MYSQL

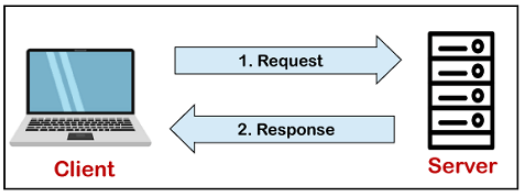
MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by **Oracle Company**.

MySQL is a Relational Database Management System (RDBMS) software that provides many things, which are as follows:

- It allows us to implement database operations on tables, rows, columns, and indexes.
- It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.
- It provides the Referential Integrity between rows or columns of various tables.
- It allows us to updates the table indexes automatically.
- It uses many SQL queries and combines useful information from multiple tables for the end-users.

MySQL Working

MySQL follows the working of Client-Server Architecture. This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services. Here, the clients make requests through a graphical user interface (GUI), and the server will give the desired output as soon as the instructions are matched. The process of MySQL environment is the same as the client-server model.



The core of the MySQL database is the MySQL Server. This server is available as a separate program and responsible for handling all the database instructions, statements, or commands. The working of MySQL database with MySQL Server are as follows:

1. MySQL creates a database that allows you to build many tables to store and manipulate data and defining the relationship between each table.
2. Clients make requests through the GUI screen or command prompt by using specific SQL expressions on MySQL.
3. Finally, the server application will respond with the requested expressions and produce the desired result on the client-side.

A client can use any MySQL GUI. But, it is making sure that your GUI should be lighter and user-friendly to make your data management activities faster and easier. Some of the most widely used MySQL GUIs are MySQL Workbench, SequelPro, DBVisualizer, and the Navicat DB Admin Tool. Some GUIs are commercial, while some are free with limited functionality, and some are only compatible with MacOS. Thus, you can choose the GUI according to your needs.

MySQL Data Types

A Data Type specifies a particular type of data, like integer, floating points, Boolean, etc. It also identifies the possible values for that type, the operations that can be performed on that type, and the way the values of that type are stored. In MySQL, each database table has many columns and contains specific data types for each column.

We can determine the data type in MySQL with the following characteristics:

- The type of values (fixed or variable) it represents.
- The storage space it takes is based on whether the values are a fixed-length or variable length.
- Its values can be indexed or not.
- How MySQL performs a comparison of values of a particular data type.

Database

A database is an application that stores the organized collection of records. It can be accessed and manage by the user very easily. It allows us to organize data into tables, rows, columns, and indexes to find the relevant information very quickly. Each database contains distinct API for performing database operations such as creating, managing, accessing, and searching the data it stores. Today, many databases available like MySQL, Sybase, Oracle, MongoDB, PostgreSQL, SQL Server, etc.

SQL QUERIES

CREATE DATABASE

Here, we are going to create a database name **"employeedb"** using the following statement:

Syntax:

Create Database database_name;

Example:

CREATE DATABASE employeesdb;

SHOW DATABASES

We can check the created database using the following query:

SHOW DATABASES;

DROP DATABASE

Syntax:

DROP DATABASE database_name;

Example:

DROP DATABASE employeesdb;

CREATE TABLE

Syntax:

CREATE TABLE table_name (column1 datatype, column2 datatype, ... columnN datatype, PRIMARY KEY (one or more columns));

Example:

CREATE TABLE CUSTOMERS (ID INT AUTO_INCREMENT, NAME VARCHAR(20) NOT NULL, AGE INT NOT NULL, ADDRESS CHAR (25), SALARY DECIMAL (18, 2), PRIMARY KEY (ID));

Output:

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
ID	int	NO	PRI	NULL	auto_increment
NAME	varchar(20)	NO		NULL	

AGE	int	NO		NULL	
ADDRESS	char(25)	YES		NULL	
SALARY	decimal(18,2)	YES		NULL	

SHOW TABLES

SHOW TABLES command to retrieve the names of tables that are present in a specific database.

Example:

SHOW TABLES;

ALTER TABLE

ALTER command is used to modify the structure of an existing table. It allows you to make various changes, such as adding, deleting, or modify columns within the table.

Syntax:

ALTER TABLE table_name [alter_option ...];

Example:

ALTER TABLE CUSTOMERS DROP ID;

ALTER TABLE CUSTOMERS ADD MOBILE_NO INT;

RENAME TABLES

RENAME TABLE statement is used to rename an existing table in a database with another name.

Syntax:

RENAME TABLE table_name TO new_name;

Example:

RENAME TABLE CUSTOMERS to BUYERS;

TRUNCATE TABLE

TRUNCATE TABLE statement is used to delete only the data of an existing table, but not the table. This command helps to TRUNCATE a table completely in one go instead of deleting table records one by one which will be very time consuming and hefty process.

Syntax:

TRUNCATE TABLE table_name;

Example:

TRUNCATE TABLE CUSTOMERS;

TRUNCATE vs DELETE

Following are some major differences between the TRUNCATE and DELETE commands, even though they work similar logically:

DELETE	TRUNCATE
The DELETE command in SQL removes one or more rows from a table based on the conditions specified in a WHERE Clause.	The TRUNCATE command is used to remove all of the rows from a table, regardless of whether or not any conditions are met.
It is a DML(Data Manipulation Language) command.	It is a DDL(Data Definition Language) command.
There is a need to make a manual COMMIT after making changes to the DELETE command, for the modifications to be committed.	When you use the TRUNCATE command, the modifications made to the table are committed automatically.
It deletes rows one at a time and applies some criteria to each deletion.	It removes all of the information in one go.
The WHERE clause serves as the condition in this case.	There is no necessity of using a WHERE Clause.
All rows are locked after deletion.	TRUNCATE utilizes a table lock, which locks the pages so they cannot be deleted.

It makes a record of each and every transaction in the log file.	The only activity recorded is the deallocation of the pages on which the data is stored.
It consumes a greater amount of transaction space compared to TRUNCATE command.	It takes comparatively less amount of transaction space.
If there is an identity column, the table identity is not reset to the value it had when the table was created.	It returns the table identity to a value it was given as a seed.
It requires authorization to delete.	It requires table alter permission.
When it comes to large databases, it is much slower.	It is faster.

TRUNCATE vs DROP

The TRUNCATE and DROP are two different commands. TRUNCATE just deletes the table's records, whereas DROP command deletes the table entirely from the database.

However, there are still some differences between these commands, which are summarized in the following table −

DROP	TRUNCATE
The DROP command in SQL removes an entire table from a database including its definition, indexes, constraints, data etc.	The TRUNCATE command is used to remove all of the rows from a table, regardless of whether or not any conditions are met and resets the table definition.
It is a DDL(Data Definition Language) command.	It is also a DDL(Data Definition Language) command.
The table space is completely freed from the memory.	The table still exists in the memory.
All the integrity constraints are removed.	The integrity constraints still exist in the table.

Requires ALTER and CONTROL permissions on the table schema and table respectively, to be able to perform this command.	Only requires the ALTER permissions to truncate the table.
DROP command is much slower than TRUNCATE but faster than DELETE.	It is faster than both DROP and DELETE commands.

RENAME COLUMN

We can change the name of one or multiple columns of a specified table using the **ALTER TABLE RENAME COLUMN** command.

Syntax:

ALTER TABLE table_name RENAME COLUMN old_column1_name TO new_column1_name, RENAME COLUMN old_column2_name TO new_column2_name, ...;

Example:

ALTER TABLE CUSTOMERS RENAME COLUMN ID TO cust_id;

INSERT QUERIES

It is used to add data to the table.

Syntax:

INSERT INTO TABLE_NAME (column1, column2, column3,...columnN) VALUES (value1, value2, value3,...valueN);

Example:

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

INSERT INTO CUSTOMERS VALUES (6, 'Komal', 22, 'Hyderabad', 4500.00), (7, 'Muffy', 24, 'Indore', 10000.00);

SELECT QUERY

SELECT command is used to fetch data from the MySQL database in the form of a result table. These result tables are called result-sets.

Syntax:

SELECT field1, field2,...fieldN FROM table_name1, table_name2... [WHERE Clause] [OFFSET M][LIMIT N]

Example:

FETCHING ALL THE COLUMNS FROM TABLE

SELECT * from CUSTOMERS;

FETCHING SPECIFIC COLUMNS FROM TABLE

SELECT ID, NAME, ADDRESS FROM CUSTOMERS;

UPDATE QUERY

UPDATE Query is used to modify the existing records in a table. This statement is a part of Data Manipulation Language in SQL, as it only modifies the data present in a table without affecting the table's structure.

