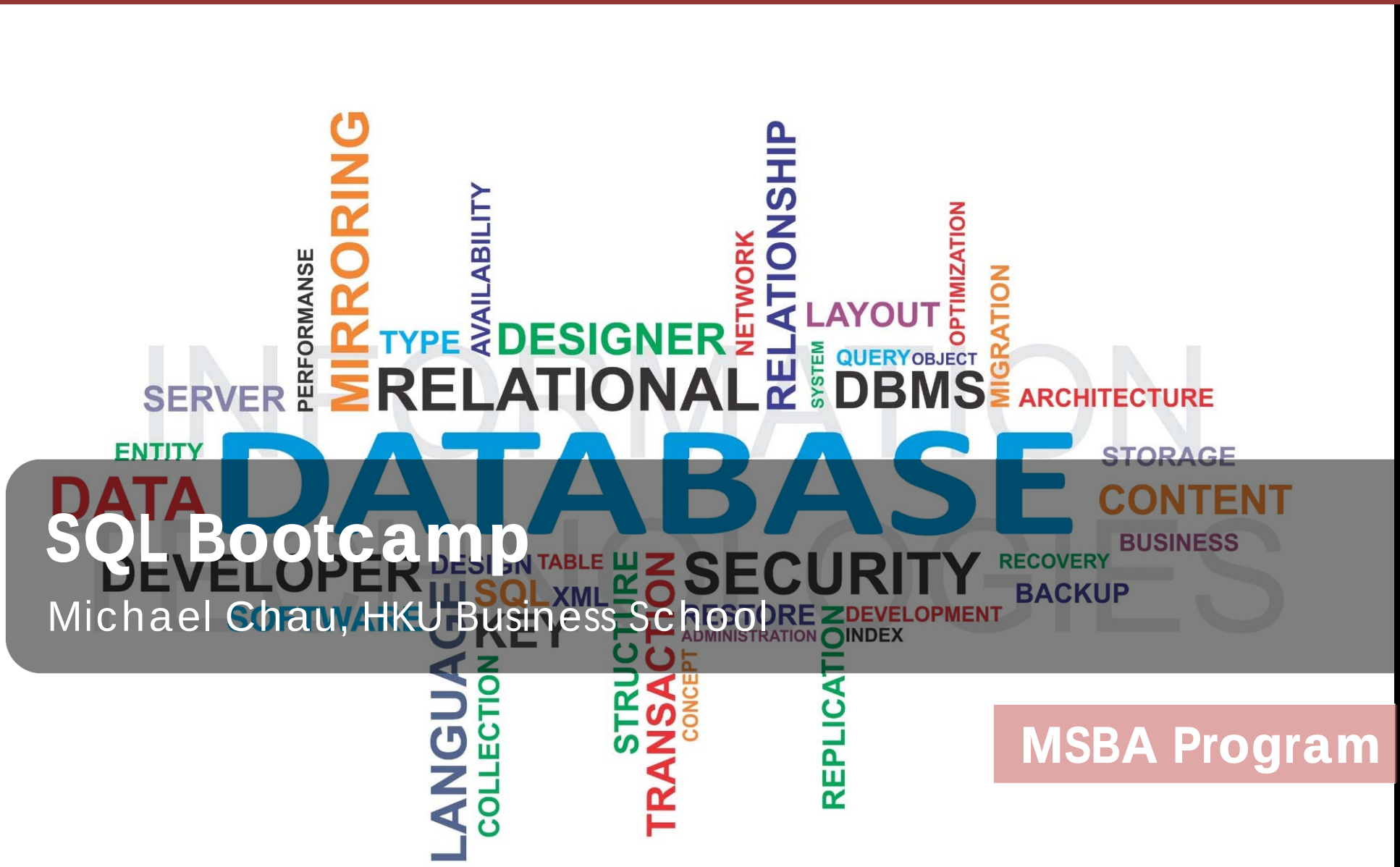


SQL Bootcamp

Michael Chau, HKU Business School

MSBA Program

About Myself – Michael Chau

- Professor, Innovation and Information Management, HKU Business School (Faculty of Business and Economics), The University of Hong Kong
- Former Warden, Lee Chi Hung Hall
- Doctoral degree in management information systems from the University of Arizona; Bachelor degree in computer science and information systems from the University of Hong Kong
- Research in big data, data mining, artificial intelligence, search engines, and social media analytics
- Published over 150 academic articles; cited over 8000 times
- Recipient of the IEEE ISI Leadership Award, INFORMS Design Science Award, HKU Outstanding Young Research Award (2014), and Knowledge Exchange Award (2013 & 2016)
- http://www.business.hku.hk/~mchau/
- Email: mchau@business.hku.hk


michaelchau8888

michaelchau

Outline

- Introduction to SQL
- SELECT statement
- Searching for patterns or ranges of values
- Sorting the results
- Aggregate functions and grouping
- Multiple table queries
- Insert, Update, and Delete statements
- Table definition

What is SQL?

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What Can SQL do?

- Execute queries against a database
- Retrieve data from a database
- Insert records in a database
- Update records in a database
- Delete records from a database
- Create new databases
- Create new tables in a database
- Create stored procedures in a database
- Create views in a database
- Set permissions on tables, procedures, and views

SQL is a Standard - BUT....

- Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.
- 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.

Database Tables

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

Database Tables

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

- Northwind Sample Database

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, SQLite, 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.

RDBMS

- Every table is broken up into smaller entities called fields. The fields in the Customers table consist of CustomerID, CustomerName, ContactName, Address, City, PostalCode and Country. A field is a column in a table that is designed to maintain specific information about every record in the table.
- A record, also called a row, is each individual entry that exists in a table. For example, there are 91 records in the Customers table. A record is a horizontal entity in a table.
- A column is a vertical entity in a table that contains all information associated with a specific field in a table.

SQL Statements

- Most of the actions you need to perform on a database are done with SQL statements.
- The following SQL statement selects all the records in the "Customers" table:

```
SELECT * FROM Customers;
```

Keep in Mind That...
SQL keywords are NOT case sensitive: select is the same as SELECT

SQL Statements

- Try it out!
- https://www.w3schools.com/sql/

Semicolon after SQL Statements?

- Some database systems require a semicolon at the end of each SQL statement.
- Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

Some of The Most Important SQL Commands

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database
- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

SELECT Statement

SELECT Statement

- The SELECT statement is used to select data from a database.
- The data returned is stored in a result table, called the result-set
- SELECT Syntax

```
SELECT column1, column2, ...
FROM table_name;
```

SELECT Statement

- Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

```
SELECT * FROM table_name;
```

Customers Table

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 Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence	12, rue des Bouchers	Marseille	13008	France

SELECT Column Example

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

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

Number of Records: 91

CustomerName	City
Alfreds Futterkiste	Berlin
Ana Trujillo Emparedados y helados	México D.F.
Antonio Moreno Taquería	México D.F.
Around the Horn	London
Berglunds snabbköp	Luleå
Blauer See Delikatessen	Mannheim
Blondel père et fils	Strasbourg
Bólido Comidas preparadas	Madrid
Bon app'	Marseille
Bottom-Dollar Marketse	Tsawassen

SELECT * Example

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

```
SELECT * FROM Customers;
```

Number of Records: 91

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 Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France

SELECT DISTINCT Statement

- SELECT DISTINCT statement is used to return only distinct (different) values.
- Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.
- SELECT DISTINCT Syntax

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

Customers Table

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 Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence	12, rue des Bouchers	Marseille	13008	France

SELECT Example Without DISTINCT

- The following SQL statement selects ALL (including the duplicates) values from the "Country" column in the "Customers" table:

```
SELECT Country FROM Customers;
```

Number of Records: 91

Country
Germany
Mexico
Mexico
UK
Sweden
Germany
France
Spain

SELECT DISTINCT Examples

- Now, let us use the DISTINCT keyword with the above SELECT statement and see the result.
- The following SQL statement selects only the DISTINCT values from the "Country" column in the "Customers" table:

```
SELECT DISTINCT Country FROM Customers;
```

Number of Records: 21

Country
Argentina
Austria
Belgium
Brazil
Canada
Denmark
Finland

SELECT DISTINCT Examples

- The following SQL statement lists the number of different (distinct) customer countries:

```
SELECT COUNT(DISTINCT Country) FROM Customers;
```

Number of Records: 1

COUNT(DISTINCT Country)
21

SQL WHERE Clause

- The WHERE clause is used to filter records.
- The WHERE clause is used to extract only those records that fulfill a specified condition.
- WHERE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Note: The WHERE clause is not only used in SELECT statement, it is also used in UPDATE, DELETE statement, etc.!

Customers Table

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 Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence	12, rue des Bouchers	Marseille	13008	France

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';
```

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 single quotes around text values (most database systems will also allow double quotes).
- However, numeric fields should not be enclosed in quotes:

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

Number of Records: 1

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

Operators in The WHERE Clause

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

AND, OR and NOT Operators

- The WHERE clause can be combined with AND, OR, and NOT operators.
- The AND and OR operators are used to filter records based on more than one condition:
  - The AND operator displays a record if all the conditions separated by AND are TRUE.
  - The OR operator displays a record if any of the conditions separated by OR is TRUE.
- The NOT operator displays a record if the condition(s) is NOT TRUE.

AND, OR and NOT Operators

- AND Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

- OR Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

- NOT Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

Customers Table

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 Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence	12, rue des Bouchers	Marseille	13008	France

AND Example

- The following SQL statement selects all fields from "Customers" where country is "Germany" AND city is "Berlin":

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

Number of Records: 1

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

OR Example

- The following SQL statement selects all fields from "Customers" where city is "Berlin" OR "München":

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

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

OR Example

- The following SQL statement selects all fields from "Customers" where country is "Germany" OR "Spain":

```
SELECT * FROM Customers
WHERE Country='Germany' OR Country='Spain';
```

Number of Records: 16

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
17	Drachenblut Delikatessend	Sven Ottlieb	Walserweg 21	Aachen	52066	Germany
22	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	C/ Moralzarzal, 86	Madrid	28034	Spain
25	Frankenversand	Peter Franken	Berliner Platz 43	München	80805	Germany
29	Galería del gastrónomo	Eduardo Saavedra	Rambla de Cataluña, 23	Barcelona	08022	Spain
30	Godos Cocina Típica	José Pedro Freyre	C/ Romero, 33	Sevilla	41101	Spain
39	Königlich Essen	Philip Cramer	Maubelstr. 90	Brandenburg	14776	Germany
44	Lehmanns Marktstand	Renate Messner	Magazinweg 7	Frankfurt a.M.	60528	Germany

NOT Example

- The following SQL statement selects all fields from "Customers" where country is NOT "Germany":

```
SELECT * FROM Customers
WHERE NOT Country='Germany';
```

Number of Records: 80

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
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
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada

Combining AND, OR and NOT Example

- You can also combine the AND, OR and NOT operators.
- The following SQL statement selects all fields from "Customers" where country is "Germany" AND city must be "Berlin" OR "München" (use parenthesis to form complex expressions):

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

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 and NOT Example

- The following SQL statement selects all fields from "Customers" where country is NOT "Germany" and NOT "USA":

```
SELECT * FROM Customers
WHERE NOT Country='Germany' AND NOT
Country='USA';
```

Number of Records: 67

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
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
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada

Searching for Patterns or Ranges of Values

LIKE Operator

- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.
- There are some wildcards often used in conjunction with the LIKE operator
- A wildcard character is used to substitute one or more characters in a string.
  - % - The percent sign represents zero, one, or multiple characters
- LIKE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

Tip: You can also combine any number of conditions using AND or OR operators.

Customers Table

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 Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada

SQL LIKE Examples

- The following SQL statement selects all customers with a CustomerName starting with "a":

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';
```

