DATA BASE MANAGEMENT SYSTEM

DBMS LAB

ETCS-256

Faculty: **Prerna Sharma**

Name: **Syeda Reeha Quasar**

Roll No.: **14114802719**

Semester: **4 (4C7)**

Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

INDEX

S.No.	Program Name	Date	Marks	Signature	Remarks
1.	To Draw an E-R Diagram to represent different entities, attributes and relations in a university.	24-03-2021			
2.	Creating Database tables and performing the operation of table creations, insert data and fetch data.	07-04-2021			
3.	Write queries for retrieving records from table using SELECT command and WHERE clause.	14-04-2021			
4.	Write SQL commands for implementing ALTER, UPDATE and DELETE.	21-04-2021			
5.	Write the queries to implement the concept of Integrity Constraints like Primary Key, Foreign Key, NOT NULL to the tables.	19-05-2021			
6.	Write the queries for implementing the following functions: MAX(), MIN(), AVG(), COUNT() and other logical pattern matching operations.	26-05-201			
7.	Write the SQL queries to implement DATE and MONTH commands.	02-06-2021			
8.	Write the SQL queries to implement Having and Group By Clauses on table.	02-06-2021			
9.	Write the SQL queries to implement the joins.	09-06-2021			
10.	Write the SQL queries to create the views.	16-06-2021			

EXPERIMENT - 1

DATABASE MANAGEMENT SYSTEMS LAB

Aim

To Draw an E-R Diagram to represent different entities, attributes and relations in a university.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 1

Aim:

To Draw an E-R Diagram to represent different entities, attributes and relations in a university.

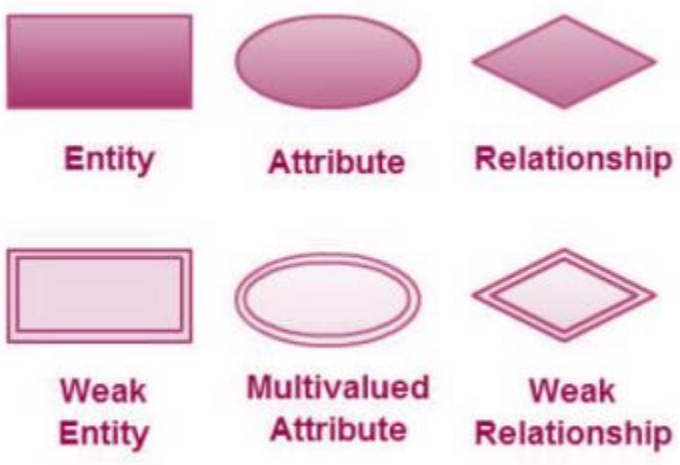
Theory:

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

ER Diagram Symbols & Notations: There are three basic elements in an ER Diagram: entity, attribute, relationship. There are more elements which are based on the main elements. They are weak entity, multi-valued attribute, derived attribute, weak relationship, and recursive relationship. Cardinality and ordinality are two other notations used in ER diagrams to further define relationships.

