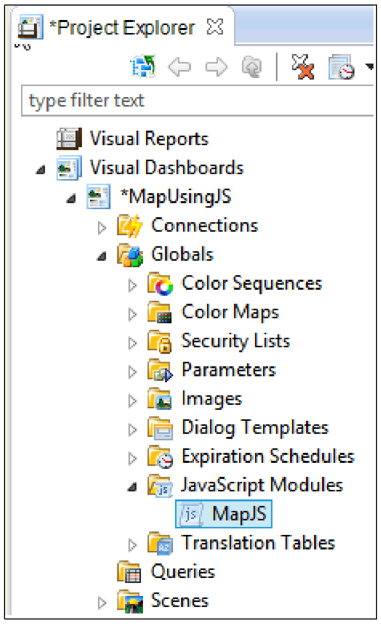


Figure 4-38 The Project Explorer after the JavaScript module is added

7. Create a global parameter to store the address value that is used in this example. In an actual dashboard, you can use an event to change the value of this parameter (for example, a text box where users type an address and the map changes when the user clicks Submit). In the Project Explorer, navigate to Parameters under Globals in your visual dashboard project. Right-click **Parameters** and select **Insert Parameter** (Figure 4-39).

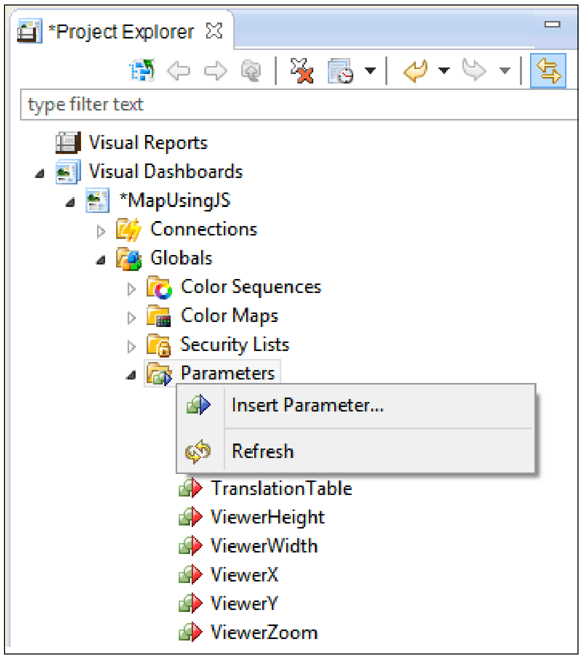
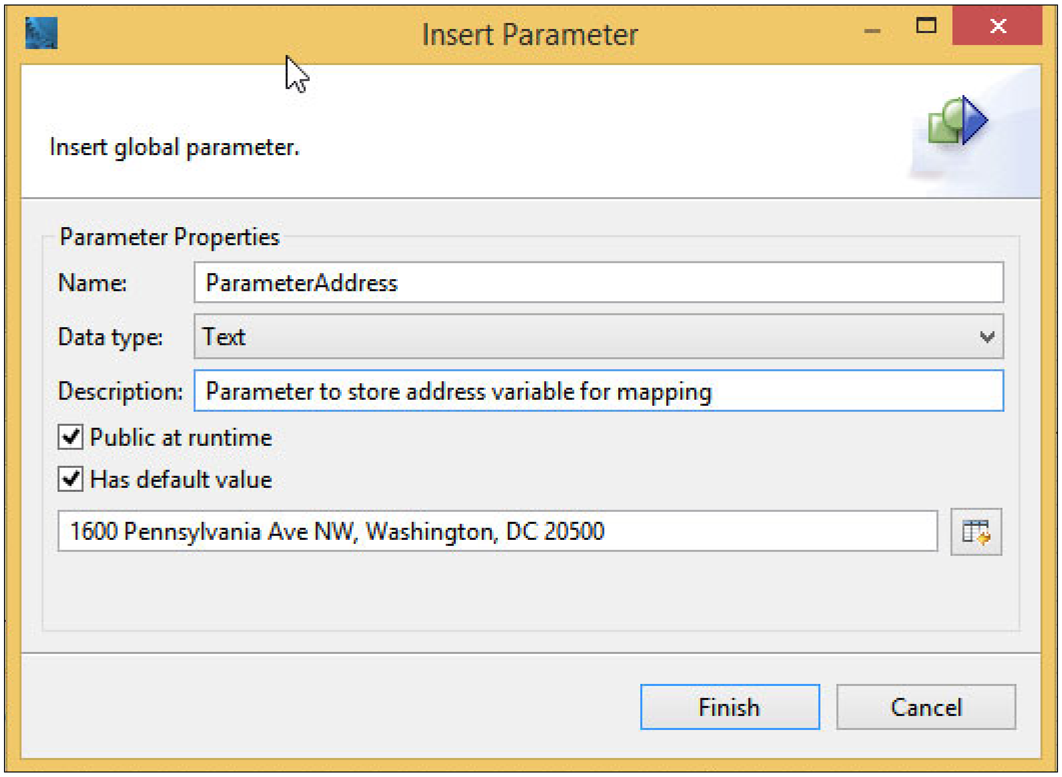


Figure 4-39 Adding a global parameter in the Project Explorer

8. On the Insert Parameter window, enter a name for the parameter, set the Data type to **Text**, enter a description, check **Public at runtime**, and check **Has default value**. We used the address of the White House for the default address. Figure 4-40 shows the finished window.



Name: ParameterAddress
Data type: Text
Description: Parameter to store address variable for mapping
☑ Public at runtime
☑ Has default value
1600 Pennsylvania Ave NW, Washington, DC 20500

Figure 4-40 New global parameter window with the ParameterAddress parameter information

9. The new parameter (with the blue icon) shows in the list of parameters with the built-in parameters (with the red icons). See Figure 4-41.

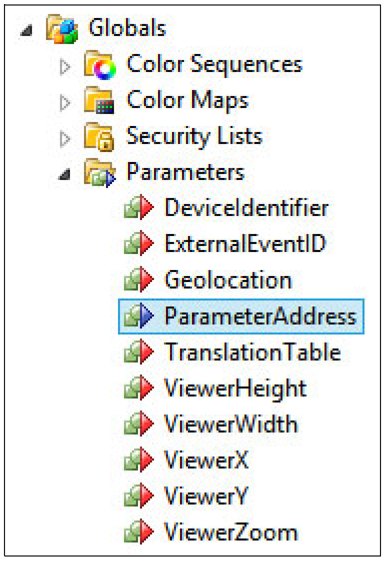


Figure 4-41 The Project Explorer with the new ParameterAddress that is added

Note: The parameter can be given any name. However, it is helpful to prepend `Parameter`, for example, ParameterAddress. You can more easily identify where it is used in a visual project.

10. Now that the JavaScript module is loaded and the address parameter with a default value is created, we use the two to plot a point on a map. In the Palette, double-click the **GoogleMap** tool under the Map folder (Figure 4-42).

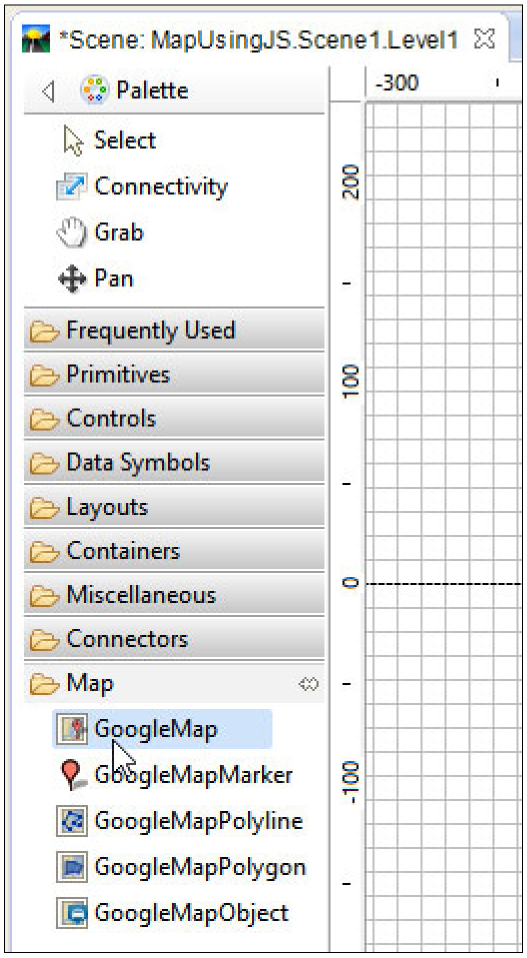
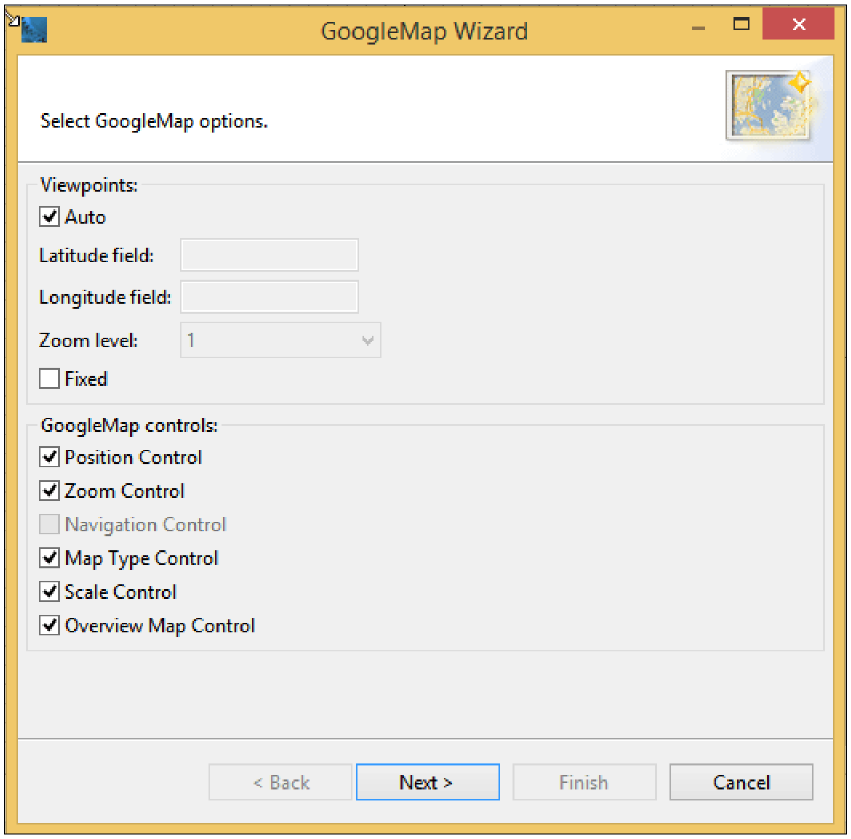


Figure 4-42 Using the GoogleMap tool in the visual dashboard palette

11. Clicking GoogleMap launches the GoogleMap Wizard. On the first page, select the view point and control options (Figure 4-43). After you select the options, click **Next**.


GoogleMap Wizard

Select GoogleMap options.

Viewpoints:

[✓] Auto

Latitude field:

Longitude field:

Zoom level: 1

[ ] Fixed

GoogleMap controls:

[✓] Position Control

[✓] Zoom Control

[ ] Navigation Control

[✓] Map Type Control

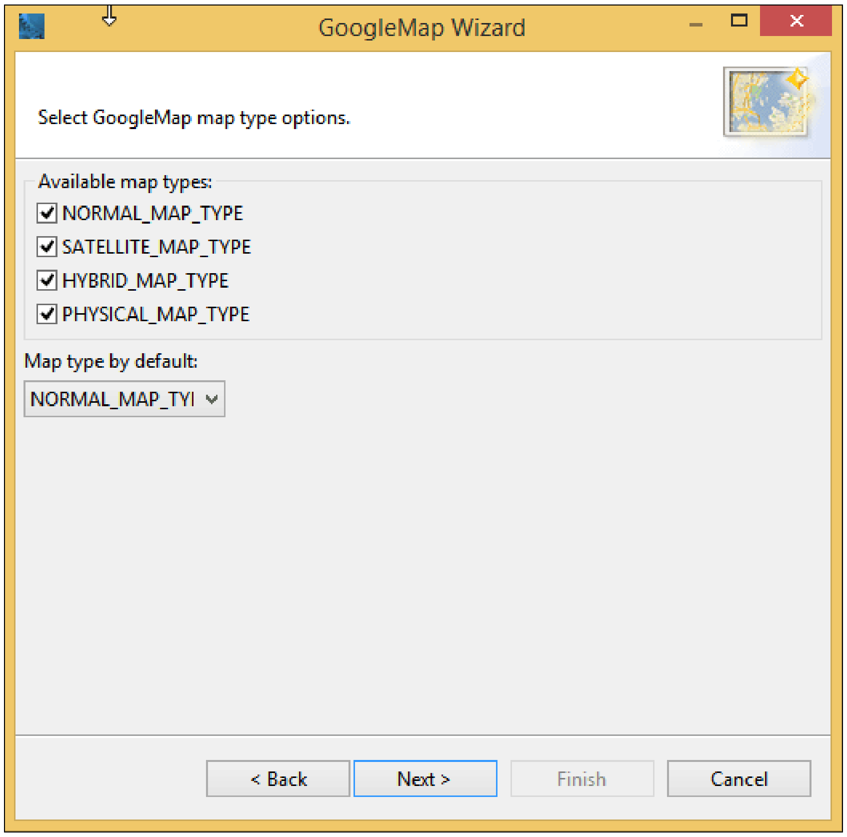
[✓] Scale Control

[✓] Overview Map Control

Figure 4-43 GoogleMap options window in the GoogleMap Wizard

Note: You can change the options after you add the map. You can adjust the values for the map object in the Properties pane.

12. Use the defaults in the map type options window (Figure 4-44). Click **Next**.



Select GoogleMap map type options.

Available map types:

- [✓] NORMAL_MAP_TYPE
- [✓] SATELLITE_MAP_TYPE
- [✓] HYBRID_MAP_TYPE
- [✓] PHYSICAL_MAP_TYPE

Map type by default: NORMAL_MAP_TYI

Figure 4-44 GoogleMap type options window in the GoogleMap Wizard

13. In this example, we selected Marker for the address parameter to use on the map. Figure 4-45 shows the Define map content window. After the options are selected, click **Next**.

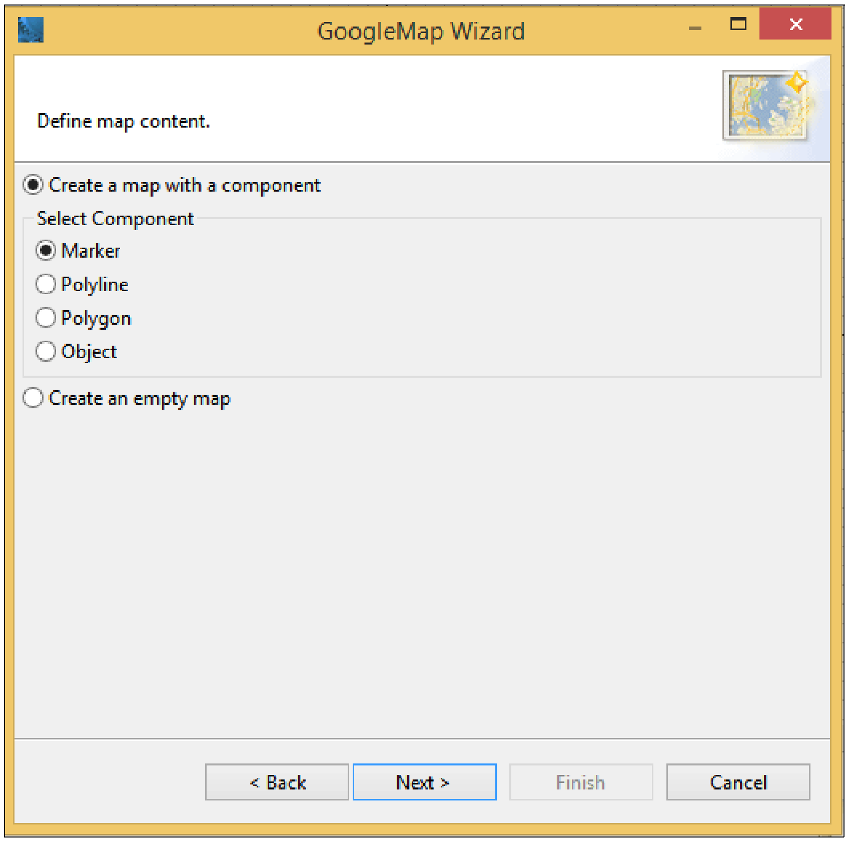
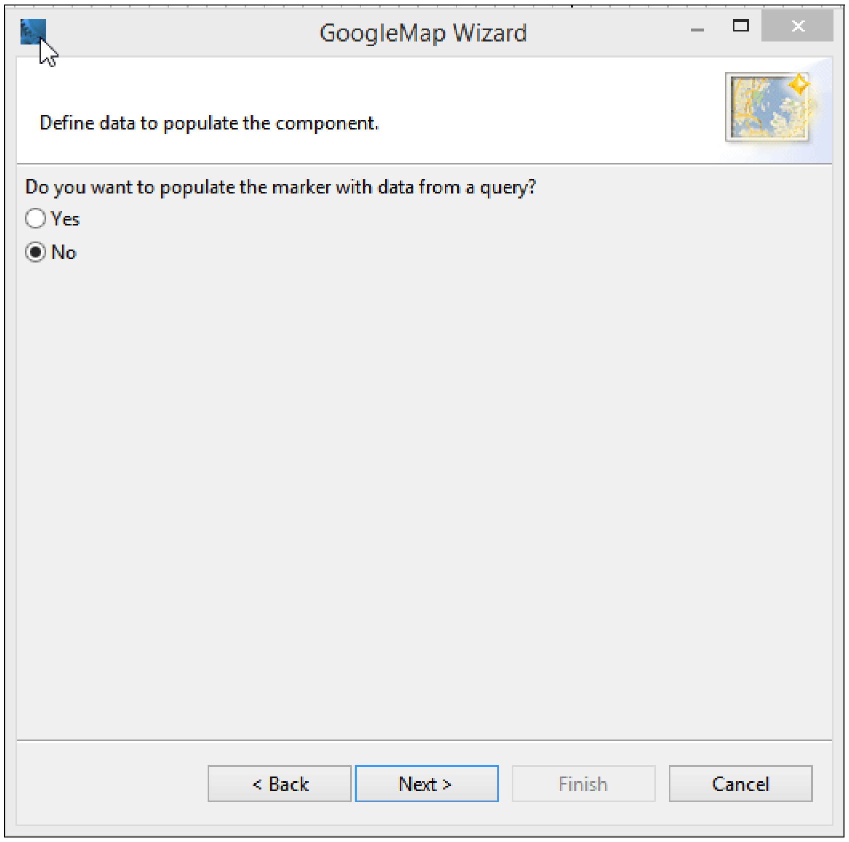


Figure 4-45 Defining map content in the GoogleMap Wizard

14. On this window, define the data to populate the component. In this example, an underlying query is not used to populate the marker. A parameter is manually populated, so click **No** (Figure 4-46).


GoogleMap Wizard

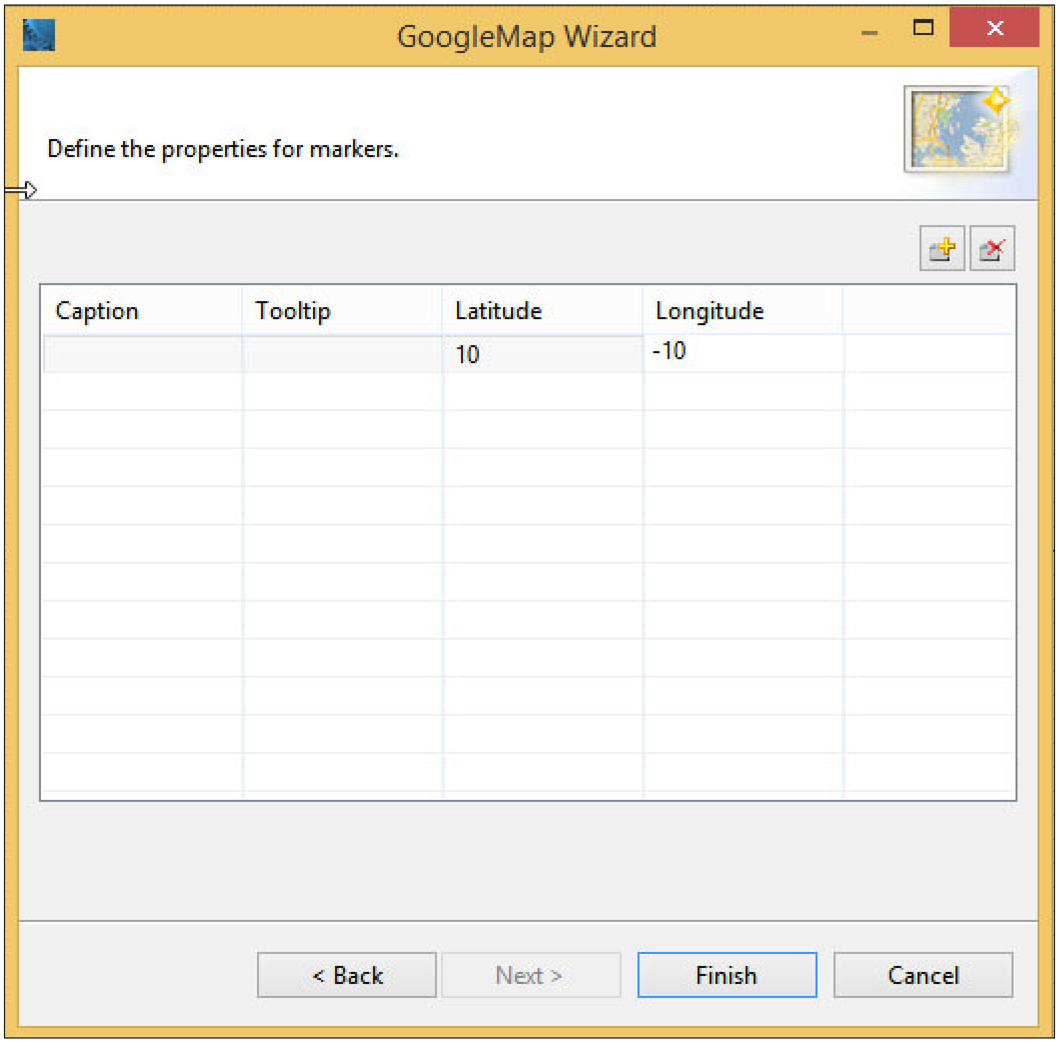
Define data to populate the component.

Do you want to populate the marker with data from a query?

Yes [✓ No]

Figure 4-46 Defining how the data is populated in the GoogleMap Wizard

15. On the next window, the properties for the marker are set. You can change the properties for the marker after you create the map. We use the Expression Designer to add the functions. Enter a numeric value for the latitude and longitude. This dummy data must be entered. If the fields are left empty, a marker is not created. After you enter values, click **Finish** (Figure 4-47).



Define the properties for markers.

Caption	Tooltip	Latitude	Longitude
		10	-10

Figure 4-47 Defining the properties for the GoogleMap marker

16. A map object opens on the window with the properties that were set by using the GoogleMap Wizard.

17. Instead of using the dummy values that were specified in the wizard, the latitude and longitude need to be generated by using the JavaScript functions on the address parameter. The latitude and longitude properties for the map marker must be set to use the JavaScript functions. Navigate to the marker object in the Project Explorer (Figure 4-48).

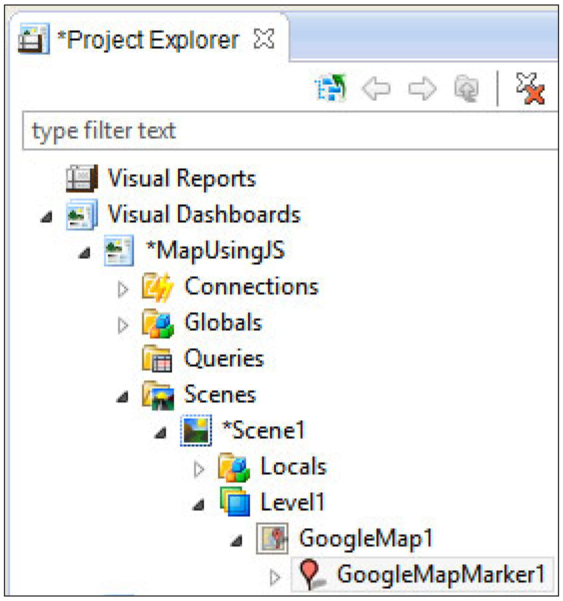
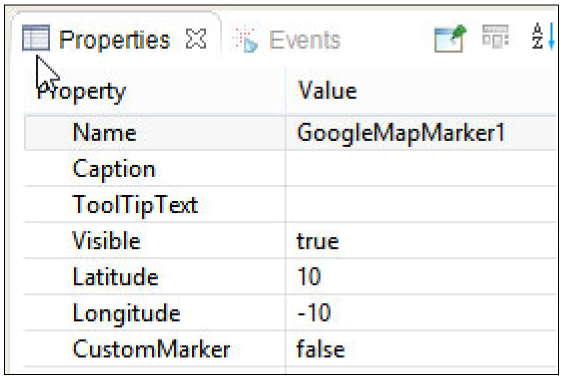


Figure 4-48   Selecting the GoogleMap marker in the Project Explorer

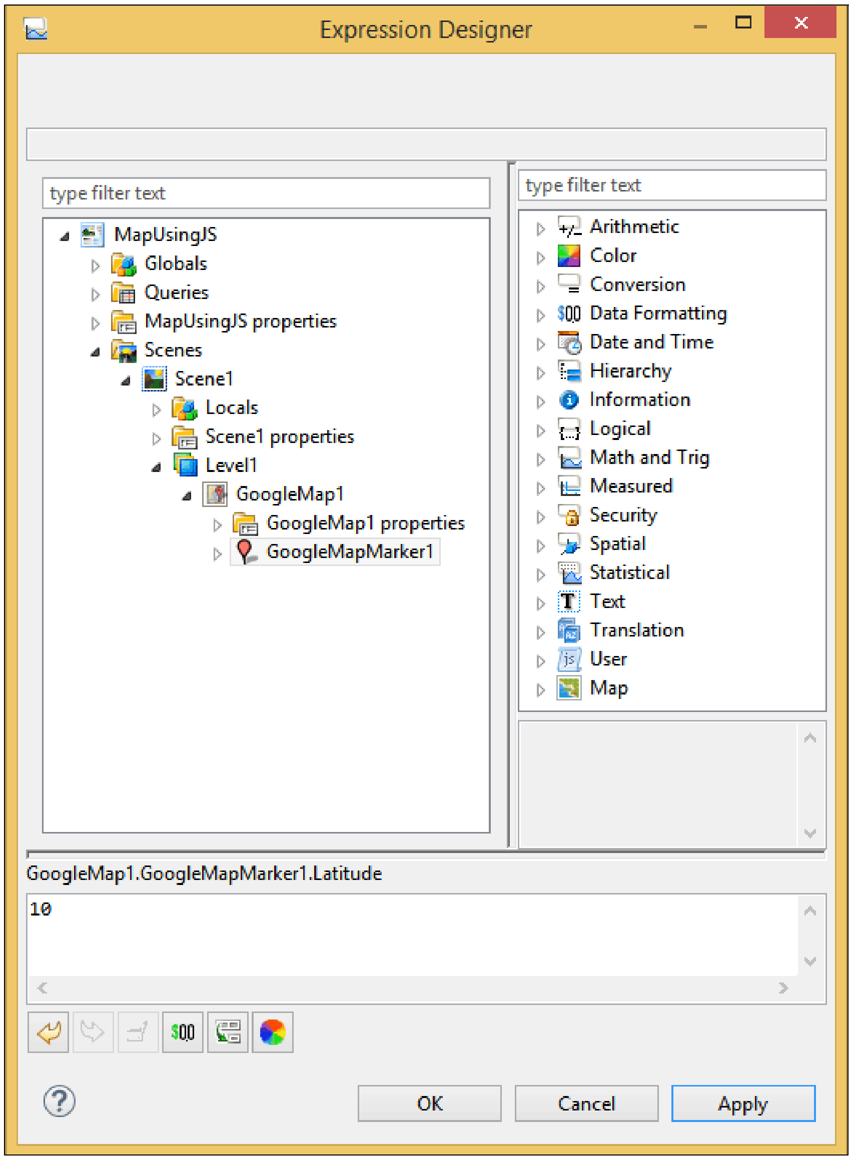
18. After this selection, the properties for the marker are listed in the Properties pane, as shown in Figure 4-49.



Property	Value
Name	GoogleMapMarker1
Caption	
ToolTipText	
Visible	true
Latitude	10
Longitude	-10
CustomMarker	false

Figure 4-49   The GoogleMap marker properties

19. Click the value (**10**) in the value box for Latitude one time. When the value is highlighted, double-click the **Latitude** value box (or click **Edit with Expression Designer**) to open the Expression Designer for the value. See Figure 4-50.

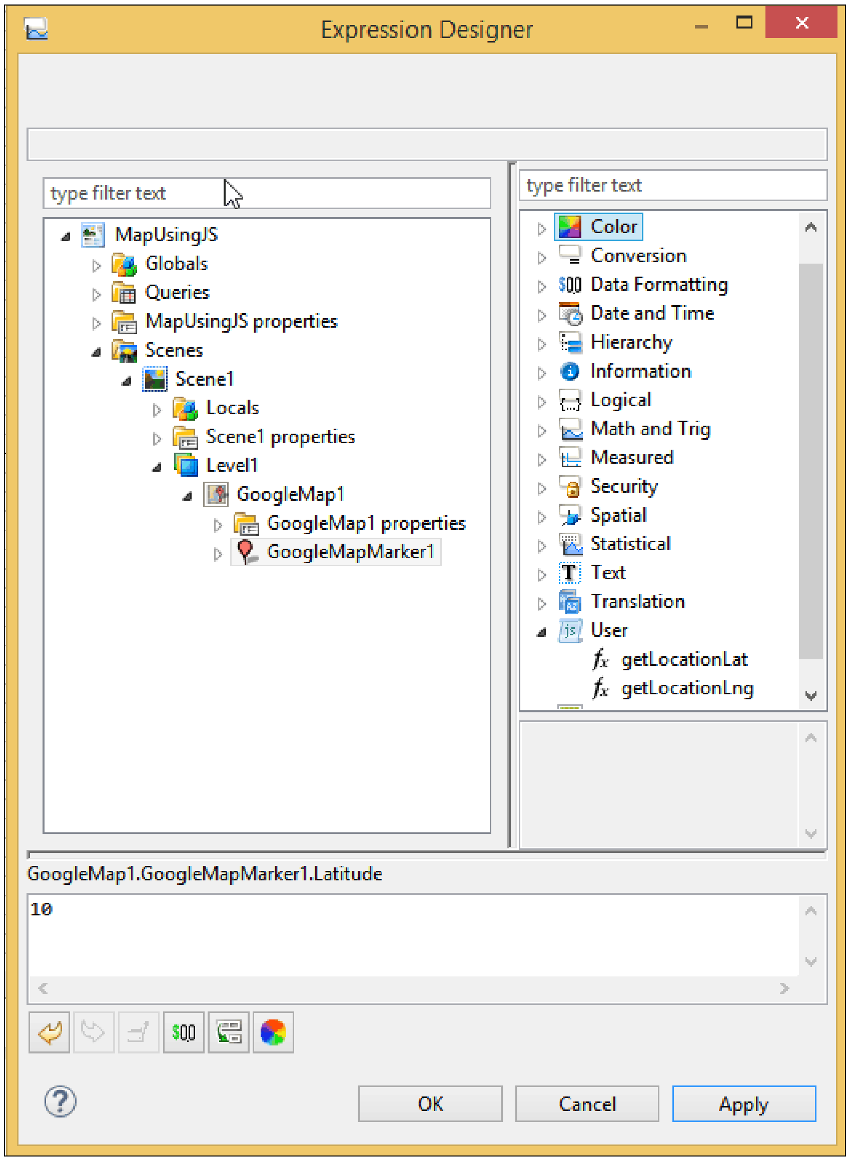


GoogleMap1.GoogleMapMarker1.Latitude

```
10
```

Figure 4-50 The Expression Designer for the Latitude value for the GoogleMap marker

20. The Expression Designer is populated with the hardcoded value. In the function list on the right, expand the **User** list of functions. The getLocationLat() and getLocationLng() functions are listed, as shown in Figure 4-51.

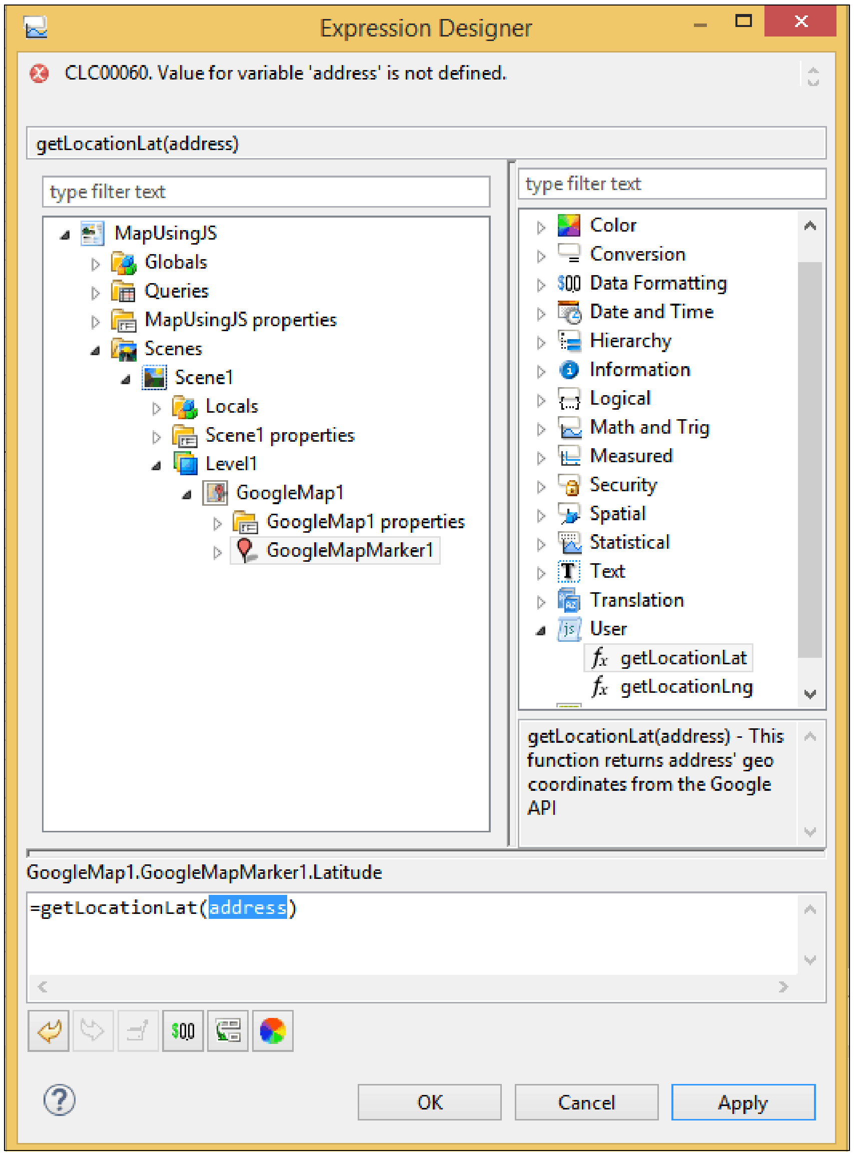


GoogleMap1.GoogleMapMarker1.Latitude

```
10
```

Figure 4-51 Expression Designer with the expanded User functions

21. In the Expression Designer window, highlight the value (**10**) and double-click the **getLocationLat()** function in the list, which replaces your value with the function (Figure 4-52).



CLC00060. Value for variable 'address' is not defined.

getLocationLat(address)

getLocationLat(address) - This function returns address' geo coordinates from the Google API

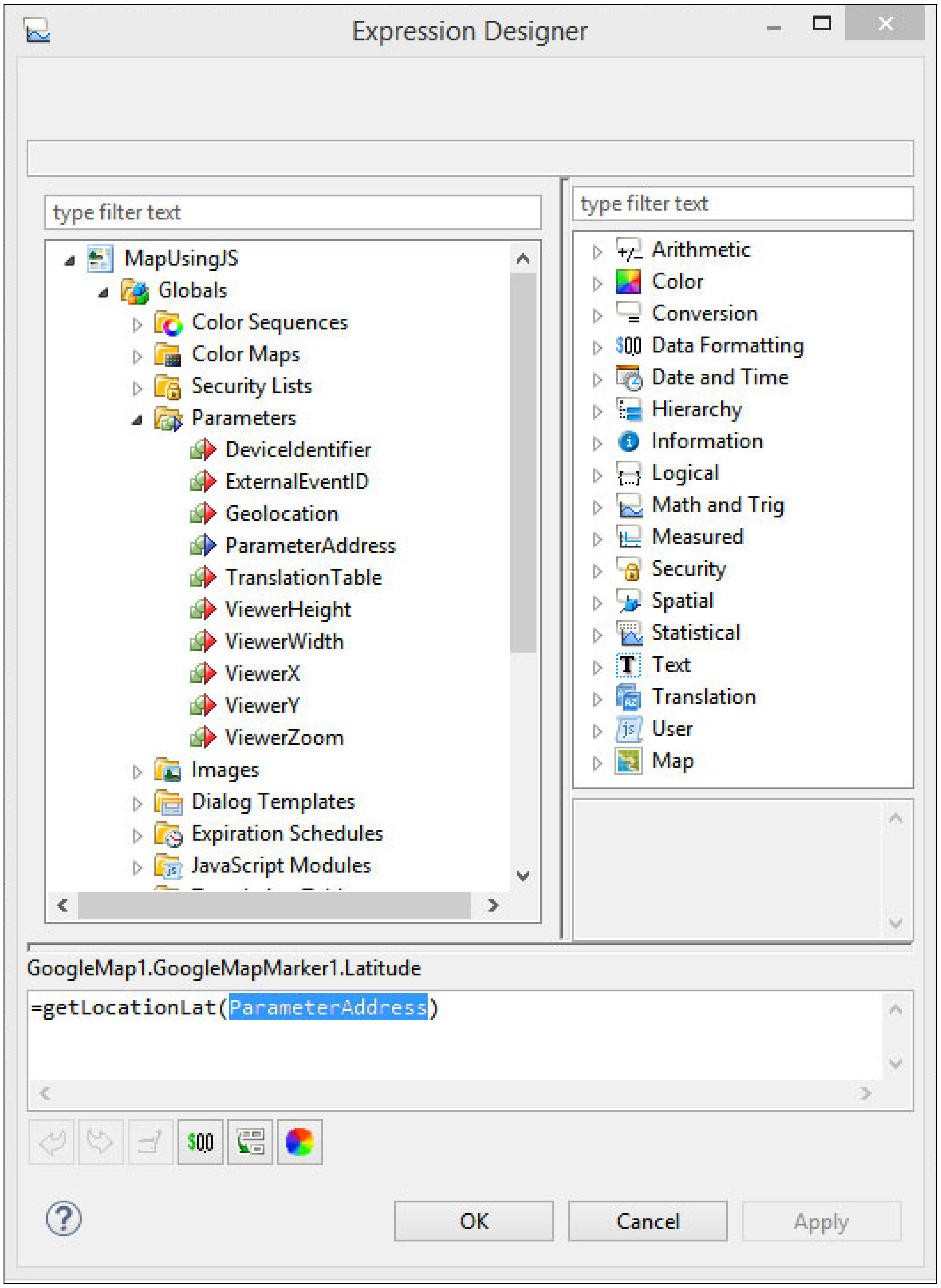
GoogleMap1.GoogleMapMarker1.Latitude

```
=getLocationLat(address)
```

Figure 4-52 Setting the getLocationLat() function to populate the Latitude value

22. The getLocationLat() function expects an address value, which is populated by the address parameter that we created earlier. Highlight **address** in the function.

23. Navigate to the ParameterAddress parameter by using **MapUsingJS** → **Globals** → **Parameters**. Double-click **ParameterAddress** to replace the address placeholder with the ParameterAddress parameter (Figure 4-53).



GoogleMap1.GoogleMapMarker1.Latitude

```
=getLocationLat(ParameterAddress)
```

Figure 4-53 Setting the getLocationLat() function to use the ParameterAddress parameter

24. After the function is set to use the parameter, click **OK**. The value for the Latitude property is set to `=getLocationLat(ParameterAddress)`.

    Use the same steps to populate the Longitude value (by using the getLocationLng function). However, the formula can be typed directly in the value box. Because the formula's format is similar to the function that is used for Latitude, you can type `=getLocationLng(ParameterAddress)` directly.

25. Click **Runtime** at the bottom of the canvas (Figure 4-54).

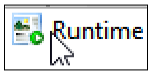


Figure 4-54 Runtime tab

26. The dashboard shows a GoogleMap with a marker on the White House (Figure 4-55).

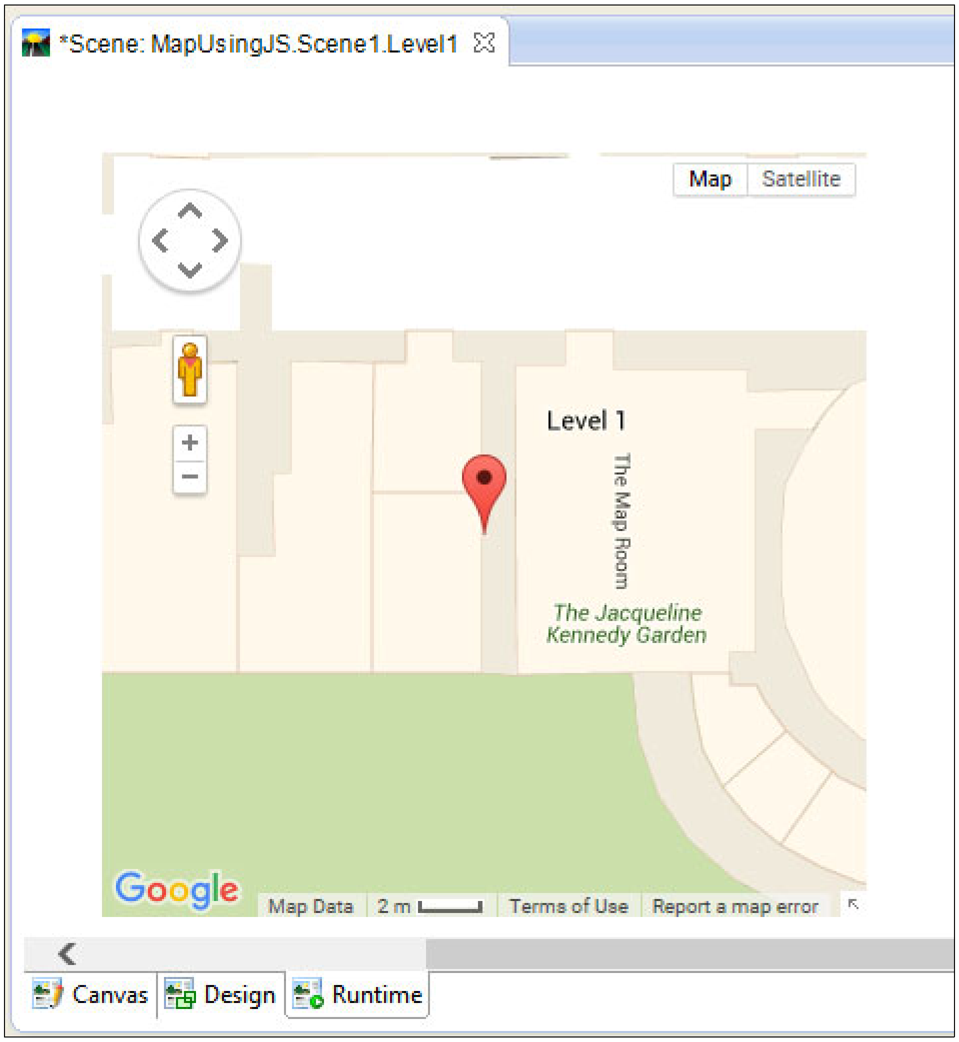


Figure 4-55 GoogleMap that shows the marker for the address in a visual dashboard

4.3 JavaScript procedures

The regular QMF procedure is sometimes called a *linear procedure* because the commands execute one at a time, in sequence. If any command fails, the procedure ends and the rest of the commands are not executed. With JavaScript procedures, QMF users can create more complex procedures and non-linear procedures.

Note: The `ERASE TABLE` command, which `DROPS` a table, does not cause the procedure to end if it fails. If the table does not exist, the "failure" is treated as a warning and the procedure continues.

In this example, a JavaScript procedure is used to check for the existence of objects before you run a query against the object. Conditional IF statements can be used in JavaScript procedures. Therefore, the procedure can run different commands that are based on meeting different conditions.

The flow of the procedure is shown:

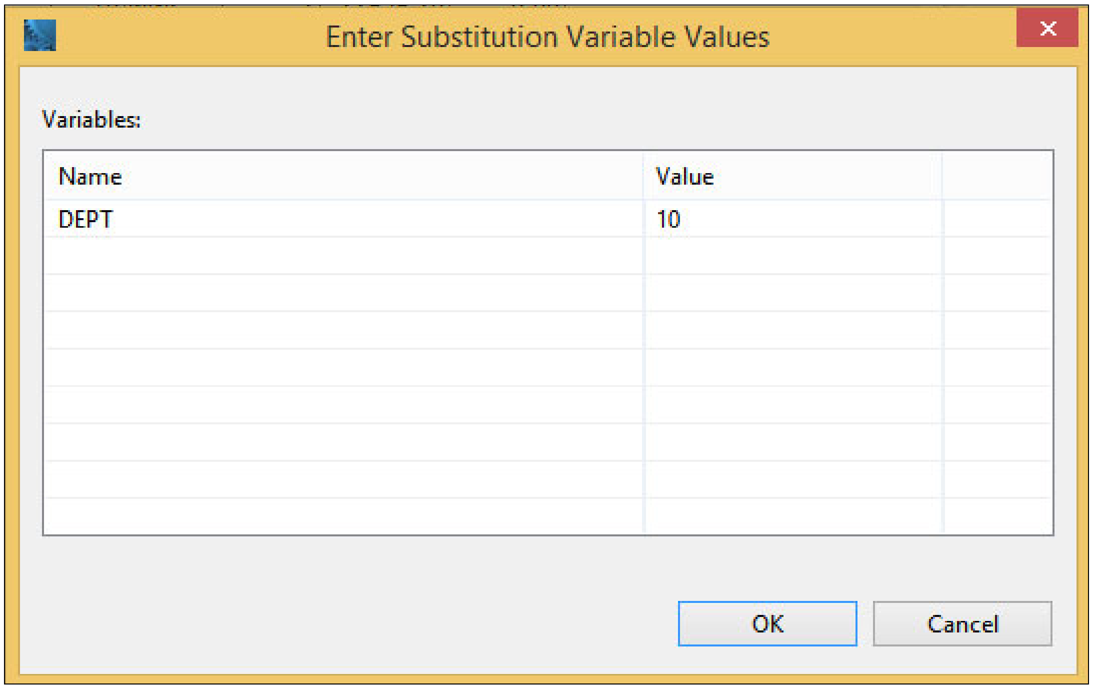
1. The procedure prompts for a department (DEPT) value and places it in a GLOBAL VARIABLE to use by the query and forms in the procedure.
2. The procedure runs the query.
3. The procedure presents the form with the results from the query and emails the form to a user.
4. If no results occur from the query, a different form with follow-up instructions is shown and the administrator is notified.

Perform the following steps:

1. Create a query that might get a result set, depending on the value that is supplied for the prompt. The following query can be used against the Q.STAFF table that is provided in the Sample repository:

   ```
   SELECT * FROM Q.STAFF WHERE DEPT = &DEPT
   ```

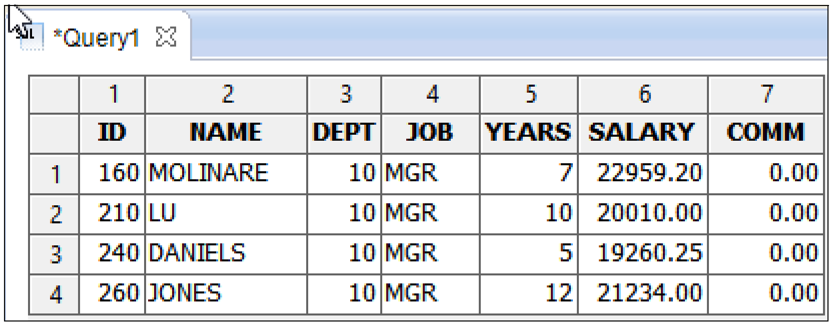
2. Run the query. A value of 10 can be used for the DEPT substitution variable (Figure 4-56).



Name	Value
DEPT	10

Figure 4-56 Enter Substitution Variable Values window

3. Figure 4-57 shows the results.



	1	2	3	4	5	6	7
	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1	160	MOLINARE	10	MGR	7	22959.20	0.00
2	210	LU	10	MGR	10	20010.00	0.00
3	240	DANIELS	10	MGR	5	19260.25	0.00
4	260	JONES	10	MGR	12	21234.00	0.00

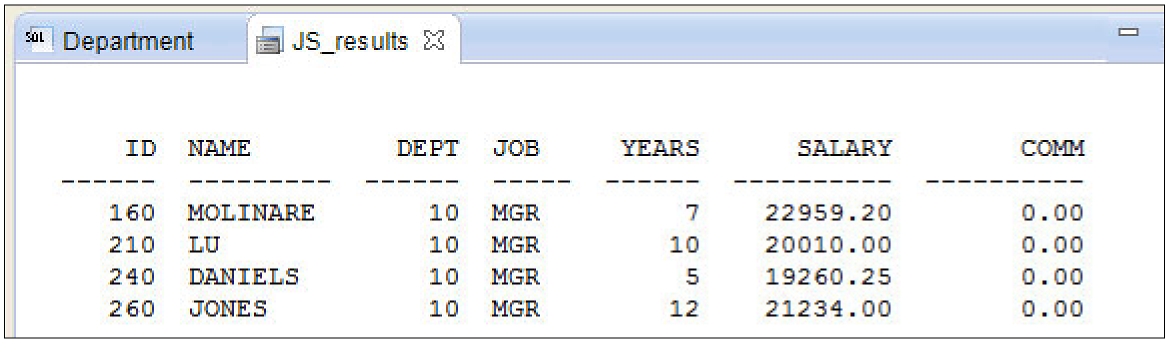
Figure 4-57 Results of the query against the Q.STAFF table

4. Save the query in the repository. Enter a name of `Department`.

5. The next step is to create two forms, one form for the case with a result set and one form for the case with no result set. With the query open, click the **New Form** icon in the toolbar (Figure 4-58).

Figure 4-58 New Form icon

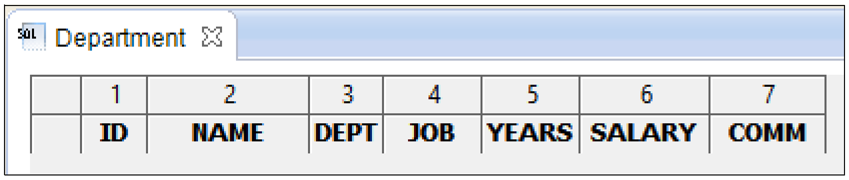
6. A basic form, which is based on the result set, is generated, as shown in Figure 4-59.



```
    ID  NAME        DEPT  JOB    YEARS     SALARY       COMM
------  ---------  ------ -----  ------  ----------  ----------
   160  MOLINARE       10  MGR        7    22959.20        0.00
   210  LU             10  MGR       10    20010.00        0.00
   240  DANIELS        10  MGR        5    19260.25        0.00
   260  JONES          10  MGR       12    21234.00        0.00
```

Figure 4-59 Form that is based on the query results

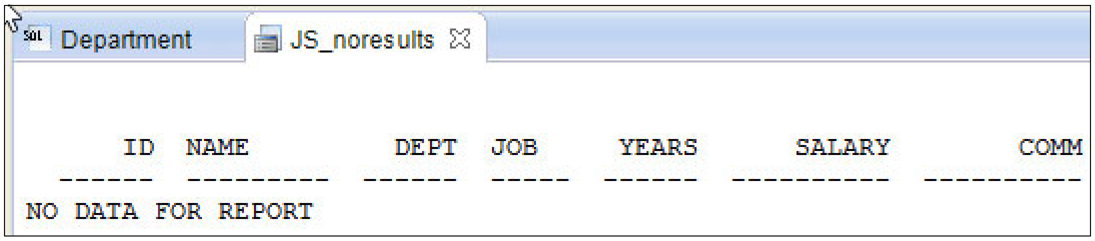
7. Save the form in the repository with a name of JS_results.
8. Run the Department query again. This time, use a value for the DEPT variable that does not exist, for example, 1000. This query produces an empty result set (Figure 4-60).



	1	2	3	4	5	6	7
	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM

Figure 4-60 Empty result set

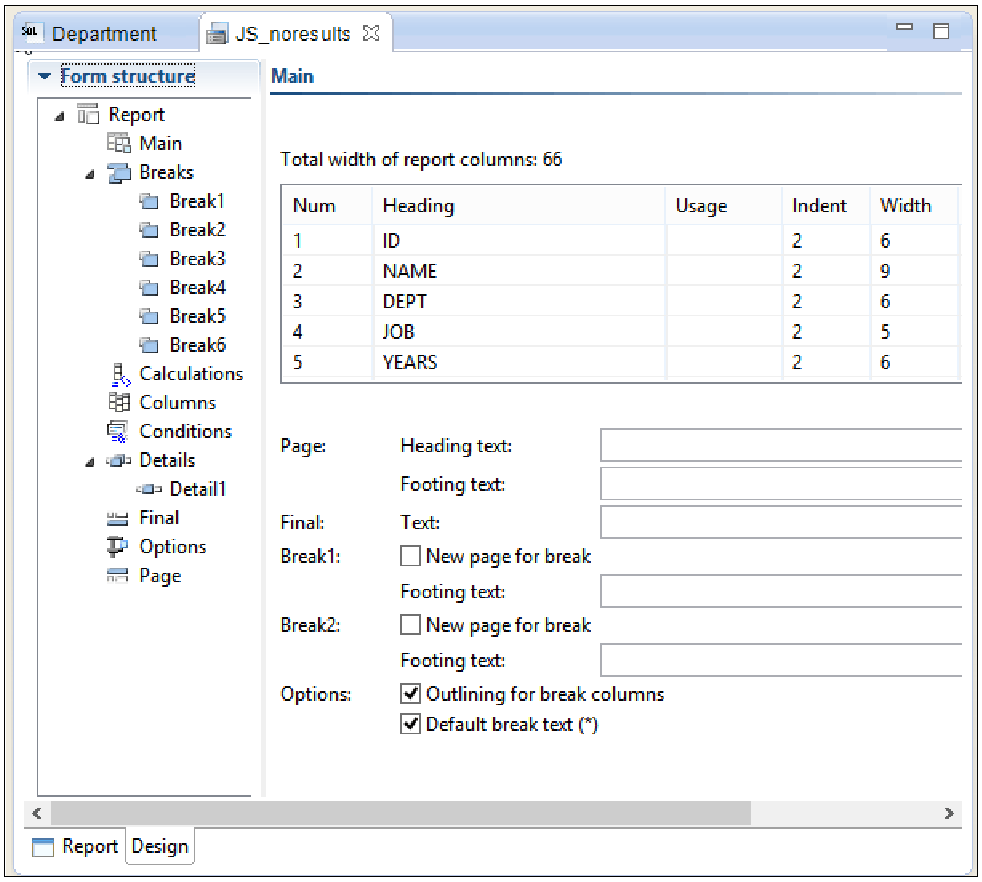
9. Click **New Form** again. Figure 4-61 shows the new form for the empty result set.



```
    ID  NAME        DEPT  JOB    YEARS     SALARY       COMM
------  ---------  ------ -----  ------  ----------  ----------
NO DATA FOR REPORT
```

Figure 4-61 New form for the empty result set

10. Modify the report to provide more information to a user that receives an empty report. Click the **Design** tab (lower-left area of the window) of the form, as shown in Figure 4-62.



Main

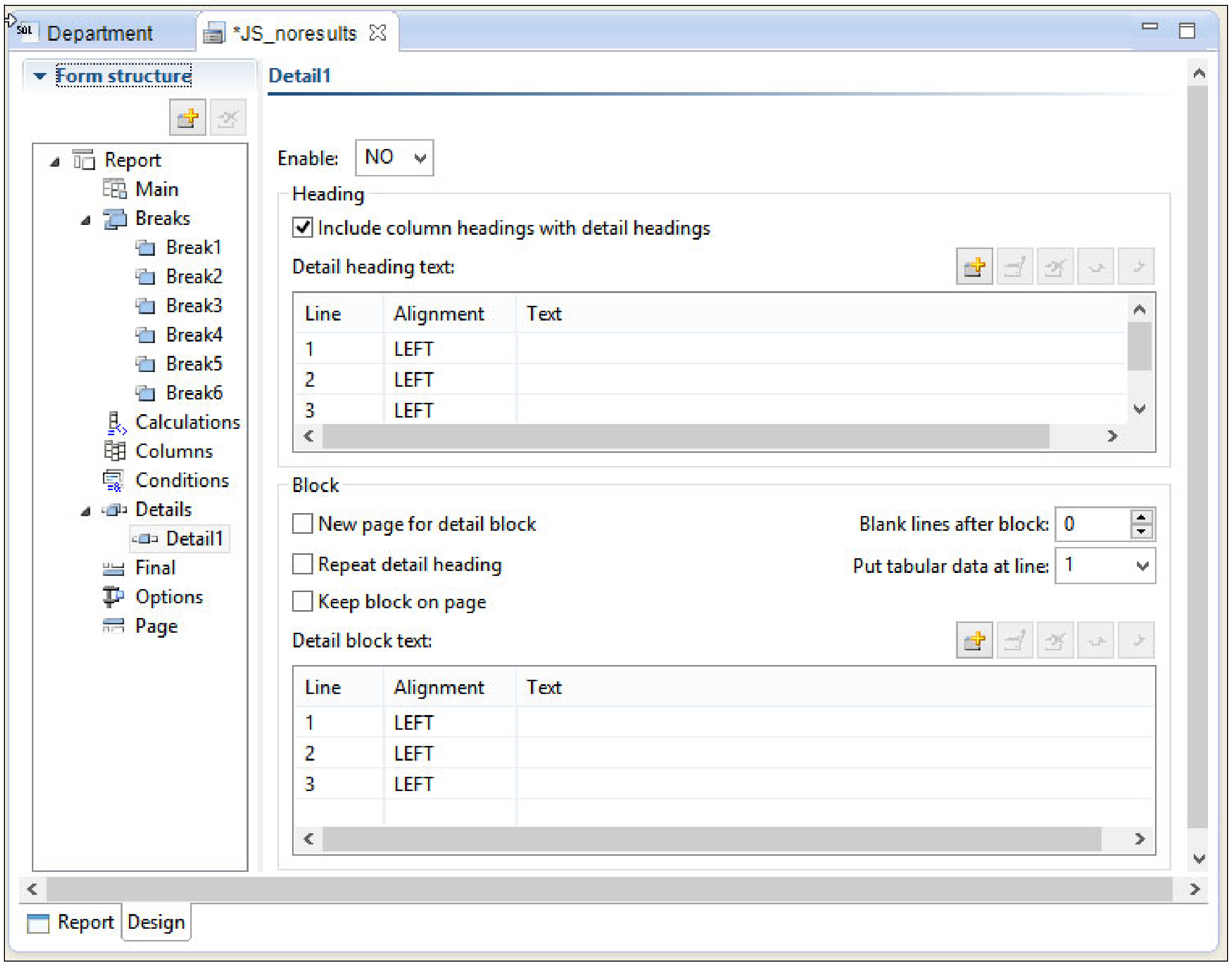
Total width of report columns: 66

Num	Heading	Usage	Indent	Width
1	ID		2	6
2	NAME		2	9
3	DEPT		2	6
4	JOB		2	5
5	YEARS		2	6

Page:	Heading text:	
	Footing text:	
Final:	Text:	
Break1:	☐ New page for break	
	Footing text:	
Break2:	☐ New page for break	
	Footing text:	
Options:	✓ Outlining for break columns	
	✓ Default break text (*)	

Figure 4-62 Design view for the form

11. In the Form structure pane, expand **Details** and click **Detail1**. While you are in this window, change Enable from YES to **NO** (Figure 4-63).


Department | *JS_noresults

Form structure

- Report
  - Main
  - Breaks
    - Break1
    - Break2
    - Break3
    - Break4
    - Break5
    - Break6
  - Calculations
  - Columns
  - Conditions
  - Details
    - Detail1
  - Final
  - Options
  - Page

Detail1

Enable: NO

Heading

[✓] Include column headings with detail headings

Detail heading text:

Line	Alignment	Text
1	LEFT	
2	LEFT	
3	LEFT	

Block

[ ] New page for detail block

Blank lines after block: 0

[ ] Repeat detail heading

Put tabular data at line: 1

[ ] Keep block on page

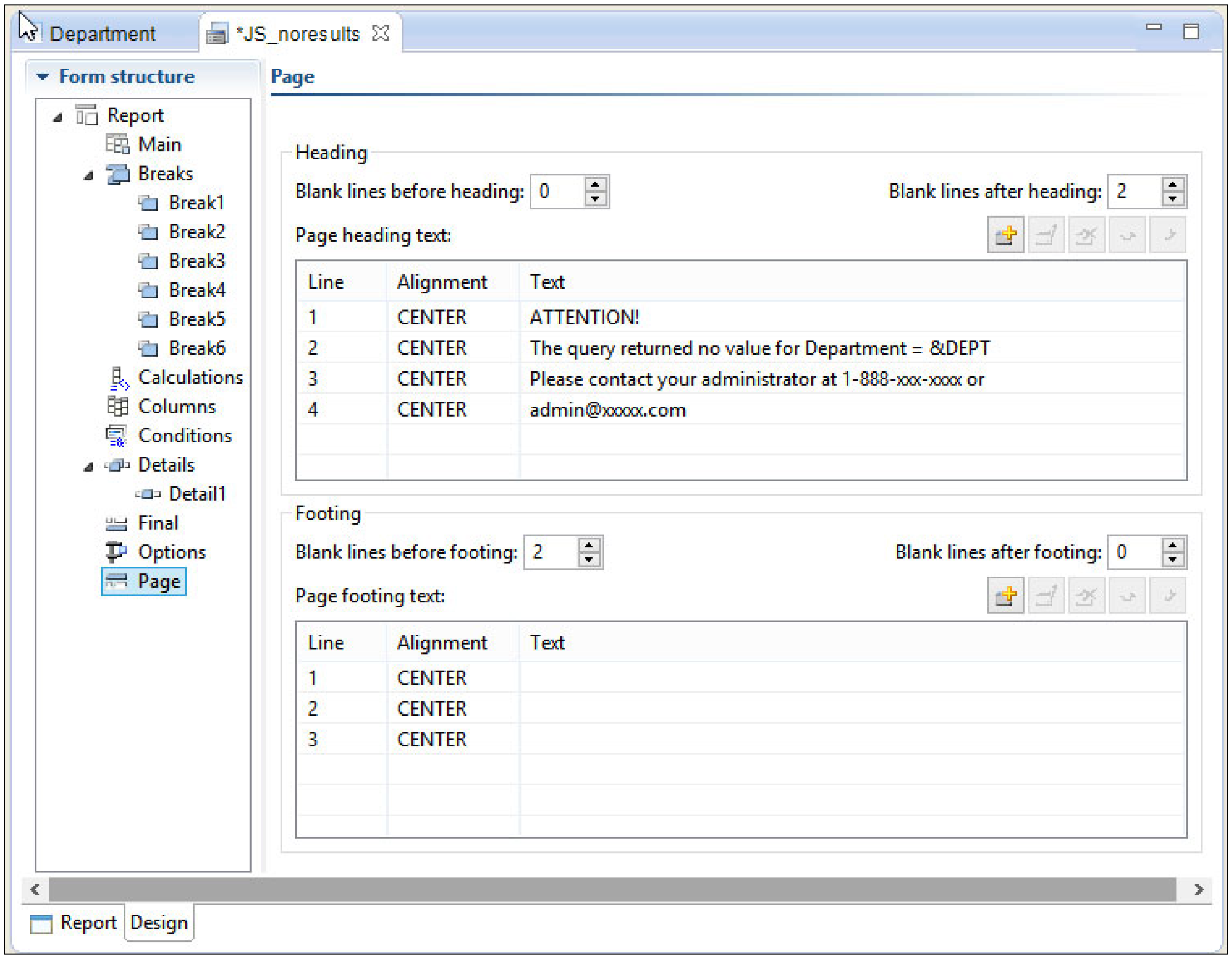
Detail block text:

Line	Alignment	Text
1	LEFT	
2	LEFT	
3	LEFT	

Report | Design

Figure 4-63 Detail1 section of the form design

12. In the Form structure pane, click **Page** for the page settings. In the Page heading text area, enter the text to match Figure 4-64. Ensure that you add a fourth line.



Page

Heading

Blank lines before heading: 0

Blank lines after heading: 2

Page heading text:

Line	Alignment	Text
1	CENTER	ATTENTION!
2	CENTER	The query returned no value for Department = &DEPT
3	CENTER	Please contact your administrator at 1-888-xxx-xxxx or
4	CENTER	admin@xxxxx.com

Footing

Blank lines before footing: 2

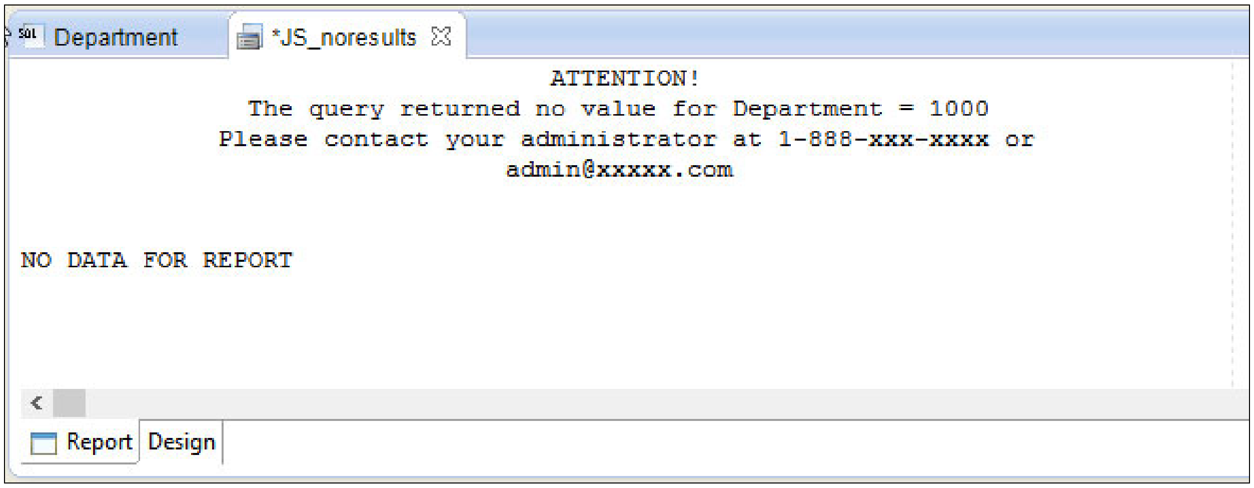
Blank lines after footing: 0

Page footing text:

Line	Alignment	Text
1	CENTER	
2	CENTER	
3	CENTER	

Figure 4-64   *Changing the page settings for the form*

13. Click the **Report** tab (lower-left area of the window). Type the substitution variable for &DEPT (1000). Figure 4-65 shows the report.



```
                         ATTENTION!
        The query returned no value for Department = 1000
      Please contact your administrator at 1-888-xxx-xxxx or
                       admin@xxxxx.com


NO DATA FOR REPORT
```

Figure 4-65   *Viewing the changes that were made to the form*

14. Save the report to the repository. Type `JS_noresults` for the name.

15. Click the **New Procedure** icon in the toolbar (Figure 4-66).

Figure 4-66 New Procedure icon

16. An empty procedure window opens (Figure 4-67).

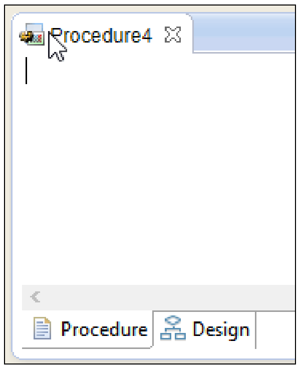


Figure 4-67 Empty procedure window

17. Paste the following procedure (Example 4-6) in the empty procedure window.

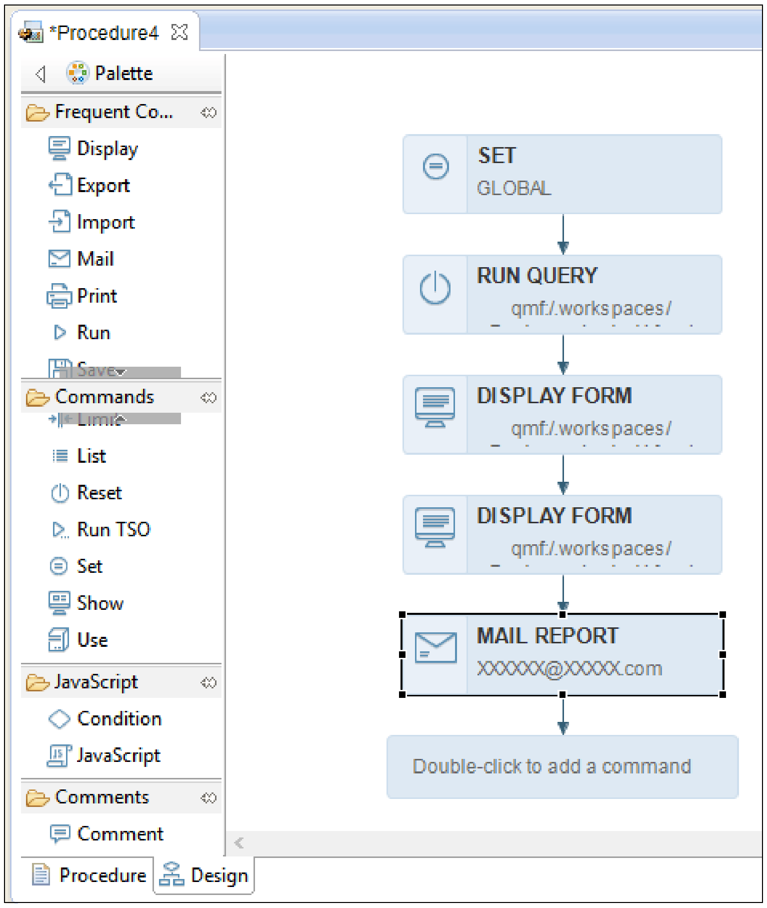
Example 4-6 Procedure

```
SET GLOBAL (DEPT=&Please_Enter_Dept_Value
RUN QUERY "qmf:/.workspaces/Business Analyst View/JavaScript/Department"
DISPLAY FORM "qmf:/.workspaces/Business Analyst View/JavaScript/JS_results"
DISPLAY FORM "qmf:/.workspaces/Business Analyst View/JavaScript/JS_noresults"
MAIL REPORT TO XXXXX@XXXXX.com
+ (BODY="The report failed with this value of DEPT"
+ FORMAT=HTML FROM=XXXXXX@XXXXX.com
+ SMTPSERVER=MAIL.XXXXXXXXXX.COM
+ SUBJECT="JSQ Report Failure"
```

18. Change `XXXXX@XXXXX.com` to the email address that you want to use for the example. Change the `MAIL.XXXXXXXXXX.COM` value to match your Simple Mail Transfer Protocol (SMTP) server.

