SQL

- **What is SQL ?**

SQL (Structured Query Language) is a programming language designed for managing data in relational database. SQL has a variety of functions that allow its users to read, manipulate, and change data. Though SQL is commonly used by engineers in software development, it's also popular with data analysts for a few reason:

  - It's semantically easy to understand and learn.
  - Because it can be used to access large amounts of data directly where it's stored, analysts don't have to copy data into other applications.
  - Compared to spereadsheet tools, data analysis done in SQL is easy to audit and replicate. For analysts, this means no more looking for the cell with the typo in the formula.

- **SELECT * Example**

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

Example - `SELECT * FROM Sales;`

- Select columns wise

Example-
```
SELECT year,
       month,
       west
FROM Sales
```

- Rename Columns

Example-
```
SELECT west AS "west Region"
FROM Sales
```

- LIMIT Clause

The LIMIT clause is used to specify the number of records to return

Example-
```
SELECT *
FROM Sales
LIMIT 100
```

- WHERE Clause

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

Example-
```
SELECT *
FROM Sales
WHERE Country = "Canada";
```

- **Comparison operators on numerical data**

The most basic way to filter data is using comparison operators. The easiest way to understand them is to start by looking at a list of them:

Equal to	=
Not equal to	<> or !=
Greater than	>
Less than	<
Greater than or equal to	>=
Less than or equal to	<=

Example-

```
SELECT *
  FROM Sales
WHERE city = "kolkata";
```

```
SELECT *
  FROM Sales
WHERE city != "Kolkata";
```

```
SELECT *
  FROM Sales
WHERE Month > "January";
```

```
SELECT *
  FROM Sales
WHERE Sale_amount < 50000
```

- ## Arithmetic in SQL

You can perform arithmetic in SQL using the same operators you would in Excel: +, -, *, /. However, in SQL you can only perform arithmetic across columns on values in a given row. To clarify, you can only add values in multiple columns from the same row together using + — if you want to add values across multiple rows, you'll need to use aggregate functions.

```
Example- SELECT year,
                month,
                west,
                south,
                west + south AS south_plus_west
         FROM sales;
```

```
Example-
    SELECT year,
           month,
           west,
           south,
           west + south - 4 * year AS new_column
    FROM sales;
```

```
Example- SELECT year,
                month,
                west,
                south,
                (west + south)/2 AS south_west_avg
         FROM sales;
```

- ## CREATE TABLE

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

Example-
```
CREATE TABLE person(
    PersonID int,
    LastName varchar(255),
    FristName varchar(255),
    Address varchar(255),
    City varchar(255),
);
```

- ## INSERT INTO

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

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

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

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.

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

- **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.

- **How to 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.

- **The IS NULL Operator**

The IS NULL operator is used to test for empty values (NULL values).

Example-
```
SELECT customerName, contactName, Address
FROM Sales
WHERE Address IS NULL ;
```

- **The IS NOT NULL Operator**

The IS NOT NULL operator is used to test for non-empty values (NOT NULL values).

Example -
```
SELECT customerName, ContactName, Address
FROM Sales
WHERE Address IS NOT NULL ;
```

- ## UPDATE Statement

The UPDATE statement is used to modify the existing records in a tabel.

Example-

```
UPDATE Sales
SET contactName = "Alan", city = "Goa"
WHERE customer ID = 1;
```

- ## UPDATE Multiple Records

It is the WHERE clause that determines how many records will be updated.

Example-

```
UPDATE Sales
SET PostalCode = 00000
WHERE Country = "India";
```

Notes:-

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

- ## DELETE Statement

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

Example- `DELETE FROM Sales WHERE CustomerName = "Bob";`

- **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:

Example -

```
DELETE FROM table_name ;
```

- **Aliases**

Aliases are used to give a table, or a column in a table, a tempoary name.
Aliases are often used to make column names more readable.
An alias only exists for the duration of that query.
An alias is created with the AS keyword.

- **Alias Column Example**

```
SELECT column_name AS alias_name
FROM table_name ;
```

- **Alias Table Example**

```
SELECT column_name (s)
FROM table_name AS alias_name ;
```

SQL Logical operators

Logical operators allow you to use multiple comparison operators in one query.
Each logical operator is a special snowflake, so we'll go through them individually in the following lessons.

- LIKE allows you to match similar values, instead of exact values.
- IN allows you to specify a list of values you'd like to include.
- BETWEEN allows you to select only rows within a certain range.
- IS NULL allows you to select rows that contain no data in a given column.
- AND allows you to select only rows that satisfy two conditions.
- OR allows you to select rows that satisfy either of two conditions.
- NOT allows you to select rows that do not match a certain condition.

LIKE Operator

```
SELECT *
  FROM Sales
WHERE "group" LIKE 'New%';
```

- IN Operator

```
SELECT*
  FROM Songs
WHERE artist IN ('Taylor swift', 'Usher');
```

- BETWEEN Operator

```
SELECT*
  FROM Songs
WHERE year_rank BETWEEN 5 AND 10;
```

- AND Operator

```
SELECT*
  FROM Songs
WHERE year = 2012 AND year_rank <= 10;
```

- OR Operator

```
SELECT*
  FROM Songs
WHERE year_rank = 5 OR artist = "Sonu";
```

- NOT Operator

```
SELECT*
  FROM Sales
WHERE NOT Country = "Japan";
```

- Combining AND, OR and NOT

```
SELECT * FROM Sales
WHERE country = 'Japan' AND (city = 'GOA' OR city = 'Puri');
```

- ORDER BY

```
SELECT *
  FROM Sales
ORDER BY country, CustomerName ;

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

- Using Comments (How to use comments)

```
SELECT*  -- This is select command
  FROM Sales
WHERE year = 2020 ;
```

```
/* Here's a comment so long and descriptive that
it could only fit on multiple lines. Fortunately,
it, too, will not affect how this code runs. */
SELECT *
  FROM Sales
WHERE year = 2015 ;
```