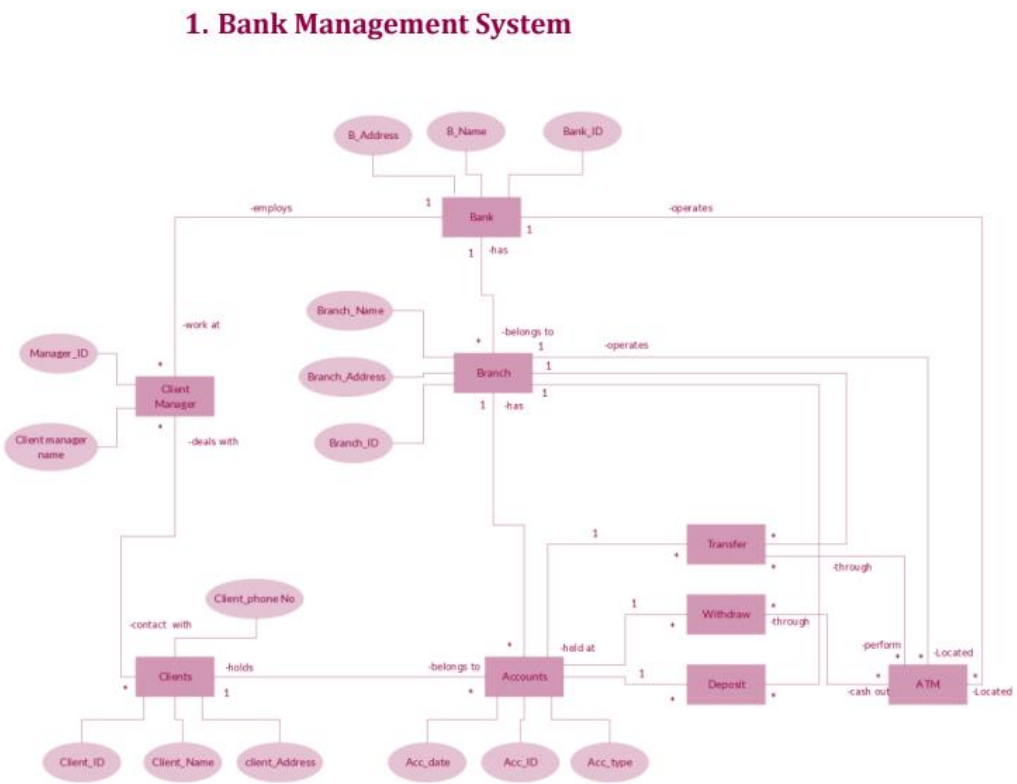


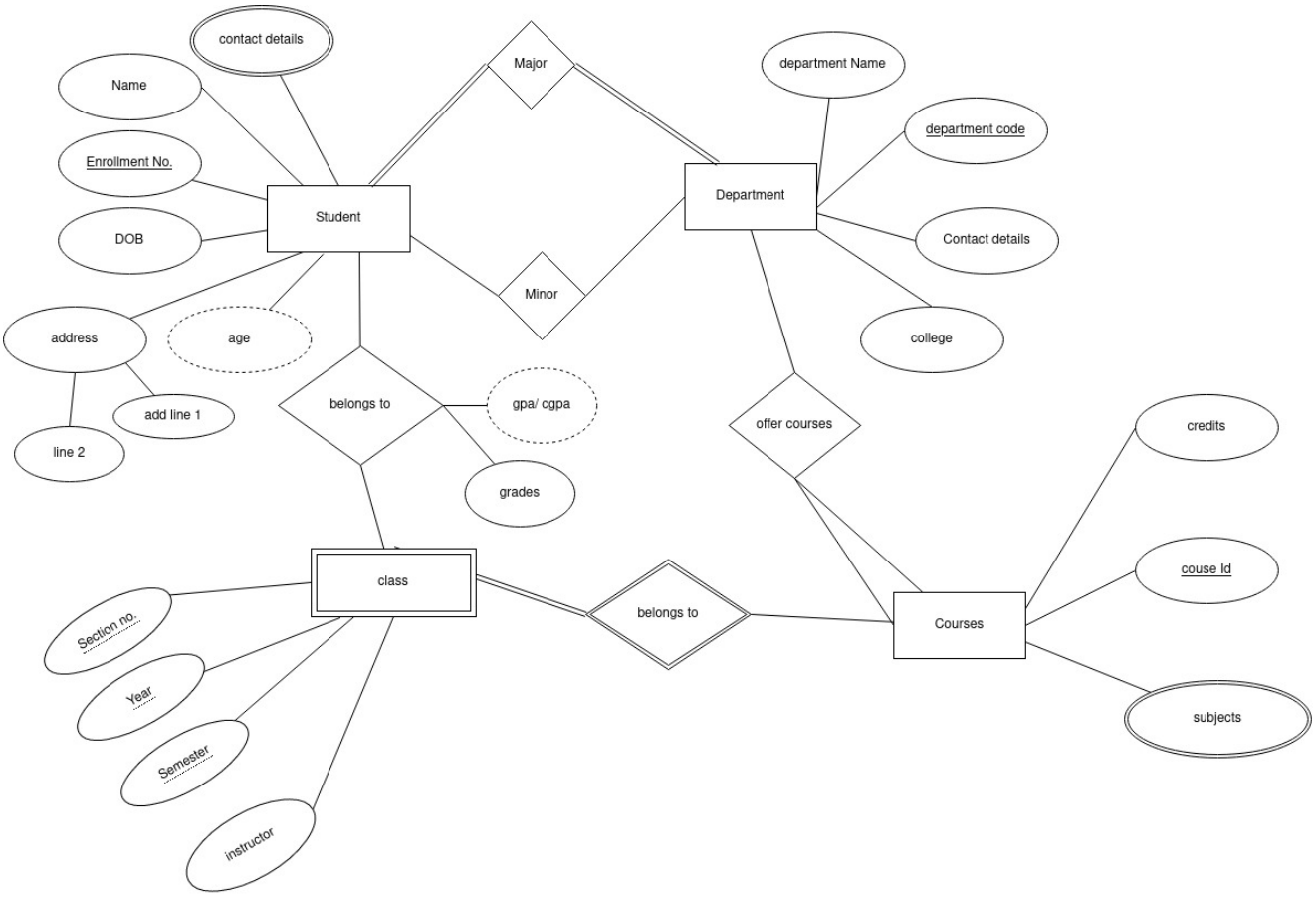
Why use ER Diagrams?

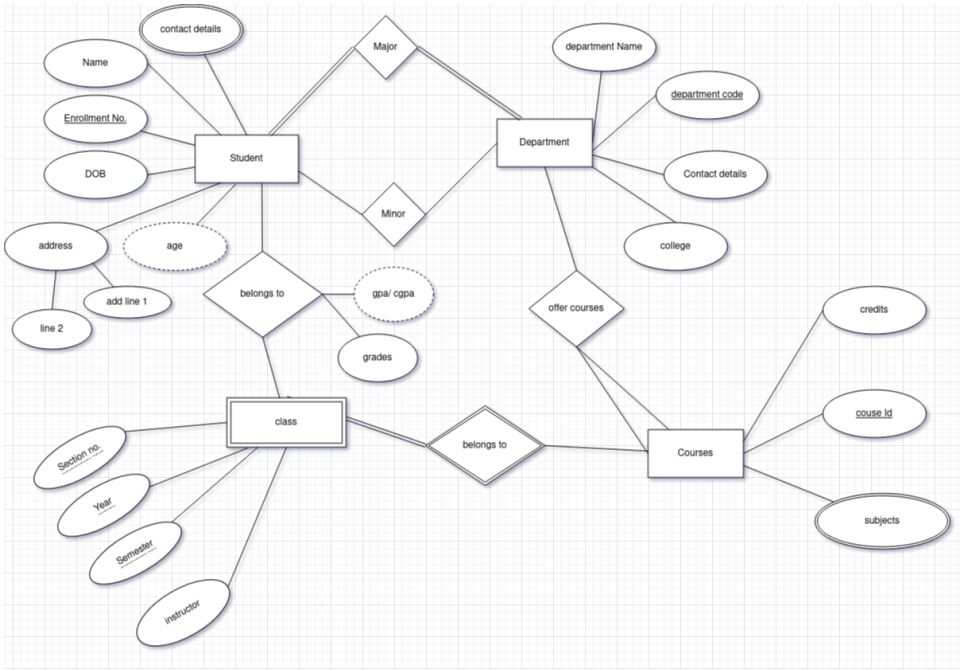
Here, are prime reasons for using the ER Diagram

- Helps you to define terms related to entity relationship modeling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
- ERD Diagram allows you to communicate with the logical structure of the database to users

