INTRODUCTION

MySQL is one of the most popular open-source Relational Database Management Systems (RDBMS).

A MySQL client might hold multiple databases. Each database might contain multiple tables with each table holding data of the same type. Each table contains rows and columns with each row denoting a single entry and each column denoting different attributes of the entries.

Data Types in MySQL

MySQL supports a list of predefined data types that we can use to effectively model our tables. These data types are:

INT:	*for integer data*.
DECIMAL:	*for decimal data.*
BOOLEAN:	*for boolean data*.
CHAR:	*for fixed-length string.*
VARCHAR:	*for variable-length string.*
TEXT:	*for long-form text.*
DATE:	*for date data.*
TIME:	*for time data.*
DATETIME:	*for date-time data.*
TIMESTAMP:	*for timestamp data.*

MySQL Commands

The CREATE TABLE statement is used in MySQL to create a new table in a database. The syntax for this is shown below:

Syntax:

```
CREATE TABLE [IF NOT EXISTS] table_name(

column1_definition,

column2_definition,

...,
```

```
table_constraints

);
```

DROP TABLE

The DROP TABLE statement is used in MySQL to drop or delete a table from the database. The syntax for this is shown below:

Syntax:

```
DROP [TEMPORARY] TABLE [IF EXISTS] table_name;
```

RENAME TABLE

The RENAME TABLE statement is used in MySQL to rename the existing tables. One or more tables can be renamed using this statement.

Syntax:

```
RENAME TABLE old_table_name to new_table_name;
```

INSERT INTO

The INSERT INTO statement is used in MySQL to insert rows into a table. One or more rows can be inserted into a table using this statement.

Syntax:

```
INSERT INTO table_name (column1, column2, ….)

VALUES (value1, value2, ....),

    (value1, value2, ...),

    (value1, value2, ...),

    .....;
```

SELECT

The SELECT statement is used for querying data. It allows you to select data from one or more tables.

Syntax:

```
SELECT column1, column 2,....

FROM table_name;
```

SELECT DISTINCT

The SELECT DISTINCT statement is used in MySQL to remove duplicate rows.

Syntax:

```
SELECT DISTINCT column1, column2,...

FROM table_name;
```

ALTER TABLE

The statement ALTER TABLE can be used in MySQL to add a column, modify a column, drop a column, rename a column from a table.

1. Add a column to a table using ALTER TABLE with ADD

The syntax for adding a column to a table is shown below:

Syntax:

```
ALTER TABLE table_name

ADD

new_column_name column_definition

[FIRST|AFTER column_name]
```

2. Modify a column using ALTER TABLE with MODIFY

We can modify one or multiple columns of a table using MODIFY with ALTER TABLE statements.

The syntax for this is shown below:

```
ALTER TABLE table_name

MODIFY

column_name column_definition

[FIRST|AFTER column_name]
```

3. Rename columns using ALTER TABLE with CHANGE COLUMN

We can rename a column of a table using the CHANGE COLUMN keyword with ALTER TABLE.

The syntax for this is shown below:

```
ALTER TABLE table_name

CHANGE COLUMN original_column_name new_column_name
column_definition

[FIRST | AFTER column_name]
```

4. Drop a column using ALTER TABLE with DROP COLUMN

We can drop a column or multiple columns using DROP COLUMN with ALTER TABLE.

The syntax for this is shown below:

```
ALTER TABLE table_name

DROP COLUMN column_name;
```

ORDER BY in MySQL

The ORDER BY clause is used in MySQL to sort the retrieved data in a particular order.

Syntax:

```
SELECT column1, column2,...

FROM table_name

ORDER BY Column1 ASC/DESC, Column2 ASC/DESC,... ;
```

Aliases in MySQL

Aliases are used to give columns or tables a temporary or simple name. AS keyword is used to create an alias.

- **Column Alias**

The syntax for column name aliases is written below.

Syntax:

```
SELECT column_name AS given_name

FROM table_name;
```

- **Table Alias**

The aliases can be used to give simple and different names to tables also.

Syntax:

```
SELECT column1, column2, ….

FROM table_name AS given_name;
```

WHERE clause in MySQL

The WHERE clause is used to apply a particular condition while selecting rows from the table. It helps in filtering the rows according to any particular condition.

Syntax:

```
SELECT column1, column2, …..

FROM table_name
```

```
WHERE condition;
```

IN Operator in MySQL

The IN operator is used to check if a value matches any of the values in a list of values. It is similar to the OR operator as if any of the values in the list matches it returns true.

Syntax:

```
SELECT column1, column2, …..

FROM table_name

WHERE column_name IN (value1, value2, ….);
```

or

```
SELECT column1, column2, …..

FROM table_name

WHERE column_name IN (SELECT statement );
```

LIKE Operator in MySQL

The LIKE operator is used in MySQL to search for a specific pattern in a string. If an expression matches the pattern, it returns true else false.

There are two wildcards in MySQL, used with the LIKE operator for searching a pattern.

- The percentage sign (%). It represents zero or more characters.
- The underscore sign (_). It represents a single character.

Syntax:

```
SELECT column1, column2, …..

FROM table_name
```

```
WHERE column_name LIKE pattern;
```

For example:

```
SELECT customer_id, customer_name, product_id

FROM customers

WHERE customer_name LIKE 'A'%;
```

IS NULL Operator in MySQL

The IS NULL is used to check if a value is NULL or not. If the value is NULL, it returns true else false.

Syntax:

```
SELECT column1, column2, …..

FROM table_name

WHERE column_name IS NULL;
```

JOIN

Joins are used in relational databases to combine data from multiple tables based on a common column between them. A foreign key may be used to reference a row in another table and join can be done based on those columns. Two or more tables may have some related data, and to combine all the data from multiple tables joins are used.

There are different types of joins in MySQL.

1. INNER JOIN
2. LEFT JOIN
3. RIGHT JOIN
4. CROSS JOIN

- **INNER JOIN**

The INNER JOIN produces the output by combining those rows which have matching column values.

Syntax:

```
SELECT column_names

FROM table1

INNER JOIN table2 ON table1.common_column=table2.common_column

INNER JOIN table3 ON table1.common_column=table3.common_column

...;
```

- **LEFT JOIN**

The LEFT JOIN returns all the rows from the left table ‘A’ and the matching rows from the right table ‘B’ in the join. The rows from the left table, which have no matching values in the right table will be returned with a NULL value in the link column.

Syntax:

```
SELECT column_names

FROM table1

LEFT JOIN table2 ON table1.common_column=table2.common_column;
```

- **RIGHT JOIN**

The RIGHT JOIN returns all the rows from the right table ‘B’ and the matching rows from the left table ‘A’ in the join. The rows from the right table, which have no matching values in the left table will be returned with a NULL value in the link column.

Syntax:

```
SELECT Column_names

FROM table1
```

```
RIGHT JOIN table2 ON table1.common_column=table2.common_column;
```

- **CROSS JOIN**

CROSS JOIN returns the cartesian product of rows from the tables in the join. It combines each row of the first table with each row of the second table. If there are X rows in the first table and Y rows in the second table then the number of rows in the joined table will be X*Y.

Syntax:

```
SELECT column_names

FROM table1

CROSS JOIN table2;
```

GROUP BY

The GROUP BY clause is used to arrange the rows in a group using a particular column value. If there are multiple rows with the same value for a column then all those rows will be grouped with that column value.

Syntax:

```
SELECT column1,column2,…

FROM table_name

WHERE condition

GROUP BY column1,column2, …

Order BY column1, column2, ….
```

The GROUP BY clause is generally used with aggregate functions like SUM, AVG, COUNT, MAX, MIN. The aggregate functions provide information about each group.

HAVING

The HAVING clause is used with the GROUP BY clause in a query to specify come conditions and filter some groups or aggregates that fulfill those conditions. The difference between WHERE and HAVING is, WHERE is used to filter rows, and HAVING is used to filter groups by applying some conditions.

Syntax:

```
SELECT column1, column2, …
FROM table_name
WHERE conditions
GROUP BY column1, column2, …..
HAVING conditions
```