Scientific Data Analysis: SQL

Jarek Szlichta

http://data.science.uoit.ca/

Introduction to SQL

- **SQL is a standard language for accessing and manipulating databases**
- What is SQL?
  - SQL stands for Structured Query Language
  - SQL lets you gain access and control over databases
  - SQL is an ANSI (American National Standards Institute) standard

What can SQL do?

- **Execute queries against a database**
  - Retrieve data from a database
  - Insert, update, delete records in a database
  - Create new databases
- **Set permissions on tables, procedures, and views**

SQL is a Standard – but …

- **Although SQL is an ANSI standard, there are different versions of the SQL language**
- However, to be compliant with the ANSI standard, they all support at least the major commands
  - such as **SELECT, UPDATE, DELETE, INSERT, WHERE** in a similar manner

Using SQL in Your Web Site

- **To build a web site that shows data from a database, you will need:**
  - A relational database management system (e.g., **Oracle, IBM DB2, SQL Server, MySQL, PostgreSQL, Access, …**)
  - Use a server-side scripting language, like PHP or ASP
  - **Use SQL to get the data you want**
  - Use HTML / CSS

RDBMS

- **RDBMS stands for Relational Database Management System**
  - RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access
- **The data in RDBMS is stored in database objects called tables**
  - A table is a collection of related data entries and it consists of columns and rows

Database Table

- A database most often contains one or more tables
  - **Each table is identified by a name (e.g. "Customers" or "Orders")**
  - **Table consists of attributes and contain records (rows) with data**

"Customers" Table

- The table below contains five records (one for each customer) and seven columns.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

SQL Select Statement

- **The SELECT statement is used to select data from a database**
  - The result is stored in a result table,
  - The result table is also often called the result-set
- We will use W3C
  - Go to the following link (press “Try it Yourself” to run the actual queries!):
    - http://www.w3schools.com/sql/

SELECT Column Example

- The following SQL statement selects the "CustomerName" and "City" columns from the "Customers" table:

```
SELECT CustomerName, City
FROM Customers
```

- The following SQL statement selects all the columns from the "Customers" table:

```
SELECT *
FROM Customers
```

Result

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique	24, place Kléber	Strasbourg	67000	France

SELECT DISTINCT

- **The SELECT DISTINCT statement is used to return only distinct (different) values**
  - In a table, a column may contain many duplicate values
  - and sometimes you only want to list the different (distinct) values
  - The DISTINCT keyword can be used to return only distinct (different) values

SELECT DISTINCT Example

- The following SQL statement selects only the distinct values from the "City" columns from the "Customers" table:

```
SELECT DISTINCT City
FROM Customers
```

SELECT DISTINCT Example

SELECT…

Number of Records: 91

City
Berlin
México D.F.
México D.F.
London
Luleå
Mannheim
Strasbourg
Madrid

SELECT DISTINCT…

Number of Records: 69

City
Berlin
México D.F.
London
Luleå
Mannheim
Strasbourg
Madrid
Marseille

Where Clause

- **The WHERE clause is used to filter records**
  - The WHERE clause is used to extract only those records that fulfill a specified criterion
    - e.g., customers from the city Berlin

Where Clause Example

- The following SQL statement selects all the customers from the country "Mexico", in the "Customers" table:

```
SELECT *
FROM Customers
WHERE Country='Mexico'
```

Where Clause Example

Number of Records: 5

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
13	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	05022	Mexico
58	Pericles Comidas clásicas	Guillermo Fernández	Calle Dr. Jorge Cash 321	México D.F.	05033	Mexico
80	Tortuga Restaurante	Miguel Angel Paolino	Avda. Azteca 123	México D.F.	05033	Mexico

Text Fields vs. Numeric Fields

- SQL requires quotes around text values
- However, numeric fields should not be enclosed in quotes:

```
SELECT *
FROM Customers
WHERE CustomerID=1;
```

Operators in the WHERE Clause

- **The following operators can be used in the WHERE clause:**

Operator	Description
=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

SQL AND & OR Operators

- **The AND & OR operators are used to filter records based on more than one condition**
  - The AND operator displays a record if both the first condition AND the second condition are true.
  - The OR operator displays a record if either the first condition OR the second condition is true