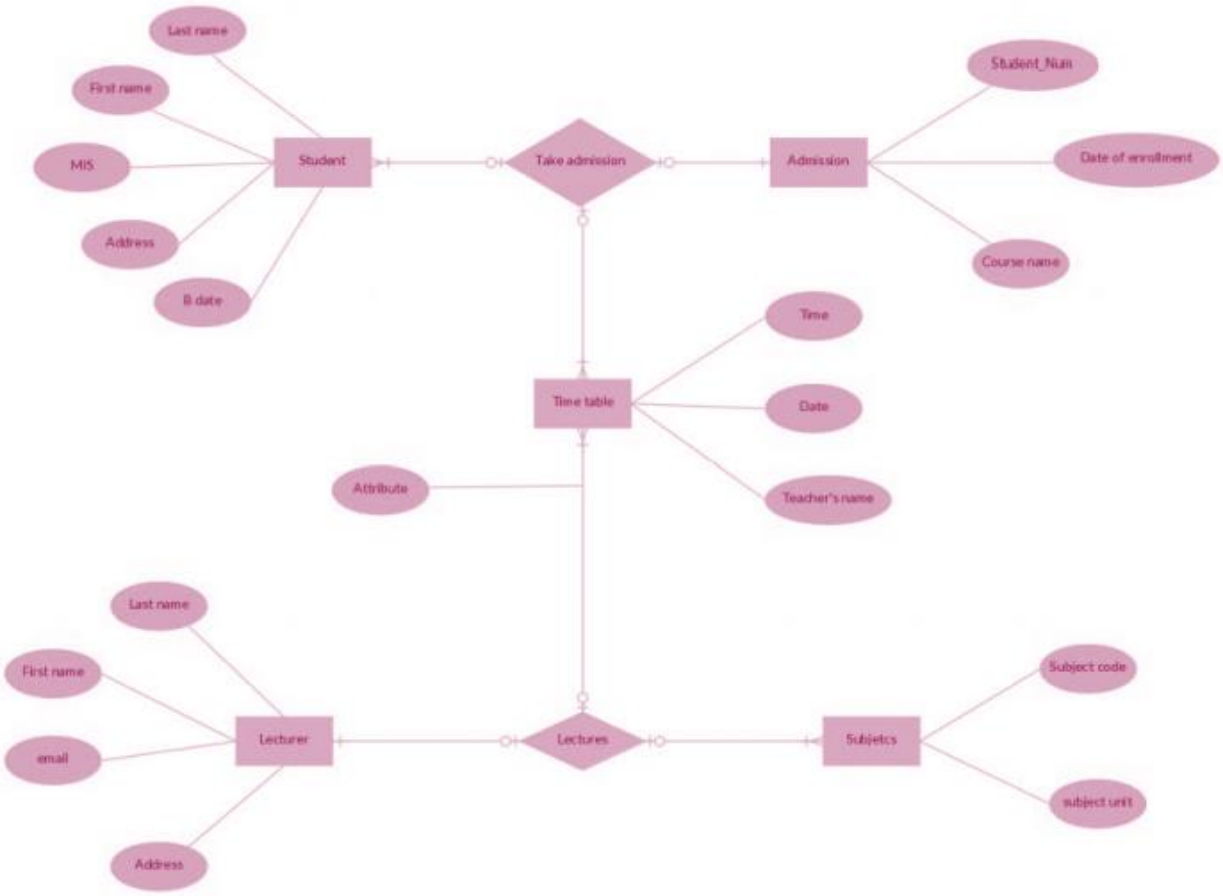
E-R Diagram




contact details
Name
Enrollment No.
DOB
Student
Major
Minor
department Name
department code
Contact details
college
Department
address
age
add line 1
line 2
belongs to
gpa/ cgpa
grades
offer courses
credits
couse Id
subjects
Courses
class
belongs to
Section no.
Year
Semester
instructor


contact details
Name
Enrollment No.
DOB
Student
Major
Minor
department Name
department code
Contact details
college
Department
address
age
add line 1
line 2
belongs to
gpa/ cgpa
grades
offer courses
credits
couse Id
subjects
Courses
class
belongs to
Section no.
Year
Semester
instructor

2. Course Registration System



Write advantages and disadvantages of top 5 databases

MySQL

Advantages –

1. Data Security

MySQL is globally renowned for being the most secure and reliable database management system used in popular web applications like WordPress, Drupal, Joomla, Facebook and Twitter. The data security and support for transactional processing that accompany the recent version of MySQL, can greatly benefit any business especially if it is an eCommerce business that involves frequent money transfers.

2. On-Demand Scalability

MySQL offers unmatched scalability to facilitate the management of deeply embedded apps using a smaller footprint even in massive warehouses that stack terabytes of data. On-demand flexibility is the star feature of MySQL. This open source solution allows complete customization to eCommerce businesses with unique database server requirements.

3. High Performance

MySQL features a distinct storage-engine framework that facilitates system administrators to configure the MySQL database server for a flawless performance. Whether it is an eCommerce website that receives a million queries every single day or a high-speed transactional processing system, MySQL is designed to meet even the most demanding applications while ensuring optimum speed, full-text indexes and unique memory caches for enhanced performance.

4. Round-The-Clock Uptime

MySQL comes with the assurance of 24X7 uptime and offers a wide range of high availability solutions like specialized cluster servers and master/slave replication configurations.

5. Comprehensive Transactional Support

MySQL tops the list of robust transactional database engines available on the market. With features like complete atomic, consistent, isolated, durable transaction support, multi-version transaction support, and unrestricted row-level locking, it is the go-to solution for full data integrity. It guarantees instant deadlock identification through server-enforced referential integrity.

6. Complete Workflow Control

With the average download and installation time being less than 30 minutes, MySQL means usability from day one. Whether your platform is Linux, Microsoft, Macintosh or UNIX, MySQL is a comprehensive solution with self-management features that automate everything from space expansion and configuration to data design and database administration.

7. The Flexibility Of Open Source

All the fears and worries that arise in an open source solution can be brought to an end with My SQL's round-the-clock support and enterprise indemnification. The secure processing and trusted software of MySQL combine to provide effective transactions for large volume projects. It makes maintenance, debugging and upgrades fast and easy while enhancing the end-user experience.

