

Damien Berahzer

Data Analytics using
MS Access





Damien Berahzer



ERNST & YOUNG

CISA
CERTIFIED INFORMATION SYSTEMS AUDITOR



City of Atlanta
2010



ISACA Atlanta – Geek Week

Before we begin

- Questions
- CDs
- Time Checkers
- Response Cards



Choose one that describes you

1. Super Geeky Nerdy
2. Geek
3. IT Advanced
4. IT Intermediate
5. IT Beginner

0%

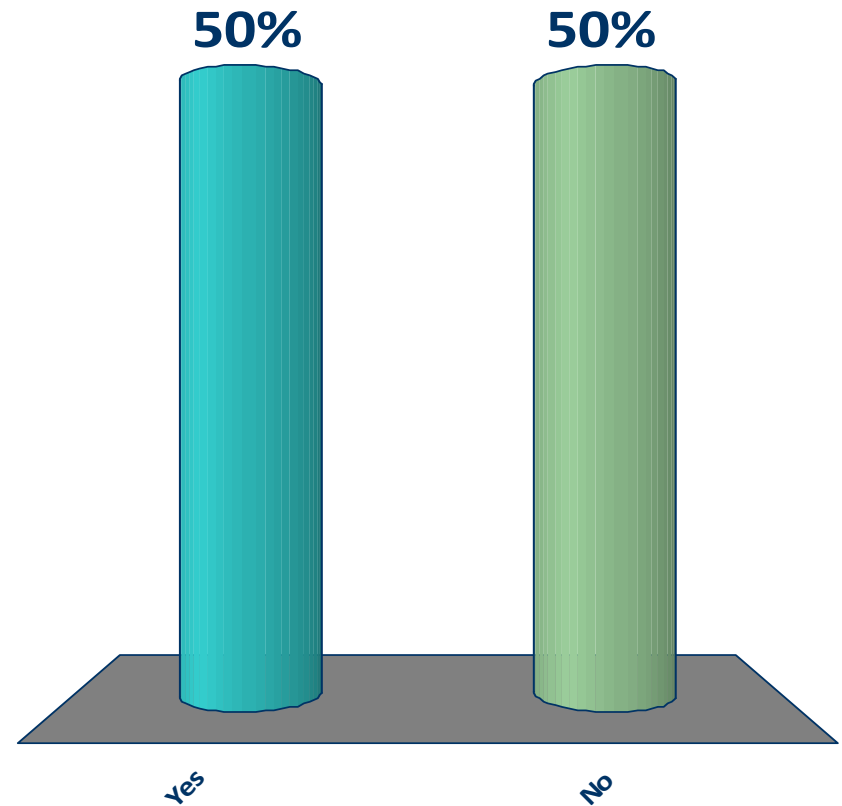


MS Access – Audit Usefulness

- SOD development
- Payroll Analysis
- Full Population Termination Testing
- JE-CAATS (Oracle, PeopleSoft, White Plains, Lawson)
- AIX Security Testing
- 911 Calls for service Analysis

Do you have MS Access at your job?

1. Yes
2. No

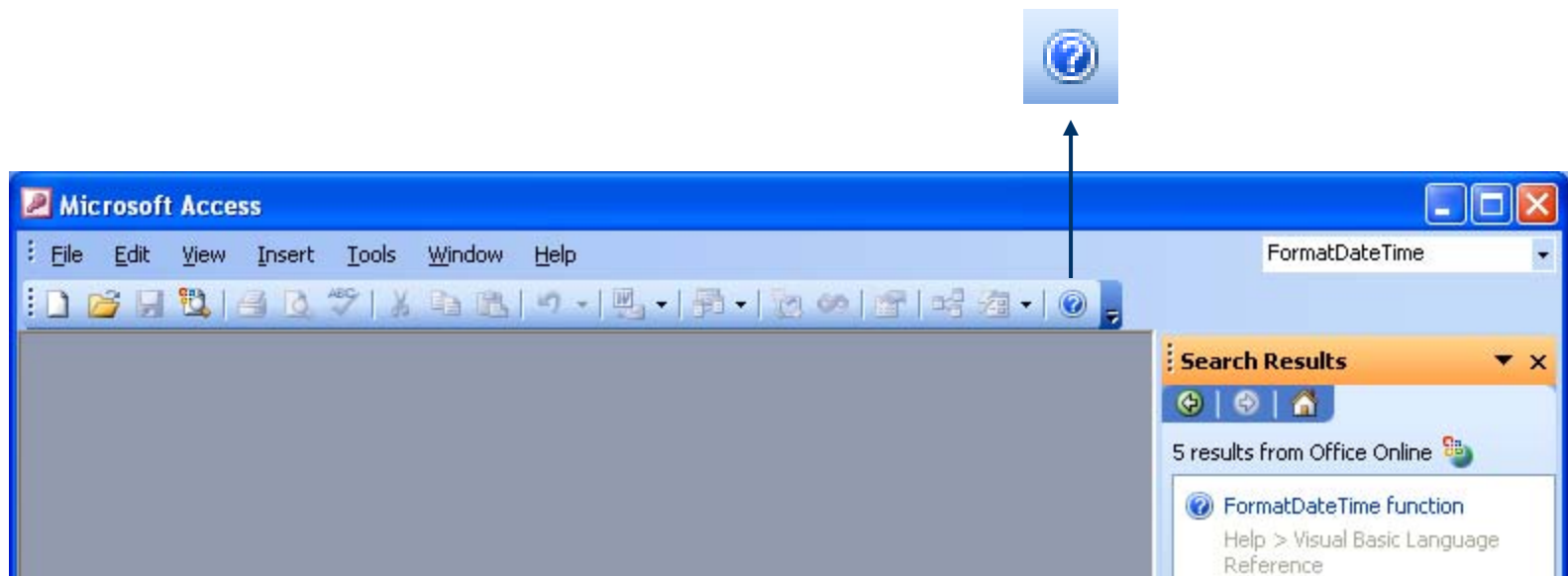


Coverage

- Help Function
- Getting what you need
- Built in Access Queries
- Query Logic Overview
- Data Import Considerations
- MS Access functions
- City Auditor Functions

Help Function

- USE IT



Getting what you need

- What to ask for
 - Data? Be specific in your requests to the extent possible.
- Some verbiage to consider
 - Pipe or Bar delimited
 - Oracle Database Format Using SQL Plus: Spool the results into a text file

```
SQL> SELECT MENU_ID, ENTRY_SEQUENCE, SUB_MENU_ID, FUNCTION_ID FROM APPS.FND_MENU_ENTRIES;
```

| MENU_ID | ENTRY_SEQUENCE | SUB_MENU_ID | FUNCTION_ID |
|---------|----------------|-------------|-------------|
| 67614 | 3 | 68196 | 1066 |
| 67616 | 7 | 68688 | |
| 67616 | 21 | 68682 | |
| 67616 | 8 | 68632 | |
| 67616 | 9 | 67614 | |
| 67616 | 10 | 67617 | |
| 67616 | 11 | | 1025 |
| 67616 | 12 | | 1154 |
| 67616 | 13 | 67705 | |
| 67616 | 14 | | 1844 |
| 67616 | 15 | | 1797 |

| MENU_ID | ENTRY_SEQUENCE | SUB_MENU_ID | FUNCTION_ID |
|---------|----------------|-------------|-------------|
| 67616 | 16 | | 1801 |
| 67616 | 17 | | 1820 |

Getting what you need continued

Other / Unsure: Delimited File versus Fixed width versus seeded report –

- Delimited – Delimiter and text qualifiers

Example:

```
20071213;100;100;Active Biweekly;Regular Pay;8.00;Decosta, Benjamin;1;20071226;  
20071214;100;100;Active Biweekly;Regular Pay;8.00;Decosta, Benjamin;1;20071226;  
20071217;100;100;Active Biweekly;Regular Pay;8.00;Decosta, Benjamin;1;20071226;  
20071218;100;100;Active Biweekly;Regular Pay;8.00;Decosta, Benjamin;1;20071226;  
20071219;100;100;Active Biweekly;Regular Pay;8.00;Decosta, Benjamin;1;20071226;
```

Getting what you need continued

- Seeded Report

| Employee Transactions & Totals | | | | | |
|--------------------------------|-------------------------|---------------|--------------|-------------------|------------|
| Time Period: | 12/13/2007 - 12/26/2007 | | | | |
| Query: | COA EMPLOYEES - C | | | | |
| Pay Codes: | (138): | | | | |
| Actual/Adjusted: | Actual Only | | | | |
| Employee: | Sombody, Uknow | ID: | 6455 | | |
| Transactions: | Day Date | Pay Code | Hours | Money | Entered By |
| | Th 12/13/2007 | UXT | 2.68 | | 3767 |
| | Th 12/13/2007 | UXT | 2.68 | | 3767 |
| | Th 12/13/2007 | UXT | 1.68 | | 3767 |
| | Th 12/13/2007 | UXT | 1.68 | | 3767 |
| | Tu 12/25/2007 | Christmas Day | 8.00 | | |
| Totals: | Pay Code | Money | Hours | Wages | |
| | Holiday Pay | \$0.00 | 8.00 | \$201.47 | |
| | OT Pay | \$0.00 | 0.00 | \$0.00 | |
| | OT Premium Pay | \$0.00 | 0.00 | (\$0.00) | |
| | Regular Pay | \$0.00 | 72.00 | \$1,813.24 | |
| | UXT | \$0.00 | 8.73 | \$219.94 | |
| | Totals: | \$0.00 | 88.73 | \$2,234.66 | |

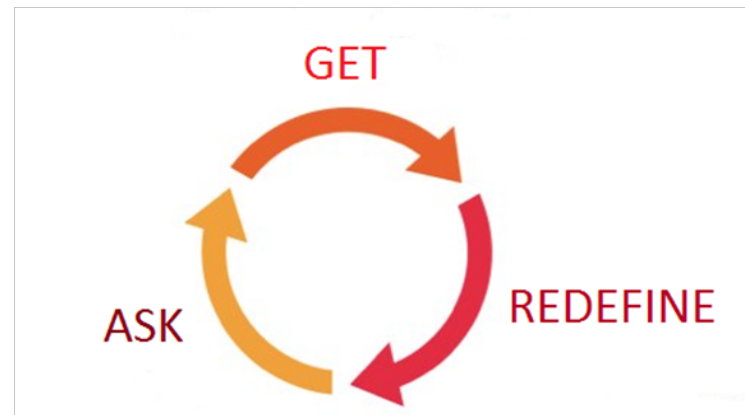
Getting what you need continued

- Record count and totals (if included in request)
 - Completeness and Accuracy Test

```
66077 rows selected.
```

