Aggregate Functions

```
CREATE TABLE products (
    productCode VARCHAR(15) PRIMARY KEY,
    productName VARCHAR(70),
    productLine VARCHAR(50),
    productScale VARCHAR(10),
    productVendor VARCHAR(50),
    productDescription VARCHAR(200),
    quantityInStock SMALLINT,
    buyPrice DOUBLE,
    MSRP DOUBLE
);

INSERT INTO products (productCode, productName,
productLine, productScale, productVendor,
productDescription, quantityInStock, buyPrice,
MSRP) VALUES
('B20_0001', 'Royal Enfield Classic 350',
'Motorcycles', '1:10', 'Indus Diecast', 'Replica of the
```

popular Royal Enfield Classic 350 with detailed features.', 1200, 462000.00, 546000.00),

('B20_0002', 'Tata Nano', 'Compact Cars', '1:18', 'Auto India', 'High quality scale model of Tata Nano, the iconic Indian car.', 850, 252000.00, 294000.00),

('B20_0003', 'Mahindra Thar', 'SUVs', '1:18', 'Mahindra Models', 'Detailed model of the rugged Mahindra Thar, perfect for collectors.', 500, 630000.00, 714000.00),

('B20_0004', 'Hero Splendor', 'Motorcycles', '1:12', 'Hero Models', 'Accurate replica of the Hero Splendor, India’s favorite commuter bike.', 3000, 100800.00, 126000.00),

('B20_0005', 'Maruti Suzuki Swift', 'Hatchbacks', '1:18', 'Maruti Models', 'Detailed model of Maruti Suzuki Swift, one of the best-selling cars in India.', 600, 336000.00, 420000.00);

SELECT AVG(buyPrice) AS AveragePrice FROM Products;

SELECT COUNT(ProductName) AS NumberOfProducts FROM Products;

SELECT COUNT(*) AS NumberOfProducts FROM Products;

SELECT MIN(buyPrice) AS SmallestPrice FROM Products;

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

SELECT SUM(quantityinstock) AS TotalStock FROM Products;
```

Date Functions

```
SELECT CURRENT_DATE();

SELECT CURRENT_TIME();

SELECT CURRENT_TIMESTAMP();


SELECT CURRENT_DATE+ INTERVAL 10 DAY;

SELECT CURRENT_DATE - INTERVAL 10 DAY;


SELECT ADDDATE("2024-07-12", INTERVAL 10 DAY);

SELECT DATEDIFF("2024-07-25", "2024-07-15");

SELECT TIME("19:30:10");

SELECT EXTRACT(MONTH FROM "2017-06-15");

SELECT EXTRACT(YEAR FROM "2017-06-15");

SELECT EXTRACT(DAY FROM "2017-06-15");
```

String Functions

```
SELECT CONCAT("SQL ", "Tutorial ", "is ", "fun!") AS
ConcatenatedString;

SELECT CONVERT("2017-08-29", DATE);

SELECT LOWER("SQL Tutorial is FUN!") AS LowercaseText;

SELECT SUBSTRING("SQL Tutorial", 5, 3) AS ExtractString;

SELECT TRIM("SQL Tutorial    ") AS RightTrimmedString;

SELECT UPPER("SQL Tutorial is FUN!") AS UppercaseText;
```

Numeric Functions

```
SELECT ABS(243.5);

SELECT CEIL(25.75);

SELECT FLOOR(25.75);

SELECT MOD(18, 4);

SELECT PI();

SELECT POWER(4, 2);

SELECT ROUND(135.375, 2);

SELECT SQRT(64);

SELECT TRUNCATE(135.375, 2);
```

Creating groups

Creating groups in SQL typically involves using the GROUP BY clause, which groups rows that have the same values in specified columns into

summary rows. Here's a basic example of how to use GROUP BY in MySQL to generate groups:

Example Schema

Let's assume we have a table named employees with the following structure:

```
CREATE TABLE employees (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50),
    department VARCHAR(50),
    salary DECIMAL(10, 2)
);
```

Sample Data

```
INSERT INTO employees (name, department, salary) VALUES

('Ravi Kumar', 'Engineering', 75000),

('Anjali Sharma', 'Marketing', 65000),

('Amit Verma', 'Engineering', 80000),

('Neha Gupta', 'Marketing', 70000),

('Vijay Singh', 'Engineering', 72000);
```

Grouping Data

Now, let's say we want to group employees by their department and calculate the average salary for each department. We can do this using the GROUP BY clause along with aggregate functions such as AVG:

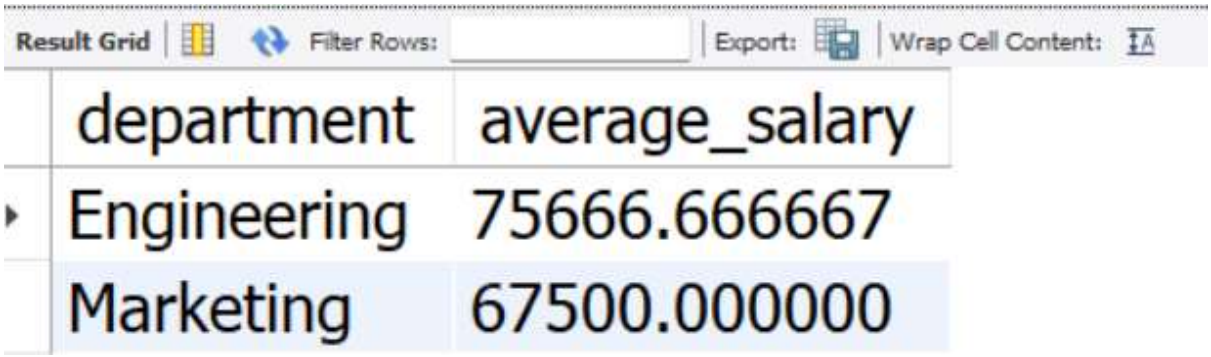
```
SELECT department, AVG(salary) AS average_salary
FROM employees
GROUP BY department;
```

Explanation

- SELECT department, AVG(salary) AS average_salary: Selects the department column and calculates the average salary for each department.
- FROM employees: Specifies the table to select data from.
- GROUP BY department: Groups the result set by the department column.

Result

The result of this query will be:



More Complex Example

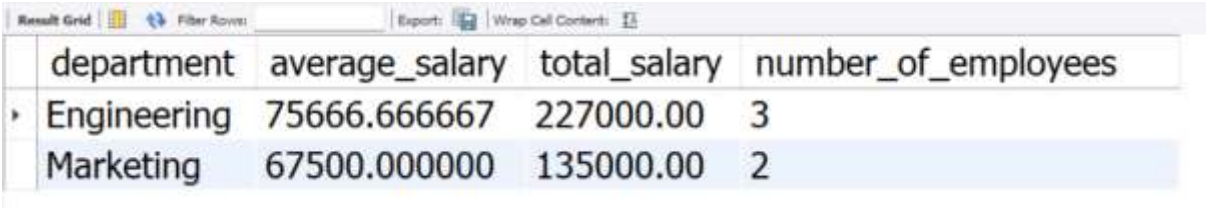
If you want to include more complex calculations or additional columns, you can extend the query. For example, to get the total salary and the number of employees in each department, you can do:

```
SELECT department, AVG(salary) AS average_salary, SUM(salary) AS
total_salary, COUNT(*) AS number_of_employees
FROM employees
GROUP BY department;
```

This query will provide a summary of each department including the average salary, total salary, and number of employees in each department.

Result

The result will look like this:



department	average_salary	total_salary	number_of_employees
Engineering	75666.666667	227000.00	3
Marketing	67500.000000	135000.00	2