Disadvantages

1. Delivering Hot Data

In large applications, the data cache stored in RAM can grow very large and be subjected to thousands or even millions of requests per second. MySQL does not have a strong memory-focused search engine. Because it is not designed for very high concurrency, users can be exposed to bottlenecks and periodic performance issues. MySQL is saddled with relatively high overhead and cannot deliver optimal speed.

2. Dealing with Highly Volatile Data

In thousands of updates per second are applied to a single database row (for example, flash online sales for high-demand concert tickets), it is crucial to maintain exact values at every second. MySQL is designed around full transactional semantics with support for long transactions and works with disk-based log durability. It is therefore not well suited for use with this kind of highly volatile data.

3. Avoid MySQL Scalability Limitations

MySQL was originally designed as a single-node system and not with the modern data center concept in mind. Today's largest MySQL installations cannot scale by using MySQL as a single system and must rely on sharding, or splitting a data set over multiple nodes or instances. However, most sharding solutions in MySQL are manual and make application code more complex. Any performance gain is lost when queries must access data across multiple shards.

4. Providing Analytics

MySQL was not designed for running complicated queries against massive data volumes which requires crunching through a lot of data on a huge scale. MySQL optimizer is quite limited, executing a single query at a time using a single thread. A

given MySQL query can neither scale among multiple CPU cores in a single system nor execute distributed queries across multiple nodes.

5. Powering Full Text Searches at Scale

MySQL can handle basic full text searches. However, because of its inability to manage parallel processing, searches do not scale well as data volumes increase.

Oracle

Advantages

1.Portability

Well, the oracle database is ported to all different platforms than all other its competition. It easily runs on almost 20 networking protocols and also on more than 100 hardware platforms. The same thing makes it easy for writing an oracle application easily by make changes safely in the operating system and hardware.

2.Backup and Recovery

It is good to be used as to take a proper backup of your entire oracle online backup as well as recovery too. With the help of using oracle database, one can easily become able to make a point-in-time recovery. For the same, you have to require storage space and also archive mechanisms.

3.High performance

It means that making a good oracle database provides you with quite good speed and also with large databases. Also, oracle database improves the performance and speed of consideration with transaction control and locking.

4.Multiple Database Support

The best advantage which users get when they make use of the oracle database is that it easily manages the multiple databases within the same transaction. The same thing is best applicable, or you can say implemented in V7.

So, all these are the best benefits which all the users of oracle database get when they make its use. To gather more information about the same process and to know how to make appropriate use of the oracle database, one should check out more and more reviews. The more reviews you go through when going to make use of the oracle database, the easier it becomes for you to get good results by using the same database.

5.Versions Changes

Oracle keeps you informed about the next major release for any potential changes so you can get prepared. It offers you good backward compatibility by which you will no longer be required to re-write an application while upgrading the DBMS. Many have worked with Oracle since V4 Beta and have never faced any unpleasant experiences in terms of syntax.

Disadvantages

1.Complexity

One of the biggest disadvantages of Oracle Database is its complexity. It is not preferable to use Oracle, especially when the users are not technically sound and lack the technical ability that is needed in order to work with the Oracle Database. Further, it is also not advisable to use Oracle if the company or an individual is seeking a database that is easy to use and contains basic features. It is not as simple as installing Oracle and get started, rather it requires specialized skills to install and maintain as it is an incredibly complex engine.

2.Cost of Oracle Database

The price of Oracle products can increase up to ten times compared to the MS SQL Server Database Solution for a mid-range solution. Hence, people are tended to go with other options that are comparatively cheaper, say, for example, you can install MySQL for free or utilize any one of several engines in a solution like AWS by investing a very nominal amount.

3.Difficult to Manage

Oracle is generally a lot more complex and difficult in terms of the management of certain activities. Hence, the pro tip here is to install a basic version and then perform configuration with minimal customization. Oracle Database is only useful when you need large size databases. The use of Oracle in small or medium-sized companies is not preferable where small databases are needed. In such a scenario, the best option would be MySQL, which is more cost-effective.

PostgreSQL

Advantages

1. PostgreSQL can run dynamic websites and web apps as a LAMP stack option
2. PostgreSQL's write-ahead logging makes it a highly fault-tolerant database

3. PostgreSQL source code is freely available under an open source license. This allows you the freedom to use, modify, and implement it as per your business needs.
4. PostgreSQL supports geographic objects so you can use it for location-based services and geographic information systems
5. PostgreSQL supports geographic objects so it can be used as a geospatial data store for location-based services and geographic information systems
6. To learn Postgres, you don't need much training as its easy to use
7. Low maintenance and administration for both embedded and enterprise use of PostgreSQL

Disadvantages-

1. Postgres is not owned by one organization. So, it has had trouble getting its name out there despite being fully featured and comparable to other DBMS systems
2. Changes made for speed improvement requires more work than MySQL as PostgreSQL focuses on compatibility
3. Many open source apps support MySQL, but may not support PostgreSQL
4. On performance metrics, it is slower than MySQL.

Cassendra

Advantages:

1.Open Source

Cassandra is Apache's open-source project, this means it is available for FREE! Yes, you can download the application and use the way you want. Infact, its open-source nature has given birth to a huge Cassandra community where like-minded people share their views, queries, suggestions related to Big Data. Further, Cassandra can be integrated with other Apache open-source projects like Hadoop (with the help of MapReduce), Apache Pig and Apache Hive.

2. Peer to Peer Architecture:

Cassandra follows a peer-to-peer architecture, instead of master-slave architecture. Hence, there is no single point of failure in Cassandra. Moreover, any number of servers/nodes can be added to any Cassandra cluster in any of the datacenters. As all the machines are at equal level, any server can entertain request from any client.

Undoubtedly, with its robust architecture and exceptional characteristics, Cassandra has raised the bar far above than other databases.