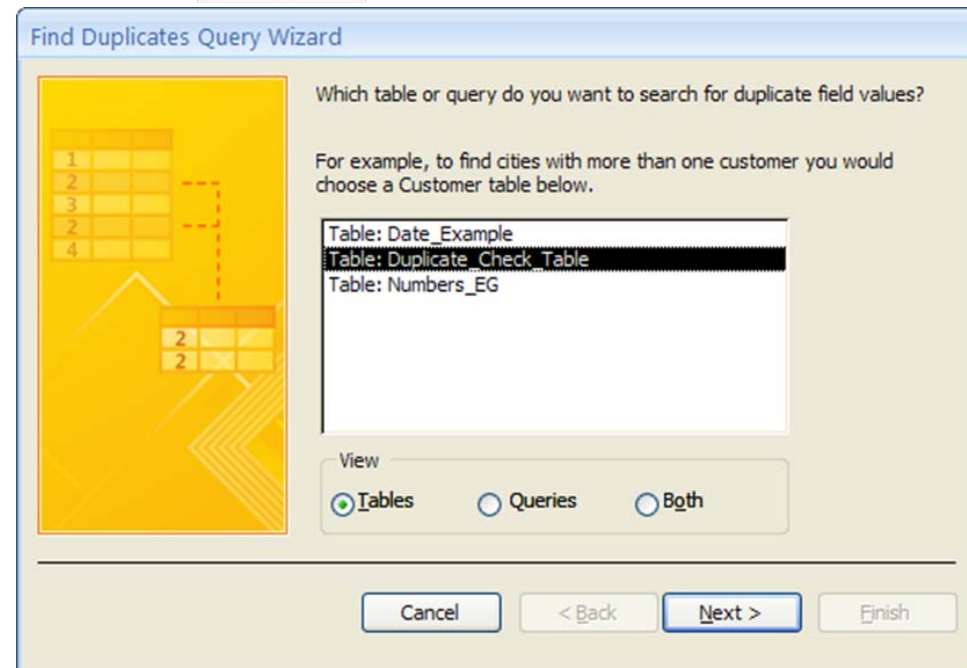
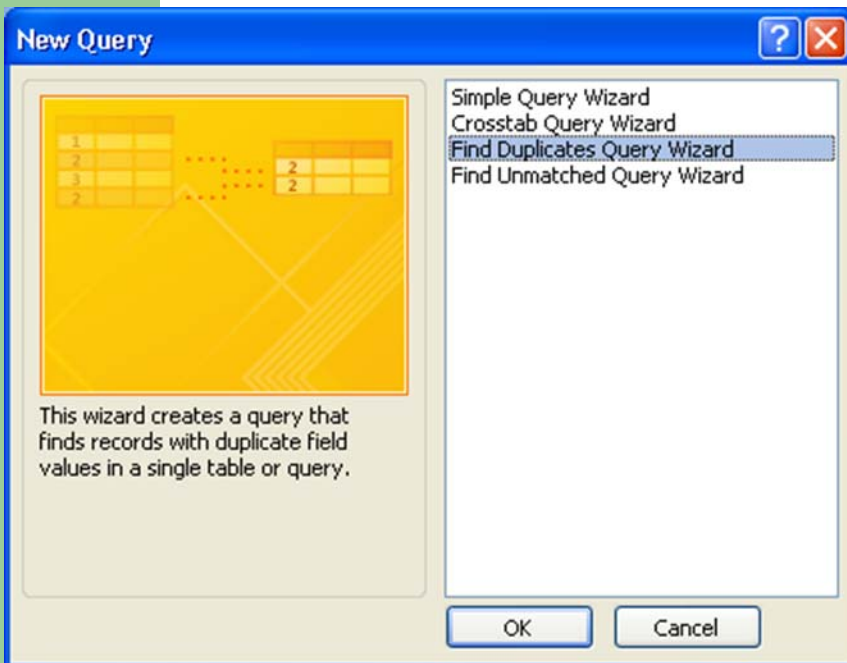
```
SQL> SPOOL OFF
```

- Understand the process may be iterative
 - Techy People don't speak the same language



Built In Access Query and Overview

- Duplicate Record Check



Built In Access Query and Overview

- Duplicate Record Check

Find Duplicates Query Wizard

Which fields might contain duplicate information?

For example, if you are looking for cities with more than one customer, you would choose City and Region fields here.

Available fields:

| | | |
|---------------|----|---------|
| Item_Name | > | Item_ID |
| Item_Location | >> | |
| Priority | < | |
| | << | |

Duplicate-value fields:

| |
|---------|
| Item_ID |
|---------|

Cancel < Back Next > Finish

Find Duplicates Query Wizard

Do you want the query to show fields in addition to those with duplicate values?

For example, if you chose to look for duplicate City values, you could choose CustomerName and Address here.

Available fields:

| | | |
|--|----|---------------|
| | > | Item_Name |
| | >> | Item_Location |
| | < | Priority |
| | << | |

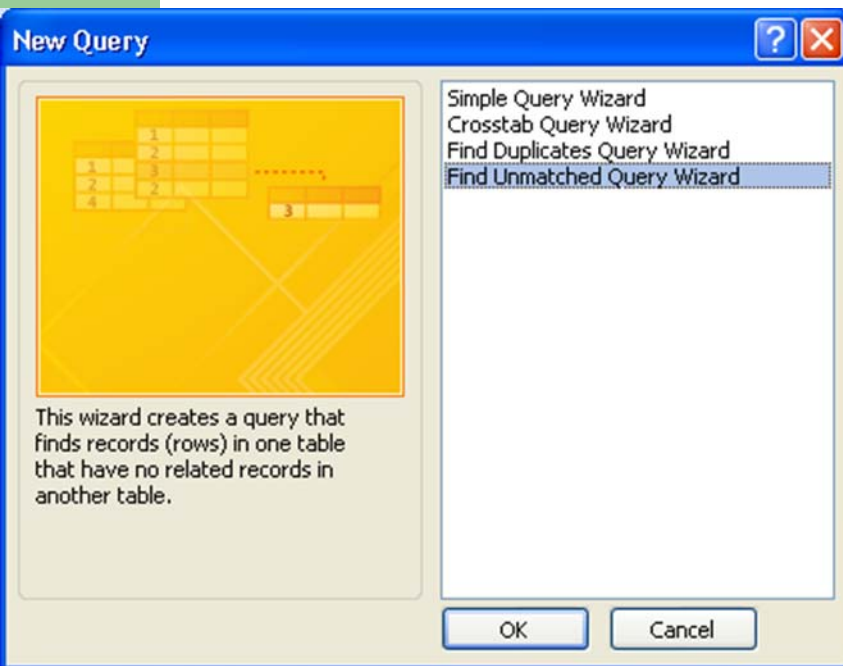
Additional query fields:

| |
|---------------|
| Item_Name |
| Item_Location |
| Priority |

Cancel < Back Next > Finish

Built In Access Query and Overview

- Unmatched Query



Built In Access Query and Overview

- Unmatched Query

Find Unmatched Query Wizard

Which table or query contains the related records?

For example, if you've already selected customers and you're looking for customers without orders, you would choose orders here.

Table: Approver
Table: Date_Example
Table: Numbers_EG

View
 Tables Queries Both

Cancel < Back Next > Finish

Find Unmatched Query Wizard

What piece of information is in both tables?

For example, a Customers and an Orders table may both have a CustomerID field. Matching fields may have different names. Select the matching field in each table and then click the <=> button.

Fields in 'Duplicate_Check_Table':
Item_ID
Item_Name
Item_Location
Priority

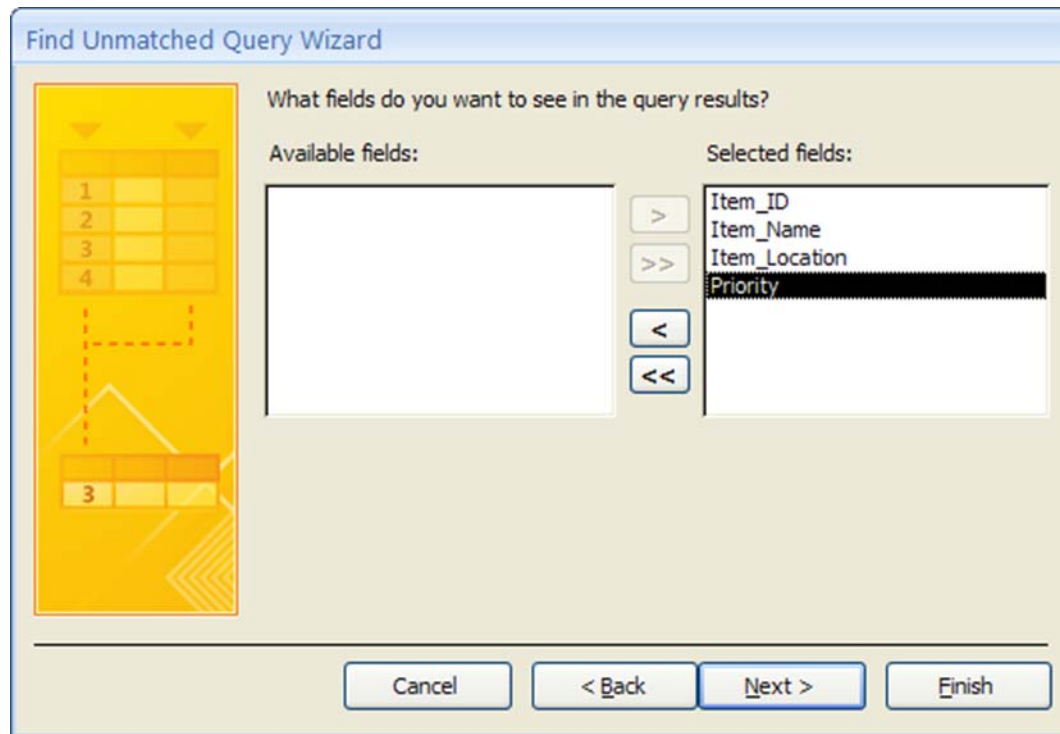
Fields in 'Approver':
ID
Item_ID
Approver
Date

Matching fields: Item_ID <=> Item_ID

Cancel < Back Next > Finish

Built In Access Query and Overview

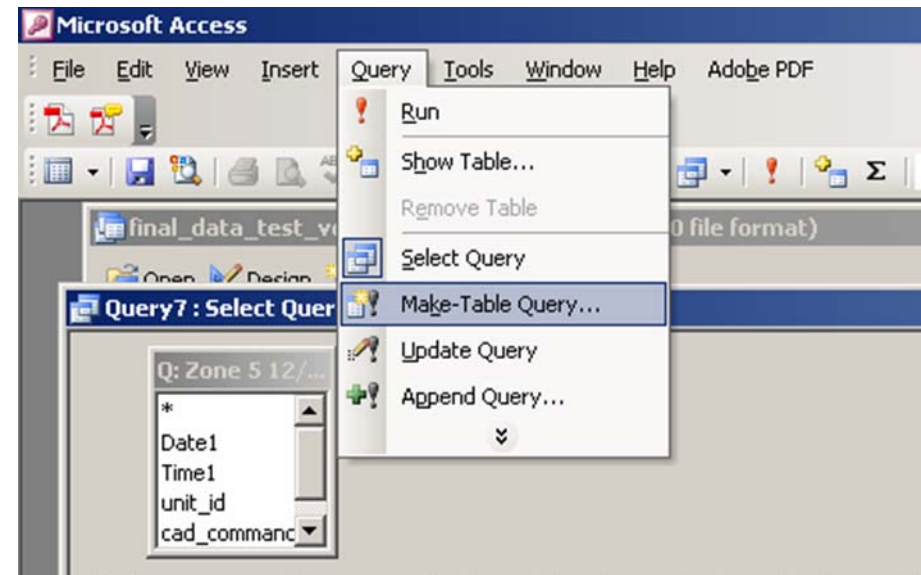
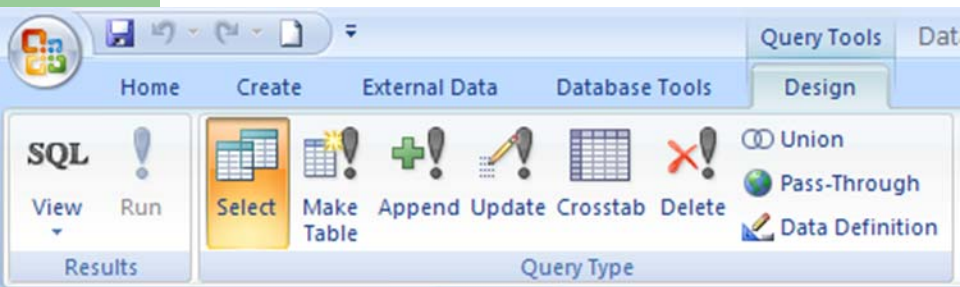
- Unmatched Query



Built In Access Query and Overview

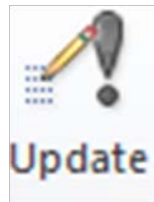
- Make Table Query

- A table of data derived from a query.
- Create a new table of existing or derived data.



