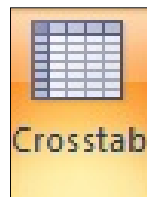
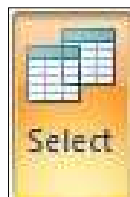


Select Queries & SQL

Select, Join, Aggregate, Crosstab, and Parameter



Structured Query Language (SQL)

- The industry-standard language for interacting with databases
 - Data Definition Language (DDL)
 - create, modify and delete database objects
 - Data Manipulation Language (DDL)
 - SELECT, INSERT, UPDATE, DELETE records
- Picky syntax, clause sequence, names, punctuation
- Access is a GUI to avoid SQL
- ACE/Jet... the Access database engines
 - implement features of SQL-89 and SQL-92 and add own enhancements
 - wildcard differences: * vs % and ? vs _
 - TRANSFORM & PIVOT
 - delimiters
 - "string"
 - #date#

Some Ways to use SQL in Access

■ Query's SQL View

- Access translates your Query Design into corresponding SQL statement
 - change the query design... Access translates to the equivalent SQL statement
- When you save a Query object, Access stores
 - the text of the SQL statement
 - the query results are not stored
 - query properties and column properties
 - query execution plan
 - database engine analyzes query, determines most efficient/quickest way to execute

■ Form/Report's RecordSource property

- SQL statement specifies fields & records available to form/report
- thus the form/report is not dependent on a saved query object
- can be set using VB code

```
Me.RecordSource = "SELECT * FROM tblRestaurant WHERE Zip='85258'"
```

- string within a string

Some Ways to use SQL in Access

■ List or Combo Box's RowSource property

- designates which columns and records are selected for display

■ Graph's RowSource property

- designates which columns and rows are selected for the chart

■ DoCmd.RunSQL Method

- runs an action query or data-definition query
- DoCmd.RunSQL "INSERT INTO tblCuisine(Cuisine) VALUES('Vegan');"

■ ADO (Callahan 12)

- rst.Open "SELECT * FROM tblClients"
- cmd.CommandText= "UPDATE Issues SET AssignedTo = 9 WHERE AssignedTo = 20"

■ DAO

- CurrentDB.Execute "DELETE * FROM emp WHERE fired = Yes"
- dbsHR.OpenRecordset("SELECT * FROM Employees", , dbReadOnly)
- qdfJuniorEarners.SQL = "SELECT * FROM titleauthor " & _
" WHERE royalty <30"

The SELECT Statement

Clause	Description
SELECT <i>column(s)</i>	Identifies which fields to extract & calculated columns
FROM <i>table(s)</i>	Identifies the source of data. If the statement uses multiple tables, use a JOIN clause to indicate how to join records. <ul style="list-style-type: none">• INNER JOIN combines records from two tables when there are matching values in a common field. If a row in one table doesn't match a row in the other it will not appear in the query results.• LEFT OUTER JOIN includes all records from the left table in the SQL statement, even if there are no matching records in the right table.• RIGHT OUTER JOIN includes all records from the right table in the SQL statement, even if there are no matching records in the left table.
[WHERE <i>criteria</i>]	Restricts the rows selected to participate in the query
[GROUP BY <i>column(s)</i>]	Summarizes the results by groups. At least one of the columns in the SELECT clause must include an aggregate function (SUM, COUNT, AVG, MAX, etc.)
[HAVING <i>constraint(s)</i>]	Defines which groups are returned
[ORDER BY <i>column(s)</i>]	Sorts results by one or more columns

Practice Time: SQL Select Queries

- Create a new query in Design view but close the Show Table dialog without selecting any tables. Switch to SQL view and enter the following SQL statement:

```
SELECT *  
FROM tblRestaurant;
```

- Run the query... it shows all fields and all restaurants
- Save the query as **qryAllRestaurants**

Practice Time: SQL Select Queries

```
SELECT field(s)
FROM table(s)
WHERE criteria
ORDER BY sortfield(s)
```

- Switch to SQL view and save as `qryRestaurantPhoneList`
- Modify the query's SQL statement

```
SELECT Restaurant, City, Zip, Phone, HomePage
FROM tblRestaurant
WHERE Phone Is Not Null
ORDER BY Zip;
```

- Run the query, it shows 5 fields, each row has a Phone, sorted by Zip
- Switch to Design View and inspect the query design
- Add `tlkpPriceCategory` to the query, add Price and SortOrder as new columns to query
- Run the query... it now returns PriceCategory info
- Switch to SQL View and notice the FROM clause

```
FROM tblRestaurant INNER JOIN tlkpPriceCategory
ON tblRestaurant.PriceCategoryID = tlkpPriceCategory.PriceCategoryID
```

- Save as `qryRestaurantPhoneListWithPriceCategory`

Inner Join Concepts

- Rows from one table are included in the result set only if there is a matching row in the other table
 - when a row in one table **doesn't match** any rows in the other table, it **won't appear** in the query results
 - the **missing rows** phenomenon
- The default join type
- aka Equijoin
- Practice Time
 - open `tblPosters` and determine how many records there are
 - new query `qryPostersAndReviews` showing Handle, MemberSince, ReviewID, ReviewDate, Stars, and RestaurantID
 - run the query
 - several posters were not displayed... since they haven't written a review
 - switch to SQL View

```
FROM tblPoster INNER JOIN tblReview
ON tblPoster.PosterID = tblReview.PosterID;
```

```
FROM tblPoster LEFT OUTER JOIN tblReview
ON tblPoster.PosteerID = tblReview.PosteerID;
```

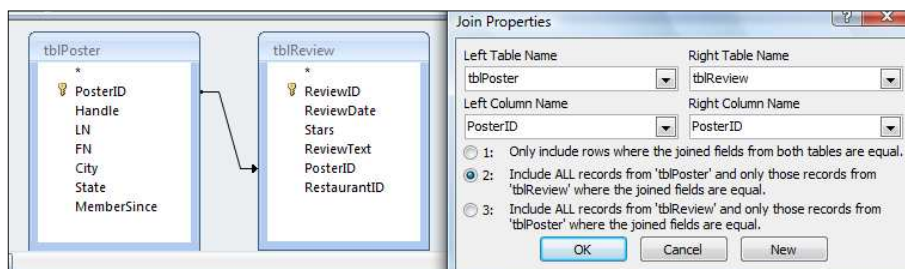
Outer Join Concepts

- Access selects all records from one table (driving table) even if there are no matching records in the other table
 - rows that do match a row in other table are shown, as usual
 - displays Null for fields when no matching row exists in non-driving table
- Driving Table
 - the table whose full set of rows is to be selected
- Practice Time
 - open qryPostersAndReviews in Design View
 - save as qryAllPostersAndReviews modify the join type to show all posters, even if there is no matching Review (make Poster the driving table)
 - run qryAllPostersAndReviews
 - which rows have Nulls?
 - which columns have Nulls?
 - how can we show only posters who have no reviews?
 - add an Is Null criteria on the non-driving table's primary key
 - modify to show only inactive posters, then save as qryInactivePosters

Setting the Join Type

```
FROM tblPoster INNER JOIN tblReview
ON tblPoster.PosteerID = tblReview.PosteerID;
```

```
FROM tblPoster LEFT OUTER JOIN tblReview
ON tblPoster.PosteerID = tblReview.PosteerID;
```

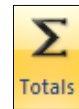


- Steps to control the join type:
 1. Double-click the join line
 2. Select the desired join type
- Arrowhead indicates an outer join & arrow points to non-driving table
- Caution: Don't include tables that lack a join line
 - Cartesian Product
 - matches every row in the left table to every row in the right table

Practice Time: Inner and Outer Joins

- Open tlkpPriceCategory and enter two new price categories:
 - **\$60 to \$80** and with SortOrder **4**
 - **\$80 to \$100** and with SortOrder **5**
- Close tlkpPriceCategory
- Create new qryRestaurantsAndPriceCategory showing RestaurantID, Restaurant name, Price, SortOrder sorted by ascending SortOrder
 - note that the new price categories are not displayed... since no restaurants are associated with these new price categories
- Switch to Design View
- Save as qryUnusedPriceCategories
- Modify to make tlkpPriceCategory the driving table
- Run qryUnusedPriceCategories
 - the two new price categories are displayed... but with Null for fields from tblRestaurant since no restaurants are associated with this price category
- Modify to show only unused price categories
- Save as qryUnusedPriceCategories

Aggregate Queries



- Aggregate Functions
 - arithmetic operations performed to summarize records
 - examples: avg, count, sum, min, max
- Totals Button
 - Total row: Sum, Avg, Min, Count, etc.
 - running the query produces a **single** row with aggregates displayed
 - use a separate column for each summary
- Practice Time
 - create a query to show the number of reviews, average star rating, highest star rating, lowest star rating. Save as qryReviewSummary. Run it.
 - switch to SQL View and notice
 - aggregate functions in the SELECT clause
 - AS keyword for column aliases

qryReviewSummary

Number of Reviews	Average Rating	Highest Rating
3	3.66666666666667	

qryReviewSummary

tblReview

- ReviewID
- ReviewDate
- Stars
- ReviewText
- PosterID
- RestaurantID

Field:	Number of Reviews: ReviewID	Average Rating: Stars	Highest Rating: ReviewID
Table:	tblReview	tblReview	tblReview
Total:	Count	Avg	Max
Sort:			
Show:	<input type="checkbox"/>	<input type="checkbox"/>	
Criteria:			

qryReviewSummary

```
SELECT Count(tblReview.ReviewID) AS [Number of Reviews],
Average(tblReview.Stars) AS [Average Rating],
Min(tblReview.Stars) AS [Lowest Rating]
FROM tblReview;
```

Aggregate Queries: GROUP BY

- Forms **subgroups** based on a specified field/expression
- Produces a summary row for **each value** of the grouping field(s)
- **Practice Time**
 - create a query that shows each cuisine and the number of restaurants associated. Save as **qryCuisineDistribution** when correct.
 - switch to SQL View and notice
 - aggregate functions in the SELECT clause
 - **AS** keyword for column aliases
 - **GROUP BY** clause

Cuisine	Number of Restaurants
American	16
Sandwiches	8
Chinese	6
Italian	6
Breakfast/Lunch	6
Desserts	5
Ice cream/Frozen novelties	4
Pizza	3

Field:	CuisineID	Number of Restaurants: CuisineID
Table:	tblRestaurantCuisine	tblRestaurantCuisine
Total:	Group By	Count
Sort:		Descending
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:		

```

qryCuisineDistribution
SELECT tblRestaurantCuisine.CuisineID, Count(tblRestaurantCuisine.CuisineID) AS [Number of Restaurants]
FROM tblRestaurantCuisine
GROUP BY tblRestaurantCuisine.CuisineID
ORDER BY Count(tblRestaurantCuisine.CuisineID) DESC;

```

Aggregate Queries: HAVING

- Limits which **groups** are returned
- Must used in with **GROUP BY**
- Is evaluated **after** **GROUP BY**
- **Practice Time**
 - modify **qryCuisineDistribution** to display only cuisines with at least 5 restaurants
 - save as **qryHighFrequencyCuisines**
 - switch to SQL View and notice
 - **HAVING** clause
 - sequence of clauses

Cuisine	Number of Restaurants
American	16
Sandwiches	8
Italian	6
Chinese	6
Breakfast/Lunch	6
Desserts	5