3. Elastic Scalability:

One of the biggest advantages of using Cassandra is its elastic scalability. Cassandra cluster can be easily scaled-up or scaled-down. Interestingly, any number of nodes can be added or deleted in Cassandra cluster without much disturbance. You don't have to restart the cluster or change queries related Cassandra application while scaling up or down. This is why Cassandra is popular of having a very high throughput for the highest number of nodes. As scaling happens, read and write throughput both increase simultaneously with zero downtime or any pause to the applications.

4. High Availability and Fault Tolerance:

Another striking feature of Cassandra is Data replication which makes Cassandra highly available and fault-tolerant. Replication means each data is stored at more than one location. This is because, even if one node fails, the user should be able to retrieve the data with ease from another location. In a Cassandra cluster, each row is replicated based on the row key. You can set the number of replicas you want to create. Just like scaling, data replication can also happen across multiple data centres. This further leads to high level back-up and recovery competencies in Cassandra.

5. *High Performance:*

The basic idea behind developing Cassandra was to harness the hidden capabilities of several multicore machines. Cassandra has made this dream come true! Cassandra has demonstrated brilliant performance under large sets of data. Thus, Cassandra is loved by those organizations that deal with huge amount of data every day and at the same time cannot afford to lose such data.

6. *Column Oriented:*

Cassandra has a very high-level data model – this is column-oriented. It means, Cassandra stores columns based on the column names, leading to very quick slicing. Unlike traditional databases, where column names only consist of metadata, in Cassandra column names can also consist of the actual data. Thus, Cassandra rows can consist of masses of columns, in contrast to a relational database that consists of a few number of columns. Cassandra is endowed with a rich data model.

7. Schema-Free

Since its creation, Cassandra is famous for being a Schema-less/schema-free database in its column family. In Cassandra, columns can be created at your will within the rows. Cassandra data model is also famously known as a schema-optional data model. In contrast to a traditional database, in Cassandra there is no need to show all the columns needed by your application at the surface as each row is not expected to have the same set of columns.

It is because of the above reasons, Cassandra is in great demand among several companies, where MySQL is getting replaced by NoSQL databases. A database that was initially created to solve the inbox search issues at Facebook, has come a long way to solve Big Data problems. Today, Cassandra is used in diverse applications…whether it is streaming videos or supporting various business units or production applications

Disadvantages:

1. Replication means data gets replicated across multiple nodes as you configure. For ex, every record I write I can have it replicated to 2 or 3 or even 10 other nodes. But this also means any bad data also gets replicated. So you have to take care to not do so.
2. Repairs - This is Cassandra specific concept and is not so trivial to understand, let alone master. Most users leave it to the database. (This is when some nodes die but don't come back up within a window that allows other nodes to pass on the data that the node missed.
3. You cannot run unanticipated queries because the data storage on disk or in mem is such that you can't query on any column you want. You will explicitly have to add indexes. This will bite you if you simply assume that you just have to create a table using CQL (Cassandra Query Language) which is modeled on SQL.

MongoDB

Advantages

There are many great features inbuilt with MongoDB. As compared to RDBMS, so let's discuss MongoDB Benefits.

1. Flexible Database

We know that MongoDB is a schema-less database. That means we can have any type of data in a separate document. This thing gives us flexibility and a freedom to store data of different types.

2. Sharding

We can store a large data by distributing it to several servers connected to the application. If a server cannot handle such a big data then there will be no failure condition. The term we can use here is "auto-sharding".

3. High Speed

MongoDB is a document-oriented database. It is easy to access documents by indexing. Hence, it provides fast query response. The speed of MongoDB is 100 times faster than the relational database.

4. High Availability

MongoDB has features like replication and gridFS. These features help to increase data availability in MongoDB. Hence the performance is very high.

5. Scalability

A great advantage of MongoDB is that it is a horizontally scalable database. When you have to handle a large data, you can distribute it to several machines.

Disadvantages -

1. Joins not Supported

MongoDB doesn't support joins like a relational database. Yet one can use joins functionality by adding by coding it manually. But it may slow execution and affect performance.

2. High Memory Usage

MongoDB stores key names for each value pairs. Also, due to no functionality of joins, there is data redundancy. This results in increasing unnecessary usage of memory.

3. Limited Data Size

You can have document size, not more than 16MB.

EXPERIMENT - 2

DATABASE MANAGEMENT SYSTEMS LAB

Aim

Creating Database tables and performing the operation of table creations ,insert data and fetch data.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 2

Aim:

Creating Database tables and performing the operation of table creations, insert data and fetch data.

Tools Used:

MariaDB

Procedure:

Creation of Table:

1. **Table Name:** CLIENT_MASTER
2. **Description:** Used to store Client Information

Commands used for Creating Table:

```
→ CREATE TABLE CLIENT_MASTER (
→ CLIENT_NO CHAR(6),
→ NAME VARCHAR(20),
→ ADDRESS1 VARCHAR(30),
→ ADDRESS2 VARCHAR(30),
→ CITY VARCHAR(15),
→ PINCODE INT(8),
→ STATE VARCHAR(15),
→ BAL_DUE FLOAT(10,2));
```

```
MariaDB [(none)]> use info
Database changed
MariaDB [info]> CREATE TABLE CLIENT_MASTER (
    -> CLIENT_NO CHAR(6),
    -> NAME VARCHAR(20),
    -> ADDRESS1 VARCHAR(30),
    -> ADDRESS2 VARCHAR(30),
    -> CITY VARCHAR(15),
    -> PINCODE INT(8),
    -> STATE VARCHAR(15),
    -> BAL_DUE FLOAT(10,2));
Query OK, 0 rows affected (0.021 sec)
```

Describing Schema of the Table:

Commands used:

→ DESCRIBE `CLIENT_MASTER` or `DESC CLIENT_MASTER`;

```
MariaDB [info]> DESC CLIENT_MASTER;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| CLIENT_NO | char(6)     | YES  |     | NULL    |       |
| NAME      | varchar(20) | YES  |     | NULL    |       |
| ADDRESS1  | varchar(30) | YES  |     | NULL    |       |
| ADDRESS2  | varchar(30) | YES  |     | NULL    |       |
| CITY      | varchar(15) | YES  |     | NULL    |       |
| PINCODE   | int(8)      | YES  |     | NULL    |       |
| STATE     | varchar(15) | YES  |     | NULL    |       |
| BAL_DUE   | float(10,2) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
8 rows in set (0.034 sec)
```

Creation of Table:

1. **Table Name:** PRODUCT_MASTER
2. **Description:** Used to store Product Information of the Client

Commands used for Creating Table:

```
→ CREATE TABLE PRODUCT_MASTER (
→  PRODUCT_NO VARCHAR(6),
→  DESCRIPTION VARCHAR(15),
→ PROFIT_PERCENT FLOAT(4,2),
→ UNIT_MEASURE VARCHAR(10),
→ QTY_ON_HEAD INT(8),
→ REORDER_LVL INT(8),
→ SELL_PRICE FLOAT(8,2),
→ COST_PRICE FLOAT(8,2));
```

```
MariaDB [info]> CREATE TABLE PRODUCT_MASTER (
    -> PRODUCT_NO VARCHAR(6),
    -> DESCRIPTION VARCHAR(15),
    -> PROFIT_PERCENT FLOAT(4,2),
    -> UNIT_MEASURE VARCHAR(10),
    -> QTY_ON_HEAD INT(8),
    -> REORDER_LVL INT(8),
    -> SELL_PRICE FLOAT(8,2),
    -> COST_PRICE FLOAT(8,2));
Query OK, 0 rows affected (0.011 sec)
```

Describing Schema of the Table:

Commands used:

➔ DESCRIBE PRODUCT_MASTER or DESC PRODUCT_MASTER;

```
MariaDB [info]> DESCRIBE PRODUCT_MASTER;
+---------------+------------+------+-----+---------+-------+
| Field         | Type       | Null | Key | Default | Extra |
+---------------+------------+------+-----+---------+-------+
| PRODUCT_NO    | varchar(6) | YES  |     | NULL    |       |
| DESCRIPTION   | varchar(15)| YES  |     | NULL    |       |
| PROFIT_PERCENT| float(4,2) | YES  |     | NULL    |       |
| UNIT_MEASURE  | varchar(10)| YES  |     | NULL    |       |
| QTY_ON_HEAD   | int(8)     | YES  |     | NULL    |       |
| REORDER_LVL   | int(8)     | YES  |     | NULL    |       |
| SELL_PRICE    | float(8,2) | YES  |     | NULL    |       |
| COST_PRICE    | float(8,2) | YES  |     | NULL    |       |
+---------------+------------+------+-----+---------+-------+
8 rows in set (0.029 sec)
```

Creation of Table:

1. **Table Name:** SALESMAN_MASTER
2. **Description:** Used to store Salesman Working Information

Commands for Creating Table:

→ CREATE TABLE SALESMAN_MASTER (

→ SALESMAN_NO VARCHAR(6),

→ SALESMAN_NAME VARCHAR(20),

→ ADDRESS1 VARCHAR(30),

→ ADDRESS2 VARCHAR(30),

→ CITY VARCHAR(20),

→ PINCODE INT(8),

→ STATE VARCHAR(20),

→ SAL_AMT FLOAT(8,2),

→ TGT_TO_GET FLOAT(6,2),

→ YTD_SALES FLOAT(6,2),

→ REMARKS VARCHAR(60));

```
MariaDB [info]> CREATE TABLE SALESMAN_MASTER (
    -> SALESMAN_NO VARCHAR(6),
    -> SALESMAN_NAME VARCHAR(20),
    -> ADDRESS1 VARCHAR(30),
    -> ADDRESS2 VARCHAR(30),
    -> CITY VARCHAR(20),
    -> PINCODE INT(8),
    -> STATE VARCHAR(20),
    -> SAL_AMT FLOAT(8,2),
    -> TGT_TO_GET FLOAT(6,2),
    -> YTD_SALES FLOAT(6,2),
    -> REMARKS VARCHAR(60));
Query OK, 0 rows affected (0.011 sec)
```

Describing Schema of the Table:

Commands used:

➔ DESCRIBE SALESMAN_MASTER or DESC SALESMAN_MASTER;

```
MariaDB [info]> DESCRIBE SALESMAN_MASTER;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| SALESMAN_NO   | varchar(6)  | YES  |     | NULL    |       |
| SALESMAN_NAME | varchar(20) | YES  |     | NULL    |       |
| ADDRESS1      | varchar(30) | YES  |     | NULL    |       |
| ADDRESS2      | varchar(30) | YES  |     | NULL    |       |
| CITY          | varchar(20) | YES  |     | NULL    |       |
| PINCODE       | int(8)      | YES  |     | NULL    |       |
| STATE         | varchar(20) | YES  |     | NULL    |       |
| SAL_AMT       | float(8,2)  | YES  |     | NULL    |       |
| TGT_TO_GET    | float(6,2)  | YES  |     | NULL    |       |
| YTD_SALES     | float(6,2)  | YES  |     | NULL    |       |
| REMARKS       | varchar(60) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
11 rows in set (0.027 sec)
```

Inserting Data in the Tables:

Table Name: CLIENT_MASTER

Commands used:

→ INSERT INTO CLIENT_MASTER VALUES ('C00001', 'Ivan','','', 'Mumbai', '400054', 'Maharashtra','15000');

→ INSERT INTO CLIENT_MASTER VALUES ('C00002', 'Mamta Muzumdar','','','Madras', '780001', 'TamilNadu', '0');

→ INSERT INTO CLIENT_MASTER VALUES ('C00003', 'Chhaya Bankar','','', 'Mumbai', '400057', 'Maharashtra', '5000');

