Big Data Analysis

Introduction to SQL - 02

- Teaching Assistant
Yash More

INDEX :

- SQL SELECT Statement
    - SQL SELECT DISTINCT Statement
- SQL WHERE Clause
    - Operators in The WHERE Clause
- SQL AND, OR and NOT Operators
    - Combining AND, OR and NOT
- SQL INSERT INTO Statement
- What is a NULL Value?
- The SQL UPDATE Statement
- The SQL DELETE Statement
    - Delete All Records
- SQL MIN() and MAX() Functions
- The SQL COUNT(), AVG() and SUM() Functions

SQL SELECT Statement :

The SQL 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;
```

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;
```

Demo Database :

Below is a selection from the "Customers" table in the Northwind sample database:

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

SELECT Column Example :

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

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

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

SQL SELECT DISTINCT Statement :

- The 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;
```

SQL SELECT DISTINCT Statement :

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

Number of Records: 21

Country
Germany
Mexico
UK
Sweden
France
Spain
Canada
Argentina

SQL SELECT DISTINCT Statement :

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.
- It 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 statements, it is also used in UPDATE, DELETE, etc.!

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

SQL 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.

SQL AND, OR and NOT Operators :

AND Syntax :

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

SQL AND, OR and NOT Operators :

AND Syntax :

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

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

SQL AND, OR and NOT Operators :

OR Syntax :

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

SQL AND, OR and NOT Operators :

OR Syntax :

```
SELECT * FROM Customers

WHERE City='Berlin' OR City='München';
```

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

SQL AND, OR and NOT Operators :

NOT Syntax :

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

SQL AND, OR and NOT Operators :

NOT Syntax :

```
SELECT * FROM Customers

WHERE NOT Country='Germany';
```

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

Combining AND, OR and NOT :

- 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');
```

Combining AND, OR and NOT :

- 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):

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

SQL INSERT INTO Statement :

The `INSERT INTO` statement is used to insert new records in a table.

INSERT INTO Syntax

- It is possible to write the `INSERT INTO` statement in two ways:

    1. Specify both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);
```

SQL INSERT INTO Statement :

The `INSERT INTO` statement is used to insert new records in a table.

INSERT INTO Syntax

- It is possible to write the `INSERT INTO` statement in two ways:

2. 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. Here, the `INSERT INTO` syntax would be as follows:

```
INSERT INTO table_name

VALUES (value1, value2, value3, ...);
```

SQL INSERT INTO Statement :

Example :

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

SQL INSERT INTO Statement :

Example :

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

```
You have made changes to the database. Rows affected: 1
```

SQL INSERT INTO Statement :

Insert Data Only in Specified Columns :

- 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):

Example :

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

SQL INSERT INTO Statement :

Example :

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

What is a NULL Value?

- A field with a NULL value is a field with no value.
- If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

Note: A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

The SQL 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!

The SQL UPDATE Statement :

UPDATE Syntax :

Before :

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

UPDATE Syntax :

```
UPDATE Customers

SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'

WHERE CustomerID = 1;
```

Before :

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

After :

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany

The SQL 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!

The SQL DELETE Statement :

Example :

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

Before :

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

The SQL DELETE Statement :

Example :

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

After :

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

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;
```

Example :

```
DELETE FROM Customers;
```

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;
```

Example :

```
DELETE FROM Customers;
```

```
You have made changes to the database. Rows affected: 93
```

SQL 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.

MIN() Syntax :

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

SQL 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.

MIN() :

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

SmallestPrice
2.5

SQL 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.

MAX() Syntax :

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

SQL 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.

MAX() :

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

LargestPrice
263.5

The SQL COUNT(), AVG() and SUM() Functions :

The COUNT() function returns the number of rows that matches a specified criterion.

COUNT() Syntax :

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

The SQL COUNT(), AVG() and SUM() Functions :

The AVG() function returns the average value of a numeric column.

AVG() Syntax :

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

The SQL COUNT(), AVG() and SUM() Functions :

The SUM() function returns the total sum of a numeric column.

SUM() Syntax :

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

Thank You!!!