Aggregate Functions

- MIN() - returns the smallest value within the selected column
- MAX() - returns the largest value within the selected column
- COUNT() - returns the number of rows in a set
- SUM() - returns the total sum of a numerical column
- AVG() - returns the average value of a numerical column

Syntax

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

SELECT MAX(column_name)
FROM table_name
WHERE condition;

SELECT AVG(column_name)
FROM table_name
WHERE condition;

SELECT SUM(column_name)
FROM table_name
WHERE condition;

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

Example

```
CREATE TABLE Employee (
  Id INT PRIMARY KEY,
  Name CHAR(1),
  Salary DECIMAL(10,2)
);

INSERT INTO Employee (Id, Name, Salary)
VALUES (1, 'A', 802),
      (2, 'B', 403),
      (3, 'C', 604),
      (4, 'D', 705),
      (5, 'E', 606),
      (6, 'F', NULL);

--Count the number of employees
SELECT COUNT(*) AS TotalEmployees FROM Employee;
```

```
-- Calculate the total salary
SELECT SUM(Salary) AS TotalSalary FROM Employee;

-- Find the average salary
SELECT AVG(Salary) AS AverageSalary FROM Employee;

-- Get the highest salary
SELECT MAX(Salary) AS HighestSalary FROM Employee;

-- Determine the lowest salary
SELECT MIN(Salary) AS LowestSalary FROM Employee;
```

Output

TotalEmployees
6

TotalSalary
3120

AverageSalary
624

HighestSalary
802

LowestSalary
403

Using Aggregate Functions with GROUP BY

Syntax
SELECT column_name, aggregate_function(column_name)
FROM table_name
GROUP BY column_name;

Example
SELECT Name, SUM(Salary) AS TotalSalary
FROM Employee
GROUP BY Name;

Output:

Name	TotalSalary
A	802
B	403
C	604
D	705
E	606
F	–

Using HAVING with Aggregate Functions
Syntax
SELECT column_name, aggregate_function(column_name)
FROM table_name
GROUP BY column_name
HAVING condition;

Example
SELECT Name, SUM(Salary) AS TotalSalary
FROM Employee
GROUP BY Name
HAVING SUM(Salary) > 600;

Output:

Name	TotalSalary
A	802
C	604
D	705
E	606

SQL LIKE Operator
LIKE operator is case-insensitive

Wildcard Characters with the SQL LIKE Operator
Wildcards are used with the LIKE operator to search for specific patterns in strings. Wildcard characters substitute one or more characters in the string. There are four wildcard characters in SQL:

1. % (Percent): Represents zero or more characters.
2. _ (Underscore): Represents a single character.
3. [] (Square Brackets): Represents any single character within brackets.
4. (Hyphen): Specify a range of characters inside brackets.

Examples of Wildcards
The below table shows some examples on how wild card can be written and what do they mean:

Pattern	Meaning
‘a%’	Match strings that start with ‘a’
‘%a’	Match strings with end with ‘a’
‘a%t’	Match strings that contain the start with ‘a’ and end with ‘t’.
‘%wow%’	Match strings that contain the substring ‘wow’ in them at any position.
‘_wow%’	Match strings that contain the substring ‘wow’ in them at the second position.
‘_a%’	Match strings that contain ‘a’ at the second position.

Pattern	Meaning
‘a_ _%’	Match strings that start with ‘a and contain at least 2 more characters.

Example

```
CREATE TABLE Supplier (
   SupplierID CHAR(2) PRIMARY KEY,
   Name VARCHAR(50),
   Address VARCHAR(100)
);

INSERT INTO Supplier (SupplierID, Name, Address)
VALUES
   ('S1', 'Paragon Suppliers', '21-3, Okhla, Delhi'),
   ('S2', 'Mango Nation', '21, Faridabad, Haryana'),
   ('S3', 'Canadian Biz', '6/7, Okhla Phase II, Delhi'),
   ('S4', 'Caravan Traders', '2-A, Pitampura, Delhi'),
   ('S5', 'Harish and Sons', 'Gurgaon, NCR'),
   ('S6', 'Om Suppliers', '2/1, Faridabad, Haryana');
```

SupplierID	Name	Address
S1	Paragon Suppliers	21-3, Okhla, Delhi
S2	Mango Nation	21, Faridabad, Haryana
S3	Canadian Biz	6/7, Okhla Phase II, Delhi
S4	Caravan Traders	2-A, Pitampura, Delhi
S5	Harish and Sons	Gurgaon, NCR
S6	Om Suppliers	2/1, Faridabad, Haryana

Example 1: Match Names Starting with ‘Ca’

```
SELECT SupplierID, Name, Address
FROM Supplier
WHERE Name LIKE 'Ca%';
```

Output:

S3	Canadian Biz	6/7, Okhla Phase II, Delhi
S4	Caravan Traders	2-A, Pitampura, Delhi

Example 2: Match Addresses Containing ‘Okhla’

```
SELECT *
FROM Supplier
WHERE Address LIKE '%Okhla%';
```

Output:

S1	Paragon Suppliers	21-3, Okhla, Delhi
S3	Canadian Biz	6/7, Okhla Phase II, Delhi

Example 3: Match Names Where ‘ango’ Appears in the Second Position

```
SELECT SupplierID, Name, Address
FROM Supplier
WHERE Name LIKE '_ango%';
```

Output:

S2	Mango Nation	21, Faridabad, Haryana

Example 4: Using LIKE with AND for Complex Conditions

```
SELECT SupplierID, Name, Address
FROM Supplier
WHERE Address LIKE '%Delhi%' AND Name LIKE 'C%';
```

Output:

SupplierID	Name	Address
S3	Canadian Biz	6/7, Okhla Phase II, Delhi
S4	Caravan Traders	2-A, Pitampura, Delhi

Example 5: Using NOT LIKE for Exclusion

```
SELECT SupplierID, Name, Address
FROM Supplier
WHERE Name NOT LIKE '%Mango%';
```

Output:

SupplierID	Name	Address
S1	Paragon Suppliers	21-3, Okhla, Delhi
S3	Canadian Biz	6/7, Okhla Phase II, Delhi
S4	Caravan Traders	2-A, Pitampura, Delhi
S5	Harish and Sons	Gurgaon, NCR
S6	Om Suppliers	2/1, Faridabad, Haryana