→ INSERT INTO CLIENT_MASTER VALUES ('C00004', 'Ashwini Joshi','','', 'Banglore', '560001', 'Karnataka','0');

→ INSERT INTO CLIENT_MASTER VALUES ('C00005', 'Hansel Colaco','','','Mumbai', '400060', 'Maharashtra','2000');

→ INSERT INTO CLIENT_MASTER VALUES ('C00006', 'Deepak Sharma','','', 'Mangalore', '560050', 'Karnataka', '0');

```
MariaDB [info]> INSERT INTO CLIENT_MASTER VALUES ('C00001', 'Ivan','','', 'Mumbai', '400054', 'Maharashtra', '15000');
Query OK, 1 row affected (0.004 sec)

MariaDB [info]> INSERT INTO CLIENT_MASTER VALUES ('C00002', 'Mamta Muzumdar','','','Madras', '780001', 'Tamil Nadu', '0');
Query OK, 1 row affected (0.003 sec)

MariaDB [info]> INSERT INTO CLIENT_MASTER VALUES ('C00003', 'Chhaya Bankar','','', 'Mumbai', '400057', 'Maharashtra', '5000');

Query OK, 1 row affected (0.001 sec)

MariaDB [info]> INSERT INTO CLIENT_MASTER VALUES ('C00004', 'Ashwini Joshi','','', 'Banglore', '560001', 'Karnataka','0');
Query OK, 1 row affected (0.001 sec)

MariaDB [info]> INSERT INTO CLIENT_MASTER VALUES ('C00005', 'Hansel Colaco','','','Mumbai', '400060', 'Maharashtra','2000');
Query OK, 1 row affected (0.001 sec)

MariaDB [info]> INSERT INTO CLIENT_MASTER VALUES ('C00006', 'Deepak Sharma','','', 'Mangalore', '560050', 'Karnataka', '0');
Query OK, 1 row affected (0.003 sec)
```

Display Table:

SELECT * FROM CLIENT_MASTER;

```
MariaDB [info]> SELECT * FROM CLIENT_MASTER;
+----------+---------------+----------+----------+-----------+---------+------------+----------+
| CLIENT_NO | NAME          | ADDRESS1 | ADDRESS2 | CITY      | PINCODE | STATE      | BAL_DUE  |
+----------+---------------+----------+----------+-----------+---------+------------+----------+
| C00001    | Ivan          |          |          | Mumbai    |  400054 | Maharashtra | 15000.00 |
| C00002    | Mamta Muzumdar |          |          | Madras    |  780001 | Tamil Nadu |     0.00 |
| C00003    | Chhaya Bankar |          |          | Mumbai    |  400057 | Maharashtra |  5000.00 |
| C00004    | Ashwini Joshi |          |          | Banglore  |  560001 | Karnataka  |     0.00 |
| C00005    | Hansel Colaco |          |          | Mumbai    |  400060 | Maharashtra |  2000.00 |
| C00006    | Deepak Sharma |          |          | Mangalore |  560050 | Karnataka  |     0.00 |
+----------+---------------+----------+----------+-----------+---------+------------+----------+
6 rows in set (0.001 sec)
```

Table Name:

PRODUCT_MASTER

→ INSERT INTO product_master VALUES('P00001','T-shirts',5,'Piece',200,50,5350,250);

→ INSERT INTO product_master VALUES('P00345','Shirts',6,'Piece',150,50,500,350);

→ INSERT INTO product_master VALUES('P06734','CottonJeans',5,'Piece',100,20,600,450);

→ INSERT INTO product_master VALUES('P07865','Jeans',5,'Piece',100,20,750,500);

→ INSERT INTO product_master VALUES('P07868','Trousers',2,'Piece',150,50,850,550);

→ INSERT INTO product_master VALUES('P07885','PullOvers',2.5,'Piece',80,30,700,450);

→ INSERT INTO product_master VALUES('P07965','DenimShirts',4,'Piece',100,40,350,250);

→ INSERT INTO product_master VALUES('P07975','LycraTops',5,'Piece',70,30,300,175);

→ INSERT INTO product_master VALUES('P08865','Skirts',5,'piece',75,30,450,300);

```
MariaDB [info]> INSERT INTO product_master VALUES('P00001','T-shirts',5,'Piece',200,50,5350,250);
Query OK, 1 row affected (0.004 sec)

MariaDB [info]> INSERT INTO product_master VALUES('P00345','Shirts',6,'Piece',150,50,500,350);
Query OK, 1 row affected (0.002 sec)

MariaDB [info]> INSERT INTO product_master VALUES('P06734','CottonJeans',5,'Piece',100,20,600,450);
Query OK, 1 row affected (0.002 sec)

MariaDB [info]> INSERT INTO product_master VALUES('P07865','Jeans',5,'Piece',100,20,750,500);
Query OK, 1 row affected (0.002 sec)

MariaDB [info]> INSERT INTO product_master VALUES('P07868','Trousers',2,'Piece',150,50,850,550);
Query OK, 1 row affected (0.001 sec)

MariaDB [info]> INSERT INTO product_master VALUES('P07885','PullOvers',2.5,'Piece',80,30,700,450);
Query OK, 1 row affected (0.003 sec)

MariaDB [info]> INSERT INTO product_master VALUES('P07965','DenimShirts',4,'Piece',100,40,350,250);
Query OK, 1 row affected (0.002 sec)

MariaDB [info]> INSERT INTO product_master VALUES('P07975','LycraTops',5,'Piece',70,30,300,175);
Query OK, 1 row affected (0.001 sec)

MariaDB [info]> INSERT INTO product_master VALUES('P08865','Skirts',5,'piece',75,30,450,300);
Query OK, 1 row affected (0.002 sec)
```

Display Table:

SELECT * FROM PRDOUCT_MASTER;

```
MariaDB [info]> SELECT * FROM PRODUCT_MASTER;
+------------+-------------+---------------+-------------+------------+------------+------------+------------+
| PRODUCT_NO | DESCRIPTION | PROFIT_PERCENT | UNIT_MEASURE | QTY_ON_HEAD | REORDER_LVL | SELL_PRICE | COST_PRICE |
+------------+-------------+---------------+-------------+------------+------------+------------+------------+
| P00001     | T-shirts    |          5.00 | Piece       |        200 |          50 |    5350.00 |     250.00 |
| P00345     | Shirts      |          6.00 | Piece       |        150 |          50 |     500.00 |     350.00 |
| P06734     | CottonJeans |          5.00 | Piece       |        100 |          20 |     600.00 |     450.00 |
| P07865     | Jeans       |          5.00 | Piece       |        100 |          20 |     750.00 |     500.00 |
| P07868     | Trousers    |          2.00 | Piece       |        150 |          50 |     850.00 |     550.00 |
| P07885     | PullOvers   |          2.50 | Piece       |         80 |          30 |     700.00 |     450.00 |
| P07965     | DenimShirts |          4.00 | Piece       |        100 |          40 |     350.00 |     250.00 |
| P07975     | LycraTops   |          5.00 | Piece       |         70 |          30 |     300.00 |     175.00 |
| P08865     | Skirts      |          5.00 | piece       |         75 |          30 |     450.00 |     300.00 |
+------------+-------------+---------------+-------------+------------+------------+------------+------------+
9 rows in set (0.000 sec)
```

Table Name:

SALESMAN_MASTER

→ INSERT INTO SALESMAN_MASTER VALUES ('S00001','Aman','A/14','Worli',
'Mumbai','400002','Maharashtra','3000','100','50','Good');

→ INSERT INTO SALESMAN_MASTER
VALUES('S00002','Omkar','65','Nariman','Mumbai','400002','Maharashtra','3000','0','
100','Good');

→ INSERT INTO SALESMAN_MASTER VALUES ('S00003','Raj','P/7','Bandra',
'Mumbai','400002','Maharashtra','3000','200','100','Good');

→ INSERT INTO SALESMAN_MASTER VALUES ('S00004','Ashish','A/5','Juhu',
'Mumbai','400044','Maharashtra','3500','200','150','Good');

```
MariaDB [info]> INSERT INTO SALESMAN_MASTER VALUES ('S00001','Aman','A/14','Worli', 'Mumbai','400002','Maharashtra','3000','100','50','Good');
Query OK, 1 row affected (0.003 sec)

MariaDB [info]> INSERT INTO SALESMAN_MASTER VALUES('S00002','Omkar','65','Nariman','Mumbai','400002','Maharashtra','3000','0','100','Good');
Query OK, 1 row affected (0.004 sec)

MariaDB [info]> INSERT INTO SALESMAN_MASTER VALUES ('S00003','Raj','P/7','Bandra', 'Mumbai','400002','Maharashtra','3000','200','100','Good');
Query OK, 1 row affected (0.001 sec)

MariaDB [info]> INSERT INTO SALESMAN_MASTER VALUES ('S00004','Ashish','A/5','Juhu', 'Mumbai','400044','Maharashtra','3500','200','150','Good');
Query OK, 1 row affected (0.001 sec)
```

Display Table:

SELECT * FROM SALESMAN_MASTER;

```
MariaDB [info]> SELECT * FROM SALESMAN_MASTER;
+------------+--------------+----------+----------+--------+---------+-------------+---------+-----------+-----------+---------+
| SALESMAN_NO | SALESMAN_NAME | ADDRESS1 | ADDRESS2 | CITY   | PINCODE | STATE       | SAL_AMT | TGT_TO_GET | YTD_SALES | REMARKS |
+------------+--------------+----------+----------+--------+---------+-------------+---------+-----------+-----------+---------+
| S00001     | Aman         | A/14     | Worli    | Mumbai | 400002  | Maharashtra | 3000.00 |    100.00 |     50.00 | Good    |
| S00002     | Omkar        | 65       | Nariman  | Mumbai | 400002  | Maharashtra | 3000.00 |      0.00 |    100.00 | Good    |
| S00003     | Raj          | P/7      | Bandra   | Mumbai | 400002  | Maharashtra | 3000.00 |    200.00 |    100.00 | Good    |
| S00004     | Ashish       | A/5      | Juhu     | Mumbai | 400044  | Maharashtra | 3500.00 |    200.00 |    150.00 | Good    |
+------------+--------------+----------+----------+--------+---------+-------------+---------+-----------+-----------+---------+
4 rows in set (0.000 sec)
```

College:

1) CREATE DATABASE COLLEGE;
2) USE COLLEGE;

```
C:\WINDOWS\System32>mysql -u root -p
Enter password: ****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 20
Server version: 10.5.9-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use college
Database changed
```

3) CREATE TABLE Students (stud_id int,LastName varchar(255),FirstName varchar(255),Address varchar(255),City varchar(255));
4) SHOW TABLES;

```
Database changed
MariaDB [College]> CREATE TABLE Students (stud_id int,LastName varchar(255),FirstName varchar(255),Address varchar(255),City varchar(255));
Query OK, 0 rows affected (0.019 sec)

MariaDB [College]> show tables;
+------------------+
| Tables_in_college |
+------------------+
| students         |
+------------------+
1 row in set (0.003 sec)
```

5) INSERT INTO Students (stud_id,LastName,FirstName,Address,City)VALUES ('1', 'solkjaer', 'Ole gunner', '15 Norway road', 'Norway');
6) INSERT INTO Students (stud_id,LastName,FirstName,Address,City)VALUES ('2', 'fernandes', 'Bruno', 'lisbon street', 'Portugal');
7) SELECT * FROM Students;

```
MariaDB [College]> INSERT INTO Students (stud_id,LastName,FirstName,Address,City)VALUES ('1', 'solkjaer', 'Ole gunner', '15 Norway road', 'Norway');
Query OK, 1 row affected (0.012 sec)

MariaDB [College]> INSERT INTO Students (stud_id,LastName,FirstName,Address,City)VALUES ('2', 'fernandes', 'