SQL | DDL, DQL, DML, DCL and TCL Commands

1. Create a Table:

Command Prompt - mysql -u root -p

```
mysql> create database empdb;
Query OK, 1 row affected (0.03 sec)

mysql> use empdb;
Database changed
mysql> select database();
+------------+
| database() |
+------------+
| empdb      |
+------------+
1 row in set (0.00 sec)

mysql>
```

2. Storing Data in a Table:

Command Prompt - mysql -u root -p

```
mysql> CREATE TABLE Employees (
    ->     EmployeeID INT PRIMARY KEY,
    ->     FirstName VARCHAR(50),
    ->     LastName VARCHAR(50),
    ->     Age INT,
    ->     Department VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.07 sec)
```

3. Updating Data in a Table:

```
mysql> -- Update the age of employee with EmployeeID 2
mysql> UPDATE Employees
    -> SET Age = 26
    -> WHERE EmployeeID = 2;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

4. Deleting Data from a Table:

```
mysql> -- Delete the employee with EmployeeID 3
mysql> DELETE FROM Employees
    -> WHERE EmployeeID = 3;
Query OK, 1 row affected (0.03 sec)
```

5. Retrieving Specific Attributes:

```
mysql> -- Retrieve EmployeeID and FirstName from the 'Employees' table
mysql> SELECT EmployeeID, FirstName
    -> FROM Employees;
+------------+-----------+
| EmployeeID | FirstName |
+------------+-----------+
|          1 | John      |
|          2 | Jane      |
+------------+-----------+
2 rows in set (0.00 sec)
```

6. Retrieving Selected Rows:

```
mysql> -- Retrieve all columns for employees in the 'IT' department
mysql> SELECT *
    -> FROM Employees
    -> WHERE Department = 'IT';
+------------+-----------+----------+------+------------+
| EmployeeID | FirstName | LastName | Age  | Department |
+------------+-----------+----------+------+------------+
|          2 | Jane      | Smith    |   26 | IT         |
+------------+-----------+----------+------+------------+
```

7. Filtering Data: WHERE Clauses:

```
mysql> -- Retrieve employees older than 28
mysql> SELECT *
    -> FROM Employees
    -> WHERE Age > 28;
+------------+-----------+----------+------+------------+
| EmployeeID | FirstName | LastName | Age  | Department |
+------------+-----------+----------+------+------------+
|          1 | John      | Doe      |   30 | HR         |
+------------+-----------+----------+------+------------+
1 row in set (0.03 sec)
```

8. Column & Table Aliases:

```
mysql> -- Retrieve EmployeeID as ID and FirstName as First_Name
mysql> SELECT EmployeeID AS ID, FirstName AS First_Name
    -> FROM Employees;
+----+------------+
| ID | First_Name |
+----+------------+
|  1 | John       |
|  2 | Jane       |
+----+------------+
2 rows in set (0.00 sec)
```

9. Using LIKE:

Command Prompt - mysql -u root -p

```
mysql> SELECT *
    -> FROM Employees
    -> WHERE LastName LIKE 'D%';
+------------+-----------+----------+------+------------+
| EmployeeID | FirstName | LastName | Age  | Department |
+------------+-----------+----------+------+------------+
|          1 | John      | Doe      |   30 | HR         |
+------------+-----------+----------+------+------------+
1 row in set (0.03 sec)
```

10. Using ALL:

```
mysql> -- Retrieve employees with a salary greater than SOME employees in the Finance department
mysql> SELECT *
    -> FROM Employees
    -> WHERE Salary > SOME (SELECT Salary FROM Salaries WHERE Department = 'Finance');
+------------+-----------+----------+------+------------+----------+
| EmployeeID | FirstName | LastName | Age  | Department | Salary   |
+------------+-----------+----------+------+------------+----------+
|          3 | Bob       | Johnson  |   35 | Finance    | 75000.00 |
+------------+-----------+----------+------+------------+----------+
1 row in set (0.00 sec)
```

11. Using SOME:

Command Prompt - mysql -u root -p

```
mysql> -- Retrieve employees with a salary greater than ALL employees in the IT department
mysql> SELECT *
    -> FROM Employees
    -> WHERE Salary > ALL (SELECT Salary FROM Salaries WHERE Department = 'IT');
+------------+-----------+----------+------+------------+----------+
| EmployeeID | FirstName | LastName | Age  | Department | Salary   |
+------------+-----------+----------+------+------------+----------+
|          1 | John      | Doe      |   30 | HR         | 50000.00 |
|          3 | Bob       | Johnson  |   35 | Finance    | 75000.00 |
+------------+-----------+----------+------+------------+----------+
2 rows in set (0.00 sec)
```

12. Using ANY:

```
mysql> -- Retrieve employees with a salary greater than ANY employee in the HR department
mysql> SELECT *
    -> FROM Employees
    -> WHERE Salary > ANY (SELECT Salary FROM Salaries WHERE Department = 'HR');
Empty set (0.00 sec)
```

13. Using EXISTS:

```
mysql> -- Retrieve employees who have a salary record in the 'Salaries' table
mysql> SELECT *
    -> FROM Employees
    -> WHERE EXISTS (SELECT 1 FROM Salaries WHERE Salaries.EmployeeID = Employees.EmployeeID);
+------------+-----------+----------+------+------------+----------+
| EmployeeID | FirstName | LastName | Age  | Department | Salary   |
+------------+-----------+----------+------+------------+----------+
|          1 | John      | Doe      |   30 | HR         | 50000.00 |
|          2 | Jane      | Smith    |   25 | IT         | 60000.00 |
|          3 | Bob       | Johnson  |   35 | Finance    | 75000.00 |
+------------+-----------+----------+------+------------+----------+
3 rows in set (0.03 sec)
```

14. UNION (Combine rows from two tables, removing duplicates):

```
mysql> -- Combine distinct rows from 'Employees' and 'NewEmployees'
mysql> SELECT * FROM Employees
    -> UNION
    -> SELECT * FROM NewEmployees;
+------------+-----------+----------+------+------------+----------+
| EmployeeID | FirstName | LastName | Age  | Department | Salary   |
+------------+-----------+----------+------+------------+----------+
|          1 | John      | Doe      |   30 | HR         | 50000.00 |
|          2 | Jane      | Smith    |   25 | IT         | 60000.00 |
|          3 | Bob       | Johnson  |   35 | Finance    | 75000.00 |
|          4 | Alice     | Johnson  |   28 | IT         | 58000.00 |
|          5 | Charlie   | Brown    |   32 | Finance    | 70000.00 |
|          6 | Eva       | Williams |   27 | HR         | 52000.00 |
+------------+-----------+----------+------+------------+----------+
6 rows in set (0.05 sec)
```

15. INTERSECT (Retrieve common rows between two tables):

```
mysql> -- Retrieve common rows between 'Employees' and 'NewEmployees'
mysql> SELECT * FROM Employees
    -> INTERSECT
    -> SELECT * FROM NewEmployees;
Empty set (0.00 sec)
```

16. EXCEPT (Retrieve rows from the first table that are not in the second table):

```
mysql> -- Retrieve rows from 'Employees' that are not in 'NewEmployees'
mysql> SELECT * FROM Employees
    -> EXCEPT
    -> SELECT * FROM NewEmployees;
+------------+-----------+----------+------+------------+----------+
| EmployeeID | FirstName | LastName | Age  | Department | Salary   |
+------------+-----------+----------+------+------------+----------+
|          1 | John      | Doe      |   30 | HR         | 50000.00 |
|          2 | Jane      | Smith    |   25 | IT         | 60000.00 |
|          3 | Bob       | Johnson  |   35 | Finance    | 75000.00 |
+------------+-----------+----------+------+------------+----------+
3 rows in set (0.00 sec)
```

17. Using COMMIT

```
mysql> -- Start a transaction
mysql> BEGIN;
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> -- Update data in 'Employees'
mysql> UPDATE Employees SET Salary = Salary * 1.1 WHERE Department = 'HR';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

SQL JOINS

1. INNER JOIN:

An INNER JOIN returns only the rows that have matching values in both tables. Rows from the tables that do not have matching values are excluded from the result set.

Example:

Command Prompt - mysql -u root -p

```
mysql> -- Inner join to retrieve Employee information with Department names
mysql> SELECT Employees.EmployeeID, FirstName, LastName, Salary, DepartmentName
    -> FROM Employees
    -> INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;
+------------+-----------+----------+----------+----------------+
| EmployeeID | FirstName | LastName | Salary   | DepartmentName |
+------------+-----------+----------+----------+----------------+
|          1 | John      | Doe      | 60000.00 | HR             |
|          2 | Jane      | Smith    | 70000.00 | IT             |
|          3 | Bob       | Johnson  | 55000.00 | HR             |
|          4 | Alice     | Brown    | 80000.00 | Finance        |
+------------+-----------+----------+----------+----------------+
4 rows in set (0.00 sec)
```

2. LEFT (OUTER) JOIN:

A LEFT JOIN returns all rows from the left table and the matched rows from the right table. If there is no match, NULL values are returned for columns from the right table.

Example:

```
mysql> -- Left join to retrieve all employees and their corresponding department names
mysql> SELECT Employees.EmployeeID, FirstName, LastName, Salary, DepartmentName
    -> FROM Employees
    -> LEFT JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;
+------------+-----------+----------+----------+----------------+
| EmployeeID | FirstName | LastName | Salary   | DepartmentName |
+------------+-----------+----------+----------+----------------+
|          1 | John      | Doe      | 60000.00 | HR             |
|          2 | Jane      | Smith    | 70000.00 | IT             |
|          3 | Bob       | Johnson  | 55000.00 | HR             |
|          4 | Alice     | Brown    | 80000.00 | Finance        |
+------------+-----------+----------+----------+----------------+
4 rows in set (0.00 sec)
```

3. RIGHT (OUTER) JOIN:

A RIGHT JOIN is the opposite of the LEFT JOIN. It returns all rows from the right table and the matched rows from the left table. If there is no match, NULL values are returned for columns from the left table.

Example:

```
mysql> -- Right join to retrieve all departments and employees in those departments
mysql> SELECT Employees.EmployeeID, FirstName, LastName, Salary, DepartmentName
    -> FROM Employees
    -> RIGHT JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;
+------------+-----------+----------+----------+----------------+
| EmployeeID | FirstName | LastName | Salary   | DepartmentName |
+------------+-----------+----------+----------+----------------+
|          3 | Bob       | Johnson  | 55000.00 | HR             |
|          1 | John      | Doe      | 60000.00 | HR             |
|          2 | Jane      | Smith    | 70000.00 | IT             |
|          4 | Alice     | Brown    | 80000.00 | Finance        |
+------------+-----------+----------+----------+----------------+
4 rows in set (0.00 sec)
```

4. FULL (OUTER) JOIN:

A FULL JOIN returns all rows when there is a match in either the left or right table. If there is no match, NULL values are returned for columns from the table without a match.

5. CROSS JOIN:

A CROSS JOIN returns the Cartesian product of the two tables, meaning all possible combinations of rows from the first and second tables.

Example:

Command Prompt - mysql -u root -p

```
mysql> -- Cross join to retrieve a combination of all employees and departments
mysql> SELECT Employees.EmployeeID, FirstName, LastName, Salary, DepartmentName
    -> FROM Employees
    -> CROSS JOIN Departments;
+------------+-----------+----------+----------+----------------+
| EmployeeID | FirstName | LastName | Salary   | DepartmentName |
+------------+-----------+----------+----------+----------------+
|          1 | John      | Doe      | 60000.00 | Finance        |
|          1 | John      | Doe      | 60000.00 | IT             |
|          1 | John      | Doe      | 60000.00 | HR             |
|          2 | Jane      | Smith    | 70000.00 | Finance        |
|          2 | Jane      | Smith    | 70000.00 | IT             |
|          2 | Jane      | Smith    | 70000.00 | HR             |
|          3 | Bob       | Johnson  | 55000.00 | Finance        |
|          3 | Bob       | Johnson  | 55000.00 | IT             |
|          3 | Bob       | Johnson  | 55000.00 | HR             |
|          4 | Alice     | Brown    | 80000.00 | Finance        |
|          4 | Alice     | Brown    | 80000.00 | IT             |
|          4 | Alice     | Brown    | 80000.00 | HR             |
+------------+-----------+----------+----------+----------------+
12 rows in set (0.00 sec)

mysql>
```

6. SELF JOIN:

A SELF JOIN is a regular join, but the table is joined with itself. This can be useful when working with hierarchical data or when comparing rows within the same table.