Important: Paths, such as `"qmf:/.workspaces/Business Analyst View/JavaScript/"`, match the paths that we used to save the objects in our repository. If you used different paths, you must modify the paths, also.

19. When you click the **Design** tab, the following procedure is shown (Figure 4-68).



SET
GLOBAL

RUN QUERY
qmf:/.workspaces/

DISPLAY FORM
qmf:/.workspaces/

DISPLAY FORM
qmf:/.workspaces/

MAIL REPORT
XXXXXX@XXXXX.com

Double-click to add a command

Figure 4-68 Design view for the specific procedure

20. Insert a condition to check the results of the `RUN QUERY` step. In the Palette, click **Condition** under the JavaScript section to activate the item. While Condition is active, click the **DISPLAY FORM** task in the procedure (Figure 4-69).

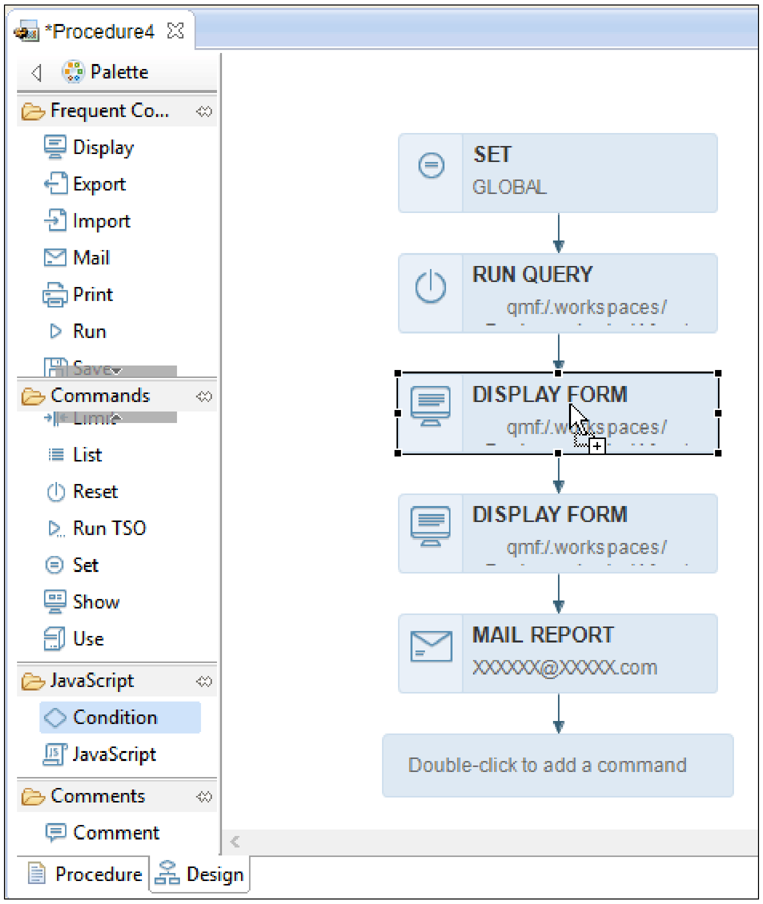
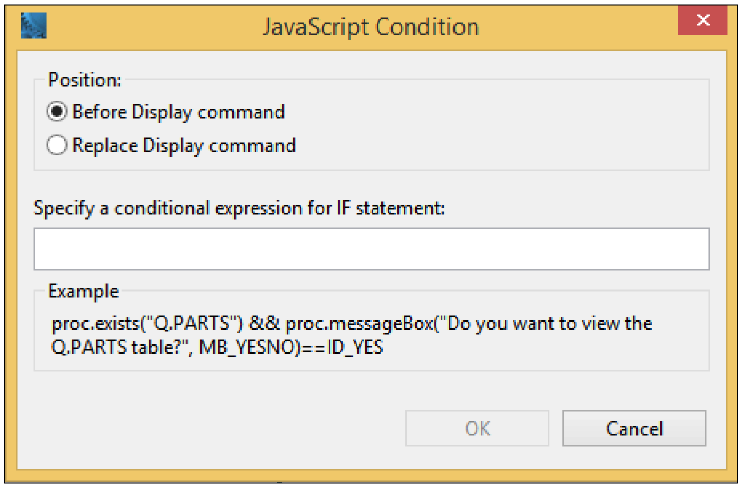


Figure 4-69 Inserting a JavaScript Condition task into the procedure

21. The JavaScript Condition window opens so that you can insert the step into the procedure (Figure 4-70).



Position:
(●) Before Display command
( ) Replace Display command

Specify a conditional expression for IF statement:

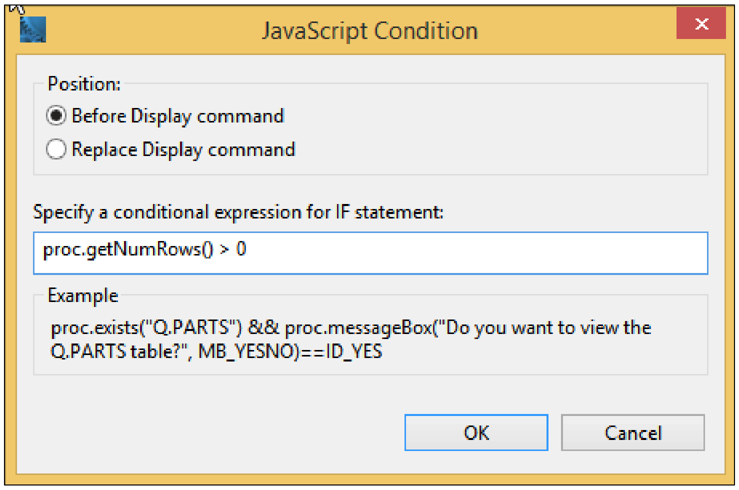
Example

proc.exists("Q.PARTS") && proc.messageBox("Do you want to view the Q.PARTS table?", MB_YESNO)==ID_YES

Figure 4-70 JavaScript Condition window

22. Leave **Before Display command** selected because the step is inserted before the forms are generated. For the condition, the built-in proc.getNumRows() function is used (Figure 4-71). This function is used to check whether the results (number of rows) from the query run are more than 0. The formula is shown:

```
proc.getNumRows() > 0
```



Position:
- [✓ Before Display command]
- Replace Display command

Specify a conditional expression for IF statement:

proc.getNumRows() > 0

Example

proc.exists("Q.PARTS") && proc.messageBox("Do you want to view the Q.PARTS table?", MB_YESNO)==ID_YES

Figure 4-71 JavaScript Condition window with the conditional expression

23. Figure 4-72 shows the resulting procedure structure.

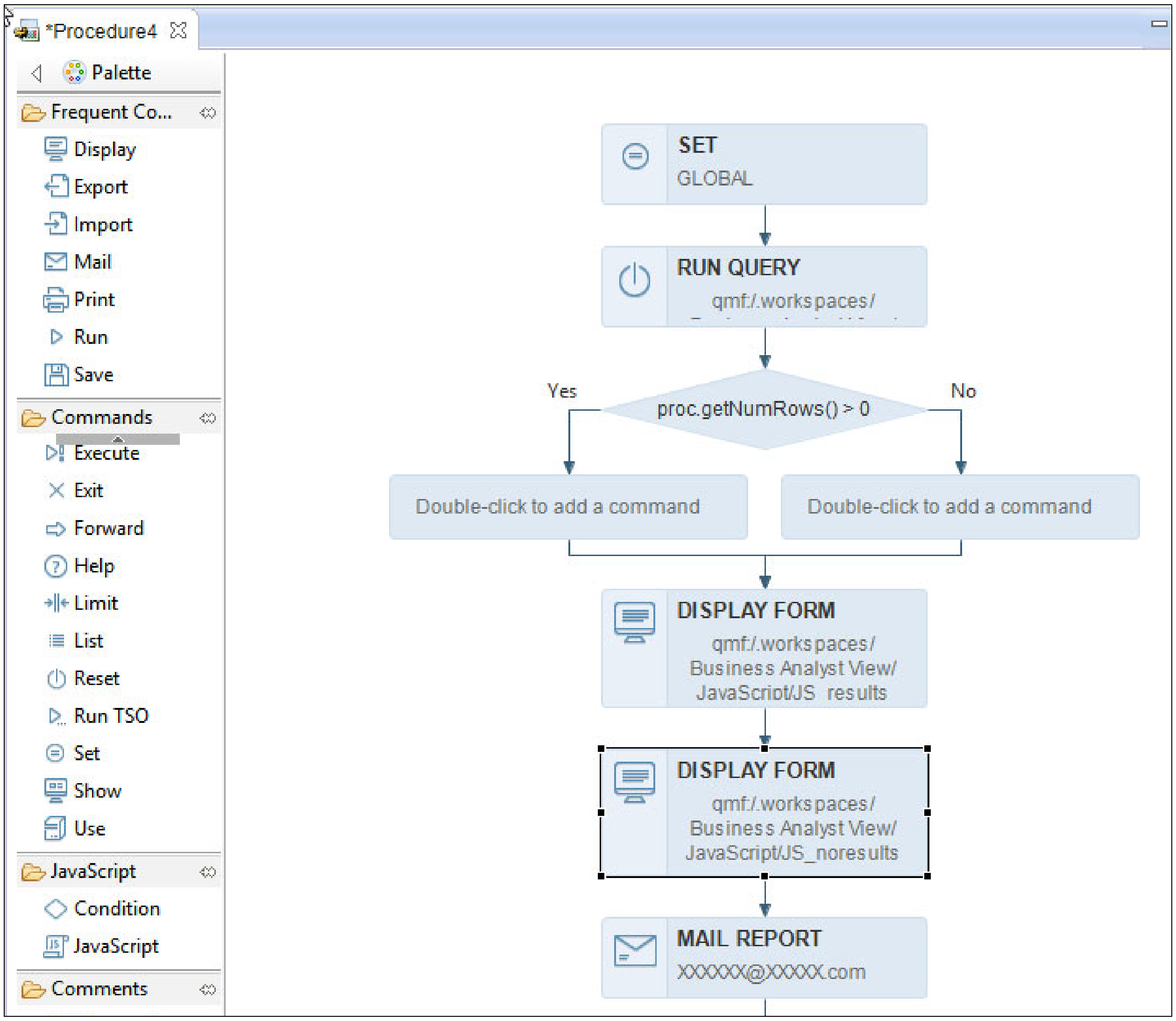


Figure 4-72 Resulting procedure layout after the JavaScript condition is added

24. The DISPLAY FORM for a query with results can be dragged above the empty node for the Yes result of the JavaScript Condition Node (Figure 4-73).

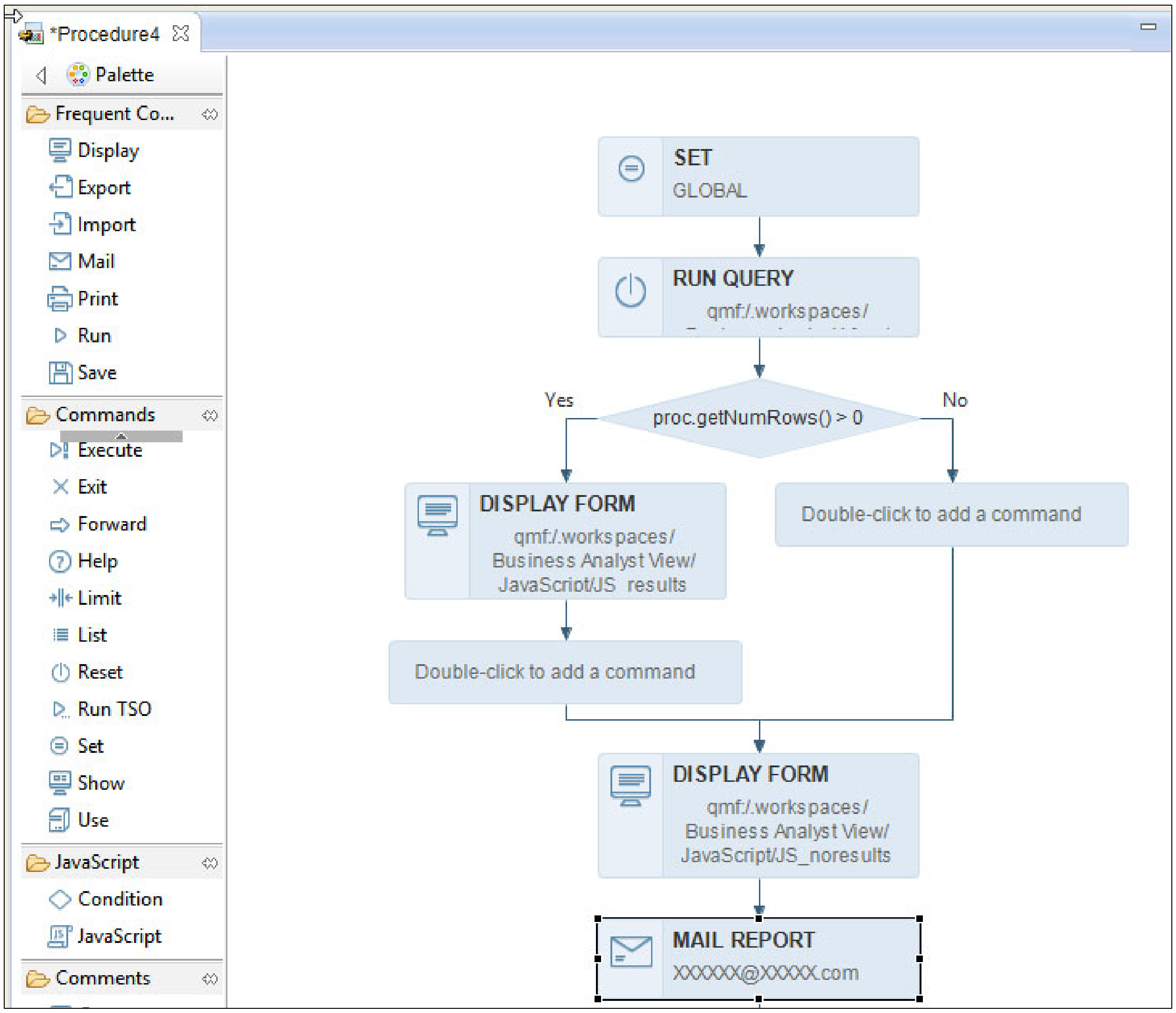


Figure 4-73 Design window for the procedure after the task is moved

25.The DISPLAY FORM and MAIL REPORT for a query with no rows in the result set can be dragged above the empty node for the No result of the JavaScript Condition Node. Figure 4-74 shows the changes.

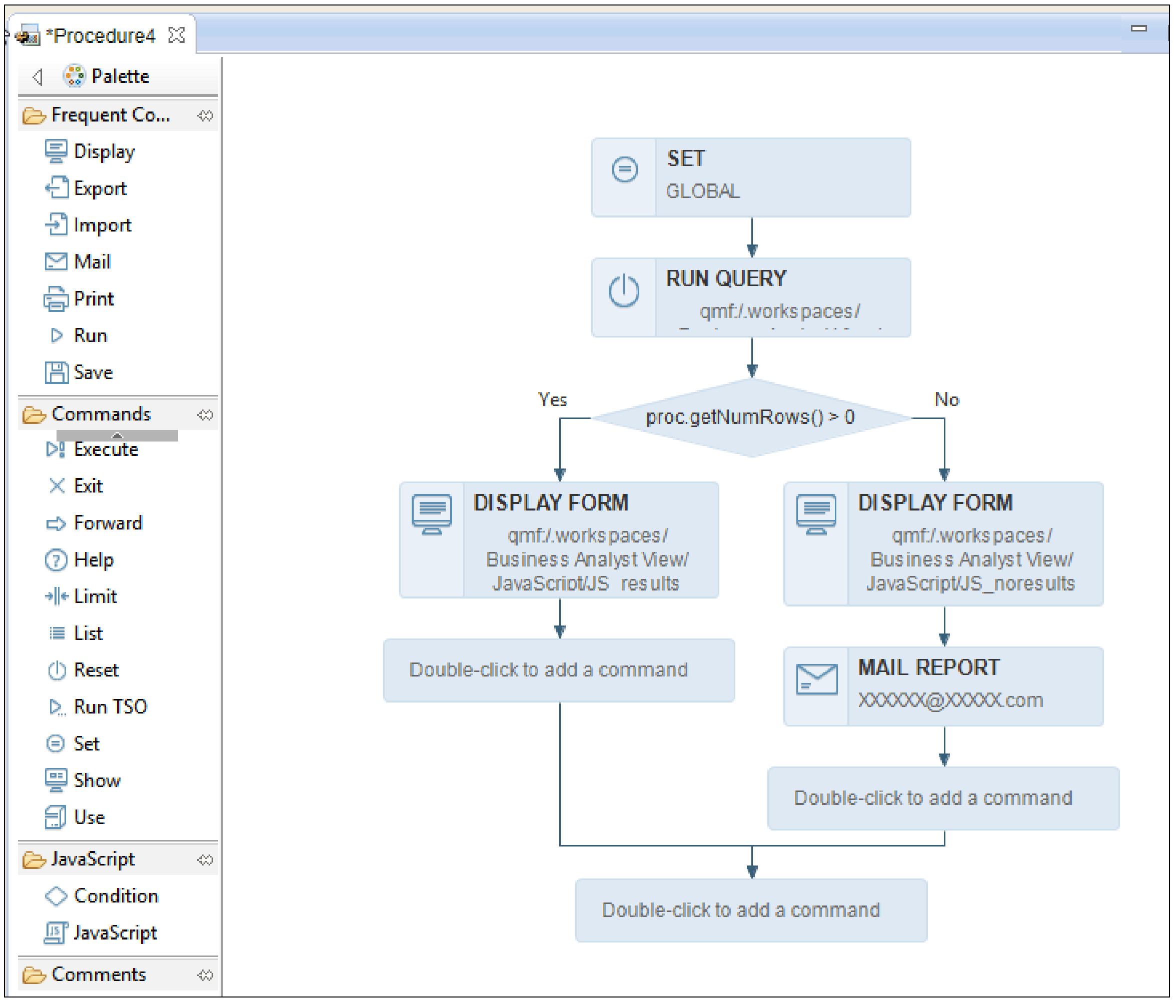


Figure 4-74 Design window for the procedure after the DISPLAY FORM and MAIL REPORT tasks are moved

26. Add a MAIL task for the form that contains values in the report. The easiest way is to copy the mail task for the failure. Right-click the **MAIL REPORT** task and select **Copy** (Figure 4-75).

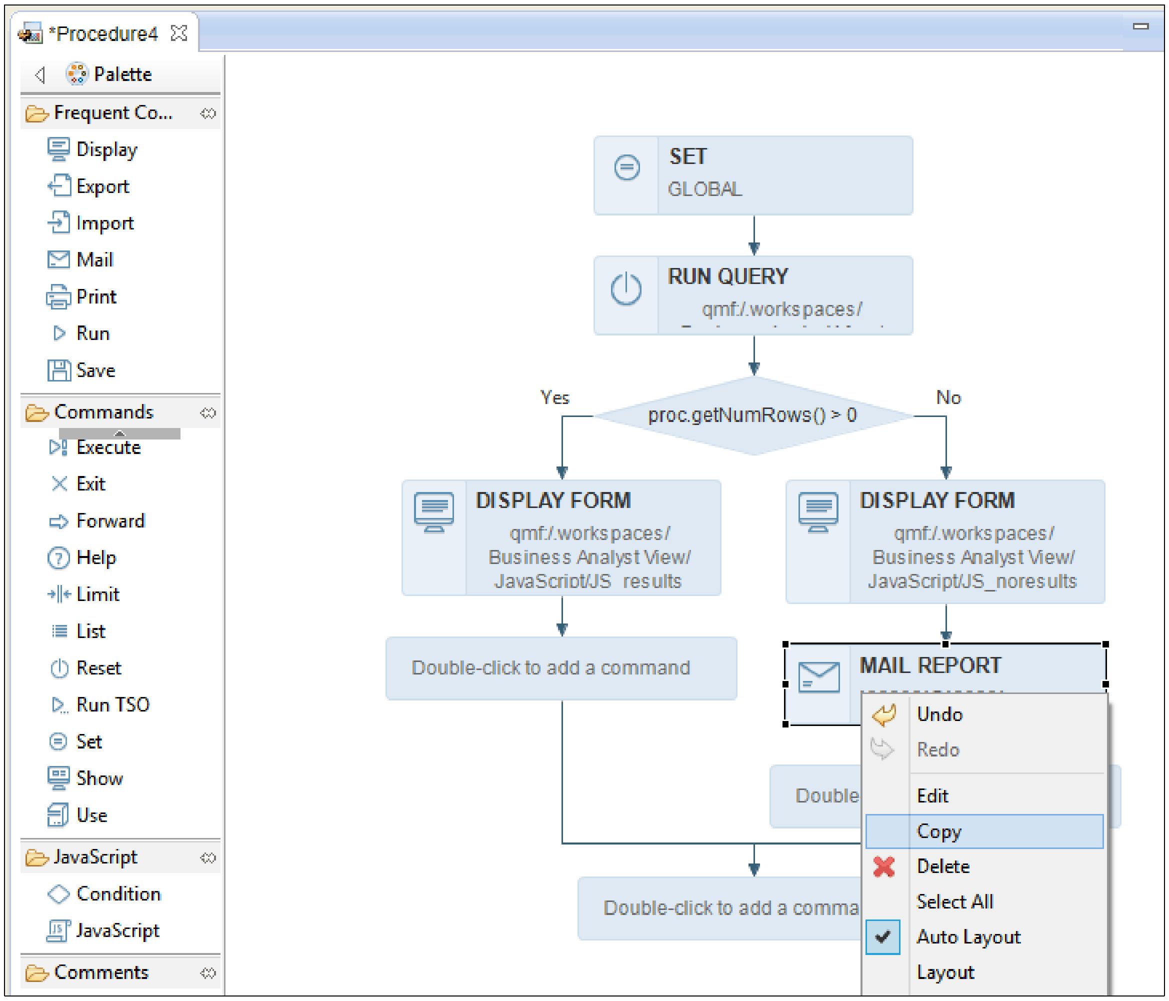


Figure 4-75   Copying the MAIL REPORT task

27. Right-click the empty **Double-click to add a command** node under the DISPLAY FORM under the Yes condition and select **Paste** (Figure 4-76).

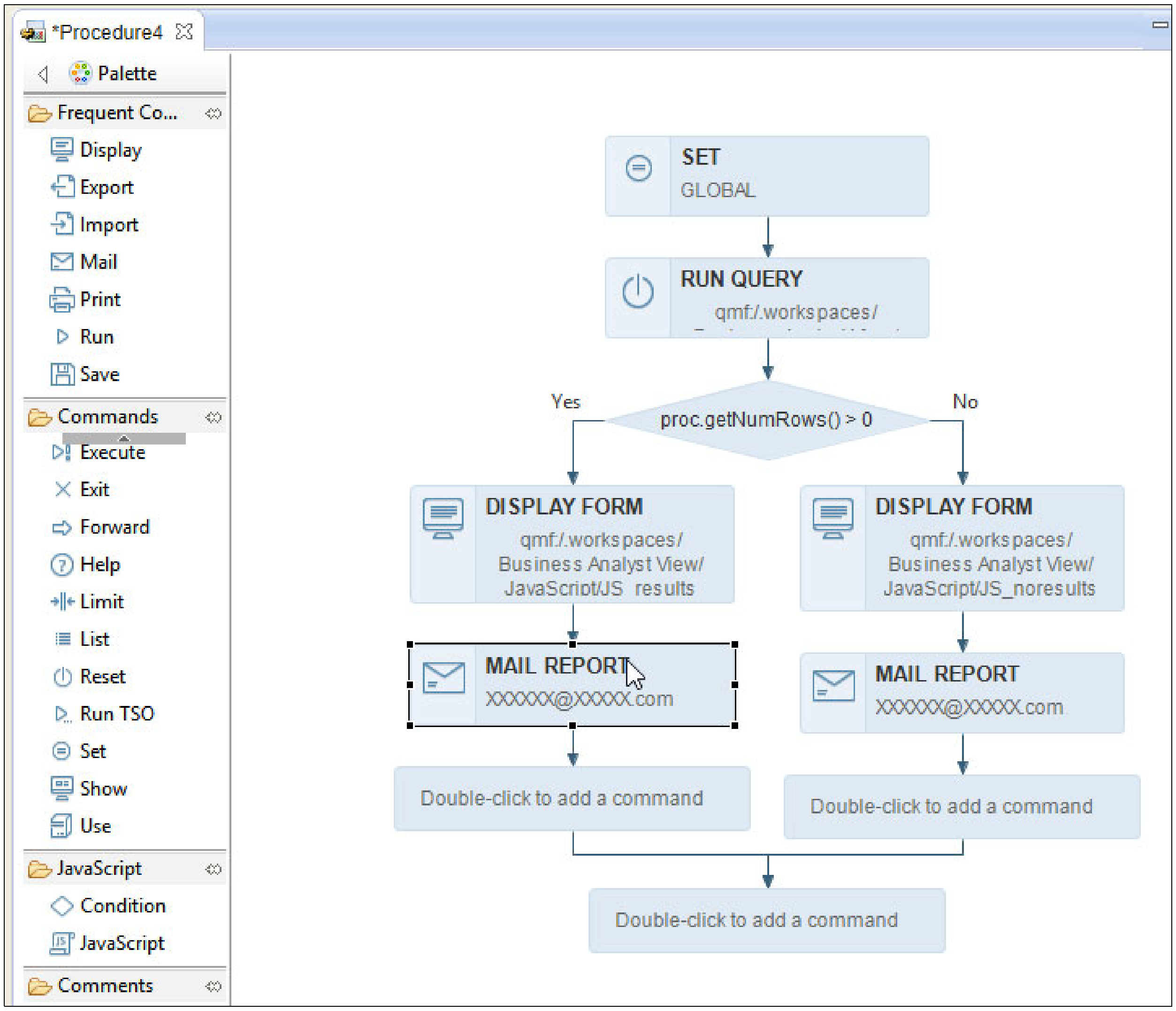
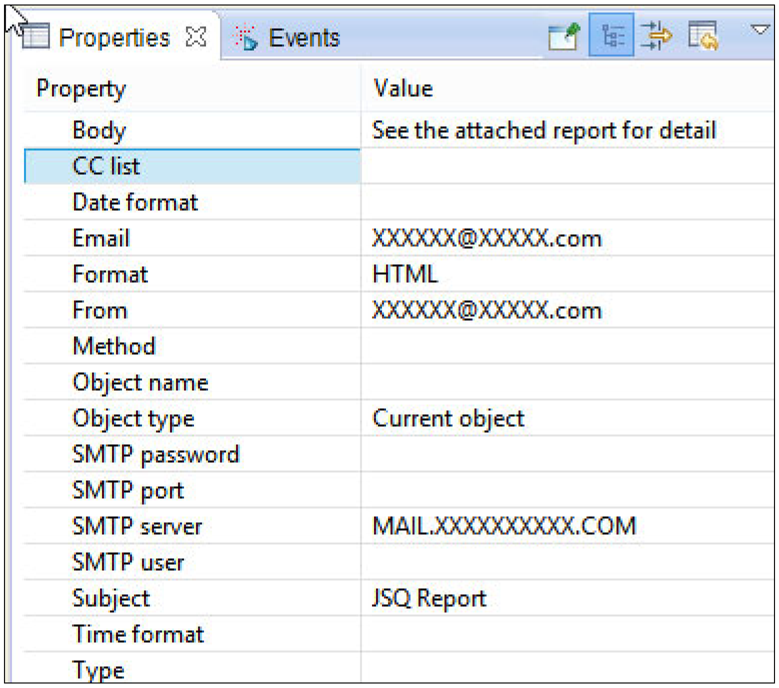


Figure 4-76 Pasting the copied MAIL REPORT task into the procedure

28. Click the new MAIL REPORT node and edit the body and subject values for the node properties, removing the indication that the report failed. See Figure 4-77 for an example.



Property	Value
Body	See the attached report for detail
CC list	
Date format	
Email	XXXXXX@XXXXX.com
Format	HTML
From	XXXXXX@XXXXX.com
Method	
Object name	
Object type	Current object
SMTP password	
SMTP port	
SMTP server	MAIL.XXXXXXXXXX.COM
SMTP user	
Subject	JSQ Report
Time format	
Type	

Figure 4-77 Changing the properties of the MAIL REPORT task

29. Figure 4-78 shows the modified procedure.

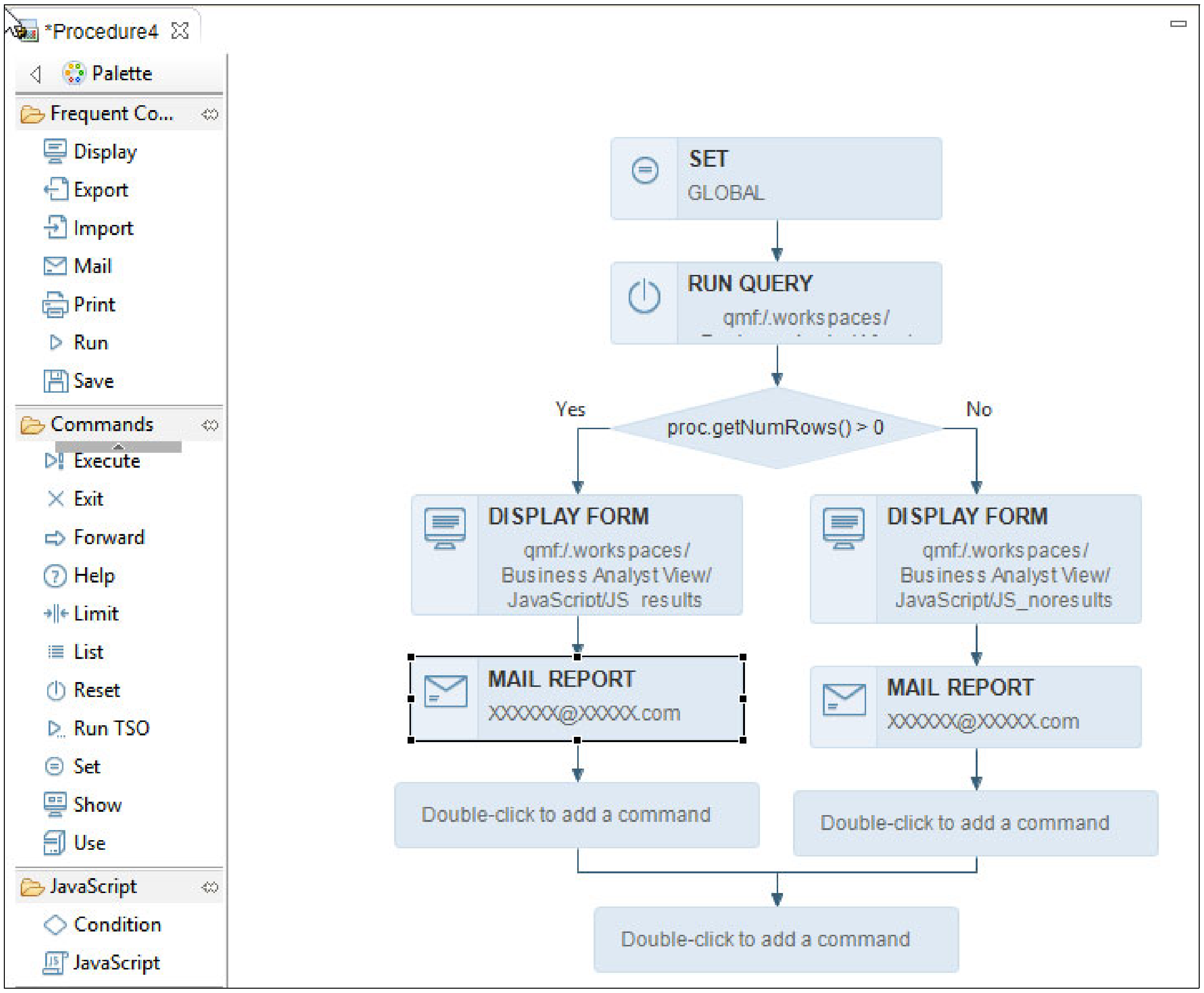
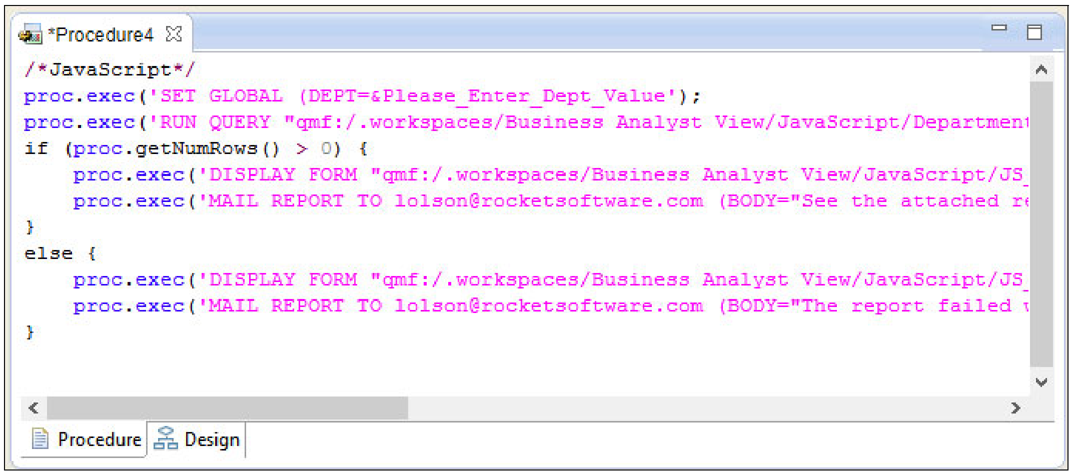


Figure 4-78 Final layout of the sample JavaScript procedure

30. To test the procedure, run the procedure with a value of an existing department number. Then, run the procedure again with an invalid department number. The procedure emails the two forms (one form for a result set with values and the other form for the empty result set). If the tests are successful, save the procedure to the repository.