Built In Access Query and Overview

- Update Query



| Number_EG | Convert_num |
|-----------|-------------|
| 0 | |
| 000067 | |
| 010 | |
| 012003 | |
| 123 | |
| 3 | |
| 721 | |
| 9 | |
| None | |
| None | |
| * | 0 |

Record: 1 of 10 No Filter

Update_Query_Example

Numbers_EG

- Number_EG1
- Convert_num

Field: Convert_num

Table: Numbers_EG

Update To: Number_eg1

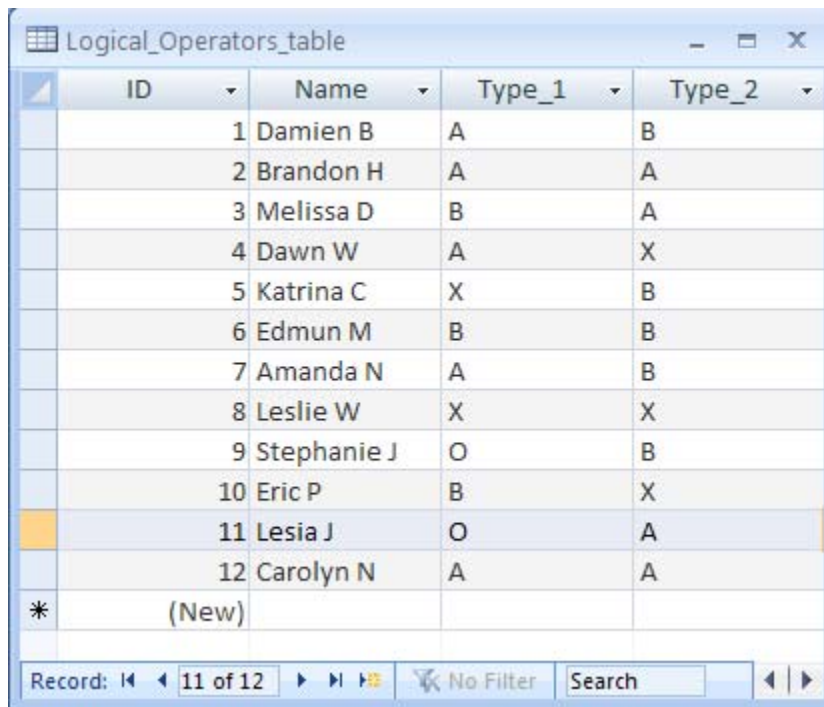
Criteria:

or:

Query Review: Logical Operators

- AND
 - Both Conditions must apply
- OR
 - Either condition can apply
- XOR
 - Either condition can apply but not both
- NOT
 - Logical Opposite of the condition

Query Review: Logical Operators



| ID | Name | Type_1 | Type_2 |
|----|-------------|--------|--------|
| 1 | Damien B | A | B |
| 2 | Brandon H | A | A |
| 3 | Melissa D | B | A |
| 4 | Dawn W | A | X |
| 5 | Katrina C | X | B |
| 6 | Edmun M | B | B |
| 7 | Amanda N | A | B |
| 8 | Leslie W | X | X |
| 9 | Stephanie J | O | B |
| 10 | Eric P | B | X |
| 11 | Lesia J | O | A |
| 12 | Carolyn N | A | A |
| * | (New) | | |

1. Where Type_1 = A **AND** Type_2 = B
2. Where Type_1 = A **OR** Type_2 = B
3. Where Type_1 = A **XOR** Type_2 = B
4. Where Type_1 = A **AND NOT** Type_2 = B

Query Review: Special Operators

- LIKE
 - Compares a string expression to a pattern in an SQL expression
 - You can use wild cards
- BETWEEN AND
 - Determines whether the value of an expression falls within a specified range of values (value 1 and value 2)
 - It is (inclusive) of both values
 - Be careful of date/time values
- IN
 - Determines whether the value of an expression is equal to any of several values in a specified list

Query Review: WHERE and HAVING Clauses

- SELECT *fieldlist*
FROM *table*
WHERE *selectcriteria*
GROUP BY *groupfieldlist*
[**HAVING** *groupcriteria*]
- **WHERE** specifies which records from the tables listed in the FROM clause are affected by a SELECT, UPDATE, or DELETE statement.
- After GROUP BY combines records, **HAVING** displays any records grouped by the GROUP BY clause that satisfy the conditions of the HAVING clause

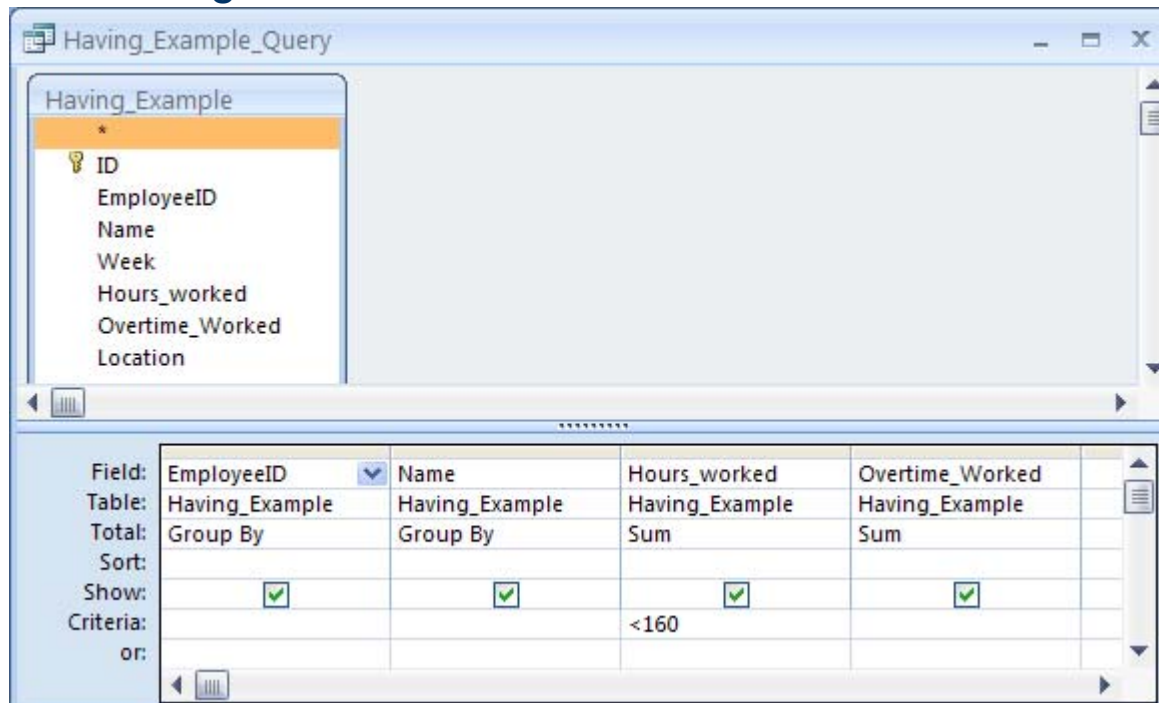
HAVING Clause example

| ID | EmployeeID | Name | Week | Hours_worked | Overtime_W | Location |
|---------|------------|------------|------|--------------|------------|----------|
| 1 | 12345 | Prad Bitt | 1 | 40 | 2 | atl |
| 2 | 12345 | Prad Bitt | 2 | 36 | 4 | atl |
| 3 | 12345 | Prad Bitt | 3 | 36 | 5 | atl |
| 4 | 12345 | Prad Bitt | 4 | 40 | 2 | atl |
| 5 | 23456 | Fegan Mox | 1 | 40 | 1 | NYC |
| 6 | 23456 | Fegan Mox | 2 | 40 | 6 | NYC |
| 7 | 23456 | Fegan Mox | 3 | 40 | 1 | NYC |
| 8 | 23456 | Fegan Mox | 4 | 40 | 3 | NYC |
| 9 | 12555 | Damon Matt | 1 | 40 | 2 | Chi |
| 10 | 12555 | Damon Matt | 2 | 36 | 4 | Chi |
| 11 | 12555 | Damon Matt | 3 | 36 | 5 | Chi |
| 12 | 12555 | Damon Matt | 4 | 40 | 2 | Chi |
| * (New) | | | | | | |

Record: 1 of 12 No Filter Search

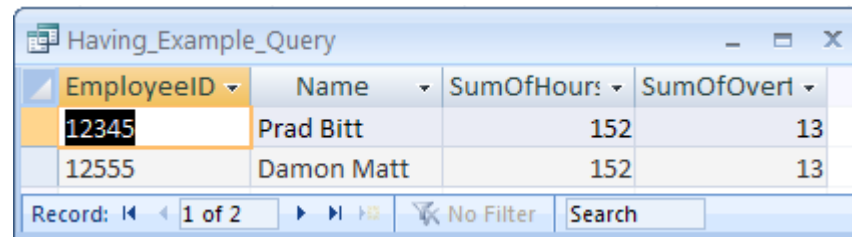
HAVING Clause example

We selected the sum of the hours worked and sum of overtime for employees having the sum of hours worked less than 160 hours



HAVING Clause example

Results



| EmployeeID | Name | SumOfHours | SumOfOverl |
|------------|------------|------------|------------|
| 12345 | Prad Bitt | 152 | 13 |
| 12555 | Damon Matt | 152 | 13 |

Record: 1 of 2 | No Filter | Search

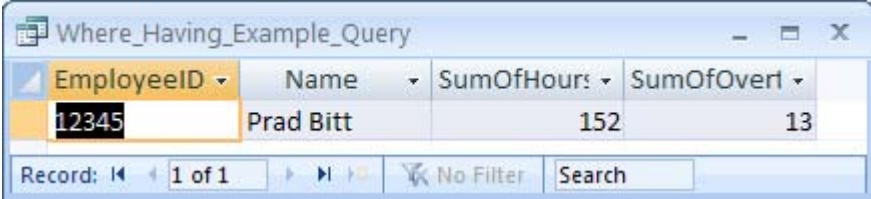
Where and Having Example

The screenshot displays the Microsoft Access interface for a query named "Where_Having_Example_Query". The query is based on the "Having_Example" table. The design grid shows the following configuration:

| Field: | EmployeeID | Name | Hours_worked | Overtime_Worked | Location |
|-----------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|
| Table: | Having_Example | Having_Example | Having_Example | Having_Example | Having_Example |
| Total: | Group By | Group By | Sum | Sum | Where |
| Sort: | | | | | |
| Show: | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| Criteria: | | | <160 | | 'Atl' |
| or: | | | | | |