Field:	CuisineID	Number of Restaurants: CuisineID
Table:	tblRestaurantCuisine	tblRestaurantCuisine
Total:	Group By	Count
Sort:		Descending
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:		>=5

```

qryHighFrequencyCuisines
SELECT tblRestaurantCuisine.CuisineID, Count(tblRestaurantCuisine.CuisineID) AS [Number of Restaurants]
FROM tblRestaurantCuisine
GROUP BY tblRestaurantCuisine.CuisineID
HAVING (((Count(tblRestaurantCuisine.CuisineID))>=5))
ORDER BY Count(tblRestaurantCuisine.CuisineID) DESC;

```

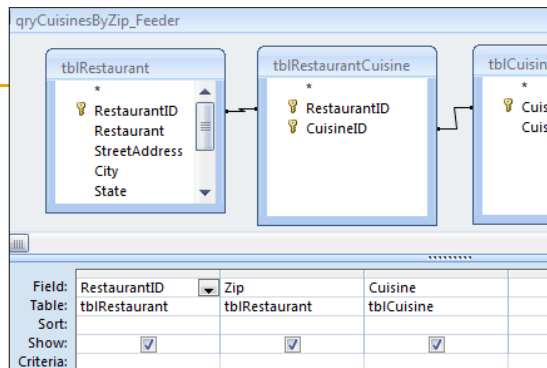
Crosstab Query Concepts

- Summarizes data in a row & column format
 - displays the **joint distribution** of two or more variables/expressions and an aggregate measure for each joint occurrence
- Produces a **snapshot**
 - a non-updatable recordset
- Crosstab Query Wizard steps:
 1. identify data source (table or query)
 - if the crosstab summarizes data from >1 table, first build a **feeder query** that joins the tables, then base the crosstab query on the feeder query
 2. identify up to 3 fields for row headings
 3. identify one field for column headings
 4. identify field to summarize, select aggregate measure, whether to include row sums
 5. name the query

qxtbCuisinesByZip				
Cuisine	Total Restaurants	85251	85258	85260
American	16		8	8
Asian	2			2
Bakery	1		1	
Barbecue	1		1	

Practice Time: Crosstab Query

- Create feeder query named **qryCuisinesByZip_Feeder**
- Use Crosstab Wizard to create **qxtbCuisinesByZip**



qxtbCuisinesByZip				
Cuisine	Total Restaurants	85251	85258	85260
American	16		8	8
Asian	2			2
Bakery	1		1	
Barbecue	1		1	
Breakfast/Lunch	6		5	1
Chinese	6		3	3
Coffee/Tea house	2		1	1
Contemporary Ameri	2		1	1

Practice Time: Crosstab Query

- Switch to Design View
 - Row Heading
 - Column Heading
 - Value
- Switch to SQL View
 - TRANSFORM clause
 - the aggregate measure
 - GROUP BY
 - the row headings
 - PIVOT clause
 - the column headings

Cuisine	Total Restaurants	85251	85258	85260
American	16	8	8	8
Asian	2			2
Bakery	1	1		
Barbecue	1	1		
Breakfast/Lunch	6	5		1
Chinese	6	3		3
Coffee/Tea house	2	1		1

Field:	Cuisine	Zip	RestaurantID	Total Restaurants
Table:	qryCuisinesByZip_Feeder	qryCuisinesByZip_Feeder	qryCuisinesByZip_Feeder	qryCuisinesByZip_Feeder
Total:	Group By	Group By	Count	Count
Crosstab:	Row Heading	Column Heading	Value	Row Heading
Sort:				
Criteria:				