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 Examples

- The following SQL statement selects all customers with a CustomerName ending with "a":

```
SELECT * FROM Customers
WHERE CustomerName LIKE '%a';
```

Number of Records: 7

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
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
30	Godos Cocina Típica	José Pedro Freyre	C/ Romero, 33	Sevilla	41101	Spain
61	Que Delícia	Bernardo Batista	Rua da Panificadora, 12	Rio de Janeiro	02389-673	Brazil
62	Queen Cozinha	Lúcia Carvalho	Alameda dos Canàrios, 891	São Paulo	05487-020	Brazil

SQL LIKE Examples

- The following SQL statement selects all customers with a CustomerName that have "or" in any position:

```
SELECT * FROM Customers
WHERE CustomerName LIKE '%or%';
```

Number of Records: 11

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
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
36	Hungry Coyote Import Store	Yoshi Latimer	City Center Plaza 516 Main St.	Elgin	97827	USA
40	La corne d'abondance	Daniel Tonini	67, avenue de l'Europe	Versailles	78000	France
43	Lazy K Kountry Store	John Steel	12 Orchestra Terrace	Walla Walla	99362	USA

Wildcard Characters in SQL Server

Symbol	Description	Example
%	Represents zero or more characters	bl% finds bl, black, blue, and blob
_	Represents a single character	h_t finds hot, hat, and hit
[ ]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
^	Represents any character not in the brackets	h[^oa]t finds hit, but not hot and hat
-	Represents a range of characters	c[a-b]t finds cat and cbt

Wildcard Characters in MS Access

Symbol	Description	Example
*	Represents zero or more characters	bl* finds bl, black, blue, and blob
?	Represents a single character	h?t finds hot, hat, and hit
[ ]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
!	Represents any character not in the brackets	h[!oa]t finds hit, but not hot and hat
-	Represents a range of characters	c[a-b]t finds cat and cbt
#	Represents any single numeric character	2#5 finds 205, 215, 225, 235, 245, 255, 265, 275, 285, and 295

IN Operator

- The IN operator allows you to specify multiple values in a WHERE clause.
- The IN operator is a shorthand for multiple OR conditions.
- IN Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

OR

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

Customers Table

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

IN Operator Example

- The following SQL statement selects all customers that are located in "Germany", "France" or "UK":

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

Number of Records: 29

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France

NOT IN Operator Example

- The following SQL statement selects all customers that are NOT located in "Germany", "France" or "UK":

```
SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France', 'UK');
```

Number of Records: 62

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
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada

IN (SELECT..) Example

- The following SQL statement selects all customers that are from the same countries as the suppliers:

```
SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers);
```

Number of Records: 69

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
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 Citeaux	24, place Kléber	Strasbourg	67000	France

BETWEEN Operator

- The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
- The BETWEEN operator is inclusive: begin and end values are included.
- BETWEEN Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Products Table

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
4	Chef Anton's Cajun Seasoning	1	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	1	2	36 boxes	21.35

BETWEEN Example

- The following SQL statement selects all products with a price BETWEEN 10 and 20:

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

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

NOT BETWEEN Example

- To display the products outside the range of the previous example, use NOT BETWEEN:

```
SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;
```

Number of Records: 48

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97
10	Ikura	4	8	12 - 200 ml jars	31
11	Queso Cabrales	5	4	1 kg pkg.	21

BETWEEN with IN Example

- The following SQL statement selects all products with a price BETWEEN 10 and 20. In addition; do not show products with a CategoryID of 1,2, or 3:

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20
AND CategoryID NOT IN (1,2,3);
```

Number of Records: 9

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
31	Gorgonzola Telino	14	4	12 - 100 g pkgs	12.5
36	Inlagd Sill	17	8	24 - 250 g jars	19
40	Boston Crab Meat	19	8	24 - 4 oz tins	18.4
42	Singaporean Hokkien Fried Mee	20	5	32 - 1 kg pkgs.	14
46	Spegesild	21	8	4 - 450 g glasses	12
57	Ravioli Angelo	26	5	24 - 250 g pkgs.	19.5
58	Escargots de Bourgogne	27	8	24 pieces	13.25
73	Röd Kaviar	17	8	24 - 150 g jars	15

BETWEEN Text Values Example

- The following SQL statement selects all products with a ProductName BETWEEN Carnarvon Tigers and Mozzarella di Giovanni:

```
SELECT * FROM Products
WHERE ProductName BETWEEN 'Carnarvon Tigers' AND
'Mozzarella di Giovanni'
ORDER BY ProductName;
```

Number of Records: 37

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
18	Carnarvon Tigers	7	8	16 kg pkg.	62.5
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
39	Chartreuse verte	18	1	750 cc per bottle	18
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35
48	Chocolade	22	3	10 pkgs.	12.75
38	Côte de Blaye	18	1	12 - 75 cl bottles	263.5

NOT BETWEEN Text Values Example

- The following SQL statement selects all products with a ProductName NOT BETWEEN Carnarvon Tigers and Mozzarella di Giovanni:
- Syntax

```
SELECT * FROM Products
WHERE ProductName NOT BETWEEN 'Carnarvon Tigers'
AND 'Mozzarella di Giovanni'
ORDER BY ProductName;
```

Number of Records: 40

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
17	Alice Mutton	7	6	20 - 1 kg tins	39
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
40	Boston Crab Meat	19	8	24 - 4 oz tins	18.4
60	Camembert Pierrot	28	4	15 - 300 g rounds	34
30	Nord-Ost Matjeshering	13	8	10 - 200 g glasses	25.89
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40
25	NuNuCa Nuß-Nougat-Creme	11	3	20 - 450 g glasses	14
77	Original Frankfurter grüne Soße	12	2	12 boxes	13

Sorting the Results

ORDER BY Keyword

- The ORDER BY keyword is used to sort the result-set in ascending or descending order.
- The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.
- ORDER BY Syntax

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Customers Table

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

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;
```

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
76	Suprêmes délices	Pascale Cartrain	Boulevard Tirou, 255	Charleroi	B-6000	Belgium
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

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;
```

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
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
47	LINO-Delicateses	Felipe Izquierdo	Ave. 5 de Mayo Porlamar	I. de Margarita	4980	Venezuela
32	Great Lakes Food Market	Howard Snyder	2732 Baker Blvd.	Eugene	97403	USA
36	Hungry Coyote Import Store	Yoshi Latimer	City Center Plaza 516 Main St.	Elgin	97827	USA
43	Lazy K Kountry Store	John Steel	12 Orchestra Terrace	Walla Walla	99362	USA
45	Let's Stop N Shop	Jaime Yorres	87 Polk St. Suite 5	San Francisco	94117	USA

ORDER BY Several Columns Example

- The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" and the "CustomerName" column. This means that it orders by Country, but if some rows have the same Country, it orders them by CustomerName:

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

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
76	Suprêmes délices	Pascale Cartrain	Boulevard Tirou, 255	Charleroi	B-6000	Belgium
15	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	São Paulo	05432-043	Brazil

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;
```

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
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium
88	Wellington Importadora	Paula Parente	Rua do Mercado, 12	Resende	08737-363	Brazil

Aggregate Functions and Grouping

MIN() and MAX() Functions

- The MIN() function returns the smallest value of the selected column.
- The MAX() function returns the largest value of the selected column.
- Syntax:

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

OR

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

Products Table

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
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97
10	Ikura	4	8	12 - 200 ml jars	31
11	Queso Cabrales	5	4	1 kg pkg.	21
12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38
13	Konbu	6	8	2 kg box	6
14	Tofu	6	7	40 - 100 g pkgs.	23.25
15	Genen Shouyu	6	2	24 - 250 ml bottles	15.5
16	Pavlova	7	3	32 - 500 g boxes	17.45
17	Alice Mutton	7	6	20 - 1 kg tins	39
18	Carnarvon Tigers	7	8	16 kg pkg.	62.5

```
SELECT MIN(Price) AS SmallestPrice
FROM Products;
```