Where and Having Example

Results



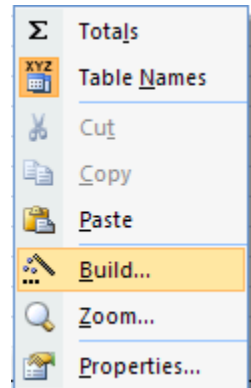
| EmployeeID | Name | SumOfHours | SumOfOver |
|------------|-----------|------------|-----------|
| 12345 | Prad Bitt | 152 | 13 |

Record: 1 of 1 | No Filter | Search

Data Import Considerations

- To clean or not to clean data
 - Manual stripping or Later Manipulation
- How to import data types
 - Text, double, dates etc

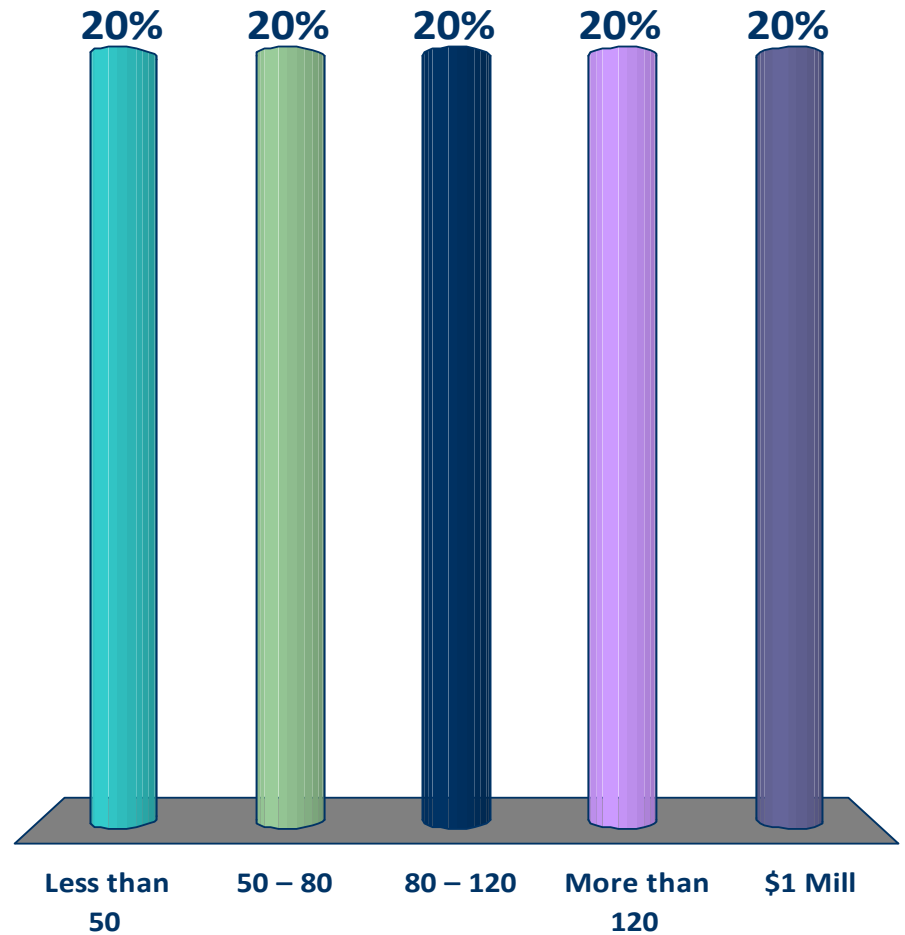
Introduction to Functions



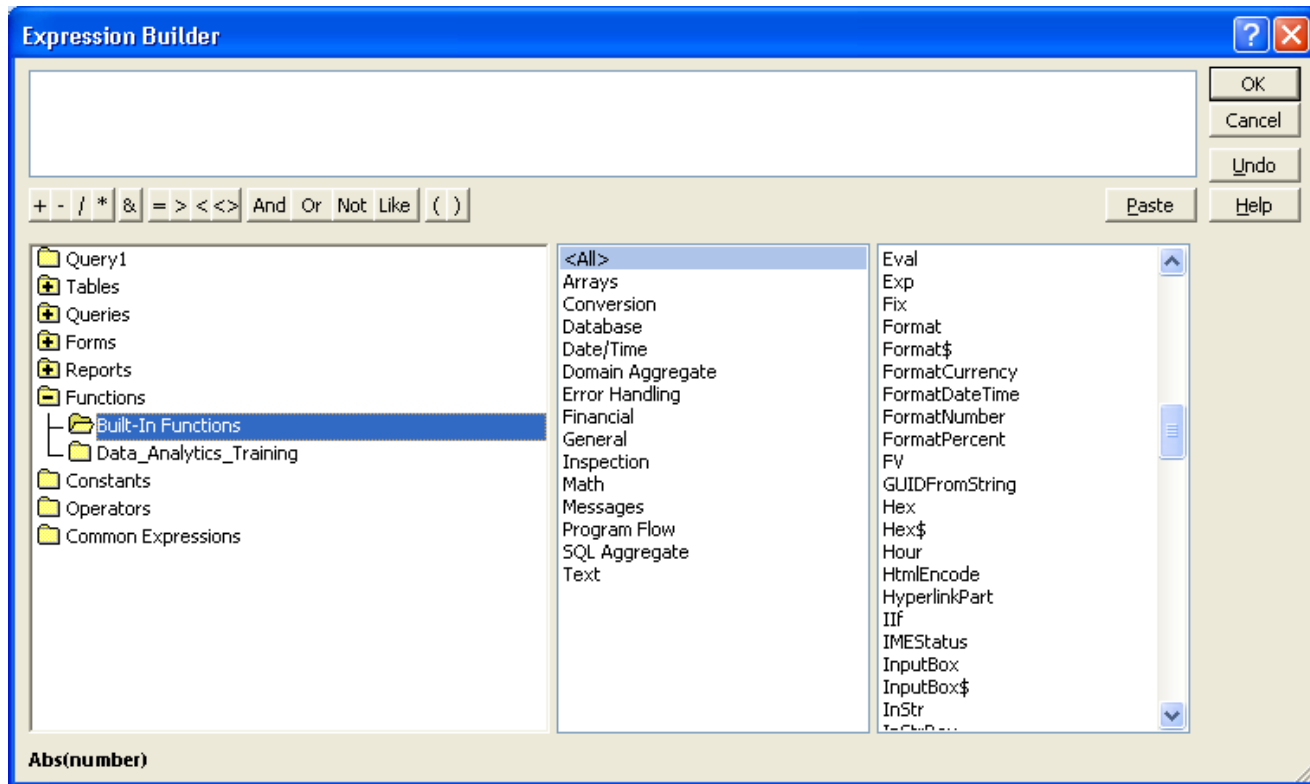
- Built-in Features that allow for analyzing and manipulation of data
- General Format
 - Function Name (variables/parameters)
 - Parameters can be 0 to multiple values
 - They allow for the formulation of complex queries