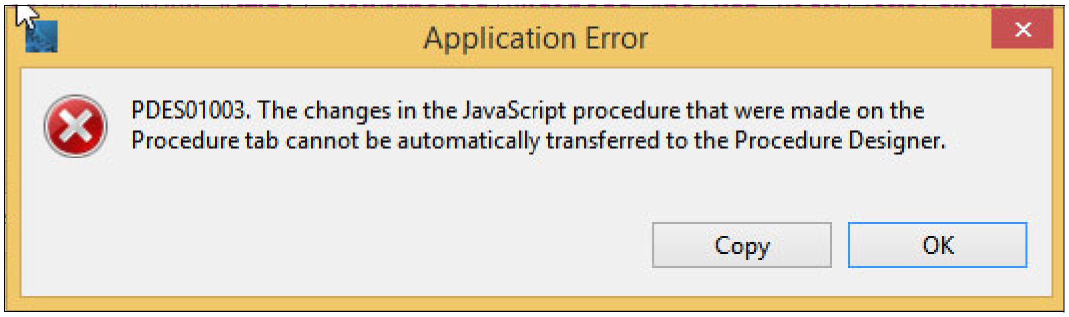
You can edit the procedure text directly in the Procedure tab. Figure 4-79 shows an example procedure.



```
/*JavaScript*/
proc.exec('SET GLOBAL (DEPT=&Please_Enter_Dept_Value');
proc.exec('RUN QUERY "qmf:/.workspaces/Business Analyst View/JavaScript/Departmen
if (proc.getNumRows() > 0) {
    proc.exec('DISPLAY FORM "qmf:/.workspaces/Business Analyst View/JavaScript/JS_
    proc.exec('MAIL REPORT TO lolson@rocketsoftware.com (BODY="See the attached r
}
else {
    proc.exec('DISPLAY FORM "qmf:/.workspaces/Business Analyst View/JavaScript/JS_
    proc.exec('MAIL REPORT TO lolson@rocketsoftware.com (BODY="The report failed
}
```

Figure 4-79 Procedure edit window

However, if a user makes editing changes that cannot be replicated in the Design editor, the user cannot switch back to the editor, even if the procedure is still valid. Figure 4-80 shows a sample error message.



PDES01003. The changes in the JavaScript procedure that were made on the Procedure tab cannot be automatically transferred to the Procedure Designer.

Figure 4-80 Message when the procedure syntax cannot be converted to the Design editor

The **MAIL** command can be difficult to manipulate in JavaScript. The **MAIL** commands contain body text that tends to span multiple lines and that needs to be encased in quotation marks. The subject text needs to be in quotation marks. Both the subject and body might need to include values from variables or query results.

The recommended approach is to specify FORMAT=HTML, which refers to the format of the email. With this format, HTML tags can be included to control the look of the body (Example 4-7).

Example 4-7 FORMAT=HTML

```
proc.exec('MAIL REPORT TO xxxxxx@xxxxx.com '
 +'(BODY="The report failed with this value of DEPT.<br>'
       +' The value needs to be<br>'
       +' in the org table" '
       +'FORMAT=HTML FROM=QMFadmin@xxxxx.com '
       +'SMTPSERVER=MAIL.XXXXXXXXXXXXX.COM '
       +'SUBJECT="Report Failure"');
}
```

To insert values from Global variables or from query results into the email subject or body, you can use the following approach. Define the sections of the **MAIL** command as variables, and then piece them together with the concatenation symbol to avoid issues with quotations within quotations and continuation characters (Example 4-8).

Example 4-8 Global variables and query results

```
/*JavaScript*/
proc.exec('SET GLOBAL (DEPT=&Please_Enter_Dept_Value');
proc.exec('RUN QUERY DB2ADMIN.JSQ_STAFF');
var M ='MAIL REPORT TO xxxxxx@xxxxx.com ';
var B ='(BODY="The report failed with this value of DEPT "';
var F =' FORMAT=HTML FROM=ssullivan@rs.com ';
var S =' SMTPSERVER=MAIL.XXXXXXXXXXXX.COM ';
var SU =' SUBJECT="JSQ Report Failure for Dept = '
+ proc.getVariable('DEPT') + '"' ;

if (proc.getNumRows() >0) {
   proc.exec('DISPLAY FORM DB2ADMIN.JSF_STAFF1');
}
else {
   proc.exec('DISPLAY FORM DB2ADMIN.JSF_STAFF2');
   proc.exec(M + B + F + S + SU);
}
```

5

Caching

IBM DB2 Query Management Facility for z/OS V11.2.1 (QMF) for Workstation and QMF for WebSphere employ caching in several areas. In QMF, *caching* refers to the act of pulling information from a data source and keeping it in memory, a temporary file, or a permanent file at the client machine. Then, when follow-up requests are made for that same information, the locally saved version of the information is used. The typical use cases are listed:

- Reducing CPU consumption at the remote database server
- Providing information to users without direct database access
- Delivering faster performance for the business user

This chapter explores these use cases, the user experience with the different types of caching, the types of information that can be cached, and the refresh mechanisms for the caching. Three types of caching are described:

- Query caching: An administrator enables query caching and sets the caching on a group by group basis, including whether group members can update their cache settings. The results of a specific SQL statement are cached at the machine that is running QMF until the results expire as determined by the administrator. Each user has an independent cache that is not shared with others.
- Dynamarts: Permanent files that contain the SQL and the query result. The dynamarts do not expire, do not require a login to the back-end database after the first run, and can be centrally located and shared.
- Virtual data sources: Logical tables that are defined by SQL, which are similar to a database view. After the first run, the results can be cached at the system that is running QMF, but not as a static file. Instead, they are cached in an SQL database so that they can be creatively queried. These cached tables are centrally located, and the data can be shared by multiple users.

This chapter includes the following topics:

- Query caching
- Dynamarts
- Virtual data sources
- Summary of caching types

Video series: The following videos that describe the QMF caching mechanism are provided to support this chapter:

- Part 1 - Personal Caching:

  `https://youtu.be/1hKt1N-ogsw`
- Part 2 - Administrative Caching:

  `https://youtu.be/8SLXOTOvpss`
- Part 3 - Dynamart:

  `https://youtu.be/bY8mjM8_riM`
- Part 4 - Virtual Data Source:

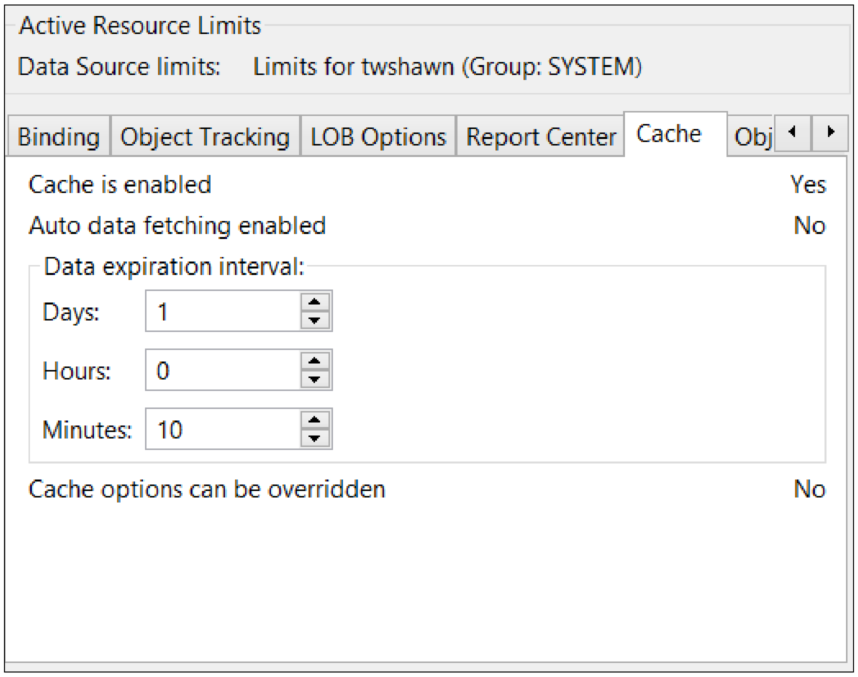
  `https://youtu.be/YceV2Qyyui8`

5.1 Query caching

The query is the fundamental QMF object that pulls business information from a data source. It consists of an SQL SELECT statement and a set of directions that inform QMF how to format and display the results. Queries support CREATE, UPDATE, INSERT, and so on, but those verbs are not relevant to caching.

The use and behavior of queries by any specific user is governed by the QMF Resource Limits. The QMF Resource Limits are a set of rules and limits that are set up by the QMF administrator. These rules control certain aspects of a user's QMF session. Among the types of activities that are controlled are the number of rows that a query might return, the SQL statements that can be used, whether the query usage is tracked, and whether caching is enabled.

A QMF user can look at the current, active resource limits by selecting the **View** → **Resource Limits**. For most limits, only the administrator can set limits. However, for a few limits, the administrator can allow users to set their own limits. Caching is this type of limit. Figure 5-1 shows the caching resource limit from a user's point of view.



Data Source limits: Limits for twshawn (Group: SYSTEM)

Cache is enabled	Yes
Auto data fetching enabled	No

Data expiration interval:

Days:	1
Hours:	0
Minutes:	10

Cache options can be overridden	No

Figure 5-1 Caching resource limit that is shown when you select View → Resource Limits

In this case, the administrator did *not* enable the user to override the cache setting of 1 day.

5.1.1 User experience: QMF for Workstation

After the administrator sets the caching resource limit, the result sets that are pulled from the database are cached on the machine that is running QMF as a file for the time that is specified in the cache setting. With QMF for Workstation, the results are stored as a temporary file on the local desktop computer. For QMF for WebSphere, they are stored on the Web Application Server that is running QMF.

If the same query is run again before the cache expires, the results are read from this file. The term *same query* calls for explanation. When the results are cached, they are *marked* with the following information: the user ID that ran the query, the exact SQL that was in the query, and the database that the query connected to. If any of this information changes, QMF considers the query as a new query and caches the results with the new combination of user ID, SQL, and database.

For example, with a query cache that is set to 1 day, the following SQL is run by the user JAMES, when the user is connected to the database REDBOOKS, at 10:25 AM:

```
SELECT NAME, ID, CURRENT TIMESTAMP AS CST FROM Q.STAFF
```

This SQL returns a particular result set that QMF caches as a file on that machine. The special register CURRENT TIMESTAMP records the time stamp from the moment that DB2 processed the SQL. If no caching is active, this field contains a new value each time that the query is run.

James saves the query to the QMF catalog, the repository, or a file.

In each of the following cases, James gets the same result as shown by the timestamp column:

- One hour later, James opens the query and runs it.
- Two hours later, James closes QMF, performs other work, opens QMF, and runs the query again.
- Three hours later, James shuts down his computer and restarts it, opens QMF, and runs the query again.
- One hour later, James creates any type of report or dashboard by using the query that is connected to the same data source.
- Two hours later, James opens the query, copies the SQL text, pastes it into a new query window, and runs it.
- One hour later, James opens a new query that is connected to the same data source and types the same SQL (not case-sensitive) as the original query.
- One hour later, James opens the query and adds several SQL comments after the end of the SQL and space with the Spacebar to make the SQL more readable.

In each of the following cases, James gets a different result as shown by the timestamp column:

- James selects **Query** → **Set Data Source** and redirects the query to a different data source (different data source = different query).
- James selects **Query** → **Set User Information** and changes his login credentials to a new user ID (different user ID = different query).
- James opens the query and changes the contents in any way that calls for the data source to re-parse it: adding comments in the body of the SQL, rearranging the columns, adding any other SQL at all, deleting any other SQL (including comments in the body) (different SQL = different query).
- James logs on to another machine, connects to the same repository and the same data source, opens the same query from the QMF catalog, and runs it (different machine = different cache).
- James runs the query after 10:25 AM the next day. (The cache expired.)

If the administrator sets the switch to allow users to “override” the cache setting, the user might change the cache expiration time with two important points to remember:

- The user must change it on a query-by-query basis. The user cannot set a global cache limit that applies to all queries.
- The user can override the administrator cache setting only on queries that were saved. If the user opens a new query, types SQL, and runs it, the query is governed by the cache that was set by the administrator. The user must save the query to the QMF catalog or to the repository workspace. Then, the cache can be set as shown next. The user cannot set the cache on queries that are saved as files.

The cached files are compressed and unreadable. The queries are unusable outside of QMF. They are stored in the user profile directory on the machine that runs QMF for Workstation. The queries must not be opened directly by any application other than QMF or they risk corruption.

QMF deletes the cache files 15 minutes after they expire. This cleanout is automatic and does not depend on someone that is running a query that calls for the cached data.

5.1.2 User experience: QMF for WebSphere

Caching works the same in QMF for WebSphere as it does in QMF for Workstation. The only differences are that *all* users share the directory for caching the data.

Certain clients might need to run QMF for WebSphere in a cluster. In a cluster, WebSphere Application Server might be installed on multiple machines and QMF for WebSphere is deployed in each instance of WebSphere Application Server. A load balancing application is added as a layer on top of this setup. Users connect to a single URL that is mapped to the load balancer. Then, the load balancer redirects the user session to one of the QMF for WebSphere instances based on rules about how busy the QMF for WebSphere instances are.

Caching might not work correctly. The cached data is stored in the QMF for WebSphere Application Data directory. Each machine with QMF for WebSphere has its own Application Data directory and its own cache. If a user runs a query in the morning when the load balancer directed the session to machine A, the cache is on machine A. If the user signs out and comes into a new session later in the day, the load balancer might direct the user session to machine B, and the cache from earlier is not accessible.

To address this situation, QMF supplies a configuration option that causes all of the instances of QMF for WebSphere in the cluster to use the same application data store on a shared drive. This option is a Java virtual machine (JVM) property that is called qmf.instance.area. It is straightforward to implement and it is documented in *Installing and Managing DB2 QMF for Workstation and DB2 QMF for WebSphere*, GC27-6742-02, which is available from the IBM Knowledge Center. Implementing the qmf.instance.area option causes a cluster to use a single cache and gives users a consistent experience regardless of the load balancer.

Note: All QMF objects that use a query automatically inherit the use of the cache from the query. These objects are listed:

- Queries
- Visual queries
- Analytical queries
- Ad hoc queries
- Forms
- Procedures
- Visual reports
- Quick reports
- Drill-down paths
- Dashboards
- Forecasts
- Prompt hierarchies
- Scheduled tasks
- QMF Vision

5.1.3 Cache use case 1

Before you try to use the cache in production, experiment with it a bit. The following queries are useful to test the cache feature:

- `SELECT CURRENT TIME AS TIME FROM SYSIBM.SYSDUMMY1`
- `SELECT CURRENT TIMESTAMP AS TIMEST FROM SYSIBM.SYSDUMMY1`

These queries return result sets that are a single cell with the current time or time stamp on record at the DB2 database. CURRENT TIME and CURRENT TIMESTAMP are DB2 Special Registers, which are similar to a bonus column that you can add to any query. The table SYSIBM.SYSDUMMY1 is a table that DB2 supplies for general-purpose, dummy queries for test and validation. A dummy query for testing the cache is referred to as a “*cache test dummy*”.

5.1.4 Cache use case 2

If your cache is on a time scale that is larger than one day, you might want to use the CURRENT DATE Special Register:

```
SELECT ID, NAME, SALARY, CURRENT DATE AS DATE FROM Q.STAFF
```

Note: No feature exists to refresh a query before the cache expires. If you need to refresh a query before the cache expires, you must open the SQL and add a carriage return to force a refresh on the next run.

5.1.5 Cache use case 3

QMF includes a feature that is called the *Query Profiler*. It reports the expired time for different parts of the query run. It is activated by clicking the Query Profiler icon in Figure 5-2.

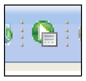
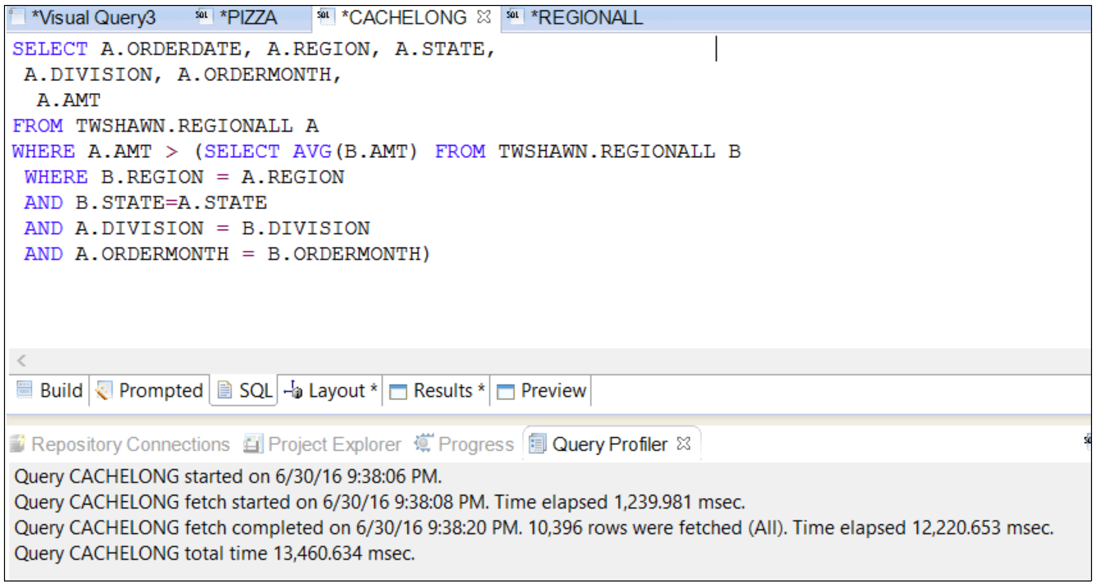


Figure 5-2 Query Profiler icon

Figure 5-3 shows SQL with a *correlated subquery*. On the first run, the Query Profiler says that it takes over 13 seconds.

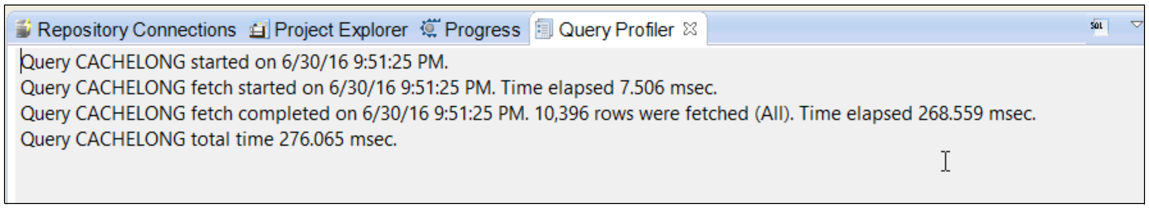


```
SELECT A.ORDERDATE, A.REGION, A.STATE,
 A.DIVISION, A.ORDERMONTH,
  A.AMT
FROM TWSHAWN.REGIONALL A
WHERE A.AMT > (SELECT AVG(B.AMT) FROM TWSHAWN.REGIONALL B
 WHERE B.REGION = A.REGION
 AND B.STATE=A.STATE
 AND A.DIVISION = B.DIVISION
 AND A.ORDERMONTH = B.ORDERMONTH)
```

```
Query CACHELONG started on 6/30/16 9:38:06 PM.
Query CACHELONG fetch started on 6/30/16 9:38:08 PM. Time elapsed 1,239.981 msec.
Query CACHELONG fetch completed on 6/30/16 9:38:20 PM. 10,396 rows were fetched (All). Time elapsed 12,220.653 msec.
Query CACHELONG total time 13,460.634 msec.
```

Figure 5-3 Long-running correlated subquery

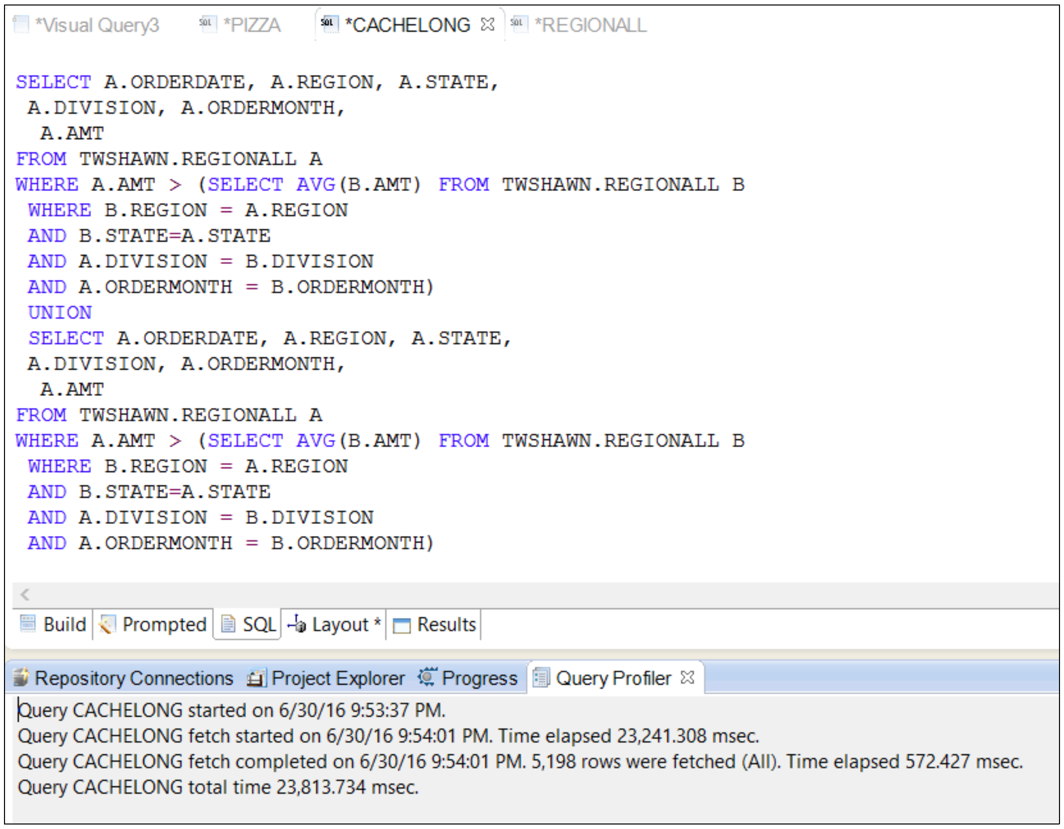
On subsequent runs, before the cache expires, it takes about a quarter of a second, as shown in Figure 5-4.



```
Query CACHELONG started on 6/30/16 9:51:25 PM.
Query CACHELONG fetch started on 6/30/16 9:51:25 PM. Time elapsed 7.506 msec.
Query CACHELONG fetch completed on 6/30/16 9:51:25 PM. 10,396 rows were fetched (All). Time elapsed 268.559 msec.
Query CACHELONG total time 276.065 msec.
```

Figure 5-4 Query Profiler for cached query

By using UNION to “union” the query to itself, the same number of records are delivered, but it now takes 23 seconds, as shown in Figure 5-5. It still returns the same number of rows, but it takes almost twice as long.

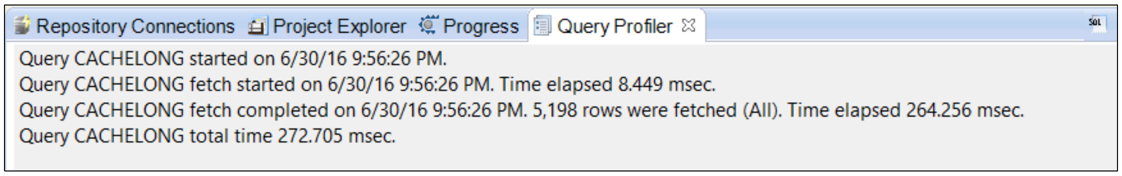


```
SELECT A.ORDERDATE, A.REGION, A.STATE,
 A.DIVISION, A.ORDERMONTH,
  A.AMT
FROM TWSHAWN.REGIONALL A
WHERE A.AMT > (SELECT AVG(B.AMT) FROM TWSHAWN.REGIONALL B
 WHERE B.REGION = A.REGION
 AND B.STATE=A.STATE
 AND A.DIVISION = B.DIVISION
 AND A.ORDERMONTH = B.ORDERMONTH)
 UNION
 SELECT A.ORDERDATE, A.REGION, A.STATE,
 A.DIVISION, A.ORDERMONTH,
  A.AMT
FROM TWSHAWN.REGIONALL A
WHERE A.AMT > (SELECT AVG(B.AMT) FROM TWSHAWN.REGIONALL B
 WHERE B.REGION = A.REGION
 AND B.STATE=A.STATE
 AND A.DIVISION = B.DIVISION
 AND A.ORDERMONTH = B.ORDERMONTH)
```

```
Query CACHELONG started on 6/30/16 9:53:37 PM.
Query CACHELONG fetch started on 6/30/16 9:54:01 PM. Time elapsed 23,241.308 msec.
Query CACHELONG fetch completed on 6/30/16 9:54:01 PM. 5,198 rows were fetched (All). Time elapsed 572.427 msec.
Query CACHELONG total time 23,813.734 msec.
```

Figure 5-5 Query that runs even longer

But, on the next run, the query is a fraction of a second because it is the same number of rows from the cache, as shown in Figure 5-6.



```
Query CACHELONG started on 6/30/16 9:56:26 PM.
Query CACHELONG fetch started on 6/30/16 9:56:26 PM. Time elapsed 8.449 msec.
Query CACHELONG fetch completed on 6/30/16 9:56:26 PM. 5,198 rows were fetched (All). Time elapsed 264.256 msec.
Query CACHELONG total time 272.705 msec.
```

Figure 5-6 Query Profiler for longer-running query from cache

DB2 is spared the repeated calculation, and the user gets the results in a fraction of the time.

5.1.6 Cache use case 4

We describe the procedure to use if the QMF administrator decides that users can override the cache. The user can change the cache on queries that are saved to the catalog or repository only. The user navigates to the query and right-clicks to select **Properties**, as shown in Figure 5-7.

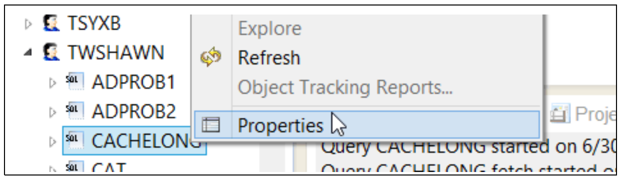
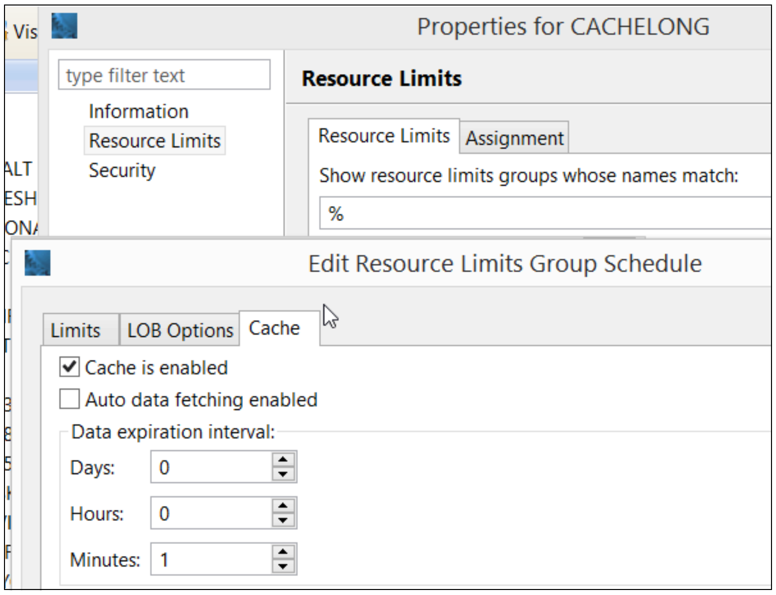

TSYXB
TWSHAWN
ADPROB1
ADPROB2
CACHELONG
CAT
Explore
Refresh
Object Tracking Reports...
Properties
Proje
Query CACHELONG started on 6/30
Query CACHELONG fetch started o

Figure 5-7 Selecting Properties to set the cache

A small set of user-controlled, query-specific resource limits is shown in Figure 5-8.



Show resource limits groups whose names match: %

☑ Cache is enabled
☐ Auto data fetching enabled

Data expiration interval:

Days: 0

Hours: 0

Minutes: 1

Figure 5-8 Setting the query cache limits

The user can change the cache limits and override the administrator's settings.

5.2 Dynamarts

Dynamarts are a QMF object that acts as a permanent cache as a file or as a QMF repository object. When a traditional QMF query is saved, the SQL is saved and the formatting instructions for the results, including graphs and charts, are saved. With a dynamart, the SQL, formatting, and retrieved data are all saved together. This approach differs from the cache that is described in the previous section in the following ways:

- The dynamart is a permanent file that does not expire.
- Dynamarts can be shared. One user can pull the data, and other users can use the results without connecting to the back-end database.
- Dynamarts can be shared by storing them centrally in the QMF repository, by saving them on a shared drive, or by transferring the file through email, File Transfer Protocol (FTP), or another transfer mechanism.

5.2.1 User experience: QMF for Workstation dynamarts

You can create a dynamart only from a visual query. Any QMF catalog query that you want to change into a dynamart must be converted to a visual query. Follow these steps:

1. Open the query and select **Query** → **Convert to Visual Query**. In Figure 5-9, the CACHELONG query from the previous example is converted.

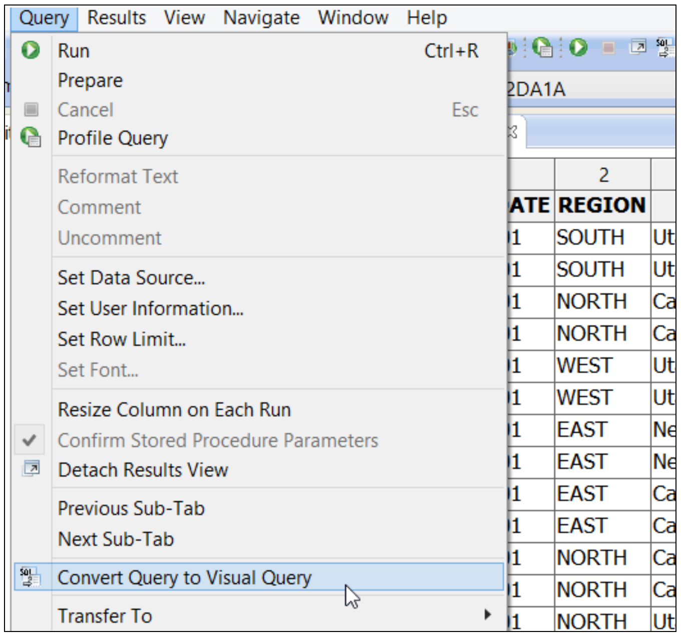
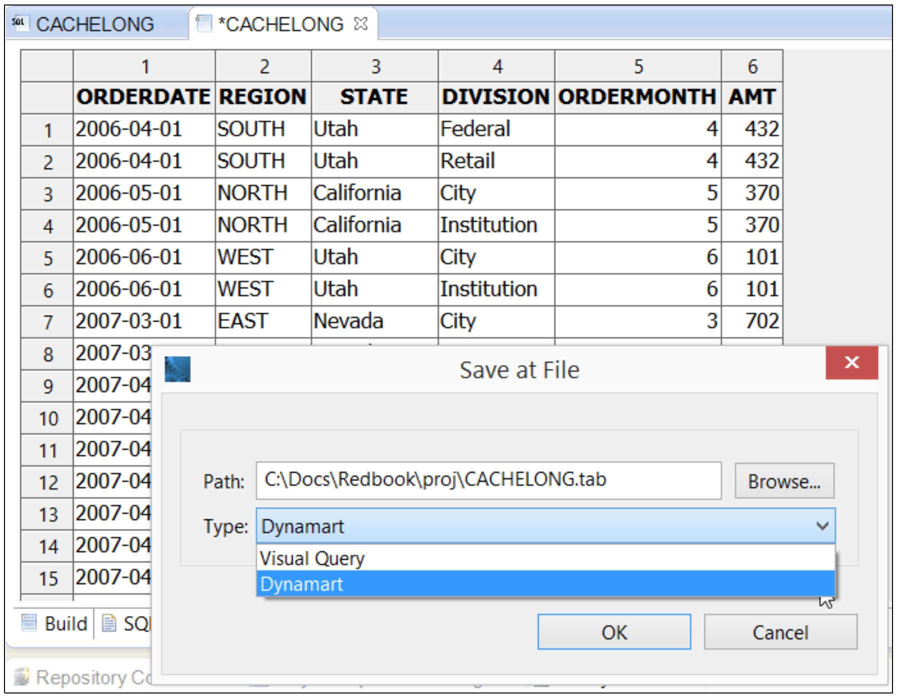


Figure 5-9 Converting a traditional QMF query to a visual query

2. To create the dynamart, run the query, select **File** → **Save at**, and choose **File** or **Repository**. On the Save at File window, navigate to the location that you want to save it to, give it a name, and most importantly, set the Type drop-down list box to **Dynamart**. Click **OK**, as shown in Figure 5-10.



	1	2	3	4	5	6
	ORDERDATE	REGION	STATE	DIVISION	ORDERMONTH	AMT
1	2006-04-01	SOUTH	Utah	Federal	4	432
2	2006-04-01	SOUTH	Utah	Retail	4	432
3	2006-05-01	NORTH	California	City	5	370
4	2006-05-01	NORTH	California	Institution	5	370
5	2006-06-01	WEST	Utah	City	6	101
6	2006-06-01	WEST	Utah	Institution	6	101
7	2007-03-01	EAST	Nevada	City	3	702
8	2007-03					
9	2007-04					
10	2007-04					
11	2007-04					
12	2007-04					
13	2007-04					
14	2007-04					
15	2007-04					

Path: C:\Docs\Redbook\proj\CACHELONG.tab
Type: Dynamart

Figure 5-10 Saving as a dynamart

The saved dynamart can be now used as the source of data for any QMF object that a regular query can be used for:

- Queries
- Procedures
- Forms
- Visual reports
- Dashboards
- Ad hoc reports
- Forecasts

The following characteristics apply when a dynamart is saved to a file:

- A dynamart has an extension of `tab`. A dynamart can be read by QMF for Workstation only. Do not try to use a word processor. The dynamart is saved in a compressed format, and a word processor might corrupt it.
- A dynamart can be transferred to other machines and used by QMF for Workstation or QMF for WebSphere.
- File-based dynamarts are difficult to implement in a QMF for WebSphere environment. Save them to the QMF for WebSphere repository after you open them.

The following characteristics apply when a dynamart is saved to a repository:

- The dynamart is saved in the repository storage database as a single record (a compressed, binary large object (BLOB)) in a repository table. If it is larger than 1 MB, it is broken up and serialized over several records in this table.
- The dynamart is centrally located and does not need to be transferred to other machines for other users to share it.

Repository-based dynamarts work well in QMF for WebSphere applications. In both cases, the dynamart can be opened and the data can be viewed and used without logging in to or otherwise connecting to the back-end data source from which the data was initially pulled.

Unlike query caching, the dynamart is not automatically refreshed by QMF but the user can initiate the refresh in one of two ways:

- When a dynamart is opened, it presents the data as though the query was run. You see the result sets, view the SQL or look at it in build mode, and create reports and visualizations. All of these capabilities use the cached data.

  But, if you click **Run** or otherwise run the dynamart, you are required to log in. The SQL is sent to the database, and the results are refreshed.

  If you close the dynamart, you are prompted to save the changes. If you do not save the changes, the old, non-refreshed, cached data stays in the dynamart. If you save the changes, the data in the dynamart is replaced by the refreshed data.
- If the dynamart is run from a procedure, a **REFRESH** parameter is used:

  ```
  RUN DYNAMART REGIONDYN (REFRESH = YES
  ```

  When this procedure is run, the SQL is run at the database, and the dynamart is refreshed and saved with the new data. All QMF objects that use the dynamart in a linking mode rather than an embedded mode reflect the changes in the refreshed data.

Note: Do not use the Query Profiler on a dynamart because it refreshes the dynamart and defeats the purpose of the caching.

5.2.2 User experience: QMF for WebSphere dynamarts

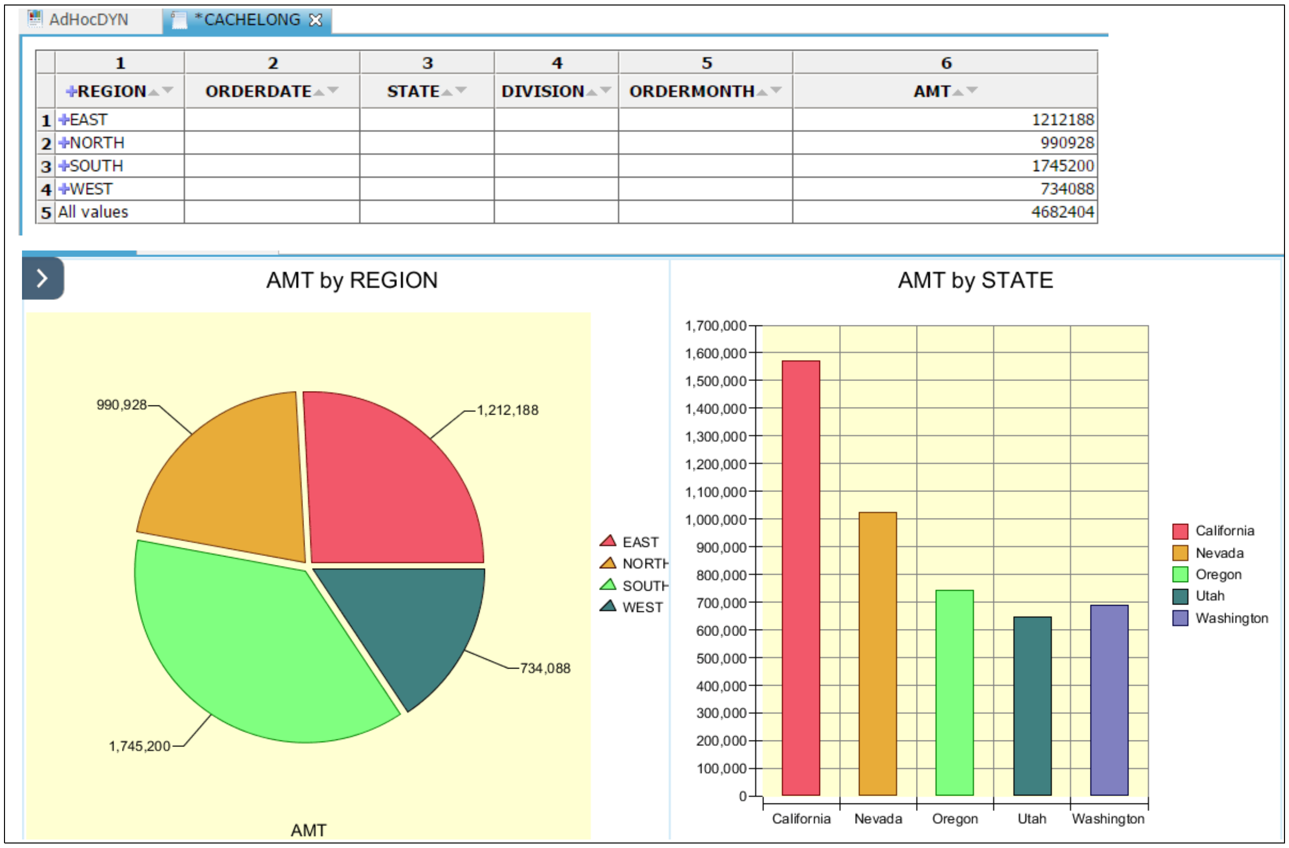
Dynamarts work in QMF for WebSphere in the same way that they work in QMF for Workstation. The exception is that dynamarts that are saved as files are difficult to implement in QMF for WebSphere. Most of the QMF for WebSphere objects (drill-down paths, visual reports, and forecasts) cannot link to a file-based dynamart.

If the QMF for WebSphere object uses an embedded dynamart, no way exists to refresh the embedded dynamart in these objects without rebuilding the object. So, for QMF for WebSphere deployments, save dynamarts to the repository workspace.

5.2.3 Dynamart use case 1

Two popular uses of queries that go beyond dumping the data to a spreadsheet are roll-up grids and ad hoc reports. Because the dynamart does not require a login to the database that holds the source data, a dynamart is useful for people who need to see and work with the data but who do not have access to the back-end database.

Figure 5-11 shows a dynamart that is manipulated into a roll-up grid and run from QMF for WebSphere. The effect is more pronounced when the dynamart is viewed live or in a video than in a screen capture. The process is that the user logs in to the repository to get the lists of reports and queries. When the roll-up query and ad hoc report are run, no prompt appears for the back-end data source and the performance is good.



	1 REGION	2 ORDERDATE	3 STATE	4 DIVISION	5 ORDERMONTH	6 AMT
1	EAST					1212188
2	NORTH					990928
3	SOUTH					1745200
4	WEST					734088
5	All values					4682404

Figure 5-11 A dynamart in use as a grid and a visualization

Both of these displays can be deployed as stand-alone web pages or embedded into other web applications. However, if the user cannot connect to the back-end data source, how do these displays ever get updated when the data changes in the back-end database? The next example addresses this situation.

5.2.4 Dynamart use case 2

Consider the following procedure:

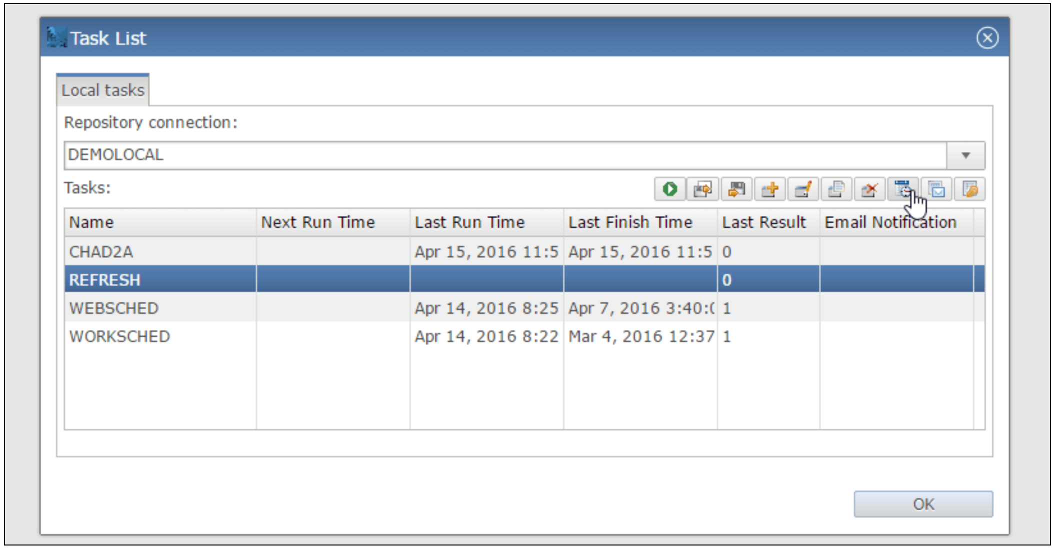
```
DISPLAY DYNAMART CACHELONG
```

This procedure opens the result set. However, the following procedure opens it and refreshes it, which updates the saved DYNAMART with the refreshed data. This example is good for using the CURRENT TIME or CURRENT TIMESTAMP to test and understand how it works.

```
DISPLAY DYNAMART CACHELONG (REFRESH = YES
```

If this procedure is saved as DYNAREFRESH, it can be put into the QMF for WebSphere Scheduler by the person who originally wrote the query.

Figure 5-12 shows the task list output when you select **View** → **Scheduled Tasks**.



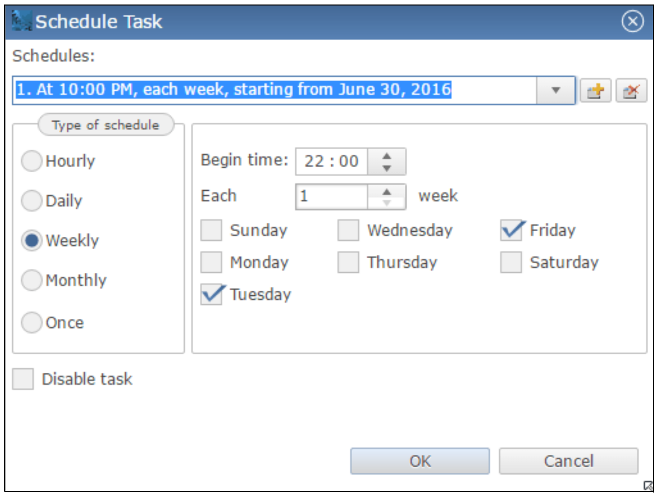
Repository connection: DEMOLOCAL

Tasks:

Name	Next Run Time	Last Run Time	Last Finish Time	Last Result	Email Notification
CHAD2A		Apr 15, 2016 11:5	Apr 15, 2016 11:5	0	
REFRESH				0	
WEBSCHED		Apr 14, 2016 8:25	Apr 7, 2016 3:40:(	1	
WORKSCHED		Apr 14, 2016 8:22	Mar 4, 2016 12:37	1	

Figure 5-12 Scheduled tasks

You can run REFRESH to refresh multiple schedules, as shown in Figure 5-13.



Schedules: 1. At 10:00 PM, each week, starting from June 30, 2016

Type of schedule: Hourly, Daily, [✓ Weekly], Monthly, Once

Begin time: 22 : 00

Each 1 week

Sunday, Monday, [✓ Tuesday], Wednesday, Thursday, [✓ Friday], Saturday

Disable task

Figure 5-13 Possible schedules for the REFRESH

After the Scheduler runs the procedure to refresh the dynamart, any report that links to that dynamart uses the up-to-date data. The user does not need to connect to the database.

5.3 Virtual data sources

Virtual data sources are a QMF construct so that a user can take queries and tables from any combination of QMF data sources and create local shortcuts or links to them. These shortcuts or links are displayed for the user as a set of tables all in the same relational database. The QMF term for one of these shortcuts or links is a *virtual table*.

You can optionally set up virtual tables so that the first time that they are accessed, the data is cached in a local QMF internal database. This method differs from the other types of caching that were mentioned previously. The cached data can be queried with various SQL because the cached data is cached as records in a relational database. For dynamarts and query caching, you get the exact result set only at the time of the cache. Only a QMF administrator can create a virtual data source or add virtual tables to a virtual data source.

5.3.1 Setting up a virtual data source

During the installation of QMF, the option for virtual data sources is selected, by default. If it was deselected during the installation process, this feature is not available until QMF for Workstation is reinstalled with the option for virtual data sources selected.

You must have the QMF administrator perspective set up and you must have QMF administrative authority to create virtual data sources. Any users can use a virtual data source after an administrator sets it up.

You must create a virtual data source by right-clicking the **Virtual Data Sources** node in the Repository Explorer or Repositories tree and selecting **New** → **Virtual Data Source**. This process is described in detail in the next section.

You can create virtual tables from queries or from tables. The preferred practice is to create the virtual tables from queries when possible because of the architecture of the virtual data source.

When the virtual table is created, QMF stores an SQL SELECT statement. If you use a table as the source, QMF stores an SQL statement that selects all of the columns and rows from the table. The business user must add the sophisticated SQL when the user queries the virtual table. But, if you design the virtual table with a sophisticated query with Group By, Case statements, or Date functions, the business user adds simple WHERE clauses only.

Also, because the virtual table can point to databases other than DB2, your business users might not know the syntax for Oracle or Microsoft (MS) SQL Server functions. So, it is best to embed the database-specific functions in the definition of the virtual tables.

The caching mechanism for virtual data sources is called the *expiration schedule*. When it is employed, the fetched data is returned to the client machine that is running QMF and cached in a local QMF database. For QMF for Workstation, each user sees the same centrally located virtual table definition, but when the table is queried, each user gets a copy of the results that are cached on the local workstation. Subsequent calls to the virtual table are handled by the local QMF database with the local copy of the data.

5.3.2 User experience: Virtual Data sources

Virtual data sources work in the same way for both QMF for Workstation and QMF for WebSphere. An administrator creates a virtual database in the following manner:

1. Call for a new virtual data source by right-clicking **Virtual Data Sources,** as shown in Figure 5-14. Click **New** → **Virtual Data Source**.

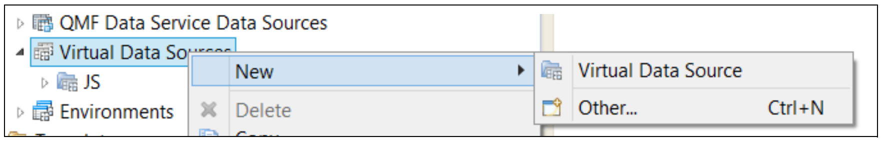
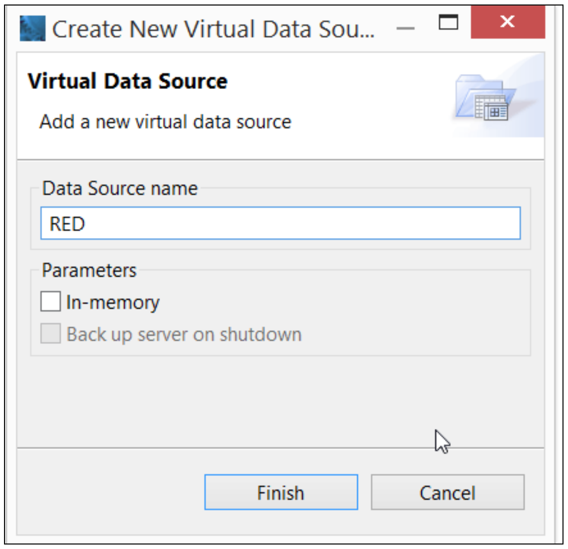


Figure 5-14 Creating a virtual data source

2. Enter a data source a name, as shown in Figure 5-15. Do not select In-memory.

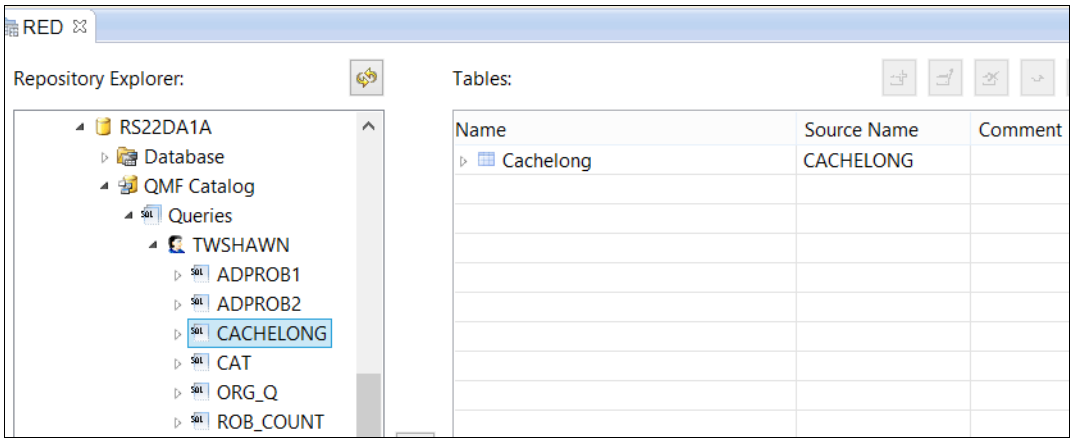


Data Source name: RED

Parameters
- [ ] In-memory
- [ ] Back up server on shutdown

Figure 5-15 Enter a name for the virtual data source

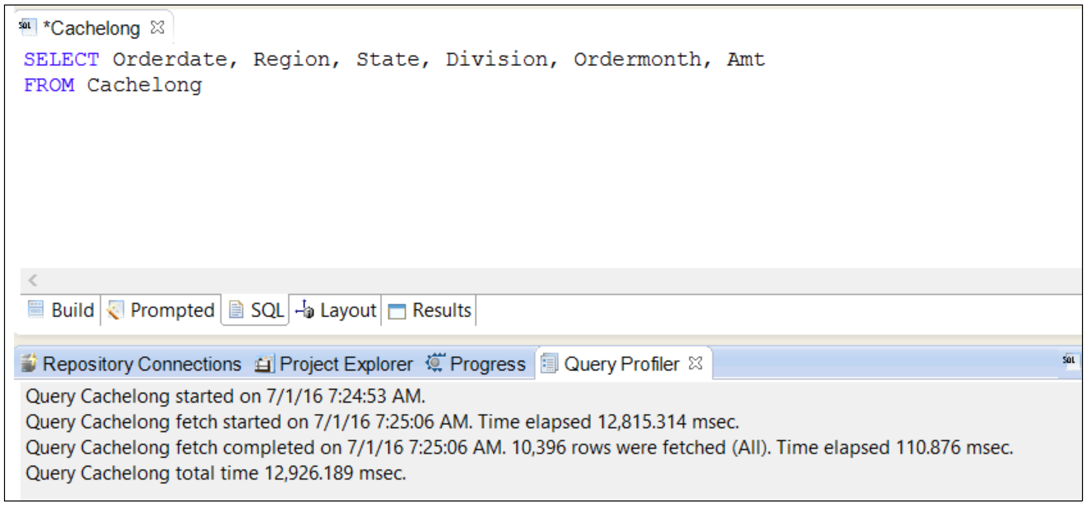
3. The administrator populates it with a virtual table that is made from a query, as shown in Figure 5-16.



Name	Source Name	Comment
Cachelong	CACHELONG	

Figure 5-16 Populating the virtual database

Users can query the virtual table as though it were an actual table. The query is profiled at 13 seconds in Figure 5-17.

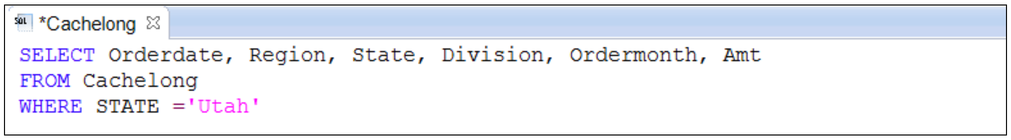


```
SELECT Orderdate, Region, State, Division, Ordermonth, Amt
FROM Cachelong
```

```
Query Cachelong started on 7/1/16 7:24:53 AM.
Query Cachelong fetch started on 7/1/16 7:25:06 AM. Time elapsed 12,815.314 msec.
Query Cachelong fetch completed on 7/1/16 7:25:06 AM. 10,396 rows were fetched (All). Time elapsed 110.876 msec.
Query Cachelong total time 12,926.189 msec.
```

Figure 5-17 Query Profiler for a virtual table

Figure 5-18 shows a virtual table that is queried with more SQL than is in its definition. This capability is similar to a view in DB2.



```
SELECT Orderdate, Region, State, Division, Ordermonth, Amt
FROM Cachelong
WHERE STATE ='Utah'
```

Figure 5-18 Querying the virtual table

The workflow of this query and virtual table is described. When the administrator creates the virtual table, QMF takes the SQL that defines the virtual table and saves it to a table in the QMF repository. The virtual table definition is centrally located and available to all users.

When the user addresses the virtual table with SQL, the virtual table definition is retrieved by the local QMF in use. Then, QMF combines the underlying SQL definition of the virtual table and the new SQL in the query to get the final query that is sent to DB2 (or whichever database is at the back end).

Therefore, it is better to put all of the back-end-specific SQL into the definition. Different databases use different SQL functions, such as the online analytical processing (OLAP) functions in DB2, and different implementations of certain common functions, such as finding the difference between dates. The SQL that defines the virtual table must have all of the complexity so that the user needs to add only simple SQL, such as WHERE clauses.

So far, no caching is used. The caching is involved with the expiration schedules branch, as shown in Figure 5-19.

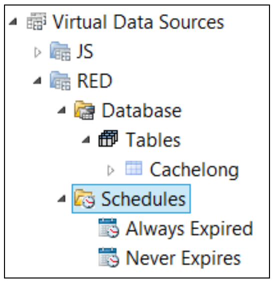


Figure 5-19 Setting up schedules

Two schedules are included, by default. The default setting is Always Expired. Therefore, every time that the virtual table is used, QMF goes back to the source data to get the results.

If you set the schedule to Never Expires, the first time that the virtual table is used, the data is downloaded to the local QMF and stored in a relational database that is embedded in QMF. From that point, the SQL is handled by that internal local QMF database. Consider this workflow:

- An administrator creates a virtual table and sets it to Never Expires.
- By using QMF for Workstation, James queries that virtual table on Monday and gets the data at that time on Monday that is cached on his local machine.
- By using QMF for Workstation, Lydia queries that virtual table on Friday and gets the data at that time on Friday that is cached on her local machine.
- If the data changes at the source database between Monday and Friday, James and Lydia work with different data from then on.

For refreshing to synchronize the caches of all of the users, the administrator can set up custom schedules:

1. As shown in Figure 5-20, right-click **Schedules** and select **New** → **Virtual Data Source Schedules**.

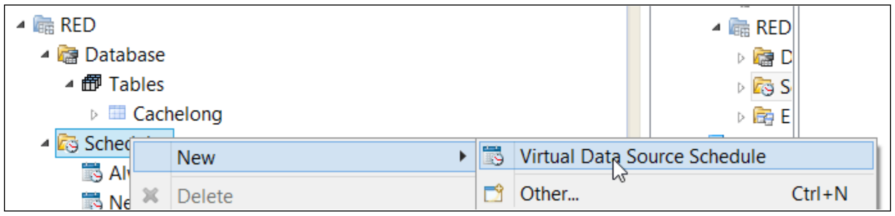
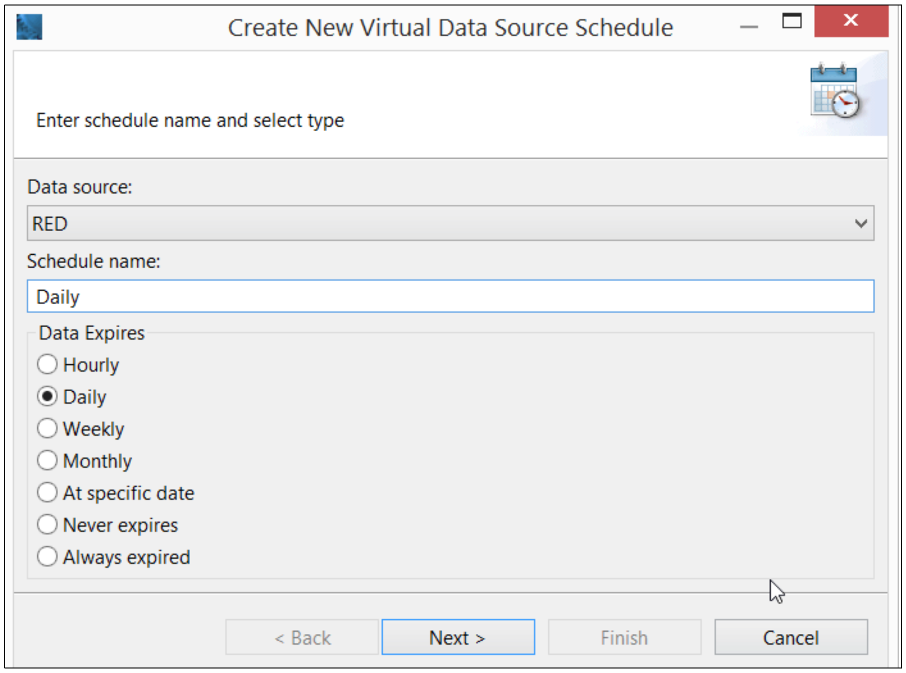


Figure 5-20 Setting up a new schedule

2. Enter a name for the schedule. Select the frequency for the data to expire, as shown in Figure 5-21.



Create New Virtual Data Source Schedule

Enter schedule name and select type

Data source: RED

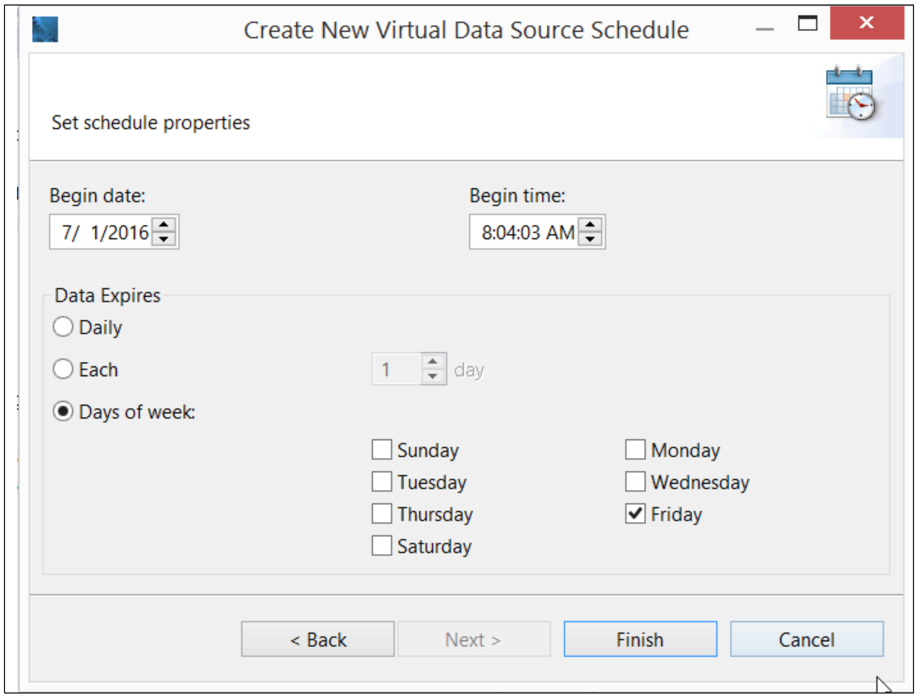
Schedule name: Daily

Data Expires

Hourly [✓ Daily] Weekly Monthly At specific date Never expires Always expired

Figure 5-21 Name the schedule

3. Enter the details of the schedule, as shown in Figure 5-22, and click **Finish**.

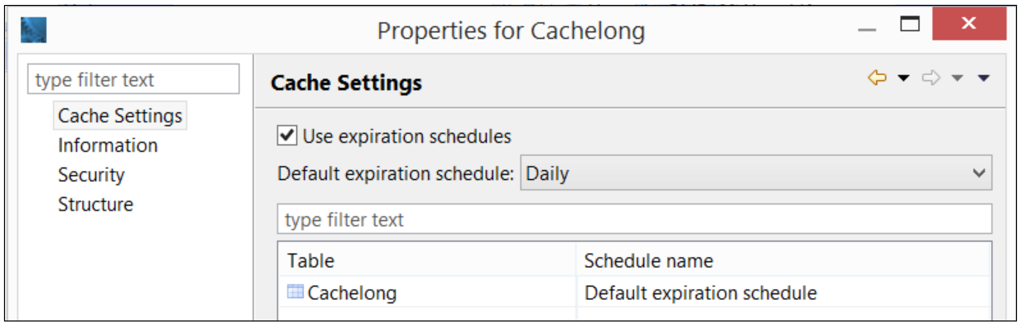


Begin date: 7/ 1/2016
Begin time: 8:04:03 AM

Data Expires
- ( ) Daily
- ( ) Each 1 day
- (●) Days of week:
  - [ ] Sunday
  - [ ] Monday
  - [ ] Tuesday
  - [ ] Wednesday
  - [ ] Thursday
  - [✓] Friday
  - [ ] Saturday

Figure 5-22 Set the details of the schedule and click Finish

4. By right-clicking the virtual table and selecting **Properties**, the administrator enters the cache settings, as shown in Figure 5-23.



[✓] Use expiration schedules
Default expiration schedule: Daily

Table	Schedule name
Cachelong	Default expiration schedule

Figure 5-23 Setting the custom expiration schedule

5. In this case, the administrator set the Daily schedule as the default. The Daily schedule is in effect for all future virtual tables in this virtual database. Alternatively, the administrator can set the schedule for the individual virtual tables on a one-by-one basis, as shown in Figure 5-24.

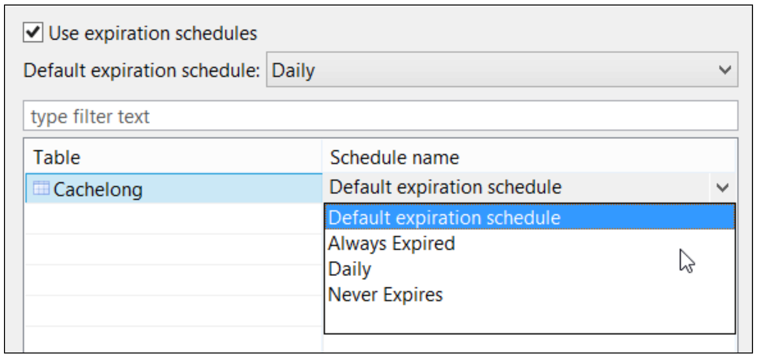

☑ Use expiration schedules
Default expiration schedule: Daily

Table	Schedule name
Cachelong	Default expiration schedule

Figure 5-24 Setting an expiration schedule for an individual query

The query profile results are shown for this virtual table with and without the expiration schedule:

- Without the expiration schedule: 13 seconds
- With the expiration schedule: 0.142 seconds

5.4 Summary of caching types

This chapter covered three types of caching:

- Query caching:
  - Query caching is controlled by the administrator.
  - Users might be authorized to override this caching.
  - User control is on queries that are saved to the catalog or repository only.
  - The result set is cached as a file on the machine that is running QMF and that is specific to the user.
  - This file is not accessible to the user or other applications. This file is not transferable or shareable with other users.
  - The result set cache is refreshed by a change of user, a change of SQL, a redirection to a different data source, and the expiration of the cache.
  - When the cache expires, it is not automatically refreshed by QMF. It is refreshed only when the query is rerun by the user or by a scheduled task.
  - This result set is a static result set. This result set can be used in all QMF objects that use queries.
- Dynamarts:
  - Dynamarts are controlled by the user.
  - Dynamarts are stored as a file on a workstation or as a large object in the repository.
  - As a file, a dynamart is accessible only by QMF but it can be shared with others who have QMF.
  - As a large object in the repository, a dynamart is centrally located and can be shared by all users.

  - Dynamarts do not require a login to the back-end data source to be opened, viewed, and used.
  - Dynamarts require a login to the back-end data source to be refreshed, which happens when they are run or refreshed in a PROC.
  - No timed expiration exists for a dynamart.
  - The result set is a static result set. The result set can be used in all QMF objects that use queries.
- Virtual database:
  - A virtual database is controlled by the administrator.
  - A virtual database is defined centrally in the repository and cached on the local machine when it is run by QMF.
  - Results are cached in a relational database in QMF. Results are specific to the user and machine that are running QMF.
  - Results can be queried with SQL instead of providing a static result.
  - Results can be used by all QMF objects.
  - The refresh schedule is set by the administrator.
  - The refresh is not automatically performed by QMF. After the expiration, the refresh occurs when the table is queried.

All of these caching mechanisms reduce the load on DB2 from many business users who are running business queries that return human-consumable result sets.

For example, a query result that contains 100,000,000 records and 200 columns is most likely meant for a program to consume, not for a person to read and manipulate. In most cases, the machine that is running QMF is a notebook or web server that is dedicated to serving many users and applications. The performance of cached results by any of these methods degrades when the result sets reach into the tens of millions.

The point at which the cache performance degrades depends on the machine that is caching it: how much disk space, how much RAM, and how many other applications are running. So, it is not possible to set a limit, such as “do not cache over 10,000,000 records”. Each user needs to try the caching with simple queries with timestamps to get used to how caching works. Then, the user needs to run large queries to understand what the local system can handle.

6

Key TSO enhancements

Numerous enhancements were made to the Time Sharing Option (TSO) portion of IBM DB2 Query Management Facility for z/OS V11.2.1 (QMF), but we highlight only two enhancements that have the greatest impact on users:

- QMF Analytics for TSO provides a suite of new charts, graphs, and statistical analyses that extend the power of QMF to provide key business information and presentations.
- QMF Enhanced Editor offers a powerful new interface that increases the productivity for long-time users and new users to the TSO interface.

This chapter includes the following topics:

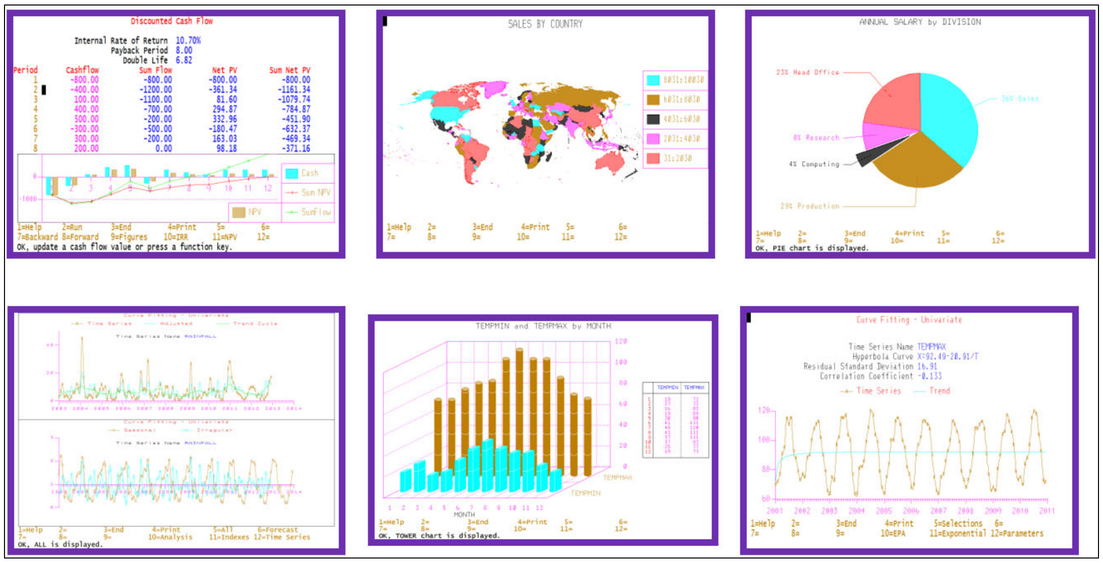
- QMF Analytics for TSO
- QMF Enhanced Editor

Note: For easier reading and printing, the colors in these screen captures are reversed from the usual black background of a TSO terminal.

