SQL/Query

Gathering Requirements
Finding Data
Options and Tools
Joins
SQL
Case studies

Also includes:
Troubleshooting guide
Query and Excel,
SQL Tricks
And more…

Query Intermediate
Created by Accounting Training.

Author Matthew Morris

Some references in the first unit were used from prior training created by:
Chad Roberts

Visit us on the intranet at:

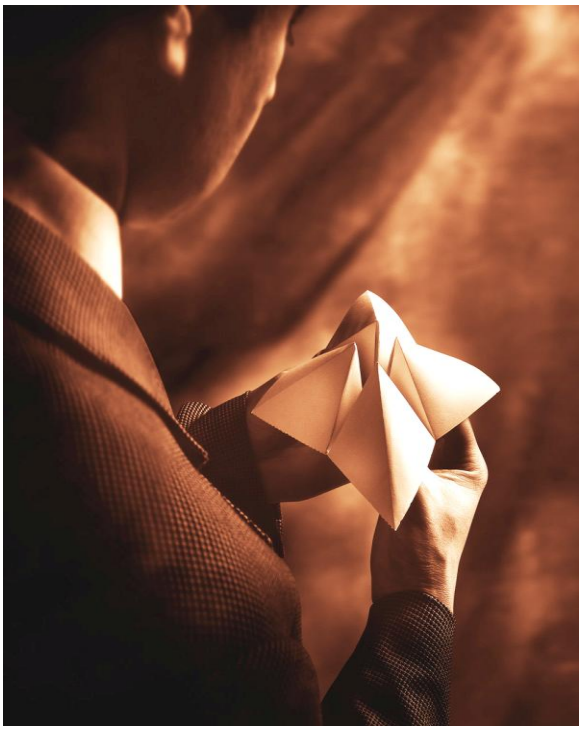
Accounting Training Department
http://accounting/Train/trainingindex.htm

Sharepoint
http://sharepoint/Accounting/Query

Table of Contents

UNIT I: ENVIRONMENT

Showcase Products
EIS400 Environment
Libraries, Tables, Members
Gathering Requirements
Research and Development
File types



INTRODUCTION

Query, simply put, means question. When building or running queries you are essentially asking business questions and getting business answers. Keeping this in mind, who knows your business better than you? A techie person? That Query guru down the hall? Or you? Nobody knows your business better than you; *you* are the most qualified person to build the queries or questions you have about your business. You will understand your question better than anyone else, and when the answer comes back, you will be the one to determine if that answer is correct or has meaning.

Query can help us answer questions that keep business executives awake at night, answer questions that help drive our business, create reports that do not currently exist, and automate cut-and-paste reports.

The key is not just navigating the application, but understanding the concepts behind building a successful query, and then asking good questions.

SHOWCASE PRODUCTS

Analyzer

Analyzer and Essbase are Showcase databases with pre-built views that can be arranged to show greater or lesser detail. The views can also be manipulated to filter, add, or sort data in many formats.
By using Analyzer, you can:

- Rank, filter, and sort data according to a number of variables or "dimensions"
- Present information in a variety of graphic formats
- Explore both summarized and detailed data in a variety of ways
- Highlight exceptions or variances that point either to problems or opportunities
- Print Analyzer reports, or include Analyzer reports in documents, presentations, or web pages

Excel Essbase Add-in

The Excel Essbase add-in is essentially Analyzer in Excel. Creating formulas and specialized reports using Analyzer data as a source within Excel allows for more options in reporting.

Query

Query allows you to query or ask questions of the database to perform research and help support business needs. Query does this by using a language called Structured Query Language or SQL. The Query application has simplified creating queries by providing a wizard that builds the SQL or Select statements for you. Query can run both scheduled and ad-hoc queries.

Creating even the most complicated reports is a simple task with Query. Here are some of the functions it can perform beyond simply creating a query:

- The Query Guide, a guided user interface that automatically steps you through the process for creating a query statement
- The Navigator, a guided user interface that automatically steps you through the process for creating an Essbase query
- A built-in data viewer
- Viewer options and query commands that can be quickly accessed with a right-click of the mouse
- Data viewer options that let you use other application programs, such as your spreadsheet or word processor, to view your data
- Query prompting capabilities that request selection criteria from the user.
- Totaling and subtotaling capabilities available for relational queries through column break groups
- Query scheduling capabilities such as automatically publishing query output to the ShowCase Server or to another PC on your network at specific intervals
- Structured Query Language (SQL) Select statement support for relational queries, including the ability to edit the statement directly in Query

- Import Query option for relational queries that allows you to import Query/400 and Query Manager SELECT statements from server data sources using ShowCase ODBC (Open Database Connectivity)
- Batch processing for relational queries if your data source is a server and uses ShowCase ODBC
- Online help and reference information, including context-sensitive help in the Query window

Report Writer

Report Writer allows users to format reports. This is a great tool for queries that are updated routinely. Users can:

- Create a variety of report types, including those with cross tabs
- Save time by using report templates
- Use summary fields to create subtotals, averages, counts, etc
- Format fonts, colors, styles, column and row sizes, and alignment
- Introduce graphics and images into reports
- Choose from a variety of printing options, including the ability to create PDF files without the need to purchase additional software

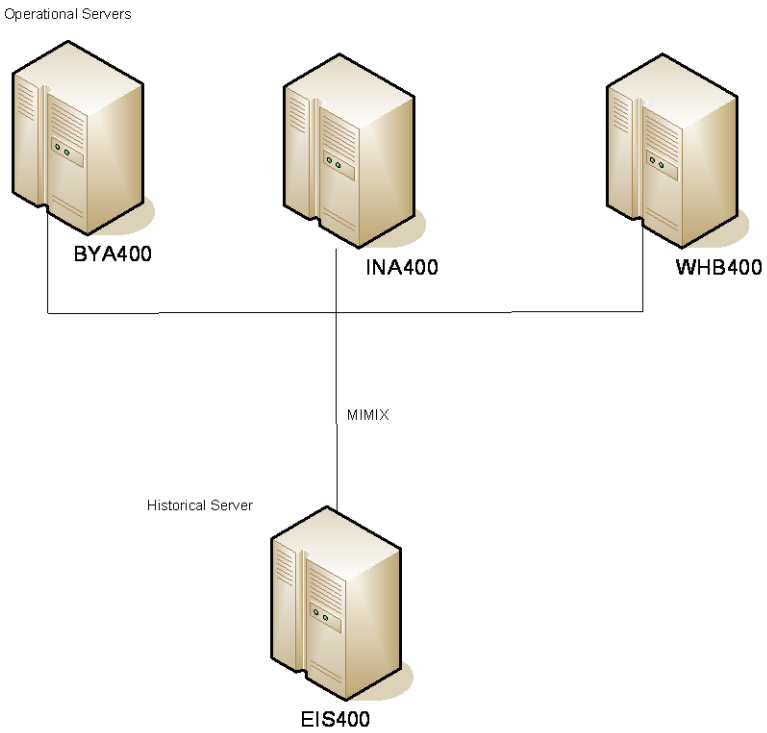
Showcase Query Add-in for Excel

The Showcase Query add-in for Excel allows you to embed multiple queries within Excel, and reference data from those queries in different cells. The power of this tool is the ability to create a template for a report that you can easily update with one click.

EIS400

EIS400

Your computer has the ability to connect to the iSeries400 (AS/400) through iSeries reports (AS/400 reports, green screen reports) or PC applications like Query. The iSeries400 is actually a series of servers that are not all the same. The INA400, BYA400, and WHB400, each support different systems for different areas of the company. Most iSeries reports reside on the INA, BYA and WHB servers. Query does not access those servers; it only accesses the EIS400.



Operational Vs. Historical

The INA400, WHB400, and BYA400 servers are all operational servers, and the EIS400 (Executive Information System) is a historical data

server. Operational data is ready for use after it has been submitted, whereas historical data has to be Mimmixed, or copied, from the operational servers to the EIS400.

Not all data from the operational servers is Mimmixed. This is because the EIS400 is a historical data server, used for analysis of the business. Analysis on an EIS400 is meant to be forecasting and exploratory. This does not mean you cannot use the EIS400 for replicating operational reporting. Ideally however, the EIS400 should not be considered as an operational server. Most data is available within 24 hours, and some data is updated throughout the day. The Mimmix process and other extract jobs affect the timing of data.

The data selection that is Mimixed to the EIS400 is typically approved by management, and is based on supporting business specific needs.

RELATIONAL DATABASE, LIBRARIES, TABLES, AND MEMBERS

Relational Database

A relational database is a body of related information stored on a computer system that appears to the user as a **Library** or **Collection** of tables. This body of information is essentially the same as that found in traditional paper record keeping tables. Each of these tables can be related together to represent the whole of a business. They are related by linking common denominators between the tables known as keys.

EXERCISE 1: LINKING RELATIONAL TABLES

Review the following spreadsheets which represent three different tables; Contacts, Gift ideas, and Party. Once you have become familiar with the tables and the information in the tables answer the following question. As you tally the answer think about how you are linking the tables together, also consider how this simple example of manually counting may be similar to work you currently do either via using multiple iSeries400 options or through spreadsheets and highlighters.

Contacts					Gift Ideas				Party			
AFFILIATION	NAME	OCCUPATION	ADDRESS	PHONE NUMBER	NAME	GIFT IDEAS	GIFTS RECEIVED	GIFTS GIVEN	NAME	Holidays	Birthday	Anniversaries
Family	Darryl	Accountant	California	(111) 222-3333	Ann	Costco card	DVDS	Wine	William	July 4th		
Family	Carol	Pharmacist	New York	(222) 333-4444	Carol	Gift card	Gift card	Costco Card	Wayne	Birthday	5/10/1972	11/13/1999
Business	Ann	Truck Driver	Texas	(555) 666-7777	Cynthia	Wine			Tony	Birthday		
Family	Jim	Lawyer	California	(777) 888-9999	Darryl	Wine	Costco Card	Gift Card	Terry	Halloween	7/24/1976	1/1/2001
Friend	Tony	Student	Florida	(123) 456-7890	Jason		Concert ticket	Board games	Susan	Cinco de Mayo		
Business	Mike	Mechanic	Iowa	(999) 888-7777	Jim	DVD's	Gift card	Costco Card	Sarah	Birthday	6/1/1977	
Business	Sarah	Construction	Washington	(654) 333-2222	Juan	Costco card			Pam	Halloween		
Business	Juan	Dentist	Wyoming	(122) 222-3333	Julie		Painting	Gift Card	Mike	Birthday		7/4/1998
Friend	Julie	Student	Arizona	(322) 223-1234	Kevin	Gift card		Sweater	Michelle	New years	2/23/1980	
Friend	Susan	Design	Washington		Michelle		Guitar lessons		Kevin	April Fools	4/1/1982	
Family	Terry	Military	California	(798) 987-7777	Mike	Gift card	Gift card	Costco Card	Julie	Birthday		9/23/2010
Business	Pam	Sales	Nevada	(222) 111-3214	Pam	Books			Juan	July 4th		
Friend	Michelle	Student	California	(111) 212-3333	Sarah	Wine			Jim	New years	12/25/1988	2/14/2009
Business	Jason	Bartender	Washington		Susan	Software	Gift card	Costco Card	Jason	Halloween	6/5/1978	
Business	William	Real estate	California	(312) 213-4444	Terry	Gift card	Costco Card	Costco Card	Darryl	New years	10/14/1985	
Family	Wayne	Management	Washington	(444) 123-5555	Tony	Gift card			Cynthia	Birthday		
Business	Cynthia	Real estate	Florida		Wayne		Gift Card	Gift card	Carol	New years	3/26/1979	6/1/2001
Friend	Kevin	Dispatcher	Washington	(654) 456-3333	William	Gift card		Gift card	Ann	Cinco de Mayo	2/3/1971	3/7/1996

Libraries/Collections
A **Collection** is an object designed to hold information. Usually, similar information is grouped by **Collection**. The information stored in a **Collection** is a group of tables and/or members. **Collections** may also be thought of as folders or libraries. Library is a newer and more common term used in place of collection. When reading help or visiting forums or message boards, you will see them both used.

Tables
A **Table** is group of fields used for storing related data. A **Table** may also be referred to as a **File**.

Member
A **Member** is an object inside of a **Table**, designed to hold a more specific level of information, which is broken down into segments. Not every **Table** has **Members**.

Another way to look at a **Library** is that it's like a library that holds a lot of books. The library, however, is specific to a particular subject matter. The **Tables** are the books inside the library and the **Members** are the possible volumes of one particular book.

Like Terms

Library	=	Collection
Table	=	Files
Members	=	Objects in a table
Columns	=	Fields
Rows	=	Records

Verification/iSeries400 Options – Verification of Data
Always verify your data. It is possible that you can have a query with accurate information based on an incorrect table or incorrect field. For example, you may have a report that lets you know when an item has been shipped to the warehouse from the vendor. You have found a table that has a field called Ship Date, so you put it in your query and the report looks good. It may even look good for three months or longer. However, the ship date you chose was the ship date that the depot ships the product from the depot to the warehouse, not the ship date that the vendor shipped it out. This illustrates why it is important to always verify that you have the right data. So how can you do this?

There are a few ways to verify that you have the correct field. One of the best ways is to use iSeries400 options to guide you. Find an option that has pieces of the report you are looking for. Once you have the option open, you can view the tables that build that option. This will lead you in the right direction for getting the correct data.

DELIVERABLE QUERY PHASES

Finding the data you're looking for is not as easy as having a cheat sheet with a few tables on a Post-it, but it's also not as hard as memorizing the entire database. Knowing a few questions to ask to understand the query request better, and being familiar with how to research data can making finding the data you're looking for less intimidating and more accurate.

The deliverable Query phases are the start-to-finish of a query. The phases embody all of the steps from the query request to running the query. The phases include: gathering requirements, research, development, and testing.

Gathering requirements
In order to even build the query, you need the requirements for the query. A request of "show us how we are doing" is not an appropriate query request; it is too vague. Gathering requirements involves asking a series of questions to find out the details you will need to build a successful query.

Research
After gathering the requirements it is time to research. From the requirements you should determine a source for the data. From this source you may need to view one or thirty tables, and become familiar with the fields in those tables.

Development
As you are researching the tables, you will begin finding fields that will work for your query, and experimenting with them.

Testing
As you begin to complete the query, you will want to ensure you have the right results by comparing to the test source.

From start to finish this process can take 30 minutes to 3 months, depending on the complexity of the query and the time resource of the individual creating the query.

GATHERING REQUIREMENTS

The reality most of us work with is that we do not have time to build a query. We do not have time to research and we do not have time to pause a moment to really find out what this person who is requesting a query wants. Taking that moment, though, will save you considerable frustration, miscommunication, and time in the long run.

Know what you are looking for

Although a query request may seem simple on a general level, specifics are needed if you are going to find the data you need to create your report. Here are a few questions you can ask to help you find the correct data.

What is your specific question?

> Query means question: what is your *specific* question? "How is our business?" is not specific.

How do you want to see it? Or, what does your answer look like?

> These will be the headers or columns in your report.

Where are you currently getting this information from?

> Whether it be a presentation or a printed report, there should be a source where the information is coming from. Typically that source will be either one or many iSeries options. Without determining a source, there is no way to verify that the data you retrieve is correct.

How do you want it filtered, or limited?

> This will determine the query's conditions. Examples are: Only Northwest region, start and end dates, a series of locations within Eastern Canada. Whatever the query is, it's a certainty that the person requesting it will want filters.

How would you like it sorted?

> This is not necessarily crucial for building the query, but is a typical request after the query has been built.

Other questions you can ask include, but are not limited to:

> What fields can be added to extend the life of the query build?
> How often would you like the query to run?
> Do you want the query to run automatically or manually?
> How would you like the data saved?
> Will this be linking with Excel reports?
> Will data need to be pulled from a spreadsheet?

Spending a few minutes gathering your requirements will help take the guesswork out of creating the query.

QUERY REQUEST FORM

REQUESTER _____ DATE _____

DEVELOPER

1 What is the specific question?

2 Where does this information currently reside? Is it in the iSeries400, a presentation, the intranet, in a paper report, etc.? What iSeries option has data similar to what you are looking for?

3 How do you want to see the results? What does the answer look like?
List the fields you want to see on a print out or screen. They will appear left to right starting with 1.

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

4 What filters or conditions would you like for this report? (e.g., Region = NW)

5 List fields the report will be sorted by and whether the fields are going to sort in an ascending or descending fashion.

Field	Sort type

RESEARCH AND DEVELOPMENT

After gathering your requirements, you are ready to begin researching to find the tables you need to build your query. In gathering requirements you should have found a source where the information is coming from. It might be an iSeries option, paper report, presentation, or intranet site. All business data should relate to an option on the iSeries400; this is the first place to begin your research.

iSeries400 Options

iSeries400 options are reports that display data from tables on operational servers. It is possible to replicate iSeries options accurately with Query. With some options, it's as easy as loading the same tables into your query. With other options, there are formulas and expressions to consider; these will be covered in a later class.

How to see the tables that build an iSeries400 option:

1. Open an option and fill out the prompts for that option to display results for the desired data.

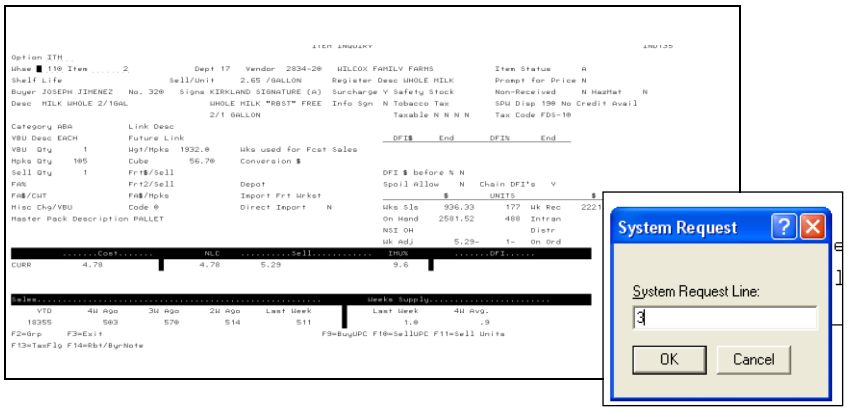
2. Simultaneously press the **Alt** button and the **Print Screen** button on your keyboard. This will either 1) bring up a system line prompt window, or 2) produce a *system line* at the bottom of your iSeries400 option. System lines allow you to see information about the current screen you are viewing.

3. Whether it is the system prompt or a system line, key a "3" and press **Enter** to view the **Display Jobs** screen.

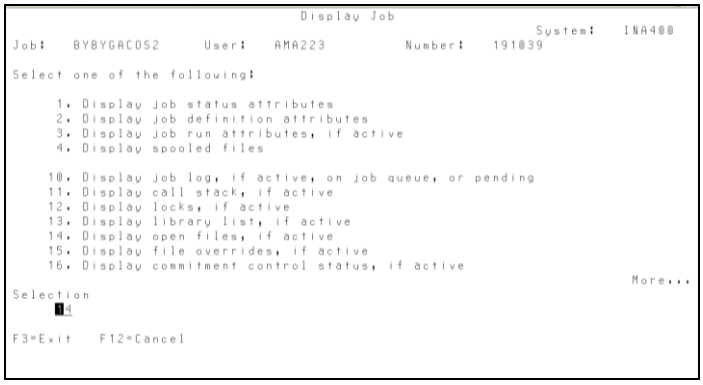
4. Select 14 to **Display Open Files**. You should now see the active files or tables that built the iSeries option.

5. Press the **F11** key on your keyboard to show which files have records.

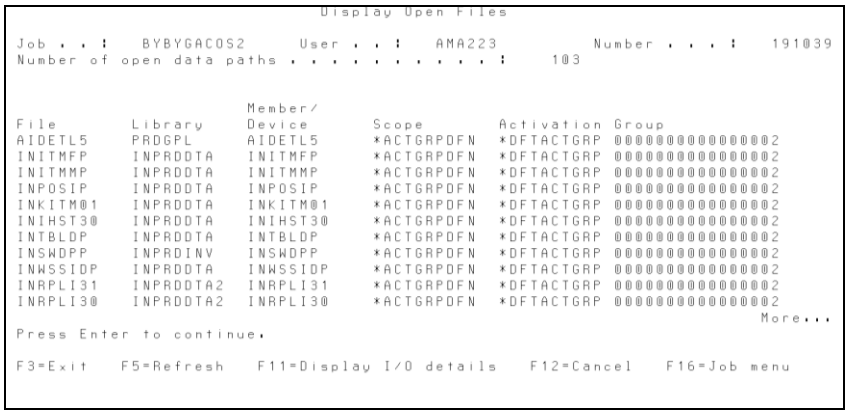
Example:
Using the Option ITM, type "110" in the **Whse** field. In the **Item** field type "2". This will give you the information on whole milk for warehouse 110.



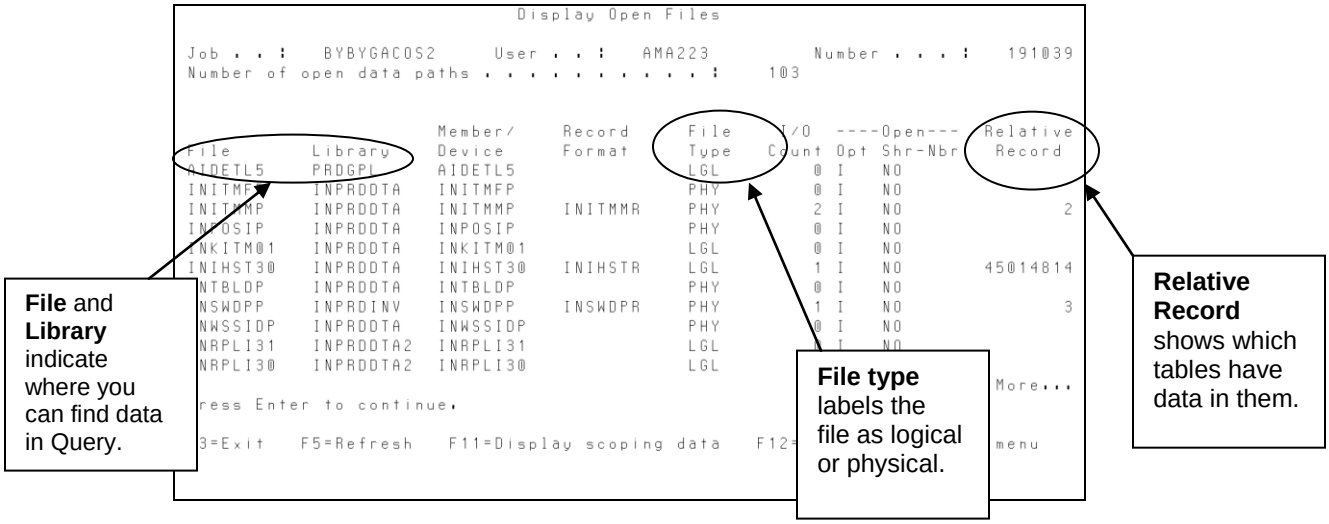
Now let's see what tables build this report. Press **Alt** and **Print Screen** simultaneously to bring up the **System Request** prompt.
Enter "14" in the **Selection** field to display the open files.



Now that **Display Open Files** is visible, press **F11** to unfold the screen to see more information about the tables that build this screen.



Now you're ready to do research to find out what tables you need to build your query.



Let's take a closer look at the **Display Open Files** table. You will notice some header information about the **Job**, **User**, **Number**, and **Number** of open data paths. This information is not important for finding the table that builds your report. Looking down the screen you will see the following columns: **File**, **Library**, **Member/Device**, **Record Format**, **File Type**, **I/O count**, **Opt**, **Shr-Nbr**, and **Relative Record**.

We will be focusing only on the **File**, **Library**, **Relative Record** and **File Type** columns.

The first column to look at is **Relative Record** number. This lets you know how many records are in the corresponding file. As you begin your deduction on what tables to use, start with the tables with the largest relative record numbers first.

File and **Library** are the next columns to view. Notice that INIHST30 in Library INPRDDTA has quite a few relative records. This table may have all the answers you are looking for, however notice in the lower right-hand corner the word "More." Press **Page Down** to see that there are actually quite a few more tables that may have information you are looking for. Knowing the different file types will save you a lot of time in your research.

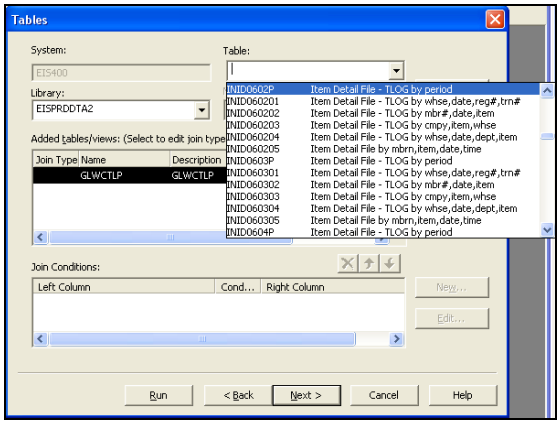
Files and libraries to ignore
Ignore these libraries or tables:

Library
QSYS
QTEMP
Tables beginning with AI or IA

File Types

There are two types of files (tables) used via the Showcase Query Tool: physical and logical.

- A physical file contains the actual data that the queries access.
- The logical file is an alternate sort order for the physical file. In some cases, logical files omit data.



Keyed Columns and Conditions

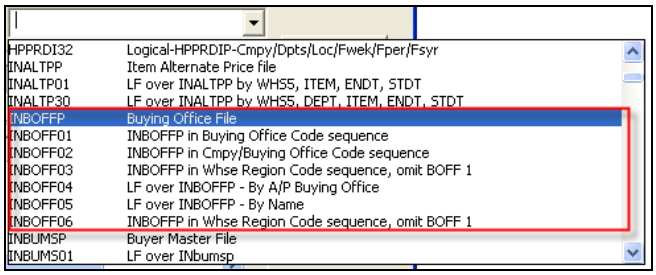
A **keyed column** can also be called an **indexed file**. Both of these terms are just another way to say a **logical file**.

When looking at the file types, you will notice some file types are PHY and some are LGL. Looking at the physical and logical file names, you may also notice that physical files have something in common, and logical files have something in common. Physical files all end with the letter “P” and all of the logical files (logicals) end with a number. The numbers are a sequential order for the number of logical tables.

Logicals are in place so that you can choose a table that is sorted in such a way that your query will run faster. However, with the automated Query optimizer that runs on the EIS400, the need to choose the correct logical is no longer necessary.

So, if each logical is derived from a physical file, then when doing your research – if it appears you would write a file down that is logical – all you have to do is simply exchange the ending numbers with a ‘P’. For example INWCTL32 is now INWCTLP.

After writing down or copying and pasting each physical or logical converted into a physical file, you will look them up in the Query Table Viewer.



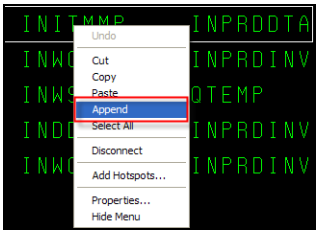
In the example above you can see that INBOFFP is the physical file. Underneath INBOFFP is INBOFF01. To the right of the file name is a description of the sort. In this example, it’s the buying office code sequence. For INBOFF02 the sort is Cmpy, and then Buying office Code Sequence. For INBOFF03 the sort is Whse Region code sequence with an omission of the field BOFF1.

It is important to note that while logical tables can optimize the performance of your query, it is perfectly fine to use the physical file to build your queries. Spending days trying to understand the logical tables and which one is the most efficient to use is not worth the investment of time.

EXERCISE 2: CREATING AN OPTION CHEAT SHEET

1. Open Excel.
2. Open an iSeries400 option.
3. Choose Option IIL Item Inquiry By Location.
4. Enter information for the prompts:
    a. Item = 2
    b. Warehouse = 110
5. Access the Display Open Files screen to view the tables that build this option.

6. Copy the library and table into Excel.
   Using the Append function can save time.



7. Now that you have the information, you will need to do some clean up. In the Display Open Files screen, it shows the file and then the library; this will need to be flipped so that in Column A you have libraries and in column B you have files.
8. Sort the spreadsheet by library.
9. Convert all logical files to physical files.
10. Delete duplicates.
11. Open Query.
12. Open the table browser and look up each table.
13. Type the descriptions of the tables in your spreadsheet.

You have now created a cheat sheet for your query based on Option IIL.
Try this exercise back at your desk with one of your own options.

iSeries400 options access all the tables at different times. You should keep this in mind when creating your cheat sheet.

- Filling out some of the prompts may access certain tables and make them active.
- Using the option again and filling out different prompts may activate other tables.

Troubleshooting
Note that going back in twice can cause problems. When searching behind the scenes, you might find many more open files than seems appropriate for any single option. This is because many of the iSeries options do not close down the open files when you cancel out of the option. The best way to find the data is to sign off your current session, sign back in, and go directly to the option you need to pull information from. You will now only see open files relevant to the current option and the criteria entered in the prompts.

I don't see the table on the EIS400
Why not?
Not all tables are transferred over from the Operational Servers to the EIS400.

What do I do?
First, see if you can find a table that can give you the same information, perhaps from a similar option. If you cannot find an alternate table, you can have a table added to Mimmix on EIS400.

Have the IS group that owns the table (i.e. Accounting, Membership, Buying, etc.) implement the request to add the table to the EIS400.

Common Libraries

If you don't have an iSeries400 option as a starting point, here is a list of commonly used collections to try:

Library	Library Name	Library Description
ACPRDDTA	Accounting Data	Accounts Payable and related data
ACPRDDTA2	General Ledger 2	GL Summary, Financial Statement, GL Warehouse Control
ACPRDDTA3	General Ledger 3	GL Detail, Receiving Item History
EISPRDCPY	Temporary Query Storage	30 days storage limit
EISPRDSAV	EISPRDCPY objects to be saved	Limited use by request through the Data Warehouse Group
EISPRDDTA	Production Query Data	Files that build Analyzer
EISPRDDTA2	Item Detail	Register Transactions
FADBFA	FA 7.1 Production Database	Infinium Fixed Assets Data
INPRDDTA	Inventory Data	Meat license download, Optical, NSI, Vendor Services, Sale Audit, Freight Matrix, Depot
INPRDDTA2	Inventory Data	Rebates Data, Spoils Allowance/ Salvage detail
INPRDINV	Inventory Production Data	Buyer Master, Item Description (ID), Vendor master, Warehouse Control, Vendor Item Master, Warehouse Item Master
MBPRDDTA	Marketing Data	Member Detail
MBPRDDTA2	Marketing Data 2	Shopping history Renewal Rate
OPCAPF	Order Power! CA Files	Vendor & Item Warehouse set-up files
OPF(W#)	Order Power	Mailing list, Order, Pricing by Warehouse
OPPRDDTA	Ordfil Prod Data	Pharmacy, E-comm, iSeries email, Membership

This narrows it down a little, but there are still several tables to go through to try to find specific information. The Accounting Department has put together a MetaData* site. This has a lot of great information on tables and what data is in them.

MetaData site:
http://accounting/Training/Query/EIS400Options/EIS400System.aspx

AS/400 Options:
http://accounting/Training/Query/EIS400Options/AS400Options.aspx

**MetaData is data about data, or a data dictionary.*

Questions for Review

Who is the best person to build your query?

What specific questions should be asked before building a query?

Can queries be embedded into Excel?

Once you have gathered your requirements, what resources do you have to find the data you are looking for?

Does Query connect to the same servers that make the iSeries option?

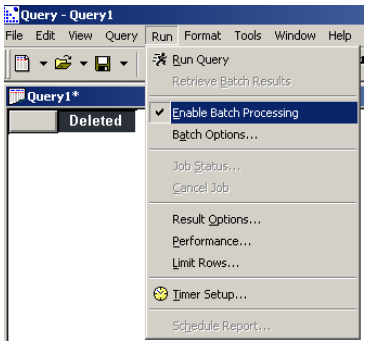
UNIT II:
QUERY WIZARD

Processing Queries
Working with Libraries, Tables, Members, and Fields in Query
Creating a Simple Query
Menus, Options, and Tools

PROCESSING QUERIES

Batch processing means your query is sent to the server and is run as a task. When the job is complete, the data is written to a temporary table for you to view.

Interactive processing is the performance of tasks on a computer system that involves continual exchange of information between the computer and a user; it is the opposite of batch processing.

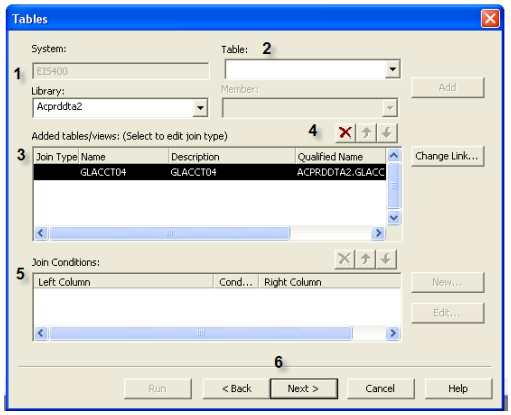


Some users may have interactive processing but most have batch. If a user with interactive processing creates a query and sends it to someone with only batch processing capability, the query *will not run*. An error message will display that says ”The settings do not match the settings your administrator has set for you.” The solution is to have the user who sent you the query enable batch processing, save the query, and then send it again.

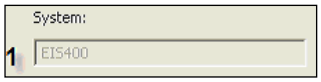
WORKING WITH COLLECTIONS, TABLES, MEMBERS, AND FIELDS IN QUERY

In this section we will discuss how to use **Collections**, **Tables**, and **Fields** in the Query tool to build a report.

Working with Collections and Tables

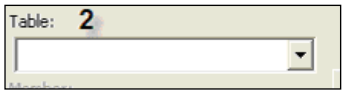


System and Library



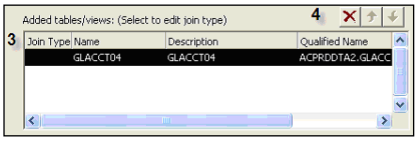
The system tells you what physical device you are connecting to; in this case it is the EIS400. The **Library** has a list box arrow. Clicking on this arrow will allow you to view all of the libraries available to you.

Table



The **Table** list box lists all the tables in the chosen library.

Added Tables

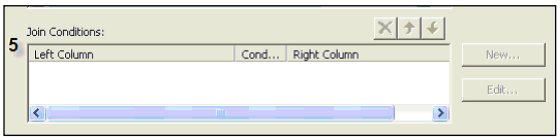


This is where you can view the tables you have selected, along with the table **Description** and **Qualified Name**. A **Qualified Name** displays not only the table but also the collection it belongs to. You can also view and add **Join Types** here.

Delete and Move

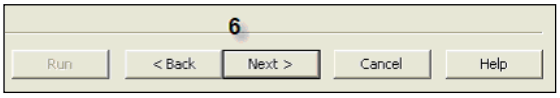
The red **X** button allows you to delete the highlighted added table. The up and down arrows allow you to move the highlighted table.

Joins



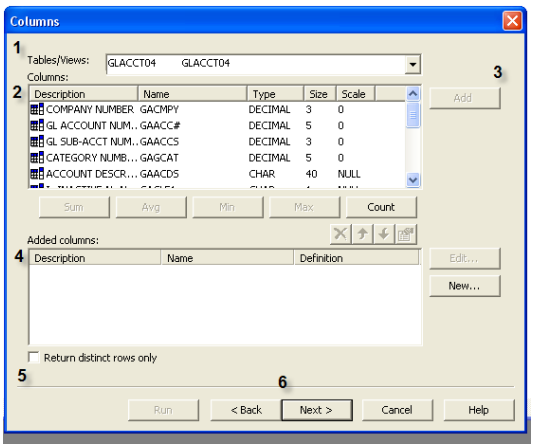
This section displays the various joins created for this query. Using the different edit buttons, you can move the joins around, delete them, or edit them.

Navigating Your Query



The buttons here allow you to go back to the previous screen, on to the next, cancel your progress, or get help with the current window. By clicking on **Next**, you will move to the **Columns** dialog box.

Working with Columns



Columns are designed to hold a more specific level of information inside a table. A column may also be referred to as a field.

Once you have selected the collection and table you want, you will see a **Columns** window. In the **Columns** window you can work with fields from any of the tables you have chosen.

1. **Tables/Views**
   This list box allows you to see what tables you have selected. If you have created any joins, you will be able to scroll through to pick different tables. As you choose different tables you will notice that the available choices in the **Columns** window change.

2. **Columns**
   The **Columns** window allows you to view the available columns in the current table selected in the **Tables/Views** list box. Select different tables to access their columns. The columns display the **Description**, field **Name**, field **Type**, the **Size** of the field, and finally the **Scale**.

3. **Add**
   By clicking on the **Add** button, the field highlighted in the **Columns** window is now added into the **Added Columns** window as part of the data your query will return.

4. **Added Columns**
   Here you can see the columns you have added. You can also delete them, move them up and down, or look at the properties of the selected column.

5. **Return distinct rows only**
   Use this feature when a table you need for your report has two fields in it:

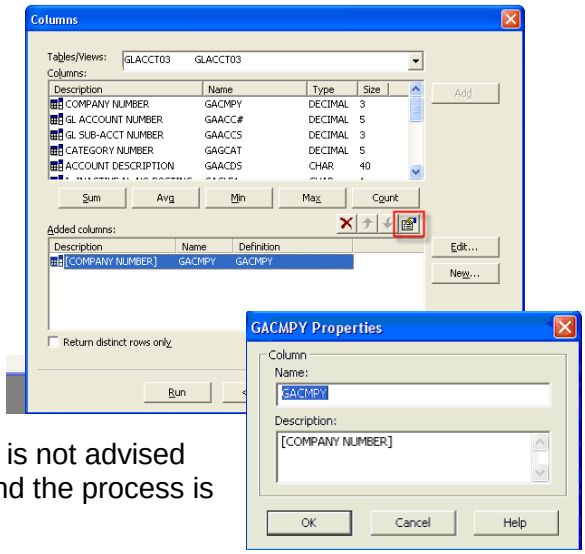
6. **Warehouse** and **Phone Numbers**. In your report you just want to see **Warehouse**. When you retrieve your results, you end up with several duplicate warehouse records. This is happening because each warehouse has multiple phone numbers.

   The solution is to check the **Return distinct rows only** box. This will ensure you do not have duplicate data. If you have checked this box and you are still retrieving duplicate data, check your joins and ensure they have been created correctly.

7. **Query Navigator**
   Click to move to the next or previous dialogue box.

Column Properties

In **Column Properties** you can change column **Name** and column **Description**. To access **Column Properties**, click on the symbol of the hand holding a file in the **Columns** window.



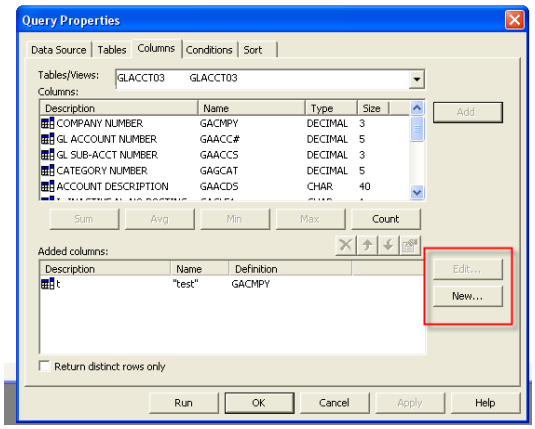
By clicking on the **Column Properties** button, the highlighted column in the **Added columns** section will be available for editing.

You can change the field name. This is often done when the query writes to a temporary table, or when a string function is involved. This is not advised however unless the query process calls for it, and the process is documented.

Changing the field description does not change the header. Changing the description again is typically reserved for process or string functions. String functions and query processes will be covered in greater detail in Query III.

Edit and New

The **Edit** button allows you to create string functions over the selected column. The **New** button allows you to create general expressions, if/then statements, and value substitutions over the selected column. These functions will be covered in greater detail in Query III and IV.

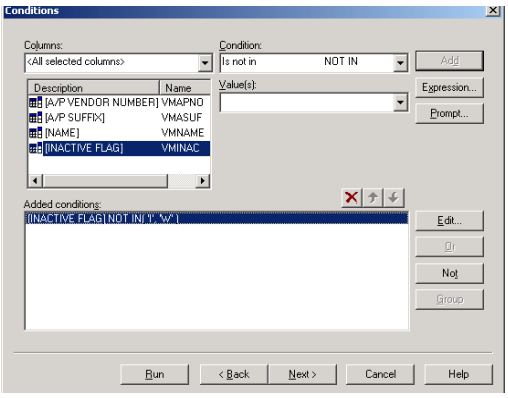


Conditions

A **Condition** is a parameter for the returned data of a column. Once you have selected the **Collection**, **Table**, and **Columns**, it's now time to put conditions on the query. For example, Mary has asked to see a warehouse report for the NW Region. The table continues to bring back the warehouses for all the regions. The solution is to find the **Region** field and select a condition that will show only records that are in the Northwest.

Setting Conditions/Criteria

After clicking on the **Next** button from the **Columns** window, the **Conditions** window displays.



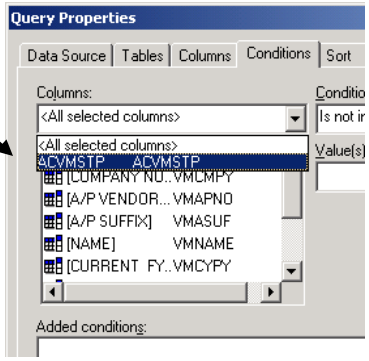
Conditions or criteria are filters used to tailor the information to fit the user's need. For example, you might choose to view results for a particular region or set of warehouses rather than the whole company.

> You can set conditions on any column in a table regardless of whether or not the column will appear in the final result set.

Note that you do not need to set any conditions to create a query; however conditions create efficiency in obtaining and presenting results.)

To set a condition:

1. Select a field in the **Columns** area. If you want to set a condition on a field not selected to be displayed, select the table name from the **Columns** dropdown menu. All of the available columns will be displayed.
2. Click on the dropdown arrow under the **Conditions** area to view all available operators. Select one operator.
3. Type in a value or values in the **Value(s)** field, or click on the dropdown arrow to see all pertinent values for the selected field. This function may take a few moments to execute. Values are case sensitive, so be sure to type them in exactly as they appear in the table. When in doubt use all uppercase.
4. Once a condition is set, click on the **Add** button. The condition is added to the **Added Conditions** area.



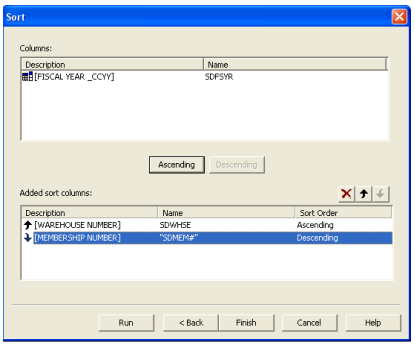
Some of the most frequently used operators are listed below:

=	Equal to one exact match
< >	Not equal to the selected value
Is in	Any and all matches from a list of values
Is not in	Excludes any and all matches from a list of values
<	Less than
>	Greater than
Is between	Use to retrieve information between two numeric values
Is like	Use for wildcard searching; "%" represents the unknown

When conditions have been set, click on the **Next** button. The **Sort** window displays.

Sort

The last step in building a query before running it is to select any sorts that you want, and place them in the order you want.



The first box is called **Columns** and lists all the columns available to pick sorts by. As you choose a sort, the column chosen moves from the **Columns** box to the **Added sort columns** box. You have a choice of **Ascending** and **Descending** sorts. If you have multiple sorts added, you can move the order of the sorts by using the up and down arrows. You can also delete sorts with the red "**X**."

EXERCISE 3: OPEN WAREHOUSES FOR WASHINGTON STATE

You want to create a list of open warehouses for Washington, similar to Option WPH – Warehouse Phone Inquiry.

1. Open Query, then
    a. Select File.
    b. Select New.
    c. Click on Query.
        To work in Query you must sign on to the EIS400 server. The sign on may appear immediately after accessing Query, or when the first existing or new query is opened.
    d. Enter your iSeries400 user profile in the *User* box.
    e. Enter your AS400 password in the *Password* box.
2. Press Enter or click on OK.
3. Add the table you will use:
    a. Click on the down arrow of the *Library* box.
    b. Click on INPRDINV.
    c. Click on the down arrow of the *Table* box.
    d. Click on the INPHON.
    e. Click on Add.
    f. Click on Next to review the columns available in that table.
4. Once you have the right table, select:
    a. Warehouse Number, Warehouse Region, Warehouse Name, State and Opening Date as your columns.
    b. Click on Next
5. Create Conditions where Region equals NW and State equals WA:
    a. Click on WAREHOUSE REGION in the *Column* box.
    b. Verify = is in the *Condition* box.
    c. Enter NW in the *Values* box. (the entry must be in capital letters.)
    d. Click on Add.
    e. Click on STATE in the *Column* box.
    f. Verify = is in the *Condition* box.
    g. Enter WA in the *Values* box. (the entry must be in capital letters)
    h. Click on Add.
    i. Verify the conditions in the *Added conditions* box.
    j. Click on Next.
6. To view warehouses in an ascending order:
    a. Click on Warehouse Number.
    b. Click on Ascending.
7. Click on Finish.
8. Make sure Enable Batch Processing is checked.
9. Select Run.

10. Click on Run Query.
11. Click on Job Status:
    - If Job Status reads “The batch job has successfully completed,” move to the next step.
    - If it reads “The batch job is still executing,” keep checking the job status, clicking on Refresh, until it reads completed. Select Close.

    (**Note:** the number of returned rows should be greater than 0. If they are not, there is no data meeting the conditions. The file is empty or the query is faulty.)
12. Make sure batch processing is checked.
13. Select Run.
14. Click on Retrieve Batch Results.
15. The screen should look similar to the in class example.

EXERCISE 4: FIND PENDING DELETED ITEMS AND ON-HAND SELL UNITS

In the second Query exercise, you will create a query that lists items for Department 12 in the Northwest. By reviewing Collection you will find that INPRDDTA has inventory production data in it. Further research will lead you to the INWWIIP file. This table is similar to the AII – All Item Sales & Inventory Detail or AIV – All Item Sales Inventory options on the iSeries400.

The following information will be defined to narrow the search: a specific region, a specific department, a specific item status.

1. Start a new query. Refer to the previous exercise for specific instructions.
2. Select or enter INPRDDTA in the *Library* box.
3. Select or enter INWWIIP in the *Table* box.
    a. Click on Add.
    b. Verify the correct table is now in the *Added tables/views* box.
    c. Click on Next.
4. Select the following columns: COMPANY NUMBER, REGION CODE, DEPARTMENT NUMBER, WAREHOUSE NUMBER, ITEM NUMBER, ITEM DESCRIPTION, ON/HAND/SELL UNITS, ITEM STATUS.
    a. Verify these are all in the *Added columns* box in the above order.
    b. Click on Next.
5. Click on COMPANY NUMBER on the left-hand side.
    a. Leave the condition as =.
    b. Type 1 in the *Value* box.
    c. Click on Add or press ENTER.
6. Click on REGION CODE on the left-hand side.
    a. Leave the condition as =.
    b. Type NW in the *Value* box.
    c. Click on Add or press ENTER.
7. Click on DEPARTMENT NUMBER on the left-hand side.
    a. Leave the condition as =.
    b. Type 12 in the *Value* box.
    c. Click on Add or press ENTER.
8. Click on ITEM STATUS on the left-hand side.
    a. Leave the condition as =.
    b. Type P in the *Value* box.
    c. Click on Add or press ENTER.
    d. Verify the conditions in the *Added* conditions box.
    e. Click on Next.
9. Click on WAREHOUSE NUMBER in the *Columns* box.
    a. Click on Ascending.
10. Click on ITEM NUMBER in the *Columns* box.

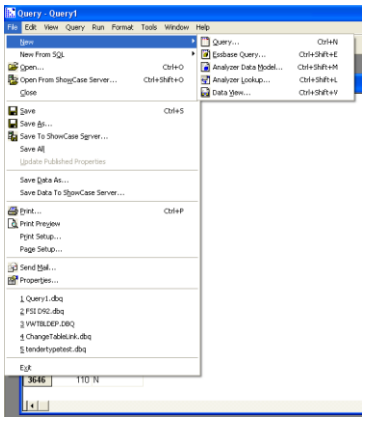
a. Click on Ascending.

11. Click on Run.
12. Make sure batch processing is checked.
13. Check the Job Status under the *Run* menu.
14. When the job is complete, click Retrieve Batch Results under the *Run* menu.
15. To eliminate the items with 0 on hand, add an additional condition, by clicking on the Condition button.
16. Click on ON HAND/SELL UNITS on the left-hand side.
    a. Click the down arrow on the Condition box and select <>.
    b. Type 0 in the *Value* box.
    c. Click on Add or press ENTER.
    d. Click on Run.

MENUS, OPTIONS, AND TOOLS

File Menus

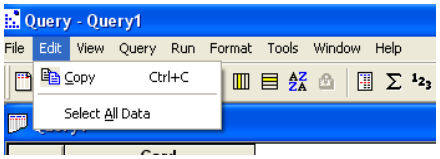
In the initial **File** menu, there are several options that are not within the scope of this course. The options that will be covered are Creating a New Query, Saving a Query, and Saving Data As. (Creating a new Query from SQL will be covered in Query II.)



Create a new query by choosing **File/New/Query**, or by clicking on the **Query** icon underneath the **File** menu.

When you save a query, it is important to note you are not saving the data, but you're saving the *query* that you built to retrieve the data. You are saving the *question*. When selecting **Save Data As**, you are saving the results from the query and not the query itself. Printer functions and browsing are all similar to Windows programs like Word or Excel.

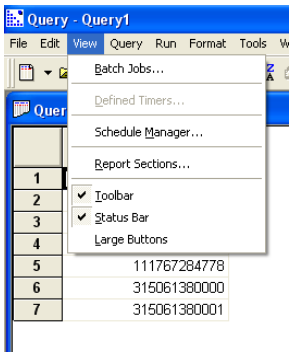
Edit



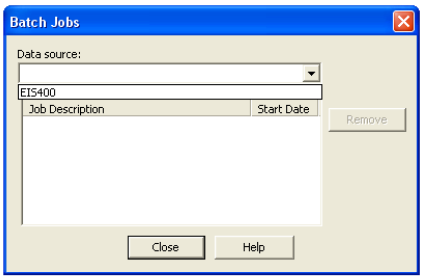
This basic function allows you to select all the data and copy it.