Syntax:

UPDATE table_name SET field1 = new-value1, field2 = new-value2 [WHERE Clause]

Example:

UPDATE CUSTOMERS SET NAME = 'Nikhilesh' WHERE ID = 6;

DELETE QUERY

If we want to delete a record from any MySQL table, then we can use the SQL command **DELETE FROM**. This statement is a part of Data Manipulation Language in SQL as it interacts with the data in a MySQL table rather than the structure.

DELETE statement can be used to delete multiple rows of a single table and records across multiple tables. However, in order to filter the records to be deleted, we can use the WHERE clause along with the DELETE statement.

Syntax:

DELETE FROM table_name [WHERE Clause]

Example:

DELETE FROM CUSTOMERS WHERE ID = 1;

DELETE FROM CUSTOMERS;

VIEWS

Views are a type of virtual tables. They are stored in the database with an associated name. They allow users to do the following −

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

CREATE VIEW

Syntax:

CREATE VIEW view_name AS select_statements FROM table_name;

Example:

CREATE VIEW first_view AS SELECT * FROM CUSTOMERS;

CREATE VIEW test_view AS SELECT * FROM CUSTOMERS WHERE SALARY>3000;

FOREIGN KEY

In the relational databases, a foreign key is a field or a column that is used to establish a link between two tables. In simple words you can say that, a foreign key in one table used to point primary key in another table.

Syntax:

```
CREATE TABLE childTable (
col1 int NOT NULL,
col2 int NOT NULL,
col3 int,
……...
PRIMARY KEY (col1),
FOREIGN KEY (col3) REFERENCES parentTable(parent_Primary_key)
);
```

Example:

1. Creation of Parent Table DataFlair

```
CREATE TABLE DataFlair(
emp_id varchar(5) NOT NULL,
name varchar(50),
location varchar(50),
experience int,
PRIMARY KEY(emp_id));
```

2. Creation of Child Table Location

```
CREATE TABLE location(
location_id varchar(5) NOT NULL,
location varchar(50) NOT NULL,
office_size int,
PRIMARY KEY(location_id),
FOREIGN KEY(location) REFERENCES dataflair(location));
```

3. Viewing the Parent and the Child Table

A. Parent Table or the DataFlair Table

```
SELECT * FROM DataFlair;
```

B. Child Table of the Location Table:

```
SELECT * FROM Location;
```

4. Running Queries on the Parent and Child Table

Example 1: To view the name of employee, location and the location id by using both the DataFlair and Location table.

```
SELECT emp_id , name,
location.location_id,DataFlair.location
FROM DataFlair RIGHT JOIN Location
ON DataFlair.location=Location.location
WHERE emp_id IS NOT NULL;
```

Example 2: To view the locations where we have an office of DataFlair as of now.

```
SELECT DISTINCT(location.location_id) AS Location_ID,
DataFlair.location AS Office_Location
```

```
FROM DataFlair RIGHT JOIN Location
ON DataFlair.location=Location.location
WHERE emp_id IS NOT NULL;
```

HOW TO DROP PRIMARY AND FOREIGN KEY

Syntax:

```
ALTER TABLE tableName
DROP PRIMARY KEY;
```

Example:

```
ALTER TABLE dataflair
DROP PRIMARY KEY;
```

Syntax:

```
ALTER TABLE tableName
DROP FOREIGN KEY foreignKeyName;
```

Example:

```
ALTER TABLE location
DROP FOREIGN KEY location;
```

DIFFERENCE BETWEEN PRIMARY AND FOREIGN KEY

Sr.No	Primary Key	Foreign Key
1	Used to maintain the unique identification of data in the table.	Used to maintain the relationship between two or more relational tables.
2	Helps us to identify data in a database table.	Helps to identify the data in another table using the connection with the foreign key.
3	A table can have only one Primary Key.	A table can have any number of Foreign Keys.
4	The primary key is unique and Not Null.	A foreign key can contain duplicate values also.

5	Primary key can’t take Null as a value.	A foreign key can take NULL entries also.
6	Primary Key can’t be modified once entered.	A foreign key can be modified at any instance of time.
7	We can have Primary keys for temporary tables as well.	We can’t have Foreign keys for the temporary tables.
8	A Primary key can be defined on its own.	For defining a Foreign key, we need a parent table with a Primary Key.
9	Primary key creates clustered indexes on the table.	Foreign key does not create indexes on the table neither clustered nor unclustered.

JOIN

As the name shows, JOIN means to combine something. In case of SQL, JOIN means "**to combine two or more tables**". The SQL JOIN clause takes records from two or more tables in a database and combines it together.

ANSI standard SQL defines five types of JOIN :

1. inner join,
2. left outer join,
3. right outer join,
4. full outer join, and
5. cross join.

INNER JOIN also known as simple join is the most common type of join.

Example:

1. **Staff table**

ID	Staff_NAME	Staff_AGE	STAFF_ADDRESS	Monthley_Package
1	ARYAN	22	MUMBAI	18000
2	SUSHIL	32	DELHI	20000

3	MONTY	25	MOHALI	22000
4	AMIT	20	ALLAHABAD	12000

2. **Payment table**

Payment_ID	DATE	Staff_ID	AMOUNT
101	30/12/2009	1	3000.00
102	22/02/2010	3	2500.00
103	23/02/2010	4	3500.00

To Join these two tables

```
SELECT Staff_ID, Staff_NAME, Staff_AGE, AMOUNT
  FROM STAFF s, PAYMENT p
  WHERE s.ID =p.STAFF_ID;
```

This will produce the result like this:

STAFF_ID	NAME	Staff_AGE	AMOUNT
3	MONTY	25	2500
1	ARYAN	22	3000
4	AMIT	25	3500
1	ARYAN	22	3000

OUTER JOIN

In the SQL outer JOIN, ***all the content from both the tables is integrated together.*** Even though the records from both the tables are matched or not, the matching and non-matching records from both the tables will be considered an output of the outer join in SQL.

There are three different types of outer join in SQL:

- **Left Outer Join**
- **Right Outer Join**
- **Full Outer Join**

Table 1: employee

EmployeeID	Employee_Name	Employee_Salary
1	Arun Tiwari	50000
2	Sachin Rathi	64000
3	Harshal Pathak	48000
4	Arjun Kuwar	46000
5	Sarthak Gada	62000
6	Saurabh Sheik	53000
7	Shubham Singh	29000
8	Shivam Dixit	54000
9	Vicky Gujral	39000

10	Vijay Bose	28000

Table 2: Department

DepartmentID	Department_Name	Employee_ID
1	Production	1
2	Sales	3
3	Marketing	4
4	Accounts	5
5	Development	7
6	HR	9
7	Sales	10

Table 3: Loan

LoanID	Branch	Amount
1	B1	15000
2	B2	10000

3	B3	20000
4	B4	100000
5	B5	150000
6	B6	50000
7	B7	35000
8	B8	85000

Table 4: Borrower

CustID	CustName	LoanID
1	Sonakshi Dixit	1
2	Shital Garg	4
3	Swara Joshi	5
4	Isha Deshmukh	2
5	Swati Bose	7
6	Asha Kapoor	10
7	Nandini Shah	9

1. Left Outer Join:

If we use the left outer join to combine two different tables, then we will get all the records from the left table. But we will get only those records from the right table, which have the corresponding key in the left table.

Syntax of writing a query to perform left outer join:

SELECT TableName1.columnName1, TableName2.columnName2 FROM TableName1 LEFT OUTER JOIN TableName2 ON TableName1.ColumnName = TableName2.ColumnName;

Example 1:

Write a query to perform left outer join considering employee table as the left table and department table as the right table.

SELECT e.EmployeeID, e.Employee_Name, e.Employee_Salary, d.DepartmentID, d.Departmen t_Name FROM employee e LEFT OUTER JOIN department d ON e.EmployeeID = d.Employee _ID;

Output:

EmployeeID	Employee_Name	Employee_Salary	DepartmentID	Department_Name
1	Arun Tiwari	50000	1	Production
2	Sachin Rathi	64000	NULL	NULL
3	Harshal Pathak	48000	2	Sales
4	Arjun Kuwar	46000	3	Marketing
5	Sarthak Gada	62000	4	Accounts
6	Saurabh Sheik	53000	NULL	NULL

7	Shubham Singh	29000	5	Development
8	Shivam Dixit	54000	NULL	NULL
9	Vicky Gujral	39000	6	HR
10	Vijay Bose	28000	7	Sales

2. Right Outer Join:

Right outer join is the reverse of left outer join. If we use the right outer join to combine two different tables, then we will get all the records from the right table. But we will get only those records from the left table, which have the corresponding key in the right table.