6.1 QMF Analytics for TSO

QMF previously supported the Interactive Chart Utility (ICU) feature of Graphical Data Display manager (GDDM). GDDM is shipped with z/OS, but ICU is an extra charge. The feature that contains the ICU is called Presentation Graphics Facility (PGF). PGF is required for QMF Analytics for TSO. The significant difference is that the new charting capabilities and the statistical analysis methods are integrated within the QMF framework and easy to use. Now, they can be saved and recovered as a new object type (analytics) within the QMF catalog.

Figure 6-1 shows examples of the types of analysis output that you can produce.


```
                    Discounted Cash Flow

         Internal Rate of Return  10.70%
                  Payback Period  8.00
                     Double Life  6.82
Period     Cashflow      Sum Flow      Net PV    Sum Net PV
     1      -800.00       -800.00     -800.00       -800.00
     2      -400.00      -1200.00     -361.34      -1161.34
     3       100.00      -1100.00       81.60      -1079.74
     4       400.00       -700.00      294.87       -784.87
     5       500.00       -200.00      332.96       -451.90
     6      -300.00       -500.00     -180.47       -632.37
     7       300.00       -200.00      163.03       -469.34
     8       200.00          0.00       98.18       -371.16

1=Help      2=Run      3=End       4=Print   5=      6=
7=Backward  8=Forward  9=Figures   10=IRR    11=NPV  12=
OK, update a cash flow value or press a function key.
```

```
OK, PIE chart is displayed.
```

```
OK, ALL is displayed.
```

```
OK, TOWER chart is displayed.
```

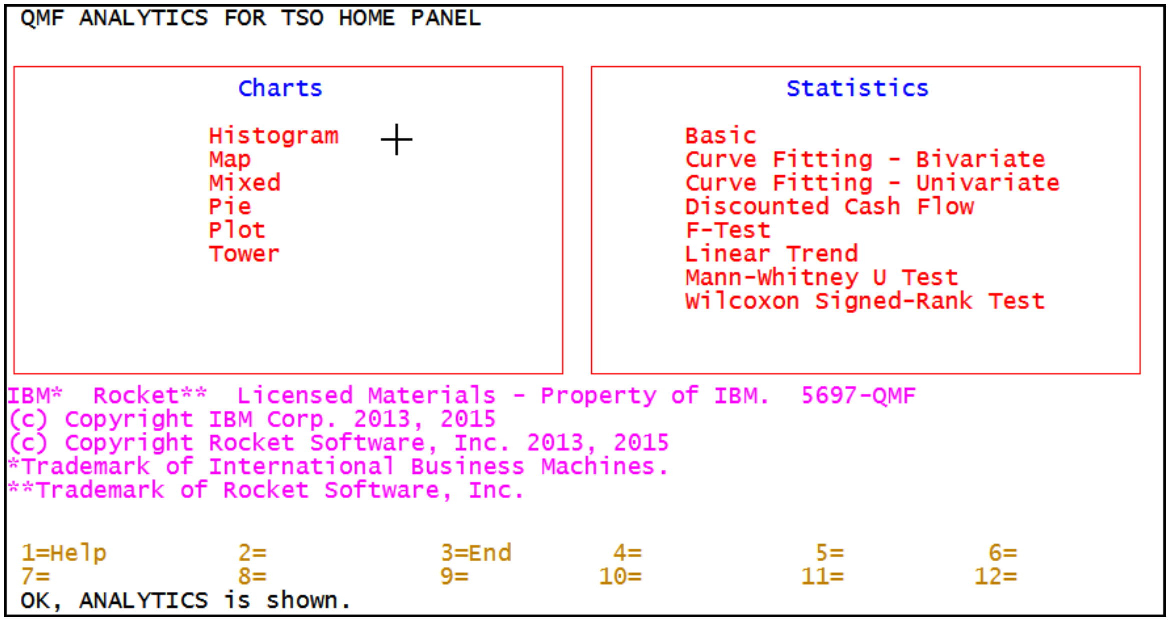
```
                Curve Fitting - Univariate

             Time Series Name TEMPMAX
              Hyperbola Curve X=92.49-20.91/T
  Residual Standard Deviation 16.91
      Correlation Coefficient -0.133

1=Help  2=  3=End  4=Print  5=Selections     6=
7=      8=  9=     10=EPA   11=Exponential   12=Parameters
```

Figure 6-1 Examples of analysis

An overview is presented. Then, the process for working with query analytics is described.

6.1.1 Chart options

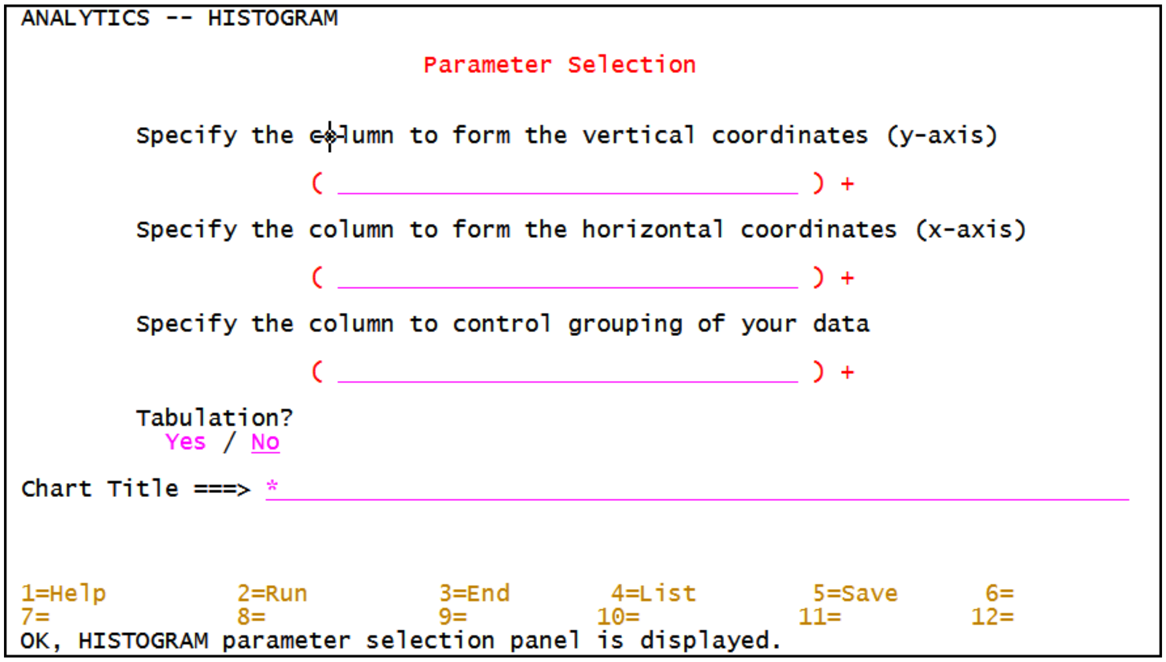
One panel is shown when query analytics is invoked (Figure 6-2). The left side lists the available charts and graphs available. The right side lists the statistical analyses that the user might want to use.



```
QMF ANALYTICS FOR TSO HOME PANEL

              Charts                                  Statistics

            Histogram                           Basic
            Map                                 Curve Fitting - Bivariate
            Mixed                               Curve Fitting - Univariate
            Pie                                 Discounted Cash Flow
            Plot                                F-Test
            Tower                               Linear Trend
                                                Mann-Whitney U Test
                                                Wilcoxon Signed-Rank Test

IBM*  Rocket**  Licensed Materials - Property of IBM.  5697-QMF
(c) Copyright IBM Corp. 2013, 2015
(c) Copyright Rocket Software, Inc. 2013, 2015
*Trademark of International Business Machines.
**Trademark of Rocket Software, Inc.

 1=Help         2=            3=End        4=           5=          6=
 7=             8=            9=          10=          11=         12=
 OK, ANALYTICS is shown.
```

Figure 6-2 The QMF Analytics for the TSO home panel

The model for both graphs and statistics assumes that the user does not have in-depth knowledge in either area and does not want to learn a new set of commands and functions for these extensions. Therefore, the interface for each of the options contains a single panel with fields to enter and a way to prompt the user for values if they do not know what is available, as shown in Figure 6-3, where we selected the histogram option.



```
ANALYTICS -- HISTOGRAM
                          Parameter Selection

        Specify the column to form the vertical coordinates (y-axis)
                    ( ________________________________ ) +
        Specify the column to form the horizontal coordinates (x-axis)
                    ( ________________________________ ) +
        Specify the column to control grouping of your data
                    ( ________________________________ ) +
        Tabulation?
          Yes / No
Chart Title ===> *____________________________________________________________

1=Help         2=Run         3=End        4=List       5=Save      6=
7=             8=            9=          10=          11=         12=
OK, HISTOGRAM parameter selection panel is displayed.
```

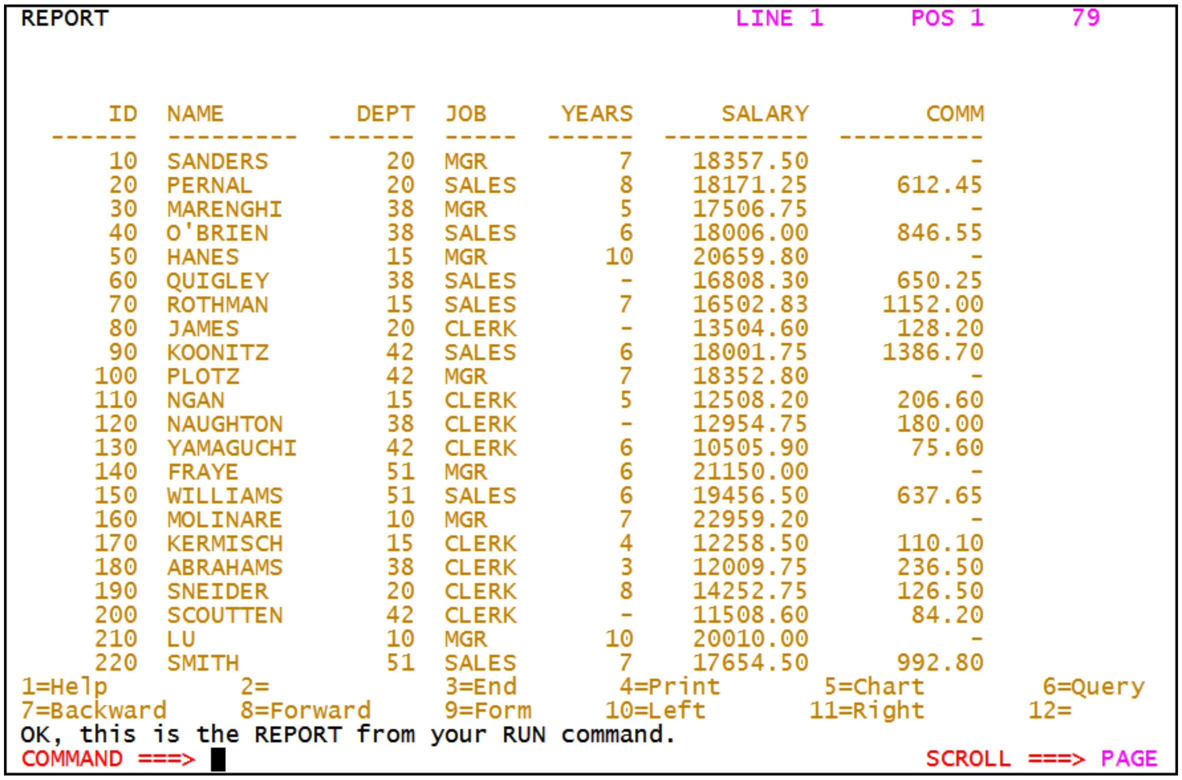
Figure 6-3 QMF Analytics for TSO histogram panel

Each analysis or chart depends on a previously executed query. In Figure 6-3 on page 175, the fields are blank. The user can press PF4 to get a pick list of the available columns. The tabulation option is a toggle switch between yes and no to control whether a table of values is shown next to the chart.

We begin with this example query:

```
SELECT * FROM Q.STAFF
```

After we execute the query (press PF2 or type `RUN` on the command line), we view our results, as shown in Figure 6-4. Q.STAFF contains these rows.



```
REPORT                                              LINE 1      POS 1      79



      ID  NAME          DEPT  JOB      YEARS      SALARY        COMM
  ------  ---------   ------  -----   ------  ----------  ----------
      10  SANDERS         20  MGR          7    18357.50           -
      20  PERNAL          20  SALES        8    18171.25      612.45
      30  MARENGHI        38  MGR          5    17506.75           -
      40  O'BRIEN         38  SALES        6    18006.00      846.55
      50  HANES           15  MGR         10    20659.80           -
      60  QUIGLEY         38  SALES        -    16808.30      650.25
      70  ROTHMAN         15  SALES        7    16502.83     1152.00
      80  JAMES           20  CLERK        -    13504.60      128.20
      90  KOONITZ         42  SALES        6    18001.75     1386.70
     100  PLOTZ           42  MGR          7    18352.80           -
     110  NGAN            15  CLERK        5    12508.20      206.60
     120  NAUGHTON        38  CLERK        -    12954.75      180.00
     130  YAMAGUCHI       42  CLERK        6    10505.90       75.60
     140  FRAYE           51  MGR          6    21150.00           -
     150  WILLIAMS        51  SALES        6    19456.50      637.65
     160  MOLINARE        10  MGR          7    22959.20           -
     170  KERMISCH        15  CLERK        4    12258.50      110.10
     180  ABRAHAMS        38  CLERK        3    12009.75      236.50
     190  SNEIDER         20  CLERK        8    14252.75      126.50
     200  SCOUTTEN        42  CLERK        -    11508.60       84.20
     210  LU              10  MGR         10    20010.00           -
     220  SMITH           51  SALES        7    17654.50      992.80
1=Help         2=            3=End      4=Print        5=Chart        6=Query
7=Backward     8=Forward     9=Form     10=Left        11=Right       12=
OK, this is the REPORT from your RUN command.
COMMAND ===>                                                 SCROLL ===> PAGE
```

Figure 6-4 Q.STAFF data

Now, we have the results to act on, so how do we invoke the new analytics panel? Use this command:

`SHOW ANALYTICS`

If you are familiar with QMF, you know that you can truncate any command to its fewest characters. The **`SHOW`** command might be truncated to **`SH`**. No other option or command in QMF begins with an A (yet). Therefore, we can shorten the command this way:

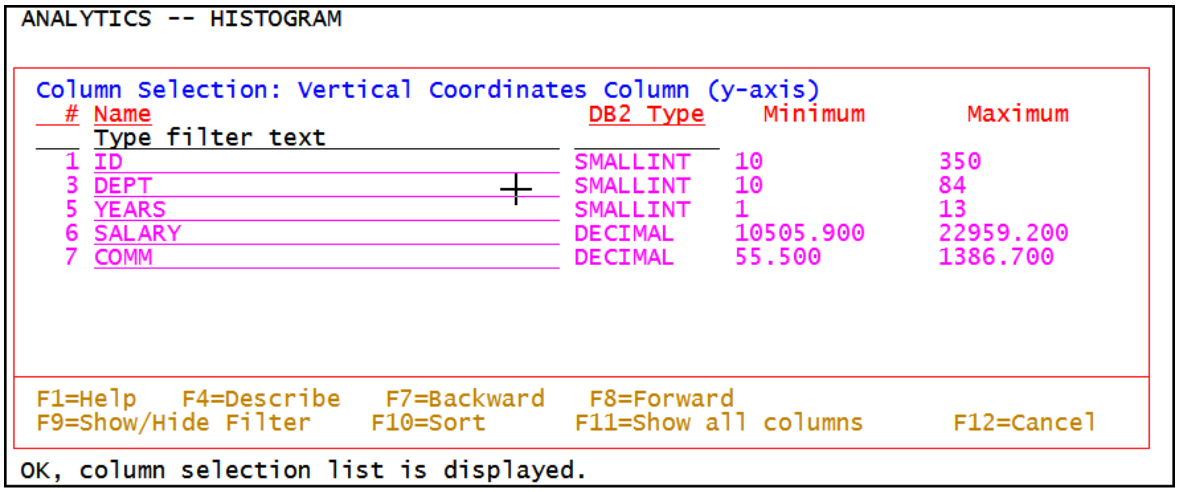
`SH A`

This command invokes the main panel that we showed you in Figure 6-2 on page 175. If the query did not run, we cannot invoke analytics. Without data to act on, the user receives an error:

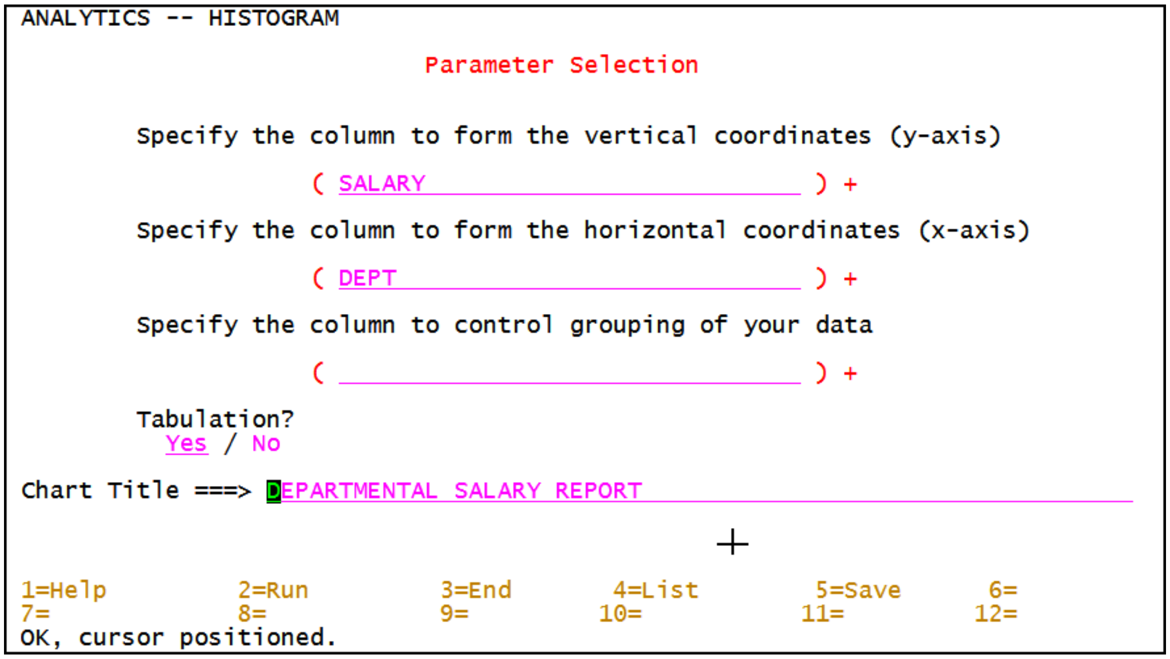
```
"No DATA object exists for SHOW ANALYTICS"
```

We return to our analytics panel. Assume that we successfully executed the query to display the data in Q.STAFF. We take the histogram option and we see the panel that is displayed in Figure 6-3 on page 175. What are the column names? If the user places the cursor on one of the fields, requests a column, and presses PF4, or mouse-clicks the input field, the user sees a list of the available columns, as shown in Figure 6-5.

Note: Initially, the list is limited to the columns of the appropriate DB2 type for the selection only. Pressing F11 toggles between displaying all columns and appropriate columns. If many columns exist, you can easily limit the list by entering a filter at the top by using a percent sign (%) as a wildcard. The minimum and maximum values for a column are listed to help you with column selection. More details about a column are available by using the F4=Describe function key.


```
ANALYTICS -- HISTOGRAM

 Column Selection: Vertical Coordinates Column (y-axis)
   # Name                                  DB2 Type    Minimum       Maximum
 ___ Type filter text                     __________
   1 ID                                  SMALLINT    10            350
   3 DEPT                                SMALLINT    10            84
   5 YEARS                               SMALLINT    1             13
   6 SALARY                              DECIMAL     10505.900     22959.200
   7 COMM                                DECIMAL     55.500        1386.700

 F1=Help   F4=Describe   F7=Backward   F8=Forward
 F9=Show/Hide Filter    F10=Sort      F11=Show all columns      F12=Cancel
OK, column selection list is displayed.
```
Figure 6-5 Selecting analytics options from a list

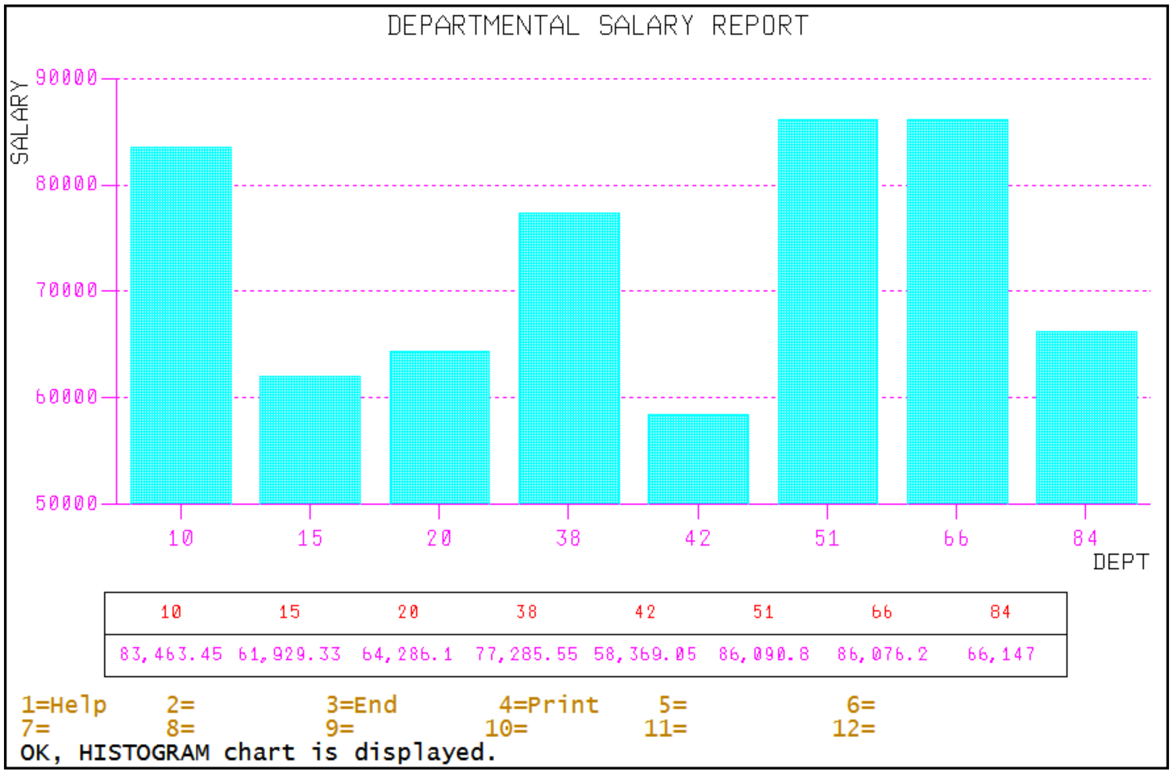
In our example, we plot the sum of SALARY (y-axis) by DEPT (x-axis). We pressed PF4 in both cases and entered the options, as shown in Figure 6-6.



```
ANALYTICS -- HISTOGRAM
                              Parameter Selection

        Specify the column to form the vertical coordinates (y-axis)

                    ( SALARY                              ) +

        Specify the column to form the horizontal coordinates (x-axis)

                    ( DEPT                                ) +

        Specify the column to control grouping of your data

                    (                                     ) +

        Tabulation?
          Yes / No

Chart Title ===> DEPARTMENTAL SALARY REPORT


1=Help         2=Run         3=End       4=List        5=Save      6=
7=             8=            9=         10=           11=         12=
OK, cursor positioned.
```

Figure 6-6 Entering the histogram options by using PF4 to select from a list

So, an x-axis, a y-axis, tabulation, and a title are set. To display the results of our options, we press PF2 to run it. Our results are shown in Figure 6-7.



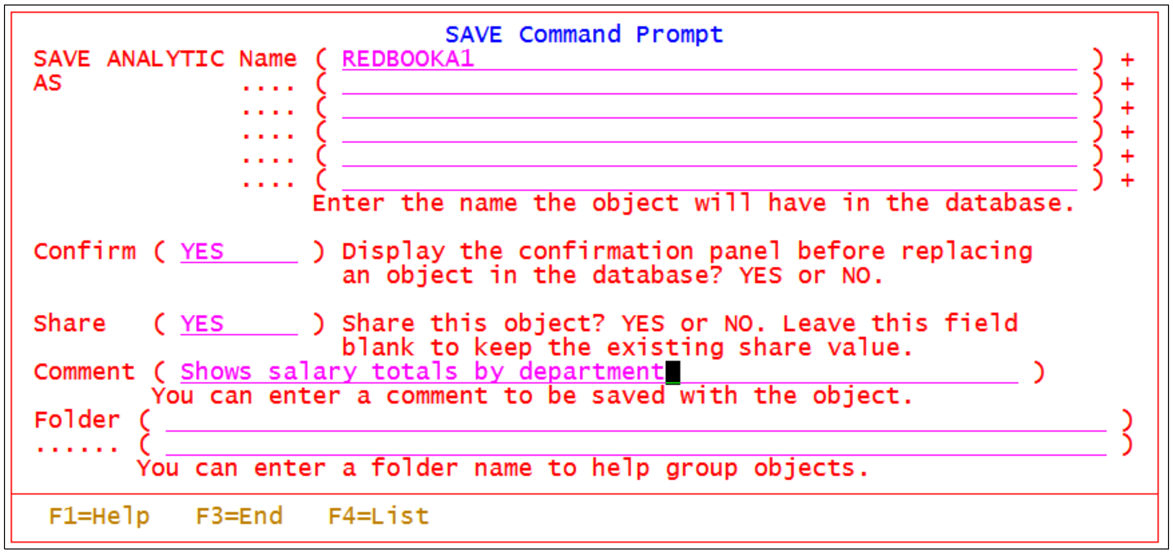
10	15	20	38	42	51	66	84
83,463.45	61,929.33	64,286.1	77,285.55	58,369.05	86,090.8	86,076.2	66,147

```
1=Help     2=        3=End       4=Print     5=          6=
7=         8=        9=         10=         11=         12=
OK, HISTOGRAM chart is displayed.
```

Figure 6-7 Results of our histogram option selections

From the chart display panel, you can copy the results to the clipboard to paste into an external application, such as email, or you can print to a configured printer.

Assume that we want to keep this chart specification available for subsequent usage. We return to the parameters panel and use the F5 save option to display the panel that is shown in Figure 6-8.



```
                               SAVE Command Prompt
SAVE ANALYTIC Name ( REDBOOKA1                                          ) +
AS             .... (                                                   ) +
               .... (                                                   ) +
               .... (                                                   ) +
               .... (                                                   ) +
               .... (                                                   ) +
                    Enter the name the object will have in the database.

Confirm ( YES     ) Display the confirmation panel before replacing
                      an object in the database? YES or NO.

Share   ( YES     ) Share this object? YES or NO. Leave this field
                      blank to keep the existing share value.
Comment ( Shows salary totals by department                       )
        You can enter a comment to be saved with the object.
Folder (                                                             )
...... (                                                             )
       You can enter a folder name to help group objects.

  F1=Help    F3=End    F4=List
```

Figure 6-8 Saving our chart options

In our example, if we press Enter, the user is notified that the object is saved in the database. Working with the charting options of QMF Analytics for TSO is simple.

Note: You cannot save the parameters from the display panel. The options are easy to use and limited to keep the user within a simplified environment. One reason that the Save action is available on the parameters panel and not the display panel is to help clarify that the parameters are saved, not the rendered chart image.

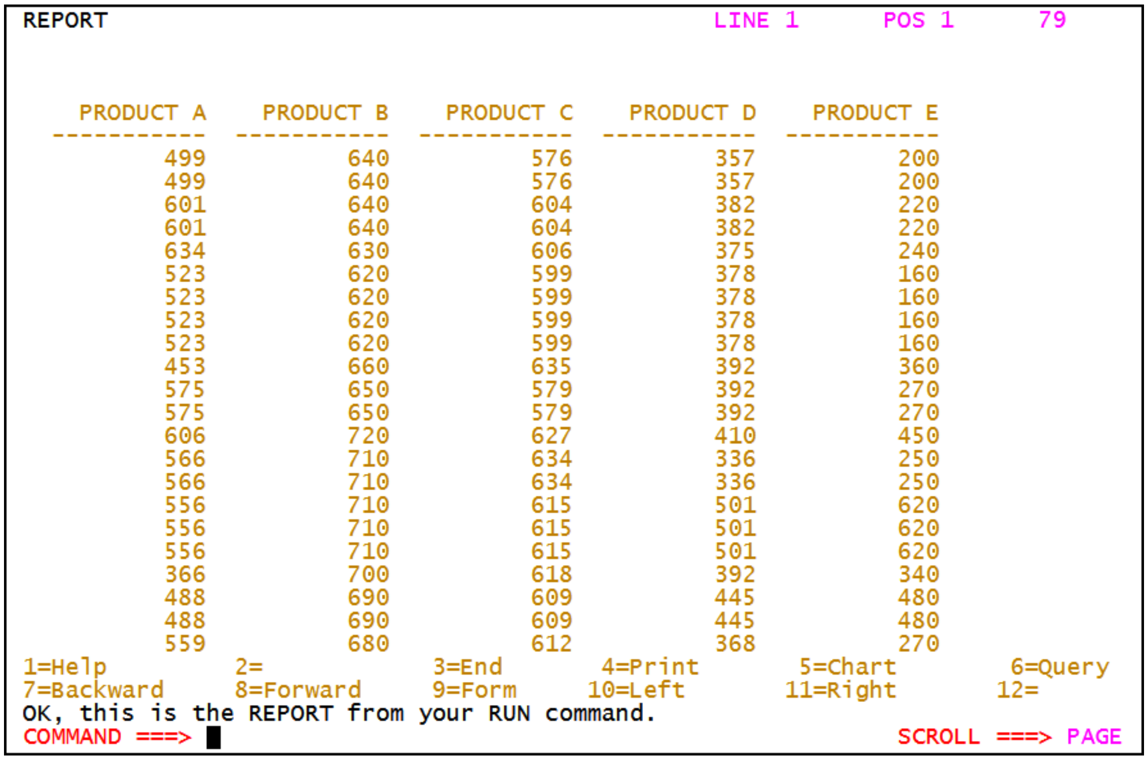
Now, if we return to QMF by ending the session (pressing PF3) we are back in our traditional QMF environment. How do we locate the object that we saved? If you are familiar with the `LIST` command, several options are available. To list your own analytics objects if you cannot remember the name, enter one of these commands:

- Use the `LIST ANALYTICS` command. The shortest version of this command is `LI AN`.
- Use the `LIST ?` command for a formatted panel where you can use wildcards and more.

To run the analysis again by using the current QMF data and display the chart, use the `DISPLAY` command against the listed analytics object.

6.1.2 Statistical analyses options

To illustrate our next example, we run a different query that tracks product breakage numbers by product. Five products, PRODUCT A through PRODUCT E, are shown in Figure 6-9.

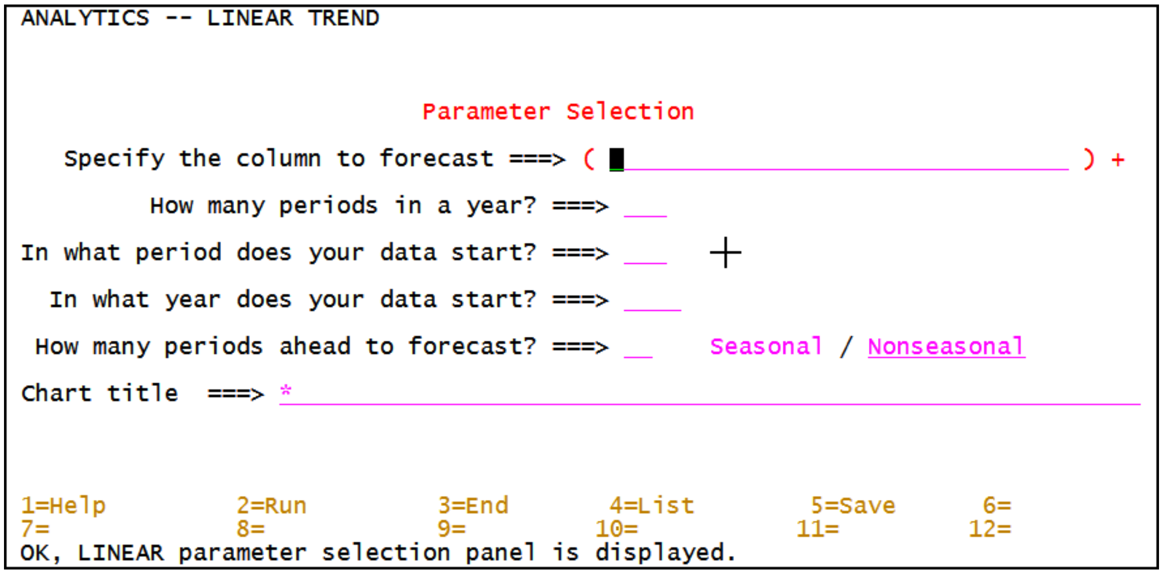


```
REPORT                                                   LINE 1      POS 1       79


   PRODUCT A    PRODUCT B    PRODUCT C    PRODUCT D    PRODUCT E
  -----------  -----------  -----------  -----------  -----------
          499          640          576          357          200
          499          640          576          357          200
          601          640          604          382          220
          601          640          604          382          220
          634          630          606          375          240
          523          620          599          378          160
          523          620          599          378          160
          523          620          599          378          160
          523          620          599          378          160
          453          660          635          392          360
          575          650          579          392          270
          575          650          579          392          270
          606          720          627          410          450
          566          710          634          336          250
          566          710          634          336          250
          556          710          615          501          620
          556          710          615          501          620
          556          710          615          501          620
          366          700          618          392          340
          488          690          609          445          480
          488          690          609          445          480
          559          680          612          368          270
1=Help         2=            3=End        4=Print       5=Chart         6=Query
7=Backward     8=Forward     9=Form      10=Left       11=Right        12=
OK, this is the REPORT from your RUN command.
COMMAND ===>                                                        SCROLL ===> PAGE
```

Figure 6-9 Results of running a query against product breakage numbers by product

No time or date information is shown in this table so we can use it in any manner that we want. Assume that the breakage figures are monthly and tracking began in January 2014. What we want to determine is what trend we might see with PRODUCT B because PRODUCT B is problematic. We want to see what we might expect over the next 36 months so that we can either predict our losses or use the data to convince manufacturing that changes must occur.

By using the data that is displayed in Figure 6-9 on page 180, we invoke analytics by using the SHOW ANALYTICS that we described previously. This time, we select the option for a LINEAR TREND on the right side. The resulting panel is shown in Figure 6-10.