View

The **View** menu allows you to view **Batch Jobs**, **Defined Timers**, the **Schedule Manager**, and **Report Sections**. The **View** menu also allows you to view the tool and status bar, along with making the button sizes larger.

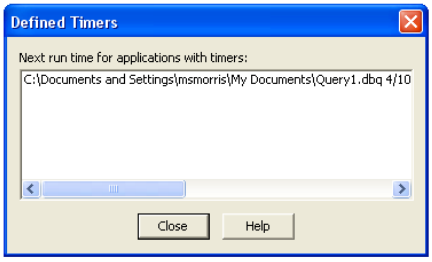


Viewing Batch Jobs



Viewing **Batch Jobs** allows you to see what query batch jobs you are allowed to remove. You may want to remove a batch job if you have submitted it more then once. This will help free up resources for others to run their jobs.

Defined Timers

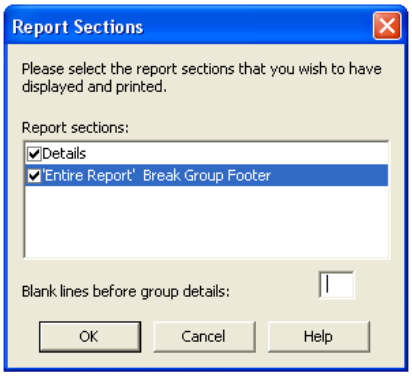


Viewing **Defined Timers** allows you to view the next time a query is scheduled to run.

Schedule Manager

The **Schedule Manager** is a function of Enterprise Reporting. This is currently not available at Costco.

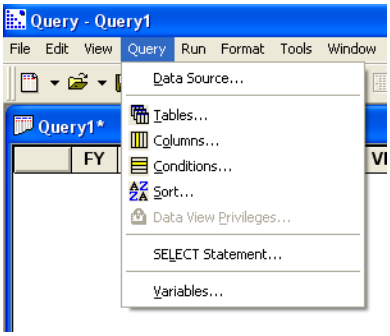
Report Sections



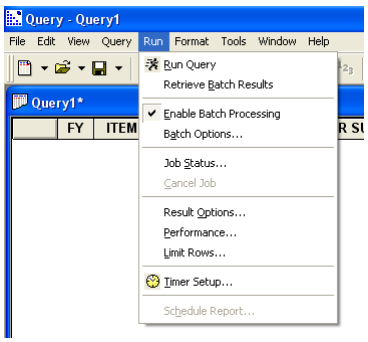
Report Sections allows you to choose whether or not you want to view the details of the current document, or the Break Group Footer.

Query

The **Query** menu contains the following: **Data Source**, **Tables**, **Columns**, **Conditions**, **Sort**, **Data View Privileges**, **Select Statement**, and **Variables**. All of these functions will be covered throughout the four Query courses.



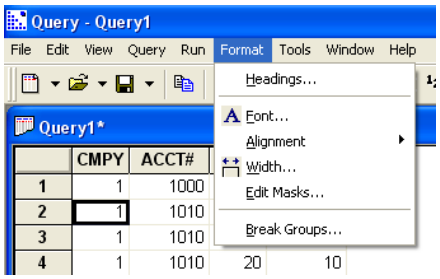
Run Menu



The **Run** menu has several options available. **Run Query** allows you to run your query. **Retrieve Batch Results** brings back your data when the query is finished.

Enable Batch Processing lets you choose between batch and interactive processing. Most profiles default to having batch only. **Batch Options** opens a window where you can view and change the job attributes, where results are written to, and scheduling for queries is available.

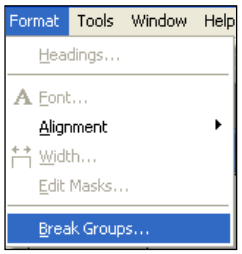
Format Menu



The **Format** menu gives you access to changing the **Headings**, **Font**, **Alignment**, and column **Width**. From the **Format** menu you can also create **Edit Masks** and **Break Groups**.

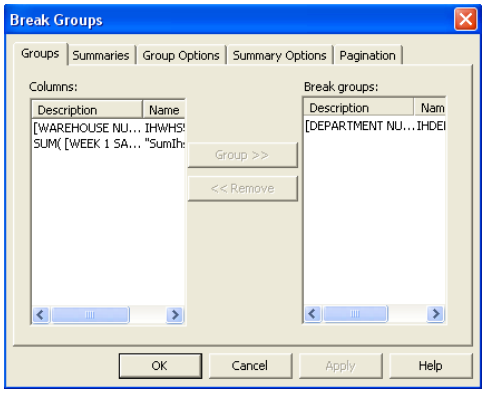
Break Groups

Break groups allow you to group a report, making it easier to read, compare, summarize and interpret data. You can add group-based calculations (ranking, percent of total, running total, and running count) based on a specified break group level. Most often break groups are used to create summaries in the report.



1. On the **Format menu**, select **Break Groups**.

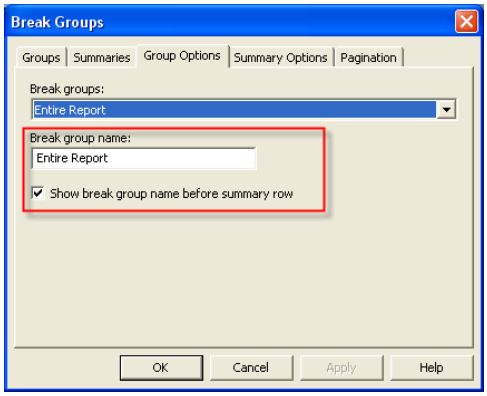
2. In the list box, select the column to apply a break group.



To change the name of the break group:

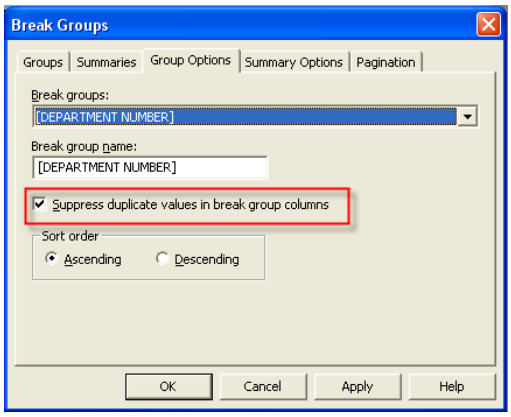
1. Select the **Group Options** tab.
2. Type the name you want in the **Break Group Name** text box.

   This name is then displayed in the column to the left of the data
   in your report.



To suppress duplicate values that appear in the break group:

1. Select the break group that you want to suppress duplicates in. This will now allow you the ability to:
   a. suppress duplicates
   b. place a sort order
2. Select the check box **Suppress Duplicate Values in Break Group Columns**.



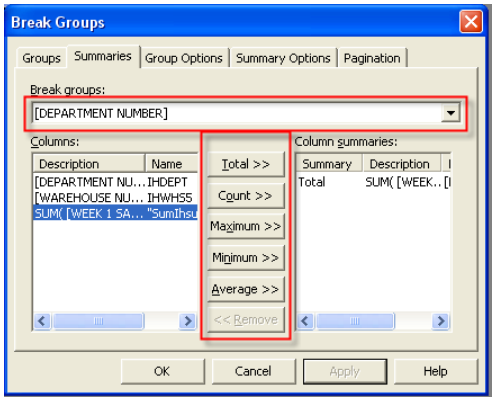
Choosing the **Ascending** or **Descending** option will apply a sort to the rows within a break group. This will take priority over any sort order specified in the Sort window or the Select Statement window. (**NOTE:** You cannot change the sort order of a column in a break group with the Sort window in Query, or a select statement.)

Summaries

Using the Summaries tab, you can place summaries on the different break groups already added to your report.

1. Select the break group that you want to place a summary on.
2. Highlight the column for your summary.
3. Choose the type of summary you want.

By not choosing any break group to place a summary on, you will be adding the summary to the entire report.



Summary Options Pagination

You can add text that appears with the summary, and you can specify whether nulls count as values.

EXERCISE 5: BREAK GROUPS FOR LOCATIONS

You want to create a list of locations grouped by Company, Country, Division, Region and State.

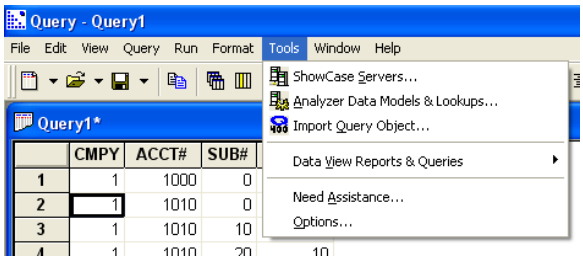
1. Create a query using the Table INWCTLP from the Library INPRDINV.
2. Select all of the fields except for the Miscellaneous flags.
3. Run the query and view the results.
4. Select the Format menu.
5. Choose Break Groups.
6. Create a group on Company Number, Country, Division, Region Code, and State.
7. Run the query and view the results.

EXERCISE 6: BREAK GROUP ON REGION AND STATE WITH SUBTOTALS AND AVERAGES OF DAILY SALES

You want to create a list of locations grouped by Company, Country, Division, Region, and State.

1. Create a query using the Table INIHSTP in the Library INPRDDTA.
2. Select fields Warehouse Number, Departments Number, Week 1 Sales.
3. Create conditions where Fiscal year equals the current Fiscal Year, the format is YY. Warehouse IS IN 110, 64, 8 and Department IS IN 11, 12, 17
4. Sort by Warehouse in an ascending Order
5. Sort by department in an ascending order.
6. Run the query and view the results.
7. Create break groups on the fields Region and State.
8. Run the query and view the results.
9. Click on the Totals Σ button.
10. Select the Format menu and Break Groups again.
11. Click on the Summary Options tab.
12. Highlight Sum([Daily Sales at Sell]).
13. Select the Average >> button.
14. Click on the Summary Options tab.
15. Click on the list box arrow under the break groups and choose Region Code.
16. In the Summary text box type “Region Subtotal”.
17. In the Column summaries box, click on the list box arrow.
18. Select Average: SUM(MBDSEL)
19. In the Summary text box type “Region Average”.
20. Repeat steps 12 through 16, modifying for the State field.
21. Run the query and view the results.

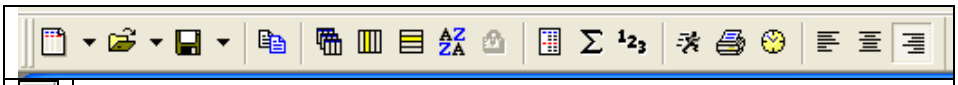
Tools Menu



The **Tools** menu has many options that are not covered in this course work. The only option covered is **Options**.

Shortcut to Tools

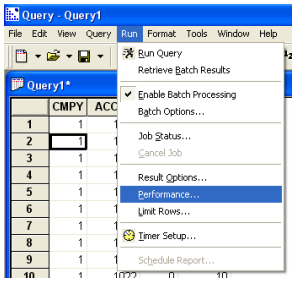
You can also access many Query functions via the shortcuts available just below the menu options. Here are the shortcuts and what they do:



	Open a new query
	Open an existing query
	Save the query
	Copy the data to the clipboard
	Access the table set up of the query
	Access the columns (fields) of the query
	Access the conditions of the query
	Access the sorts for the query
	Specify break groups and summaries
	Create a sum or total of the data
	Count the number of responses in the column
	Run the query
	Print the query
	Set the timer to run the query
	Align data to the left
	Center data
	Align date to the right

Tools (refresh, timer, estimator)

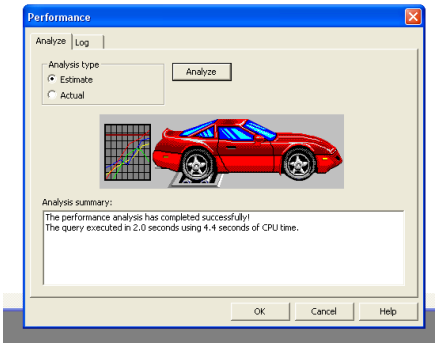
Performance



1. On the **Run** menu, click **Performance**.
2. Select an analysis type from the **Analysis type** box.
3. Click **Analyze**.

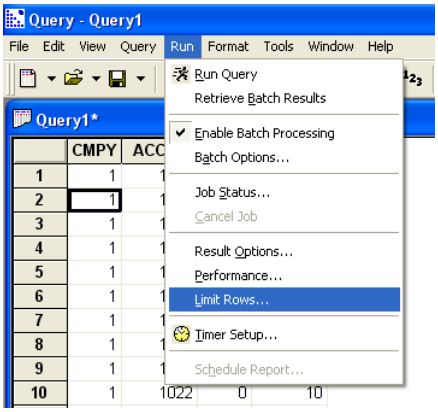
If the query you are analyzing contains prompts, you are asked to enter values for the prompts. **Performance Analyzer** evaluates the query. A message indicates when the analysis is complete. The **Analysis summary** box displays the estimated or actual run time of the query, depending on which option is selected in the **Analysis type** box, and then gives suggestions for improving the query's performance.

Analysis Types: If you select **Estimate**, **Performance Analyzer** will evaluate the active query without running it. This option is the fastest way to evaluate the performance of a query.
If you select **Actual**, **Performance Analyzer** will run the active query to evaluate its performance. This option returns the most accurate and complete information.

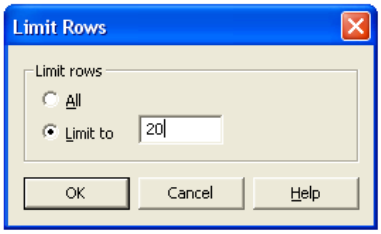


Tip:
The **Performance** command is only enabled when your data source is an iSeries option.

Limit Rows



This limits the amount of rows retrieved from the data source, which comes in handy when building or testing a query. Rather than retrieving thousands of rows, you can retrieve 5, 20, 100, or any specified amount you want. Choose enough to ensure you are getting the results you expected. Always remember to verify your data.

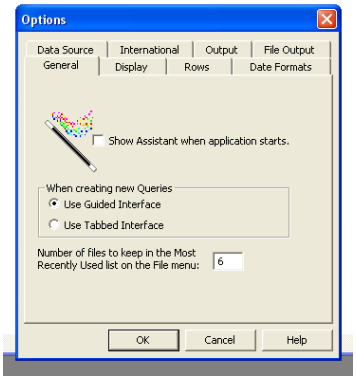


Options

Within **Options** there are eight tabs:

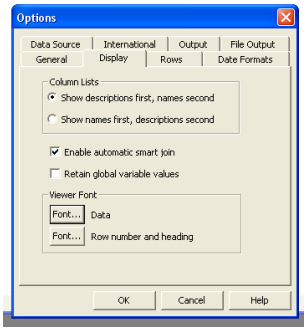
1. **General**
2. **Display**
3. **Rows**
4. **Data Formats**
5. **Data Source**
6. **International**
7. **Output**
8. **File Output**

General



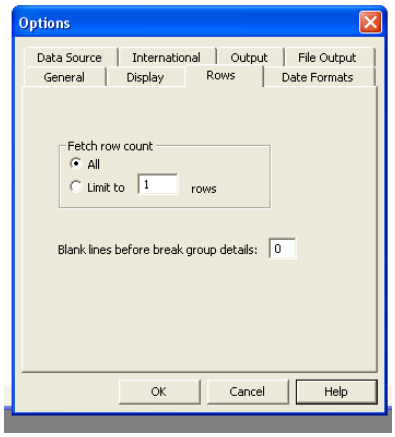
General gives you a few options for setting up the query wizard. A helpful item on this tab is the **File Menu** option. Here you can change the number of recent queries ran that displays on the **File Menu**.

Display



The **Display** option specifies the default settings for how table and column names are displayed, and whether the application automatically joins tables.

Rows



In the **Rows** tab you can set an automatic default for how many rows you want retrieved for your queries. Likewise you can enter a default for blank lines to view before breaks in your report. It is best to use the default settings.

Data Formats

This will be covered at a later time. It is highly recommended to keep the default settings for this tab.

Data Source

See Troubleshooting.

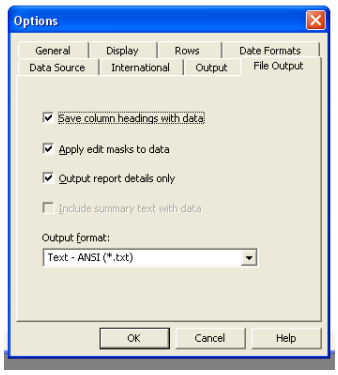
International

This is not applicable, as Query at Costco is only offered in English.

Output

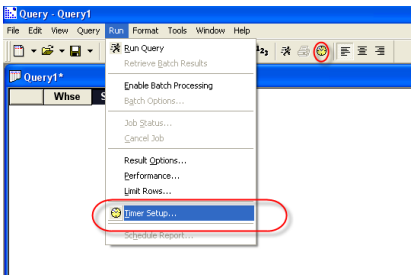
The **Output** dialogue box gives you options for changing the default output of your data. It is best to keep the default settings.

File Output



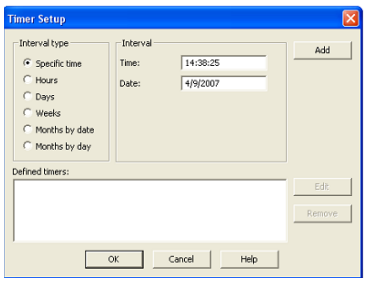
The **File Output** dialogue box gives you options for changing the default file output of your data. **File Output** is essentially how you want the results of your data displayed. For example, you can choose whether or not to save column headings with the data, or apply edit masks you have created in Query to your **File Output**. Keep the default settings.

Timer Setup in Query



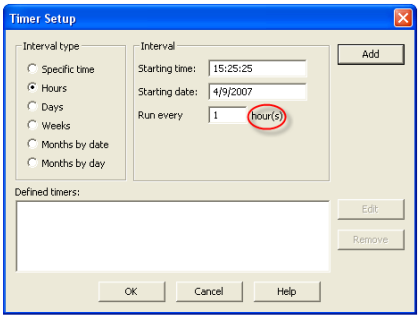
The **Timer Setup** feature allows you to schedule an automatic run of a query. You can access the **Timer Setup** feature in Query through the **Run** menu, or through the **Timer** shortcut located next to the **Printer** icon.

Once selected, the **Timer Setup** dialogue box will appear.



The dialogue box has **Interval type**, **Interval**, **Defined timers**, and the ability to **Add**, **Edit**, and **Remove** defined timers. To create a defined timer, first choose whether the interval is a specific time or if the interval for the query will be on certain hours, days, weeks, etc.

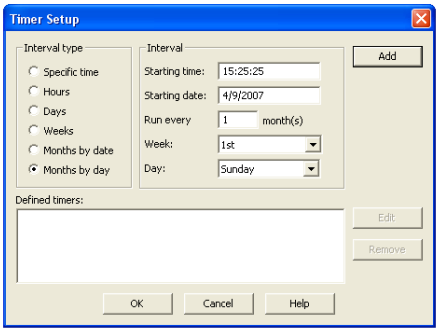
Different interval types will change the options in the Interval window. The previous example represents what you will see in the **Interval type Specific time**.



The intervals: **Hours**, **Days**, **Weeks**, and **Months by date** give you a **Run every** option. The option changes with the interval type selected. Notice **Hours** is the type and **hour(s)** is interval for the **Run every** option.

As you select other interval types, the interval will change with the type. **Months by day** has the ability to choose the **Week** and the **Day** of the month.

Scheduled queries run automatically while the query application is open. Internal buffer errors can happen if you schedule queries to run when you are not at work and leave Query open. This error means your Query application is no longer connected to the EIS400 and will need to be reconnected.



A query may be scheduled to run at a specific time and/or frequency. Scheduling a query is very helpful when:

- The query is very large
- The query is run regularly
- The query needs to run during a user's absence

Steps for scheduling a query:

1. Save the query before scheduling it to run.
2. Schedule the query.
3. Run the query. The query will now be in the batch jobs to process at the scheduled time.

(**Note:** The query does not need to be open at the time it is scheduled to run.)

Questions for Review

True or False: You should always check the return distinct rows only box.

What condition would you use to enter multiple values?

True or False: You have to add a column in your viewable report in order to place a condition on that column.

What is a library, a table, a member, a field, and a record? What are other terms you could use instead?

What is the difference between a physical file and logical file?

Describe Batch versus Interactive.

Can you schedule queries to run when you are not here?

NOTES

UNIT III:
TROUBLESHOOTING AND MAINTENANCE

Modifying an Existing Query
Troubleshooting
Reference

MODIFYING AN EXISTING QUERY

Case Study

Mr. Anderson has completed the Query course and arrives at his desk to try out his new skills. He opens what appears to be a simple query that he runs every Monday to add a new column that his manager has wanted for some time. After adding the column, Mr. Anderson runs the query only to find that it begins failing.

What happened? It could be any number of things that took place. It's possible that the query was in fact one of a three-step process, or that the query was embedded in an Excel spread sheet and just by adding the single column, it has created a domino effect of problems.

Best Practice

The best way to avoid the above scenario is to create a few query folders: one for testing, one for backup and one for production. Let's say you have a query called Adjustments that looks at price adjustments, and you want to make some changes to the report. First, create a testing copy of the query, and develop the new changes with the testing query. When the testing query works fine, you are ready to put it into production and back it up.

If there are spreadsheets or other processes involved with the query, you should first talk to the person who created the process, talk with a Query expert, or continue on to the next Query course to determine if there are special considerations you need to make.

TROUBLESHOOTING

Table of Contents for Troubleshooting

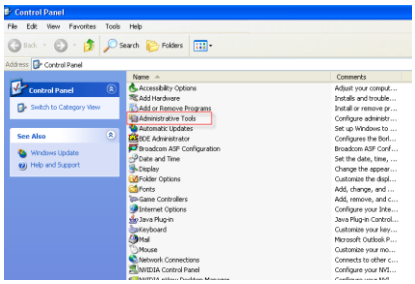
Installation and Profiles

Most computers have Query on the desktop. It is installed by Microsystems with each new rollout. Microsystems also installs Query upgrades as well. If you are noticing complications after an install or upgrade, the best thing to do is call the Help Desk. However, there are a few things you can try first that may solve the problem.

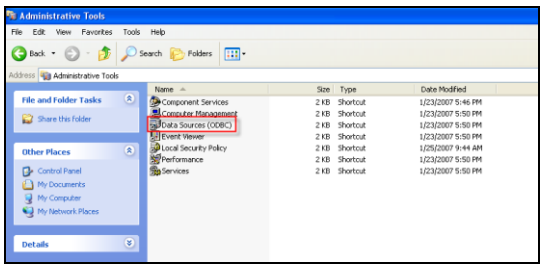
1. Check your Outlook notifications to see if Microsystems has sent a notification regarding the issue you are experiencing.

2. Check your ODBC connection. What is ODBC? Short for ***O**pen **D**ata**B**ase **C**onnectivity*, it is a database access method. ODBC makes it possible to access any data from any application, regardless of which database management system is handling the data.

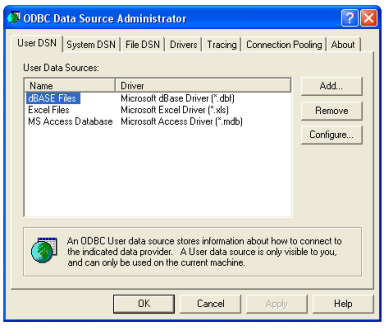
How do you check your ODBC connection? Open your **Control Panel**. You can do this by selecting **Start/Settings/Control Panel:**



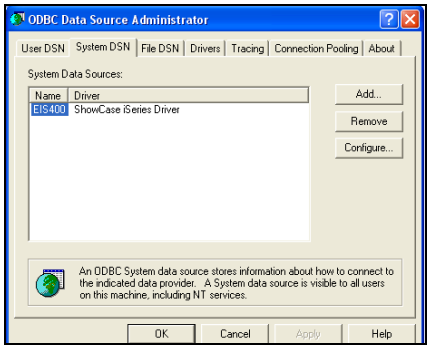
3. From the **Control Panel**, select **Administrative Tools:**



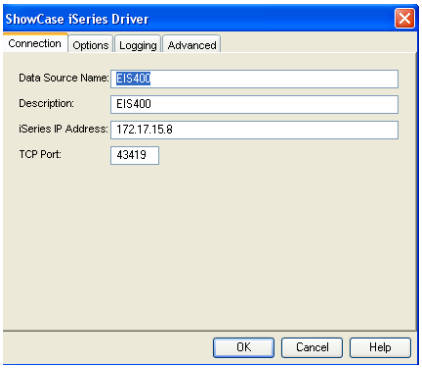
4. You will now see the **ODBC Data Source Administrator** window. Click on the **System DSN** tab.



5. Once you are on the **System DSN** tab, click on the **Add** button.



6. Choose the **Showcase iSeries Driver**. If asked to configure this driver, you will need the following information:
    - Datasource name: EIS400
    - Description: EIS400
    - ISeries IP Address: 172.15.8
    - TCP Port: 43419



7. Try building a simple query. This is just a test to see if it is an application problem or a Query problem. If you can run a simple Query, this is an indicator that the problem is not with the application, but rather with the report.

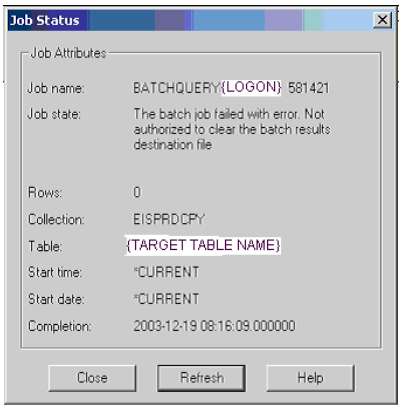
Profiles

Having Query on your PC does not mean that you can use Query. Your manager must request access for you. This is a common reason for Query connection problems. If you have transferred departments or changed jobs, a new profile request may be needed to give you access to Query. The profile requests are located at P:\Forms\Profile Requests, or through Outlook forms. The profile name and password for Query is the same name and password that you use for signing on to the iSeries400.

Common Errors Help Desk

Problem - Not authorized to clear the batch results destination file

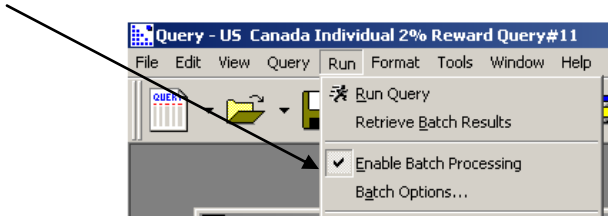
Upon completion of the query, the user gets the following message:



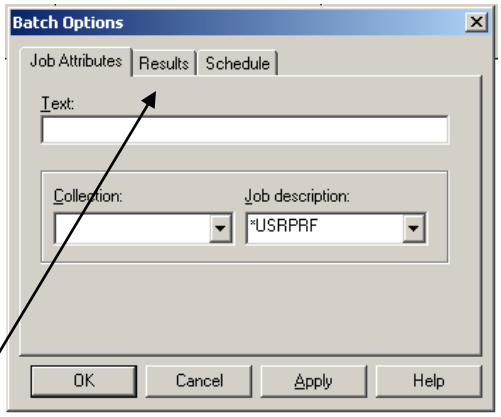
Solution

a. You are running query results to a table that you don't have permissions to, or that does not exist. You will need to change the name of the table to create a new table.

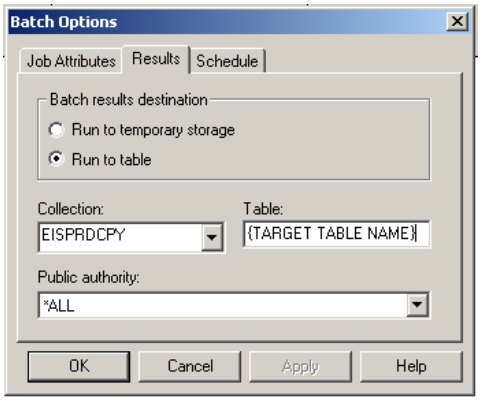
Open your query, go to the menu bar, and select **Run**, **Batch Options**.



b. The **Batch Options** window will open up.



Click on the **Results** tab.



c. You will now see the **Results** part of the **Batch Options** window.

d. You should have a {TARGET TABLE NAME} in the **Table** field.

You will need to change the name of the table. For example, just add a '1' or '2' to the end of the name. Keep table names to no more than ten alpha or numeric characters, though. The iSeries400 (AS/400) handles ten or fewer characters best. Also, make sure the table you have renamed does not already exist on EISPRDCPY, or you will most likely end up with the same message and no results.

e. Click **OK**. Save your query; it should run now.

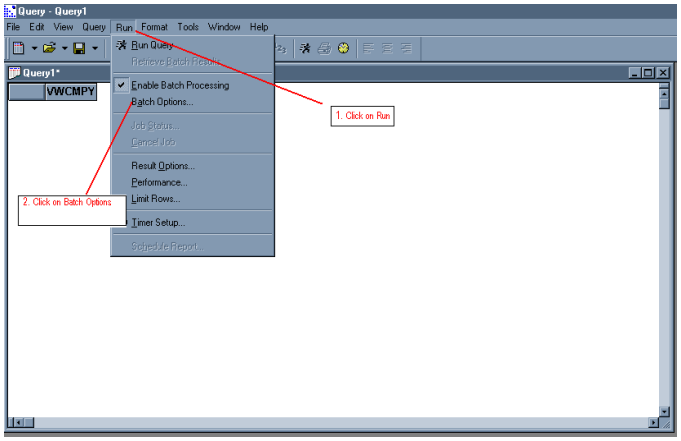
Not Authorized To Job Schedule QDFTJOBSCD
"Error in native method- file name too long- null 0"

Why did this happen?
If you have created a table name that is longer than ten characters, it may work for a time. However, if the EIS400 is brought down for any reason, and then comes back up, the table name will have truncated and the queries will no longer link up.

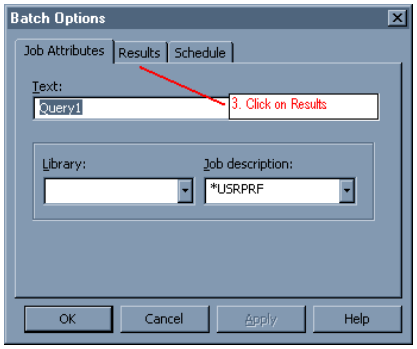
The solution is to rename the table to be no more than ten characters long. Work Tables are covered in Query II and III. Avoid leading symbols or numbers in table names (e.g., 401kreport or “401kRPT”).

How to rename a table

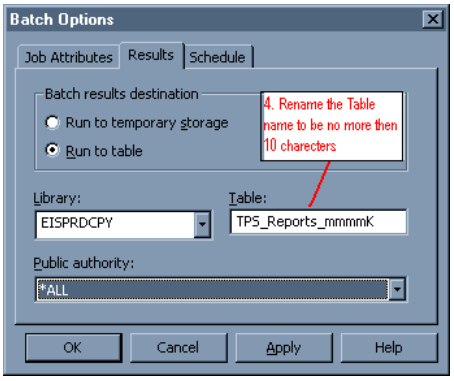
a. Open the query with the error, and then click on **Run** from the menu options.
b. Click on **Batch Options**.



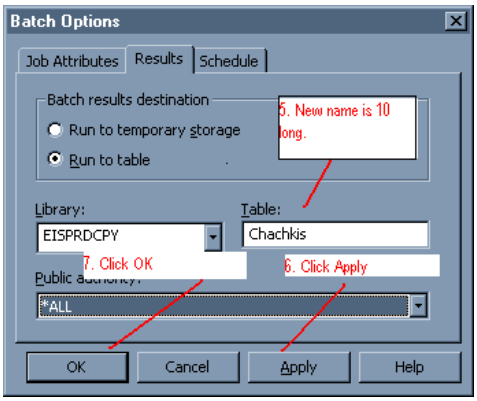
In the **Batch Options** window, select **Results**.



c. Select the **Table** box to rename the table.



d. Rename the table to be no more than ten characters.
e. Click **Apply**; this may take a moment.
f. Click **OK**.



g. Save the query.

Change Table Link error

This error can occur for two reasons. Either your profile does not have access to the **Library** or **Table**, or the table or a field in the table has had an update you need to connect to.

The following questions can determine a profile issue:

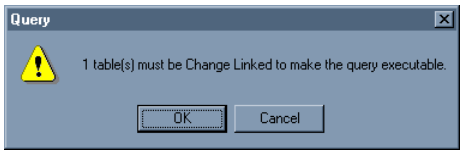
- When was the last time you ran the query? If you ran this query recently and nothing has changed with your profile, you can rule out your profile as the issue.
- Is this a new query?
- Did someone email this query to you?
- If this is a new query or a query that was sent to you, or if something has changed in your profile, check to see if you can access the table independently of this query. You can do this by creating a new query and browsing through the library and tables to see if you can view the table. If you can view the table, this is not a profile issue. *If you cannot view the table or library, this is a profile issue.* Your manager will need to fill out a profile request form to gain access to the collection or table.

If it's not a profile issue:
A more common reason for this error is that a table or field has been changed.

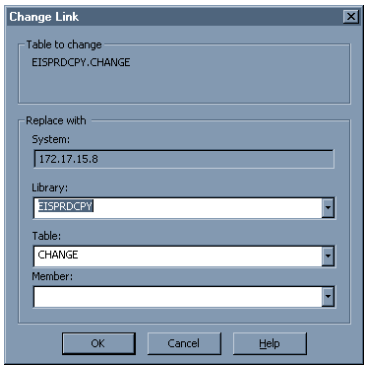
Determine what the problem is:
1. Each change link table error will specify what has changed. Determine what has been changed.
2. Update the query to reflect the new table or field names.

Here are the steps to fix the problem, if your query needs to be updated.
Example:
Library INPRDDTA and Table INWCTLP was changed to
Library INPRDDTA and Table INWCTLFP

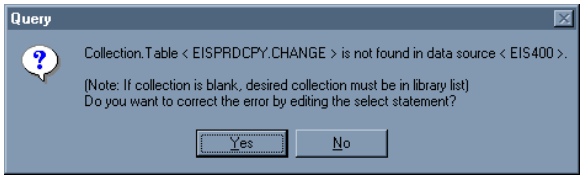
1. You will first see the initial error. Click **OK**.



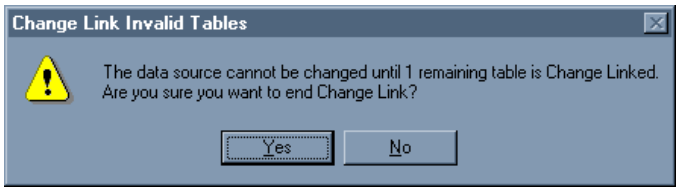
2. Click **Cancel** on the following window, to determine if it is a table or field that needs to be updated.



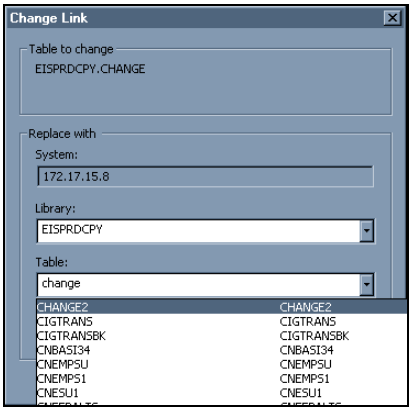
a. You will see one of the following error messages. The first means a table change needs to occur. Continue on to step 3.

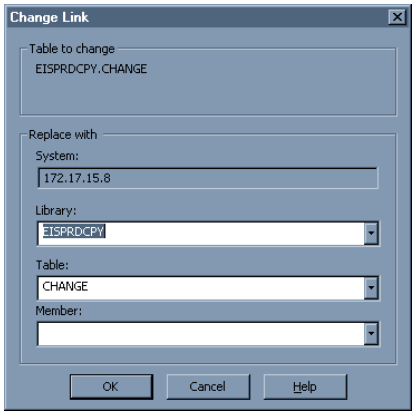


b. The second error message means a field change needs to occur. Continue down this document to the steps for field change and follow those directions.



3. Next you will see a **Change Link** window. Select the **Table** list box arrow to display all tables available. You can also key in the new table name if you know it. Typically, an email is sent out letting users know of tables that will be changed.
Note: There is a grayed-out box at the top of the window called **Table to change**. Make sure the library matches the library from the **Table to change** message with the drop-down menu below.

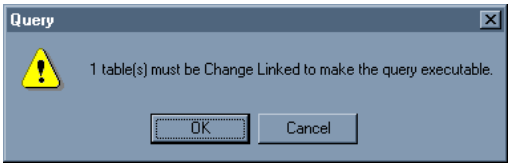




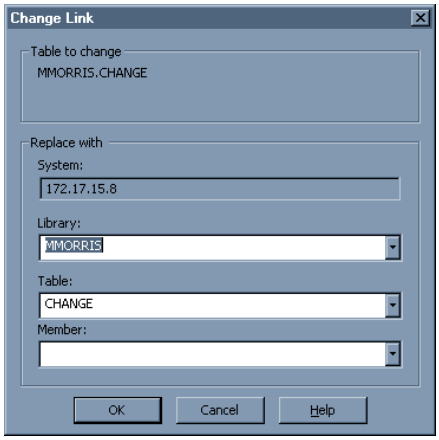
4. After finding the correct table, select **OK** and run the query (unless there is more than one table that has been updated and needs changing).

If your table does not need to be updated, maybe a field has changed and needs to be updated.
Example:
Library INPRDDTA and Table INWCTLP and Field WCWHSE was changed to
Library INPRDDTA and Table INWCTFP and Field WCWHS5.

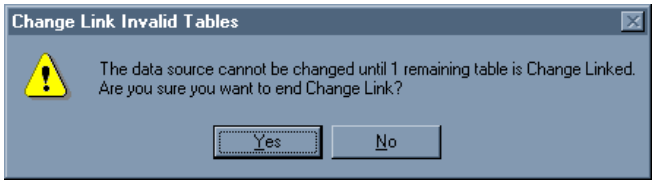
1. You will first see this initial error. Click **OK**.



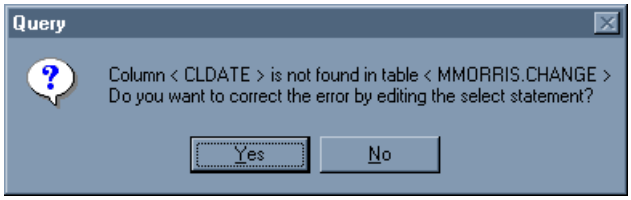
2. Next you will see a **Change Link** window. You will not be able to access fields from this window, so select **Cancel**.



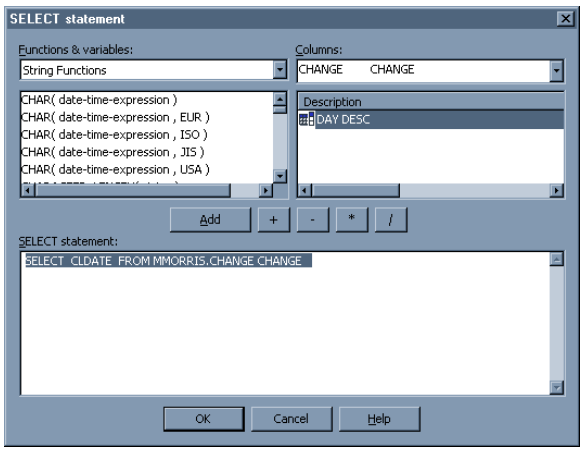
3. Select **Yes**.



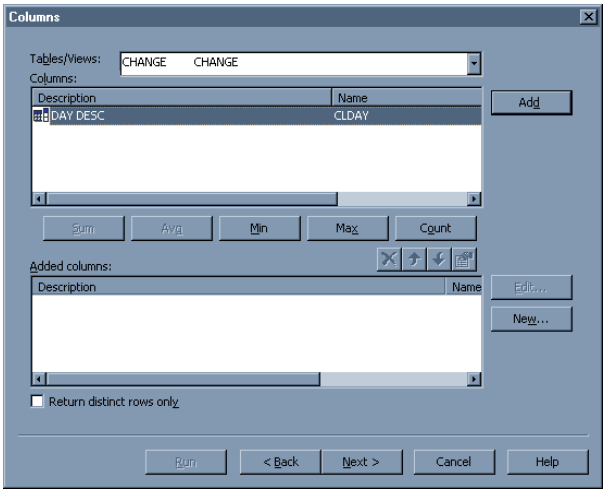
4. You will now see a window that will tell you what field needs to be changed. In this scenario, CLDATE is not found in library/table MMORRIS.CHANGE. Make a note of the library, table, and field.



Select **Yes** to view the select statement. The select statement will appear in the lower window. With more complex select statements, it is best to copy and paste the entire statement into a Word document, and update the statement in Word then copy the revisted SQL Statement back into your Query.. This makes it easier to edit.

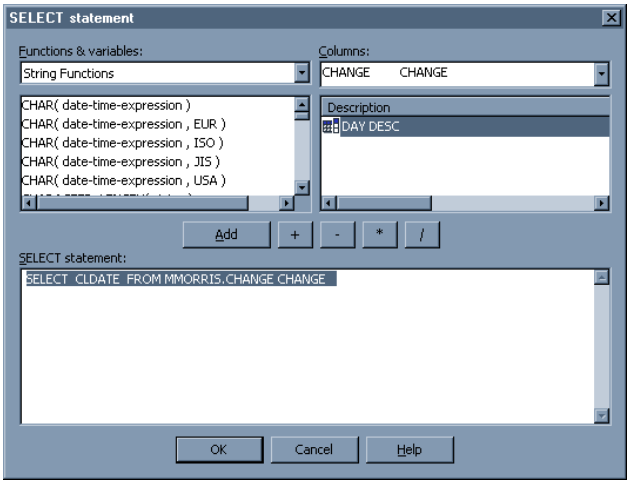


5. The next step in this process is to cancel out of the query, open a new query, and then browse to the library and table from the error message in step 4. In this scenario, the Library is MMORRIS and the table is CHANGE. After adding the library and table, you can then navigate to **Columns** and view all of the columns.

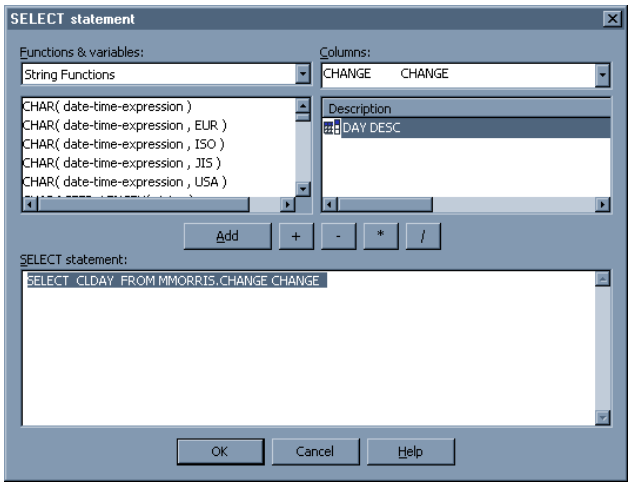


6. The next step is to find the new column. In this scenario finding the new column is simple, as there is only one to choose from. CLDATE was changed to CLDAY. In some cases it may not be so easy, and you will have to know what you are looking for. For the most part it may be self evident (i.e., a column named WCWHSE no longer exists). After reviewing the new library and table, a column called WCWHS5 is now present. There is a good chance this is the field you are looking for. If you are not sure, check for email sent by the Data Warehouse group to notify users of changes. After the correct field is determined, make a note of it and close out of the new query.

7. Open the query with the error and follow the steps that lead to the **Select statement**, as outlined in steps 1 - 5.



8. In the **Select statement** window, change the field CLDATE to CLDAY. Click **OK** and save the query. At this time you may find that there is another field that has been updated. If this is the case, repeat the process until all fields have been updated. Be sure to save the query!



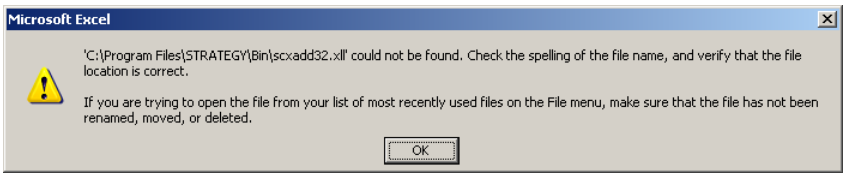
Internal Buffer Error

This error occurs when a user has signed into Query, and then left it open overnight. The system comes down at night and when it does, all queries are terminated. This causes the internal buffer error. The solution is to completely close and then restart Query.

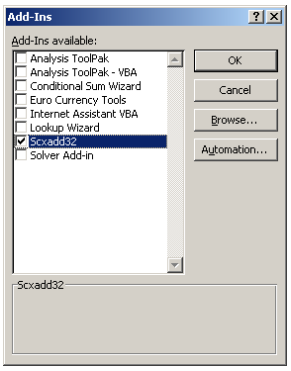
Showcase Query Add-in

You may notice after an upgrade that you have lost your **Showcase Query Add-in**. Here are the steps for adding it back in.

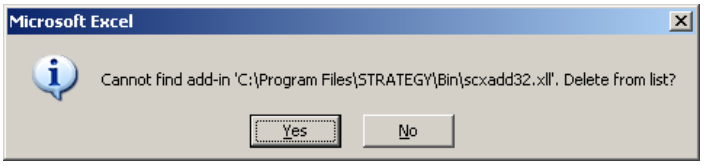
1. Open Excel. You will only receive the following error if you still have the old version of the add-in loaded.



2. Go to **Tools** and then **Add-Ins**.

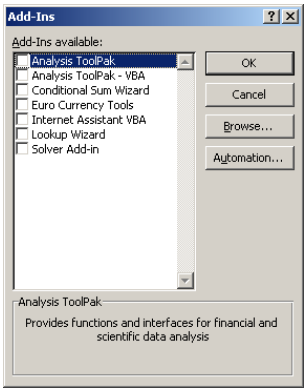


3. Uncheck “Scxadd32”, click **Yes** to the message seen below, and then click **OK**. The old **Add-In** has now been removed from Excel.

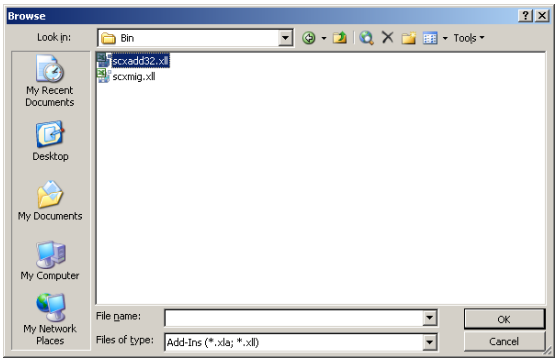


4. Open Excel.

5. To add the new **Showcase Add-in**, go to the **Tools** menu, down to **Add-Ins**, and then click the **Browse** button.

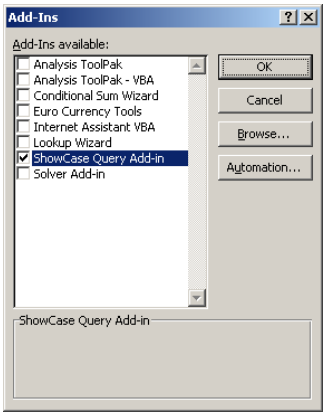


6. Click on the drop-down arrow and browse to the following location: C:\Program Files\Showcase Suite\Bin.

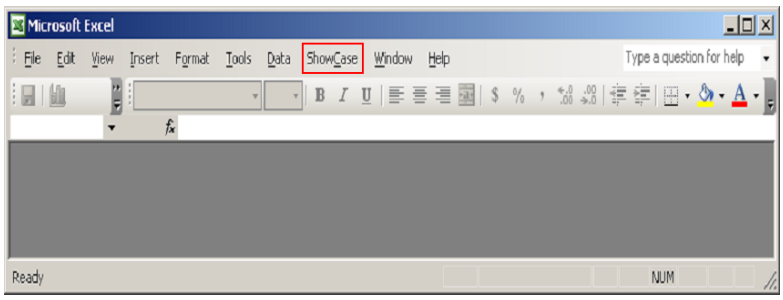


7. Highlight the following file “Scxadd32.xll” and click on the **OK** button.

8. The **Showcase Query Add-in** should now be available. Place a check mark next to it and click the **OK** button.

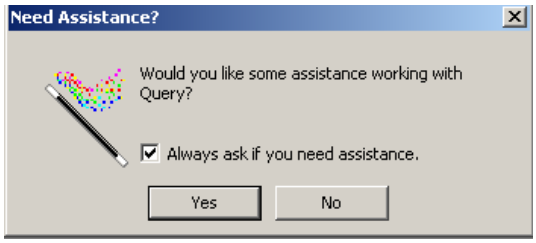


9. The **Showcase** menu should now be available at the top of the screen.

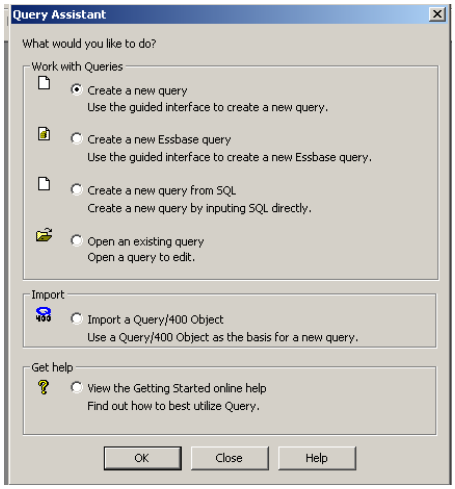


Choosing your Data Source

When you first open Query, you will be asked if you would like assistance. By un-checking **Always ask if you need assistance**, you can remove this wizard.



If you decide to use the wizard by clicking **Yes**, the following window will open.



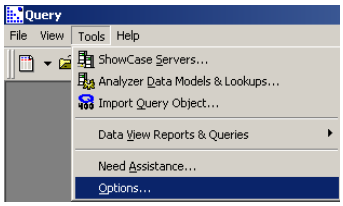
The only items from this window covered in the scope of this class are:

- **Create a new query**
- **Create a new query from SQL**
- **Open an existing Query**
- **Getting Help**

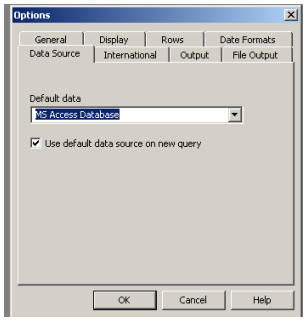
All items can be accessed easily from the **Tools** menu.

Closing out of the Query Assistance will bring up the Query interface. You will see four options available: **File**, **View**, **Tools** and **Help**. The first step is choosing your data source.

Choosing your Data source

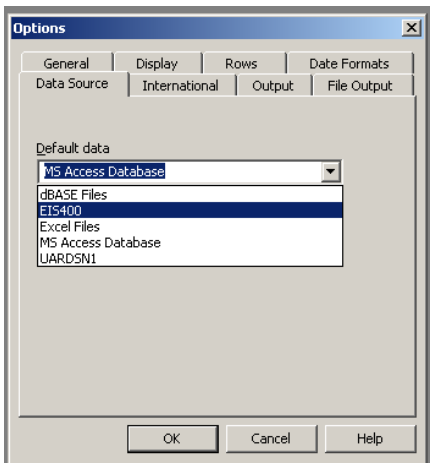


1. Select **Tools** and then **Options** from the drop-down menu.



2. In the **Options** window select the **Data Source** tab.

3. Notice the checkbox next to **Use default data source on new query**. Make sure this is checked.



4. In the **Data Source** tab, click on the black list box arrow to choose a data source. For queries you will want the EIS400. If EIS400 is not available, you will need to set up your ODBC or contact the Help Desk.

After successfully setting up the EIS400 as your default data source, you will not be prompted to do so again.

Troubleshooting Questions

Has anything changed with the profile?
Has anything changed with the software (upgrades)?
Is this a timing issue?
Has something changed with the business (new period or new items etc.)?
Has anything changed with the query?

ADDITIONAL EXERCISES

EXERCISE 7: PUTTING IT ALL TOGETHER

Create a query showing all the managers and supervisors in the NW and LA using Option ORG – Warehouse Organizational Detail. Create a break group on Region Code and Warehouse Number. After you have successfully run the query, save it and schedule it to run 5 minutes from the current time. Close out of the Query application wait 6 minutes to reopen the application to retrieve your results.

EXERCISE 8: LOOKING UP YOUR COSTCO MEMBERSHIP CARD NUMBER

In this exercise you will be looking up your Costco card number. This can actually be used for many different purposes. Some departments may use this to verify a deadline for renewal; others may want to review shopping history.

LIBRARY/TABLE
    MBPRDDTA/MECABSP
COLUMNS
    CARDNUMBER, CARD ENABLE DATE, CARD STATUS, CARD DISABLE DATE, CARD STATUS REASON
CONDITIONS
    CARDNUMBER = (enter your own card number)

EXERCISE 9: SUMMARIZE DAILY SALES

Summarize the daily sales in dollars and in units for item #2 at warehouse 110 for a specific date. For this query try changing the viewer in Result options so your results display in Excel rather than in Query.

LIBRARY/TABLE
    EISPRDDTA2/INIDyyppP

    *The yymm equals year and period. For instance if you wanted Fiscal year 2012 Period 06, you would look for the table INID1206P.

COLUMNS
    WAREHOUSE, ITEM NUMBER, DAILY SALES IN UNITS, DAILY SALES AT SELL

    *Once Daily sales has been added. Click on Daily sales in units in the added columns window, then click on the Sum button. Do the same for Daily sales at sell.
CONDITIONS
    COMPANY NUMBER = 1
    WAREHOUSE NUMBER = 110
    SALES DATE = (any sales date you want for the period you have chosen. If you don't know what the format for the sales date is, click on the list box arrow in the Values field.
ITEM NUMBER = 2

EXERCISE 10: DATES

Interested in dates? Try the UTCLDRP table, found in the PRDGPL collection, which contains calendar information. Try running three or four different queries with conditions to get use to the data in this table. You will find later that being familiar with UTCLDRP is very helpful.

EXERCISE 11: BUILD A QUERY; VIEW RESULTS (PRACTICE)

1. Select Company Number, Warehouse Number, Department Number, Membership Number, Item Number, Sales Date From Collection, and Table EISPRDDTA2/INIyyppP. Where the conditions are Company number = 1 and Warehouse =110.
2. Next click on the timer set up and schedule the query to run 10 minutes from now. You will be asked to save the query.
3. Once the query is saved, click on Retrieve batch results, and completely close out of Query.
4. In 20 minutes, open the saved query and view your results.

EXERCISE 12: INCLUDING iSeries OPTIONS

Bring up a few iSeries400 options and use the steps on page 8 to look at the tables used to build the options. Create a Query that retrieves the same data found on the iSeries Report.

EXERCISE 13: CATEGORY LISTINGS

Select the fields Company, Department, Item, Item Description, and category codes one, two and three from the library INPRDDTA and the table INITMMP. In the conditions, create a condition for your company: US is 1 and Canada is 4. Also, you will want to create a condition for your department.

EXERCISE 14: SHC

Using the Table INCATHP, create a query that resembles Option SHC – sales History by Category.

EXERCISE 15: VENDORS

List the vendors and their items.
Use table INWWIIP in the library INPRDDTA. Sample Vendor number 381.

Questions for Review

Should you call the Help Desk to have them build you a Query?

Where can you get additional help for Query?

If you receive one of the error messages featured in the troubleshooting guide, what should you do?

Most troubleshooting is necessary due to incorrect data. If the data is normally correct at 8:00 a.m., but today it's not, should you start the troubleshooting process in the morning, or wait until later in the day? Why?

How would you troubleshoot a query that was built incorrectly but the numbers are close?

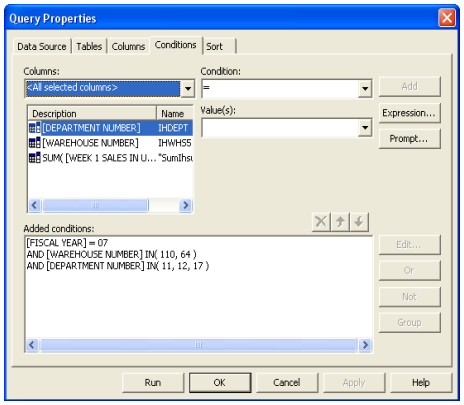
UNIT IV: CONDITIONS/PROMPTS

Batch Options
Conditions
Prompts

CONDITIONS

This chapter discusses the following:

- Explanation of the Conditions window
- Review of the Conditions window tools and functionality
- Discussion on how and why conditions are used
- Examples of values
- Adding conditions in the right order
- How to edit conditions



The Conditions Window

The **Conditions Window** lists the specified conditions for the query. From this window, you can choose to add new conditions, edit existing conditions, or remove conditions. Also you can group the conditions or specify relationships between the conditions.

The data source compares each value of the column you specify to the search condition you specify. If the comparison is true, the row appears in the data.

Multiple conditions are processed in order from top to bottom; however, conditions enclosed in parentheses are processed first. Now let's take a closer look at the specific functionality of the Conditions window.

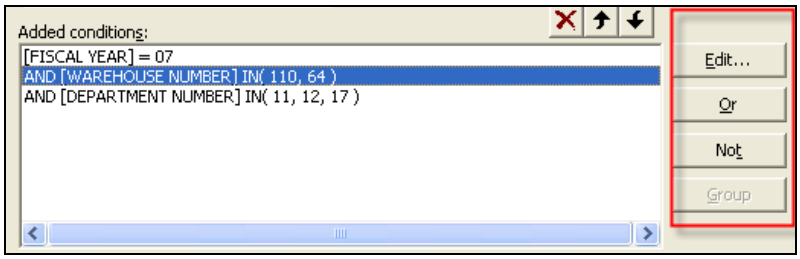
Columns
A dropdown list box contains a list of selected tables for your report. This box contains a list of columns in the currently selected table. Select a table and a column from the list boxes.

When **<All Selected Columns>** appears in the drop-down list box, the list of columns include all columns that are in your report. Query defaults to **<All Selected Columns>**.

Condition
=, >, <, <=, >=, <>, BETWEEN, NOT BETWEEN, IS IN, and IS NOT IN conditions are covered in Query I.
In this section, we cover the following conditions: LIKE, and NOT LIKE.

The EXISTS, IS NULL, IS NOT NULL, ALL, and ANY conditions require knowledge of SQL, and will be covered in Query III.

AND/OR/NOT
AND, OR, and NOT are basic Boolean instructions that allow you to manipulate how your conditions behave.



AND
When the operator is **AND**, the row appears in the result data, if both search conditions are true.

Example:
Region = NW
AND State = WA
This will bring results back that match both criteria.

OR
When the operator is **OR**, the row appears in the result data, if either or both search conditions are true.
Example: You may want to find out some information on Northwest Region warehouses and the state of Georgia warehouses:
Region = NW
OR State = GA

This will bring back results for the Northwest Region and for the state of Georgia, whether or not both conditions appear in the same row of data.

The button for **AND/OR** is enabled when you have one or more conditions.

The button toggles between applying **AND**/**OR**, just by clicking it.

By highlighting the different conditions, you can apply the **AND**/**OR** operator to each condition. The default is set to **AND**.

NOT
Choose this button to add the **NOT** operator to the currently selected Conditions. The **NOT** operator will retrieve results if the search conditions are not true. Like the **AND/OR** button, this function is enabled when you select one or more conditions.

Using **AND**, **OR,** and **NOT** options in different orders will produce different results. As with all steps in Query, be sure to know your data and know what you want. Always verify your results.

Group/Ungroup
Columns
Grouping by a column in conjunction with calculation functions, such as sum, will create group summarizations. In other words, if you have a data result that shows sales by warehouse without a group, it may look like this:

WAREHOUSE	**SUM(SALES)**
WHS 110	584,000
WHS 64	320,000
WHS110	496,000

By grouping the Warehouse columns together, the results are summarized, or grouped differently.

WAREHOUSE	**SUM(SALES)**
WHS 110	1,080,000
WHS 64	320,000

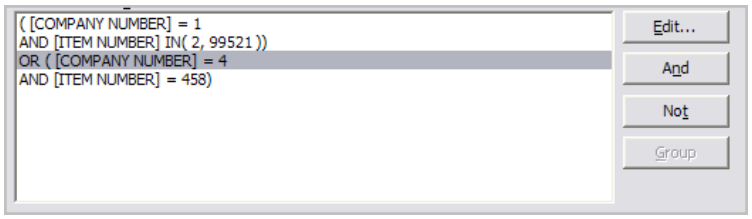
When joining large files together, Group By can cause poor performance. To avoid poor performance, create temporary table from the join, and then run a second query using the Group By function from the new table.

Conditions
Select more than one condition, and then click on the **Group/Ungroup** button to determine how you would like your conditions to be evaluated. Grouping places parentheses around the highlighted conditions and evaluates them in order, from the first condition to the last.
Grouping conditions allows you to add multiple conditions that may not otherwise work.

EXERCISE 16: GROUP COMPANY AND ITEM NUMBERS

A simple example is US Milk is item number 2 and Canada milk is item number 99521. Using IS IN will not work because although an item number may be associated with an item in one country it doesn't mean it will be associated to the same item in another country.

1. Create a new query using the library INPRDDTA and the table INITMMP.
2. Select the fields Company Number, Item number, Item description.
3. Set the condition Where Item number equals 2.
4. Run the query and view the results.
5. Add the condition where company number equals 1.
6. Press the shift key down and highlight the Item number and company condition.
7. With both conditions select click on the Group button.
8. Add the conditions where company equals 4 and item number equals 99521.
9. Press the shift key down and highlight the new Item number and company conditions.
10. With both conditions select click on the Group button.
11. Take a moment and notice the parenthesis around the two groups. Higlight the beginning of the second group and change the AND to OR.
12. Make sure your conditions look like the image below.



13. Run the query and view the results

Using Grouping with conditions is a an effective way to work with filter on categories and subcategories of values. How do you categorize information in your department? How can grouping your conditions help you turn several queries into one step?

LIKE

This operator searches for a specified string pattern. It behaves much like the "**=**" condition, however it also has a wildcard option, **"%"**, which can be used in the front, middle, or end of a search word. Wildcard has the ability to specify a search based on only a part of a column's data. For example, if you want to search on all the members who have a first name starting with "JOH", use the LIKE condition with the wildcard. In the values section, type *JOH%*.

The underscore "_" used with a wildcard will skip the first character or however many underscores you enter. For example if you were looking for data with ON in the second and third spot of a field and the names were CONNER, CONE and COFFEE, you could skip the first character by using an underscore. "_ON%" Adding more underscores will allow you to skip more leading characters.

EXERCISE 17: SEARCH LOCATIONS

Search for all location names beginning with WEST; remember to use the wildcard symbol %.

1. Open a new Query using the Query wizard; File, New, Query.
2. Using the library ACPRDDTA2 and the table GLWCTL.
3. Select the Name field.
4. Choose the condition “Is Like”.
5. Place a wildcard search in the Values field that will bring back all names that start with “WEST”.

EXERCISE 18: SEARCH FOR KIRKLAND SIGNATURE ITEMS

1. Open a new Query.
2. Using the library INPRDDTA and the table INITMMP, add the fields IMDES1 and IMDES2.
3. Choosing the Conditions “Is Like” in combination with Boolean search functions, find all of the Kirkland or KS products in Department 24. (Hint: try creating two groups of conditions for Department 24.)

Other wildcards:

It may be possible that you have to search for a “%” symbol or “_”. If this is the case, the wildcard searches below will assist you.

- Member Name LIKE “_ill%”. All values in the Member Name column when the second through fourth letters are ”ill”, such as Jill, Bill and William.

- What if what you’re looking for has an underscore in it, such as ILL_?
  Member Name LIKE “ILL+_” ESCAPE “+” this entry will bring all the values back that are “ILL_”. The “+” adds the “_” symbol, and the word Escape “+” tells Query not to look for the “+” sign; it is a function within the condition.

- If what you are looking for has a “%” symbol in the data, such as ILL%, the search would look like this: Member Name LIKE “ILL+%%" ESCAPE “+”. This will retrieve all values in the Member Name beginning with ”ILL%“ (e.g., ILL%SONS).

NOT LIKE

This is simply the inverse of the LIKE condition.
When to use IS LIKE or NOT like may not always obviously present itself. Working with Wine there was a problem with cases of wine showing up in the SKU counts. By placing a NOT LIKE %CASE% on the item description the query was able to exclude the cases of Wine and made the Query exact for SKU counts.

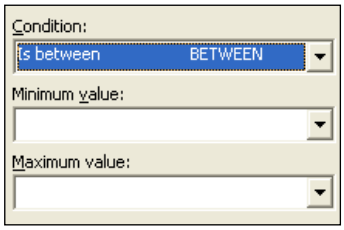
Value(s)

By selecting the black list box arrow, you will see a list of literal values for the column. This can be helpful when wondering the exact format of text for a condition, such as dates. The values entered into this box directly relate to the condition choice in the highlighted column box, and will therefore be different for each column.

Example

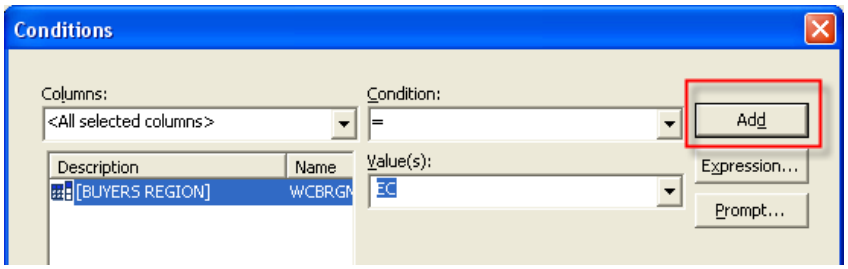
If the Region column is highlighted, the condition "=" is selected, and the value entered is NW, then your condition is "where Region equals NW."

If you enter data in the values box that does not exist in the highlighted column, no data will return. Entering a value of 110 to be a condition on the Region column will result in no records.



If you selected the **Is Between** condition from the Condition box, two boxes appear instead of just one. Select from the drop-down list, or enter a maximum and a minimum value for the **Is Between** condition.

Add

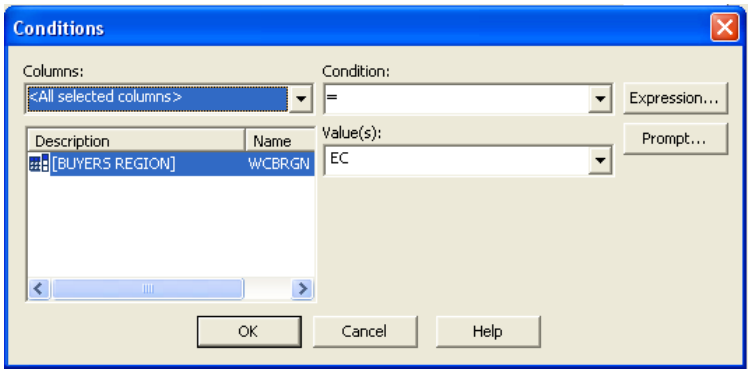


Choose this button to add the specified condition to the query.
When you choose this button, the new condition is added to the Added Conditions list.

Expression

Choose this button to create complex expressions.
When you choose this button, the Expression dialog box appears. (**Note:** Expressions are covered in Query III and IV).

Edit



The edit button opens the Edit Conditions dialog box. From here, you can change your conditions or edit Expressions or Prompts.

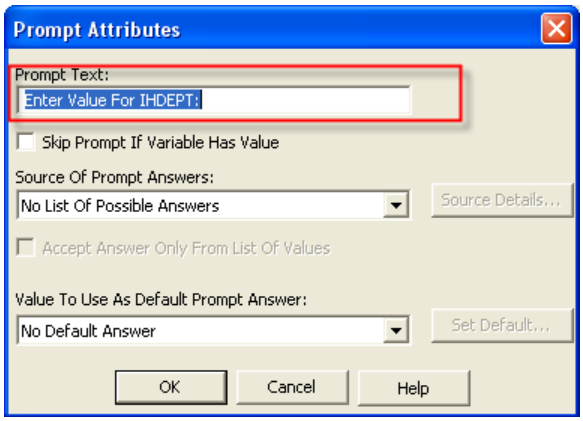
EXCEL AND PROMPTS

A **prompt** is a way of making conditions appear on the first screen when you run your query. Using different types of prompts, you can enter the desired condition, choose from a list, or have your prompt default to an answer. In this section, we will look at what prompt types are available, and how to configure them.

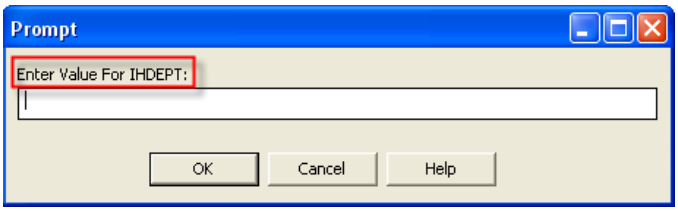
In the Prompt Attributes window, there are three distinct areas where you can modify or manipulate your prompt:

- Prompt Text
- Source of Prompt Answers
- Value to Use as Default Prompt Answer

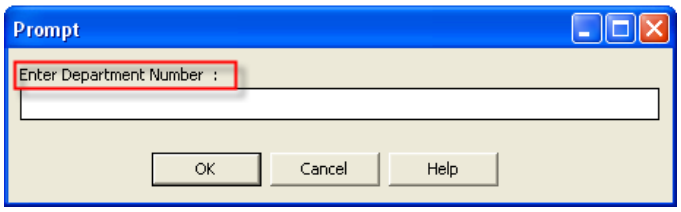
Prompt Text



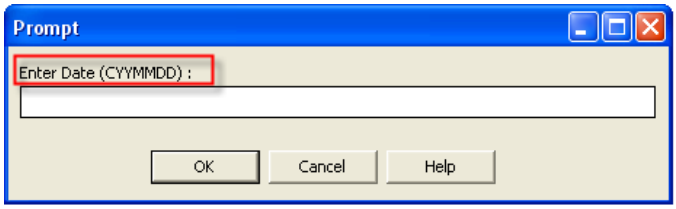
This allows you to type useful text information into your prompt, rather than the default value. If you ran the query with the above prompt text, it would look like this:



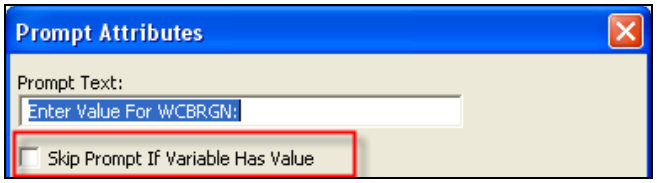
By entering new text in the prompt text box, we can change the message to be more meaningful.



This can come in very handy if you have special instructions for the prompt, or for giving an example of text or date format.

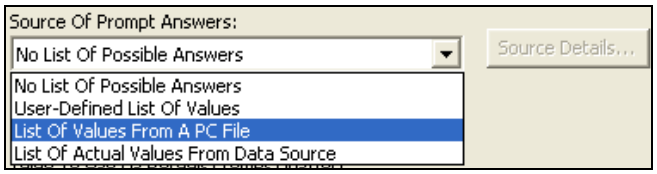


Under the prompt text dialogue box, there is a "**Skip Prompt If Variable Has Value**" check box.

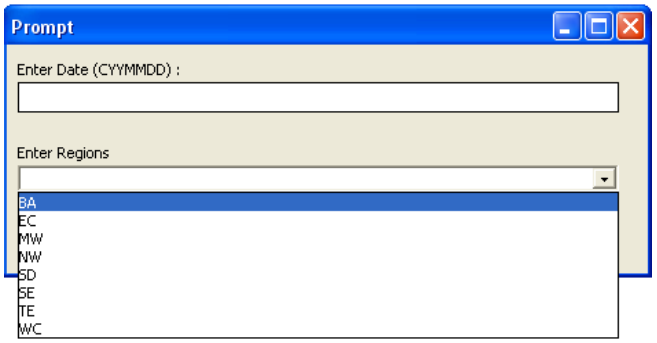


Check this box to skip the prompt if the user has already entered a value for the variable during a previous run (i.e., if you have already entered a value for the variable you want to use on a previous run of the query).

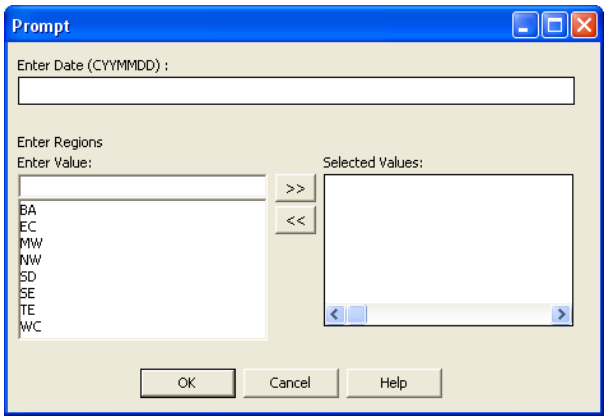
Source of Prompt Answers



This box contains a group of sources from which to retrieve a list of possible prompt answers at run time. When you run your query, rather than having to type a value, you can select the values you want.



The above example was created using an EQUALS condition. Notice what happens when we change the EQUALS condition to an IS IN condition:

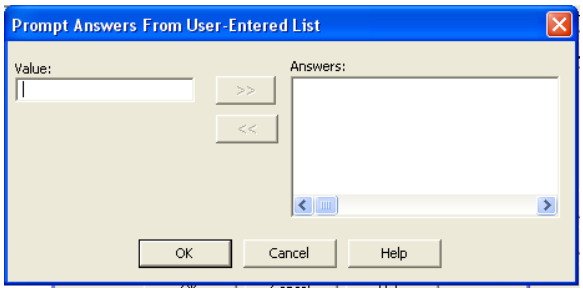


Now it's possible to choose multiple values already preloaded into the query.

Source of Prompt Answers options

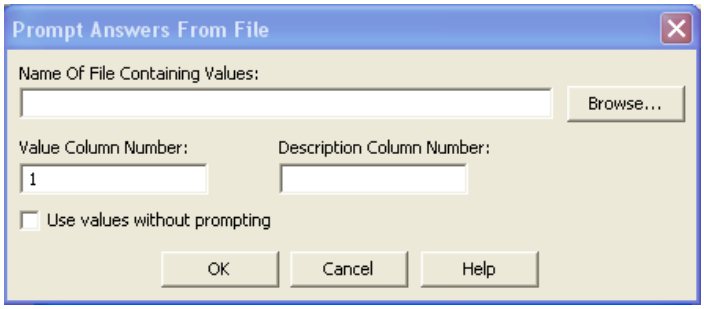
- **No List of Possible Answers**
  This is the default. A user of the query must enter a value at run time instead of selecting from a list of values.

- **User-Defined List Of Values**
  In this window, you can key in the values and add them to a list of possible answers window. When finished, select OK. Now when the query is run, values can be selected from the previously created list.



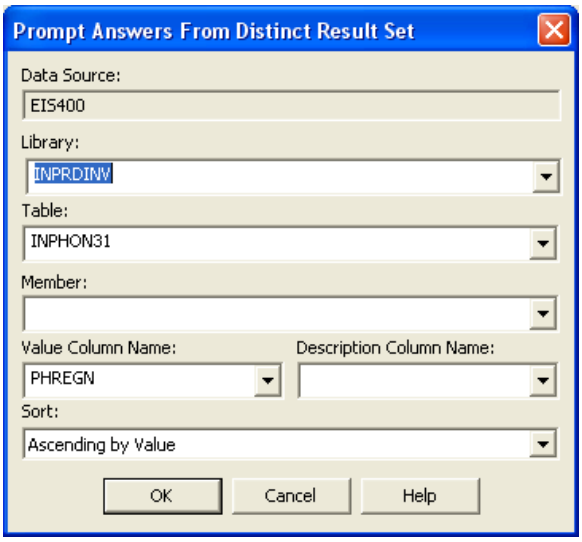
- **List Of Values From A PC File**
  By selecting the list of values from a PC file option, the Prompt Answers From File window opens, allowing you to browse to the PC file containing the values you want for your prompt. The file can be in one of the following formats:

- Microsoft Excel (XLS) bif4 – ANSI only
- Borland dBASE IV (DBF) – ANSI only
- Text (TXT) – ANSI and Unicode (UTF-16)
- Comma Separated Values (CSV) – ANSI and Unicode (UTF-16)



Updates to the chosen PC file will be recognized and incorporated; however, changes to the file name will create an error requiring you to reestablish the link to the file.

- **List Of Actual Values From Data Source**
  Selecting this option will open the Prompt Answers From Distinct Results Set window. In this window you can specify the Library, Table, Member, Description Column Name, and the Sort you would like your values to appear in.

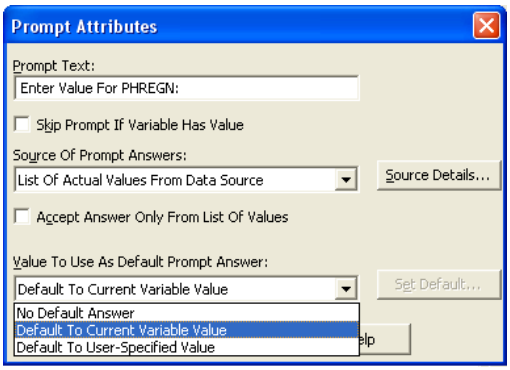


- **Source Details**
  When connecting to the EIS400 and selecting this button, the option behaves just as though you had chosen the List Of Actual Values From Data Source option.

- **Accept Answers Only From List Of Values**
  By clicking on this checkbox, it forces, the user to only enter values from the list of values provided.

- **Value To Use As Default Prompt Answer**
  Leaving this set to the default will make it so your prompt is empty when you run the Query. You can set this to default to current variable value or to a user specified value.

Setting the default to the current variable value will put the last entry used into your prompt. For instance if you had a list of regions and selected only BA when you opened the query to run again, BA would be the default in your prompt.



- **Default to User-Specified Value**
  The Default User Specified Value option behaves differently in that you can choose a default value. When the Query is run the default value will already be loaded into the prompt. After choosing the default, by using the Set default button.

EXERCISE 19: PROMPTS

In this exercise you will create a prompt from a PC file. Remember to close out of Excel before running the query.

1. Open Excel.
2. In column A, row 1 type: 110.
3. In column A, row 2 type: 64.
4. Save the Excel spreadsheet to your desktop.
5. Create a new query.
6. Add table INIHSTP from library INPRDDTA.
7. Add Item Number, Warehouse Number, YTD Sales In Units and YTD Sales in Dollars.
8. In the Conditions window highlight Warehouse Number.
9. Set the condition to “Is In.”
10. Click on the Prompt button.
11. In the Prompt Text field, type “Enter Warehouse Number”.
12. In the Source of Prompt Answers field choose “List of Values from a PC file.”
13. Click on the Source Detail button.
14. Click on Browse.
15. Navigate to the spreadsheet you created on your desktop.
16. Click Open in the Browse window.
17. Click OK in the Prompt Answers From File Window.
18. Check the ”Accept answers only from a list of values” box.
19. Click OK in the Prompt Attributes Window.
20. Click OK in the Query Properties Window.
21. Add other conditions for Fiscal year to = YY(current Fiscal year) Dept = 17 AND Item number to = 2.
    The fields for Fiscal year, Dept and Item number will not be displayed if “all selected columns” is left in the conditions window. To change this, click on the drop-down arrow and choose the table.
22. Run the query. You should see a prompt with Warehouse 110 and 64.
23. Choose both warehouses. Be sure to click on the arrow to move the values into the selected box.
24. Click OK.
25. Retrieve your results.

Try editing this query to automatically load any values in the condition without the prompt pop-up window.

What would be better than Excel to use for loading values into a prompt?

Questions for Review:

What symbol would you use in an IS LIKE condition to represent the unknown?

What Boolean search would you use to include results?

Which Boolean search would you use to exclude results?

Can you load values into a prompt from a PC file?

UNIT V: JOINS

What is a Join?
Joins in Showcase Query
Join Types
Using Conditions After a Join

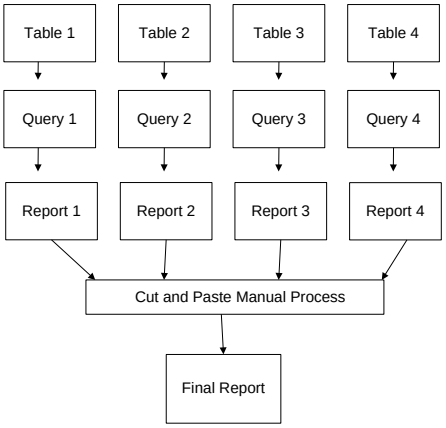
JOINS

Joins can be one of the more complicated concepts to understand in the realm of Query or Structured Query Language (SQL). One analogy to help understand joins is that of a bridge connecting one island to another. The bridge joins the islands together. After a review of joins, you will see that this analogy, although helpful, leaves something to be desired. Rather than trying to simplify the join concept, it is better to just get right into it. This chapter will look at what a join is, what the different types of joins are, when to use joins, how to use joins, and examples of joins.

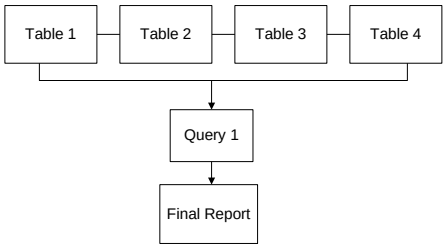
What is a Join?

Sometimes you have to select data from two or more tables to obtain the result you want. One of the most powerful features of SQL is the ability to use data across several tables to derive with one result. Without this capability, you would have to run several reports, and then combine those reports manually to arrive at the one result you were looking for.

Without Joins



With Joins



To retrieve joined data from two or more tables, the tables first must share a meaningful relationship and be linked together. This is a join.

EXERCISE 20: MATCHING GAME

Link the INVT1001P table to the table with the meaningful relationship.

INVT1001P
Last 4 Acct#
Tender Amount
Business Type
Sales Date
Y=Foreign Exists
Membership#
Reg#
Tender Type
Warehouse Number

GLTRPJP
GL Account Number
GL Sub-Acct Number
Adjustment Number
Batch Date As YYMMDD
Batch Time as HHMMSS
Check #
Check Date As YYMMDD
Comment

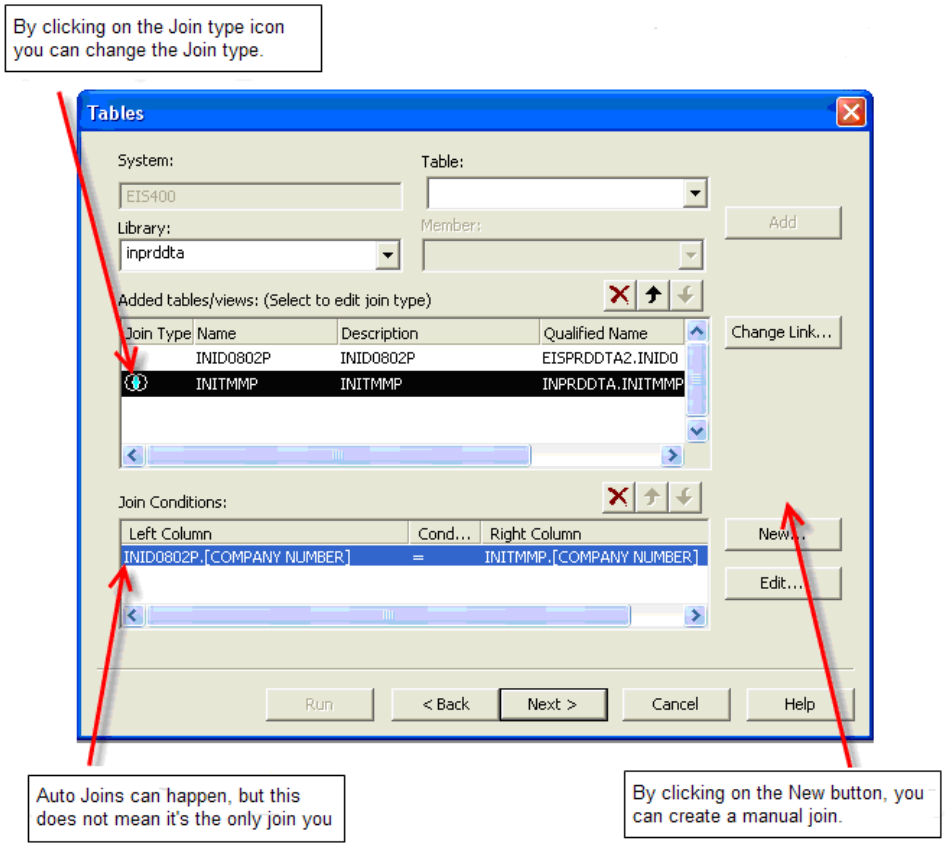
MEMSRLP
Account IPK
Enrollment Date
Enrollment Facility
File Code
Inactive Date
Linked File Code
Linked-entity IPK'
Membership Fee Flag

INWCTLP
Address Line 1
Address Line 2
Area Code
RA Print Start Date
City
Division
Warehouse Number

FAPYB
Internal Accounting Loc#
Year End Accum. Reserve
Year end Cost Basis
Book Name
Company
Internal Physical Loc#

Creating Joins in the Showcase Query application

Let's look at the steps for creating a join in the Showcase Query application. In the **Tables dialog box**, you may accept any default joins displayed in the **Join Conditions box** by clicking **OK** or **next** or selecting the **Columns dialog box**. However, to create the most efficient results, you may need to add multiple joins.



To change the join type, click the **Join Type** symbol in the **Join Type/Name** column **under Added Tables/Views** on the **Tables dialog window**.

From the dropdown list, click on the join type to assign the specific join you want to the two tables.

Join types available: Inner Join, Left Outer Join, Right Outer Join, Exception Join, Right Exception Join, and Cross Join. These join types will be covered in detail later in the chapter.

Tips:

Specifying a Left Outer Join is the equivalent of selecting Asking for Unmatched Rows.

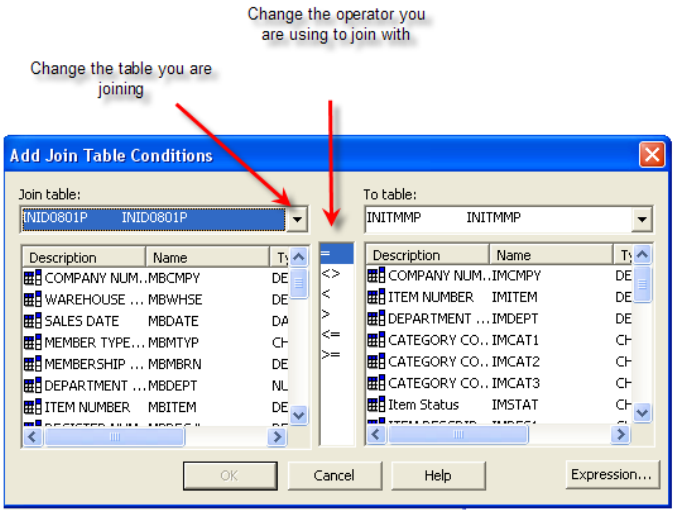
You will improve server performance and run faster queries when joining tables by specifying the table with the most rows as the left or the top table in a join condition.

By default, Automatic Smart Join is enabled in Query. To disable this function, on the Tools menu, click Options and then click the Display tab. Deselect the Enable Automatic Smart Join option.

To sort the information in the list boxes of the Tables dialog box, click on the column heading by which you want to sort your results.

Join Operators

The **JOIN** operator matches rows by comparing values in one table with values in another, via an arithmetic function such as equals or less than.



You can decide exactly what constitutes a match. Your choices fall into two broad categories:

- **Match on Equality**

  Typically, you match rows when the respective column values are equal. For example, to create a result set in which each row contains a full description of each **Item** (that is, with columns from the **Item Sales** table and the **Item Description** table), you use a join, matching rows where the values of **Item Sales** in the respective tables are equal.

- **Other**

  You can match rows using some test other than equality. For example, to find the employees and the jobs for which they are under-qualified, you can join employees with jobs, matching rows in which the job's minimum required level exceeds the employee's job level.

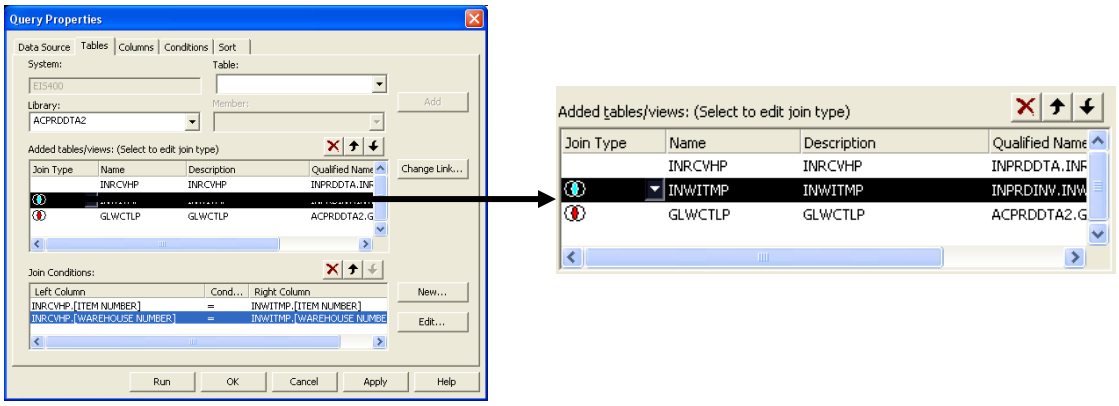
LEFT AND RIGHT TABLES

When adding tables to your query, they will read from top to bottom. Later you will see that the tables in the Wizard that are on top are known as the “left tables.” The table joined below the top table is the “right table.” Top to bottom, left to right.

These tables can also be referred to as primary and secondary tables. The Primary table is on top; the secondary is below the primary.

If you have three tables, One being the first or top table, Two being the second and Three being the third, the rule still applies that you will only have one left table and one right table at any given time. If you join the first table to the second table, the first is the left or primary table and the second is the right or secondary table.

If you join the second table to the third table the same rule applies. The second table is now the left/primary table, and the third table is the right/secondary.



Challenge

There is a join between the first table and the third table. Which one is the "right" table?

KEYS

Finding meaningful relationships is the first step in understanding joins. This is done by using keys. Before talking about joins, there should be an understanding of keys. There are primary keys and foreign keys. Keys form a meaningful relationship between two or more tables. Another way to understand keys is to look at them as links.

PRIMARY KEYS

Tables in a database can be related to each other with keys. A primary key is a column with a unique value for each row. The purpose is to bind data together, across tables, without repeating all of the data in every table. All tables should have at least one primary key.

The primary key can either be a normal attribute that is guaranteed to be unique (such as a member number in a table with no more than one record per member), or it can be generated by the database (such as a globally unique identifier, most commonly a sequential number).

Imagine a membership database that contains three tables. The first table, **MEMBERS**, contains a record for each member at Costco Wholesale. The second table, **ADDRESS**, contains a record for each member and their address. The third table, **SHOP HISTORY**, contains member numbers and their shop history (i.e., each record represents a member).

MEMBERS table

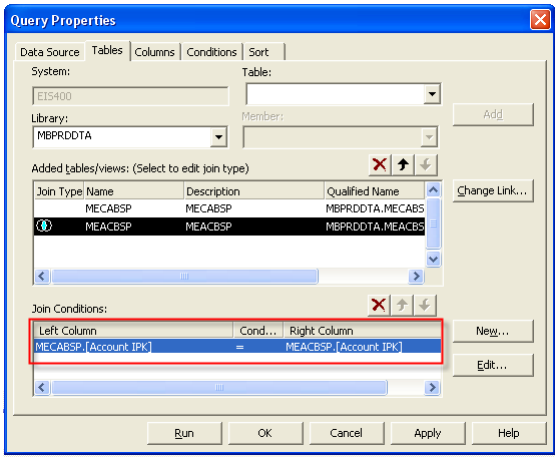
Column Name	Characteristic
Member Number	Primary
First Name	
Last Name	

ADDRESS table

Column Name	Characteristic
Member Number	Primary
Address	
City	

SHOP HISTORY table

Column Name	Characteristic
Member Number	Primary
Item number	
Sales $	



There would be multiple records for each member (possibly representing multiple addresses where they have lived) and multiple records for each shop history (representing all the member shops). A member's unique ID number would be a good choice for a primary key in the **MEMBERS** table. The member's first and last name would not be a good choice, as there is always the chance that more than one member might have the same last name.

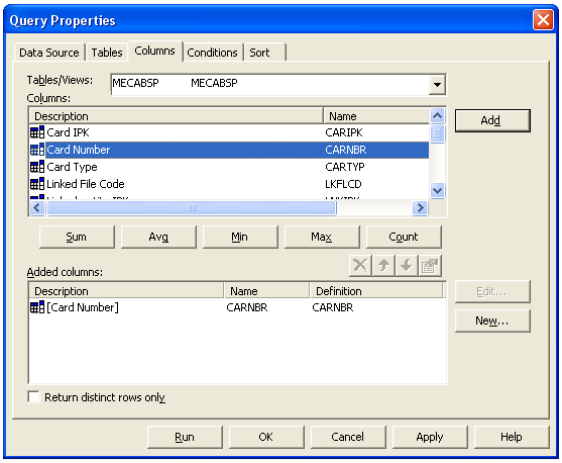
EXERCISE 21: SMART JOIN ACCOUNT EXPIRATION

Create a query that retrieves memberships and account expiration dates.

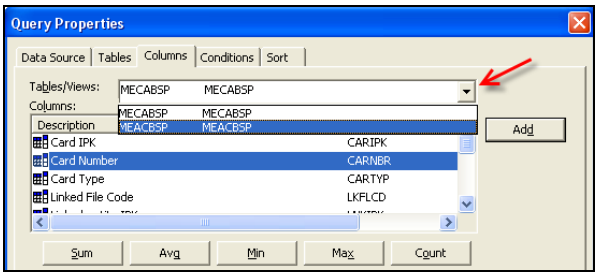
1. Open Query Wizard and add **MECABSP** from the library **MBPRDDTA**. This is a membership relationship file.
2. Next, add **MEACBSP** from the library **MBPRDDTA**. This is the Account Basis file.

Notice when you add **MEACBSP** that it creates an auto join on **Account IPK**. This is the primary key. It has been automatically set up for you.

3. Click on Next.
   From **MECABSP** add the field **Card Number.**



4. Click on the Table/Views list box to select **MEACBSP.**



5. From **MEACBSP** add **the Account Expiration Date** field.
6. Click on Next.
7. In the conditions editor place a condition on **Card number** to equal your member number or number 99.
8. Run the Query and retrieve the results.
9. **Save this Query to your desktop; it will be used in a later exercise**.

Primary keys may consist of a single attribute or multiple attributes in combination. For instance, you may have the simple example of member number being the obvious key. However, some tables have fields like Company, Region, State, and Warehouse. To create a key you would need to use a combination of fields. Some of the fields are known as foreign keys.

FOREIGN KEYS
A foreign key is a field or fields that point to the primary key of another table. These keys are used to create relationships between tables.

Returning to the Members database, let's imagine that we wanted to add a table containing item detail information. This new table might be called **ITEM DETAIL** and would contain information about item sales for a warehouse by day. Let's say that you would also like to include information about the members and their first and last names, but it would be redundant to have the same information in two tables (first and last name in the **MEMBERS** table and in the **ITEM DETAIL** table). Instead, we can create a relationship between the two tables.

Let's assume that the **ITEM DETAIL** table uses Items as the primary key. To create a relationship between the two tables, we add a new column to the **ITEM DETAIL** table called member number. Member number in the **ITEM DETAIL** table is a foreign key.

ITEM DETAIL table

Column Name	Characteristic
Item Number	Primary
Warehouse number	
Member Number	Foreign key
Sales in units	

MEMBERS table

Column Name	Characteristic
Member Number	Primary
First Name	
Last Name	

Note that there is no uniqueness for a foreign key. There might be more than one member belonging to a single item. Similarly, there's no requirement that an entry in the **MEMBERS** table have any corresponding entry in the **ITEM DETAIL** table. It is possible that we'd have an item with no sales and a member with no item purchases.)

There are several choices in deciding which columns from each table should be matched. For example, you can join warehouses to vendors by matching the values of state in the respective tables. Such a join yields a result set in which each row describes vendors and warehouses located in the same state.

Note also that you can use multiple columns to match rows from the joined tables. For example, to find vendor-to-warehouse pairs in which the vendor and warehouse are located in the same city, use a join operation matching the respective state columns and the respective city columns of the two tables. You need to match both city and state because it is possible that different states could have like-named cities (e.g., Portland, Oregon and Portland, Maine).

EXERCISE 22: DAILY SALES BY REGION BY UPC

1. Create a query using library EISPRDDTA2 and the most current INID table. For example INIDYYPDP, where YY equals Fiscal year and PD equals Period.
2. Create a join on company number with the library INPRDINV and the table INWCTLP.
3. Add the fields Item Number, Daily Sales in Units, and Sell UPC.
4. Sum Daily Sales in Units.
5. Add the condition Company =1.
6. Add two prompts, an "=" prompt on Item number, and an "is between" prompt for the date.
7. Run the query using an item number and a date range, and view the results.
8. Write down the number of records.
9. Remove the join on company and create a join using warehouse number.
10. Add a condition for Region = NW.
11. Run the query using the same item number and a date range, and view the results.
12. Notice the number of records.
13. Why are the results different?

Joining Tables on Multiple Columns

You can join tables with multiple matching columns. That is, you can create a query that matches rows from the two tables only if they satisfy the correct conditions for the query that is being built. If the data-base contains a relationship matching multiple foreign-key columns in one table to a multi-column foreign-keys in the other table, you can use this relationship to create a multi-column join to essentially force a primary key with one or more foreign key joins.

When you are dealing with multiple foreign-keys, the objective is to join the tables in such a way that you do not create duplicate records. You may need to join many foreign-keys to get the right results. In the example below there is a Checkbook table and a Bank statement table

The question you are asking is what transactions have cleared my bank account.

Check Number being the primary key its very easy to have a one to one relationship between the two tables and verify what has cleared and what has not cleared.

Checkbook table

Check #	Date	Description	Debit	Credit	Balance
1000	9/1/2011	Open Account		1000	1000
1001	9/1/2011	Le Frock	97.82		902.18
1002	9/1/2011	22 doors	23.95		878.23
1003	9/10/2011	Payday		1500	2378.23
1004	9/10/2011	Macy's	98.72		2279.51
1005	9/10/2011	Le Frock	22.34		2257.17
1006	9/10/2011	22 doors	79.16		2178.01
1007	9/10/2011	PSE	100		2078.01
1008	9/10/2011	Rent	1200		878.01
	9/10/2011	ATM	200		678.01
1010	9/10/2011	Mammas pizza	15.67		662.34
1011	9/10/2011	Cab	7		655.34

Bank Statement table

Check #	Date	Description	Debit	Credit	Balance
1000	9/4/2011	Open Account		1000	1000
1001	9/4/2011	Le Frock	97.82		902.18
1002	9/4/2011	22 doors	23.95		878.23
1003	9/10/2011	Payday		1500	2378.23
1004	9/15/2011	Macy's	98.72		2279.51
1006	9/14/2011	22 doors	79.16		2200.35
1007	9/14/2011	PSE	100		2100.35
1008	9/15/2011	Rent	1200		900.35
1009	9/10/2011	ATM	200		700.35
1010	9/15/2011	Mammas pizza	15.67		684.68

Not all tables have primary keys. Lets look at the Checkbook table and the Bank statement table again without the Primary Key.

Checkbook table

Date	Description	Debit	Credit	Balance
9/1/2011	Open Account		1000	1000
9/1/2011	Le Frock	97.82		902.18
9/1/2011	22 doors	23.95		878.23
9/10/2011	Payday		1500	2378.23
9/10/2011	Macy's	98.72		2279.51
9/10/2011	Le Frock	22.34		2257.17
9/10/2011	22 doors	79.16		2178.01
9/10/2011	PSE	100		2078.01
9/10/2011	Rent	1200		878.01
9/10/2011	ATM	200		678.01
9/10/2011	Mammas pizza	15.67		662.34
9/10/2011	Cab	7		655.34

Bank Statement table

Date	Description	Debit	Credit	Balance
9/4/2011	Open Account		1000	1000
9/4/2011	Le Frock	97.82		902.18
9/4/2011	22 doors	23.95		878.23
9/10/2011	Payday		1500	2378.23
9/15/2011	Macy's	98.72		2279.51
9/14/2011	22 doors	79.16		2200.35
9/14/2011	PSE	100		2100.35
9/15/2011	Rent	1200		900.35
9/10/2011	ATM	200		700.35
9/15/2011	Mammas pizza	15.67		684.68

Joining on dates is not enough. What two fields would you need to join on to answer the question what items have cleared the account? How would joining this incorrectly effect the results returned?

EXERCISE 23: TOP 20 US ITEMS FOR THE YEAR

Understanding the business and being familiar with the data in the tables helps with determining how the tables should join together. The phrasing in the question or conditions can be clues as to what fields may be the join.

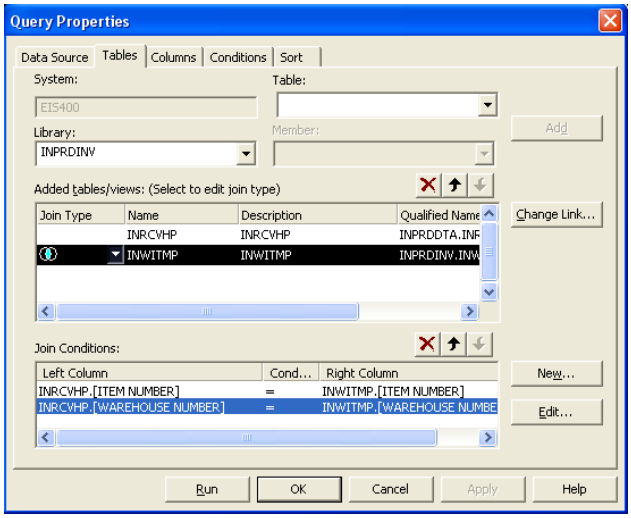
1. Create a query using table INIHSTP in the library INPRDDTA and table INITMMP in the library INPRDDTA.
2. When both tables are added, a smart join is created on Item Number. Is this correct? If so why? If not what is the correct join?
3. From the table INIHSTP add the fields Department Number, Item Number, YTD Sales in Dollars, and YTD Sales in Units. From the INITMMP table add the field Item Description.
4. Place a sum on the sales and units fields.
5. Place a condition on the fiscal year to equal the current fiscal year, and on the company to equal 1.
6. Sort Descending on YTD Sales in Dollars.
7. In the Run menu, limit the rows to 20.
8. Run the query and view the results.

EXERCISE 24: DEPT 45 CATEGORY CHECK FOR COMPANY 4

1. Create a query using library EISPRDDTA2 and the most current INID table. Example INID1001P would equal Fiscal Year 10 and Period 01.
2. Add the Table INITMMP from the library INPRDDTA.

   When both tables are added a smart join is created on Company Number. This is correct. Why?

3. Another join has to be created. Choose the field you think needs to be joined and continue to the next step.
4. From the INID table add the fields
   a. Warehouse Number
   b. Sales Date
   c. Department Number
   d. Daily Sales in Units
   e. Daily Sales at Sell
5. From the INITMMP table add the fields Item Number and Item Description.
6. Add the conditions Company = 4, Department Number = 45 Category Code = A and Sales Date <= the Current date.
7. Sort Ascending on Warehouse Number and Sales Date.
8. Run the query and view the results.



EXERCISE 25: MULTIPLE FIELD JOIN ITEM SALES AND DESCRIPTIONS

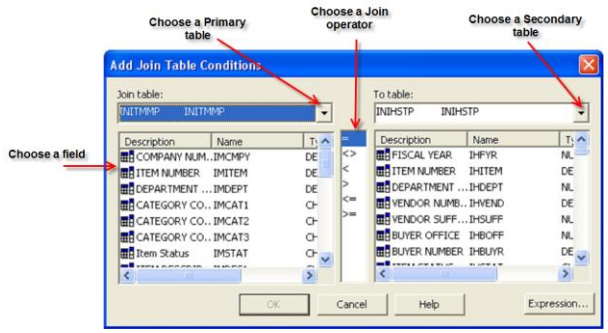
Create a primary key using multiple fields.
In this exercise, create a query that gives item sales and item description information. In order to do this, join the **INRCVHP** file with the **INWITMP** file.

1. Add the most current member from the table INRCVHP and the INPRDDTA library. Click on the Member drop-down arrow and select the most current member. Members are divided up by fiscal year and Period. For Example member MYYYYPP represents M = the beginning of the member name, YYYY represents the current fiscal year and PD represents the period.

2. Add the table INWITMP from INPRDINV.

As with the previous exercise, an auto join has been created. However, to make this query work, you will need to join multiple columns to the primary key desired.

3. Click on the New button. This will allow you to create new joins.



4. Verify that the first join table is INRCVHP and the second join table is INWITMP.
5. Highlight the ITEM NUMBER field in both tables and click OK to ensure there is a relationship between ITEM NUMBERS in both tables.
6. In this query you will want to know shipping information about warehouses, so create a join on the WAREHOUSE field in both tables.
   (**Note:** INRCVHP has three warehouse fields. For the purposes of this exercise, use RHSHW5. If you were to run into this on your own, the solution would be to open up INRCVHP and run a simple query on the three warehouse fields to become familiar with the data that exists in these fields.)
7. Add the fields
   a. WAREHOUSE NUMBER
   b. ITEM NUMBER
   c. PO NO-WHSE NO PRFX WHEN PRT
   d. WHSE RECEIVING LOG LINE NO
   e. DEPARTMENT NUMBER
   f. VENDOR NUMBER. QTY ORDER IN SELL UNITS
   g. QTY RECEIVED IN SELL UNITS
   h. NET LANDED COST/SELLING UNIT
   i. From the table INRCVHP.
8. Add the field SELL PRICE TO WHOLESALER from the table INWITMP.
9. Click on the Return distinct rows only button.
10. Add the conditions WAREHOUSE =110, DEPARTMENT = 17, and ITEM NUMBER=2 from the INRCVHP table.

11. Run the query and view the results.

There are primary keys, foreign keys, and primary keys created by using multiple fields. Which one do you use? It depends on your query. Any one of the keys could give you a correct answer or an incorrect answer. The first step in creating a join is to know what you are looking for. The second step is to know your data. Open the tables and look at the data in the tables. Some tables have logical structures that key tables in an order that will work better for your join. Becoming familiar with the data and reviewing how the tables are keyed is an important first step.

Simplifying the Join Process

The best way to proceed with joins is to find the lowest common relationship in the two tables you are trying to join that relate to your question. Let's say you were trying to join the two tables below and your question was about item sales.

Table 1

Company
Region
Item Number
Item Description
Department
Category Code

Table 2

Company
Region
Department
Item Number
Item Sales in Units
Item Sales in Dollars

The join to start with is Item number to Item number. Depending on your tables this simple join may be enough. Run a query with this join and view your data for accuracy. If this join is not ideal, add another join with the next logical step above your first join. In this example it would be Department number. Run the Query again to test for accuracy. Continue adding new joins until the data matches results you would expect.

Question: Couldn't I save a lot of time just by joining all the common fields?

Answer: Although joining several common fields can create a primary key join, it can slow performance and in some cases give you inaccurate data. Take a look again at the Item Sales and Item Description tables.

Table 1

Company
Region
Item Number
Item Description
Department
Category Code

Table 2

Company
Region
Department
Item Number
Item Sales in Units
Item Sales in Dollars

It would seem a join could be made on Company, Region, and Department. However, if the Item Description Table has only Company 1 information, and the Item Sales information has Company 1 and 4, a join on Company will filter out Company 4

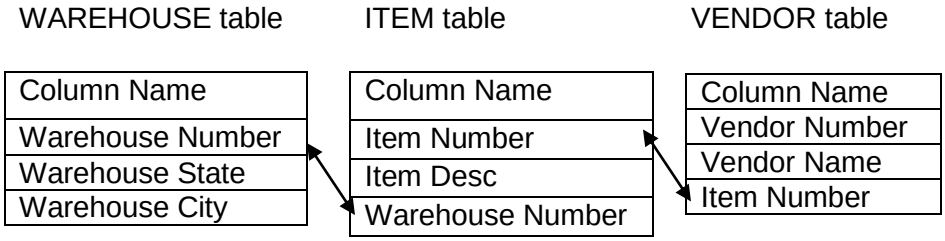
information. This is not a problem if the intent is to filter out Company 4 information, but if not, your data will be inaccurate. The key is to be familiar with the individual tables.

Combining three or more tables

Each join combines two tables. However, you can use multiple joins within one query to assemble data from any number of tables.

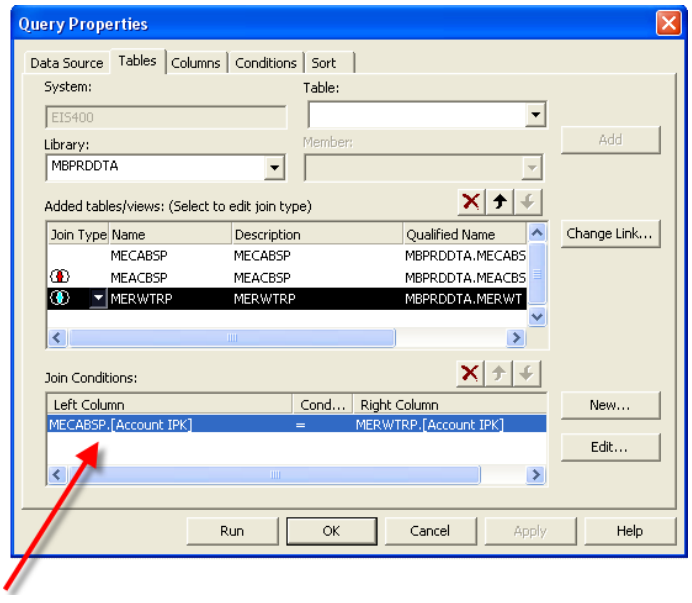
Example: There are three tables: WAREHOUSE, ITEM, and VENDOR.

To create a result set in which each row contains a warehouse phone number, item and vendor, you must combine data from three tables: Warehouse, Item, and Vendor. The resulting join might look like this:



You can include a table in a join even if you do not want to include any of that table's columns in a result set. For example, to establish a result set in which each row describes an item and location pair in which that store sells that item you include columns from two tables: Items and Locations. However, you must use a third table, Sales, to determine which locations have sold which items.

Going back to the island and bridge analogy, you would build a bridge from island #1 to island #2 so that you could then build a bridge from island #2 to island #3 and share the resources from island #1 and #3 without ever using anything from island #2 (other than the bridge).



EXERCISE 26: MEMBERSHIP RENEWALS

In this exercise, **Date Renewed** and the **Renewal Amount** will be added to the previous query. Creating a multiple table join.

1. Open the query you created in **Exercise 25.**
2. In the Tables editor add **MERWTRP**; this is the **Renewal Track File**.

Notice that when **MERWTRP** is added, another auto join is created for you.

3. Click on the **Columns** tab.

4. Click on the table view list box to select MERWTRP.
5. From **MERWTRP** add the Date Renewed and Renewal Amount to the query.
6. Run the query and retrieve the results.

EXERCISE 27: OPEN PO BY WEEK

1. Create a query using library INPRDINV and the table INPMSTP.
2. Add the table INWCTLP from the library INPRDINV.
3. Create a join on Company Number and Warehouse Number.
4. From INPMSTP add the fields Department Number, SUM on Total Sell of Order, Warehouse Number, PO No-Whse No PRFX WHEN PRT.
5. Add conditions from INPMSTP where Department = 17.
6. Add conditions from INPRDINV Where Region = NW.
7. Sort ascending on Department.
8. Run the query and view the results.
9. Write down the number of records.
10. Add the table UTCLDRP from the library PRDGPL.
11. Find the join that will work.

EXERCISE 28: KIRKLAND SIGNATURE SALES

1. Create a query using library INPRDINV and the table INWITMP.
2. Add the fields Warehouse Number, Department Number, Item Number, Private Label.
3. Add the Conditions Company = 1, Department = 17 and Private Label = k.
4. Add the table INIHSTP from the library INPRDDTA.
5. Add the field Period 1 Sales in Dollars, and sum the field.
6. Add the condition Fiscal Year = Current Fiscal Year.

   What two joins should exist between INWITMP and INIHSTP? Keep in mind that INIHSTP is added for items sales information, and contains location information.

7. Run the query and view the results.
8. Add the table INWCTLP from the library INPRDINV.

   What one join should exist?

9. Add the conditions Region = NW and State = WA.
10. Run the query and view the results.
11. Add the table INITMMP from the library INPRDDTA.
12. This table will have two joins; one join with INWCTLP and another join with INWITMP. Find and create the correct joins.
13. Add the field Item Description or IMDESC1.
14. Run the query and view the results.

The auto join in the above exercises is convenient, however; it is not always, what is needed to make the most efficient join.

Using a Table Twice in One Query

You can use the same table two (or more) times within a single query. There are several situations in which you would do this, for example:

- **Creating a self-join with a reflexive relationship**
  You can join a table to itself using a reflexive relationship; a relationship in which the referring foreign key columns and the referred-to primary key columns are in the same table. For example, suppose the **Employee** table contains an additional column, **Employee Manager**, and that a foreign key exists from manager to employee. Within each row of the **Employee** table, the **Manager** column indicates the employee's boss. More precisely, it indicates the employee's boss ID.
  By joining the table to itself using this reflexive relationship, you can establish a result set in which each row contains a boss's name and the name of one of the boss's employees. This would be the same as putting a condition on the **Manager** field to equal a particular manager or managers, to see the employees working in their groups. However, you may not know the manager's particular name or names, and therefore you wouldn't be able to use a condition; you would use a reflexive join.

- **Creating a self-join without a reflexive relationship**
  You can join a table to itself without using a reflexive relationship. For example, you can establish a result set, and then join to that result set. Often this is done to create filters or relationships within larger databases.

Example 1
You may want to create a result set or work file with just warehouses and item numbers, and then use that same work file to join to the table you used to create the file. Now you can run several different queries using the warehouse item file as filter for item sales, or any other date you would be looking for.
Example 2
You can establish a result set containing the other items by the vendor or vendors and items. In this case, you use the vendor items twice; once to find the vendors of a particular item, and once to find items by the vendor. In other words, a two-step process:

- Create the results
- Use the results in your next query

However, this can also be done just as effectively and more efficiently with a condition, depending on the size of the database.

Types of Joins

When you join tables, the type of join that you create affects the rows that appear in the result set. You can create the following types of joins:

- Inner Join
  An inner join displays only the rows that have a match in both joined tables. For example, if you have a table of book titles and a table of publishers, you can join the two tables to create a result set that shows the publisher name for each title. In an inner join, titles for which you do not have publisher information are not included in the result set, nor are publishers with no titles. An Inner Join, also called a Direct Join, combines data from tables that have a column of information in common.

Left	Right
Barry	WA
Shawn	ID
Kelly	NY

- Left-Outer Join
  A left-outer Join returns data two tables have in common *and* includes data from the primary, or first table selected, which does not have matching data to join to in the secondary table. Selecting this join type is the equivalent of selecting the Return Unmatched Rows option on the Join dialog box in previous versions of this application.

Left	Right
Jason	-
Shawn	ID
Wendy	-

- Right-Outer Join
  A right-outer join returns data two tables have in common *and* includes data from the secondary, or second table selected, which does not have matching data to join to in the primary table.

Left	Right
Barry	WA
-	ID
Kelly	NY

- Exception Join
  An exception join returns only the data from the primary, or first table selected, which does not have matching data to join to in the secondary table.

Left	Right
Kelly	-
Brad	-
David	-

- Right-Exception Join
  A right-exception join returns only the data from the secondary, or second table selected, which does not have matching data to join to in the primary table.

Left	Right
-	WA
-	ID
-	NY

- Cross Join
  A cross join matches every row of the primary table with every row of the secondary table. This type of join results in a Cartesian product of the tables, is generally detrimental to slow performance, and is not desired.

Left	Right
-	WA
-	-
Brad	NY

EXERCISE 29: JOIN FILE TYPES FILL IN THE BLANK

Use random names and area codes to represent the different types of joins.

Right Exception Join

Left	Right

Left Outer Join

Left	Right

Exception Join

Left	Right

Right Outer Join

Left	Right

Cross Join

Left	Right

Inner Join

Left	Right

EXERCISE 30: JOIN TYPES

What type of join should you use? For this exercise, Table 1 represents the Primary table and Table 2 represents the Secondary table.
Table 1 = Primary = Left
Table 2 = Secondary = Right

1. Table 1 has pending deleted items. Table 2 has item sales. You need to find all items that are pending deleted that have no sales data.

2. Table 1 has a list of vendors. Table 2 has a list of addresses for vendors. Table 2 is missing some data. For your purposes, you want to see all of the vendor information. You also want to see all of the address information you can find, even if some of the address information is missing.

3. Table 1 has a list of members. Table 2 has a list of items purchased. You are running a ground beef recall, and need to get an exact match of members that have purchased this item.

4. Table 1 has item information. Table 2 has item sales information. You want to see all of the data in both tables, regardless of a match.

5. Table 1 has a list of item descriptions. Table 2 has a list of item sales. Table 1 is missing some data. For your purposes, you want to see all of the item sales information. You also want to see all of the item description information you can find, even if some item descriptions are missing.

6. Table 1 has item sales. Table 2 has pending deleted items. You need to find all items that are pending deleted that have no sales data.

How to Set Up Conditions After a Join

Tables should be set up so the larger table is on the left and the smaller table is on the right, to list the greatest filter first.

Example:
If you have a query with an item description file and an item sales file, your primary table would be the item sales file. You would then join the item description file (secondary) to the item sales table (primary). Likewise, your conditions should go in order of the first table to the last table, creating the most efficient filter possible.

It is not obvious how much data a table holds. For example, Item Sales will of course store more date than a table only holding Item Descriptions. The reason why the Item Sales should be your first or primary table is because you want to filter it with conditions to make it a smaller set of data.

Look again at the example of the Item Description file and the Item Sales file. If you have joined the tables, you can place conditions on the item sales file of Company and Warehouse. Next you can put conditions on the Item Description Table of item numbers, department numbers, and category codes.

Each Query is different. The key is to try to put the greatest filter of data first. Ask the question; “What will make this set of data smaller?”

TIPS on keeping joins simple:

1. Run basic Queries over all tables being joined and become familiar with the data results. Understand the data you are going to join.
2. Determine your primary table and build a simple query over that table. Test the data and ensure it is built correctly this becomes your test data that your joins should synch up with.
3. Methodically test the different join relationships until you have created a synch with your test data.
4. Review your results of the join to make sure that not only you have a synch with your test data but the results are accurate.
5. The Question being asked will often hold clues as to what fields would be the right join.
6. Conditions often times hold clues to what the best join would be.

Overall gather requirements become familiar with your data and have a simple query built over your primary table for testing your joins.

Questions for Review:

What is a join?

Which represents a unique record, a primary or foreign key?

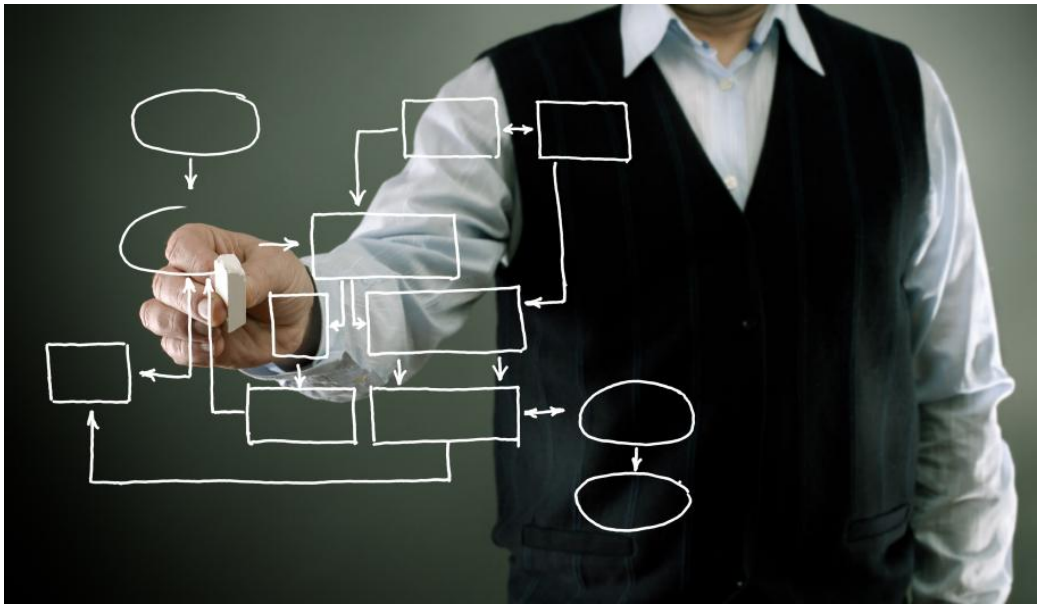
True or False: The best way to start a join is to build a query over one table?

Should you begin testing your join at the top of tables or the bottom?

.

UNIT VI: TEMPORARY TABLES

Create Temporary Tables
Change Table Link Error



TEMPORARY TABLES

Imagine for a moment a one table with all of the answers. A table created where all of the joins have been created for you. This table uses raw fields to create columns from formulas that you normally have to do with excel or a calculator every day. What if this table existed? This table contains all of your business critical answers. This is possible with temporary tables.

You can begin thinking about how to design a temporary table by looking at various reports you have with common data elements. If you have 15 reports that have 10 of the exact same fields, wouldn't it be nice to have one table that contained all of those fields for you.

Take a moment and consider what columns you would have in your perfect table.

Columns for the perfect table:

Now that you have some ideas of what you want in your temporary table lets look at making a few.

Batch Options

Some individuals have access to all or only one of the PRDCPY libraries. The PRDCPY libraries are EISPRDCPY, EISPRDCPY2, LAWPRDCPY, and MBPRDCPY. For this lesson the focus will be on EISPRDCPY, rather than listing all of the PRDCPY libraries each time.

EISPRDCPY is a collection that stores tables created by Query users. The tables in EISPRDCPY are typically created for two reasons:

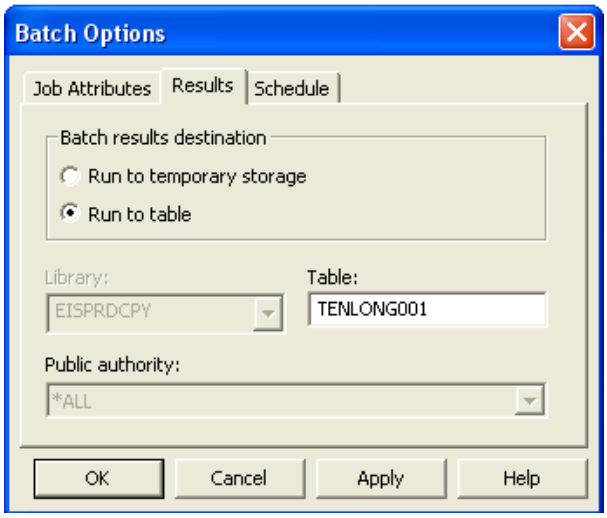
1. Query user has a large or complex query:
   A query that uses more than three tables may be divided into multiple steps to keep the queries to a manageable level of complexity.
2. Working with a subset of data from a large database:
   For example if you have a query that looks up information about members from the Northwest, you may decide to create your own table that has just members from the Northwest region, rather than running the query every time over the data of all members at Costco. Also, it can be used as a work table for your other queries.

These are the two major reasons to create a table in EISPRDCPY but there are many other uses for temporary tables.

The temporary tables on EISPRDCPY are stored as long as they are used at least once within a 30-day period. If a Query user does not access the table either via updating or via querying the table, it will be removed automatically after 30 days.

Tables can be created by accessing the **Batch results** option in the **Run** menu:

1. Select the **Run** menu.
2. Select **Batch Results**. The **Batch Options** window will open.
3. Select the **Results** tab.
4. Select **Run** to table.



The table field will now allow you to type a table name. Table names should not be any longer than ten characters and should not have leading numbers or symbols.

If leading numbers are used, the system will place quotes around the table name on the database, making it harder to link to for other queries.

If the table name is longer than ten characters, it will appear to work; however on the system, the name is truncated to ten characters and a data link is added to the rest of the table name. If the link is broken because of scheduled maintenance, an upgrade, or unscheduled downtime, the link between the query and the table will be broken. You will no longer be able to update the temporary table. Instead, you will get a security violation error.

Here is a best practice for naming your table:

- First two characters are your division:
  e.g., Human Resources = HR
- Next two characters are your specific department:
  e.g., Benefits = BE
- The next two characters are your position:
  e.g., Specialist = SP
- Finally, give a number for the report:
  e.g., 001
- The table name would look something like this: HRBESP001

NOTE: If you are creating several tables, it may be a good idea to create an index. This will assist in keeping track of your tables. Simply save a Word document in your G drive folder, and log your table names.

Example:

- **HRBESP001** Company summary table
- **HRBESP002** Employee Regional use table
- **HRBESP003** Benefit Analysis

Creating temporary tables that are massive in size defeats the purpose of using EISPRDCPY. Use temporary tables only when business needs call for them.

TROUBLESHOOTING TEMPORARY TABLES

Case Study:

A query is built called **Query 1**. **Query 1** writes to an EISPRDCPY table called **Table1**. **Query 1** only has one field in it: Warehouse. **Query 2** accesses the **Table 1** and another

table in the INPRDDTA collection to create a report. After running the query successfully several times, a field needs to be added to **Query 1,** so that now **Query 1** has two fields: Warehouse and Region. The query will not update **Table 1** because it will not be able to add the new field.

Instead, a new table will have to be created called **Table1a**. After **Table1a** has been created, **Query 2** will have to be updated to join to **Step1a** rather than to **Step1**.

After 30 days, if **Table1** is not accessed, it will be deleted.

This issue can also arise when a table you are querying has a file that has been changed. For example, when the warehouse field changed from three characters to five, queries that wrote to EISPRDCPY tables had to be updated to reflect the new field. Then the queries had to write to a new table as opposed to writing over the one that was used in the past.

When working with Temporary Tables use table name TRAINFL##, where FL equals your first and last initials, and ## equals a sequential number for each new table. Example: John Smith would be TRAINJS01. The next table created would be TRAINJS02.

EXERCISE 31: WRITE RESULTS TO A TABLE AND VIEW THE RESULTS

This query extracts the sales summary, but the union of periods creates a record for each period. The second query simply summarizes across the periods. This is a time saver, as summarizing in the first step would make for a longer run time.

1. Open EX16_step1.dbq.
2. Select the Run menu.
3. Make sure Enable Batch Processing is checked.
4. Select Batch Options.
5. Select Results tab.
6. Check the Run to Table button.
7. Key in the table name.
8. Select OK.
9. Run the query.
10. Retrieve Batch Results and review data.
11. Open a new query.
12. Choose the library EISPRDCPY and the table name created above.
13. Add all columns offered.
14. In the added columns, highlight Units, and then select the Sum button.
15. Highlight Dollars and select the Sum button.
16. Run the query.
17. Retrieve batch results and review data.

EXERCISE 32: WHO ELSE IS ON MY CARD?

You will need your membership card for this exercise. In this Query process, you will run your card number to retrieve the account IPK. The account IPK will be written to a temporary table and used to retrieve all the cards related to your membership card.

1. Open EX4step1.dbq.
2. Within Batch Options key in a new table name.

3. Run the query and retrieve batch results.
4. Open EX4step2.dbq.
5. The temporary table will have to be changed to link to the table that was created in step 3.
    a. In the Query menu choose Tables.
    b. Highlight the previous Temporary Table.
    c. Select the Change Link button.
    d. A message will prompt, “Any change link will apply all pending changes. Do you want to continue?” Click on the Yes button.
    e. Navigate to the EISPRDCPY library and the table created in step 3.
6. Click on the OK button.
7. Select OK
8. Run the query.
9. Retrieve the batch results and review.

EXERCISE 33: COUNT and SUM

Many times it is necessary to Summarize counts that have been done. In this example the request is to summarize a count of items that have had sales activity from one week to the next to see if the activity is increasing or decreasing with overall SKUS in the department.

1. Create a query from the Library INPRDDTA and the Table INIHSTP.
2. Select the fields Item number, Week 1 Sales in dollars, Week 1 Sales in units.
3. Highlight and sum both fields using the Sum button.
4. Select the Count button.
5. Click on Return distinct rows only.
6. Set your conditions Where Company = 1 AND Fiscal Year = YY (*YY= current Fiscal Year*) AND Department = 12 AND Warehouse Number = 110.
7. Still within the Conditions window in the Columns list Box change the setting to <All selected Columns>.
8. Set a Condition where SUM([Week 1 Sales in units]) > 1.
9. Run the Query and view the results.
10. In the Columns section try to place a Sum on the count.
    (later you might try to use code but you will get this error)

11. Run the results of this Query to a temporary table and Sum the count in the temporary table.

    How can you make the Sum 1 record?

CHANGE TABLE LINK ERROR - A REVIEW

Change Table Link Error

This error can occur for two reasons. Either your profile does not have access to the library or table, or the table or a field in the table has had an update you need to connect to.

The following questions can determine a profile issue:

- When was the last time you ran the query?
  If you ran this query recently and nothing has changed with your profile, you can rule out your profile as the issue.
- Is this a new query?
- Did someone e-mail this query to you?
- If this is a new query or a query that was sent to you, or if something has changed in your profile, check to see if you can access the table independently of this query. You can do this by creating a new query and browsing through the library and tables to see if you can view the table. If you can view the table, this is not a profile issue. If you cannot view the table or library, this is a profile issue. Your manager will need to fill out a profile request form to gain access to the collection or table.

If it's not a profile issue:

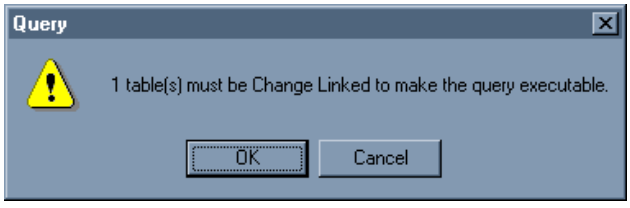
A more common reason for this error is that a table or field has been changed.
Determine what the problem is:

1. Each change link table error will specify what has changed.
2. Determine what has been changed.
3. Update the query to reflect the new table or field names.

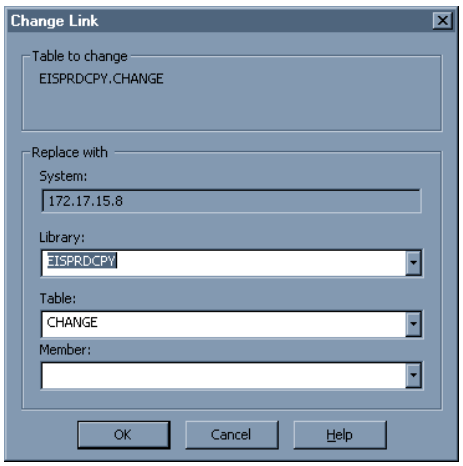
Here are the steps to fix the problem, if your query needs to be updated:

Example:
Library INPRDDTA and Table INWCTLP was changed to
Library INPRDDTA and Table INWCTLFP

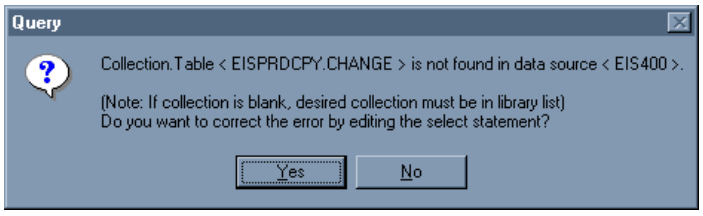
1. You will first see the initial error. Click OK.



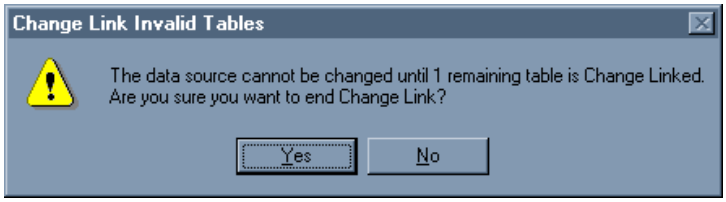
2. Click Cancel on the following window, to determine if it is a table or a field that needs to be updated.



a. You will see one of the following error messages. The first indicates that a table change needs to occur. Continue on to step 3.

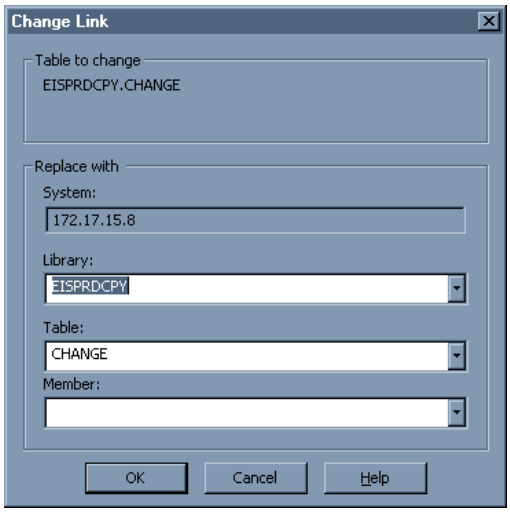


b. The second error message indicates that a field change needs to occur. Follow directions in the section *A Field Has Changed and Needs To Be Updated*.

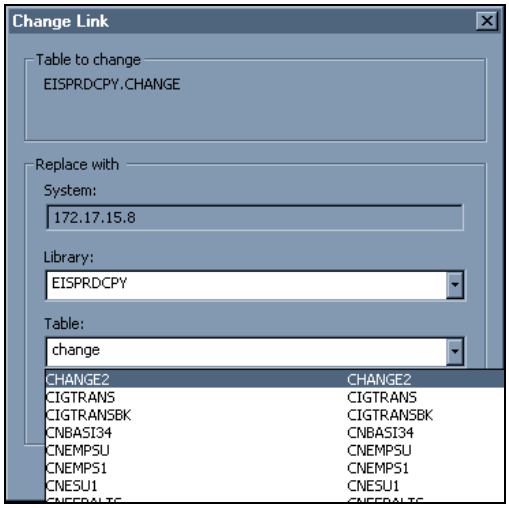


3. Next you will see a Change Link window. Select the Table list box arrow to display all tables. You can also key in the new table name if you know it. Typically, an e-mail is sent out letting users know of tables that will be changed.

   Note: There is a grayed-out box at the top of the window called “Table to change.” Make sure the library matches the library from the “Table to change” message with the drop-down menu below.



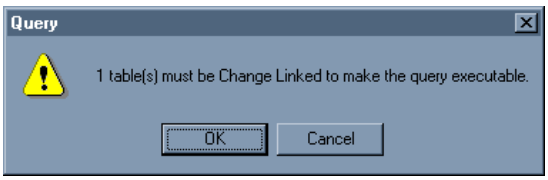
5. Select OK and run the query (unless there is more than one table that has been updated and needs changing).



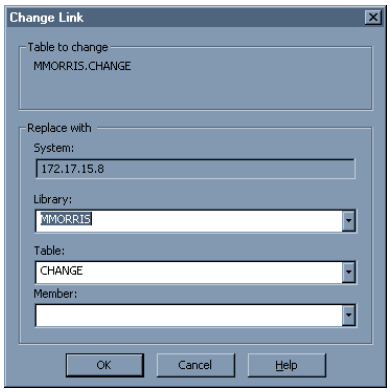
A Field Has Changed and Needs To Be Updated

Example
Library INPRDDTA and Table INWCTLP and Field WCWHSE was changed to
Library INPRDDTA and Table INWCTFP and Field WCWHS5.

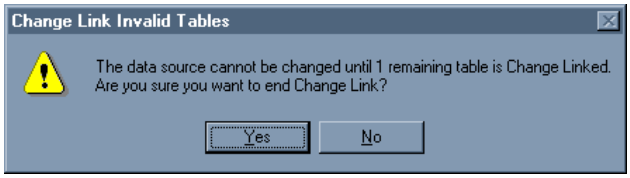
1. You will first see this initial error. Click OK.



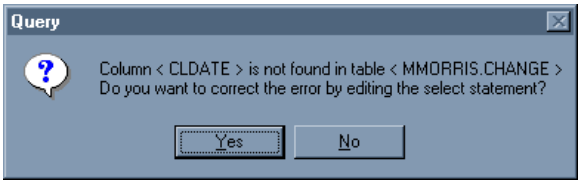
2. Next you will see a Change Link window. You will not be able to access fields from this window, so select Cancel.



3. Select Yes.

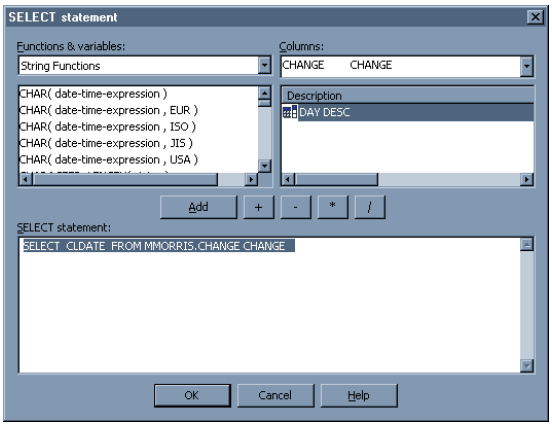


4. You will now see a window that will tell you what field needs to be changed. In this scenario, CLDATE is not found in library/table MMORRIS.CHANGE. Make a note of the library, table and field.



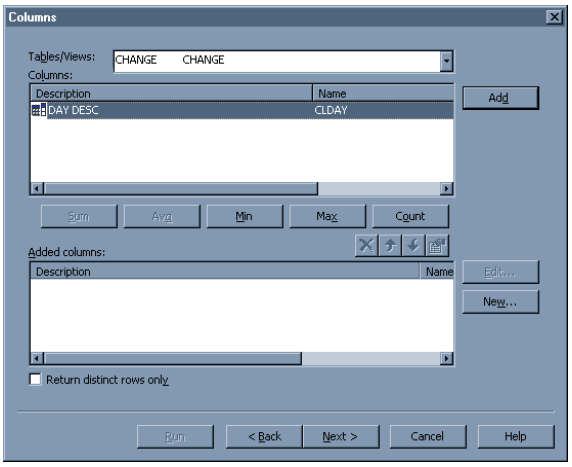
5. Select Yes to view the select statement. The select statement will appear in the lower window. With more complex select statements, it is best to copy and paste the entire statement into a Word document, and update the statement in Word. This makes it easier to edit.

6. Cancel out of the query, and open a new query. Browse to the library and table from the error message in step 4. In this scenario, the library is MMORRIS and the table is CHANGE. After adding the library and table, you can then navigate to and view all of the columns.

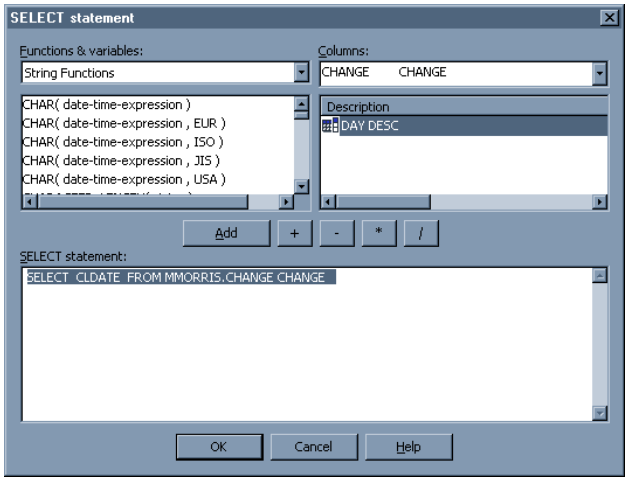


7. Find the new column. In this scenario finding the new column is simple, as there is only one to choose from. CLDATE was changed to CLDAY. In some cases it may not be so easy, and you will have to know what you are looking for. After

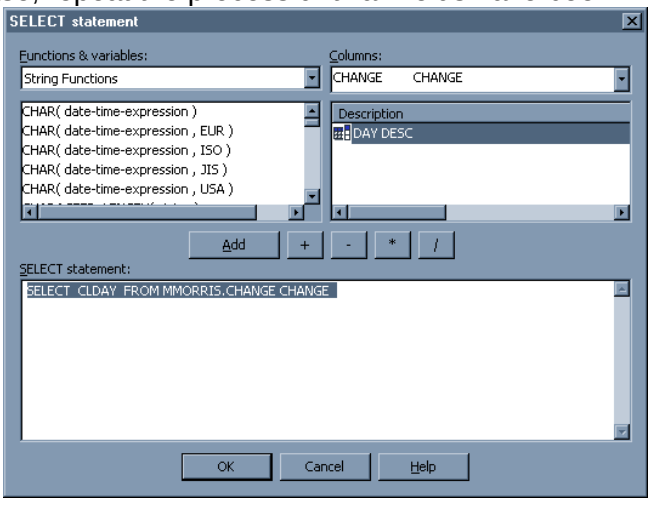
reviewing the new library and table a column called WCWHS5 is now present. There is a good chance this is the field you are looking for. If you are not sure, check for e-mail sent by the Data Warehouse group to notify users of changes. After the correct field is determined, make a note of it and close out of the new query.



8. Open the query with the error and follow the steps that lead to the **Select Statement** window, as outlined in steps 1 - 5.



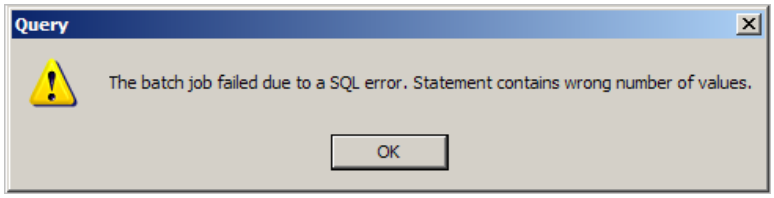
9. In the **Select statement** window, change the field CLDATE to CLDAY. Click OK and save the query. At this point you may find that there is another field that has been updated. If this is the case, repeat the process until all fields have been updated. Be sure to save the query!



EXERCISE 34: STATEMENT CONTAINS WRONG NUMBER OF VALUES

1. Using EX4step1.dbq, add the column **File Code** to your query.
2. Run the query and retrieve batch results.

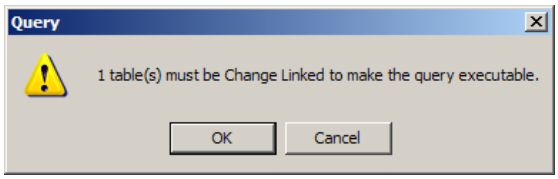
3. The following error should come up:



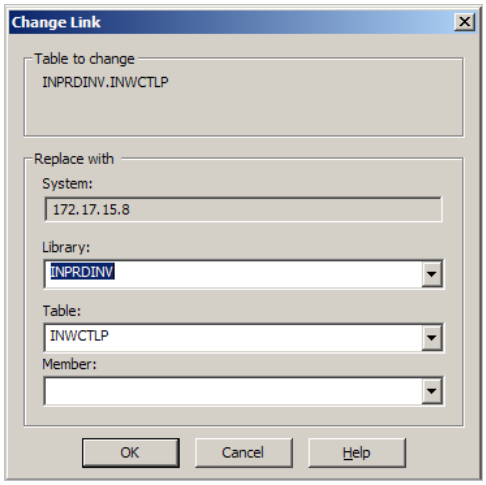
4. Select OK.
5. In the **Batch Options** window, rename the temporary table.
6. Run the Query again.
7. Run EX4step2.
    a. The query will run without error, but it is not accessing the new table.
8. Change the table link in EX4step2 to link with the newly created table.
9. Add the **File code** field to the columns.
10. Run and retrieve batch results.

EXERCISE 35: CHANGE LINK ERROR

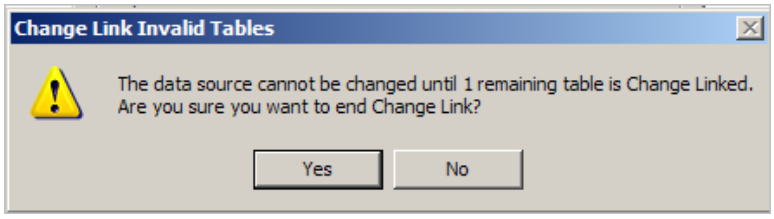
1. Run EX6step1WhsListLine.dbq.



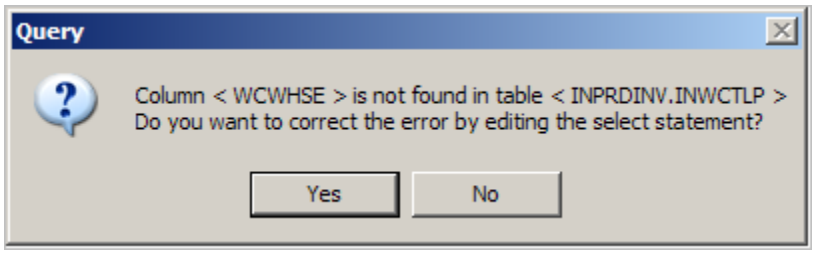
2. Select OK; the change table link window will appear.
   Being familiar with the query will show that this is actually the correct library and table combination. A field is incorrect and needs to be changed.



3. Click on the Cancel button. Another window will pop up asking if you are sure you want to cancel.



4. Select the Yes button. Another window will pop up showing the first field that is inaccurate. In this case it is WCWHSE.



5. Select No.
6. Open INPRDINV.INWCTLP and find what the new name of WCWHSE is named.
7. Write this name down. Follow the above steps again but this time when on the final prompt in step 4, select Yes. This will allow you to edit the select statement and change all of the WCWHSE fields to the accurate field.

EXERCISE 36: COMPARE AND FIND LOWER PRICE ITEMS

In this exercise we will be looking at the sell price of items in one warehouse and comparing those prices to another warehouse to see which warehouse has the lower price. This will require two queries. In the Query IV course, you will see how to do this in one step. For the purpose of this unit it will be divided into two queries.

Query 1 will be a list of items and sell prices from the warehouse that will be compared.

1. Create a query from the Library INPRDDTA and the table INWWIIP.
2. Select the fields Warehouse number, Item number, Item description, Sell price to wholesaler, Item status.
3. Create conditions where Warehouse number is an = (prompt) AND Department is a Department IS IN (prompt).
4. Create additional condition where Item status <> D.
5. Sort by the first by Warehouse number and then by Item Number.
6. Run the results to a new temporary table in EISPRDCPY.

Query 2 will compare the sell price of items from the first query to the warehouse from the second query, and display where the warehouse from the second query has lower prices.

7. Join the temporary table to INWWIIP. What field or fields would make the best join?
8. Add the table INFSELP from the library INPRDINV. This table will be used to filter FUTURE SELL DATE to equal NULL.
9. Join INFSELP to INWWIIP with Warehouse number and Item number.
10. The inner join type will not produce the correct results between INFSELP once the FUTURE SELL DATE is set to equal NULL.
11. SELECT DISTINCT the following fields from the associated table: INWWIIP.ZXWHSE, INWWIIP.ZXITEM, INWWIIP.ZXDES1, INWWIIP.ZXSTAT, INWWIIP.ZXOHUN, INWWIIP.ZXSELL
12. Create a new field using the General expression. Subtract INWWIIP.ZXSELL from the ZXSELL in your temporary table the expression should look like this: TEMPTABLENAME.ZXSELL - INWWIIP.ZXSELL
13. In the columns section of the Query wizard, add the field FSFSDT from the INFSELP table.
14. Create these conditions:
    a. '=' condition and prompt on INWWIIP.ZXWHSE
    b. INWWIIP.ZXALT <> 'Y'
    c. TEMPTABLENAME.ZXSELL - INWWIIP.ZXSELL <> 0
    d. FSFSDT IS NULL
15. Run the query and view the results. If you have no results, try entering a different department in step 1, then re-run step 2.

From simplifying complex multiple joins to creating business specific tables there are numerous reasons why you might create a temporary table.

Questions for Review:

How long are Temporary tables available if they have not been read or updated?

How long should the table name be?

What are at least three reasons you would build a temporary table?

After you build a temporary table can you add an additional field to that table?

UNIT VII: BASIC SQL

General Expressions
SQL
Unions

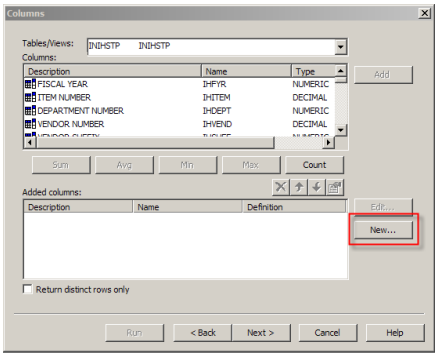
GENERAL EXPRESSIONS

General expressions use values, operators and functions with one or more value to determine a value. A simple expression is 2x2, more complex expressions will be discussed in the following units.

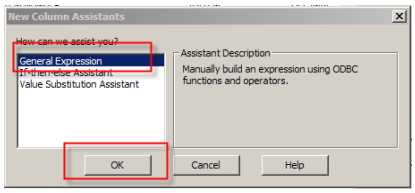
How to Start a New Column or a New General Expression

Follow these steps whenever starting a new general expression.

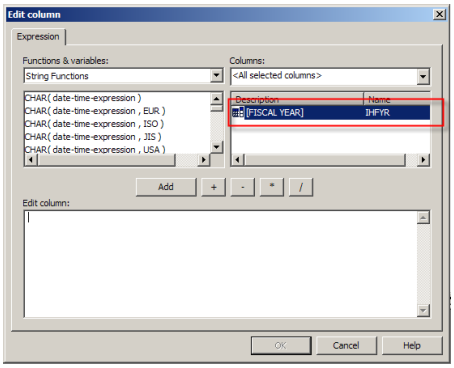
1. Using INPRDDTA/INITMMP create a new query.
2. In the Columns tab, select the New button.



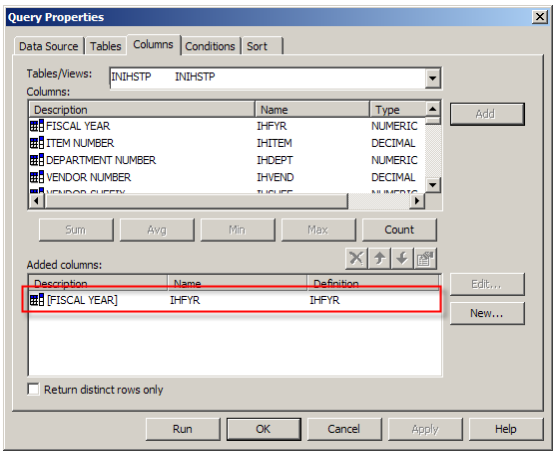
3. This will display the New Column Assistant. The column assistant has three options:
   a. General Expression
   b. If-then-else assistant
   c. Value Substitution assistant
4. If-then and the Value Substitution assistants are covered in Query IV.
   Highlight General Expression and select the OK button.



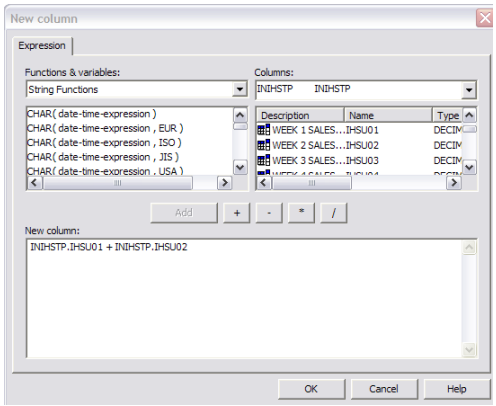
This opens the SQL editor.



5. In the Columns box double-click Fiscal Year.
6. Select OK.
    a. The new column should appear in the Added columns: box.



7. Now that you can access the General expressions, start a new General Expression
8. Double click on Week 1 Sales in Units.
9. Click on the “+” button.
10. Double click on Week 2 Sales in Units



11. Click OK
12. With this new Field added, apply a few conditions and view your results.

Computed Columns and Column properties

In General Expressions you can use basic math to add two columns together such as a Column for Week 1 and a Column for Week 2 can be added together to give you a new Column that equal Week 1 + Week 2. In this same fashion you can add several columns together to give you a representation of an entire Period, Quarter or Year. Not only can you create Columns by adding two or more columns together but those columns that you have created can now be referenced as Columns to be used in the creation of new columns.

Here is an example where the Query needs to show the percentage of growth from Period 1 to Period 2 from a table that only has week sales.

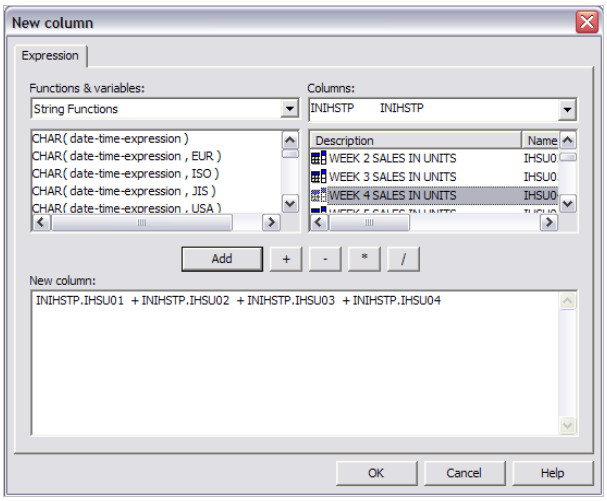
Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8

The first step in this process is adding the weeks together to represent the periods.

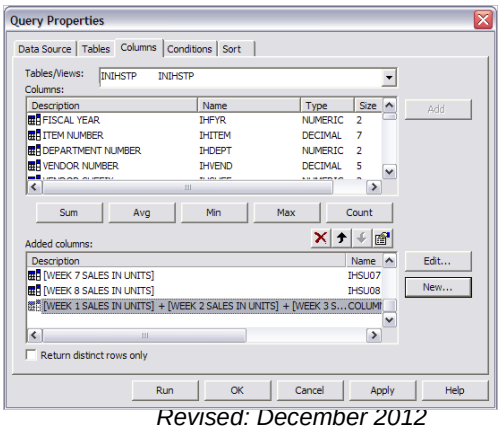
Week 1 Units + Week 2 Units + Week 3 Units + Week 4 Units — Period 1

Week 5 Units + Week 6 Units + Week 7 Units + Week 8 Units — Period 2

This can be done by simply clicking on New in the columns section of the Query Wizard and then on General Expressions.

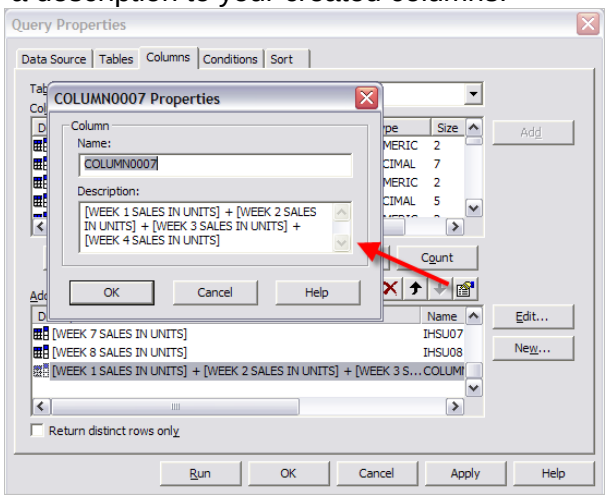


In the image above period 1 has been created. Once OK is selected you will be taken back to the columns screen of the Query wizard.



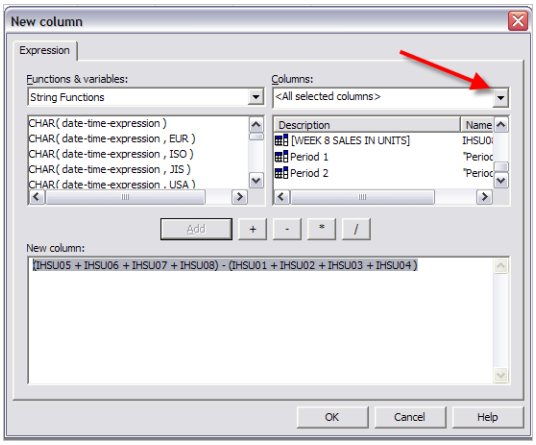
Notice the field that has been created is not very user friendly. The name of the field can be changed using the column properties button.

Column Properties allows you to name and give a description to your created columns.



Column names can be up to 28 characters. Column Description can contain up to 50 Characters. When working with several columns that you are creating to create other columns it will be important to name your columns something meaningful. Likewise if you are writing a created columns to a temporary table you will want to rename your columns to something meaningful.

Stacking your columns this way is an easy way to create complicated code by merely pointing and clicking. Created column1 + Created Column 2 = New Created Column 3. You can even take New created column 3 and New created column 4 (that was created using computed columns) and create your final column 5. This can be stacked over and over, but be sure to document the logic of what you are doing, as the code will become very complicated and hard to recreate or support.



In the example the Column created by adding 4 weeks together will simply be called Period 1. The process will be repeated to created another column called Period 2. Now with two created columns for Period 1 and Period 2, I can subtract period 2 from period 1 to create a new column called Period Difference. By clicking on New button in the columns section of the query wizard and General expressions I can do the math needed for this next step. Take note that when in this section you will need to Click on the List Box under columns and change it to <All selected columns>. This will allow you to navigate to the columns Period 1 and Period 2 that have been created.

When adding Columns that you have created using Math functions be aware that Query does not put in parenthesis for you. In this example After Period 2 and Period 1 were added the Expression was IHSU05 + IHSU06 + IHSU07 + IHSU08 - IHSU01 + IHSU02 + IHSU03 + IHSU04. Parenthesis will need to be added to make it correct.

With these tools complete the next exercise to find the percentage of growth from Period 1 to Period 2.

EXERCISE 37: PERCENTAGE OF GROWTH

1. In the INPRDDTA library select the INIHSTP table.
2. Open the Columns section of the Query wizard and click on New under Added Columns.
3. Add Week 1 Sales Units to the edit column box.
4. Click on the + button.
5. Add Week 2 Sales Units to the edit column box.
6. Click on the + button.
7. Add Week 3 Sales Units to the edit column box.
8. Click on the + button.
9. Add Week 4 Sales Units to the edit column box.
10. Click on the + button.
    a. IHSU01 + IHSU02 + IHSU03 + IHSU04
11. Click OK.
12. Set conditions to equal the current fiscal year, item number = 2
13. Run the query and view the results.
14. Create another computed column for units in week 5,6,7 and 8.
15. Run the query and view the results.
16. Create another new column.
17. Check all selected columns in the columns box to see the two created columns.
18. Add the second created column.
19. Place parenthesize at the beginning and the end of the calculation.
20. Click on the – button.
21. Add the first created column.
22. Place parenthesize at the beginning and the end of the calculation.
23. Run the query and view the results.
24. Create another new column.
25. Add the newly created third column.
26. Place parenthesize at the beginning and the end of the calculation.
27. Click on /.
28. Add the second created column.
29. Place parenthesize at the beginning and the end of the calculation.
30. Click on *.
31. Type in 100.
    a. ((IHSU05 + IHSU06 + IHSU07 + IHSU08) - (IHSU01 + IHSU02 + IHSU03 + IHSU04)) / ((IHSU01 + IHSU02 + IHSU03 + IHSU04)) * 100
    b. This will calculate the percentage of difference from period 2 to period 1.
32. Run the query and view the results.
33. You can now remove the unneeded Columns and keep on the last Column.
34. You can also rename the last column to something meaningful.

RETAIL MATH

These are common retail industry standard formulas. How can you use Query to recreate these in your department?

Net Receipts

(Purchases + transfers in + returns from customers + overages) – (Transfers out + returns to vendors) = Net receipts

Ending Inventory

Beginning inventory
- Sales
- Transfers out
- Returns to vendor
- Markdowns
- Employee discounts
- Shrinkage
+ Purchases
+ Returns from customers
+ Transfers in
+ Markups
= Ending inventory

Average Inventory

For six months: BOM* inventory added for past 7 months / 7
For one year: BOM inventory added for past 13 months / 13

Note:*BOM = Beginning of month

Turnover

Season = $ Total sales for season / $ Average inventory for season
Year = $ Total sales for year / $ Average inventory for year

Stock-to-Sales Ratio

$ Retail beginning inventory / $ Sales

Fresh Goods Factor
$ Receipts of the past two months / $ BOM inventory for current month

Open to Buy

For current month: Planned BOM inventory for next month
+ Planned sales for current month
+ Planned markdowns for current month
+ Planned shrinkage for current month
- Open orders for current month
- Actual BOM inventory for current month
= Open to buy for current month

For future months: Planned purchase for the month
- Open orders for the month
= Open to buy for the month

Purchases Needed for a Season

If season has not begun: Planned end-of- season inventory
+ Planned sales for season
+ Planned markdowns for season
+ Planned shrinkage for season
- Planned beginning-of-season inventory
= Purchase required for season

If season is in progress: Planned end-of-season inventory
+ Actual sales for season
+ Actual markdowns for season
+ Actual shrinkage for season
- Actual beginning-of-season inventory
= Purchases required for season

KEY PERCENTAGE RELATIONSHIPS

Percentage change this year (TY) from last year (LY)
(TY – LY) / LY

Percentage change this year from plan (PL)
(TY – PL) / PL

Shrinkage as a percentage of sales
$ Shrinkage / $ Sales

Markdowns as a percentage of Sales
$ Markdowns / $ Sales

Monthly percentage distributions
Sales: $ Sales for the month / $ Sales for the season = Monthly %
$ Sales for the season * monthly % = $ sales for the month

Markdowns: $ Markdowns for the month / $ Markdowns for the season = Monthly %
$ Markdowns for the season * Monthly % = $ Markdowns for the month

Initial Mark up %
(Sell – cost) / Sell

Percentage of difference between Comp shop and Sale price

MARGIN AND PROFIT

Gross Margin
Sales – cost of goods sold (COGS) = Gross margin
-or-
Sales – cost complement = Initial markup – reduction at cost = Gross margin
(where reduction at cost = [markdown $ + shrink $])

Merchandise Margin
Gross margin + vendor discounts

Merchandise Margin as a Percentage of Sales
$ Merchandise margin / $ sales

Vendor Discounts
Invoice amount of goods * shipping term % = Vendor discount $

Return on Inventory Investment at Cost (ROII)
(Merchandise margin % * turnover) / Cost complement = ROII
-or-
(Merchandise margin $ / average inventory) / Cost complement %

WHAT ARE SOME FORMULAS OR CALCULATIONS YOU USE IN YOUR REPORTING?

SELECT STATEMENT OR SQL

This chapter introduces SQL (Structured Query Language), also called *select statements*, and how to read simple Select Statements. Select, From, Where, Join, Group By and Sort are all covered.

What are the elements of a select statement? How does it work?

The SELECT Statement (also called a query or SQL) is a command used to retrieve data from a relational database. It is named after the operative keyword "SELECT." SQL consists of clauses that define what data results are returned (selected) from a database and how the data is returned.

SQL is used to communicate questions to the database.

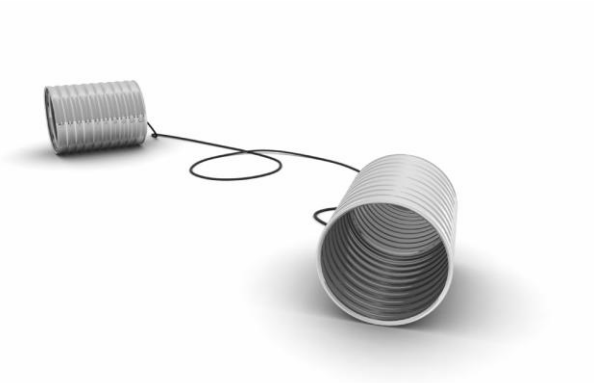
Select Clauses

The three main clauses are SELECT, FROM and WHERE.

- SELECT
  - Allows you to select certain columns from a table
  - Determines which columns of information are downloaded

- FROM
  - Specifies the tables from which the query extracts data

- WHERE
  - Defines the relationships between the tables (join conditions)
  - Determines which rows are selected from the tables



In this example, we will use **INITMMP**, the item detail table. This table resides in the **INPRDDTA** library.

```
SELECT ITEM #
FROM INITMMP.INPRDINV
WHERE ITEM # = 1
```

Two other clauses are the GROUP BY and ORDER BY clauses.

- GROUP BY
  Group By eliminates duplicate rows of information. This clause is required if a summary function is used in the SELECT, WHERE, or HAVING clauses.

- ORDER BY
  Determines how the data results are ordered when displayed (i.e., sorted).

SELECT Statements only require a SELECT and a FROM clause to run

EXERCISE 38: SQL CIRCLE THE CORRECT COMMAND

1. What SQL command would you use to choose a table and library?

A) SELECT **B)** FROM **C)** WHERE **D)** ORDER BY

2. What SQL command would you use to sort your report?

A) SELECT **B)** FROM **C)** WHERE **D)** ORDER BY

3. What SQL command would you use to choose columns?

A) SELECT **B)** FROM **C)** WHERE **D)** ORDER BY

4. What SQL command would you use to set up conditions?

A) SELECT **B)** FROM **C)** WHERE **D)** ORDER BY

EXERCISE 39: SQL FILL IN THE BLANK

Fill in the blanks to select item number, department number, and vendor number information from the library INPRDDTA and the table INIHSTP, where the item number equals “2”.

_____ ITEM NUMBER, DEPARTMENT NUMBER, VENDOR NUMBER

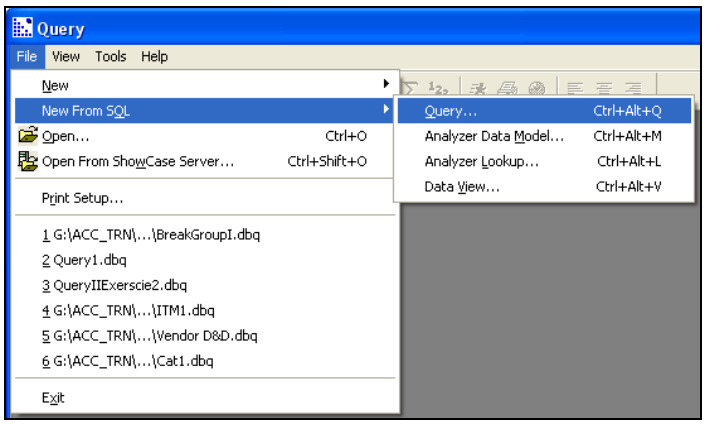
_____ INPRDDTA.INIHSTP

_____ ITEM NUMBER = 2

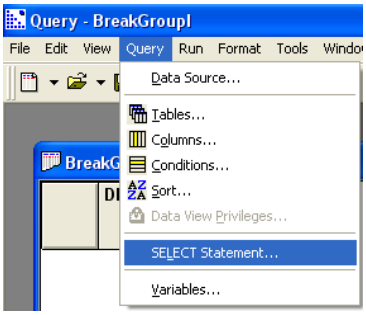
_____ 2 DESC

Accessing the SQL Editor

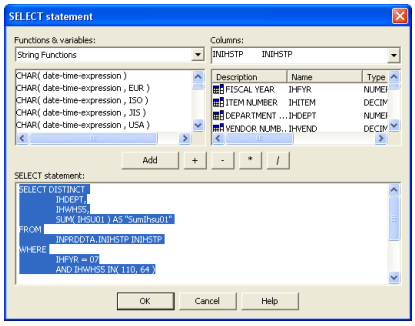
In Query, access to the select statement editor can occur in two ways. The first is by selecting the File Menu / New From SQL / Query.



The second way to access the SELECT Statement editor is with an existing query. From the **Query Menu** choose **SELECT Statement**. This opens the SELECT Statement editor.



There are three boxes in the SELECT Statement editor: Functions and Variables, Columns, and SELECT Statement. At this point, we will only be focusing on the SELECT STATEMENT box. The other boxes will be covered in detail in the Query III and Query IV classes.



UNION/SUB-SELECT

A union is a query within a query, or a sub-select. There are other types of sub-select statements that will be covered in Query III and Query IV. In this section, you will learn what a union is, the four rules to building a union, and how to create a union.

What is a Union

A union is a way to merge two like tables together. The union command is used to select related information from two tables, much like the join command. However, when using the union command, all selected columns need to be of the same data type.

Example

You have kept spreadsheets for 10 years that keep track of your banking transactions. Each year you start a new spreadsheet, however, all of the spreadsheets have the exact same columns. You decide you want to look at all your banking transactions on the same spreadsheet, so you combine all of the spreadsheets together for one master spreadsheet. This is a union, and it can be performed in an automated fashion for tables in Query.

When to Use a Union

Unions come in handy when you want to combine tables with exact information together.

Example

For this example there are two tables: table 1, **PERIOD 1 SALES,** and table 2, **PERIOD 2 SALES**.

You want to create a report that gives a total of Period 1 and Period 2 sales. You could run two queries, write the results to a spreadsheet, and then link the spreadsheets to do a summarization formula. However, you could combine the two tables with a union, and then run just one query to do the whole process in one step.

Four Union Rules

When creating a union, remember these rules:

1. Each SELECT Statement (the sub-select and the main SELECT) must have the same number of columns.

   The following example illustrates a case where a UNION cannot be created:

   Select statement 1
   SELECT ITEM, ITEM DESCRIPTION, ITEM CAT1, ITEM CAT2

   Select statement 2
   SELECT ITEM, ITEM CAT1, ITM CAT2 ITM CAT 3 ITEM SALES IN UNITS

2. Each column in each SELECT Statement must be the same as the corresponding column in all of the other SELECT statements.
   If you have a warehouse field in the first SELECT Statement that is five characters long, it needs to be five characters long in the second select statement. Likewise, the type of the field should be the same. For example, if the field in the first statement is a decimal field, then the corresponding column in the second statement should also be a decimal field.

   The following example illustrates a case where a UNION cannot be created. CHAR represents alpha characters.

SELECT statement 1
SELECT ITEM [Decimal (5,0)], ITEM DESCRIPTION[CHAR 25], ITEM CAT1 [CHAR 1], ITEM CAT2[CHAR 1]

SELECT statement 1
SELECT ITEM [Decimal (5,0)], ITEM DESCRIPTION[CHAR 25], ITEM CAT1[CHAR 1], ITEM CAT2[CHAR 3]

QUESTION
What two fields cause the above statements to be incompatible?

3. There can be only one sort or ORDER BY clause for the entire query, and it must appear after all UNION statements.
   The following example illustrates a case where the syntax is correct:

   SELECT ITEM, ITEM SALES
   FROM EISPRDDTA2.INID0801P
   UNION
   SELECT ITEM, ITEM SALES
   FROM EISPRDDTA2.INID0802P
   ORDER BY 1

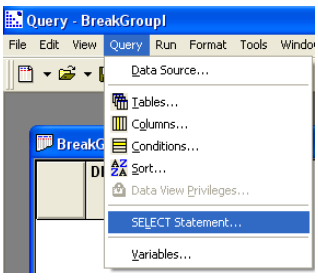
4. A UNION removes duplicate rows from the results; a UNION ALL does not**.**
   If you know that UNION will not produce any duplicate rows, use UNION ALL. The resulting query will run faster. This is because the database engine will not have to go through the results to remove duplicate rows.

Creating a Union

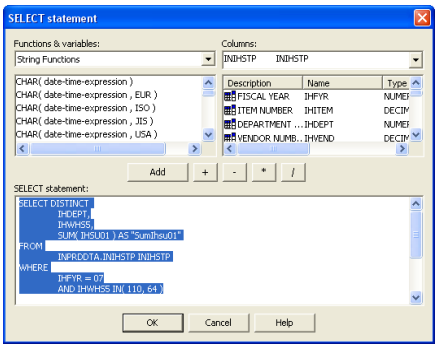
In order to create a union, you must manually alter the SELECT Statement.

The steps for building a union:
In this example, there is an existing query that a union will be added to.

1. Select the Query menu and then click on **SELECT Statement**.



2. The SELECT Statement editor window will display.



3. Copy the select statement and paste it into a Word document.
4. Type UNION or UNION ALL at the end of the select statement before ORDER BY.
5. Add the new Select Statement that you want to “union” with the existing query. Use Copy and Paste, and then alter the table name.
6. Copy the SELECT Statement with the union and paste it back into the **SELECT Statement** editor.

EXERCISE 40: CREATING A UNION OVER SALES INFORMATION

Use the following SELECT Statement to guide you through the Query Wizard to build building the query to be used for this exercise. FY = Fiscal Year and PD = Period

```
SELECT MBDATE, MBWHSE, MBDEPT, MBITEM, MBDSEL, MBDSUN
FROM EISPRDDTA2.INID FYPD P
WHERE MBWHSE = 110 AND MBDEPT = 17 AND MBITEM = 2
ORDER BY 4 DESC
```

1. After you have created the query, open the **SELECT Statement** editor.
2. Copy the SELECT Statement into a Word document.
3. Between the MBITEM=2 condition and the ORDER BY clause, enter a blank line and type UNION.
4. Copy the entire statement from before the UNION ALL and paste it between the UNION all clause and the ORDER BY clause. It should look like this:

   ```
   SELECT MBDATE, MBWHSE, MBDEPT, MBITEM, MBDSEL, MBDSUN
   FROM EISPRDDTA2.INID FYPD P
   WHERE MBWHSE = 110 AND MBDEPT = 17 AND MBITEM = 2
   UNION
   SELECT MBDATE, MBWHSE, MBDEPT, MBITEM, MBDSEL, MBDSUN
   FROM EISPRDDTA2.INID FYPD P
   WHERE MBWHSE = 110 AND MBDEPT = 17 AND MBITEM = 2
   ORDER BY 4 DESC
   ```

5. In the second SELECT statement, modify the table INIDFYPDP to be INIDFYPDP.
6. Copy the SELECT Statement from the Word document and replace the SELECT Statement in Query.
7. Click the OK button and run the query.
8. After you have retrieved the results, open the SELECT Statement once again in query. Notice the columns may have an association to the table they are related to.

When working with unions you will not only have to maintain or update the tables, you may have to update the fields as well. In the above exercise, you may have noticed that after you ran the query, the fields changed to have the corresponding table attached to them. If you were to add the next period or table **INID1203P**, you would have to update each field to represent the Period 3 table. It would look like this:

```
SELECT DISTINCT INID1201P. MBWHSE, INID1201P.MBDEPT,
INID1201P.MBITEM, INID1201P.MBDSEL, INID1201P.MBDSUN
FROM EISPRDDTA2.INID1201P INID1201P
WHERE INID1201P.MBWHSE = 110 AND INID1201P.MBDEPT = 17
AND INID1201P.MBITEM = 2
UNION ALL
SELECT DISTINCT INID1202P.MBWHSE, INID1202P.MBDEPT,
INID1202P.MBITEM,  INID1202P.MBDSEL, INID1202P.MBDSUN
FROM EISPRDDTA2.INID1202P INID1202P
WHERE INID1202P.MBWHSE = 110 AND INID1202P.MBDEPT = 17 AND
INID1202P.MBITEM = 2
UNION ALL
SELECT DISTINCT INID1203P.MBWHSE, INID1203P.MBDEPT,
INID1203P.MBITEM,INID1203P.MBDSEL, INID1203P.MBDSUN
FROM EISPRDDTA2.INID1203P INID12023P
WHERE INID1203P.MBWHSE = 110 AND INID1203P.MBDEPT = 17 AND
INID1203P.MBITEM = 2
ORDER BY 4 DESC
```

Tip:
If you are working with a number of unions, it can be tedious to manually update each field. Using the Find and Replace feature in Word can be very helpful. If you use this feature, be sure to double-check that the fields you didn't want updated were not inadvertently changed.

Troubleshooting

When creating unions you may receive this error message.

This means that IND1203P.MBWHSE cannot be found in the corresponding library. The solution is to update your SELECT Statement so that the fields can be found in the table, and the conditions can also be linked to the table.

EXERCISE 41: FIND THE ERROR IN THE UNION

The above error message came from the following **SELECT statement**. Find the error.

```
SELECT DISTINCT INID1201P.MBWHSE, INID1201P.MBDEPT,
INID1201P.MBITEM,  INID1201P.MBDSEL, INID1201P.MBDSUN
FROM EISPRDDTA2.INID1201P INID1201P
WHERE INID1201P.MBWHSE = 110 AND INID1201P.MBDEPT = 17
AND INID1201P.MBITEM = 2
UNION ALL
SELECT DISTINCT INID1202P.MBWHSE, INID1202P.MBDEPT,
INID1202P.MBITEM,  INID1202P.MBDSEL, INID1202P.MBDSUN
FROM EISPRDDTA2.INID1202P INID1202P
WHERE INID1202P.MBWHSE = 110 AND INID1202P.MBDEPT = 17 AND
INID1202P.MBITEM = 2
UNION ALL
SELECT DISTINCT INID1203P.MBWHSE, INID1203P.MBDEPT,
INID1204P.MBITEM,
INID1203P.MBDSEL, INID1203P.MBDSUN
FROM EISPRDDTA2.INID1203P INID2023P
WHERE INID1202P.MBWHSE = 110 AND INID1203P.MBDEPT = 17 AND
INID1203P.MBITEM = 2
ORDER BY 4 DESC
```

When creating unions, be sure to keep them manageable. Large unions will take up resources and be difficult to maintain. Unions are a great way to combine data so that multiple queries do not have to be run constantly. However, unions will need to have documentation on how to update them. Some unions need to be updated every month, every six months, or once every year. Having documentation is very important in the maintenance of queries using unions.

Questions for Review:

What does SQL stand for?

Which part of the select statement creates a sort?

Is a union the same as a join?

Can a query have both a union and a join?

What is the difference between Union and Union All?

UNIT VIII:
STRING FUNCTIONS

SQL Overview

SQL or Structured Query Language is the language used to communicate questions across our database. Reading a Basic SQL statement and the Union clause were covered in the previous unit. This text assumes that you are proficient with basic SQL.

The focus of this unit will be on String functions. A String is a string of characters, and a String function being a way of manipulating those characters.

When working with String Functions you can do so via keying them into the Select statement window or by using the query wizard and using the New button under columns to create a new General Expression. Creating String functions or complex SQL with the General Expressions Wizard is a simple point and click rather than needing to type code out.

STRING FUNCTIONS

String functions modify character data. String represents a string of characters; ABCDEFG is a string of characters. Modifications can include (but are not limited to) removing characters, including spaces, or including characters.

Operators

Operators are divided into four categories: Arithmetic, comparison, logical and string. All categories follow the same rules found in math. Essentially operators are math functions.

Operator type	Character(s)	Description
Arithmetic	+ - * / **	Addition or prefix plus Subtraction or prefix minus Multiplication Division Exponentiation
Comparison	= ¬= < ¬< > ¬> <= >=	Equal to Not equal to Less than Not less than Greater than Not greater than Less than or equal to Greater than or equal to
Logical	¬ & |	Not, Exclusive-or And Or
String	||	Concatenation

Delimiters

Delimiters are used to separate, or mark the start and end of items of data. (Similar to the punctuation found in grammar.)

Name	Delimiter	Use
Comma	,	Separates elements of a list; precedes the BY NAME option
Period	.	Connects elements of a qualified name; decimal or binary point
Semicolon	;	Terminates a statement
Equal sign	=	Indicates assignment or, in a conditional expression, equality
Colon	:	Connects prefixes to statements; connects lower-bound to upper-bound in a dimension attribute; used in RANGE specification of DEFAULT statement
Blank	b	Separates elements
Parentheses	( )	Enclose lists, expressions, iteration factors, and repetition factors; enclose information associated with various keywords
Locator	-> =>	Denotes locator qualification (pointers and offsets) Denotes locator qualification (handles)
Percent	%	Indicates %statements and %directives
Note: Omitting certain symbols can cause errors that are difficult to trace. Common errors are unbalanced quotes, unmatched parentheses, unmatched comment delimiters, and missing semicolons.		

Expressions
Within String Functions the term Expression is used. Expression operators create an expression, used to change or modify values returned. Expressions in SQL generally fall into one of four categories including: Boolean, Numeric, Character, and/or Date Expressions.

Argument
An argument can be a literal value, a variable, or an expression in a SQL statement.

String Functions covered:

- CHARACTER_LENGTH
- CONCAT
- DIFFERENCE
- DIGITS
- INSERT
- LCASE
- LEFT
- LENGTH
- LOCATE
- LTRIM
- POSITION
- POSSTR
- REPEAT
- REPLACE
- RIGHT
- RTRIM
- SOUNDEX
- SPACE
- SUBSTRING
- TRIM
- UCASE

CHARACTER_LENGTH

CHARACTER_LENGTH(expression)
CHAR_LENGTH(expression)

Character_Length will count the number of characters in a field.
Character_Length can be used when performing research. It can also be used with development and testing of queries.

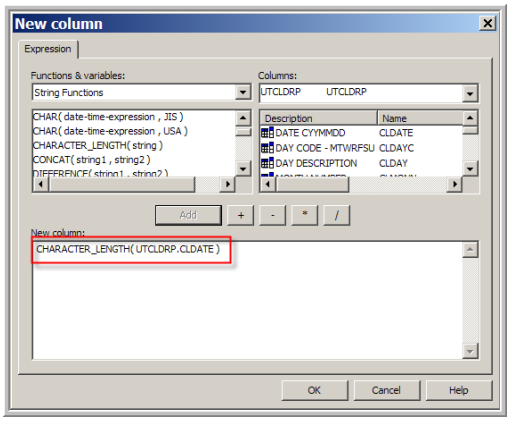
Character_Length returns the length of the character string, including blanks.

EXERCISE 42: CHARACTER LENGTH

	DATE CYYMMDD	CHARACTER_LENGTH([DATE CYYMMDD])
1	770101	6

Notice the length of the CLDATE field is represented in the newly created column.

1. Using PRDGPL and UTCLDRP add the field CLDATE.
2. Select the New button in the Columns window, and highlight General Expressions.
3. Chose CHARACTER_LENGTH(string).
4. Verify String Functions is highlighted. Within the Columns section of the SQL editor, double-click on CLDATE. CLDATE replaces String. This is what you should see:



5. Click OK.
6. In the Run menu limit the rows to one.
7. Run the query.

CONCAT

CONCAT(string_1, string_2)

CONCAT combines two fields or strings together.

String_1 and string_2 can come either from columns or entered text, and can be either character strings or graphic strings. Both must be of the same type. If they are character strings, their combined length must be 32,766 or less. If they are graphic strings, their combined length must be 16,383 or less.
For example a string that consists of string_1 is followed by string_2. If both are fixed-length character strings, the result is a fixed-length character string. Otherwise, the result is a varying-length character string. If either string can be null, the result can be null. If either string is null, the result will be the null value.

EXERCISE 43: CONCAT – Combining Two Fields

	ITEM NO	CAT	CAT2	CAT3	CONCAT([CATEGORY CODE - 1ST LETTER], [CATEGORY CODE - 2ND LETTER])
1	25	M	C	P	MC

In this exercise the IMCAT1 field and the IMCAT2 field will be combined together into one field.

1. Select IMITEM, IMCAT1, IMCAT2, IMCAT3 From INPRDDTA/INITMMP Where IMCMPY=1 AND IMITEM=25 AND DEPT = 14.
2. Run the query and view the results.

3. Start a new General Expression.
4. Under String functions double-click on CONCAT(string1, string2).
5. Verify that string1 is highlighted. Under Columns, double-click on IMCAT1.
6. Verify that string2 is now highlighted. Under Columns, double-click on IMCAT2.
   The expression should be: CONCAT(INITMMP.IMCAT1 , INITMMP.IMCAT2).
7. Run the query. Notice that IMCAT1 and IMCAT2 are now combined into one field.

EXERCISE 44: CONCAT CONCAT – Combining Three Fields

	ITEM NO	CAT	CAT2	CAT3	CONCAT(CONCAT([CATEGORY CODE - 1ST LETTER], [CATEGORY CODE - 2ND LETTER]), [CATEGORY CODE - 3RD LETTER])
1	25	M	C	P	MCP

In this exercise The IMCAT1, IMCAT2 fields and the IMCAT3 field will be combined together into one field.

1. Follow the steps 1 through 5 of Exercise 43.
2. Verify that string1 is highlighted. Under functions and variables, double-click on CONCAT(string1, string2) again. The expression should now be: CONCAT(CONCAT(string1 , string2), string2).
3. Verify that string1 is highlighted. Under Columns, double-click on IMCAT1
4. Verify that the first string2 is now highlighted. Under Columns, double-click on IMCAT2.
5. Verify that the second string2 is now highlighted. Under Columns, double-click
   on IMCAT3.
6. The expression should be: CONCAT(CONCAT(INITMMP.IMCAT1, INITMMP.IMCAT2) , INITMMP.IMCAT3).
7. Run the query.

DIFFERENCE
DIFFERENCE(string1, string2)

DIFFERENCE compares two strings to and returns an integer result ranging from 1 to 4, 1 being least similar and 4 being most similar. This uses the SOUNDEX coding for phonetic searches.

EXERCISE 45: DIFFERENCE

	WHSE MANAGER	REGION
1	CAROLYN HAGEMEYER	NW
2	CAROLYN ADAMS	SD
3	CAROLINE WALGREN	NE
4	CARLOS MACIAS	MX
5	CARL GOLSTON	BA

1. Select PHWMGR, PHREGN From INPRDINV/INPHONP.
2. Run the query and view the results.

3. Navigate to Conditions in the Query wizard.
4. Start an Expression.
5. Remove the highlighted field in the Expressions window. Double-click on DIFFERENCE(string1, string2)
6. Replace String1 with PHWMGR and string2 with ‘CAROL’. The expression should now be: DIFFERENCE(PHWMGR, 'CAROL')
7. Add what similarity you would like the difference to for this exercise it should equal 3. The expression should look like this DIFFERENCE(PHWMGR, 'CAROL') = 3.
8. Run the query and view the results.

DIGITS

DIGITS(string_1)

DIGITS returns a character string that is the absolute value of a number. If the number contains a decimal point, this function removes the decimal point, creating a string of numbers.

String_1 is the number to be converted into a character string. It can be either an integer or a decimal.

Example:

Expression	Returns Characters
DIGITS(1.245)	01245
DIGITS(150.25)	15025

EXERCISE 46: DIGITS

	DIGITS([DATE CYYMMDD])	DATE CYYMMDD
1	0770101	770101

The results will be numbers, however they have been converted to characters. This conversion gives additional flexibility to manipulating or filtering results.

1. Create a query Using PRDGPL and UTCLDRP.
2. Start a new General Expression.
3. Double-click on DIGITS(number) in the String Functions.
4. Verify that the number is highlighted.
5. Replace the number with the field CLDATE.
6. Select OK and run the query.

The results will be numbers; however they have been converted to characters. This conversion gives additional flexibility to manipulating or filtering results.

INSERT

INSERT(string1 , start , length_ , string2)

This function replaces on or more characters in another character string.

String1 represents the string where the new characters will be placed.

Start is the point in the string where the characters will be replaced.
Length is the number of characters in string1 that will be replaced.
String2 represents the characters to be inserted into string1.

EXERCISE 47: INSERT

	DESCRIPTION	INSERT([ITEM DESCRIPTION], 1, 1, 'S')
1	MILK 2% LOWFAT 2/1 GALLON	SILK 2% LOWFAT 2/1 GALLON
2	MILK WHOLE 2/1 GALLON	SILK WHOLE 2/1 GALLON
3	MILK NONFAT 2/1 GALLON	SILK NONFAT 2/1 GALLON

1. Select IMDES1 From INPRDDTA/INITMMP Where IMCMPY=1 AND IMITEM IN 1,2,12 AND DEPT = 17.
2. Run the query and view the results.
3. Navigate to Columns in the Query wizard.
4. Start a new General Expression.
5. Under String Functions double-click INSERT(string1 , start , length_ , string2).
6. Replace String1 with IMDES1, replace start with 1, replace length with 1 replace string2 with ‘S’. The expression should now look like this : INSERT(IMDES1, 1, 1, 'S')
7. Run the query and view the results.

LCASE

LCASE(string_1)

LCASE converts a field to lower case.

String_1 represents a field that will be converted to lower case.

EXERCISE 48: LCASE

	DESCRIPTION	LCASE([ITEM DESCRIPTION])
1	DIET COKE 24/12 OZ CAN	diet coke 24/12 oz can

1. Select IMDES1 From INPRDDTA/INITMMP Where IMCMPY=1 AND IMITEM=25 AND DEPT = 14.
2. Run the query and view the results.
3. Navigate to Columns in the Query wizard.
4. Start a new General Expression.
5. Under String Functions, double-click on LCASE(string1).
6. Verify string1 is highlighted.
7. In Columns, double-click on IMDES1. The expression should now be: LCASE(IMDES1).
8. Run the query and view the results.

LEFT

LEFT(string_1, length)

Starting from the left side of the original string, LEFT will return a number of specified characters in a field.

String_1 is either a character string, or graphic string from which the result is derived.
Length is an integer that specifies the size of the result.

EXERCISE 49: LEFT

	DESCRIPTION	LEFT([ITEM DESCRIPTION], 10)
1	DIET COKE 24/12 OZ CAN	DIET COKE

1. Select IMDES1 From INPRDDTA/INITMMP Where IMCMPY=1 AND IMITEM=25 AND DEPT = 14.
2. Run the query and view the results.
3. Open the SQL editor.
4. Start a new General Expression.
5. Under String Functions, double-click on LEFT(string1, count_).
6. Verify string1 is highlighted.
7. In Columns, double-click on IMDES1. The expression should now be: LEFT(IMDES1, count_).
8. Replace “count_) with the number 10.
9. Run the query and view the results.

LENGTH

LENGTH(string_1)

LENGTH counts the number of characters that are in a value in a field. For example a field can be 25 characters long with a value that is only three characters long.

Example:

Bob

String_1 is the value for which the length will be returned. The valid data types for string_1 depend on the data source being queried. (See the Compatibility section below.)

EXERCISE 50: LENGTH

	DESCRIPTION	LENGTH([ITEM DESCRIPTION])
1	DIET COKE 24/12 OZ CAN	22

1. Select IMDES1 From INPRDDTA/INITMMP Where IMCMPY=1 AND IMITEM=25 AND DEPT = 14.
2. Run the query and view the results.
3. Open the SQL editor.
4. Start a new General Expression.
5. Under String Functions, double-click on LENGTH(string1).
6. Verify string1 is highlighted.

7. In Columns, double-click on IMDES1. The expression should now be: LENGTH(IMDES1).
8. Run the query and view the results.

LOCATE

LOCATE(string1, string2, start)
The function LOCATE returns the starting position of the first occurrence of a string within another.

EXERCISE 15: LOCATE

1. With a pre-existing query open, navigate to the columns tab.
2. Start a new General Expression.
3. Under String Functions double-click on LOCATE(string1, string2, start)
4. Replace string1 with 'RED' and string2 with 'RED, BLUE, APPLE'. The string should now read: LOCATE('RED','RED, BLUE, APPLE', start)
5. Replace start with the number 11.
6. Run the query and view the results.
7. Replace the number 11 with the number 13.
8. Run the query and view the results.

LTRIM

LTRIM(string_1)

LTRIM function removes blank spaces at the beginning of a string.
String_1 must be a string expression, and is the string from which leading spaces will be removed.

EXERCISE 51: LTRIM

	DESCRIPTION	LTRIM([ITEM DESCRIPTION])
1	P100	P100

9. Select IMDES2 From INPRDDTA/INITMMP Where IMCMPY=1 AND IMITEM=25 AND DEPT = 14.
10. Run the query and view the results.
11. Open the SQL editor.
12. Start a new General Expression.
13. Under String Functions, double-click on LTrim(string1).
14. Verify String1 is highlighted.
15. In Columns, double-click on IMDES2. The expression should now be: LTrim(IMDES2).
16. Run the query and view the results.

POSITION

POSITION(string1 IN string2)
POSITION function displays the starting position of the first occurrence of string1 within string2.

EXERCISE 52: POSITION

	DESCRIPTION	POSSTR([ITEM DESCRIPTION], 'MILK')
1	MILK 2% LOWFAT 2/1 GALLON	1
2	MILK WHOLE 2/1 GALLON	1
3	MILK WHOLE 1 GALLON	1
4	1% MILK ONE GALLON	4
5	1% MILK 2/1 GALLON SL16	4
6	MILK 2% REDUCED FAT 1 GAL	1
7	MILK NONFAT 2/1 GALLON	1
8	CHOCOLATE MILK 12/160Z	11
9	MASTERS BUTTERMILK BREAD	15
10	QKR COOKY+MILK BAR 32/1Z	11

1. Select IMDES1 From INPRDDTA/INITMMP.
2. Start a new General Expression.
3. Under String Functions, double-click on POSITION(string1 IN string2).
4. Verify String1 is highlighted.
5. Enter the word ‘MILK’. The expression should now be: POSITION(‘MILK’ IN string2).
6. Replace string2 with the field IMDES1. The expression should now be: POSITION(‘MILK’ IN IMDES1).
7. Run the query and view the results.
8. Navigate to the Conditions window.
9. Highlight the newly created field.
10. Make the expression be greater then 0. POSITION(‘MILK’ IN IMDES1') > 0.
11. Run the query and view the results.

POSSTR

POSSTR(string1, string2)
POSSTR function displays the starting position of the first occurrence of string1 within string2.

EXERCISE 53: POSSTR

1. Select IMDES1 From INPRDDTA/INITMMP.
2. Start a new General Expression.
3. Under String Functions double-click on POSSTR(string1, string2).
4. Verify String1 is highlighted.
5. In Columns double-click on IMDES1. The expression should now be: POSSTR(IMDES1, string2).
6. Replace string2 with the word ‘MILK’. The expression should now be: POSSTR(IMDES1, ‘MILK’).
7. Run the query and view the results.
8. Navigate to the Conditions window.
9. Click on the Expressions button.
10. Remove the highlighted field in the expressions window.
11. Add the POSSTR string function.
12. Replace string1 with IMDES1 and string2 with ‘MILK. The expression should look like this: POSSTR(IMDES1, 'MILK').
13. Make the expression greater then 0. POSSTR(IMDES1, 'MILK') > 0.
14. Run the query and view the results.

POSITION AND POSSTR seem to be the same function. POSTION is character based where as POSSTR is strict byte-count based. It is recommended to use POSITION.

REPEAT

REPEAT(string , count_)

REPEAT returns a character string that consists of another string repeated a specified number of times. String represents the string to be repeated and count represents the number of times it will be repeated.

EXERCISE 54: REPEAT

	DESCRIPTION	REPEAT([ITEM DESCRIPTION], 2)	REPEAT('MILK', 3)
1	MILK 2% LOWFAT 2/1 GALLON	MILK 2% LOWFAT 2/1 GALLONMILK 2% LOWF	MILKMILKMILK
2	MILK WHOLE 2/1 GALLON	MILK WHOLE 2/1 GALLON MILK WHOLE 2/1	MILKMILKMILK
3	MILK WHOLE 1 GALLON	MILK WHOLE 1 GALLON MILK WHOLE 1 G	MILKMILKMILK
4	1% MILK ONE GALLON	1% MILK ONE GALLON 1% MILK ONE GA	MILKMILKMILK

1. Select IMDES1 From INPRDDTA/INITMMP Where IMCMPY=1 AND IMDEPT=17.
2. Run the query and view the results.
3. Open the SQL editor.
4. Start a new General Expression.
5. Under String Functions, double-click on REPEAT(string , count_).
6. Verify String is highlighted.
7. In Columns, double-click on IMDES1. The expression should now be: REPEAT(IMDES1 , count_).
8. Replace count with the number 2.
9. Run the query and view the results.
10. Start a new General Expression.
11. Under String Functions, double-click on REPEAT(string , count_).
12. Verify String is highlighted.
13. Replace String with the word MILK in single quotes. The expression should now be: REPEAT('MILK' , count_).
14. Replace count with the number 3.
15. Run the query and view the results.

REPLACE

REPLACE(string1, string2, string3)

Replace function reads a field looking for a string of characters, and then replaces those characters with what is desired.

String1 is the field being read. String2 is the string of characters to be replaced. String3 is what string2 will be replaced with.

EXERCISE 55: REPLACE

	DESCRIPTION	REPLACE([ITEM DESCRIPTION], 'COKE', 'SPRITE')
1	COKE DIET 24/12Z CAN	SPRITE DIET 24/12Z CAN

1. Select IMDES1 From INPRDDTA/INITMMP Where IMCMPY=1 AND IMITEM=25 AND DEPT = 14.
2. Run the query and view the results.
3. Open the SQL editor.
4. Start a new General Expression.
5. Under String Functions double-click on REPLACE(string1, string2, string3).
6. Verify String1 is highlighted.
7. In Columns double-click on IMDES1. The expression should now be: REPLACE(IMDES1 , string2 , string3).
8. Highlight string2 and in single quotes type the word COKE. The expression should now be: REPLACE(IMDES1 , 'COKE' , string3).
9. Highlight string3 and in single quotes type the word 'SPRITE". The expression should now be: REPLACE(IMDES1 , 'COKE' , 'SPRITE').
10. Run the query and view the results
11. Using what you have learned, remove the '/' in the description.

RIGHT

RIGHT(string_1, length)

This function returns a character string that consists of a specified number of characters, starting from the right side of the original string.

String_1 is either a character string, or graphic string from which the result is derived.
Length is an integer that specifies the size of the result.

EXERCISE 56: RIGHT

	DATE CCYYMMDD	RIGHT([DATE - CCYYMMDD], 4)
1	19770101	0101

1. Select CLDAT8 From PRDGPL/UTCLRP.
2. Run the query and view the results.
3. Open the SQL editor.
4. Start a new General Expression.
5. Under String Functions, double-click on RIGHT(string, count_).
6. Verify String is highlighted.
7. In Columns, double-click on CLDAT8. The expression should now be: RIGHT(CLDAT8, count_).
8. Replace count_ with the number 4.
9. Run the query and view the results.

RTRIM

RTRIM(string_1)
RTRIM removes blank spaces at the beginning of a string.

String_1 must be a string expression, and is the string from which leading spaces will be removed.

EXERCISE 57: RTrim

	1	1	CONCAT([ITEM DESCRIPTION], [ITEM DESCRIPTION])		CONCAT(RTRIM([ITEM DESCRIPTION]), [ITEM DESCRIPTION])
1	P=100	P=100	P=100	DIET COKE 24/12 OZ CAN	P=100DIET COKE 24/12 OZ CAN

1. Select IMDES2 From INPRDDTA/INITMMP Where IMCMPY=1 AND IMITEM=25 AND DEPT = 14.
2. Run the query and view the results.
3. Start a new General Expression.
4. Under String Functions, double-click on RTrim(string1).
5. Verify String1 is highlighted.
6. In the Columns, double-click on IMDES2. The expression should now be: RTrim(IMDES2).
7. Run the query and view the results.
8. It will appear as though nothing has happened. Create a new field by making a CONCAT with the field IMDES2 and IMDES1. This will display the blanks in IMDES2.
9. Create a concat with the new Rtrim field created and IMDES1.
10. Rerun the query and view the results.

SOUNDEX

SOUNDEX(string)

SOUNDEX is a phonetic algorithm. SOUNDEX is a code that is generated to give phonetic meaning to a string of characters. The code consists of the first letter of the string, and three numbers; each represents a type of letter. SOUNDEX is primarily used in searching for names that have similar phonetic spelling.

Code	Letters	Description
1	B, F, P, V	Labial
2	C, G, J, K, Q, S, X, Z	Gutterals and sibilants
3	D, T	Dental
4	L	Long liquid
5	M, N	Nasal
6	R	Short liquid

SKIP A, E, H, I, O, U, W, Y Vowels (and H, W, and Y) are skipped

EXERCISE 58: CREATE YOUR SOUNDEX

Use the chart above and the rules to create your SOUNDEX code.
You just visited the Northeast regional office, and worked with somebody named Caroline or Carolyn, but you cant remember her last name.

	PHREGN	PHWMGR
1	NW	CAROLYN HAGEMEYER
2	SD	CAROLYN ADAMS
3	NE	CAROLINE WALGREN

1. Select PHWNAM, PHREGN From INPRDINV/INPHONP Where IMCMPY=1.
2. Run the query and view the results.
3. Under the Conditions tab click on the “Expression…” button.
4. In the Expression window you may see a field that is highlighted. Delete the field from the Expression window.
5. Under String Functions, double-click on SOUNDEX(string1).
6. Verify String1 is highlighted.
7. In Columns, double-click on PHWMGR. The expression should now be: SOUNDEX(PHWMGR). Keep the expression window open and type in = ‘C645’ The expression should now be: SOUNDEX(PHWMGR) = ‘C645’

Also see (DIFFERENCE)

SPACE

SPACE(count)

SPACE places the number of spaces specified in the parenthesis. For example, (2) will bring back two spaces, (23) will bring back 23 spaces. This function works well in conjunction with CONCAT.

Count is the number of spaces to return.

EXERCISE 59: SPACE

1. Select IMITEM, IMDES2, From INPRDDTA/INITMMP Where IMCMPY=1 AND IMITEM=25 AND DEPT = 14.
2. Run the query and view the results.
3. Start a new General Expression.
4. Under String Functions, double-click on Concat(string1, string2).
5. Verify String1 is highlighted. Double-click on Concat. The expression should now look like this: CONCAT(CONCAT(string1 , string2) , string2).
6. Highlight String1, then in Columns, double-click on IMITEM.
7. String 2 should now be highlighted under Functions. Double-click on SPACE. The expression should now look like this: CONCAT(CONCAT(INITMMP.IMITEM , SPACE(count_)) , string2).
8. Highlight count and replace with the number 5, or any number you would like to enter for the test.
9. Verify string 2 is highlighted. In Columns, double-click on IMDES1. The expression should now look something like this: CONCAT(CONCAT(IMITEM, SPACE(5)), IMDES1).
10. Run the query and view the results.

SUBSTRING

SUBSTRING(string_1, start, length)

SUBSTRING returns a string extracted from another string.

String_1 is the string from which the result will be extracted. It must be a character string or a graphic string.

Start specifies the one-based position of the first character to extract from string_1. It must be a positive integer and cannot be greater than the length of string_1.

Length specifies the number of characters to extract. This is an optional parameter. If it is not specified, the resulting subset will consist of that portion of string_1 from start through the end of string_1.

EXERCISE 60: SUBSTRING

	DESCRIPTION	SUBSTRING([ITEM DESCRIPTION], 7, 6)
1	ALMOND POPPY SEED MUFFINS	POPPY

1. Build a simple query using library INPRDDTA and the table INITMMP.
2. Add the Item Description column or IMDES1.
3. Add the conditions of Company 1, Department = 62 and Item =60703.
4. Run the query and view the results. The results should display an item description of ALMOND POPPY SEED MUFFINS.
   NOTE: As the business changes, this information will more than likely change, making these SUBSTRING instructions outdated. The purpose of this exercise is to isolate one word in the description.
5. Begin a new General Expression.
6. In String Functions, select SUBSTRING.
7. Verify that SUBSTRING(string , start , length_) is in the New column window.
8. Replace String with IMDES1 or Item Description 1.
9. Use Start and the Length to isolate the word POPPY.

TRIM

TRIM(BOTH strip-character FROM string)
TRIM(LEADING strip-character FROM string)
TRIM(string)
TRIM(TRAILING strip-character FROM string)

TRIM removes blanks or characters from the beginning or ending of a string.

TRIM(BOTH strip-character FROM string) removes both leading and trailing characters.
TRIM(LEADING strip-character FROM string) removes leading characters.
TRIM(string) removes leading and trailing blanks from a string.
TRIM(TRAILING strip-character FROM string) removes trailing characters.

EXERCISE 61: TRIM(BOTH strip-character FROM string)

	ITEM NO	DESCRIPTION	DESCRIPTION	TRIM(BOTH '*' FROM [ITEM DESCRIPTION])
1	1	MILK 2% LOWFAT 2/1 GALLON	*BBS*	BBS*
2	2	MILK WHOLE 2/1 GALLON	*BBS*	BBS*
3	12	MILK NONFAT 2/1 GALLON	*BBS*	BBS*

1. Select IMDES1, IMDES2 From INPRDDTA/INITMMP Where IMCMPY=1 AND IMITEM IN 1, 2, 12.
2. Run the query and view the results.
3. Start a new General Expression.
4. Under String Functions, double-click on TRIM(BOTH strip-character FROM string).
5. Replace strip-character with '*'. The expression should now be: TRIM(BOTH '*' FROM string).
6. Replace String with IMDES2. The expression should now be: TRIM(BOTH '*' FROM IMDES2).
7. Run the query and view the results.

Based on the definition of TRIM BOTH the expected result would be BBS. Why is it BBS*?
How can you fix this using TRIM(string)?

EXERCISE 62: ITEMS MARKED DOWN (RIGHT)

1. Items ending in .97 are marked down. Track markdown list by warehouse and find out how many are on hand.
2. Use the table INWITMP in the library INPRDINV.
3. Select Columns WIWHS, WIDEPT, WITEM, WIOHUN, WISELL.
4. In Conditions have Warehouse = 110.
5. Sell price to Wholesaler = 97 hint Sell price to Wholesaler will use the string functions RIGHT within the conditions.
6. Create a prompt for ON HAND/ SELL UNITS using > as the condition.
7. Run the query
8. Place a number in the prompt for the amount of items you wish to see that are OH that have been marked down.
9. View the results.
10. Take note of how many records were displayed.
11. Create a Join on the table INITMMP in the library INPRDDTA

This table contains Item descriptions category codes etc. Add a few of these fields and run the query again. View the results. Is the number of records the same?

Try adding the table INWCTLP from the library INPRDINV and set a prompt for Regions.

This query can be modified to include warehouse contact information or vendor information.

How would you modify this to suit your needs? How would you also view markdowns that a warehouse made?

EXERCISE 27: Convert this SQL statement into a query with items, descriptions, and conditions on a CONCAT.

SELECT DISTINCT IDITEM, IDDEPT, CONCAT(IDCAT1, CONCAT(IDCAT2, IDCAT3)) AS COLUMN0000, IDDES1

```
FROM INPRDINV.INIDSCP INIDSCP
WHERE IDCMPY = 1
AND IDDEPT = 20
AND CONCAT( IDCAT1, CONCAT( IDCAT2, IDCAT3 ) ) IN( 'JBB', 'JFA', 'JGG', 'JLL',
'JMM', 'JMN', 'JOO', 'JPP', 'JSS', 'JTT', 'LEE', 'LII', 'LKK', 'LSS', 'OBE', 'QAC', 'RTT',
'THH', 'TMM', 'TPP', 'TSS', 'UCA', 'UCB', 'UCC', 'UHH', 'UHI', 'UKK', 'UMM', 'UPP',
'WHH', 'WJJ', 'WKK', 'WMM', 'WPP' )
ORDER BY
        1
```

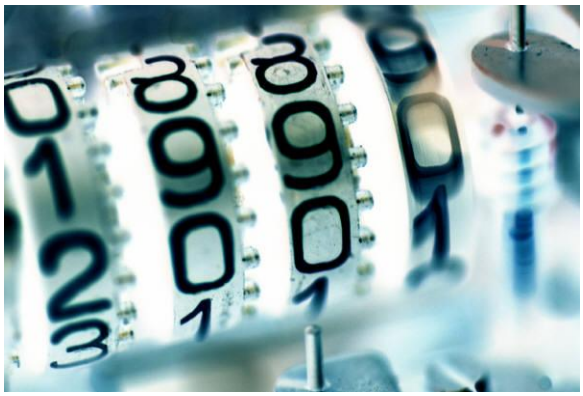
EXERCISE 63: PROPER NAME

Using the table MEPRNMP create proper case names using CONCAT SUBSTRING LCASE AND UCASE.

UNIT IX: TIME AND DATE FUNCTIONS

TIME AND DATE FUNCTIONS
NUMERIC FUNCTIONS
DATA TYPE CONVERSION FUNCTIONS

Time and Date Functions



The time and date functions can be used to manipulate time and date values by changing the format, or by applying logic to the time and dates.

DATE

All date examples were run May 12, 2010.

CURDATE()

This brings back the system date.

CURRENT_DATE

Returns the current date and does not accept parameters.

CURRENT_TIME

Returns the current time.

CURRENT_TIMESTAMP

Returns the current system timestamp.

CURRENT_TIMEZONE

Retrieves the data source's current time zone, based on Greenwich Mean Time.

CURTIME()

Returns system time.

Examples

CURDATE( )	CURRENT_DATE	CURRENT_TIME	CURRENT_TIMESTAMP	CURRENT_TIMEZONE	CURTIME( )
5/12/2010	5/12/2010	6:42:08 AM	5/12/2010 6:42:8.571571	-70000	6:42:08 AM

DATE(expression)

This will return the date as a character or numeral, depending on the expression. Expression is the character value for which a date will be returned. It must be a timestamp, a date, a positive number less than or equal to 3652059, a valid string representation of a date, or a string seven characters long.

Valid string representations of a date are in date format. For example, the character value "1998-02-01" will return a date value, but the character value "February 1, 1998" will not.

an expression is a string seven characters long, it must be in the form yyyynnn, where yyyy denotes the year and nnn denotes the Julian calendar date (dates between 001 and 366, January 1st is 001 and December 31st is 365 (except during a leap year, when December 31st is 366) .

Example

CURRENT_TIMESTAMP	DATE(CURRENT_TIMESTAMP)
5/12/2010 7:45:4.244883	5/12/2010

EXERCISE 64: DATE(expression, CYYMMDD)

This takes the CYYMMDD format and returns it as the DATE format.

PRDT	DATE([DATE PRINTED], CYYMMDD)
1010507	5/7/2001

1. Create a simple query using the *INPRDDTA* library and the *INPO50P* table.
   Note: This table looks at POs over $50,000.
2. Add the *PUPRDT* field.
3. Run the query and view the results.
4. In the *Columns* section, click **New**.
5. Choose **General Expression**.
6. Under *Functions & variables*, choose **Time/Date Functions**.
7. Double Double-click on **DATE(expression , CYYMMDD)**.
8. Replace *expression* with the *PUPRDT* field.
9. Run the query and view the results.

This same logic applies to:

DATE(expression , CYYDDD)
DATE(expression , DDMMYY)
DATE(expression , DDMMYY , CHAR)
DATE(expression , DDMMYYYY)
DATE(expression , MMDDYY)
DATE(expression , MMDDYY , CHAR)
DATE(expression , MMDDYYYY , CHAR)
DATE(expression , YYDDD , CHAR)
DATE(expression , YYDDMM)
DATE(expression , YYDDMM , CHAR)
DATE(expression , YYYYDDD)
DATE(expression , YYYYDDMM)
DATE(expression , YYYYDDMM , CHAR)
DATE(expression , YYYYMMDD)
DATE(expression , YYYYMMDD , CHAR)
DATE(expression, YYDDD)

DAY(expression)

The expression returns the day as a number. The expression must be a date, timestamp, date duration, or timestamp duration. Some fields may need to be changed to work with DAY. DAY(DATE('5/10/2010')) will return twelve digits.

EXERCISE 65: DAY(expression)

	MBDATE	DAY([SALES DATE])
1	5/10/2010	10

1. Create a query using *EISPRDDTA2* and the table *INID**FYPD**P*
   Note: Where FY = fiscal year and PD = period, use the last period of the current fiscal year.
2. In the *Columns* section, select **MBDATE** and click on **Return distinct rows only**.
3. Click the **New** button.
4. Choose **General Expression**.
5. Under *Functions & variables*, choose **Time/Date Functions**.
6. Double Double-click on **DAY(expression)**.
7. Replace HIghlight *expression* and double-click on **MBDATE**.
8. Set a condition of **Sales Date = DD/MM/YYYY** (pick any date within the period you chose in Step 1).
9. Run the query and view the results.

DAYNAME(date_expression)

This expression returns a character string containing the name of the day.

MBDATE	DAYNAME([SALES DATE])
5/10/2010	Monday

EXERCISE 66: DAYNAME(date_expression)

1. Create a query using *EISPRDDTA2* and the table *INID**FYPD**P*.
   Note: Where FY = fiscal year and PD = period, use the last period of the current fiscal year.
2. In the *Columns* section, select **MBDATE** and click on **Return distinct rows only**.
3. Click the **New** button.
4. Choose **General Expression**.
5. Under *Functions & variables*, choose **Time/Date Functions**.
6. Double Double-click on **DAYNAME(expression)**.
7. Replace Highlight *date_expression* and double-click on MBDATE.
8. Set a condition of **Sales Date =** any date from the *Value* drop-down menu.
9. Run the query and view the results.

DAYOFMONTH(date_expression),
Returns the day of the month in the date expression.

DAYOFWEEK(date_expression)
Returns the day of the week in the date expression.

DAYOFYEAR(date_expression).
Returns the day of the year in the date expression.

MBDATE	DAYOFMONTH([SALES DATE]	DAYOFWEEK([SALES DATE])	DAYOFYEAR([SALES DATE])
5/10/2010	10	2	130

EXERCISE 67: DAYOFYEAR(date_expression)

1. Create a query using *EISPRDDTA2* and the table *INID**FYPD**P*
   Note: Where FY = fiscal year and PD = period, use the last period of the current fiscal year.
2. In the *Columns* section, select **MBDATE**.
3. Click the **New** button.
4. Choose **General Expression**.
5. Under *Functions & variables*, choose **Time/Date Functions**.
6. Double Double-click on **DAYOFMONTH(date_expression).**
7. Replace Highlight *date_expression* and double-click on **MBDATE**.
8. Click **ok.**
9. Repeat steps 3-8, creating new expressions for both *DayofWeek(date_expression)* and *DayofYear(date_expression).*
10. Create conditions where **MBDATE =** any date from the *Value* drop-down menu, **Company Number = 1**, **Warehouse Number = 110**, **Department Number = 17**, and **Item Number = 2**.
11. Run the query and view the results.

DAYS(expression)

Counts the number of days from January 1, 0001 and returns a representation of the expressions date.

	MBDATE	DAYS([SALES DATE])
1	5/10/2010	733902
2	5/11/2010	733903

EXERCISE 68: Days(expression)

1. Create a query using *EISPRDDTA2* and the table *INID**FYPD**P*.
   Note: Where FY = fiscal year and PD = period, use the last period of the current fiscal year.
2. In the *Columns* section, select **MBDATE**.
3. Click the **New** button.
4. Choose **General Expression**.
5. Under *Functions & variables*, choose **Time/Date Functions**.
6. Double Double-click on **DAYS(expression)**.
7. Replace Highlight *Expression* and double-click on **MBDATE**.
8. In the *Columns* section, check the **Return distinct rows only** box.
9. **Company Number = 1**, **Warehouse Number = 110**, **Department Number = 17**, and **Item Number = 2**.
10. Run the query and view the results.

HOUR(time_expression)

Returns the hour part of a value as a number. It must be a time, timestamp, time duration or timestamp duration: HOUR(CURRENT_TIMESTAMP).

MICROSECOND(expression)

: MICROSECOND(CURRENT_TIMESTAMP).

MINUTE(time_expression)

Returns the minute part of a value as a number. It must be a time, timestamp, time duration or timestamp duration: MINUTE(CURRENT_TIMESTAMP).

MONTH(date_expression)

Returns month as a number: MONTH(MBDATE).

MONTHNAME(date_expression)

Returns character string containing the name of the day.

MONTH([SALES DATE])	MONTHNAME([SALES DATE])
5	May

EXERCISE 69: MONTHNAME(date_expression)

1. Create a query using EISPRDDTA2 and the table INID**FYPD**P.
   Note: Where FY = fiscal year and PD = period, use the last period of the current fiscal year.
2. In the *Columns* section, select **MBDATE**.
3. Click the **New** button.
4. Choose **General Expression**.
5. Under *Functions & variables*, choose **Time/Date Functions**.
6. Double Double-click on **MONTH(date_expression)**.
7. Replace Highlight *expression* and double-click on **MBDATE**.
8. Click **OK**.
9. Repeat steps 3-8, creating a new expression for *MONTHNAME(date_expression)*.
10. Create conditions where **Month(Sales Date) =** the number representing the month you selected in step 1 (ex: Period 13: August = 8) **Company Number = 1**, **Warehouse Number = 110**, **Department Number = 17**, and **Item Number = 2**.
11. Run the query and view the results.

NOW()

This returns the date and time from the system.

Example

NOW( )
5/12/2010 13:38:18.593959

QUARTER(date_expression)

Returns the quarter of the year where the date resides. A number between 1 and 4 is returned to represent the quarter:

1 = January – March
2 = April – June
3 = July – September
4 = October - December

Example

QUARTER([SALES DATE])
2

SECOND(time_expression)

Returns the seconds as a number. The expression must be a time, timestamp, time duration or timestamp duration.

TIME(expression)

Returns the time from a value. The expression must be a time, timestamp, time duration or timestamp duration.

NOW( )	TIME(NOW( ))
5/13/2010 6:58:30.82674	6:58:30 AM

EXERCISE 70: TIME(expression)

1. Create a query using EISPRDDTA2 and the table INID**FYPD**P.
   Note: Where FY = fiscal year and PD = period, use the last period of the current fiscal year.
2. In the *Columns* section, select **MBDATE**.
3. Click the **New** button.
4. Choose **General Expression**.
5. Under *Functions & variables*, choose **Time/Date Functions**.
6. Double Double-click on **TIME(expression)**.
7. Replace HIghlight *expression* and replace with **NOW().**
8. Create conditions where **MBDATE =** any date from the *Value* drop-down menu, **Company Number = 1**, **Warehouse Number = 110**, **Department Number = 17**, and **Item Number = 2**.
9. Run the query and view the results.

TIMESTAMP(date-expression , time-expression)

This allows you to reformat that timestamp using date and or time expressions, similar to a concatenation:
TIMESTAMP(CURRENT_DATE, CURRENT_TIME).

Example

CURRENT_TIMESTAMP	CURRENT_DATE	CURRENT_TIME	TIMESTAMP(CURRENT_DATE, CURRENT_TIME)
5/13/2010 13:2:28.335743	5/13/2010	1:02:28 PM	5/13/2010 13:2:28

TIMESTAMPADD(interval , integer, timestamp_expression)

Returns a calculated timestamp based off of the type of interval and integer:
TIMESTAMPADD(SQL_TSI_MONTH, 5, MBDATE).

CURRENT_DATE	TIMESTAMPADD(SQL_TSI_MONTH, 5, [SALES DATE])
5/13/2010	10/10/2010 0:0:0

Intervals

SQL_TSI_FRAC_SECOND	Nanoseconds
SQL_TSI_SECOND	Seconds
SQL_TSI_MINUTE	Minutes
SQL_TSI_HOUR	Hours
SQL_TSI_DAY	Days
SQL_TSI_MONTH	Months
SQL_TSI_YEAR	Years

EXERCISE 71: TIMESTAMPADD(interval , integer, timestamp_expression)

1. Create a query using *EISPRDDTA2* and the table *INID**FYPD**P*.
   Note: Where FY = fiscal year and PD = period, use the last period of the current fiscal year.
2. In the *Columns* section, select **MBDATE**.
3. Click the **New** button.
4. Choose **General Expression**.
5. Under *Functions & variables* choose **Time/Date Functions**.
6. Double Double-click on **TIMESTAMPADD(interval, integer, timestamp_expression)**.
7. Replace *interval* with **SQL_TSI_MONTH**.
8. Replace *integer* with **5**.
9. replace Highlight *timestamp_expression* and double-click on **MBDATE**.
10. Create conditions where **MBDATE =** any date from the *Value* drop-down menu, **Company Number = 1**, **Warehouse Number = 110**, **Department Number = 17**, and **Item Number = 2**.
11. Run the query and view the results.

TIMESTAMPDIFF (interval, timestamp_expression1, timestamp_expression2)

Returns the difference between two timestamps. Timestamp 2 will be subtracted from time stamp 1.

MBDATE	CURRENT_DATE	TIMESTAMPDIFF(SQL_TSI_DAY, CURRENT_DATE, [SALES DATE])
5/10/2010	5/13/2010	3
5/11/2010	5/13/2010	2
5/12/2010	5/13/2010	1

EXERCISE 72: TIMESTAMPDIFF (interval, timestamp_expression1, timestamp_expression2)

1. Create a query using *EISPRDDTA2* and the table *INID**FYPD**P*.
   a. Where FY = fiscal year and PD = period, use the last period of the current fiscal year.
2. In the *Columns* section, select **MBDATE**.
3. Click the **New** button.
4. Choose **General Expression**.
5. Under *Functions & variables,* choose **Time/Date Functions**.
6. Double-click on **TIMESTAMPDIFF (interval, timestamp_expression1, timestamp_expression2)**.
7. Replace *interval* with **SQL_TSI_DAY**.
8. Replace *timestamp_expression1* with **CURRENT_DATE**.
9. Replace Highlight *timestamp_expression2* and double-click on **MBDATE**.
10. Create conditions where **MBDATE =** any date from the *Value* drop-down menu, **Company Number = 1**, **Warehouse Number = 110**, **Department Number = 17**, and **Item Number = 2**.
11. Run the query and view the results.

WEEK (date_expression)

Returns the week of the year, counting from January.

Example

MBDATE	WEEK([SALES DATE])
5/10/2010	20
5/11/2010	20

YEAR(date_expression)

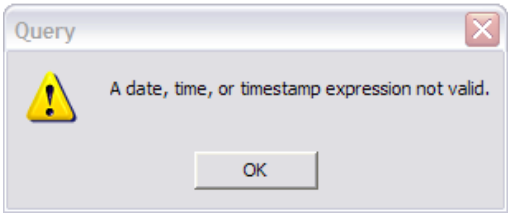
Returns the year.

MBDATE	YEAR([SALES DATE])
5/10/2010	2010
5/11/2010	2010

EXERCISE 73: CALCULATING AGE

(DAYS(DATE(CURRENT_DATE)) - DAYS(DATE(BTHDAT))) / 365

1. Create a new query using the library *MBPRDDTA* and the table *MEPRBSP*.
2. Select the fields **PERIPK** and **BTHDAT**.
3. Create conditions where *BTHDATE* does not equal 1/1/0001 and *PERIPK* is between 0 and 101.
4. Run the query and view the results.
5. In the *Columns* window, select **New**.
6. Choose **General Expression**.
7. Under *Functions & variables* choose, **Time and Date Functions**.
8. Double Double-click on **CURRENT_DATE**.
9. Run the query and view the results.
10. In the *Columns* window, select **New**.
11. Choose **General Expression**.
12. Under Functions & variables, choose **Time and Date Functions**.
13. In the *New Columns* section, add the **CURRENT_DATE** column.
14. Click the **Subtract** button.
15. Add the **BTHDAT** column, and then divide by 365.
16. Run the query.
17. An error should display.



Although the above logic seems like it would work, both date fields first need to be converted to dates that are the same format, and then those dates need to be converted to days, so the math will work.

18. Delete the new field causing the error.
19. In the *Columns* window, select **New**.
20. Choose **General Expression**.
21. Under *Functions & variables,* choose **Time and Date Functions**.
22. Click on **DAYS(expression)**.

23. Replace *expression* with **DATE(expression)**.
    The *Edit column* field should be displaying: DAYS(DATE(expression))
24. Replace *expression* with **CURRENT_DATE**.
    The *Edit column* field should be displaying: DAYS(DATE(CURRENT_DATE))
    Current Date has been converted to the date format so that Days can count the number of DAYS to the CURRENT DATE.
25. Run the query and view the results.
26. Repeat steps 19 - 24 to convert *BTHDAT* to *DATE* and then count the DAYS.
27. Run the query and view the results.
    With both of the fields turned into day counts, it is now possible to do some math.
28. In **General Expression**, take the converted *BTHDAT* field and subtract it from the converted *CURRENT_DATE* field, place parenthesis around the equation, and then divide by 365.
    The *Edit column* field should be displaying: ((DAYS(DATE(CURRENT_DATE))) - (DAYS(DATE(BTHDAT)))) / 365
29. Run the query and view the results.

Although there is the correct age the results are missing the remainder in days. This can be resolved as we will find out later with the Decimal function in the Numeric Functions.

Special Days

In some cases, you may always want to see yesterday's numbers, but not have to look up the date or key it in every morning. You can use SQL to create a way for Query to automatically recognize yesterday, or last month, or last month/last year.

EXERCISE 74: Special Days

Create a query that automatically will return special days:

1. Create a query using the library *PRDGPL* and the table *UTCLDRP*.
2. In the *Columns* section, click **New**.
3. Click on **General Expression**.
4. Click on the drop-down box under *Functions & variables* and choose **Time/Date Functions**.
5. Double-click on **DATE(expression)**.
6. Replace *expression* with **DAYS(expression)**.
   You should have: *DATE(DAYS(expression))*
7. Replace *expression* with **CURRENT_DATE**.
   You should have: *DATE(DAYS(CURRENT_DATE))*
8. Run the query and view the results.
9. Rename the header **Today**.
10. Create the following special days using the same process from steps 2-9:
    a. DATE(DAYS(CURRENT_DATE) - 365) AS "Today Last Year"
    b. DATE(DAYS(CURRENT_DATE) - 1) AS "Yesterday"
    c. DATE(DAYS(CURRENT_DATE) - 8) AS "Yesterday Last Week"
    d. DATE(DAYS(CURRENT_DATE) - 366) AS "Yesterday Last Year"
    e. DATE(DAYS(CURRENT_DATE) - 7) AS "Last week",
    f. DATE(DAYS(CURRENT_DATE) - 372) AS "Last Week Last Year"

How are dates used in your department?

- Billing date by day of week?
- Changes in day of week by year?
- Comparison of days of the week by two dates?
- Estimation based on day of week?
- Estimation based on previous year?
- How many customers on a given day?
- Order date to ship date?

One of the 6 rights of merchandising is having the product at the right time. How could using time and date functions help with this?

NUMERIC FUNCTIONS[i]

ABS

Returns the absolute value of a number.

Expression	Returns
ABS(-19.54)	19.54

ABSVAL(expression)

Returns the absolute value of a number. Converts negatives into positives.

Expression	Returns
ABSVAL(-19.54)	19.54

ACOS (float_expression)

Opposite of the COS function, this returns the arc cosine of a number in radians.

Expression	Returns
ACOS(-1)	3.14159. . .

ANTILOG (expression)

This function returns the anti-logarithm (base 10) of a number.

Expression	Returns
ANTILOG(2)	100

ASIN (float_expression)

This function returns the arc sine of a number in radians.

Expression	Returns
ASIN(1)	1.57079. . .

ATAN (float_expression)

This function returns the arc tangent of a number in radians.

Expression	Returns
ATAN(1)	.78539. . .

ATANH (float_expression1, float_expression2)

This function returns the hyperbolic arc tangent of a number in radians.

Expression	Returns

ATANH(.905148254) 1.50

COS (float_expression)

This function returns the cosine of a number.

Expression	**Returns**
COS(3.14159265359)	-1.0

COSH (expression)

This function returns the hyperbolic cosine of a number.

Expression	**Returns**
COSH(.5)	1.12762. . .

COT (float expression)

This function returns the cotangent of a number.

Expression	**Returns**
COT(.785398163397448)	1.0

DECIMAL (expression, precision_, scale_)

This function returns the packed decimal representation of a number.

Expression	**Returns**
DECIMAL(1.123, 2, 1)	1.1 (as an SQL_DECIMAL data type)

DEGREES (numeric_expression)

This function returns the number of degrees in an angle.

Expression	**Returns**
DEGREES(RAD)	the approximate value 180.0

When

the The host variable RAD is a decimal (4,3) host variable with a value of 3.142.

DOUBLE_PRECISION (expression)

This function returns a floating-point representation of a number.

Expression	**Returns**
DOUBLE(25)	25.

EXP (float_expression)

This function returns the base of the natural logarithm (e) raised to a power specified by the expression.

Expression	Returns
EXP(1)	2.71828. . .

FLOAT (expression)

This function returns a floating point representation of a number.

Expression	Returns
FLOAT(1)	1.0 (as an SQL FLOAT data type)

FLOOR (numeric expression)

This function returns a large integer that is the integer representation of a number.

Expression	Returns
FLOOR(374.99)	374

INTEGER(expression)

This function returns a large integer that is the integer representation of a number.

Expression	Returns
INTEGER(374.99)	374

LN (expression)

This function returns the natural logarithm of a number.

Expression	Returns
LN(1)	0
LN(2.718285)	1

LOG (float_expression)

Depending on the syntax you use, this function returns either the logarithm, base 10, of a number (using server syntax), or the natural logarithm of a number (using ODBC syntax).

Expression	Returns	When
LOG(100)	2	Server syntax is used
LOG(100)	3	ODBC syntax is used

LOG10(float_expression)

This function returns the common logarithm (base 10) of a number. The LOG and ANTILOG functions are inverse operations.

Expression	Returns	When
LOG10(100)	2	Server syntax is used
LOG10(100)	3	ODBC syntax is used

MOD (integer_expression1, integer_expression2)

This function divides one number by another and returns the remainder.

Expression	Returns
MOD(TOTPRC,QTYORD)	5

When
TOTPRC = 25 and
QTYORD = 10

PI()

This function returns a floating point value for pi.

Expression	Returns
PI( )	3.14159. . .

POWER (numeric_expression, integer_expression)

This function returns the result of raising the first argument to the power of the second argument.

Expression	Returns	When
POWER(3,QTYORD)	27	QTYORD=3

RADIANS (numeric _expression)

This function returns the number of radians converted from the number of degrees in the argument.

Expression	Returns
Radians(45)	0.7854
Radians(90)	1.5708

REAL (expression)

This function returns a single-precision floating-point representation of a number.

Expression	Returns
REAL(25.678)	25.678
REAL(25000.678)	25000.68

ROUND (numeric_expression, integer_expression)

This function returns the specified number rounded to the specified integer; the number rounded to the specified places to the right of the decimal point. If the integer is negative, the number is rounded to the specified places to the left of the decimal point.

Expression	Returns
ROUND(177.3589,2)	177.3600
ROUND(177.3589,-2)	200

SIN (float_expression)

This function returns the sine of a number. It is the inverse of the ASIN function.

A double-precision floating point value that is the sine of value.
If value can be null, the result can be null. If value is null, the result is the null value.

Expression	Returns
SIN(1.5707963267949)	1.0

SINH (expression)

This function returns the hyperbolic sine of a number.

SINH(value)

Value is the number for which the hyperbolic sine will be returned. It must be a number that is specified in radians.

A double-precision floating point value that is the hyperbolic sine of value.

Expression	Returns
SINH(1.5)	2.12. . .

SMALLINT (expression)

The argument must be a number, a character string representation of a number, a character string representation of an integer, or a character string representation of a floating-point number.

Expression	Returns
SMALLINT(25.675)	25

SQRT (float_expression)

Returns the square root of a number.

Expression	Returns
SQRT(16)	4

TAN (float_expression)

This function returns the tangent of a number. It is the inverse of the ATAN function.

Expression	Returns
TAN(.785398163397488)	1.0

TANH (expression)

This function returns the hyperbolic tangent of a number. It is the inverse of the ATANH function.

Expression	Returns
TANH(.5)	.46211. . .

TRUNCATE (numeric_expression, integer_expression)

This function returns the specified number truncated to the specified integer.

Expression	**Returns**
TRUNCATE(177.3589,2)	177.35
TRUNCATE(177.3589,-2)	170

ZONED (expression, precision_, scale_)

This function returns the zoned decimal representation of a number.
Value is the number for which the zoned decimal representation will be returned. It must be a number. Precision is the precision of the result. This is an optional parameter. If it is specified, it must be in the range of 1 to 31. If it is not specified, the default will be determined by the data type of value.

Data type of value	Default precision
Floating point	15
Decimal	15
Numeric	15
Non-zero scale binary	15
Large integer	11
Small integer	5

Scale is the scale of the result. This is an optional parameter. If it is specified, it must be in the range of 0 to precision. If it is not specified, the default is zero.
A decimal with the precision and scale specified by precision and scale that is the zoned decimal representation of value.

Expression	**Returns**
ZONED(1.123, 2, 1)	1.1 (as an SQL_NUMERIC data type)

CREATING RANGES

By using *Round* (numeric expression, integer expression) you can reduce the precision of a number, including break groups and concats, and you can create ranges.

ROUND
Round allows you to specify a number of decimal places for the numeric expression to round to. By specifying a positive number, you will have a rounded number to the power of 10. In the cases of ages, they normally run two digits, so the number that would need to be rounded would be 1. The *Round* statement would look like this. *ROUND(AGE , 1)*; this will round 34 to 30 and 35 to 40.

EXERCISE 75: WHAT AGE BRACKET BUYS MORE UNITS?

It should be noted that this is a partial report, and the results are an example of only 200 members over one period.

1. Create a query using table *MEPRBSP* from the library *MBPRDDTA*.
2. Select the columns **Person IPK**, **Current_date** (create this field via a new General Expression), **Birthdate**.

Create a field that calculates age:

3. In the *Columns* window, select **New**.
4. Chose **General Expression**.
5. Under *Functions a variables*, choose **Time and Date Functions**.
6. Double click on **DAYS(expression)**.
7. Replace *expression* with **DATE(expression)**.
   You should have*: DAYS(DATE(expression))*
8. Replace *expression* with **CURRENT_DATE**.
   You should have: *DAYS(DATE(CURRENT_DATE))*
   Current Date has been converted to the date format so that *Days* can count the number of Days to the Current Date.
9. Click OK
10. Under *Functions a variables*, choose **Time and Date Functions**.
11. Double click on **DAYS(expression)**.
12. Replace *expression* with **DATE(expression)**.
    You should have*: DAYS(DATE(expression))*
13. Replace *expression* with ***BTHDAT***.
    You should have: *DAYS(DATE(BTHDAT))*
14. Click OK
15. In *General Expression,* take the converted *CURRENT_DATE* field and subtract it from the converted *BTHDAT* field, then place parenthesis around the equation, and finally, divide by 365.
    You should have: ((DAYS(DATE(CURRENT_DATE))) - (DAYS(DATE(BTHDAT)))) / 365
16. Click OK.
17. Create a condition where **Person IPK is Between 1 and 500**.
18. Create a condition where **Birth Date <> 1/1/0001**.
19. Run the query and view the results.
20. Run the query and view the results.
21. Create a new column using **General Expression**.
22. Select the drop-down arrow for *Functions & variables*.

23. Choose **Numeric Functions**.
24. Double-click on **ROUND(numeric_expression , integer_expression)**.
25. Replace *numeric_expression* with the created age field (in Step 15).
26. Replace integer expression with **-1**:
    You should have: *ROUND(((DAYS(DATE(CURRENT_DATE))) - (DAYS(DATE(BTHDAT)))) / 365 , -1)*
27. In *batch options,* create a table name that is unique, beginning with **Train**, then first, middle, last initial and then a number. For example John Andrew Smith would be *TRAINJAS1*. Each table after this would have a sequential number *TRAINJAS2*, *TRAINJAS3*, etc.
28. Run the results of this query to the temporary table.
29. Save the query as **AgeRange step1**.
30. Close the query.
31. Create a new query from: *EISPRDDTA2.INIDFYPD06P*, *MBPRDDTA.MECABSP*, *MBPRDDTA.MEMSRLP*, and *EISPRDCPY.TEMPORARYTABLE*.
32. *MECABSP* joins to *INIDFYPDP* by **card number** to **membership number**.
33. *MEMSRLP* joins to *MECABSP* by **Membership IPK**.
34. *EISPRDCPY.TEMPORARYTABLE* joins to *MEMSRLP* by **linked-entity IPK** to **Person IPK**.
35. Select **Person IPK**, **Current Date**, **Birthdate**, **Created Age Column**, **Rounded Age Column**, **Item Number**, **Daily Sales in Units**, **Daily Sales at Sell**, and **Warehouse Number**.
36. Run the results toCreate a second temporary table and run the results to that table.
37. Save the query as **AgeRange Step2**.
38. Close the query.
39. Create a query using *TRAINFML2*.
40. Select the fields **Rounded Age** and **Daily Sales in Units**.
41. Create a *Sum* on *Daily Sales in Units*.
42. Create a break group on the *Rounded Age range* field.
43. Run the query and view the results.

This could have been done in one query. However, the number of records to process would cause the query to run for a long time. Creating a temporary table makes for a quicker filter, and the records are processed faster.

So why did the exercise take the results from step 2 and put them in a temporary table as well? The answer is because the query called for summarization and rounding, in addition to having several joins. It would be faster to take the results and write them to a temporary table, and then create aggregate functions.

Cleaning up the data:

1. Let's finish up this query with some basic clean up. Double-click on the **Summed Units** field and rename it **Total Units**.
2. Double-click on the **Rounded Age** field, and rename it **Age Range**.

In the *Age Range* field, we just have the rounded number, but what would be better is to see the range. Create a new field using **General Expression**.

3. Under *String Functions* click **Concat** twice:
   You should have: *CONCAT(CONCAT(string1 , string2) , string2)*
4. Replace *string1* with the **age range** column and **-5**:

You should have: *CONCAT(CONCAT(COLUM00004 - 5 , string2) , string2)*

5. Replace *string2* with a hyphen :
   You should have: *CONCAT(CONCAT(COLUM00001 - 5 , ‘ – ‘) , string2)*
6. Replace the last *string2* with the **Age Range** column **+4**.
7. Click **ok**, then **Apply**.

Run the query and view the results.

Additional fields were written to the temporary table in case more research was needed, or if other questions developed. What age range is buying salmon? If one warehouse’s age range is higher and another is lower, would this affect merchandising? Are there trends on when age ranges shop? What is main trend in age ranges of men purchasing Hathaway shirts at a particular warehouse? How will that affect stocking at another warehouse? What other questions could you think of regarding age ranges? What about different ranges, like date or price? How about replacing sales numbers with the words high, medium, and low?

DATA TYPE CONVERSION FUNCTIONS

There are two choices for converting expressions from one type to another: Cast and CONVERT.

CASTCAST is best when converting between **decimal** and **numeric** values to preserve the number of decimal places in the original expression.

CONVERT is better when converting between date and time values, fractional numbers, and monetary signifiers.

When in doubt about which one to use, default to CAST. With many SQL functions needing data types to match, CAST or CONVERt can be useful tools in combination with other functions.

CAST

CAST(expression as data_type)

Supported Casts between data types:

Supported Casts between data types

Source data type	SMALLINT INTEGER	DECIMAL NUMERIC	REAL DOUBLE	CHAR VARCHAR	GRAPHIC VARCHAR	DATE	TIME	TIMESTAMP
SMALLINT	Y	Y	Y	Y	--	--	--	--
INTEGER	Y	Y	Y	Y	--	--	--	--
DECIMAL	Y	Y	Y	Y	--	--	--	--
NUMERIC	Y	Y	Y	Y	--	--	--	--
REAL	Y	Y	Y	Y	--	--	--	--
DOUBLE	Y	Y	Y	Y	--	--	--	--
CHAR	Y	Y	Y	Y	*	Y	Y	Y
VARCHAR	Y	Y	Y	Y	*	Y	Y	Y
GRAPHIC	--	--	--	*	Y	--	--	--
VARGRAPHIC	--	--	--	*	Y	--	--	--
DATE	--	--	--	Y	--	Y	--	Y
TIME	--	--	--	Y	--	--	Y	Y
TIMESTAMP	--	--	--	Y	--	Y	Y	Y

*Conversion only supported for USC-2 graphic

CAST CAST can be used for several conversion functions, such as converting rates to decimals and converting decimals to characters for concatenation.

CONVERT

This function converts data from one SQL data type to another SQL data type.

CONVERT(value_expression, data_type)
The *Value_expression* is a column name, the result of another scalar function, or a literal value. *Data_type* is the data type to convert the expression to.

DATA TYPES
CHAR, NUM, DEC, SMALLINT

UNIT X: VALUE EXPRESSIONS

Three Valued Logic
NULLS
COALESCE
CASE

VALUE EXPRESSIONS

Value expressions are ways to alter the data returned. CASE can change the actual display of data, NULLIF the value, and COALESCE the organization.

There are many situations where you might find it useful to alter the data returned by a SQL query based on a few rules. For example, you may want to display customers gender as “Male” or “Female” based on whether their title is ”Mr.“ or one of ”Miss,” ”Mrs.“ or ”Ms.” The CASE expression can easily do this as well as create ranges.

The NULLIF expression is useful in situations where you want a returned value to be NULL. For example instead of returning 0 or returning a blank field, NULLIF will return NULL.

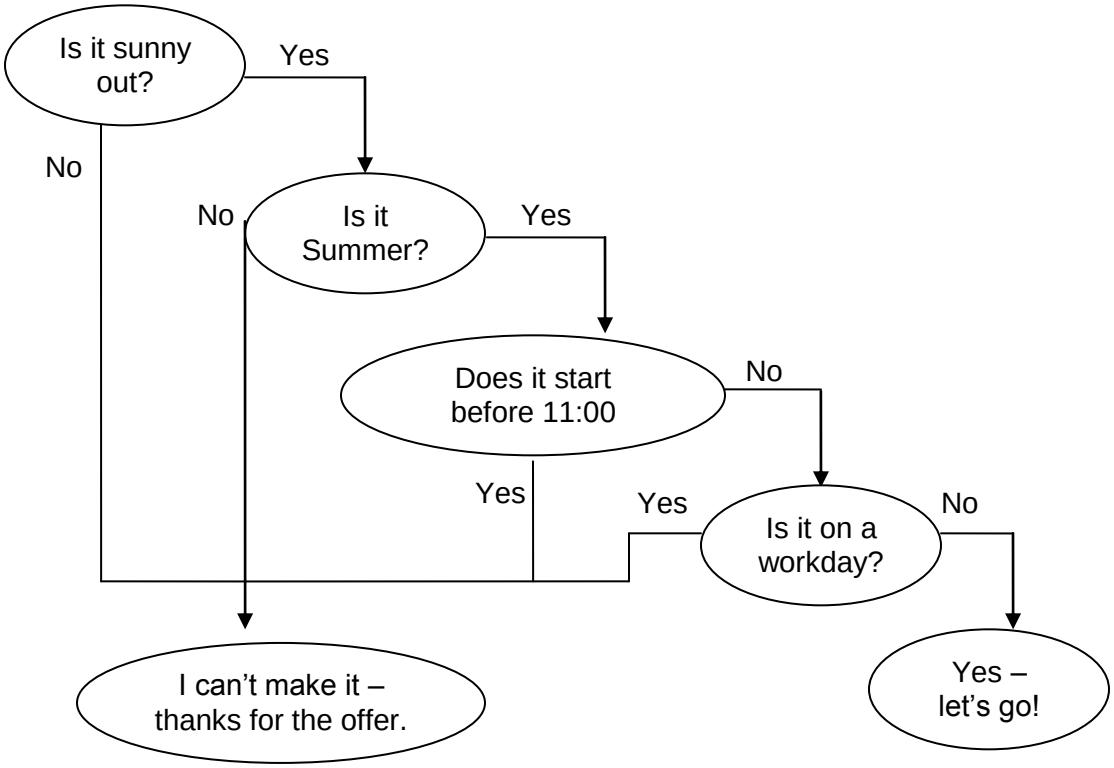
COALESCE returns the first non-NULL parameter in a series of columns.

Three Valued Logic, Unknown and Decision Trees

Three valued logic simple put gives an answer of True, False or Unknown. In decisions we need to solve sometimes for one or all three to understand our decisions.

Most decisions are made out of several scenarios. Let's say someone invites you to a picnic. How do you decide whether or not to go?

You may want to go to the picnic, but if it's not sunny out you will decide not to go. If it is sunny out, you still may not go if it's winter. You may also not go if it's before 11:00 a.m., or on a day that you work. Visually, you might see it as:



This is a very simplistic example. In reality, our decisions are much more complex, and we don't even realize it because our subconscious is processing this data much like query would. For example:

"I'll go to the picnic if it's between 70 and 85 degrees, and when it's after a payday, and when it's Saturday or Sunday, and when it's between 11:00 a.m. and 4:00 p.m. Then when Cal Anderson park = open when softball = no practice, and when groceries = bought, and when car = running, then picnic; otherwise when groceries <> bought, then buy groceries; when car <> running, then fix car. Otherwise stay home."

We naturally process a lot of information very quickly to make decisions. What decisions do you make on a daily basis at work? In the business world, some of the decisions we make take a lot of research. Query can process this information for us just as logically and quickly as you knowing you don't want to go to a picnic on a sunny day at 2:00 p.m. when it's below freezing.

Transferring your business knowledge into Query is crucial in supporting effective decision making and accurate business intelligence. Remember that it's extremely important to gather all the steps you need, not taking anything for granted.

Consider for a moment the decisions you make on a daily basis. You arrive at those decisions through a series of True, False and unknown scenarios. Following the thread of logic you can determine outcomes to decisions that need to be made. If then else or Case statements is how you can incorporate decisions trees or three level logic into SQL. Rather than True, False and unknown the results can be, Buy, Sell, Hold or Order, Stop order, Call, or Pay, Don't pay, Research. The power of coming to these conclusions quickly and accurately becomes apparent when applied to daily and weekly reports and research that can be done in moments.

Write out a few scenarios and decisions you make on a daily basis

CASE

Case can be used to reformat output, as well as dynamically sort results and group results into ranges.

A CASE statement uses IF THEN logic. This can be used to change the value of a result, or to showing Boolean answers within the logic. CASEstatements are either Simple or Searched.

Simple and Searched CASEStatements Defined

A simple CASE statement changes one value to another.

CASE expression WHEN value1 THEN result1

CASE Department WHEN 70 THEN 1HR PHOTO

The logic here is stating WHEN Department is 70 THEN display 1HR PHOTO.

A searched CASE statement allows for Boolean expressions in the WHEN clause.

CASE WHEN booleanExpression1 THEN result1 ELSE result

CASE WHEN SELL BETWEEN 0 and 10.00 THEN SELL ELSE NULL

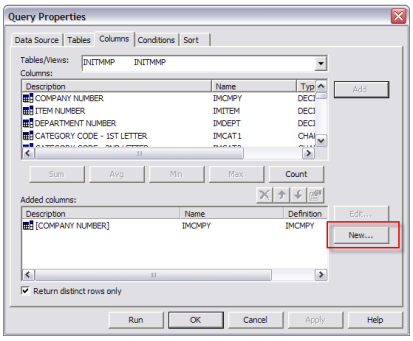
The case above will create a column with a range of SELL between 0 and 10.00. If data is outside that range, the result will return NULL.

EXERCISE 76: SIMPLE CASE

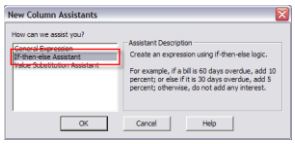
1. Using the library *INPRDDTA* and the table *INITMMP*, select the **Company Number** field.
2. Click on **Return distinct rows only** in the *Columns* window.
3. Run the query and view the results.

The data should be two records showing the results 1 and 4.
Company 1 represents United States; Company 4 represents Canada. Using simple CASE 1 and 4 can show this relationship.

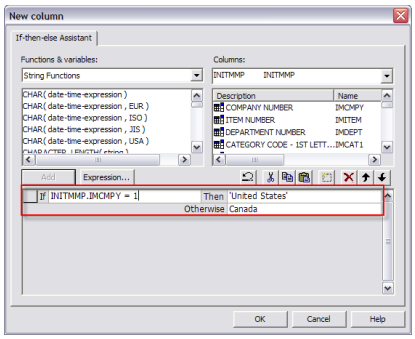
4. Open the *Columns* section of the Query wizard.
5. Click **New** in the *Added columns* section.



6. Highlight If-then-else Assistant.



7. Click OK.
8. Place your curser in the *If* text box.
9. Double-click on **Company Number**.
10. Type next to company number **= 1**.
11. Place the curser in the Then text box.
12. Type the value **'United States'**.
13. Place the curser in the Otherwise text box.
14. Type the value **"Canada"**.



15. Click OK.
16. Run the query and view the results.

NOTES:
Think about the reports you currently run and numbers that require action:
Order more product? Call the Vendor? Inform your manager of updates?
Can you think of uses for a simple CASE statement that will improve your reports?

EXERCISE 77: SIMPLE CASE – Using CASE to turn department numbers into department names

1. In the *INPRDDTA* library select the *INITMMP* table.
2. Open the *Columns* section of the Query wizard and click **New** under *Added columns*.
3. Select the **If-then-else Assistant**.
4. Select the *If* text box.
5. Double-click on **Department Number**.
6. Type **=011**.
7. Select the *Then* text box.
8. Type "**Tobacco"**.
9. Leave the *Otherwise* textbox blank.
10. Click **OK**.
11. Repeat steps 2-10, creating new fields for
    a. Dept 012 Candy
    b. Dept 013 Food
    c. Dept 014 Sundries
12. When creating the CASE statement for Sundries, select the *Otherwise* text box and type **Null**.
13. Click on Return distinct rows only.
14. Create conditions for **Company = 1** and **Department IN 11, 12, 13, 14**.
15. Run the query and view the results.
16. Save this Query for later exercise.

Notice all of the NULLS. We will discuss NULLS and how to handle them shortly.

EXERCISE 78: SEARCH CASE

Search CASE statements are set up in the same fashion as Simple statements. The only difference is Boolean functions are used. In this example, a range is being created using BETWEEN.

1. In the *EISPRDDTA2* library select the *INID**FYPD**P* table where FY = Fiscal Year and PD = Period.
2. Open the *Columns* section of the Query wizard and click **New** under *Added columns*.
3. Select the **If-then-else Assistant**.
4. Select the *If* text box.
5. Double-click on **Daily Sales at Sell.**
6. Under functions and variables change the category to Predicate Operator to **IS BETWEEN** and double click on it.
7. Type "**0.00 AND 10.00"**
8. Select the *Then* text box.
9. Double-click on **Daily Sales at Sell.**
10. Leave the *Otherwise* text box blank
11. Click on **OK**.
12. Create conditions where Company =1 and Department = 12 and Warehouse = 110.

UNKNOWN OR NULL

In decisions we often times need to filter out all of the unknown to find what is known.

It can be difficult to fully understand what unknown or NULL is, for example sometimes Unknown is not unknown because it is unknown and that is what is known. Let's take a look at the Cat in the box analogy to understand Nulls a little better.

There is a black cat and a white cat, and a cat-in-a-box. The cat in the box color is unknown.

If asked what color that cat in the box is the answer would have to be unknown. Not only can you not determine the color but you don't even know if there is a cat in the box.

There ways to approach the box that may or may not have the cat inside of it.
NI – no information: you don't know anything about the contents of the box.
OTH – other: you asked about dogs, not cats and so the box does not contain a cat it has been filtered and changed because the question is now different.
UNK – Unknown: you know there is a cat in the box but the color is unknown.
ASKU – asked but unknown – the invoice on the box is missing.
NAV – temporarily unavailable : the box invoice was sent separately from the box
NASK – not asked: an invoice was not created for the box
MSK – masked – the information about the box is private.
NA – not applicable: The question being asked does not represent cats or colors
NP – not present: there is a blank invoice on the box that has arrived

Consider A form for memberships. A member may or may not come to the counter to fill a form out to become a member. Think about the fields that may or may not have been filled out about the member.

NI – No information you don't know anything about the member
OTH – You are asking about vendors in a membership file
UNK – you are asking about membership that does exist but the name of the member is unknown.
ASKU – Although the membership exists the form is blank.
NAV – The membership form has not been entered yet.
NASK – A form was not created for the member example member 99
MSK – some of the membership information is private (credit card numbers)
NA – The question does not apply you are asking about members under the age over the age of 21.
NP – A b lank form has been entered with a new member number.

As you can see Unknowns can cover a lot of scenarios. In most logic however, you are trying to determine if something is true or false. In a recall program you may want to filter out all members where information regarding phone numbers and addresses are unknown or it could be that you want to find the names of those members to find alternate forms of research to contact those members.
When working with NULLS you may want to convert NULLS to a 0 (NULLIF) or you may want to convert values to a NULL (IFNULL) finally you may want to filter all NULLS out to discover you TRUE AND FALSE answers (COALESCE).

NULLIF

(expression_1,expression_2)

Expression_1 represents the field or expression being valued; if the value equals that of expression_2, the returned value will be Null. Example: a field with phone numbers. When a phone number is 0, rather than returning a value of 0, NULLIF will convert the phone number that is 0 to NULL. NULLIF(phone numbers, 0)

If the arguments are of a different data type, the comparison is not valid. The value of the first argument will always be returned.

EXERCISE 79: CHANGE 0 to NULL

Make phone number Null if it is 0.

1. Create a new query using table *INPHON* from the library *INPRDINV*.
2. Select columns **Warehouse Number**, **Miscellaneous Phone Number 2**, and **Miscellaneous Phone Number 3**.
3. Run the query and view the results.
4. In the *column* section, click the **New** button to create a new column.
5. Choose **General Expression**.
6. In the *Functions and variables* drop-down box, choose **Value Expressions**.
7. Double-click on **NULLIF(expression, expression)**.
8. Replace the first expression with the *Miscellaneous Phone Number 2* field.
9. Replace the second expression with **0**.
    a. Your Value Expression should look like this: *NULLIF (PHMS#2, 0)*.
10. Click **OK**.
11. Run the query and view the results.

This Query will be used with the next exercise.

IFNULL

IFNULL(expression_1,return_value)

IFNULL is the reverse of Null if it will replace NULL with a value. Although it may seem this should be found in the same category as NULLIF it is not. IFNULL is found under System Functions. IFNULL is useful when inserting missing data into temporary tables or working with averages. If the arguments are of a different data type, the comparison is not valid. The value of the first argument will always be returned.

NOTES:

COALESCE

COALESCE(expression_1,expression_2, expression_n)

This function returns the first string or field or column that is not NULL.

If there are three columns, and Column 1 has blanks throughout the records, and Column 2 has blanks throughout the records, and Column 3 has blanks throughout the records, coalesce would read all the columns in the order you asked and show the first record that was *not* blank.

Column 1	**Column 2**	**Column 3**
NULL	2	3
NULL	NULL	3
1	2	NULL
1	NULL	NULL

COALESCE(Column1, Column 2, Column 3)
Will return
Column 1
2
3
1
1

COALESCE(Column3, Column 1, Column 2)
Will return
Column 1
3
3
1
1

EXERCISE 80: SOLVE THE COALESCE

What would COALESCE(Column2, Column 3, Column 1) return?
Column1

_____
_____
_____
_____

Two or more strings must be specified, and all must be the same type. Therefore, if one is a date, all must be dates; if one is a number, all must be numbers.

EXERCISE 81: COALESCE Warehouse phone numbers

We want to contact the warehouse without tying up the main lines, so we start with the miscellaneous warehouse numbers that may or may not exist in the system. If one does

not exist, we would like that replaced with the office number. If the office number does not exist, then the receiving number; if the receiving number does not exist, then the Warehouse location number, so we can determine if it's a new or closed building.

The first Select statement shows the data. The second Select statement shows that the data is cleaned up. Using some string functions introduced in Query III, along with COALESCE and NULLIF, you can begin to see the power of combining different functions in SQL.

1. Begin by building a simple query that views warehouse phone numbers.
2. In the file menu, choose **New from SQL - Query**.
3. In the *SELECT Statement:* text box, type the following Select Statement;
   SELECT PHWHS5, PHAREA,"PHMS#3","PHMS#4","PHMS#5", PHOFC, PHRCV
   FROM INPRDINV.INPHON ORDER BY 1.
4. Run the query and view the results.
5. Convert each selected phone field 0 results to Null using NULLIF.
   a. In the *Columns* window, click **New**.
   b. Choose **General Expression**.
   c. From the *Functions and variables* drop-down box, choose **Value Expressions**.
   d. Double-click on **NULLIF(expression, expression**).
   e. Replace the first expression with **PHMS#3**.
   f. Replace the second expression with **0**.
   g. Repeat this process for each selected phone number.
6. Run the query and view the results.
7. Now its time to search for the first column that has a phone number using COALESCE.
   a. In the columns window, click **New**.
   b. Choose **General Expression**.
   c. In the *Functions and variables* drop-down box, choose **Value Expressions**.
   d. Double-click on **COALESCE(expression , expression)**.
   e. There are five phone number fields, so the value expression has to be altered to work with five fields. Change the expression to **COALESCE(expression , expression, expression , expression, expression)**.
   f. Replace each expression with the newly-created phone numbers field (the fields beginning with NULLIF…). The statement should look like this; *COALESCE(NULLIF("PHMS#3", 0), NULLIF("PHMS#4", 0), NULLIF("PHMS#5", 0), NULLIF(PHOFC, 0), NULLIF(PHRCV, 0))*
   g. Run the query and view the results.
   h. Delete all other phone number fields added to the query except the last one created.

How might you use this to collate sales, categories, items or your own business information?

Take a look at the at Exercise 77 again. Use Coalesce to clean this data up.

INITIAL MARK UP (IMU)

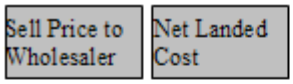
IMU is the difference between what we pay for product, vs. what we sell it for. We cannot just sell products for any price we want. There are parameters to how much we can mark something up. The parameters are based on company standards, department goals, and our competition. Company standards ensure that we do not mark anything up more than 14% or less than 3%. Some departments may need to work within a smaller range like 8 to 12% to maintain sales IMU. Finally, we have to sell at a lower price than our competition, which also affects our IMU. The initial mark-up, or IMU, can be represented in dollars, or as a percentage.

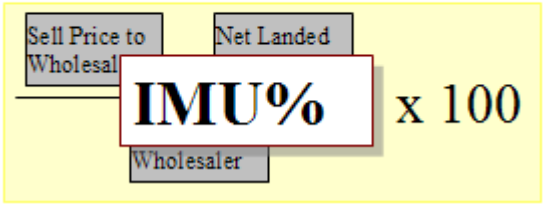
In Dollars: Sell Price – Cost = IMU
As a Percentage: Sell Price – Cost / Sell Price = IMU%

Let's outline what we will be creating in the next exercise: Costco and our competitors sell many of the same items. If our IMU is only, say, 2%, and our lowest competitors' sell price is at least 5% or more than our sell price, then the question must be asked: "Can we mark up our product more, still be within our parameters, and yet meet our business needs?

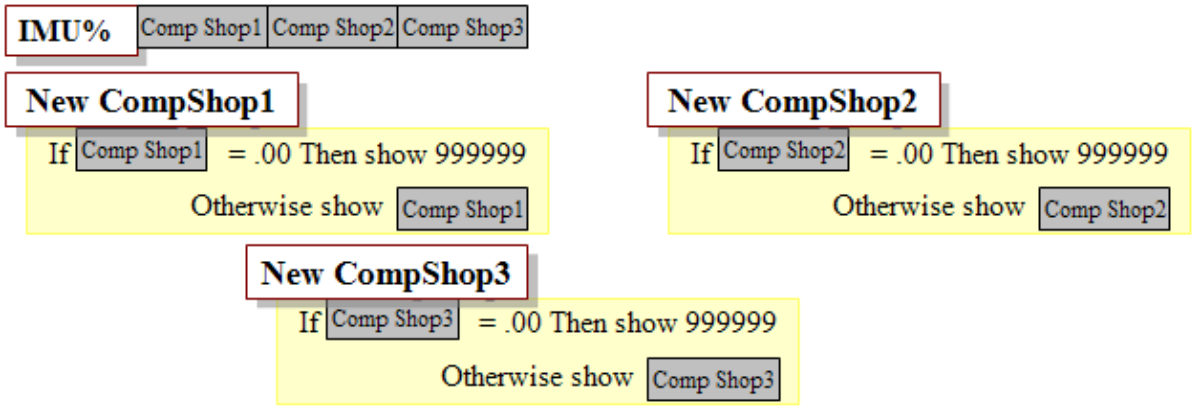
Here's the thought process—an overview—of the next exercise:

1. First we create a field to display IMU%, using two existing fields and some math.

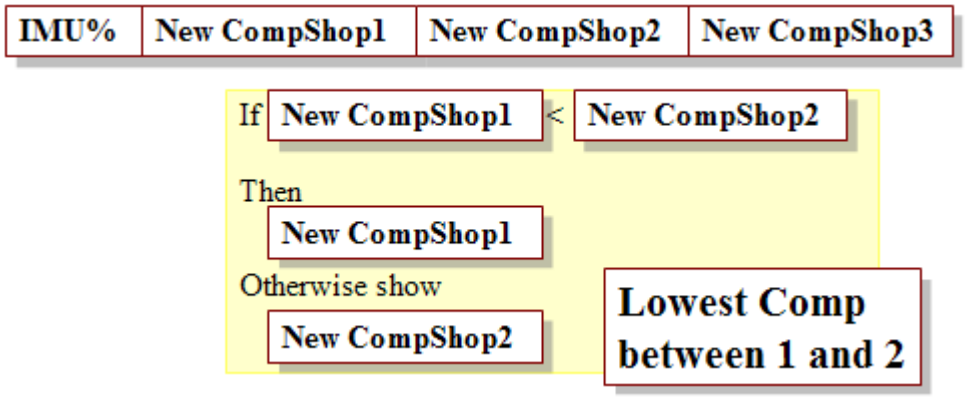




2. Next we use three existing competitive shop fields. In order to find the lowest competitive shop field, each shop that is .00 has to be converted to 9999999 (this eliminates the .00, or "no comp shop" from showing as the lowest price) using CASE Statements. Then, we'll create three new competitive shop fields.

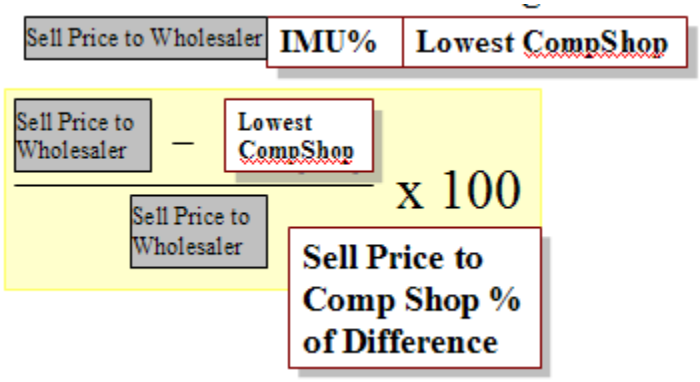


3. The next field to create compares which comp shop is the lowest. This is done in two parts. Part one involves creating a field using a CASE statement that compares *New Comp Shop 1* with *New Comp Shop 2*.

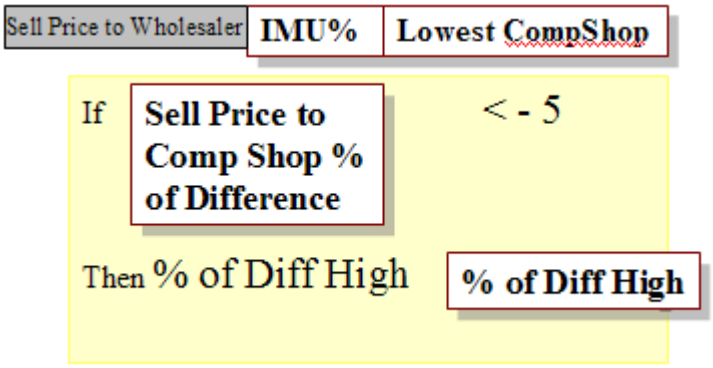


Part two involves taking the previous field created—*Lowest Comp between 1 and 2*—which does as its name implies, and comparing that with the *New Comp Shop 3* field (created in Step 2) to see which is the lowest, and form another new field—*Lowest CompShop*.

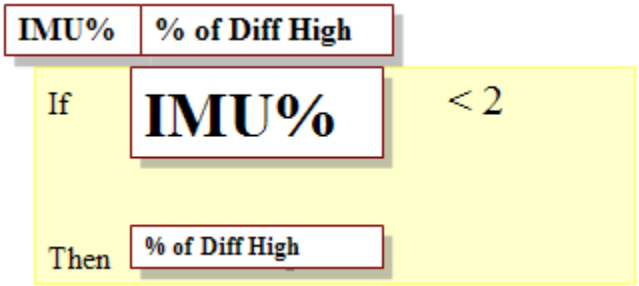
4. With this newly-created *Lowest CompShop* field, we now create another new field that takes the *Sell Price to Wholesaler* field and compares it to the *Lowest CompShop* field, calculating the percentage of difference to see if it is 5% or more. Again, this is done in two parts. Part one, creating a field that calculates the percentage of difference and part two, using that field to display the differences that are higher then 5%.



Let's now use this new field—*Sell Price to Comp Shop % of Difference*—in a Case statement to determine percentages of difference higher than -5%.



5. Our next newly-created field will use the computed IMU% field to find any IMU% amounts that are less than 2%. If there are any less than 2%, display the % of Difference High results.

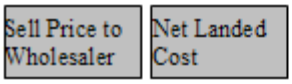


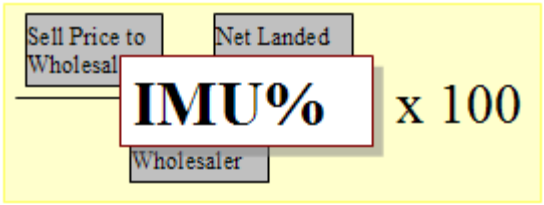
The final field is a combination of several fields, but ultimately displays when our IMU% is low and our sell price is low, compared to our comp shops.

EXERCISE 82: IMU% LOW

For this exercise we will be using three tables:
INWITMP from the *INPRDINV* library
INCMPSP from the *INPRDDTA2* Library
INWCTLP from the *INPRDINV* library

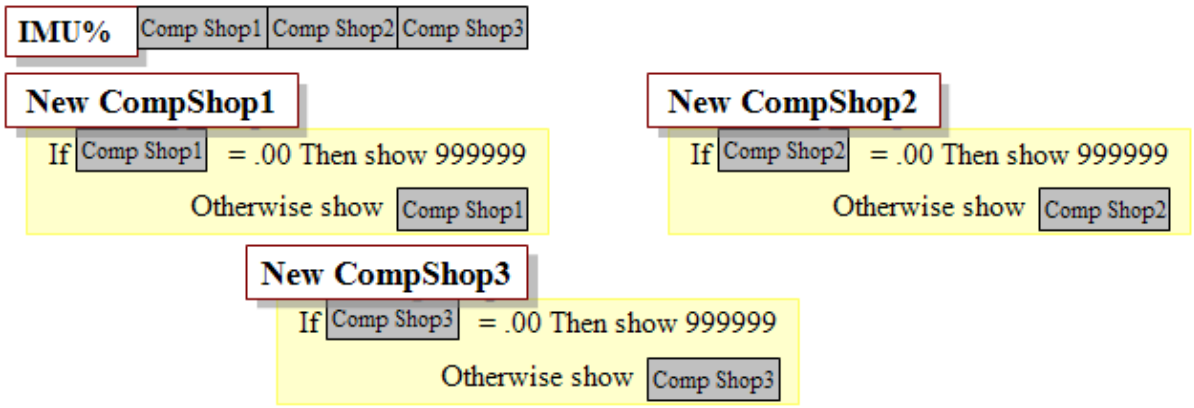
Part 1
Building the basic query and the IMU%





1. Begin by first adding *INWITMP* from *INPRDINV* to your query.
2. In the *Columns* tab, select **Buyer Number (WIBUY5)**, **Warehouse Number**, **Item Number**, **On Hand/Sell Units**, **Net Landed Cost/Selling Unit**, **Sell Price to Wholesaler**, **Weekly Sales Forecast in Sell Units**, **Weekly Receiving in Units**, and **On Order in Sell Units**.
3. Create conditions where **Company = 1** and **on hand/sell units > 5**.
4. For a test, use **warehouse = 110**.
5. Create a Prompt for department.
6. Run the query and use department 12 for the prompt.
7. View the results and take note of the record count (177).
8. Join INCMPSP to INWITMP
   - Use a left outer join for INCMPSP on **Item Number**, **Buyer Number**, **Department Number** and **Warehouse Number**.
9. Join INWCTLP to INCMPSP and INWITMP.
   - Use an inner join on **Warehouse Number**.
10. In Columns add the field *Name* from INWCTLP.
11. Create a condition that **Miscellaneous Flag 1 (WCFLG1) <> Y**.
12. Sort by buyer number in descending order.
13. Run the query and view the results.
14. Create a field that represents IMU%, which is (*Sell Price to wholesaler – Net Landed Cost/Selling Unit) / Sell price to Wholesaler * 100*.
    - Begin by clicking **New** in the *Columns* section of the Query wizard, then **General Expression**. Here you can create the IMU% formula.
15. Rename the newly-created column as **IMU%**, using *column properties* in the *Columns* window.
16. Create a condition that **IMU% > 0**.
17. Run the query and view the results.

Part 2
Create the lowest competitive shop column.



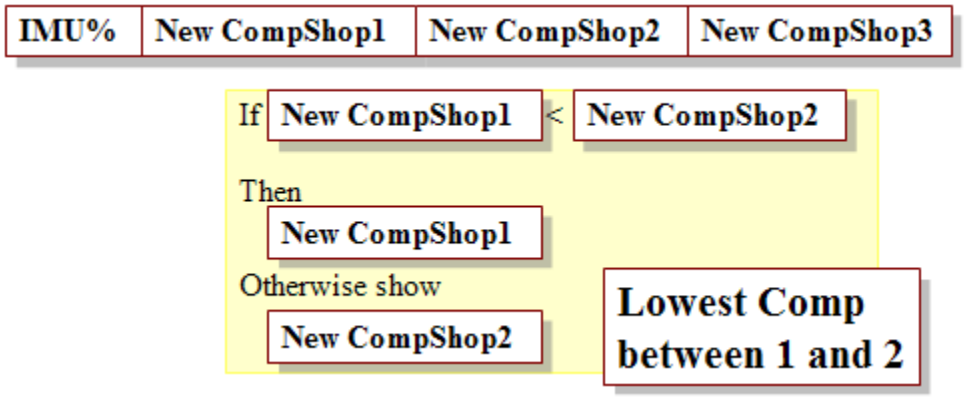
Now that your query is built, it's time to build some CASE statements.

There are three comp shop fields. We want to compare all three, and find the lowest comp shop. However, some comp shops just show .00, so we will want to exclude when the comp shop = 0. For now we will just have 0 register as 9999999, so our case statement will work.

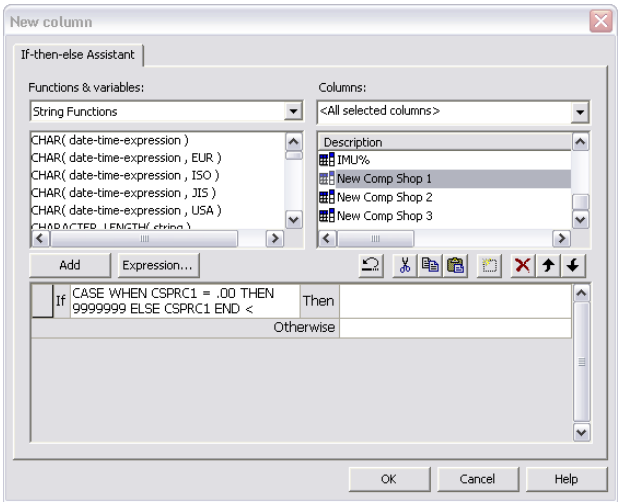
Creating the lowest comp shop field.

1. Remove test condition of Warehouse = 110.
2. Add three fields from *INCMPSP*:
   - **Competitor #1 Sell Price**.
   - **Competitor #2 Sell Price**.
   - **Competitor #3 Sell Price**.
3. Run the query and view the results.
4. Convert all .00 competitive shops to 9999999.
   - Click **New** in the *Columns* section of the Query wizard.
   - Choose the **If-then-else Assistant**.
   - In the *If* text box, add Competitor #1 Sell Price = .00.
   - In the *Then* text box, type 9999999.
   - In the *Otherwise* text box, add Competitor #1 Sell Price.
5. Repeat step 4 for competitor shops #2 and # 3.
6. Run the query and view the results.
7. With the new comp shops created, we can now delete the three competitor sell prices that were not created using the case statements.
8. Rename your three newly-created fields **New Comp Shop1**, **New Comp Shop 2**, and **New Comp Shop 3**.

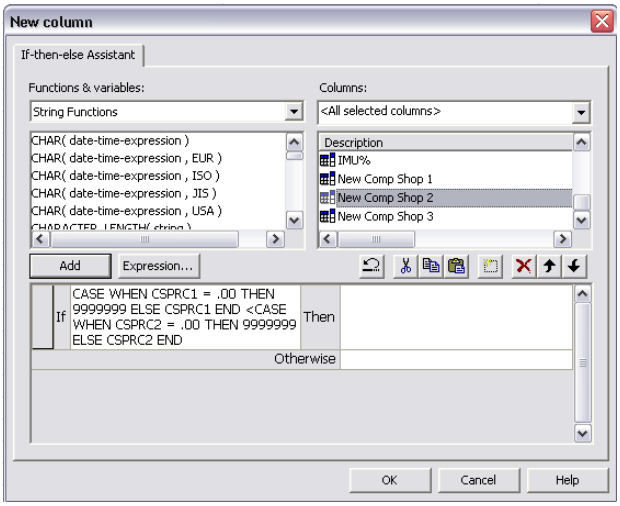
Part 3
Take these three *New Comp Shop* computed fields and find the one with the lowest price.



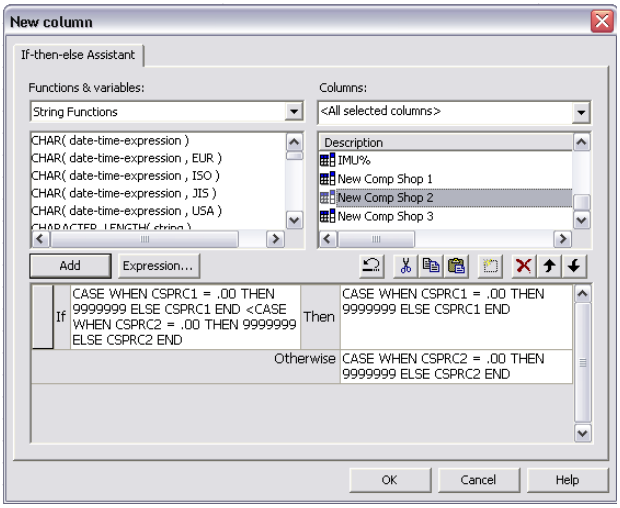
1. In the *Columns* section of the Query wizard, click on **New**.
2. Choose the **If-then-else Assistant**.
3. Under Columns choose **<All selected columns>**.
4. Double-click on **New Comp Shop 1**.
5. In the If text box type **<**.



6. Double-click on **New Comp Shop 2**.



7. Click in the *Then* text box.
8. Double-click on **New Comp Shop 1**.
9. Select the *Otherwise* text box.
10. Double-click on **New Comp Shop 2**.

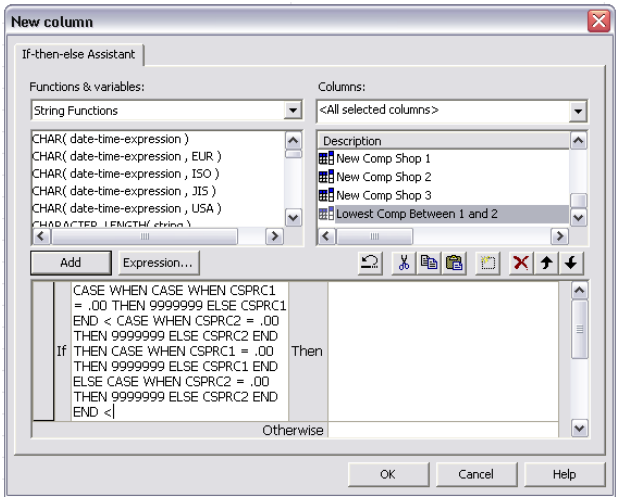


11. Run the query and view the results.
12. Rename the newly-created field **Lowest Comp Between 1 and 2**.
    - This column shows, as described, the lowest-priced comp shop between comp shop 1 and 2.

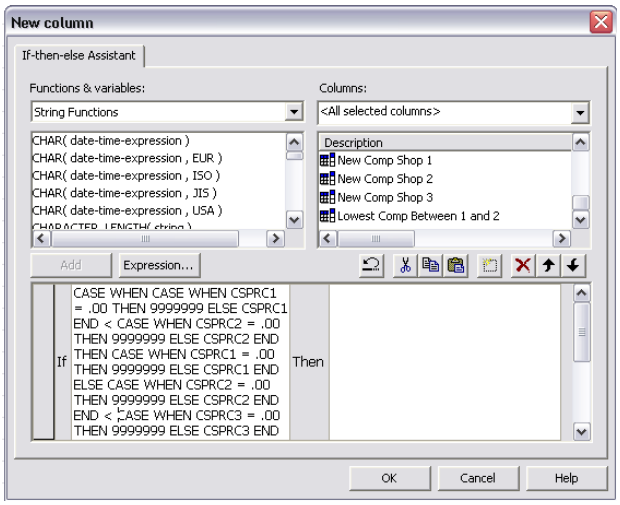
Part 4

We will create a new field that compares the *Lowest Comp Shop Between 1 and 2* field to the *New Comp Shop 3* field, to see which is lower.

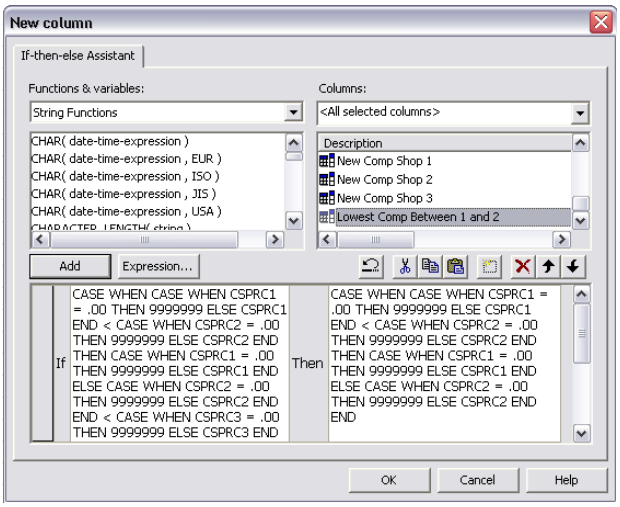
1. In the *Columns* section of the Query wizard, click on **New**.
2. Choose the **If-then-else Assistant**.
3. Under *Columns,* choose **<All selected columns>.**
4. Double-click on **Lowest Comp Between 1 and 2**.
5. In the *If* text box type **<**.



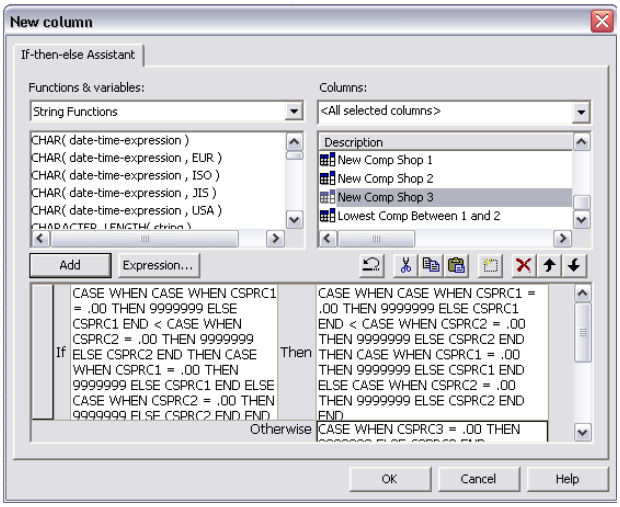
6. Double-click on **New Comp Shop 3**.



7. Click the *Then* text box.
8. Double-click on **Lowest Comp Shop Between 1 and 2**.



9. Select the *Otherwise* text box.
10. Double-click on **New Comp Shop 3**.

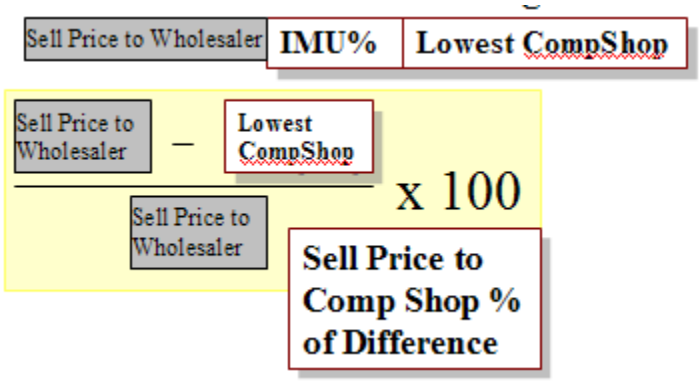


11. Run the query and view the results.
12. Rename the newly-created field **Lowest Comp Shop Overall**.
    - ➢ You now have a field that compares three comp shops and displays the lowest one.

Part 5

Create the logic "Where the percentage of difference with the lowest competitive shop price is greater than -5%."

First, create a new computed field that looks at the percentage of difference between our lowest comp shop and our sell price.



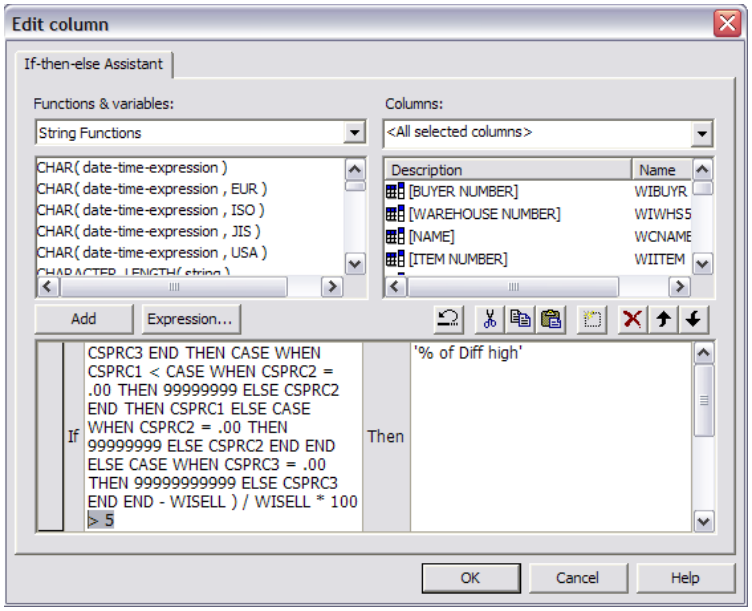
1. In the *Columns* section of the query wizard, click on **New**.
2. Choose **General Expression**.
3. Under *Columns* choose **<All selected columns>.**
4. Double-click on **Sell Price to Wholesaler**.
5. Click the **-** button.
6. Double-click on **Lowest Comp Shop Overall**.
7. Type **(** at the beginning of the statement, and **)** at the end of the statement.
8. After the closing parentheses, enter the **/** symbol.
9. Double-click on **Sell Price to Wholesaler**.
10. Enter the ***** symbol.
11. Key **100**.

- The statement will read as follows:

```
( WISELL - CASE
        WHEN
        CASE
                WHEN
                CASE
                        WHEN CSPRC1 = .00 THEN 9999999
                        ELSE CSPRC1
                END <
                CASE
                        WHEN CSPRC2 = .00 THEN 9999999
                        ELSE CSPRC2
                END THEN
                CASE
                        WHEN CSPRC1 = .00 THEN 9999999
                        ELSE CSPRC1
                END
                ELSE
                CASE
                        WHEN CSPRC2 = .00 THEN 9999999
                        ELSE CSPRC2
                END
        END <
        CASE
                WHEN CSPRC3 = .00 THEN 999999999
                ELSE CSPRC3
        END THEN
        CASE
                WHEN
                CASE
                        WHEN CSPRC1 = .00 THEN 9999999
                        ELSE CSPRC1
                END <
                CASE
                        WHEN CSPRC2 = .00 THEN 9999999
                        ELSE CSPRC2
                END THEN
                CASE
                        WHEN CSPRC1 = .00 THEN 9999999
                        ELSE CSPRC1
                END
                ELSE
                CASE
                        WHEN CSPRC2 = .00 THEN 9999999
                        ELSE CSPRC2
                END
        END
        ELSE
        CASE
                WHEN CSPRC3 = .00 THEN 999999999
                ELSE CSPRC3
```

END
END) / WISELL * 100

12. Rename the newly-created field **Sell Price to Comp % of Diff**.
13. Run the query and view the results.
    - This calculates the percentage of difference between the Costco's sell price and the lowest competitive shop price.



.

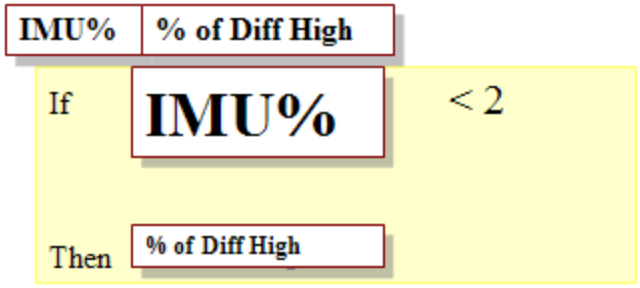
14. Create a new condition on this field to be greater than -100.
    - This will remove those comp shops that are .00

Next, create a new computed field to clean up the query by getting rid of all the 9999999 entries.

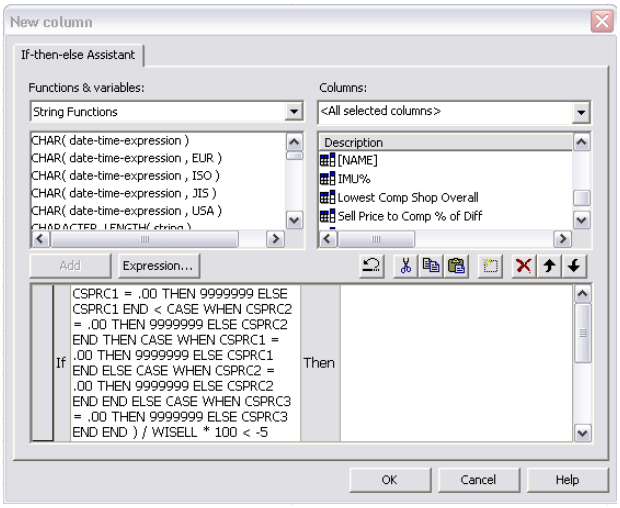
1. In the *Columns* section of the Query wizard, click on **New** .
2. Choose the **If-then-else Assistant**.
3. Under *Columns*, choose **<All selected columns>**.
4. In the *If* text box add **Sell Price to Comp % of Diff**.
5. At the end of the string in the *If* text box, type **< -5**.
6. Click in the *Then* text box.
7. Type **'% of Diff high'**.
8. Click **OK**.
9. Run the query and view the results.
10. Rename the newly-created field **% of Diff High**.

Part Six

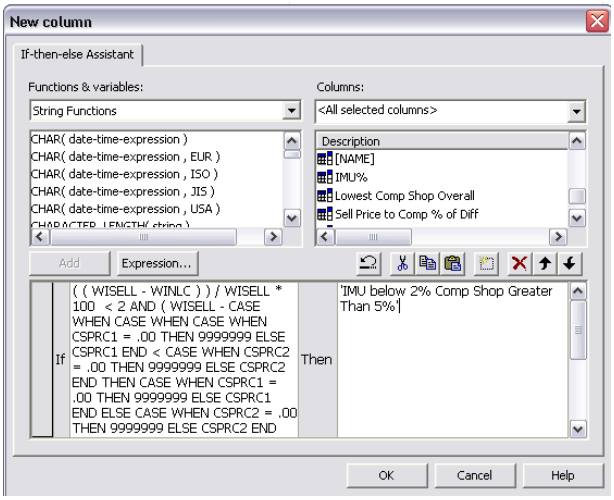
Finally, the last column to create will compare the *% of Diff High* with the *IMU%* to display when our IMU% is less than 2% and our percentage of difference with the lowest competitive shop is higher than 5%.



1. In the *Columns* section of the Query wizard, click on **New**.
2. Choose the **If-then-else Assistant**.
3. Under *Columns* choose **<All selected columns>**.
4. Add the **IMU%** field in the *If* text box.
5. After the IMU% formula type **< 2**.
6. Type **AND**.
7. Add the **Sell Price to Comp % of Diff** field in the *If* text box.
8. To the same string, add **< -5**.



9. In the Then box, type **'IMU below 2% Comp Shop Greater Than 5%'**.



10. Select **OK**.
11. Review query.
12. Rename the newly-created field **Action Required**.

Now, let's get rid of all the Nulls.

1. In the *Conditions* window, choose **<All selected columns>**.
2. Choose the **Action Required** field.
3. Select the condition **<>** and type the value **NULL**.
4. Choose the **Lowest Comp Shop Overall** field and the condition of **<** then type the value **100**.
5. Select **OK**.
6. Run the query and view the results.
   - You can now delete the **New Comp Shop 1**, **New Comp Shop 2**, **New Comp Shop 3**, **Lowest Comp Between 1 and 2**, and **% of Diff High** columns and the query will continue to run.

7. Sort **Sell Price to Comp % of Diff** in *Ascending* order, then move that column to the top of the *Added sort columns* area.

.

UNIT XI: SQL TRICKS

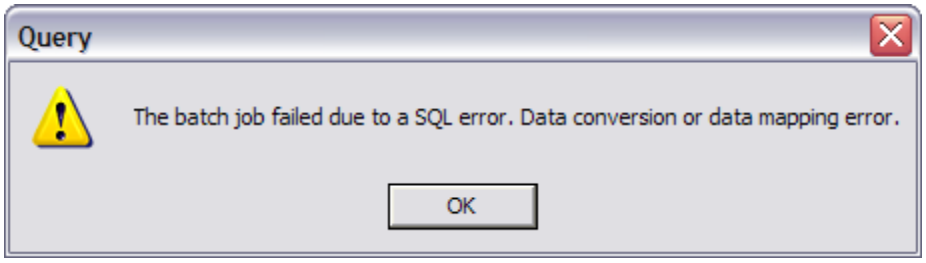
Avoid Dividing by Zero
Search for a String Across Columns
Ranking
Running Queries over Excel

SQL RECIPES

Query has a lot of tools for creating not just reports, but straightforward answers that help drive our business. Here are just a few ways to combine the tools together to create dynamic reporting. The Query courses that have been created will get you started. You could create a book or a set of courses about how you have used Query to answer the questions that keep you or your managers awake at night. With so many tools and flexibility, there is little limitation to what Query can do for you. Be sure to research such tricks as; Creating a running total, Subqueries, or Pivoting results.

Avoid Dividing by Zero

There are several cases where dividing by zero could cause this error message; for example, when you are trying to create a percentage:



Search for a String across Columns

When you are looking for a value across a number of columns, it can be searched for automatically rather than manually or individually. Here is a simple example of members who purchased quantities of items in different departments.

Name	Dept12	Dept24	Dept17
Justin	5	0	9
Sarah	1	2	25
Carlos	15	0	6
Chris	8	1	0
Dwayne	0	0	2
Tori	1	7	11
Naomi	13	3	14
Esha	7	10	10
Dwayne	16	2	9

Take a moment to determine how many card holders have purchased 11 or more items in just one department, no matter what the department number is. How many members have purchased 11 or more items in a single department?

This is a simple example, but it still took a moment to review. Now imagine you had 20 columns and 2000 records to go through. An automated way to search for a value in multiple columns would be helpful.

EXERCISE 83: FIND TIM

In this example, there are several different managers at various levels. You remember that the manager's name was Tim, but that's all. What you need to do is combine all of the warehouse fields, and search through the concat.

1. Join *INPRDDTA.INRORGP* to *INPRDINV.INWCTLP*. Considering the results will show warehouse phone numbers and warehouse managers, what would be the best join to use? Should you use more then one? If you are not sure, what is the best action to take?
2. Select **Warehouse Number**, **Warehouse Phone Number**, **Warehouse Manager**, **Assistant Manager1**, **Assistant Manager2** and **Admin Manager**.
3. Run the query and view the results. Can you find all the Managers with the name Tim? It wouldd take quite awhile, wouldn't it? Let's sort for Tim.
4. In the *Columns* section, create a new column using **General Expression**.
5. Concat all of the Manager fields.
6. In the *Conditions* section, use this new field to create an *is like* condition, looking for the string TIM.

Note: Keep in mind that TIM might be at the beginning, middle, or end of the concatenated string.

7. Run the query and view the results.
8. At this point you can delete the created field.

Note: Although this field can be deleted, the Query application will not accept a copy-and-paste of the created field in the conditions (i.e., you cannot create a new query by SQL and just paste the statement in with the created field missing).

Challenge:
Can you make this better with CASE statements or prompts? Modify this query to suit your own needs. Can you apply this concept to other queries, reports, or business needs you have?
NOTES:

HINT:
The join is on Region Code and Warehouse Number.

EXERCISE 84: Finding the Difference Between Week 1 and Current Week Sales.

First, we'll create a basic query, including creating one unique field that calculates on-hands minus on-hands in RTV:

1. Create a query using the library *INPRDDTA* and the *INWWIIP* table.
2. Select **ZXWHSE**, **ZXDEPT**, **ZXCAT1**, **ZXVEND**, **ZXSUFF**, **ZXITEM**, **ZXDES1**, **ZXSLUC**, and **ZXSLS1**.
3. Create a new column by subtracting *ZXNSIU* from *ZXOHUN*.
4. Create conditions where **Company = 1, Region Code = NW**, **Warehouse = 110, Department =17**, and **Primary Vendor = Y**.
5. Sort by department in ascending order, then warehouse number in ascending order.
6. Run the query and view the results.

Next, we'll compare the current weeks' sales with Week 1 sales:

Note: Current weeks' sales is being used to generate zeros for this exercise.

1. In the *Columns* window, select **New**.
2. Choose **General Expression**.
3. Subtract *Current Week Sales* from *Week 1 Sales*.
4. Rename this new field using column properties as **Sales Diff**.
5. Run the query and view the results.

This demonstrates that the fields work fine with math functions.

Finally, we'll find the growth (loss) percentage between current weeks' sales and previous weeks' sales.

1. In the *Columns* window, select **New**.
2. Choose General Expression.
3. Divide *Current Week Sales* by *Week 1 Sales*, and then multiply by 100. This will produce a data mapping error in batch processing mode.

4. Now that you've produced the error, delete the field you just created in step 3, and create it correctly via *NULLIF*. Use the following logic: **ZXSLUC / NULLIF(ZXSLS1, 0) * 100**.

Ranking

Sorting can organize your sales by rank easily enough, but what if you want to show the rank position for more than one column? RANK() allows for ranking of multiple columns.

EXERCISE 85: Rank 4 Weeks of Sales

1. Join *INPRDDTA.INIHSTP* to *INPRDDTA.INITMMP* by **Company Number** and **Item Number**.
   Question: Why does this query need to be joined by Company Number?
2. Select **Fiscal Year**, **Item Number**, **Item Description**, **Week 1 Sales in Units**, **Week 2 Sales in Units**, **Week 3 Sales in Units**, and **Week 4 Sales in Units**.
3. Create conditions where **Company Number = 1**, **Fiscal Year = 10**, **Warehouse Number = 110**, **Department Number = 61**, **Week 1 Sales in Units > 0**, **Week 2 Sales in Units > 0**, **Week 3 Sales in Units > 0**, and **Week 4 Sales in Units > 0**.
4. Run the query and view the results.
5. Open the *Columns* section of the Query wizard.
6. Highlight *Week 1 Sales in Units*.
7. Create a new column using **General Expression**.
8. In the *New column* text box, type: **RANK( ) OVER(ORDER BY IHSU01 DESC)**.
9. Run the query and view the results.
10. Create ranks for the other weeks:
    - RANK( ) OVER(ORDER BY IHSU02 DESC)
    - RANK( ) OVER(ORDER BY IHSU03 DESC)
    - RANK( ) OVER(ORDER BY IHSU04 DESC)
11. Run the query and view the results.

Creating conditions on the *new rank* columns does not function in Showcase Query. However, writing the results to a temporary table opens up several possibilities.
How could ranking be used to show the effectiveness of items using coupons, or on the fence, or end caps? Could this information be useful in negotiating with vendors? Could ranking be used to create relationships between items? When salmon ranks in the top 5 for the meat department, do wine sales go up? Is this related to road shows? If not, can this help with scheduling the next road show?

Think about ranking and your department. What are some ways ranking could be used to help you uncover trends or useful information?

NOTES:

Running Query over Excel

Now you can capture information from the Internet from competitors to demographics to weather, and run queries to see how the information relates to our business.

EXERCISE 86: Running Query over Excel

1. Open Excel.
2. Create a spreadsheet with the following weather information:

Day	Weather
Monday	Sunny
Tuesday	Cloudy
Wednesday	Raining
Thursday	Snow
Friday	Snow/Rain
Saturday	Sunny
Sunday	Partly Sunny

3. Highlight all of the cells.
4. Select the Insert menu.
5. Choose Define.
6. Click on Add.
7. Click on OK.
8. Save the Excel spreadsheet.
9. Open Query.
10. Connect to Excel Files, rather then to EIS400..
11. Browse to the location of the new spreadsheet.
12. Use the drop-down box for Tables, and choose Defined Table.
13. Click on Add.
14. Click Next.
15. Add your columns and run the query.

Modify this query to suit your own needs. Can you apply this concept to other queries, reports, or business needs you have?
NOTES:

RUNNING QUERIES TO EXCEL

There are several reasons why you might want to run your queries to an Excel spreadsheet. Perhaps you want to manipulate your data with formulas, create links,

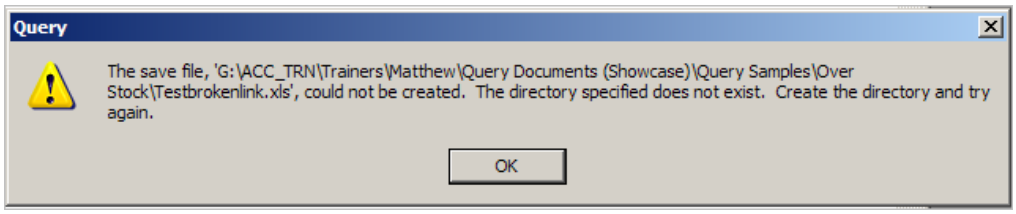
macros, or just create a more visually engaging report. Whatever the case, there are some pitfalls to avoid when running your queries to Excel.

Showing results in Excel - a review from Query Fundamentals

1. Select *Enable batch processing.*
2. Select *result options.*
3. Within *result options* select *display results in viewer,* and then choose *Excel worksheet.*
4. Select *write results to a file.*
5. Click on *options.*
6. Click on the list box arrow to browse to your selected spreadsheet.

Troubleshooting Query to Excel

Broken links



Spreadsheets should be located in a file or folder that you do not plan on moving. When running a query directly to a specific spreadsheet, a link is created to the location of that spreadsheet. If the spreadsheet (or folder the spreadsheet resides in) is renamed, the link will be broken. Your link is like your address. If you move your file, you will change your address and break the link. To fix a broken link, follow the steps you originally used to have the query write to a spreadsheet.

Excel version conflicts

Rather than having Query display results in the Query viewer, the results can be displayed in Excel. As Query is upgraded or Excel is upgraded, Excel version conflicts can occur. For example:

- A query called EXCELCONFLICTS opens results in Excel 4.0 that is called TESTSPREADSHEET.XLS.
- Query then gets an upgrade, and defaults to an Excel 8.0 viewer.
- Now, when running the EXCELCONFLICTS query, it will open as an Excel 8.0 version inside of the 4.0 version of the saved spreadsheet called TESTSPREADSHEET.XLS.
- This causes a problem because 4.0 is an older version of Excel. Trying to work with conflicting versions, the application will crash.

Solution:

1. Open the TESTSPREADSHEET.XLS without Query.
2. Save as the latest version of Excel.

RUNNING QUERY IN EXCEL

Using Query add-ins, you can access queries from within Excel. Any formatting or reports that you create are retained each time a query is run; only the data changes.

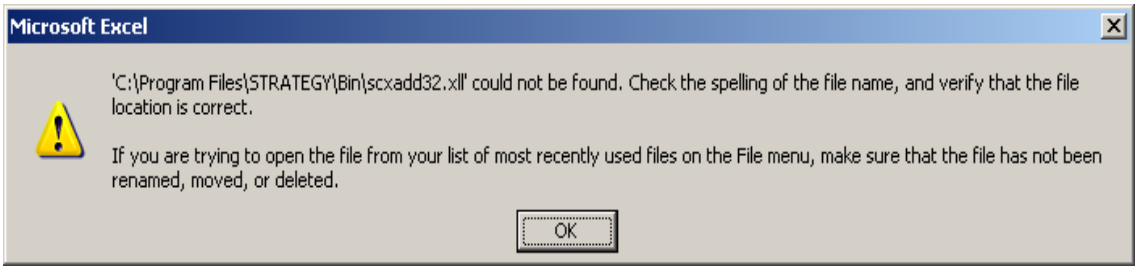
Also, any time you open a workbook, you have the option of refreshing the data via all queries or individual queries that have been embedded in the report.

Before running queries in Excel, verify that you have the Showcase add-in.

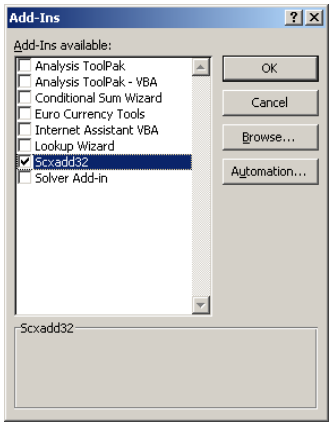
Adding Showcase in Excel

These steps were created with Query 8.0. With different installs and versions of Query, these steps may change slightly. Consult the Helpdesk for assistance with these steps if you are using a newer version.

1. Open Excel. If you still have the old version of the add-in loaded, you will receive this error message:

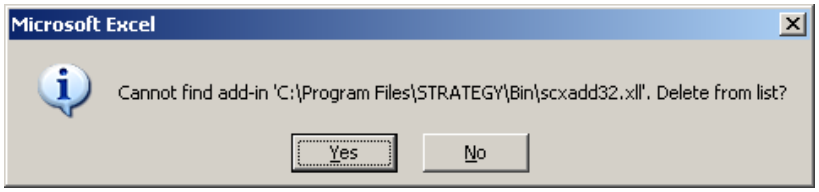


2. The solution is to remove the old add-in. Select *Tools* and then *Add-Ins*.



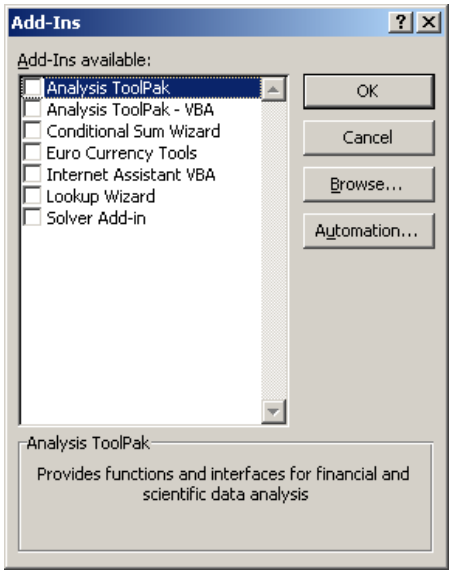
3. Uncheck “Scxadd32” if this is not in your Add-ins and continue to the next step.

4. Click Yes to the message seen below, and then Click OK. The old add-in has now been removed from Excel.

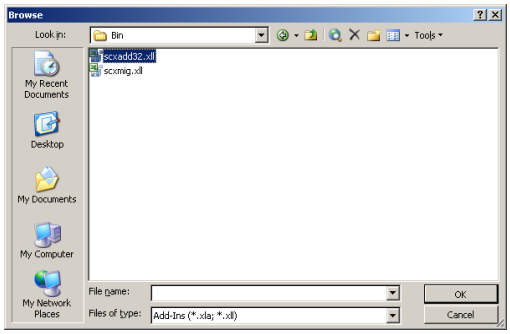


5. Open Excel.

6. To add the new Showcase add-in, open the Tools menu, scroll down to Add-Ins, and then click the Browse button.

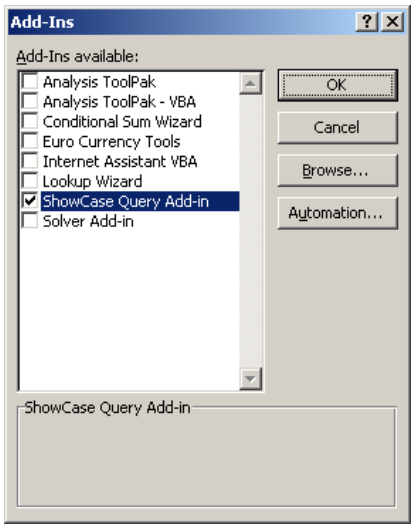


7. Click on the drop-down arrow and browse to the following location: C:\Program Files\Showcase Suite\Bin.



8. Highlight the following file “Scxadd32.xll” and click the OK button.

9. The “ShowCase Query Add-in” add-in should now be available. Place a check mark next to it, and then click the OK button.



10. The Showcase menu should now be available as an Excel menu option.

The Hyperion and Showcase add-ins have been known to not work together in Excel. Hyperion may need to be disabled before the Showcase add-in will work.

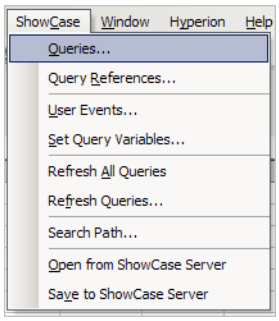
Running Multiple Queries in Excel

Before embedding Queries into Excel, create and test your queries to make sure they work, and then store the queries in a secure folder. Do not change the name and location of the folder. If your query moves, or the folder is renamed, the report will have to be re-linked to the new location or name. Now you are ready to embed queries into Excel.

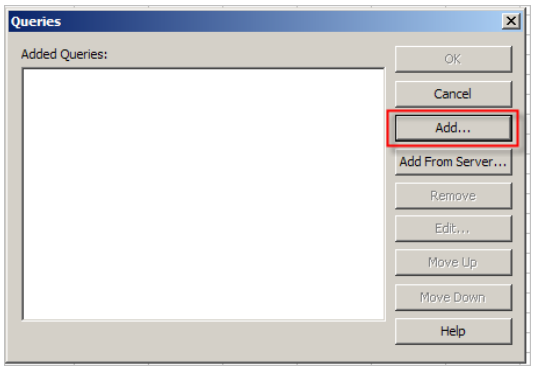
Adding the query as a reference in Excel

1. Open Excel.
2. Click on the *Showcase menu*.

3. Within the *Showcase menu*, select the *Queries* option.

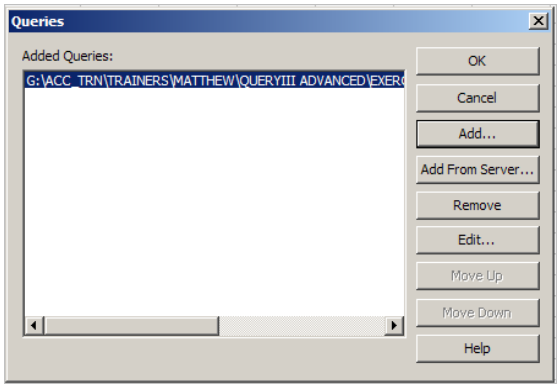


4. An *Added Queries* window will appear.

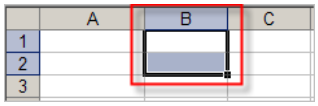


5. Select the *Add* button.
    a. A Windows browser will appear. This will give you the ability to browse to where your query is located.
6. Locate the correct query.
7. Click the Open button.
8. A prompt will appear for your iSeries400 profile name and password. Enter your profile name and password.
9. Your query will appear. Click OK.

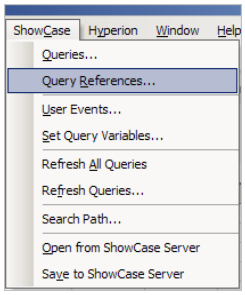
10. You can add more queries by clicking on the *Add* button, or proceed to your spreadsheet by clicking on the OK button.



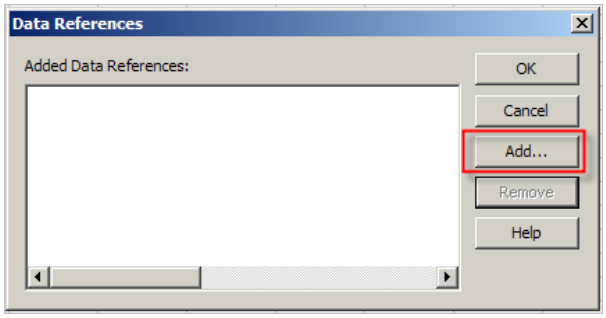
11. In your spreadsheet highlight where you would like to place your data. If the data will extend past one row, select two rows for each field that is to be added.



12. Select the *Showcase* Menu and the *Query References* option. This will open the *Data References* window.

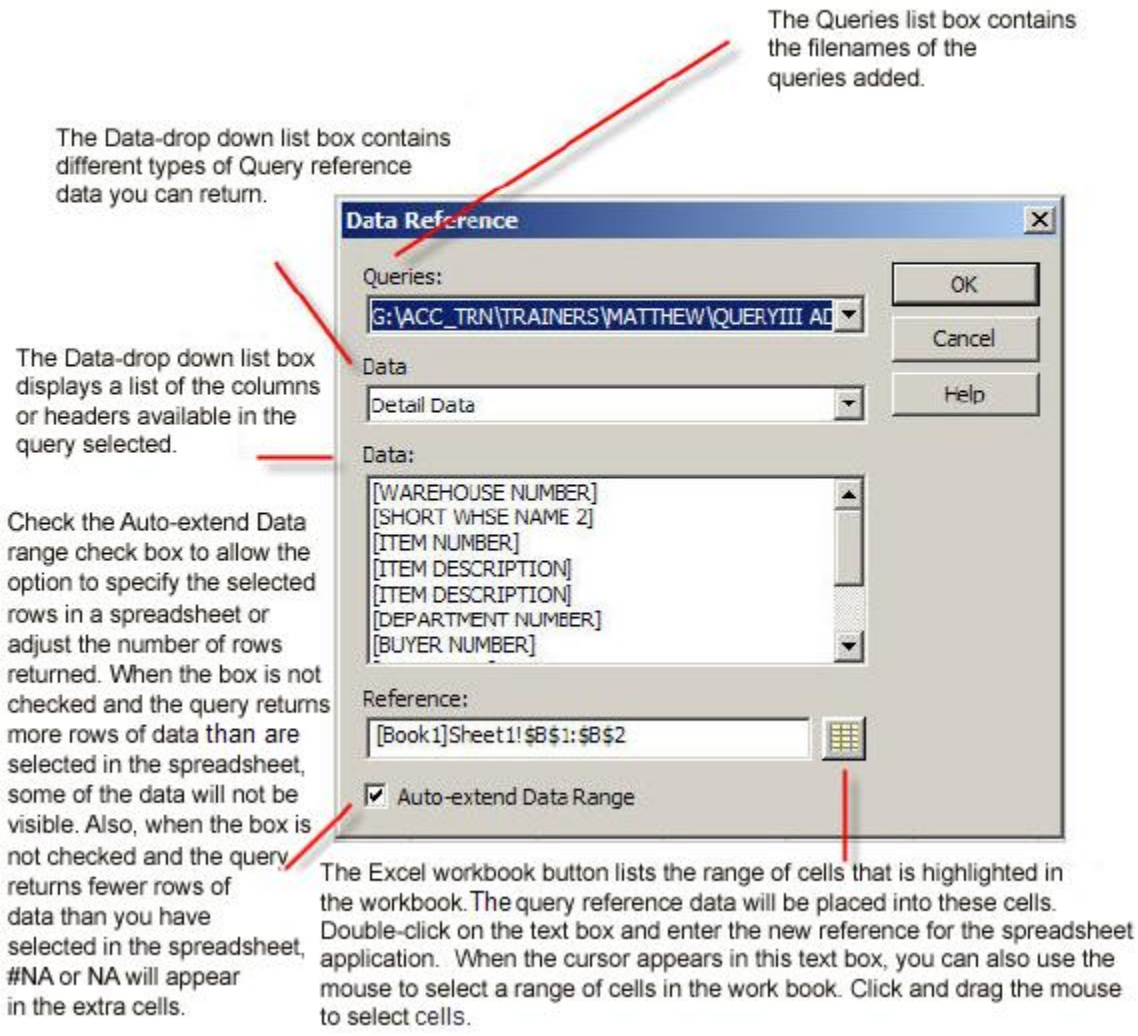


13. In *Data References* choose the *Add* button.



This will open the *Data Reference* window.

DATA REFERENCE WINDOW



Data
The data drop-down list box contains different types of Query references data you can retrieve: Extendable Detail Area, Query Viewer Data, Extendable Query Viewer Area, Column Headings, Variables, and Query Properties. Detail Data is the default, and is the most used data type.

EXTENDABLE DETAIL AREA
When you select Extendable Detail Area in the Data References window, Query does not actually return any query data to the cells referenced, but it will adjust the number of rows selected in the spreadsheet to match the number of rows of detail data returned.

QUERY VIEWER DATA
By selecting Viewer Data, Query will return any combination of break group and summary information. Formatting is not applied.

EXTENDABLE QUERY VIEWER AREA
When selecting Extendable Viewer Area, Query does not return query data to the cells referenced, however the number of rows selected will automatically adjust to match the number of rows of data returned.

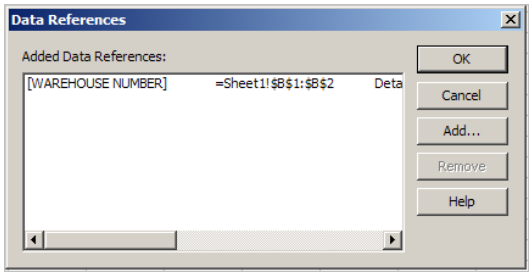
COLUMN HEADINGS
When you select Column Headings, Query will return the column headings. The Auto-Extend Data Range check box is disabled.

VARIABLES
When you select Variables, Query will return a list of local and global variables for the selected query. The Auto-Extend Data Range check box is disabled.

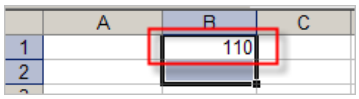
QUERY PROPERTIES
When you select Query Properties, Query will return the properties you request from the selected query. The Auto-Extend Data Range check box is disabled.

Value	Information Query Returns
Property.Names	A list of all the query properties that can be requested
Property.Description	The description information in the Query Summary Info dialog box
Property.LastSql	The last SELECT statement in the Query Summary Info dialog box
Property.RunStatus	The status of the query. It can have these values: 0 Has not run 1 Success, it is not running and the last run was successful 2 Running 3 Failed, user canceled 4 Failed, caused by error
Property.OutputFile	The filename that the last run saved data to
Property.BatchQuery	Returns 0 if the query is interactive, 1 if the query is to run in batch
Property.SubmittedToBatch	Returns 1 if the batch query has been submitted, 0 if not submitted or if it's an interactive query

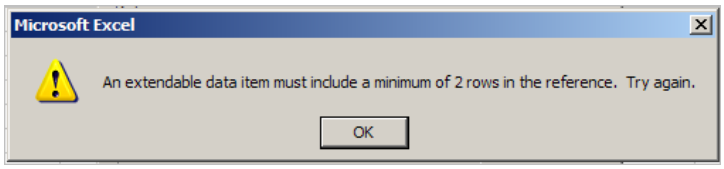
14. By selecting a field and clicking on the *OK* button, you will have added a Data Reference from your Query into your spreadsheet.



15. Click on *OK* to go back to the spreadsheet, where you can see your results.



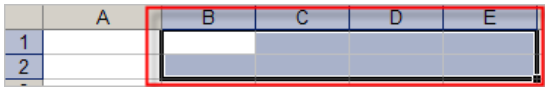
ERROR MESSAGE RECEIVED WHEN MORE THAN ONE ROW IS NOT SELECTED



Selecting two rows allows the reference to extend multiple rows. For more information on extendable rows, see **EXTENDABLE QUERY VIEWER AREA.**

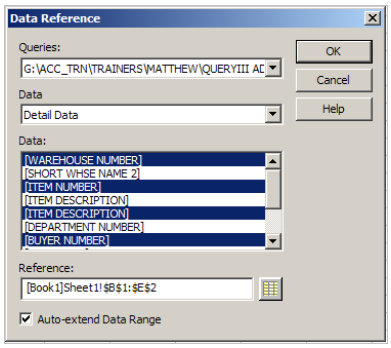
Adding Multiple References or Fields

1. Open Excel. Highlight by selecting four columns and two rows.

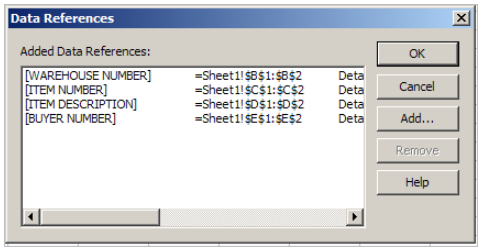


2. Open the Showcase menu, and choose the Query References option.

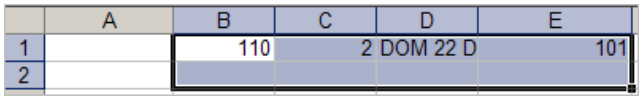
3. Choose four fields from the Data Reference window.



4. Click on the OK button, and verify that all four fields were selected.



5. Click OK again to see the results in Excel.

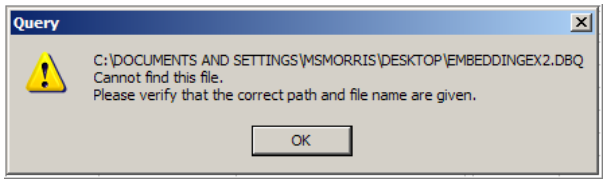


	A	B	C	D	E
1		110		2 DOM 22 D	101
2					

6. Save your spreadsheet. The next time this report needs to be run, simply open the report, go to the Queries menu, and select Refresh all Queries.

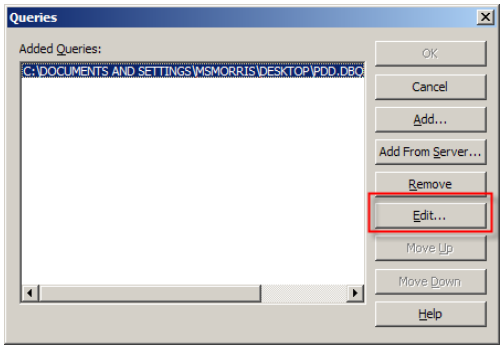
Re-linking Query Reference

If a query is moved or renamed, the link will be lost. An error message will display the next time the query is run.



To re-link the query, edit the link so it will point to the new location.

1. In the Excel spreadsheet choose the Showcase menu.
2. Within the Showcase menu, choose the Queries link.
3. Highlight the query that needs to be re-linked, and select the edit button.

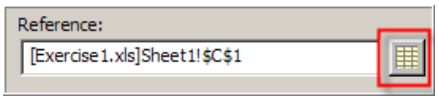


4. Browse and then link to the new location of the query.

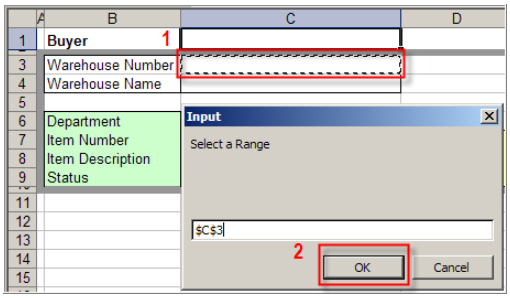
EXERCISE 87: Embedding a Query into Excel

In this exercise you will link specific fields from a query to cells in a spreadsheet.

1. Open the Exercise1.xls spreadsheet, found in the Exercise1 folder.
2. Verify that the Showcase menu is added. If the Showcase menu is not added, follow the directions outlined in the *Adding Showcase to Excel* section.
3. Add EMBEDDINGEX1.DBQ as a reference to the new spreadsheet.
4. Highlight cell C1.
5. Select Query References from the Showcase Menu.
6. Select the Add button.
7. Uncheck Auto-Extend Data Range box.
8. Scroll through the Data window to the BUYER NUMBER field. Highlight that field.
9. Select the OK button.
10. The Data References window will appear.
11. Click on the *Add* button.
12. Select the Excel spreadsheet icon.



13. Click on the C3 cell.
14. Click OK in the Input window.



15. Uncheck the Auto-Extend Data Range box.
16. Highlight the SHORT WHSE NAME 2 field.
17. Select the OK button.
18. Click on the *Add* button.
19. Repeat the above steps for each of the fields represented in the Excel worksheet.
20. When all fields in the Excel spreadsheet have been referenced to the query, click OK in the Query References window.
21. From the Showcase menu, select Refresh Queries.
22. Highlight the query to be refreshed.
23. Select the OK button.
24. Enter 110 for the warehouse and 2 for the item in the prompt.
25. Review the results in the spreadsheet.
26. Refresh all queries again and try other warehouses and items.

Exercise 88: Creating an Extendable and Non-Extendable Report

In this exercise, you will create a post distribution depot report by using two queries. The first query will retrieve data used to fill in the prompts on the second query. The second query will act as a reference for headers and extendable data fields in the report.

1. Run the Ex2Step1.DBQ in the Exercise 2 folder. This Query is only for getting data that you will use in the next steps.
2. At the query prompt, type in yesterday's date and the current day as the date range. The format for the date field is CYYMMDD or 1100601 for June 1, 2010.
3. Copy a few records into word for use later. This data will be used to fill in the prompts for EXstep2.DBQ.
4. Open Ex2Results.xls.
5. Remove any existing queries:
    a. Click on the Showcase menu.
    b. Click on Queries.
    c. Highlight any queries that exist.
    d. Click on the Remove button.
6. Add Ex2Step2.DBQ as a reference to the new spreadsheet.
7. Use the data reference window to create these data references:

    a. WAREHOUSE NUMBER (PDAWH5) to Cell B3 as non-extendable.
    b. PO NO-WHSE NO PRFX WHEN PRT to Cell D3 as non-extendable.
    c. DEPARTMENT NUMBER to Cell F3 as non-extendable.
    d. VENDOR NUMBER – VENDOR SUFFIX to Cell H3 as non-extendable.
    e. ITEM NUMBER to Cell B6 as non-extendable.
    f. QUANTITY SHIPPED to Cell C6 as non-extendable.
    g. FIRST VIA WHSE REC DATE to Cell D6 as non-extendable.
    h. QTY REC BY VIA WHSE to Cell E6 as non-extendable.

    i. WAREHOUSE NUMBER (PSWHS5) A8 to A9 as extendable.
    j. ITEM NUMBER B8 to B9 as extendable.
    k. QTY RECEIVED IN SELL UNITS C8 to C9 as extendable.
    l. DATE SHIPPED D8 to D9 as extendable.
    m. NET LANDED COST/SELLING UNIT E8 to E9 as extendable.
    n. SELL PRICE TO WHOLESALER F8 to F9 as extendable.
    o. SELL PRICE TO WHOLESALER –NET LANDED COST/SELLING UNIT / SELL PRICE TO WHOLESALER G8 to G9 as extendable.

8. When all fields in the Excel spreadsheet are referenced to the query, click OK.
9. From the Showcase menu select Refresh Queries.
10. Highlight the query to be refreshed.
11. Select the OK button.

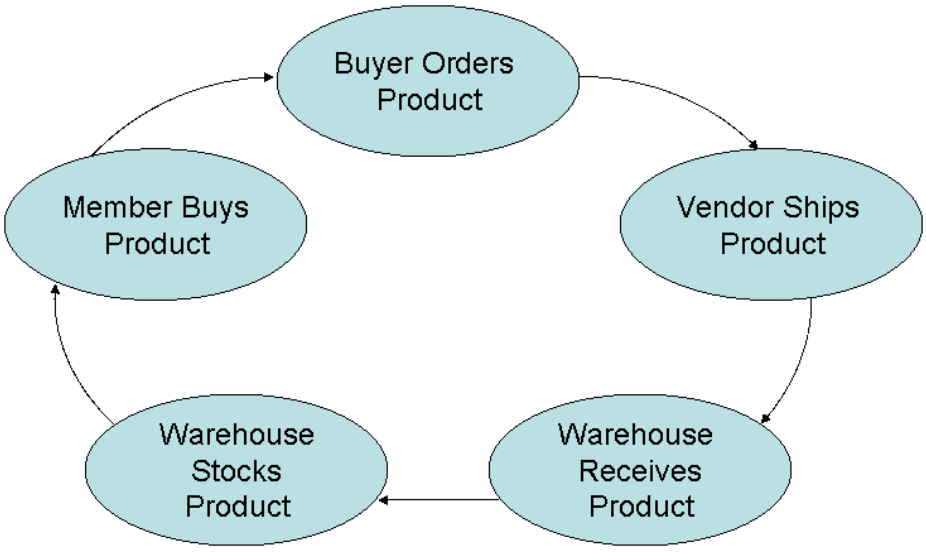
12. Enter data from Step 3 for the prompts.
13. Review the results in the spreadsheet

UNIT XII: BUSINESS INTELLIGENCE

CASE STUDIES
BUSINESS INTELLIGENCE

Improved efficiency ties directly to cost savings. What does that mean for Query users? It means removing redundant reports, automating manual reporting, and researching and creating time-sensitive, useful reports that require action. We use Query to create powerful reports that don't currently exist. Using these reports, we can better analyze our business to work smarter and give us a competitive edge.

Business is about events that occur and responses to those events:



- How effective are we when doing this?

- How quick are we about doing this?
- Was the shipment on time? If not, when do we know? How quickly can we respond?
- Is the quantity correct?
- What needs to be adjusted?

With something as simple as shipping product, many actions are taken based on a variety of conditions and reports. Are we meeting the Six Rights of Merchandising? Processing information faster for more informed decision making supports actions that support our business model, while growing our business and giving us a competitive edge. How quickly can a location communicate new information to the corporate or regional offices? Then, how long does it take us to get that information processed, researched, and obtain a decision and take action? In short, how long is it between the time that action impacts the location and the discovery of the original information?

Here is a look at Kmart and Walmart showing the success and failure of investing in business intelligence. Let's take a brief look at them:

Kmart

Kmart once enjoyed strong standing in the marketplace before struggling with bankruptcy, and finally merging with Sears. What happened? Kmart's strategy was to offer low-price, high-quality products. Kmart's positive public image helped the company maintain their position in the market as long as possible, before the company finally succumbing to their internal failings.

Some of Kmart's primary flaws involved their shipment processes, excess inventory, shrinkage, and an inability to target specific customers. The company used the shotgun approach of, "Everyone is my customer." Another important aspect for their failure was the company's inability to choose feasible locations, which greatly affected their image.

Kmart's IT had been tagged as ancient and incompetent. The flow of data and information in the network of more than 2,000 stores was considered slow, and left the company "behind the times."[1].

The main objectives of Kmart's IT division were to cut costs and reduce waste. It would have been better if Kmart had been able to focus on managing the supply and demand of their products that would help them to foretell demand: Using IT to ensure you have the right products, at the right time, in the right quantity, and the right condition.

Once a mainstay in the market place, Kmart—even after its merger with Sears — is still not what they once were[2]. Although Kmart works with marketing, management, sales, orders, pricing, and all other aspects that their competitors do, they worked *harder* and not smarter.

[1] Turner, 2003, p. 121

[2] In November 2009, Kmart reported its first year-over-year sales increase of 0.5% since 2005, and only the second such increase since 2001. – Reuters, November 19, 2009

Walmart

In contrast to Kmart, Walmart believes their technology is a strategic competitive advantage over their competitors. The company shows their technological focus by investing more than their competitors in technology. Their philosophy is "the more our business learns about our information, the more we learn about the business itself, and the more information we need." As data at Walmart became more accurate, understandable, and dynamic, the desire for information has been driven even higher. Many basic merchandising information needs can take 80% of a merchants' time to perform. With the introduction of business intelligence tools, those times can be reduced to 20% of a merchants' time.

Walmart's business-intelligence model is such that they won't rest until they have all company data available for analysis. According to Paul Westerman, a key designer in Walmart's data-warehouse design, "they will give all of this information to their internal buyers and external suppliers to exploit and demand measurable improvement," because without business sponsorship, there is no chance for business intelligence growth. Ralph Kimball's book *Data Warehouse Lifecycle Toolkit* puts an immediate emphasis on business understanding the importance of data-warehouse tools as related to the improvement of business. Understanding the importance of tools like Query must be appreciated by the business. The ability to translate your business knowledge into a business-intelligence tool allows the ability to become more accurate and agile with decision making and course corrections. *Informed decisions first require being informed*, because the better the information is, the better the decision will be. And the faster the information is received, the faster that action can be taken.

Kmart vs Walmart

In its prime, Kmart was the industry leader over Walmart. However, Walmart now clearly leads Kmart, in part because Walmart has been able to use IT to their advantage. Walmart uses their IT resources to coordinate data to help them with important aspects of their business, such as:

- Marketing
- Management
- Sales
- Ordering
- Stock management
- Pricing
- Supply chain management

Doing It Manually Works, Right?

Sure, but is it efficient? Here's an example:

Product recalls are a fact of life in the retail industry. Unlike many retailers, Costco has the data necessary to identify members who have more than likely purchased recalled product. Originally, Costco's recall program required the use of automated tapes and at its worst would take **weeks** to devise member address lists for recalled products. Today, with the use of an ETL[3] program and Query, this process can be done in under two

[3] ETL is a data-transfer program. ETL is an acronym that stands for Extract, Transform, and Load.

hours. How much liability can this absolve Costco of by being able to notify members quickly of potential problem items? Just because you're a whiz at the abacus and it gets the job done doesn't mean it's the best tool to keep you competitive in today's marketplace!

As you learn more about the data you are using and why, Query will become a powerful tool for you to measure and diagnose warehouse, departmental, vendor, and especially item-level data, in addition to all aspects of business performance.

Costco Merchandising and Business Intelligence

How can we ensure we are meeting the Six Rights of Merchandising to the absolute best of our ability quickly, accurately and efficiently? Do we indeed have the right merchandise? What qualifies the merchandise as being the right merchandise? Do we have that merchandise in the right place? If we moved the flowers closer to the wine, would we sell more wine? Would we sell more flowers? Would we sell less of the items in between? Do we have the right quantity? Are we selling out quickly? Do we have overstock? Ideally, should we sell out quickly at the point of saturation? What is that point? Can we always be improving on our business and our decisions? Of course! Let's take a closer look at the Six Rights of Merchandising, keeping in mind that this method can be applied to any business process we have.

1. The Right Merchandise
2. In the Right Place
3. At the Right Time
4. In the Right Quantity
5. In the Right Condition
6. At the Right Price

The Right Merchandise

- We sell only quality products.
- We sell a limited number of items, and we make sure they're the best in their field.
- The quality of our private-label items is equal to, or exceeds, that of national brands.

Claim 1

- *We sell only quality products.*
  Here we have the word quality, which is subjective and left to an individual's own understanding of the product. Being unquantifiable, we are left with certain measurements such as claims a product has. For example, one product might claim "all-natural butter". Is this the case?

Claim 2

- *We sell a limited number of items, and we make sure they're the best in their field.*
  Again' 'best in their field' is hard to quantify. However, limited number of items is not, because we can easily manage our SKU count. Can you visualize a report that informs us of our SKU counts in warehouses, regions, and the company? If we have a limited number of SKUs of which some items are slow movers, other items are selling out, and yet others are tremendously overstocked, how are we managing these items? Is there a better way to manage our SKUs than what

we're doing right now? If so, how would that impact our business and give us an advantage over our competitors?

Claim 3

- *The quality of our private-label items is equal to, or exceeds. that of national brands.*
  Here again is that word quality. Take bed sheets, for example: Do we capture the data that describes the thread count of national brands compared to our brand? How often is this data updated? Are we making good buying decisions based on up-to-date information?

In the right place

- As good merchants, our merchandise is carefully placed for optimum sales performance, with end caps and displays designed to let the quality of the product speak for itself.
- Costco pays attention to the basics and we like to keep things simple.

Claim 1

- *As good merchants, our merchandise is carefully placed for optimum sales performance, with end caps and displays designed to let the quality of the product speak for itself.*
  In this claim, we speak of product placement and sales performance. Currently, we identify when an item is on an End Cap vs. when it is on The Fence. It wouldn't be a stretch to measure sales at different times of the year—comparing End Caps vs. Bays vs. Fence placements. Let's say Proctor & Gamble (P&G) pays more to have Colgate Toothpaste on an End Cap, so we make the deal with P&G to put Colgate on an End Cap. Was that the best business decision? What if Helium tanks would have sold better that week on an End Cap—even without the Colgate deal? Which is the better service to our members and to our business? Can we specifically identify what products would be best on End Caps throughout the year? Of course, certain items should be displayed during different seasons or special events. What about trends and products that are not so obvious? Measuring and comparing product sales will bring to light whether or not we have optimal sales performance.

  Warehouse managers have different philosophies on placing televisions. Some put the expensive ones up front and create excitement about quality products. Others put the more affordable ones up front and encourage sales. Who is right? Could they both be right? How would times of the year and demographics factor in? Think about smaller items we carry and having our products in the right place. Can we be doing this better? Should we capture more information about product placement in the locations? Should we capture data about whether an item like table wine is on the floor or in a fine-wine box?

  Consider that Walmart continues to add data and data elements and have continued to grow their business intelligence since 1989. The benefits we achieve through basic reporting may be large, but continual development will lead to greater understanding of our business, better decisions, and a greater competitive edge.

Claim 2

- *Costco pays attention to the basics and we like to keep things simple.*
  What are 'the basics' and how do we measure it? How can you define 'simple' in your business area?

At the right time

- For seasonal items, we want to be in early and out early.
- We are constantly bringing in new, exciting merchandise that our members know might not be there next time. This creates urgency to buy and also gives our warehouses a treasure-hunt atmosphere.

Claim 1

- For seasonal items, we want to be in early and out early.
  This is a time measurement. How do we currently make this happen? By looking at trends, can we do a better job of ensuring that we don't have seasonal items left over at seasons end, but also had enough merchandise to not frustrate our members and lower sales?

Claim 2

- We are constantly bringing in new, exciting merchandise that our members know might not be there next time. This creates urgency to buy and also gives our warehouses a treasure-hunt atmosphere.

  What is our rate of new items? Do we introduce new products every week, every two weeks, or every month? How does this relate to the frequency of shops that a member has and the quantity of purchase? Let's take a report that looks at the top 200 members (sales-wise) at the Tumwater warehouse. Of those members, Bill is on the list, and Bill is an ideal candidate to represent all 200 members. Measuring Bill's shopping history, we can see the days Bill shops and what Bill buys. Now let's look at our treasure-hunt items. Is Bill buying any? Based on our claim of Bill not knowing whether the product will be there again, does this create urgency to increase the frequency of shops and the amount of Bill's treasure-hunt buys? With this type of information, can we make better decisions regarding when to carry a product, where to place it, for how long, and at what time? Should certain departments or certain warehouses have more treasure-hunt items or less?

Consider the last three rights of Merchandising: What are measurable elements of these statements that can be translated into actionable reports? The goal is to not just work harder at our business, but to work smarter.

In the right quantity

- We want to always have enough of our basic merchandise to serve our members well at all times.
- We don't keep a lot of surplus inventory in our buildings; it isn't cost-effective.
- We have rapid turnover of inventory, with goods constantly arriving and selling just as fast in heavy-laden members' carts.

Measurable information

In the right condition

- We sell only high-quality goods, and we want these to be in the best condition—that means nothing faulty, damaged, or broken.
- Our fresh products really are fresh, and are backed by the highest food-safety standards in the industry.

Measurable information

At the right price

- We figure out the least expensive and most efficient way to get an item from the manufacturer to the member.
- We sell items for as little as we can, instead of for as much as we can
- We always pass on any savings reduction in cost to the member.
- Our private label items always represent at least a 20% savings over the national brand, and usually more – some of them can mean a savings of more than 50% .
- If we can't be competitive in our price with an item, we won't carry it.

Measurable information

Data retrieval, Analysis and Presentation

We use Query to answer questions that help drive the business. Query accesses historical data to help us study our past–in order to see our future. Most often, a person developing queries is known as a *decision-support specialist* or a business liaison, someone who can create queries to support the business, or transform business logic into automated queries. These roles are part of a bigger world known as data warehousing.

Simply put, data warehousing is stored data designed for reporting and analysis. Although databases typically store all of the information about a company, simple questions can be very hard to answer accurately using the data from the system. Most databases store granular-level information, which may work well for the accountant who is researching individual purchase orders, but for broader company-level questions it becomes more challenging to aggregate the data. What further complicates matters is when a broader-level question crosses over departments or divisions. Factor in internal disagreements about which data is correct. Buyers may have one view of cost or sales, while Accounting may have another.

The purpose of a decision-support specialist or business liaison is to handle the challenges that lead to answering questions accurately. The best way to do this is to qualify definitions. If a request for a report includes fields such as 'Warehouse' and 'Ship Date' information, it is the responsibility of the decision-support specialist to not just assume the warehouse or ship date, but rather to qualify the definition. That means finding out that indeed 'Warehouse' actually equals all locations including depots only in the US, and 'Ship Date' is the ship date as defined by RTV.

In the role of decision-support specialist, you will develop your own interview techniques to qualify facts for your reports. There is more information on the web or through The Data Warehouse Institute on data modeling and fact-qualifying analysis.

For now, here is a simple checklist that will get you started on some projects when you leave the class:

TAKE AN INVENTORY OF CURRENT REPORTS IN YOUR DEPARTMENT.
What decisions are being made from these reports?
Can any reports be deleted?
Can any reports be consolidated?

REVIEW REPORTING FOR ACCURACY.
What is the source?
Are you using the right tables?
Are you using the right fields?
Are you using the right formulas?
Are you using the correct conditions?
Are the business rules and math being applied consistent?

AUTOMATE COPY AND PASTE REPORTING.
Automate manual reports.

Where are you getting the paper reports from?
Automate into a query.

DECISION SUPPORT
Recognize decisions needed that have no support.
Ask questions to develop questions.

When doing this you will see Query is the first step in retrieving data, cleaning it to be accurate and beginning analysis. Having spreadsheets with a lot of numbers cannot compete with intelligent reports and presentations that call for an immediate action.

MISCELLANEOUS

Here are some items (presented in note format) that were out of scope of the class, but are related. If you have specific questions about any of these items, please contact the Query trainer.

HAVING AND EXISTS

HAVING

The *Having* clause is available because the *Where* clause cannot be used with aggregate functions. The Showcase Query wizard will automatically add the *Having* clause for you.

Having Syntax

```
SELECT
        MBWHSE,
        MBITEM,
        MBDSEL,
        SUM( MBDSUN ) AS "SumMbdsun"
FROM
        EISPRDDTA2.INID1005P INID1005P
WHERE
        MBCMPY = 1
        AND MBWHSE = 110
        AND MBDEPT = 17
        AND MBITEM = 2

GROUP BY
        MBWHSE,
        MBITEM,
        MBDSEL
HAVING
        SUM( MBDSUN ) < 10
```

What order would *Having* need to appear if we were working with *Unions*?
The same rules apply, only the *Order By* has to be at the bottom of the *Union Select* statement.

EXISTS

The *Exists* condition is considered to be met if the subquery returns at least one row.

The syntax for the *Exists* condition is:
SELECT columns
FROM tables
WHERE EXISTS (sub query):

The *Exists* condition can be used in any valid SQL statement – *Select*, *Insert*, *Update* or *Delete*.

Example #1
Let's take a look at a simple example. The following is an SQL statement that uses the *Exists* condition:

```
SELECT *
FROM suppliers
WHERE EXISTS
        (Select * from orders where suppliers, supplier_id = orders.supplier_id)
```

This select statement will return all records from the suppliers table where there is at least one record in the orders table with the same supplier_id.

Example #2
NOT EXISTS
The *Exists* condition can also be combined with the *Not* operator.
For example:

```
SELECT *
FROM suppliers
WHERE not exits (select * from orders WHERE suppliers.supplier_id =
orders.supplier_id)
```

This will return all records from the suppliers table, where there are no records in the orders table for the given supplier_id.

```
From "table_name1"
WHERE EXISTS
(SELECT *
FROM "table_name2" WHERE [where Condition])
```

Please note that instead of *, you can select one or more columns in the inner query. The effect will be identical.

Let's use the same example tables:

Table Store_Information

Store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
San Diego	$250	Jan-07-1999
Los Angeles	$300	Jan-08-1999
Boston	$700	Jan-08-1999

Table Geography

Region_name	Store_name
East	Boston
East	New York
West	Los Angeles
West	San Diego

And we issue the following SQL query:

```
SELECT SUM(Sales) FROM Store_Information
WHERE EXISTS
(SELECT * FROM Geography
WHERE region_name='West')
```

We'll get the following results:

SUM(Sales)
2750

At first this may appear confusing, because the subquery includes the [region_name = 'West'] condition, yet the query summed up stores for all regions. Upon closer inspection, we find that since the subquery returns more than zero, the *Exists* condition is true, and the condition placed inside the inner query does not influence how the outer query is run.

SUBQUERIES

Quantified Subqueries**:**

A quantified subquery allows several types of tests and can use the full set of comparison operators. It has the following general format:

> value-1 {=|>|<|>=|<=|<>} {ANY|ALL|SOME} (query-1). The comparison operator specifies how to compare value-1 to the single query column value from each subquery result row. The *Any*, *All*, and *Some* specifies give the type of match expected. *Any* and *Some* must match at least one row in the subquery. *All* must match all rows in the subquery, or the subquery must be empty (produce no rows).

For example, list all parts that have suppliers:

```
SELECT *
FROM p
WHERE pno =ANY (SELECT pno FROM sp)
```

pno	descr	color
P1	Widget	Blue
P2	Widget	Red

ANY ALL HAVING EXISTS

The *Any* clause works just like the *In* but allows >, >=, <, <=, =, <>

Any:
This operator is used in conjunction with a comparison operator to compare the value of an expression to the set of values produced by a subquery. The condition evaluates to TRUE if any value in the subquery result satisfies the comparison. The statement is false only if all comparisons are false.

Syntax:
Expression_1 comparison ANY (subquery)
The value of expression_1 is compared to the results from subquery.
Comparison specifies the predicate comparison operator (<, <=, =, <>, >=, >) that will be used to compare expression_1 to the result of subquery.
Subquery must specify a single column.

Example:
Expression Returns When
A < ANY (select Q from R) TRUE A is less than any one of the values of Q returned by the subquery.
The EXISTS condition is considered "to be met" if the subquery returns at least one row.
The syntax for the EXISTS condition is:

```
SELECT columns
FROM tables
WHERE EXISTS ( subquery );
```

The *Exists* condition can be used in any valid SQL statement - *Select*, *Insert*, *Update*, or *Delete*.

Example #1: Exists

Let's take a look at a simple example. The following is an SQL statement that uses the *Exists* condition:

```
SELECT *
FROM suppliers
WHERE EXISTS
(select *from orders
where suppliers.supplier_id = orders.supplier_id);
```

This select statement will return all records from the suppliers table where there is at least one record in the orders table with the same supplier_id.

Example #2 - NOT EXISTS

The *Exists* condition can also be combined with the *Not* operator:

```
SELECT *
FROM suppliers
WHERE not exists
(select * from orders Where suppliers.supplier_id = orders.supplier_id);
```

This will return all records from the suppliers table where there are no records in the orders table for the given supplier_id. All This operator is used in conjunction with a comparison operator to compare the value of an expression to the set of values produced by a subquery. The condition evaluates to TRUE only when the comparison is satisfied for all values in the subquery result. If any comparison is false, then the result is false.

Expression_1 comparison ALL (subquery)
The value of expression_1 is compared to the results from subquery.
Comparison specifies the predicate comparison operator (<, <=, =, <>, >=, >) that will be used to compare expression_1 to the result of subquery.
Subquery must specify a single column.

Example

Expression	Returns	When
A < ALL (select Q from R)	TRUE	A is less than all values of Q returned by the subquery.

Variables

Creating variables:

1. On the Query menu, click **Variables**.
2. Click **Add**.
3. Type a name for the variable in the *Variable Name* box.
4. In the *Variable Value Type* box, select the type of value you want to assign to the variable.
5. In the *Variable Data Type* box, select the data type to assign to the variable.
6. To use a value of NULL, if no value is set for the variable, select **Null Capable**. (For more information about *Null Capable*, see the *Add Variable* and *Variable Properties* dialog boxes.).
7. To assign a restricting range for the values, type the values in the *Minimum Variable Value* and *Maximum Variable Value* boxes.
8. To prompt the user to enter a value for the variable when the query is run, select **Prompt For Value At Run Time**.
9. Click **OK**.
10. By default, variables are created as local variables. To create a global variable, select **Share Variable Across Queries**.
11. After creating a variable, you must set a value for it. For instructions, see *Setting the Variable Value*.

12. After creating a variable, you must incorporate it into the query. For instructions, see *Incorporating Variables Into a Query*.

13. If you selected *Prompt for Variable at Run Time*, you have the option of specifying additional prompt criteria. For instructions, see *Defining prompts for variables*.

14. To specify that variable values should be changed to uppercase before rows are selected, select **Uppercase Values**.

15. When you click **OK**, the *Variables dialog box* appears and your new variable is in the *Variables* list. If the variable is local, a single ampersand (*&*) appears before the variable name. If the variable is global, two ampersands (*&&*) appear before the variable name. If *<Empty string value>* appears, it indicates you have not yet set a variable value.

16. If you change a variable that is used in multiple queries, you are prompted to change the variable in all queries that use that variable.

Comparisons will require study and research.

Add Variable and Variable Properties Dialog Boxes

These dialog boxes specify or change the following information for a variable:

- The variable name
- The type of variable
- The data type of the variable value
- The minimum and maximum variable values

You can also specify whether the user should be prompted for a variable value.
The only difference between the *Add Variable* and *Variable Properties* dialog boxes is that the *Add Variable* dialog box specifies a new variable and the *Variable Properties* dialog box changes an existing variable.

The following paragraphs describe how to use each part of either of these dialog boxes:

Variable Name
This box specifies the name of the variable. Enter or edit the variable name.
A variable name can include underscores (_) and any alphanumeric characters, except spaces and ampersands (&). The maximum length of a variable name is 64 characters.

Share Variable Across Queries
Select this check box to share the variable you are adding with all Query or Report Writer files. If this check box is selected, two ampersands (&&) are added to the variable name in the *Variables* box of the *Variables Dialog* box when you choose *OK*.

Variable Value Type
This box specifies the type of variable. Select one of the following variable types:

Single Value
Select this option to specify that a single value must be entered as the variable value.

Range of Values (BETWEEN)
Select this option to specify that any two values may be entered as the variable value.

List of Values (IN)
Select this option to specify that a list of values may be entered as the variable value.

Variable Data Type
This box specifies the data type for the variable. Any value set to the variable is treated as the data type set in this box. Select one of the following data types:

Variable Length Character String
Select this option to specify that the variable is a character string with no pre-determined length. When you select this option, the *Maximum String Length* box appears. Choose this when the variable entered can vary in length, such as in the case of salespeople's names.

Fixed Length Character String
Select this option to specify that the variable is a character string with a set length. When you select this option, the *Maximum String Length* box appears.
Choose this when all of the possible variables entered are the same length, such as in the case of two-letter postal abbreviations for state names.

Date
Select this option to specify that the variable is a date. You can change the input format of the date value in your Windows Control Panel. You can also specify this value in the ODBC format, YYYY-MM-DD, where YYYY is the year, MM is the month, and DD is the day.

Time
Select this option to specify that the variable is a time. You can change the input format of the time value in your Windows Control Panel. You can also specify this value in the ODBC format, HH-MM-SS, where HH is the hour (01-24), MM is the minute (01-60), and SS is the second (01-60).

Time stamp
Select this option to specify that the variable is a time stamp. This value contains the date and time as specified in your Windows Control Panel. You can also specify this value in the ODBC format, YYYY-MM-DD HH-MM-SS, where YYYY is the year, MM is the month, DD is the day, HH is the hour, MM is the minute, and SS is the second.

Tiny integer (-127 to +127)
Select this option to specify that the variable is an integer value from -127 to +127.

Small integer (-32767 to +32767)
Select this option to specify that the variable is an integer value from -32,767 to +32,767.

Integer (-2147483647 to +2147483647)

Select this option to specify that the variable is an integer value from -2,147,483,647 to +2,147,483,647.

Big integer

Select this option to specify that the variable is any integer value from -9,223,372,036,854,775,807 to +9,223,372,036,854,775,807.

Real

Select this option to specify that the variable value is a real number. A real number can be any positive or negative number, including decimal and exponential values. A real number can contain up to seven significant digits.

Float

Select this option to specify that the variable value is a floating-point number. A floating point number is the same as a real number, except that it can contain up to 15 significant digits. Your data source may use the double data type instead of float.

Double

Select this option to specify that the variable value is a double-precision number. A double-precision number is the same as a real number, except that it can contain up to 15 significant digits. Your data source may use the float data type instead of double.

Bit (0 or 1)

Select this option to specify that the variable value is either 0 or 1.

Decimal

Select this option to specify that the variable value is any decimal value. When selected, the *Precision* and *Scale* boxes appear.

Numeric

Select this option to specify that the variable value is any numeric value. When selected, the *Precision* and *Scale* boxes appear.

Maximum String Length

This box specifies the maximum length allowed for the variable value. Enter a length in characters. This box is only available when you select either *Variable Length character string* or *Fixed Length character string* from the *Variable Data Type* box.

Precision

This box specifies the precision of the variable value. Enter a precision value. This box is only available when you select either *Decimal* or *Numeric* from the *Variable Data Type* box.

Scale

This box specifies the scale of the variable value. Enter a scale value. This box is only available when you select either *Decimal* or *Numeric* from the *Variable Data Type* box.

Uppercase Value

Select this check box to indicate that any character variable value should be changed to uppercase before using it to select rows.

Null Capable

Select this check box if the user is allowed to run the query without specifying a prompt value for this variable. To avoid runtime errors, do not set the prompt as *null capable* if the variable is used in expressions such as a *Case* statement.

Minimum Variable Value

This box specifies the minimum value that can be set for the variable. Enter the minimum value. Any value entered at run time that is lower than this value generates an error message.

Maximum Variable Value

This box specifies the maximum value that can be set for the variable. Enter the maximum value. Any value entered at run time that is higher than this value generates an error message.

Prompt For Value At Run Time

Select this check box to prompt the user to enter a variable value at run time.

Prompt Attributes

Choose this button to set properties associated with a variable prompt. This button is only enabled when the *Prompt for value at run time* check box is selected. When you choose this button, the *Prompt Attributes* dialog box appears.

References

Bounds, J. 2004, Kmart sues i2 over software, Dallas Business Journal, viewed 15 March 2008, < http://www.bizjournals.com/dallas/stories/2004/10/18/story1.html>.

Corporate History 2007, Kmart, viewed 15 March 2008, http://www.kmartcorp.com/corp/story/general/corporate_history.stm.

Michael Porter Five Forces Model 2008, viewed 15 March 2008, http://www.brs-inc.com/porter.asp.

Morganosky, M., 1997, 'Format Change in US Grocery Retailing', International Journal of Retail & Distribution Management, vol. 25, no. 6, pp. 211-218.

The Associated Press, 2004, A Look at Kmart's History, Los Angeles Times, viewed 15 March 2008, http://www.latimes.com/business/investing/wire/sns-ap-kmart-timeline,1,1736638.story.

Turner, M. L., 2003, *Kmart's Ten Deadly Sins: How Incompetence Tainted an American Icon*, Wiley, Hoboken, NJ.

Zyman, S., & Brott., A 2002, *The End of Advertising as We Know It*, John Wiley and Sons.

ANSWERING QUESTIONS THAT HELP DRIVE THE BUSINESS

The right merchandise, the right price, the right time, the right quantity, the right condition and the right place are less of a question and more of an answer with Query. SQL/*Query* gives you a foundation that you can build on to create effective reports.

Transform copy and paste processes to automated queries. Create reports that do not currently exist by navigating through the EIS400 database.

Develop Queries that help answer the questions that keep you and your managers awake at night.

- What is Query
- Showcase products
- Real time vs. Historical data
- EIS400 data accuracy
- Libraries, Tables, Members
- Relational Databases
- Physical vs. Logical files
- Gathering requirements
- Researching tables
- A look into metadata
- SQL and UNIONS
- String Functions
- Time and Date Functions
- Value Expressions
- Troubleshooting guide
- Over 80 Exercises

.

Accounting Training

Part of the Query Training series

- **Query Fundamentals**
- **Query Intermediate**
- **Query Advanced**
- **Query Expert**

Visit us on the intranet at
http://accounting/Train/Query/QueryMainPage.htm