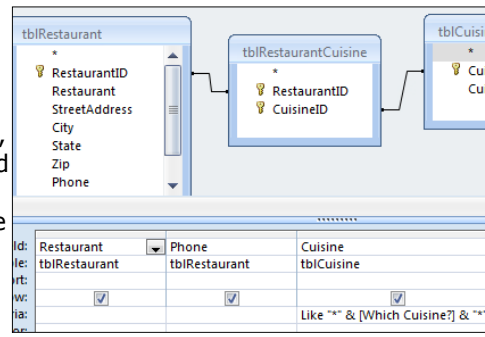
```

TRANSFORM Count(qryCuisinesByZip_Feeder.RestaurantID) AS CountOfRestaurantID
SELECT qryCuisinesByZip_Feeder.Cuisine, Count(qryCuisinesByZip_Feeder.RestaurantID) AS [Total Restaurants]
FROM qryCuisinesByZip_Feeder
GROUP BY qryCuisinesByZip_Feeder.Cuisine
PIVOT qryCuisinesByZip_Feeder.Zip;
    
```

Parameter Query

- Ideal when run a query with different criteria values each time
- Prompt user to enter value which is substituted prior to query execution
 - place [prompt message] in Criteria row to define prompt
 - can have multiple prompt placeholders
- Practice Time
 - create qryRestaurantCuisineSearch that prompts for a cuisine keyword, then shows restaurant & phone and cuisine for each restaurant offering the cuisine. Your solution should be flexible enough to find cuisines where the keyword is a part of the cuisine name.

Restaurant	Phone	Cuisine
Amarone	480-391-2222	Pizza/Italian
Uncle Sam's	480-419-1111	Pizza/Italian
Andreoli Italian Grocer	480-614-1980	Italian
Amarone	480-391-2222	Italian
Galileo Bread Emporium	480-998-7280	Italian
Voce Ristorante & Lounge	480-609-0429	Italian



Practice Time: Parameter Query

- Create a query named `qryRestaurantStarSearch` that shows ReviewDate, Stars, ReviewText, PosterID, Restaurant name, and Phone for each review that meets or exceeds the number entered in response to the prompt.

Review Date	Stars	Review Text	PosterID	Restaurant	Phone
7/5/2009	5	An elegant venue; the music is	Skipper	Voce Ristorante & Lounge	480
7/20/2009	4	Clean bathrooms; need more	babysitter	Ben & Jerry's	480

Practice Time

- Create a new Blank form
 - `frmRestaurantByZipCodeSelector`
 - no Record Source (unbound)
 - no Record Selectors
 - no Navigation Buttons
- Place the **combo box**
 - Name: `cboZip`
 - RowSource set to `SELECT DISTINCT zip FROM tblRestaurant ORDER BY zip;`
 - set label's caption
 - Form View to verify
- Place the **list box**
 - name it
 - set properties
 - RowSource : Null

Property Sheet	
Selection type: List Box	
Format	Data Event Other All
Visible	Yes
Column Count	5
Column Widths	0";2.5";2";1";1"
Column Heads	Yes
Width	6.5"
Height	3.5"

Property Sheet	
Selection type: List Box	
Format	Data Event Other All
Control Source	
Row Source	
Row Source Type	Table/Query
Bound Column	1
Allow Value List Edits	Yes
List Items Edit Form	

Practice Time

- Enter this code to make list box show restaurants in the selected zip code
- Notice the **concatenation** to harvest the zip code value from the combo box
 - string within a string

```
Private Sub cboZip_AfterUpdate()  
On Error GoTo err_handler  
  
Dim strSQL As String  
  
strSQL = "SELECT RestaurantID, Restaurant, StreetAddress, City, Phone " & _  
" FROM tblRestaurant " & _  
" WHERE Zip ='" & Me.cboZip & "'" & _  
  
'Debug.Print strSQL  
  
Me.lstRestaurants.RowSource = strSQL  
  
exit_point:  
Exit Sub  
  
err_handler:  
Select Case Err.Number  
  
Case Else  
MsgBox Err.Number & ": " & Err.Description  
  
End Select  
  
Resume exit_point  
  
End Sub
```

```
reetAddress, City, Phone FROM tblRestaurant WHERE Zip = '85258'  
reetAddress, City, Phone FROM tblRestaurant WHERE Zip = '85260'  
reetAddress, City, Phone FROM tblRestaurant WHERE Zip = Me.cboZip
```