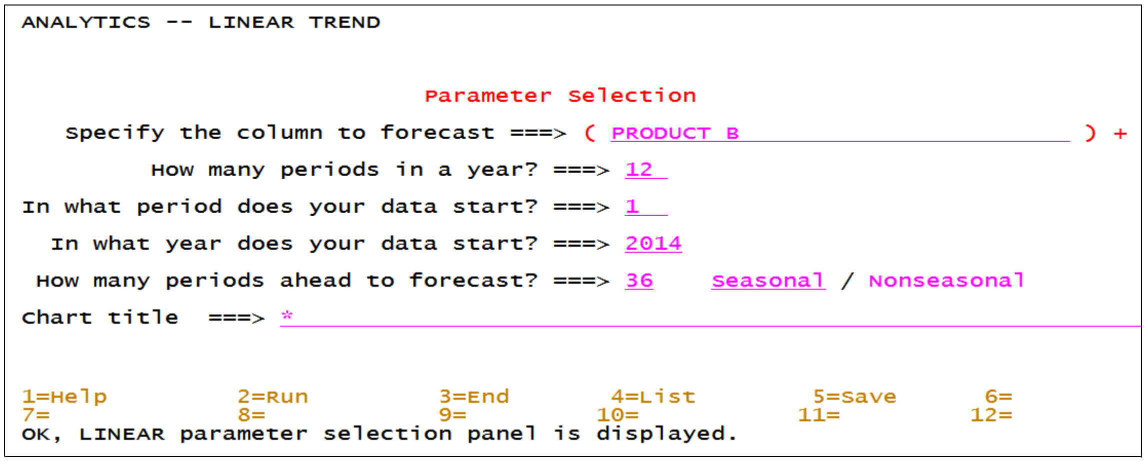


```
ANALYTICS -- LINEAR TREND


                           Parameter Selection

   Specify the column to forecast ===> (                               ) +

         How many periods in a year? ===> ___

In what period does your data start? ===> ___

  In what year does your data start? ===> ____

 How many periods ahead to forecast? ===> __     Seasonal / Nonseasonal

Chart title   ===> *


1=Help         2=Run          3=End       4=List        5=Save      6=
7=             8=             9=         10=           11=         12=
OK, LINEAR parameter selection panel is displayed.
```

Figure 6-10   The linear trend options

We are presented with a formatted panel with several options to complete. If the data is seasonal (predictable by different times of the year to vary), we select the seasonal option. In our case, we have a consistent pattern of breakage.

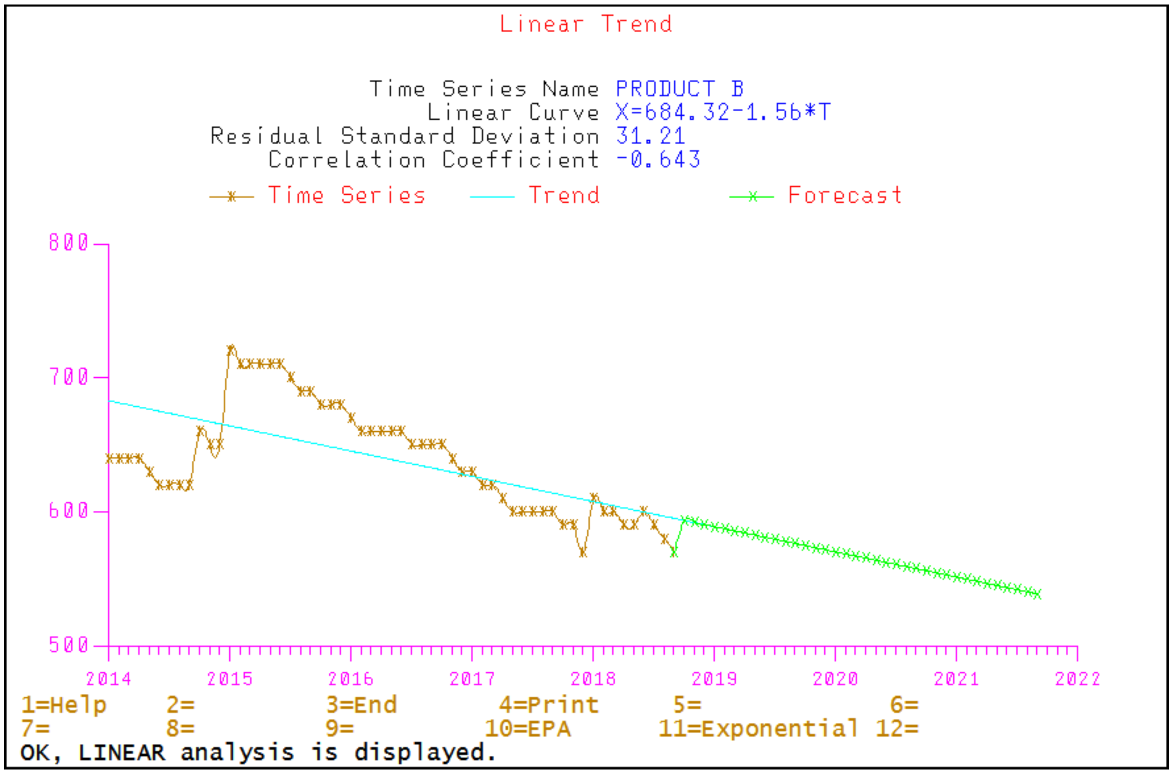
We fill out the options, as shown in Figure 6-11.



```
ANALYTICS -- LINEAR TREND


                           Parameter Selection

   Specify the column to forecast ===> ( PRODUCT B                     ) +

         How many periods in a year? ===> 12

In what period does your data start? ===> 1

  In what year does your data start? ===> 2014

 How many periods ahead to forecast? ===> 36     Seasonal / Nonseasonal

Chart title   ===> *


1=Help         2=Run          3=End       4=List        5=Save      6=
7=             8=             9=         10=           11=         12=
OK, LINEAR parameter selection panel is displayed.
```

Figure 6-11   Specifying the values for a linear trend

When we run the analysis by pressing PF2, we see a brown line of data and an extended line of data in a different color (Figure 6-12).

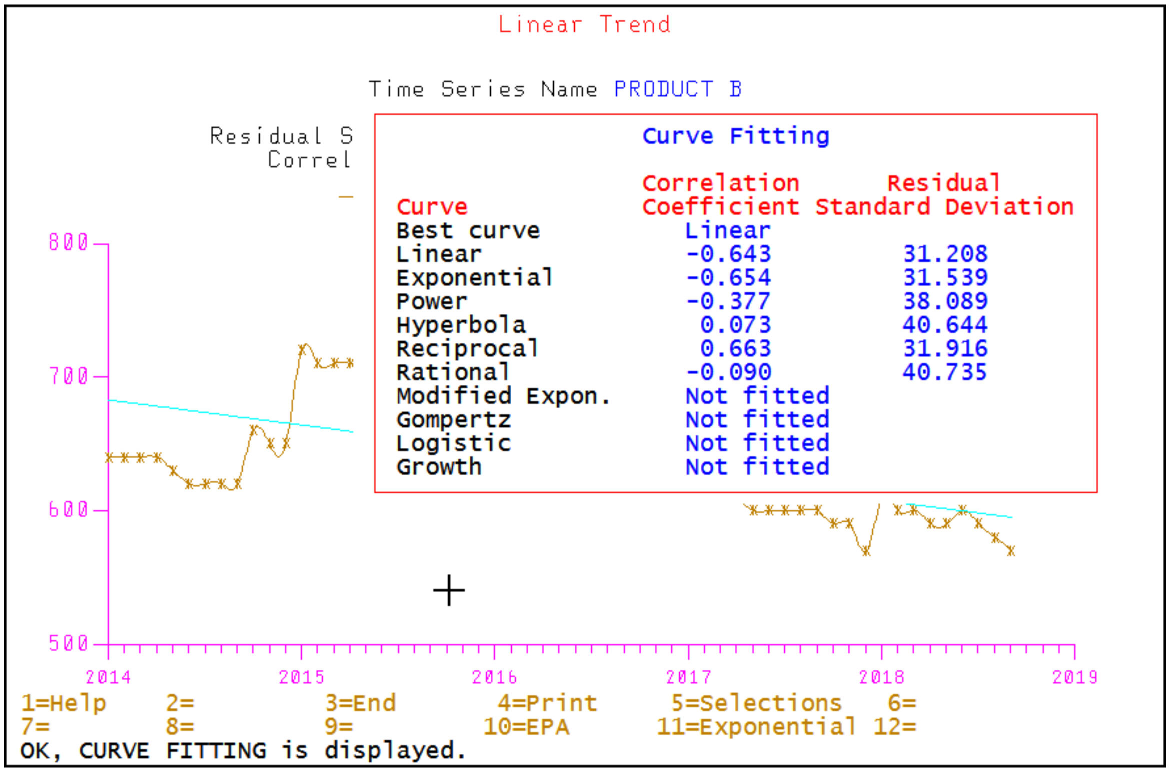


```
          Time Series Name PRODUCT B
              Linear Curve X=684.32-1.56*T
Residual Standard Deviation 31.21
    Correlation Coefficient -0.643

1=Help     2=          3=End        4=Print       5=                 6=
7=         8=          9=          10=EPA        11=Exponential  12=
OK, LINEAR analysis is displayed.
```

Figure 6-12   Linear trend results

Apparently, perception is not reality because the trend appears to go down with time. Other display options are available, as shown by PF10 and PF11. More options are available, depending on which path of analysis you take.

One path is illustrated in Figure 6-13. Query analytics for TSO makes several decisions for the user transparently, such as determining that the linear trend option is the best fit for the available data, but it also executes several others as shown. The user optionally can switch to another method. Certain methods are rejected as inappropriate for the data. Those rejected methods are labeled “not fitted”.



```
                        Curve Fitting

                        Correlation      Residual
Curve                   Coefficient  Standard Deviation
Best curve                 Linear
Linear                     -0.643          31.208
Exponential                -0.654          31.539
Power                      -0.377          38.089
Hyperbola                   0.073          40.644
Reciprocal                  0.663          31.916
Rational                   -0.090          40.735
Modified Expon.            Not fitted
Gompertz                   Not fitted
Logistic                   Not fitted
Growth                     Not fitted
```

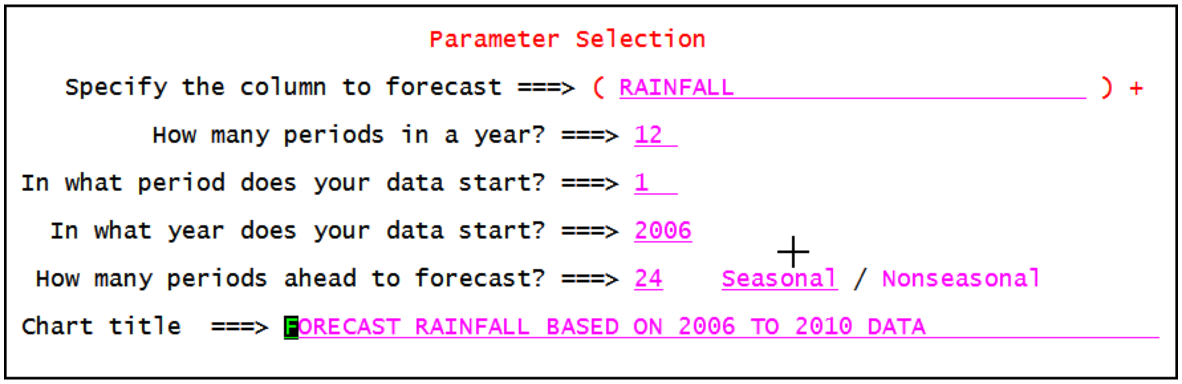
```
1=Help     2=          3=End       4=Print      5=Selections    6=
7=         8=          9=         10=EPA       11=Exponential  12=
OK, CURVE FITTING is displayed.
```

Figure 6-13 Linear trend results with other options

Another example is using QMF sample data with the table Q.CLIMATE_10YR. This sample table includes many years of climate data, such as rainfall by month. By using the last 5 years of this data, we can project the next 2 years of rain. The query is shown:

```
SELECT * FROM Q.CLIMATE_10YR WHERE YEAR > 2005
```

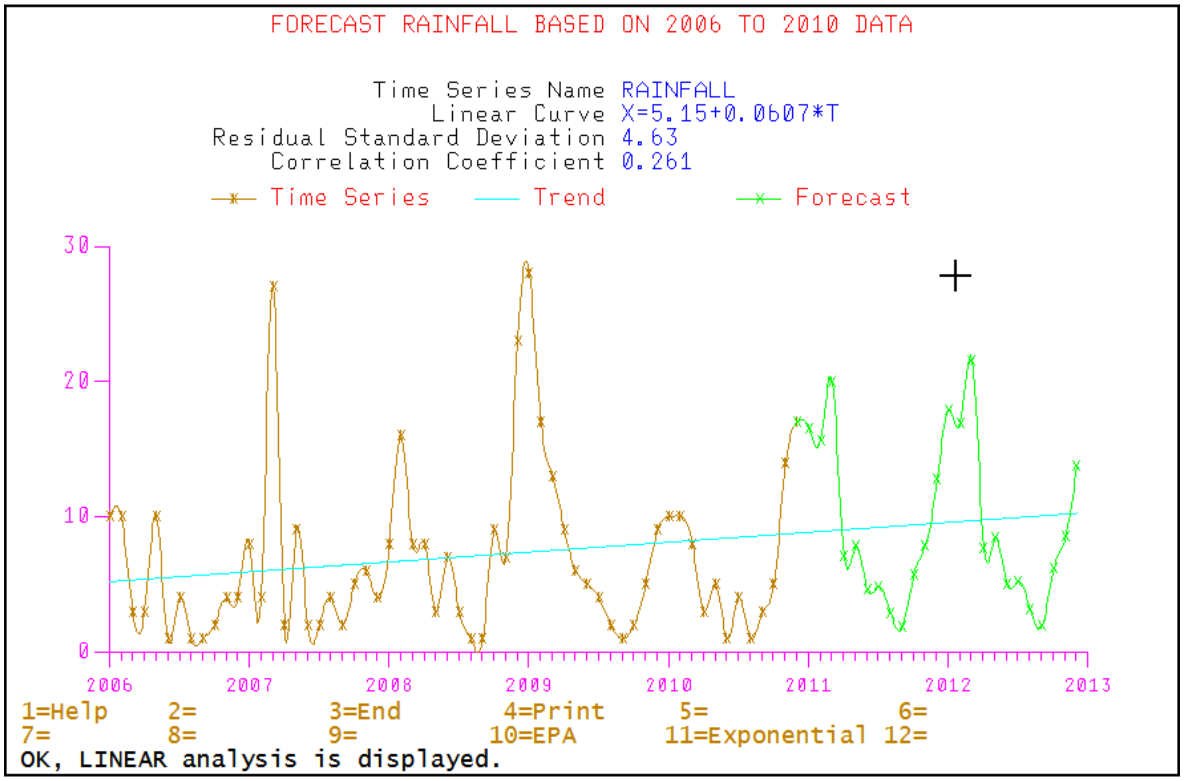
Your parameters in order for the linear trend are shown in Figure 6-14. This time, the **Seasonal** parameter is selected because the data is seasonal.



```
                         Parameter Selection

   Specify the column to forecast ===> ( RAINFALL                      ) +

         How many periods in a year? ===> 12

In what period does your data start? ===> 1

  In what year does your data start? ===> 2006

 How many periods ahead to forecast? ===> 24    Seasonal / Nonseasonal

Chart title   ===> FORECAST RAINFALL BASED ON 2006 TO 2010 DATA
```

Figure 6-14 Parameters for forecasting rainfall by using the linear trend method

Figure 6-15 shows the results.



```
       Time Series Name RAINFALL
           Linear Curve X=5.15+0.0607*T
Residual Standard Deviation 4.63
    Correlation Coefficient 0.261

1=Help     2=          3=End       4=Print      5=              6=
7=         8=          9=         10=EPA       11=Exponential  12=
OK, LINEAR analysis is displayed.
```

Figure 6-15 Predicting annual rainfall as part of a linear trend

This new set of QMF analyses and charts are easy to use, require little skill, and guide you by using simple menu options.

For example, you do not need to know how to formulate a complex statistical analysis statement to use a function, such as linear trending. You need to know your data and how to query it but the rest is easy.

In the next section, we show you how editing is much easier and how the QMF editing capabilities are enhanced to make you more productive.

6.2 QMF Enhanced Editor

QMF contains its own edit panel to create and change queries and procedures. It is a simple text editor that accommodates the basic 3270 panel types. Therefore, it is limited in function.

The user also can invoke the Interactive System Productivity Facility (ISPF) editor and pass QMF query and procedure lines back and forth. Functions, such as copying lines, repeating lines, and more, are available in this editor. However, no direct interaction exists with QMF, such as the ability to run an edited object within ISPF.

Important: To use the ISPF editor or the QMF Enhanced Editor, QMF must be started under ISPF.

6.2.1 Invoking editing options in QMF

Different editing options are available in QMF. The three most widely used options are listed:

- Standard editor
- ISPF editor
- QMF Enhanced Editor

To invoke the QMF Enhanced Editor, you must be running QMF under ISPF, which appears to be the most common environment in use today. Global variable settings are available in the QMF Enhanced Editor (EE) that automatically make it the default editor. First, we cover the three edit modes:

- QMF editor (default) from the home panel. Use PF6 for queries or PF10 for procedures.
- ISPF. You can pre-allocate a data set by using this command: `ALLOC FI(DSQEDIT) UNIT(SYSDA) NEW`. Or, you can enter this command from QMF:

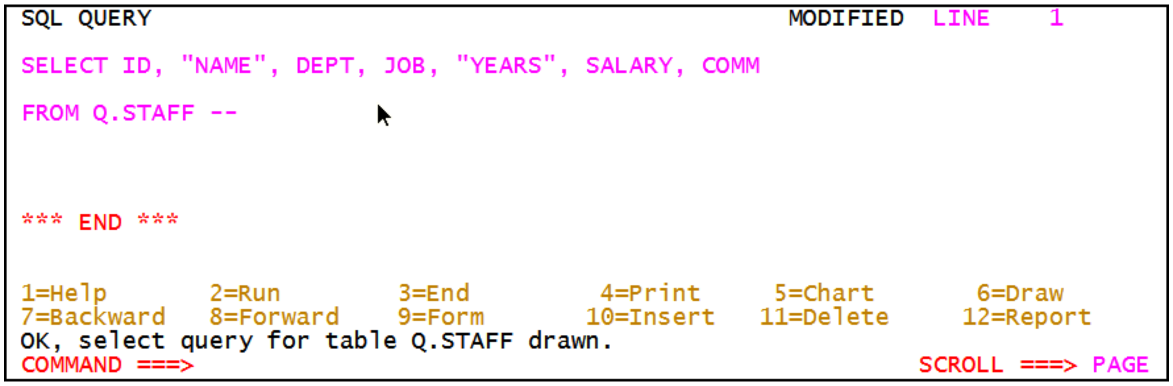
  ```
  EDIT [query/proc/name] (EDIT = PDF
  ```

- For the QMF Enhanced Editor, you enter this command:

  ```
  EDIT [query/proc/name] (EDIT = EE
  ```

We start with the QMF Enhanced Editor major functional areas. We cover global variable settings for the QMF Enhanced Editor last. For more information, see the *Enhanced Editor Overview* PDF document in the additional material that is supplied with this book.

Start with a query where we used the `DRAW [table]` command in QMF. Figure 6-16 shows the results where we issued the command `DRAW Q.STAFF`. You must be on the QUERY EDIT panel (PF6).



```
SQL QUERY                                                  MODIFIED  LINE    1
SELECT ID, "NAME", DEPT, JOB, "YEARS", SALARY, COMM
FROM Q.STAFF --


*** END ***

1=Help       2=Run        3=End         4=Print     5=Chart       6=Draw
7=Backward   8=Forward    9=Form       10=Insert   11=Delete     12=Report
OK, select query for table Q.STAFF drawn.
COMMAND ===>                                                   SCROLL ===> PAGE
```

Figure 6-16 DRAW Q.STAFF

The DRAW command is used by many QMF users so that QMF writes a SELECT statement that shows all of the columns in a table or multiple tables.

Note: QMF uses a comma to specify delineation between columns.

We used a small table for this example. Typically, tables have more columns.

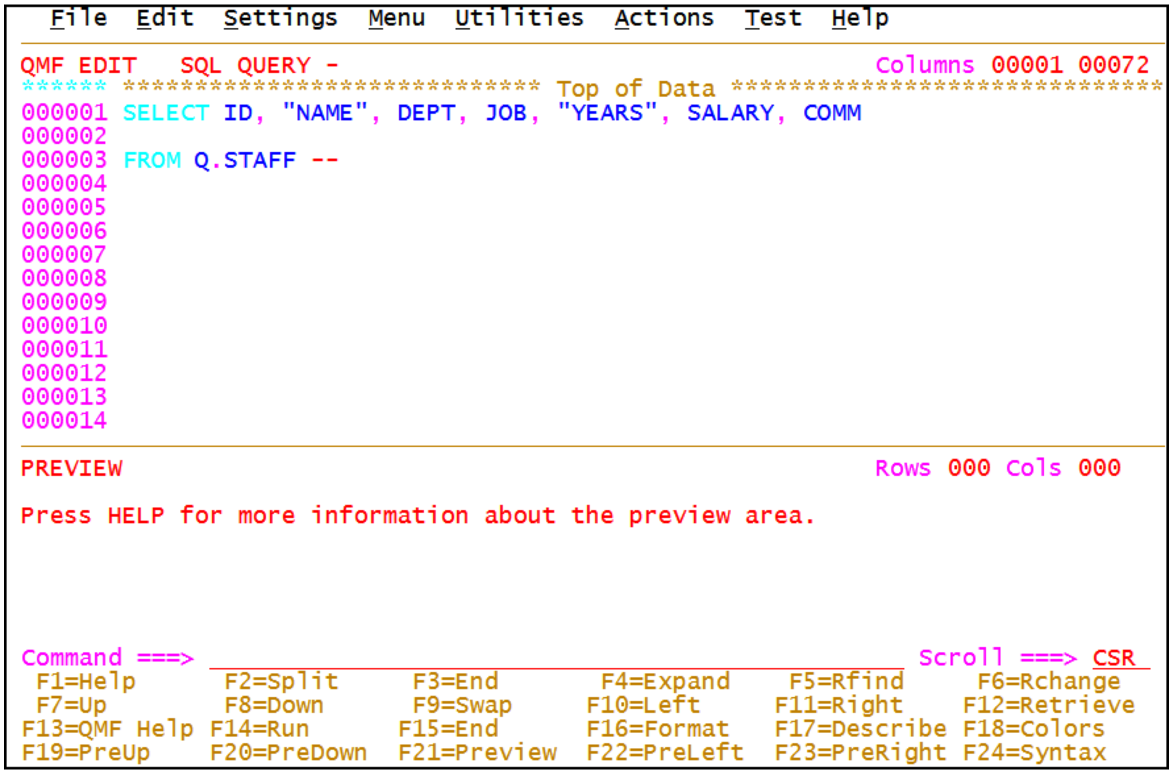
Assume that you want to use the AS statement to rename a number of columns. You do not have enough room on any line to insert the characters that you want. For example, you want ID to show as PERSONAL_IDENTIFICATION, DEPT to show as DEPARTMENT_REPORTING_TO, and COMM to show as COMMISSION.

By using the traditional QMF editor, you must use the cursor to move to the point where you want to split the line and press PF10 to insert.

If you used other editors, you probably press Enter at the wrong times and watch the cursor simply jump back and forth from the command line to the query statements. But, if you use the QMF Enhanced Editor, you can enter the following statement:

```
EDIT QUERY (EDIT=EE
```

Then, you invoke EE and see a panel that is similar to Figure 6-17.



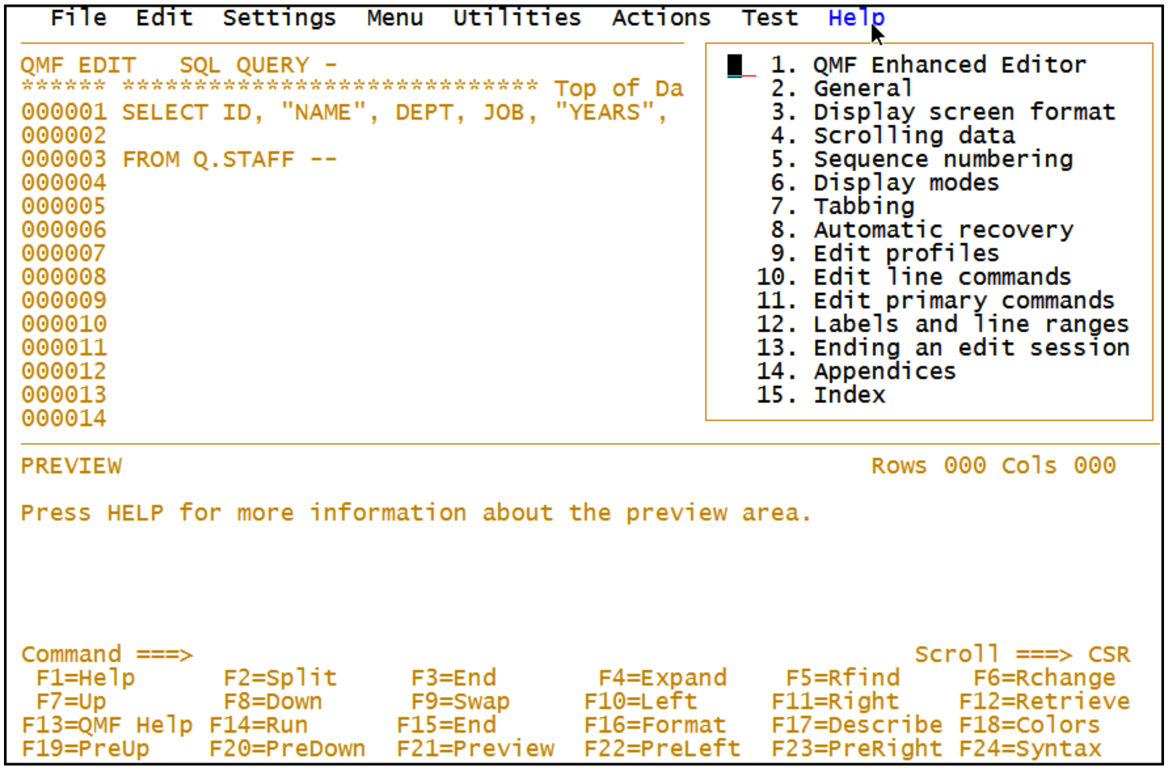
```
   File  Edit  Settings  Menu  Utilities  Actions  Test  Help

QMF EDIT   SQL QUERY -                                     Columns 00001 00072
****** ****************************** Top of Data ******************************
000001 SELECT ID, "NAME", DEPT, JOB, "YEARS", SALARY, COMM
000002
000003 FROM Q.STAFF --
000004
000005
000006
000007
000008
000009
000010
000011
000012
000013
000014

PREVIEW                                                    Rows 000 Cols 000

Press HELP for more information about the preview area.




Command ===> ______________________________________________  Scroll ===> CSR
 F1=Help      F2=Split     F3=End       F4=Expand    F5=Rfind     F6=Rchange
 F7=Up        F8=Down      F9=Swap     F10=Left     F11=Right    F12=Retrieve
F13=QMF Help F14=Run      F15=End      F16=Format   F17=Describe F18=Colors
F19=PreUp    F20=PreDown  F21=Preview  F22=PreLeft  F23=PreRight F24=Syntax
```

Figure 6-17 Use of the QMF Enhanced Editor

Immediately, you see that lines are available for each statement. We describe the action bars later. Now, standard ISPF options are available, such as copying lines or deleting lines. This book is not a tutorial for ISPF, but you can find many references within the HELP options or online ISPF documentation.

Note: In Figure 6-17, ISPF shows 24 function key (PF) settings across four lines at the bottom of the panel. You can tailor this display by using the **PFSHOW TAILOR** command, or hide function keys by using the **PFSHOW OFF** command to allow more space for editing.

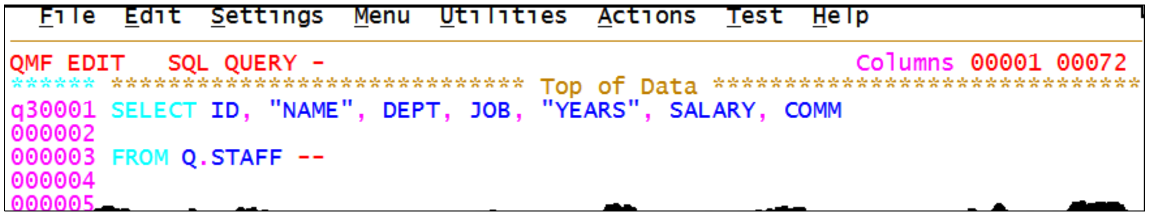
We suggest that you take time to browse the online help that is in the QMF Enhanced Editor, as shown in Figure 6-18. The format and syntax of line edit commands are contained in this facility.



```
  File  Edit  Settings  Menu  Utilities  Actions  Test  Help
QMF EDIT   SQL QUERY -                            █_ 1. QMF Enhanced Editor
****** ****************************** Top of Da      2. General
000001 SELECT ID, "NAME", DEPT, JOB, "YEARS",        3. Display screen format
000002                                               4. Scrolling data
000003 FROM Q.STAFF --                               5. Sequence numbering
000004                                               6. Display modes
000005                                               7. Tabbing
000006                                               8. Automatic recovery
000007                                               9. Edit profiles
000008                                              10. Edit line commands
000009                                              11. Edit primary commands
000010                                              12. Labels and line ranges
000011                                              13. Ending an edit session
000012                                              14. Appendices
000013                                              15. Index
000014
PREVIEW                                                         Rows 000 Cols 000

Press HELP for more information about the preview area.




Command ===>                                                       Scroll ===> CSR
 F1=Help      F2=Split     F3=End       F4=Expand    F5=Rfind     F6=Rchange
 F7=Up        F8=Down      F9=Swap     F10=Left     F11=Right    F12=Retrieve
F13=QMF Help F14=Run      F15=End      F16=Format   F17=Describe F18=Colors
F19=PreUp    F20=PreDown  F21=Preview  F22=PreLeft  F23=PreRight F24=Syntax
```

Figure 6-18 Using HELP in the QMF Enhanced Editor

One of the many available commands when you use the QMF Enhanced Editor is a Query formatting line command **Q**, which makes it easier to read your queries. To format three lines from line 1, we enter `q3` in the prefix area of row 1 (Figure 6-19).



```
  File  Edit  Settings  Menu  Utilities  Actions  Test  Help
QMF EDIT   SQL QUERY -                                    Columns 00001 00072
****** ****************************** Top of Data ******************************
q30001 SELECT ID, "NAME", DEPT, JOB, "YEARS", SALARY, COMM
000002
000003 FROM Q.STAFF --
000004
000005
```

Figure 6-19 Formatting a query with the Q line command

After you press Enter, the query is formatted, as shown in Figure 6-20.

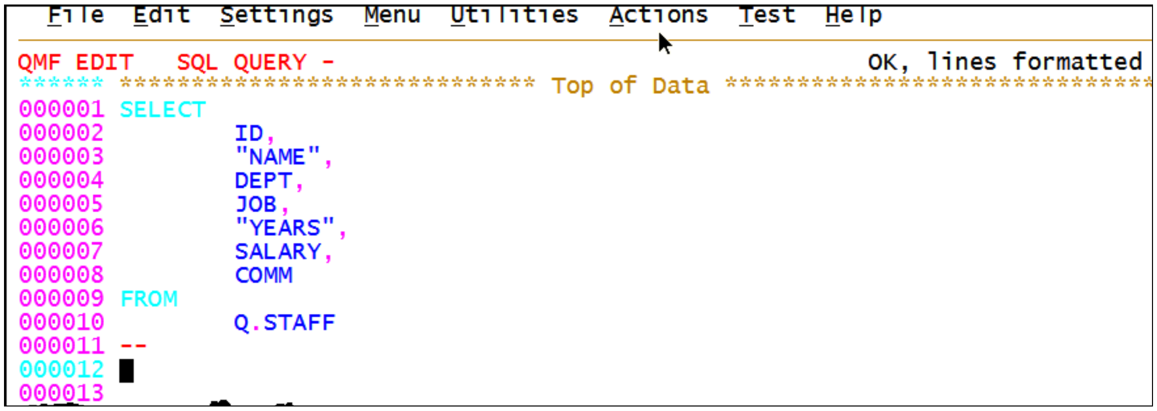

```
  File  Edit  Settings  Menu  Utilities  Actions  Test  Help
QMF EDIT   SQL QUERY -                                     OK, lines formatted
****** ***************************** Top of Data ******************************
000001 SELECT
000002         ID,
000003         "NAME",
000004         DEPT,
000005         JOB,
000006         "YEARS",
000007         SALARY,
000008         COMM
000009 FROM
000010         Q.STAFF
000011 --
000012 
000013
```

Figure 6-20 Formatted query

With each column on its own line, enough room exists to append additional text. When we finish, our query is similar to Figure 6-21.

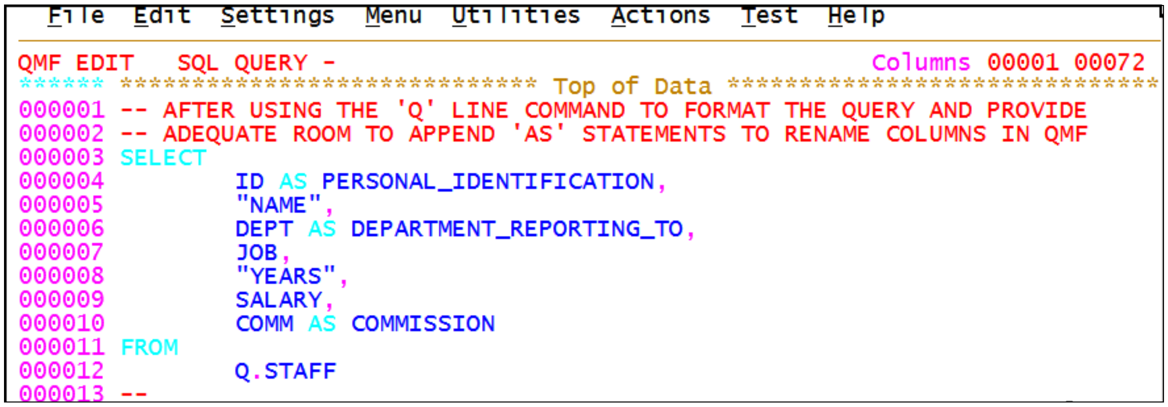
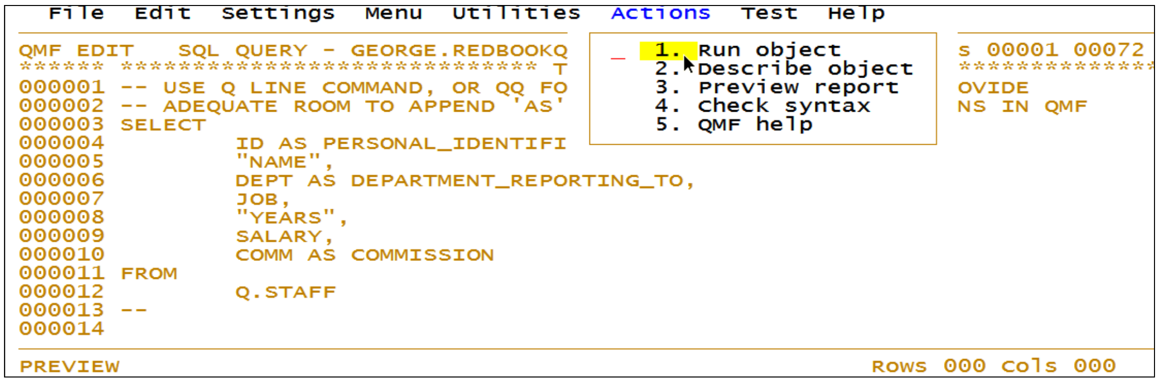

```
  File  Edit  Settings  Menu  Utilities  Actions  Test  Help
QMF EDIT   SQL QUERY -                                    Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 -- AFTER USING THE 'Q' LINE COMMAND TO FORMAT THE QUERY AND PROVIDE
000002 -- ADEQUATE ROOM TO APPEND 'AS' STATEMENTS TO RENAME COLUMNS IN QMF
000003 SELECT
000004         ID AS PERSONAL_IDENTIFICATION,
000005         "NAME",
000006         DEPT AS DEPARTMENT_REPORTING_TO,
000007         JOB,
000008         "YEARS",
000009         SALARY,
000010         COMM AS COMMISSION
000011 FROM
000012         Q.STAFF
000013 --
```

Figure 6-21 Enhancing our query

In our modified query, we added a couple of comment lines and renamed a few columns with the AS clause. It is far more readable and gives a better sense of how the columns are separated with a comma.