AND Operator Example

- The following SQL statement selects all customers from the country "Germany" AND the city "Berlin", in the "Customers" table:

```
SELECT * FROM Customers
WHERE Country='Germany'
AND City='Berlin'
```

AND Operator Example

Number of Records: 1

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany

OR Operator Example

- The following SQL statement selects all customers from the city "Berlin" OR "München", in the "Customers" table:

```
SELECT * FROM Customers
WHERE City='Berlin'
OR City='München'
```

OR Operator Example

Number of Records: 2

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
25	Frankenversand	Peter Franken	Berliner Platz 43	München	80805	Germany

Combining AND & OR

- **You can also combine AND and OR (use parenthesis to form complex expressions)**
- The following SQL statement
  - selects all customers from the country "Germany“
  - AND the city must be equal to "Berlin" OR "München", in the "Customers" table:

```
SELECT * FROM Customers
WHERE Country='Germany'
AND (City='Berlin' OR City='München')
```

ORDER BY

- **The ORDER BY keyword is used to sort the result-set**
  - The ORDER BY keyword is used to sort the result-set by one or more columns
- The ORDER BY keyword sorts the records in ascending order by default
  - To sort the records in a descending order, you can use the DESC keyword.

ORDER BY Example

- The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" column:

```
SELECT * FROM Customers
ORDER BY Country;
```

ORDER BY Example

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
64	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	1010	Argentina
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
59	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	5020	Austria
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium
76	Suprêmes délices	Pascale	Boulevard Tirou, 255	Charleroi	B-6000	Belgium

ORDER BY DESC Example

- The following SQL statement selects all customers from the "Customers" table,
  - sorted DESCENDING by the "Country" column:

```
SELECT * FROM Customers
ORDER BY Country DESC;
```

ORDER BY DESC Example

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
47	LINO-Delicateses	Felipe Izquierdo	Ave. 5 de Mayo Porlamar	I. de Margarita	4980	Venezuela
33	GROSELLA-Restaurante	Manuel Pereira	5ª Ave. Los Palos Grandes	Caracas	1081	Venezuela
35	HILARIÓN-Abastos	Carlos Hernández	Carrera 22 con Ave. Carlos Soublette #8-35	San Cristóbal	5022	Venezuela
46	LILA-Supermercado	Carlos González	Carrera 52 con Ave. Bolívar #65-98 Llano Largo	Barquisimeto	3508	Venezuela
32	Great Lakes Food Market	Howard Snyder	2732 Baker Blvd.	Eugene	97403	USA

ORDER BY Several Columns Example

- The following SQL statement selects all customers from the "Customers" table, sorted ascending by the "Country" and ascending by the "CustomerName" column:

```
SELECT * FROM Customers
ORDER BY Country, CustomerName;
```

ORDER BY Several Columns Example

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	1010	Argentina
64	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
59	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	5020	Austria
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium

ORDER BY Several Columns Example

- The following SQL statement selects all customers from the "Customers" table,
  - sorted ascending by the "Country" and descending by the "CustomerName" column:

```
SELECT * FROM Customers
ORDER BY Country ASC,
        CustomerName DESC;
```

ORDER BY Several Columns Example

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
64	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	1010	Argentina
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
59	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	5020	Austria
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
76	Suprêmes délices	Pascale Cartrain	Boulevard Tirou, 255	Charleroi	B-6000	Belgium

SQL Insert Into

- **The INSERT INTO statement is used to insert new records in a table**
- It is possible to write the INSERT INTO statement in two forms
  - The first form does not specify the column names where the data will be inserted, only their values
  - The second form specifies both the column names and the values to be inserted

Insert Example

- Assume we wish to insert a new row in the "Customers" table.
- We can use the following SQL statement:

```
INSERT INTO Customers
    (CustomerName, ContactName, Address, City,
        PostalCode, Country)
VALUES ('Cardinal','Tom B. Erichsen','Skagen
    21','Stavanger','4006','Norway')
```

Insert Data Only in Specified Columns

- It is also possible to only insert data in specific columns.
- Following SQL statement will insert a new row, but only insert data in the "CustomerName", "City", and "Country" columns
  - (and the CustomerID field will of course also be updated automatically):

```
INSERT INTO Customers
(CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

SQL Update

- The UPDATE statement is used to update existing records in a table
- Assume we wish to update the customer "Alfreds Futterkiste" with a new contact person and city:

```
UPDATE Customers
SET ContactName='Alfred Schmidt',
        City='Hamburg'
WHERE CustomerName='Alfreds Futterkiste'
```

Update Warning

- Be careful when updating records. If we had omitted the WHERE clause, in the example above, like this:
  - **UPDATE Customers**
    SET ContactName='Alfred Schmidt',
    City='Hamburg'
- How would the customer table would look like?

SQL DELETE

- The DELETE statement is used to delete rows in a table
- Assume we wish to delete the customer "Alfreds Futterkiste" from the "Customers" table:

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste'
      AND ContactName='Maria Anders';
```

Create Table and SQL TOP Clause

- W3C
  - http://www.w3schools.com/sql/
- **Create Table** is used to create a table in a database
- **The SELECT TOP clause is used to specify the number of records to return**
  - The SELECT TOP clause can be very useful on large tables with thousands of records
  - Returning a large number of records can impact on performance

SQL TOP Example

- The following SQL statement selects the four first records from the "Customers" table:

```
SELECT TOP 4 *
FROM Customers;
```

- The he following SQL statement selects the first 50% of the records from the "Customers" table:

```
SELECT TOP 50 PERCENT *
FROM Customers;
```

TOP 4

Number of Records: 4

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

SQL Like

- **The LIKE operator is used in a WHERE clause to search for a specified pattern in a column**
- The following SQL statement selects all customers with a city starting with the letter "s":

```
SELECT * FROM Customers
WHERE City LIKE 's%'
```

SQL Like

Number of Records: 14

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
15	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	São Paulo	05432-043	Brazil
21	Familia Arquibaldo	Aria Cruz	Rua Orós, 92	São Paulo	05442-030	Brazil
30	Godos Cocina Típica	José Pedro Freyre	C/ Romero, 33	Sevilla	41101	Spain
35	HILARIÓN-Abastos	Carlos Hernández	Carrera 22 con Ave. Carlos Soublette #8-35	San Cristóbal	5022	Venezuela
45	Let's Stop N Shop	Jaime Yorres	87 Polk St. Suite 5	San Francisco	94117	USA
59	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	5020	Austria
62	Queen Cozinha	Lúcia Carvalho	Alameda dos Canàrios, 891	São Paulo	05487-020	Brazil
70	Santé Gourmet	Jonas Bergulfsen	Erling Skakkes gate 78	Stavern	4110	Norway
81	Tradição Hipermercados	Anabela Domingues	Av. Inês de Castro, 414	São Paulo	05634-030	Brazil
86	Die Wandernde Kuh	Rita Müller	Adenauerallee 900	Stuttgart	70563	Germany
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway
93	Cardinal	*null*	*null*	Stavanger	*null*	Norway

SQL Wildcards

- A wildcard % character can be used to substitute for any other character(s) in a string.
- A wildcard _ character can be used to substitute for a single character.
- The following SQL statement selects all customers with a City starting with "ber":

```
SELECT * FROM Customers
WHERE City LIKE 'ber%'
```

SQL Wildcards

Number of Records: 3

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
14	Chop-suey Chinese	Yang Wang	Hauptstr. 29	Bern	3012	Switzerland
49	Magazzini Alimentari Riuniti	Giovanni Rovelli	Via Ludovico il Moro 22	Bergamo	24100	Italy

SQL IN

- **The IN operator allows to specify multiple values in a WHERE clause**
- The following SQL statement selects all customers with a City of "Paris" or "London":

```
SELECT * FROM Customers
WHERE City IN ('Paris','London')
```

SQL IN

Number of Records: 8

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
16	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London	WX1 6LT	UK
19	Eastern Connection	Ann Devon	35 King George	London	WX3 6FW	UK
53	North/South	Simon Crowther	South House 300 Queensbridge	London	SW7 1RZ	UK
57	Paris spécialités	Marie Bertrand	265, boulevard Charonne	Paris	75012	France
72	Seven Seas Imports	Hari Kumar	90 Wadhurst Rd.	London	OX15 4NB	UK

SQL Between

- **The BETWEEN operator is used to select values within a range**
- The following SQL statement selects all products with a price BETWEEN 10 and 20:

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

SQL Between

Number of Records: 29

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
15	Genen Shouyu	6	2	24 - 250 ml bottles	15.5
16	Pavlova	7	3	32 - 500 g boxes	17.45
21	Sir Rodney's Scones	8	3	24 pkgs. x 4 pieces	10
25	NuNuCa Nuß-Nougat-Creme	11	3	20 - 450 g glasses	14
31	Gorgonzola Telino	14	4	12 - 100 g pkgs	12.5

SQL Aliases

- **SQL aliases are used to temporarily rename a table or a column heading**
- The following SQL statement specifies two aliases, one for the CustomerName column and one for the ContactName column
  - It requires double quotation marks or square brackets if the column name contains spaces:

```
SELECT CustomerName AS Customer,
       ContactName AS [Contact Person]
FROM Customers;
```

SQL Aliases

Number of Records: 91

Customer	Contact Person
Alfreds Futterkiste	Maria Anders
Ana Trujillo Emparedados y helados	Ana Trujillo
Antonio Moreno Taquería	Antonio Moreno
Around the Horn	Thomas Hardy
Berglunds snabbköp	Christina Berglund
Blauer See Delikatessen	Hanna Moos
Blondel père et fils	Frédérique Citeaux
Bólido Comidas preparadas	Martín Sommer

SQL Joins

- **An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them**
- The most common type of join is: **SQL INNER JOIN (simple join)**.
  - An SQL INNER JOIN returns all rows from multiple tables where the join condition is met.

SQL INNER JOIN Example

```
SELECT Orders.OrderID,
    Customers.CustomerName,
    Orders.OrderDate, Orders. OrderID,
    Customers.OrderID
FROM Orders
INNER JOIN Customers
ON Orders.CustomerID=Customers.CustomerID
```

SQL INNER JOIN Example

Number of Records: 196

OrderID	CustomerName	OrderDate	Orders.CustomerID	Customers.CustomerID
10308	Ana Trujillo Emparedados y helados	9/18/1996	2	2
10365	Antonio Moreno Taquería	11/27/1996	3	3
10383	Around the Horn	12/16/1996	4	4
10355	Around the Horn	11/15/1996	4	4
10278	Berglunds snabbköp	8/12/1996	5	5
10280	Berglunds snabbköp	8/14/1996	5	5
10384	Berglunds snabbköp	12/16/1996	5	5
10436	Blondel père et fils	2/5/1997	7	7

Different SQL JOINs

- **INNER JOIN**: Returns all rows when there is at least one match in BOTH tables
- **LEFT JOIN**: Return all rows from the left table, and the matched rows from the right table
- **RIGHT JOIN**: Return all rows from the right table, and the matched rows from the left table
- **FULL JOIN**: Return all rows when there is a match in ONE of the tables

SQL Union

- **The SQL UNION operator combines the result of two or more SELECT statements**
- The following SQL statement selects all the **different** cities (only distinct values) from the "Customers" and the "Suppliers" tables:

```
SELECT City FROM Customers
      UNION
SELECT City FROM Suppliers
ORDER BY City;
```

NULL Values

- **NULL values represent missing unknown data.**
- By default, a table column can hold NULL values.
- NULL values are treated differently from other values.
- NULL is used as a placeholder for unknown or inapplicable values.

Persons Table with NULL values

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola		Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari		Stavanger

How Can We Test for NULL values?

- **It is not possible to test for NULL values with comparison operators, such as =, <, or <>**
- We will have to use the IS NULL and IS NOT NULL operators instead
- Select only the records with NULL values in the "Address" column

```
SELECT LastName, FirstName, Address
FROM Persons
WHERE Address IS NULL
```

Result

- The result-set will look like this:

LastName	FirstName	Address
Hansen	Ola	
Pettersen	Kari	

SQL IS NOT NULL

- How do we select only the records with no NULL values in the "Address" column?
- We will have to use the IS NOT NULL operator:

```
SELECT LastName, FirstName, Address
FROM Persons
WHERE Address IS NOT NULL
```

LastName	FirstName	Address
Svendson	Tove	Borgvn 23

SQL Aggregate Functions

- **SQL aggregate functions return a single value, calculated from values in a column**
  - AVG() - Returns the average value
  - COUNT() - Returns the number of rows
  - FIRST() - Returns the first value
  - LAST() - Returns the last value
  - MAX() - Returns the largest value
  - MIN() - Returns the smallest value
  - SUM() - Returns the sum

SQL Aggregate Functions and Nested Queries

- The AVG() function returns the average value of a numeric column
- The following SQL statement gets the average value of the "Price" column from the "Products" table:

```
SELECT AVG(Price) AS PriceAverage
FROM Products
```

- The following SQL statement selects the "ProductName" and "Price" records that have an above average price:

```
SELECT ProductName, Price
FROM Products
WHERE Price >
        (SELECT AVG(Price) FROM Products)
```

SQL Count()

- The COUNT() function returns the number of rows that matches a specified criteria

```
SELECT COUNT(CustomerID) AS
        OrdersFromCustomerID7
FROM Orders
WHERE CustomerID=7;
```

SQL Count(*)

- **The COUNT(*) function returns the number of records in a table**
- The following SQL statement counts the total number of orders in the "Orders" table:

```
SELECT COUNT(*) AS NumberOfOrders
FROM Orders;
```

SQL First()

- **The FIRST() function returns the first value of the selected column**
- The following SQL statement selects the first value of the "CustomerName" column from the "Customers" table:

```
SELECT FIRST(CustomerName)
        AS FirstCustomer
FROM Customers;
```

SQL Last()

- The Last() function returns the last value of the selected column.
- The following SQL statement selects the last value of the "CustomerName" column from the "Customers" table:

```
SELECT LAST(CustomerName)
        AS FirstCustomer
FROM Customers;
```

SQL Max()

- The following SQL statement gets the largest value of the "Price" column from the "Products" table:

```
SELECT MAX(Price) AS HighestPrice
FROM Products;
```

SQL Min()

- **The MIN() function returns the smallest value of the selected column**
- The following SQL statement gets the smallest value of the "Price" column from the "Products" table:

```
SELECT MIN(Price) AS SmallestOrderPrice
FROM Products
```

SQL Sum()

- **The SUM() function returns the total sum of a numeric column**
- The following SQL statement finds the sum of all the "Quantity" fields for the "OrderDetails" table:

```
SELECT SUM(Quantity) AS TotalItemsOrdered
FROM OrderDetails;
```

SQL Group By

- **The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns:**
- The following query groups orders by “CustomerID”:

```
SELECT CustomerID
From Orders
Group by CustomerID
```

- The following query counts number of orders by each customer:

```
SELECT CustomerID, count(*)
From Orders
Group by CustomerID
```

SQL Group By

Number of Records: 74

CustomerID	counterOrders
2	1
3	1
4	2
5	3
7	4
8	1
9	3
10	4

SQL Having Clause

- **The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions**
- The following query counts number of orders by each customer and only selects the customers who made more than one order

```
SELECT CustomerID, count(*)
From Orders
Group by CustomerID
Having count(*) > 1
```

SQL UCASE()

- **The UCASE() function converts the value of a field to uppercase.**
- The following SQL statement converts the "CustomerName" column to uppercase:

```
SELECT UPCASE(CustomerName)
        AS Customer, City
FROM Customers;
```

SQL LCASE()

- **The LCASE() function converts the value of a field to lowercase.**
- The following SQL statement converts the "CustomerName" column to lowercase:

```
SELECT LCASE(CustomerName)
        AS Customer, City
FROM Customers;
```

Course Plug-in 3030

- If you want to learn more advanced and complex SQL techniques take CSCI 3030U: Database Systems and Concepts
  - The course covers fundamentals of relational algebra and database designs

SQL Reading List

- **Recommended**
  - http://www.w3schools.com/sql/default.asp
  - “Try it Yourself”!
- **Optional**
  - A Very Short History of Big Data
    - http://www.forbes.com/forbes/welcome/
    - http://www.forbes.com/sites/gilpress/2013/05/09/a-very-short-history-of-big-data/#19f443be55da/

SQL Quiz

- It is just a nice way to see how much you know, or do not know, about SQL
- Contains 25 questions
  - http://www.w3schools.com/sql/sql_quiz.asp