Number of Records: 1

SmallestPrice
2.5

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

Number of Records: 1

MAX(Price)
263.5

COUNT(), AVG() and SUM() Functions

- The COUNT() function returns the number of rows that matches a specified criterion.
- The AVG() function returns the average value of a numeric column.
- The SUM() function returns the total sum of a numeric column.

```
SELECT COUNT(ProductID)
FROM Products;
```

Number of Records: 1

COUNT(ProductID)
77

```
SELECT AVG(Price)
FROM Products;
```

Number of Records: 1

AVG(Price)
28.866363636363637

OrderDetails Table

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40
6	10250	41	10
7	10250	51	35
8	10250	65	15
9	10251	22	6
10	10251	57	15
11	10251	65	20
12	10252	20	40
13	10252	33	25
14	10252	60	40

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

Number of Records: 1

SUM(Quantity)
12743

```
SELECT SUM(Quantity)
FROM OrderDetails
WHERE OrderID = 10250;
```

Number of Records: 1

SUM(Quantity)
60

GROUP BY Statement

- The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".
- The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.
- Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

Number of Records: 21

COUNT(CustomerID)	Country
3	Argentina
2	Austria
2	Belgium
9	Brazil
3	Canada
2	Denmark
2	Finland
11	France
11	Germany
1	Ireland
3	Italy
5	Mexico

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

Number of Records: 21

COUNT(CustomerID)	Country
13	USA
11	Germany
11	France
9	Brazil
7	UK
5	Spain
5	Mexico
4	Venezuela
3	Italy
3	Canada
3	Argentina
2	Switzerland

HAVING Clause

- The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.
- Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
ORDER BY COUNT(CustomerID) DESC;
```

Number of Records: 5

COUNT(CustomerID)	Country
13	USA
11	Germany
11	France
9	Brazil
7	UK

Multiple Table Queries

JOIN

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Orders Table and Customers Table

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

- Selecting all orders together with the customer's name:

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

Number of Records: 196

OrderID	CustomerName	OrderDate
10248	Wilman Kala	1996-07-04
10249	Tradição Hipermercados	1996-07-05
10250	Hanari Carnes	1996-07-08
10251	Victuailles en stock	1996-07-08
10252	Suprêmes délices	1996-07-09
10253	Hanari Carnes	1996-07-10
10254	Chop-suey Chinese	1996-07-11
10255	Richter Supermarkt	1996-07-12
10256	Wellington Importadora	1996-07-15
10257	HILARIÓN-Abastos	1996-07-16
10258	Ernst Handel	1996-07-17
10259	Centro comercial Moctezuma	1996-07-18
10260	Old World Delicatessen	1996-07-19
10261	Que Delícia	1996-07-19
10262	Rattlesnake Canyon Grocery	1996-07-22

- Instead of:

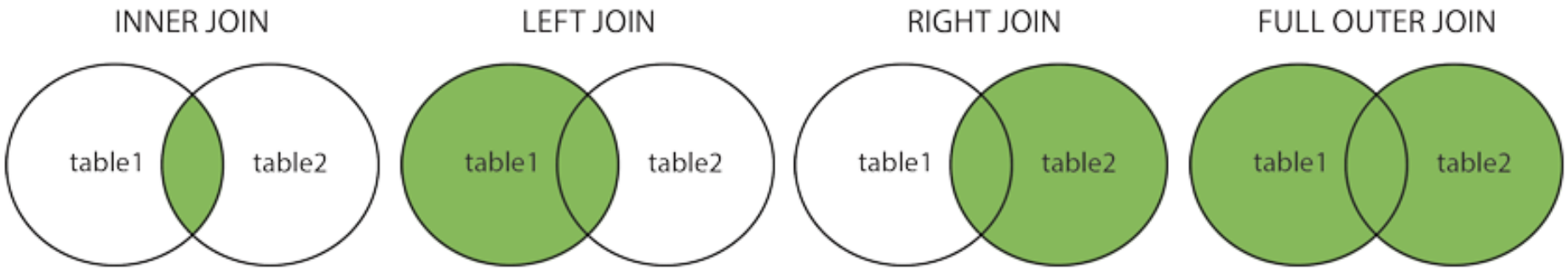
```
SELECT Orders.OrderID,
Customers.CustomerName, Orders.OrderDate
FROM Orders INNER JOIN Customers
ON Orders.CustomerID=Customers.CustomerID;
```

- We can also write it as:

```
SELECT Orders.OrderID,
Customers.CustomerName, Orders.OrderDate
FROM Orders, Customers
WHERE Orders.CustomerID=Customers.CustomerID;
```

Different Types of SQL JOINs

- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table



- Left Outer Join example:

```
SELECT Orders.OrderID,
Customers.CustomerName, Orders.OrderDate
FROM Customers LEFT OUTER JOIN Orders
ON Orders.CustomerID=Customers.CustomerID;
```

Number of Records: 213

OrderID	CustomerName	OrderDate
null	Alfreds Futterkiste	*null*
10308	Ana Trujillo Emparedados y helados	1996-09-18
10365	Antonio Moreno Taquería	1996-11-27
10355	Around the Horn	1996-11-15
10383	Around the Horn	1996-12-16
10278	Berglunds snabbköp	1996-08-12
10280	Berglunds snabbköp	1996-08-14
10384	Berglunds snabbköp	1996-12-16
null	Blauer See Delikatessen	*null*
10265	Blondel père et fils	1996-07-25
10297	Blondel père et fils	1996-09-04
10360	Blondel père et fils	1996-11-22
10436	Blondel père et fils	1997-02-05

UNION

- The UNION operator is used to combine the result-set of two or more SELECT statements.
- Each SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement must also be in the same order
- Syntax

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

Returns the cities (only distinct values) from both the "Customers" and the "Suppliers" table:

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

Number of Records: 94

City
Aachen
Albuquerque
Anchorage
Ann Arbor
Annecy
Barcelona
Barquisimeto
Bend
Bergamo
Berlin
Bern
Boise
Boston
Brandenburg
Bruxelles
Bräcke

Joining Multiple Tables

- Data from multiple tables can be retrieved together by using multiple joins.
- Usually, it is easier to write the query using AND operators instead of multiple JOIN operators

OrderDetails Table

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40
6	10250	41	10
7	10250	51	35
8	10250	65	15
9	10251	22	6
10	10251	57	15
11	10251	65	20
12	10252	20	40
13	10252	33	25
14	10252	60	40

- Retrieve all information needed to prepare an invoice for an order

```
SELECT Orders.OrderID, Customers.CustomerID,
Customers.CustomerName, Products.ProductID,
Products.ProductName, Products.Price,
OrderDetails.Quantity,
Products.Price * OrderDetails.Quantity AS Total
FROM Orders, Customers, OrderDetails, Products
WHERE Orders.CustomerID=Customers.CustomerID
AND Orders.OrderID=OrderDetails.OrderID
AND OrderDetails.ProductID=Products.ProductID;
```

Number of Records: 518

OrderID	CustomerID	CustomerName	ProductID	ProductName	Price	Quantity	Total
10248	90	Wilman Kala	11	Queso Cabrales	21	12	252
10248	90	Wilman Kala	42	Singaporean Hokkien Fried Mee	14	10	140
10248	90	Wilman Kala	72	Mozzarella di Giovanni	34.8	5	174
10249	81	Tradição Hipermercados	14	Tofu	23.25	9	209.25
10249	81	Tradição Hipermercados	51	Manjimup Dried Apples	53	40	2120
10250	34	Hanari Carnes	41	Jack's New England Clam Chowder	9.65	10	96.5
10250	34	Hanari Carnes	51	Manjimup Dried Apples	53	35	1855
10250	34	Hanari Carnes	65	Louisiana Fiery Hot Pepper Sauce	21.05	15	315.75
10251	84	Victuailles en stock	22	Gustaf's Knäckebröd	21	6	126
10251	84	Victuailles en stock	57	Ravioli Angelo	19.5	15	292.5
10251	84	Victuailles en stock	65	Louisiana Fiery Hot Pepper Sauce	21.05	20	421
10252	76	Suprêmes délices	20	Sir Rodney's Marmalade	81	40	3240
10252	76	Suprêmes délices	33	Geitost	2.5	25	62.5
10252	76	Suprêmes délices	60	Camembert Pierrot	34	40	1360

INSERT, UPDATE, and DELETE Statements

INSERT INTO Statement

- The INSERT INTO statement is used to insert new records in a table. INSERT
- It is possible to write the INTO statement in two ways.
- The first way specifies both the column names and the values to be inserted:
- INSERT INTO Syntax