Important: No comma is used after the last column in the list.

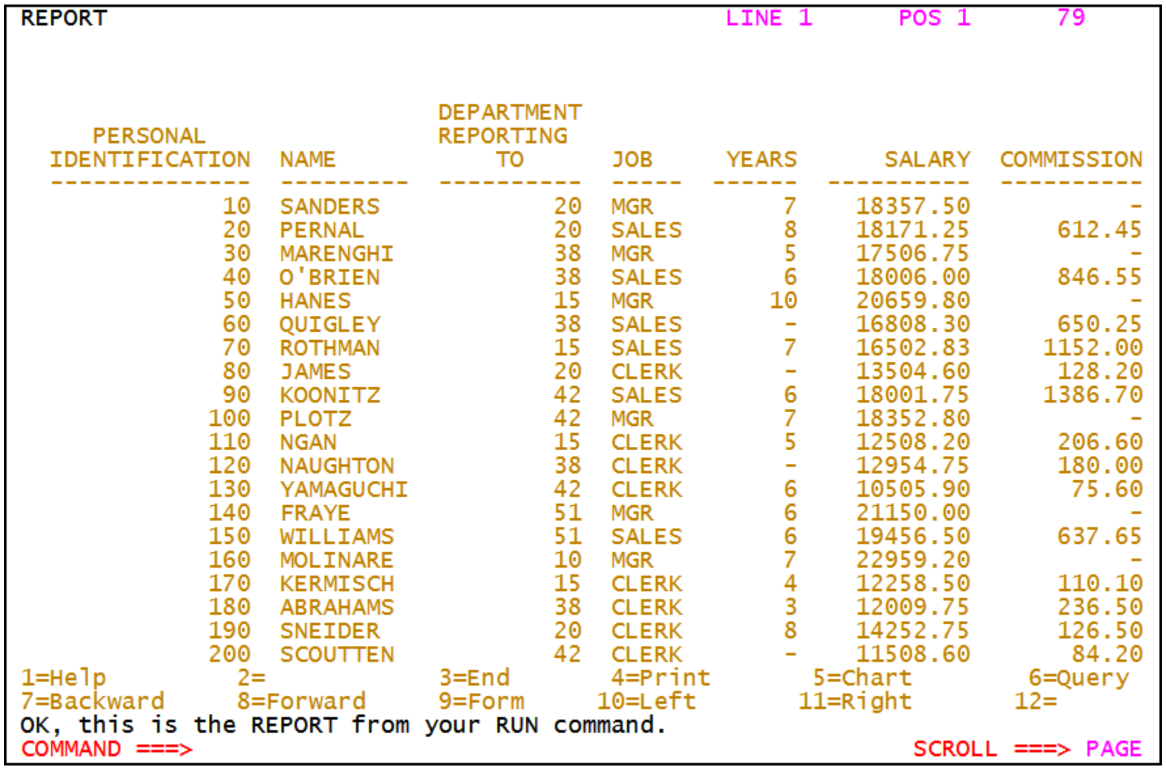
So, how do we run this query? If we use the standard ISPF editor, we return to QMF (PF3) and run it from QMF. Instead, we look at the synergy that is available with QMF Enhanced Editor and QMF instead. Go up to the action bars at the top of the panel and click **Actions**, as shown in Figure 6-22.



```
  File  Edit  Settings  Menu  Utilities  Actions  Test  Help
QMF EDIT   SQL QUERY - GEORGE.REDBOOKQ  _  1. Run object       S 00001 00072
****** ***************************** T     2. Describe object  *************
000001 -- USE Q LINE COMMAND, OR QQ FO     3. Preview report   OVIDE
000002 -- ADEQUATE ROOM TO APPEND 'AS'     4. Check syntax     NS IN QMF
000003 SELECT                              5. QMF help
000004          ID AS PERSONAL_IDENTIFI
000005          "NAME",
000006          DEPT AS DEPARTMENT_REPORTING_TO,
000007          JOB,
000008          "YEARS",
000009          SALARY,
000010          COMM AS COMMISSION
000011 FROM
000012          Q.STAFF
000013 --
000014
PREVIEW                                                        Rows 000 Cols 000
```

Figure 6-22   Running the query from the action bar

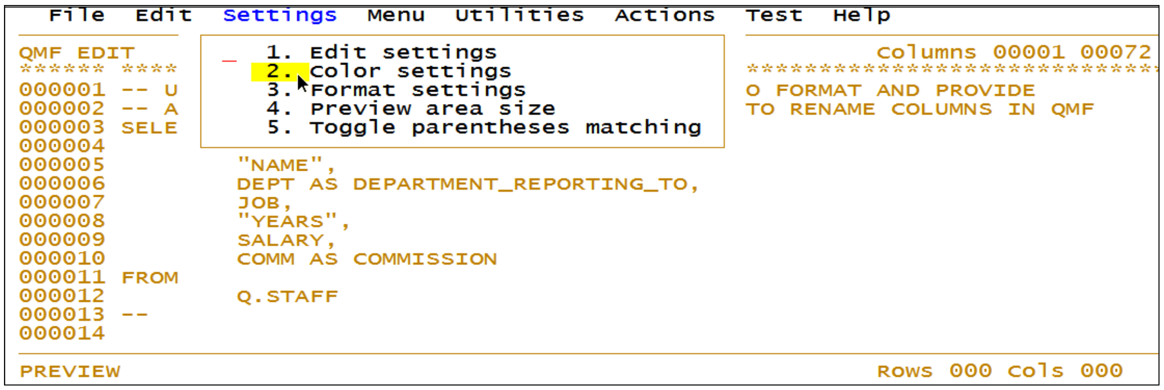
We can run the query directly by clicking **Run object** from the QMF Enhanced Editor without ending our edit session, as shown in Figure 6-23. Certain QMF options are not available when you use the QMF Enhanced Editor, such as pressing PF9 from query results to invoke QMF forms.



```
REPORT                                                LINE 1     POS 1      79



                          DEPARTMENT
   PERSONAL               REPORTING
IDENTIFICATION  NAME          TO       JOB     YEARS     SALARY  COMMISSION
--------------  ---------  ----------  -----  ------  ---------  ----------
            10  SANDERS            20  MGR         7   18357.50           -
            20  PERNAL             20  SALES       8   18171.25      612.45
            30  MARENGHI           38  MGR         5   17506.75           -
            40  O'BRIEN            38  SALES       6   18006.00      846.55
            50  HANES              15  MGR        10   20659.80           -
            60  QUIGLEY            38  SALES       -   16808.30      650.25
            70  ROTHMAN            15  SALES       7   16502.83     1152.00
            80  JAMES              20  CLERK       -   13504.60      128.20
            90  KOONITZ            42  SALES       6   18001.75     1386.70
           100  PLOTZ              42  MGR         7   18352.80           -
           110  NGAN               15  CLERK       5   12508.20      206.60
           120  NAUGHTON           38  CLERK       -   12954.75      180.00
           130  YAMAGUCHI          42  CLERK       6   10505.90       75.60
           140  FRAYE              51  MGR         6   21150.00           -
           150  WILLIAMS           51  SALES       6   19456.50      637.65
           160  MOLINARE           10  MGR         7   22959.20           -
           170  KERMISCH           15  CLERK       4   12258.50      110.10
           180  ABRAHAMS           38  CLERK       3   12009.75      236.50
           190  SNEIDER            20  CLERK       8   14252.75      126.50
           200  SCOUTTEN           42  CLERK       -   11508.60       84.20
1=Help        2=            3=End       4=Print       5=Chart       6=Query
7=Backward    8=Forward     9=Form     10=Left       11=Right      12=
OK, this is the REPORT from your RUN command.
COMMAND ===>                                                  SCROLL ===> PAGE
```

Figure 6-23   Viewing query results from within the QMF Enhanced Editor

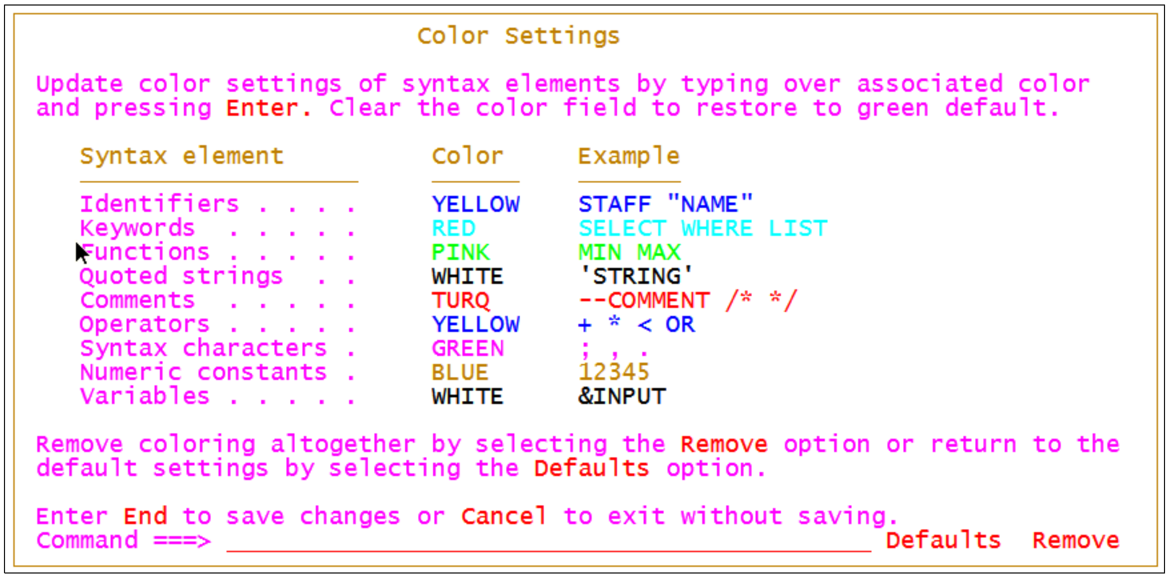
Different colors highlight various sections of the query examples, such as SELECT and AS. You can customize the color settings for different QMF statements and messages by clicking **Settings**, as shown in Figure 6-24.



```
   File   Edit   Settings   Menu   Utilities   Actions   Test   Help
 ---------------------------------------------------------------------
 QMF EDIT   | _    1. Edit settings                  |   Columns 00001 00072
 ****** ****|      2. Color settings                 | *************************
 000001 -- U|      3. Format settings                | O FORMAT AND PROVIDE
 000002 -- A|      4. Preview area size              | TO RENAME COLUMNS IN QMF
 000003 SELE|      5. Toggle parentheses matching    |
 000004      ------------------------------------------
 000005          "NAME",
 000006          DEPT AS DEPARTMENT_REPORTING_TO,
 000007          JOB,
 000008          "YEARS",
 000009          SALARY,
 000010          COMM AS COMMISSION
 000011 FROM
 000012          Q.STAFF
 000013 --
 000014
 ---------------------------------------------------------------------
 PREVIEW                                           Rows 000 Cols 000
```

Figure 6-24 The settings action bar option

From the Settings menu, select **Color Settings**. You are presented with a set of colors to choose from. You can assign them to particular areas of a query, as shown in Figure 6-25.



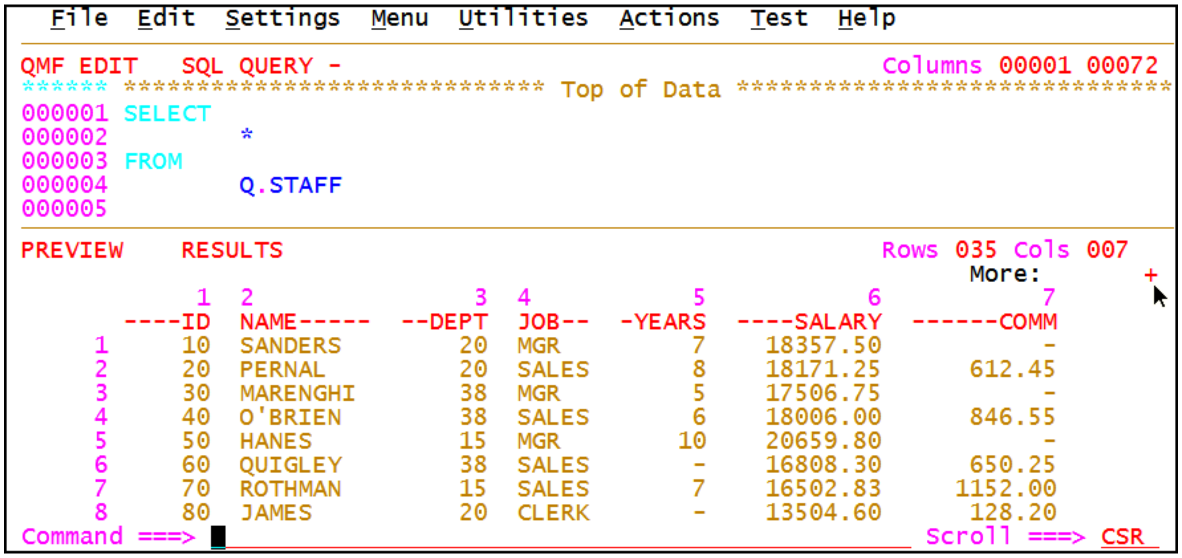
```
                         Color Settings

 Update color settings of syntax elements by typing over associated color
 and pressing Enter. Clear the color field to restore to green default.

    Syntax element            Color      Example
    ____________________      _____      _______

    Identifiers . . . .       YELLOW     STAFF "NAME"
    Keywords  . . . . .       RED        SELECT WHERE LIST
    Functions . . . . .       PINK       MIN MAX
    Quoted strings  . .       WHITE      'STRING'
    Comments  . . . . .       TURQ       --COMMENT /* */
    Operators . . . . .       YELLOW     + * < OR
    Syntax characters .       GREEN      ; , .
    Numeric constants .       BLUE       12345
    Variables . . . . .       WHITE      &INPUT

 Remove coloring altogether by selecting the Remove option or return to the
 default settings by selecting the Defaults option.

 Enter End to save changes or Cancel to exit without saving.
 Command ===> ____________________________________________ Defaults   Remove
```

Figure 6-25 Color option settings

Use of color can greatly enhance your work with the editor. So many new features and functions are available with the QMF Enhanced Editor, but we cannot cover all of them in this book.

We want to show you one more option. Figure 6-24 shows an option to specify preview area size. PREVIEW is an option in the new QMF Enhanced Editor to provide a small subset of the rows from a query so that the user can browse the results to see whether the results are what they really want. It executes a `FETCH` command and specifies a maximum of 100 rows to return.

We start with a new query and go against the Q.STAFF table one more time, but we do *not* run the query. Instead, we click **Actions** → **Preview report**. A subset of data is returned, as shown in Figure 6-26.

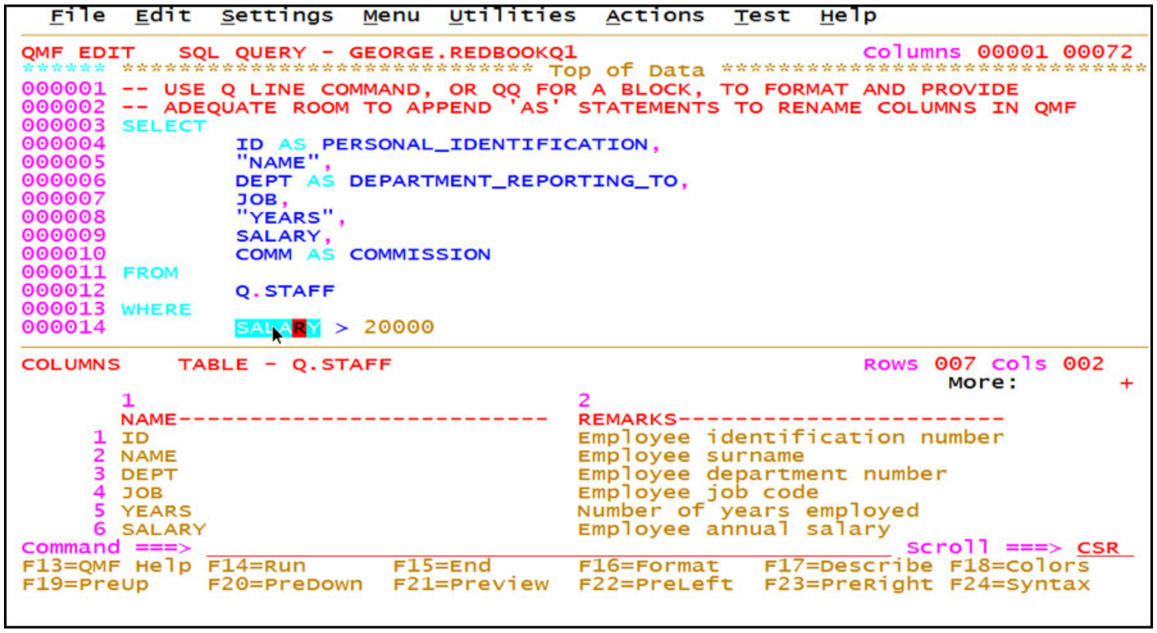


```
  File  Edit  Settings  Menu  Utilities  Actions  Test  Help

QMF EDIT    SQL QUERY -                                    Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 SELECT
000002        *
000003 FROM
000004        Q.STAFF
000005

PREVIEW    RESULTS                                         Rows 035 Cols 007
                                                                 More:      +
           1  2                3  4          5           6           7
       ----ID  NAME-----   --DEPT  JOB--  -YEARS  ----SALARY  ------COMM
     1     10  SANDERS         20  MGR         7    18357.50           -
     2     20  PERNAL          20  SALES       8    18171.25      612.45
     3     30  MARENGHI        38  MGR         5    17506.75           -
     4     40  O'BRIEN         38  SALES       6    18006.00      846.55
     5     50  HANES           15  MGR        10    20659.80           -
     6     60  QUIGLEY         38  SALES       -    16808.30      650.25
     7     70  ROTHMAN         15  SALES       7    16502.83     1152.00
     8     80  JAMES           20  CLERK       -    13504.60      128.20
Command ===>                                                  Scroll ===> CSR
```

Figure 6-26 The PREVIEW panel in the QMF Enhanced Editor

We can browse the data to see whether the data is what we intended to query or merely to get a sense for the values and format that are returned. The size of the preview area can be customized through the Settings action bar menu, with a default that fits the size of your terminal. A size of 0 turns off the preview area so that all lines are available for editing.

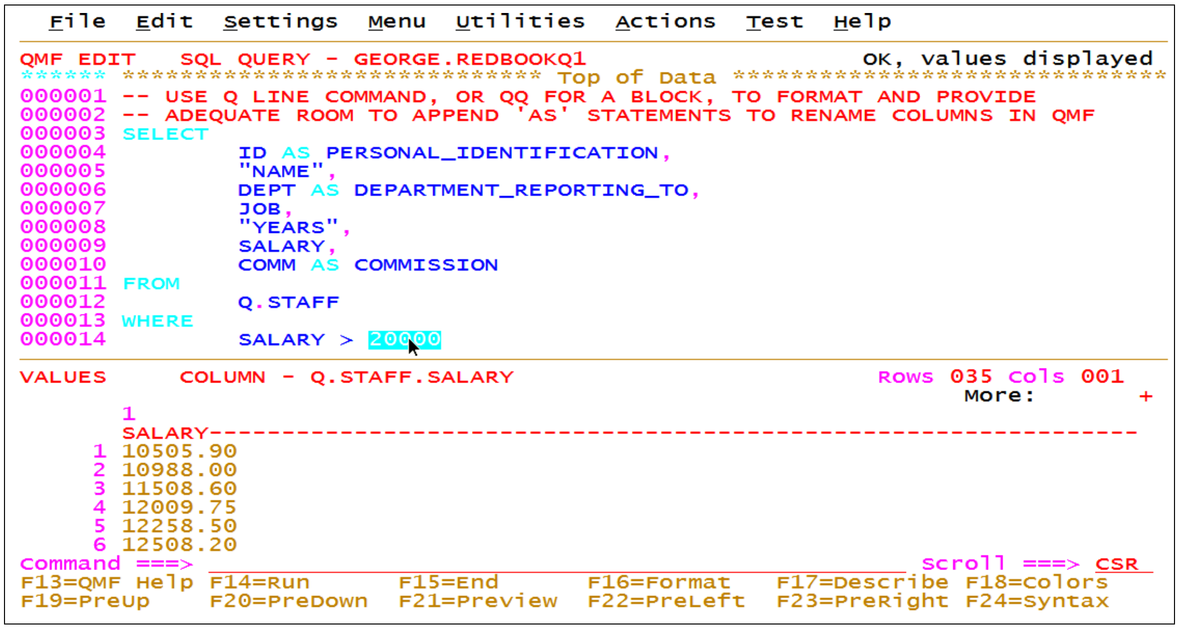
One exciting feature in the QMF Enhanced Editor is the capability to prompt for columns or values of columns. In Figure 6-27, we added a WHERE clause to our query to illustrate this function. If we place the cursor under the column, SALARY is highlighted. If we either press Enter or use the mouse (double-click, if enabled), we see that SALARY is highlighted in reverse video. A list of columns that are available to add to our WHERE clause shows in the preview area underneath. Select or double-click one of the listed columns to replace SALARY in the WHERE clause.



```
  File  Edit  Settings  Menu  Utilities  Actions  Test  Help
QMF EDIT    SQL QUERY - GEORGE.REDBOOKQ1                     Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 -- USE Q LINE COMMAND, OR QQ FOR A BLOCK, TO FORMAT AND PROVIDE
000002 -- ADEQUATE ROOM TO APPEND 'AS' STATEMENTS TO RENAME COLUMNS IN QMF
000003 SELECT
000004         ID AS PERSONAL_IDENTIFICATION,
000005         "NAME",
000006         DEPT AS DEPARTMENT_REPORTING_TO,
000007         JOB,
000008         "YEARS",
000009         SALARY,
000010         COMM AS COMMISSION
000011 FROM
000012         Q.STAFF
000013 WHERE
000014         SALARY > 20000
COLUMNS     TABLE - Q.STAFF                                  Rows 007 Cols 002
                                                                   More:      +
       1                                2
       NAME--------------------------   REMARKS------------------------
     1 ID                               Employee identification number
     2 NAME                             Employee surname
     3 DEPT                             Employee department number
     4 JOB                              Employee job code
     5 YEARS                            Number of years employed
     6 SALARY                           Employee annual salary
Command ===>                                               Scroll ===> CSR
F13=QMF Help F14=Run      F15=End      F16=Format   F17=Describe F18=Colors
F19=PreUp    F20=PreDown  F21=Preview  F22=PreLeft  F23=PreRight F24=Syntax
```

Figure 6-27 Prompts for columns in a WHERE clause in the QMF Enhanced Editor

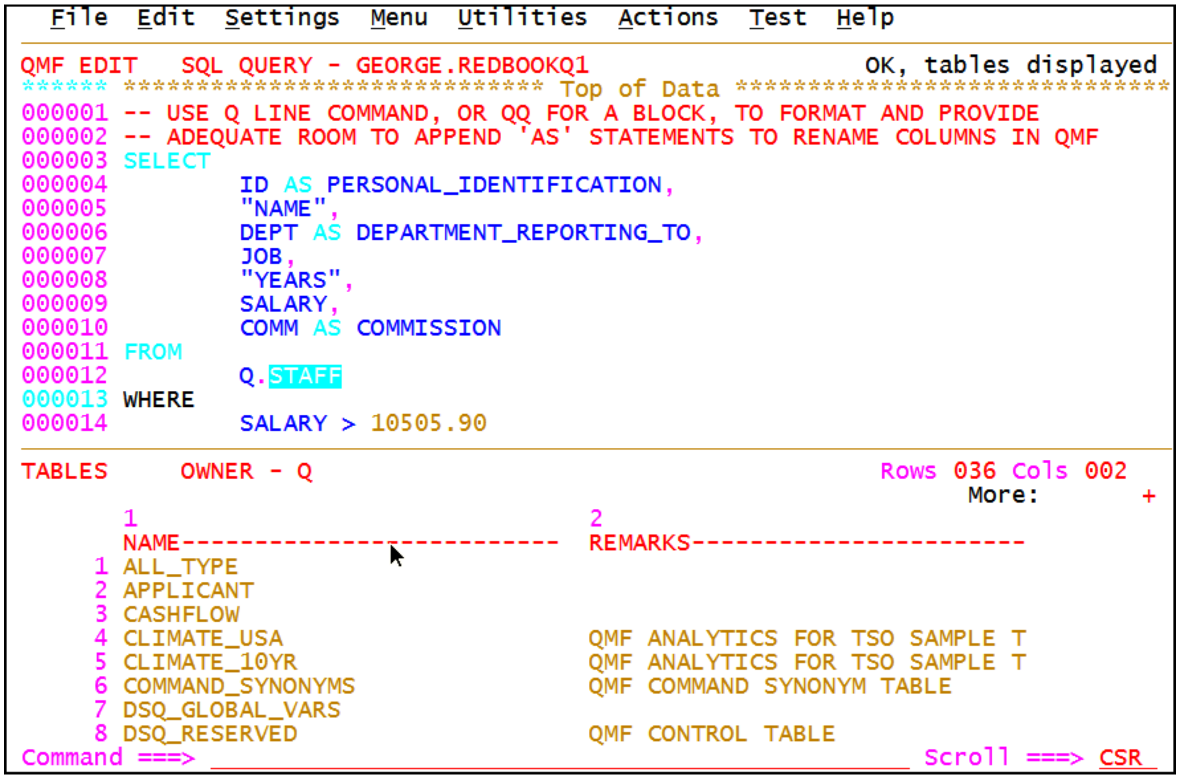
Also, QMF can display a list of values for a column with a mouse double-click (or position the cursor and press Enter), as shown in Figure 6-28. A range of values is returned in the preview area so that we get a sense of how large or small the values that we want to use to delimit our search are. You can double-click a value in the preview area to replace the value that is in reverse video in your query.



```
  File  Edit  Settings  Menu  Utilities  Actions  Test  Help
QMF EDIT   SQL QUERY - GEORGE.REDBOOKQ1                    OK, values displayed
****** ***************************** Top of Data ******************************
000001 -- USE Q LINE COMMAND, OR QQ FOR A BLOCK, TO FORMAT AND PROVIDE
000002 -- ADEQUATE ROOM TO APPEND 'AS' STATEMENTS TO RENAME COLUMNS IN QMF
000003 SELECT
000004         ID AS PERSONAL_IDENTIFICATION,
000005         "NAME",
000006         DEPT AS DEPARTMENT_REPORTING_TO,
000007         JOB,
000008         "YEARS",
000009         SALARY,
000010         COMM AS COMMISSION
000011 FROM
000012         Q.STAFF
000013 WHERE
000014         SALARY > 20000
VALUES     COLUMN - Q.STAFF.SALARY                             Rows 035 Cols 001
                                                                    More:     +
       1
       SALARY----------------------------------------------------------------
     1 10505.90
     2 10988.00
     3 11508.60
     4 12009.75
     5 12258.50
     6 12508.20
Command ===> ________________________________________________ Scroll ===> CSR
F13=QMF Help F14=Run      F15=End      F16=Format   F17=Describe F18=Colors
F19=PreUp    F20=PreDown  F21=Preview  F22=PreLeft  F23=PreRight F24=Syntax
```

Figure 6-28 Prompts for value ranges in the WHERE clause

If you click the mouse on a table name, it lists tables from the owner. In our example (Figure 6-29), we use the owner that is named Q. Therefore, when we place the cursor under the table name or use the mouse, we see a list of tables that are owned by Q.

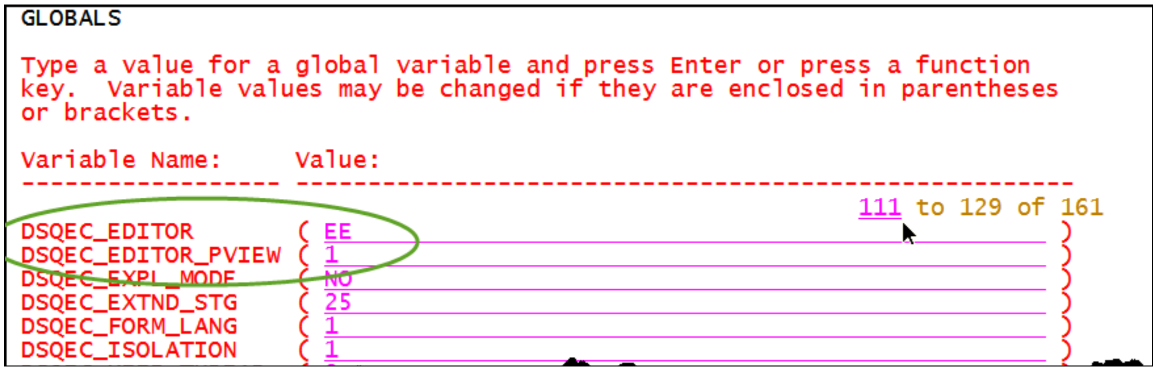


```
  File  Edit  Settings  Menu  Utilities  Actions  Test  Help
QMF EDIT   SQL QUERY - GEORGE.REDBOOKQ1                   OK, tables displayed
****** **************************** Top of Data ******************************
000001 -- USE Q LINE COMMAND, OR QQ FOR A BLOCK, TO FORMAT AND PROVIDE
000002 -- ADEQUATE ROOM TO APPEND 'AS' STATEMENTS TO RENAME COLUMNS IN QMF
000003 SELECT
000004         ID AS PERSONAL_IDENTIFICATION,
000005         "NAME",
000006         DEPT AS DEPARTMENT_REPORTING_TO,
000007         JOB,
000008         "YEARS",
000009         SALARY,
000010         COMM AS COMMISSION
000011 FROM
000012         Q.STAFF
000013 WHERE
000014         SALARY > 10505.90
TABLES     OWNER - Q                                       Rows 036 Cols 002
                                                                   More:     +
       1                                2
       NAME-----------------------------  REMARKS-----------------------
     1 ALL_TYPE
     2 APPLICANT
     3 CASHFLOW
     4 CLIMATE_USA                        QMF ANALYTICS FOR TSO SAMPLE T
     5 CLIMATE_10YR                       QMF ANALYTICS FOR TSO SAMPLE T
     6 COMMAND_SYNONYMS                   QMF COMMAND SYNONYM TABLE
     7 DSQ_GLOBAL_VARS
     8 DSQ_RESERVED                       QMF CONTROL TABLE
Command ===>                                                  Scroll ===> CSR
```

Figure 6-29 Query Assist for a table that shows other tables that are owned by Q

We end with a description of global variables. We recommend that you use the QMF Enhanced Editor for future QMF query and procedure work. If you are not an ISPF user, ISPF takes a small amount of getting used to, but it is worth it.

QMF has a growing pool of global variables that are used to set values and options for common use or for an individual's use. For the QMF Enhanced Editor, focus on two global variables, as shown in Figure 6-30. To see them, use the **SHOW GLOBAL** command in QMF and scroll until you see the variables.



```
GLOBALS

Type a value for a global variable and press Enter or press a function
key.  Variable values may be changed if they are enclosed in parentheses
or brackets.

Variable Name:     Value:
------------------ ------------------------------------------------------
                                                          111 to 129 of 161
DSQEC_EDITOR       ( EE                                                  )
DSQEC_EDITOR_PVIEW ( 1                                                   )
DSQEC_EXPL_MODE    ( NO                                                  )
DSQEC_EXTND_STG    ( 25                                                  )
DSQEC_FORM_LANG    ( 1                                                   )
DSQEC_ISOLATION    ( 1                                                   )
```

Figure 6-30 Global variables for the QMF Enhanced Editor settings

Set DSQEC_EDITOR to `EE` to invoke the QMF Enhanced Editor for queries and procedures. Then, the user enters `EDIT [query/proc]`, and the QMF Enhanced Editor is invoked.

DSQEC_EDITOR_PVIEW can be set to `0` to prevent the use of the preview area. A QMF administrator can use this global variable to prevent users from using “Preview area size” by setting the variable to `0` and stopping users from changing this variable.

Note: If you set the DSQEC_EDITOR_PVIEW global variable to `0` to disable the preview, when the QMF Enhanced Editor is used next, it sets the user’s preview area size setting to `0`. Then, if the DSQEC_EDITOR_PVIEW is set to `1`, the user must change the preview area size setting from 0 before the preview area is shown.

Summary

A conscious effort was made to enhance all elements of the QMF family, including the TSO interface, so that existing and new users find value in its use and application as an enterprise business intelligence and analytics tool.

The TSO enhancements evolved from a series of user requests and satisfy many outstanding requirements. If you are not a user with extended analytics and graphics needs, the QMF Enhanced Editor can make you far more productive. If you need more analysis within QMF and a need to present your results in a more graphical fashion, use the query analytics capability.

We hope you are interested enough and excited enough to explore the other reference materials about both new features that we referenced. We barely introduced the functions that are available in both new options.

A

Additional material

This book refers to additional material that can be downloaded from the internet as described in the following sections.

Locating the web material

The web material that is associated with this book is available in softcopy on the internet from the IBM Redbooks web server. Point your web browser at this website:

`ftp://www.redbooks.ibm.com/redbooks/SG248370`

Alternatively, you can go to the IBM Redbooks website:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG248370.

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material compressed file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM support and publications

For more information, see the following website:

- DB2 Query Management Facility Library:

  `http://www.ibm.com/support/docview.wss?uid=swg27048346`

For more information, see the following documents:

- *Getting Started with DB2 QMF Vision*, GC27-8805-01
- *DB2 QMF Data Service Getting Started Guide*, GC27-8806-01
- *DB2 QMF Data Service Customization Guide*, GC27-8807-01
- *DB2 QMF Data Service Solutions Guide*, GC27-8808-01
- *DB2 QMF Data Service Studio User's Guide*, SC27-8816-01
- *Installing and Managing DB2 QMF for Workstation and DB2 QMF for WebSphere,* GC27-6742-02

Help from IBM

IBM Support and downloads:

ibm.com/support

IBM Global Services:

ibm.com/services

Redbooks In-Place Analytics with Live Enterprise Data with IBM QMF

(0.2”spine)
0.17”<->0.473”
90<->249 pages

SG24-8370-00

ISBN 0738442046

Printed in U.S.A.