Example:

Command Prompt - mysql -u root -p

```
tments.DepartmentID' at line 3
mysql> -- Self Join: Retrieve employees who are in the same department
mysql> SELECT e1.FirstName AS Employee1, e2.FirstName AS Employee2, d.DepartmentName
    -> FROM Employees e1
    -> JOIN Employees e2 ON e1.DepartmentID = e2.DepartmentID AND e1.EmployeeID < e2.EmployeeID
    -> JOIN Departments d ON e1.DepartmentID = d.DepartmentID;
+-----------+-----------+----------------+
| Employee1 | Employee2 | DepartmentName |
+-----------+-----------+----------------+
| John      | Bob       | HR             |
+-----------+-----------+----------------+
1 row in set (0.03 sec)
```

7. NATURAL JOIN:

A NATURAL JOIN is based on the equality of all columns with the same name in both tables. It automatically matches columns with the same name and returns the result.

Example:

```
mysql> -- Natural Join: Retrieve employees and their departments (based on common column DepartmentID)
mysql> SELECT Employees.*, Departments.DepartmentName
    -> FROM Employees
    -> NATURAL JOIN Departments;
+------------+-----------+----------+--------------+----------+----------------+
| EmployeeID | FirstName | LastName | DepartmentID | Salary   | DepartmentName |
+------------+-----------+----------+--------------+----------+----------------+
|          1 | John      | Doe      |            1 | 60000.00 | HR             |
|          2 | Jane      | Smith    |            2 | 70000.00 | IT             |
|          3 | Bob       | Johnson  |            1 | 55000.00 | HR             |
|          4 | Alice     | Brown    |            3 | 80000.00 | Finance        |
+------------+-----------+----------+--------------+----------+----------------+
4 rows in set (0.00 sec)

mysql>
```

Joins with GROUP BY, HAVING, and Aggregate Functions

```
mysql> -- Joins with GROUP BY, HAVING, and Aggregate Functions
mysql>
mysql> -- Retrieve the average salary of employees in each department
mysql> SELECT Departments.DepartmentName, AVG(Employees.Salary) AS AvgSalary
    -> FROM Employees
    -> INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID
    -> GROUP BY Departments.DepartmentName;
+----------------+--------------+
| DepartmentName | AvgSalary    |
+----------------+--------------+
| HR             | 57500.000000 |
| IT             | 70000.000000 |
| Finance        | 80000.000000 |
+----------------+--------------+
3 rows in set (0.00 sec)
```

Using HAVING to Filter Aggregated Data

Command Prompt - mysql -u root -p

```
mysql> SELECT Departments.DepartmentName, AVG(Employees.Salary) AS AvgSalary
    -> FROM Employees
    -> INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID
    -> GROUP BY Departments.DepartmentName
    -> HAVING AVG(Employees.Salary) > 55000;
+----------------+--------------+
| DepartmentName | AvgSalary    |
+----------------+--------------+
| HR             | 57500.000000 |
| IT             | 70000.000000 |
| Finance        | 80000.000000 |
+----------------+--------------+
3 rows in set (0.03 sec)

mysql>
```

Using Nested Query

Command Prompt - mysql -u root -p

```
mysql> -- Nested Query: Retrieve department names with their average salary
mysql> SELECT DepartmentName, AvgSalary
    -> FROM (
    ->     SELECT Departments.DepartmentName, AVG(Employees.Salary) AS AvgSalary
    ->     FROM Employees
    ->     INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID
    ->     GROUP BY Departments.DepartmentName
    -> ) AS DepartmentAverages;
+----------------+--------------+
| DepartmentName | AvgSalary    |
+----------------+--------------+
| HR             | 57500.000000 |
| IT             | 70000.000000 |
| Finance        | 80000.000000 |
+----------------+--------------+
3 rows in set (0.00 sec)
```

Using Correlated Subquery

```
mysql> -- Correlated Subquery: Retrieve employees with salary above their department's average
mysql> SELECT EmployeeID, FirstName, LastName, DepartmentName, Salary
    -> FROM Employees e
    -> INNER JOIN Departments d ON e.DepartmentID = d.DepartmentID
    -> WHERE Salary > (
    ->     SELECT AVG(Salary)
    ->     FROM Employees
    ->     WHERE DepartmentID = e.DepartmentID
    -> );
+------------+-----------+----------+----------------+----------+
| EmployeeID | FirstName | LastName | DepartmentName | Salary   |
+------------+-----------+----------+----------------+----------+
|          1 | John      | Doe      | HR             | 60000.00 |
+------------+-----------+----------+----------------+----------+
1 row in set (0.03 sec)

mysql>
```