How Many functions in Access

1. Less than 50
2. 50 – 80
3. 80 – 120
4. More than 120
5. \$1 Mill



Expression Builder accesses Functions



MS Access Functions (real fun)

- **Left** (returns a string of a certain length starting from the left, e.g. Dav from David)
 - **Syntax = Left(*string*, *length*)**
- **Right** (returns a string of a certain length starting from the right, e.g. rip from drip)
 - **Syntax = Right(*string*, *length*)**
- **Mid** (returns a string of a certain length starting from one position and ending at another, e.g. **not** from **another**)
 - **Syntax = Mid(*string*, *start*, [*length*])**

MS Access Functions (real fun cont'd)

- LTRIM (removes leading spaces from a string)
 - **Syntax = LTrim(*string*)**
- RTRIM (removes trailing spaces from a string)
 - **Syntax = RTrim(*string*)**
- TRIM (removes both leading and trailing spaces from a string)
 - **Syntax = Trim(*string*)**

MS Access Functions (real fun cont'd)

- **FormatDateTime** (formats a date/time value)
 - Syntax = **FormatDateTime**(*Date*[,*NamedFormat*])
 - Named Format can be

| Constant | Value | Description |
|----------------------|-------|--|
| vbGeneralDate | 0 | Display a date and/or time. If there is a date part, display it as a short date. If there is a time part, display it as a long time. If present, both parts are displayed. |
| vbLongDate | 1 | Display a date using the long date format specified in your computer's regional settings. |
| vbShortDate | 2 | Display a date using the short date format specified in your computer's regional settings. |
| vbLongTime | 3 | Display a time using the time format specified in your computer's regional settings. |
| vbShortTime | 4 | Display a time using the 24-hour format (hh:mm). |

MS Access Functions (real fun cont'd)

- FormatDateTime Example

dates_example : Select Query

| Date Example | General Date | Long Date | Short Date | Long Time | Short Time |
|-----------------------|-----------------------|---------------------------|------------|------------|------------|
| 11/10/2008 1:15:19 PM | 11/10/2008 1:15:19 PM | Monday, November 10, 2008 | 11/10/2008 | 1:15:19 PM | 13:15 |

Record: 2 of 2

| Field: | General Date: FormatDateTime([Date_Eg],0) | Long Date: FormatDateTime([Date_Eg],1) | Short Date: FormatDateTime([Date_Eg],2) | Long Time: Forma |
|-----------|---|--|---|-------------------------------------|
| Table: | | | | |
| Sort: | | | | |
| Show: | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| Criteria: | | | | |
| or: | | | | |

MS Access Functions (real fun cont'd)

- DatePart (returns the part of a date specified)

– Syntax =

DatePart(*interval*, *date*[,*firstdayofweek*[, *firstweekofyear*]])

- Interval can be

| Setting | Description |
|---------|-------------|
| yyyy | Year |
| q | Quarter |
| m | Month |
| y | Day of year |
| d | Day |
| w | Weekday |
| ww | Week |
| h | Hour |
| n | Minute |
| s | Second |

MS Access Functions (real fun cont'd)

- DateDiff (calculates the amount of time units elapsed between two dates)
 - Syntax = DateDiff(*interval, date1, date2[, firstdayofweek[, firstweekofyear]]*)

- Interval can be

| Setting | Description |
|---------|-------------|
| yyyy | Year |
| q | Quarter |
| m | Month |
| y | Day of year |
| d | Day |
| w | Weekday |
| ww | Week |
| h | Hour |
| n | Minute |
| s | Second |

MS Access Functions (real fun cont'd)

- **Hour** (returns the 24 hour value for the hour of the day)
 - Syntax = **Hour(*time*)**
- **Weekday** (returns a numeric value representing the day of the week)
 - Syntax = **Weekday(*date* [, *firstdayofweek*])**
- **WeekdayName** (returns the string representing the day of the week)
 - Syntax = **WeekdayName(*weekday* [, *abbreviate*] [, *firstdayofweek*])**

MS Access Functions (real fun cont'd)

- **Len** (returns the length of a string)
 - **Len(*string* | *varname*)**
- **IIF** (if a condition exists something is done, if not then something else is done)
 - Syntax = **Iif(*expr*, *truepart*, *falsepart*)**
- **&** is known as concatenation of joining. It is used to join strings together
 - Example “b” & “lame” = blame

Created Functions

- Importing the files : 1.10 a – c for import of these functions
- Calcworkdays1 (returns the number of workdays occurring between two dates assuming no holidays)
 - Syntax = calcworkdays1(StartDate, EndDate)
- GetWeekDay (returns the name of the day of the week: e.g. Monday)
 - Syntax = Day_Of_Wk («ConvertDate»)
- Seasons (returns the name of the season for which the day happens to occur: e.g. Spring)
 - Syntax = seasons («ConvertDate»)

Microsoft Visual Basic Editor