Syntax of writing a query to perform right outer join:

SELECT TableName1.columnName1, TableName2.columnName2 **FROM** TableName1 RIGHT OUTER JOIN TableName2 **ON** TableName1.ColumnName = TableName2.ColumnName;

Example 1:

Write a query to perform right outer join considering employee table as the left table and department table as the right table.

Query:

SELECT e.EmployeeID, e.Employee_Name, e.Employee_Salary, d.DepartmentID, d.Department_Name **FROM** employee e RIGHT OUTER JOIN department d **ON** e.EmployeeID = d.Employee_ID;

Output:

EmployeeID	Employee_Name	Employee_Salary	DepartmentID	Department_Name
1	Arun Tiwari	50000	1	Production

3	Harshal Pathak	48000	2	Sales
4	Arjun Kuwar	46000	3	Marketing
5	Sarthak Gada	62000	4	Accounts
7	Shubham Singh	29000	5	Development
9	Vicky Gujral	39000	6	HR
10	Vijay Bose	28000	7	Sales

3. Full Outer Join:

If we use a full outer join to combine two different tables, ***then we will get all the records from both the table,***e., we will get all the records from the left table as well as the right table.

MySQL doesn't support FULL OUTER JOIN directly. So to implement full outer join in MySQL, we will execute two queries in a single query. The first query will be of LEFT OUTER JOIN, and the second query will be of RIGHT OUTER JOIN. We will combine the first and second query with the UNION operator to see the results of FULL OUTER JOIN.

Syntax of writing a query to perform full outer join:

SELECT TableName1.columnName1, TableName2.columnName2 **FROM** TableName1 LEFT OUTER JOIN TableName2 **ON** TableName1.ColumnName = TableName2.ColumnName **UNI ON SELECT** TableName1.columnName1, TableName2.columnName2 **FROM** TableName1 R IGHT OUTER JOIN TableName2 **ON** TableName1.ColumnName = TableName2.ColumnName

Example 1:

Write a query to perform full outer join considering the employee table as the left table and department table as the right table.

Query:

SELECT e.EmployeeID, e.Employee_Name, e.Employee_Salary, d.DepartmentID, d.Departme nt_Name **FROM** department d LEFT OUTER JOIN employee e **ON** e.EmployeeID = d.Employ ee_ID **UNION SELECT** e.EmployeeID, e.Employee_Name, e.Employee_Salary, d.DepartmentI

D, d.Department_Name **FROM** department d RIGHT OUTER JOIN employee e **ON** e.Employe eID = d.Employee_ID;

Output:

EmployeeID	Employee_Name	Employee_Salary	DepartmentID	Department_Name
1	Arun Tiwari	50000	1	Production
3	Harshal Pathak	48000	2	Sales
4	Arjun Kuwar	46000	3	Marketing
5	Sarthak Gada	62000	4	Accounts
7	Shubham Singh	29000	5	Development
9	Vicky Gujral	39000	6	HR
10	Vijay Bose	28000	7	Sales
2	Sachin Rathi	64000	NULL	NULL
6	Saurabh Sheik	53000	NULL	NULL
8	Shivam Dixit	54000	NULL	NULL

LEFT JOIN

Join operation in SQL is used to **combine multiple tables together into a single table.** If we use ***left join to combine two different tables, then we will get all the records from the left table***. But we will get only those records from the right table, which have the corresponding key in the

left table. Rest other records in the right table for which the common column value doesn't match with the common column value of the left table; then, it is displayed as NULL.

Query to perform the left join operation in SQL-

```
SELECT TableName1.columnName1, TableName2.columnName2 FROM TableName1 LEFT J
OIN TableName2 ON TableName1.ColumnName = TableName2.ColumnName;
```

Example:

```
SELECT e.EmployeeID, e.Employee_Name, e.Employee_Salary, d.DepartmentID, d.Departmen
t_Name FROM employee e LEFT JOIN department d ON e.EmployeeID = d.Employee_ID;
```

Output:

EmployeeID	Employee_Name	Employee_Salary	DepartmentID	Department_Name
1	Arun Tiwari	50000	1	Production
2	Sachin Rathi	64000	NULL	NULL
3	Harshal Pathak	48000	2	Sales
4	Arjun Kuwar	46000	3	Marketing
5	Sarthak Gada	62000	4	Accounts
6	Saurabh Sheik	53000	NULL	NULL
7	Shubham Singh	29000	5	Development
8	Shivam Dixit	54000	NULL	NULL
9	Vicky Gujral	39000	6	HR

10	Vijay Bose	28000	7	Sales