SQL AGGREGATE FUNCTIONS

- SQL Aggregate functions are functions where the values of multiple rows are grouped as input on certain criteria to form a single value result of more significant meaning.
- It is used to summarize data, by combining multiple values to form a single result.
- SQL Aggregate functions are mostly used with the GROUP BY clause of the SELECT statement.
- Most Common Aggregate Functions:
  - Count()
  - Sum()
  - Avg()
  - Min()
  - Max()

1 . COUNT() Function :

- The COUNT() function provides the number of rows that matches a specified condition.
- Syntax –

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

2. AVG() Function :

- The AVG() function provides the average value of a numeric column.
- Syntax –

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

3. SUM() Function :

- The SUM() function provides the total sum of a numeric column.
- Syntax –

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

4. MIN() and MAX() Functions

- The SQL MIN() and MAX() functions are the aggregate functions in SQL, that operate on group of data and return one output.
- The SQL Min() function returns the minimum value of the selected columns, and the SQL MAX() function is used to return the maximum value of the selected columns.

- **The MIN()** function returns the smallest value in the column.
- The MIN() function can be used with the DISTINCT keyword to return the minimum value among unique values in a column.
- Syntax

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

- **The MAX()** function returns the largest value in the column.
- The MAX() function in SQL can be used in combination with other SQL clauses and functions, such as GROUP BY, HAVING, and subqueries which can be useful for data analysis and reporting
- Syntax

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

- In this MIN() function example, we will fetch the details of customer with minimum age:

```
SELECT MIN(Age) FROM Customer ;
```

- In this MAX() function example, we will find customer details with the highest Age.

```
SELECT Max(Age) FROM Customer;
```

SQL DATA CONSTRAINTS

- SQL data constraints are rules applied to table columns to ensure the integrity, accuracy, and reliability of the data within the database.
- Constraints enforce specific rules at the table level, preventing invalid data from being inserted or updated in the database.
- Types of SQL Data Constraints
  - NOT NULL Constraint
  - UNIQUE Constraint
  - PRIMARY KEY Constraint
  - FOREIGN KEY Constraint
  - CHECK Constraint
  - DEFAULT Constraint
  - INDEX Constraint (not a traditional constraint, but related to performance and uniqueness)

- The SQL NOT NULL forces particular values or records should not to hold a null value.
- NOT NULL constraints make sure that a column does not hold null values, or in other words, NOT NULL constraints make sure that you cannot insert a new record or update a record without entering a value to the specified column.
- For example, in the query below the “EMPID” will not accept NULL values when the EMPLOYEES table is created because NOT NULL constraints are used with these columns.

```
CREATE TABLE Emp(
    EmpID INT NOT NULL PRIMARY KEY,
   Name VARCHAR (50),
    Country VARCHAR(50),
    Age int(2),
  Salary int(10));
```

UNIQUE Constraint

- The UNIQUE constraint ensures that all values in a column are distinct. This constraint prevents duplicate values in a column.
- CREATE TABLE Employees (
-     EmployeeID INT NOT NULL UNIQUE,
-     Email VARCHAR(100) UNIQUE,
-     FirstName VARCHAR(50) NOT NULL,
-     LastName VARCHAR(50) NOT NULL
-);
- Here, both EmployeeID and Email columns must have unique values.

PRIMARY KEY Constraint

- PRIMARY KEY in SQL is a column (or group of columns) that uniquely identifies the records in that table.
- A primary key must contain unique values and can not have any NULL value.
- There can only be one primary key in a table, but that primary key can consist of one or more columns.
- When there are two or more columns in the primary key it is called a **composite key.**
- A primary key automatically has a UNIQUE constraint defined on it, and it ensures that there are no duplicate or NULL values in that column.

```
CREATE TABLE Employees (
    EmployeeID INT NOT NULL,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    PRIMARY KEY (EmployeeID)
);
```

- In this example, EmployeeID is the primary key, which uniquely identifies each record.

FOREIGN KEY Constraint

- A foreign key is a column or a combination of columns in a table that establishes a link between two tables in a relational database. It refers to the primary key in another table, creating a relationship between them.
- SQL Foreign Key constraint establishes a relationship between two tables by requiring values in one table’s column to match values in another table’s primary key column.
- A foreign key is created in the CREATE TABLE or ALTER TABLE statement. The foreign key in a table should match the primary key in the referenced table for every row. This is called Referential Integrity. Foreign key ensures referential integrity.

```
CREATE TABLE Departments (
    DepartmentID INT NOT NULL PRIMARY KEY,
    DepartmentName VARCHAR(50) NOT NULL
);
```

-

```
CREATE TABLE Employees (
    EmployeeID INT NOT NULL PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    DepartmentID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);
```

- Here, DepartmentID in the Employees table is a foreign key that references the DepartmentID column in the Departments table.

CHECK Constraint

- The CHECK constraint ensures that all values in a column satisfy a specific condition.

```
CREATE TABLE Employees (
    EmployeeID INT NOT NULL PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Age INT,
    CHECK (Age >= 18)
);
```

- In this example, the CHECK constraint ensures that the Age column only contains values greater than or equal to 18.

DEFAULT Constraint

- The DEFAULT constraint provides a default value for a column when no value is specified.

```
CREATE TABLE Employees (
    EmployeeID INT NOT NULL PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    HireDate DATE DEFAULT CURRENT_DATE
);
```

- Here, the HireDate column will default to the current date if no value is provided.