```
INSERT INTO table_name (column1, column2,
column3, ...)
VALUES (value1, value2, value3, ...);
```

INSERT INTO Statement

- If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. The INSERT INTO syntax would be as follows:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

INSERT INTO Example

- The following SQL statement inserts a new record in the "Customers" table:

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

Did you notice that we did not insert any number into the CustomerID field? The CustomerID column is an auto-increment field and will be generated automatically when a new record is inserted into the table.

Insert Data Only in Specified Columns Example

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

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

Customers Table with New Data

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	null	null	Stavanger	null	Norway

UPDATE Statement

- The UPDATE statement is used to modify the existing records in a table.
- UPDATE Syntax

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Note: Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

Customers Table

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

UPDATE Table Example

- The following SQL statement updates the first customer (CustomerID = 1) with a new contact person *and* a new city.

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City=
'Frankfurt'
WHERE CustomerID = 1;
```

Customers Table with Updated Data

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	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

UPDATE Multiple Records Example

- The following SQL statement will update the contactname to "Juan" for all records where country is "Mexico":

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

Customers Table with Updated Data

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	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

Update Warning

- Example:

```
UPDATE Customers
SET ContactName='Juan';
```

Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

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

DELETE Statement

- The DELETE statement is used to delete existing records in a table.
- DELETE Syntax

```
DELETE FROM table_name WHERE condition;
```

Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

Customers Table

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 DELETE Example

- The following SQL statement deletes the customer "Alfreds Futterkiste" from the "Customers" table:
- Syntax

```
DELETE FROM Customers WHERE CustomerName='Alfreds
Futterkiste';
```

Customers Table with Deleted Data

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
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

Delete All Records

- It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

```
DELETE FROM table_name;
```

Delete All Records Example

- The following SQL statement deletes all rows in the "Customers" table, without deleting the table:

```
DELETE FROM Customers;
```

Result:

You have made changes to the database. Rows affected: 91

Table Definition

CREATE TABLE Statement

- The CREATE TABLE statement is used to create a new table in a database.
- Syntax

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

CREATE TABLE Example

- The following example creates a table called "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City:

```
CREATE TABLE Persons (
    PersonID INT,
    LastName VARCHAR(255),
    FirstName VARCHAR(255),
    Address VARCHAR(255),
    City VARCHAR(255)
);
```

PersonID	LastName	FirstName	Address	City

PRIMARY KEY

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

PRIMARY KEY

- Example

```
CREATE TABLE Persons (
    PersonID INT NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255),
    Address VARCHAR(255),
    City VARCHAR(255),
    PRIMARY KEY (PersonID)
);
```

DROP TABLE Statement

- The DROP TABLE statement is used to drop an existing table in a database.
- Syntax

```
DROP TABLE table_name;
```

Note: Be careful before dropping a table. Deleting a table will result in loss of complete information stored in the table!

Summary

- SQL is very useful for data analytics and is not difficult to learn.
- Materials from this set of slides are extracted from https://www.w3schools.com/sql/
- Try writing and testing queries yourself

- More SQL and database topics will be covered in the course MSBA7024 Database Design and Management

Questions and Comments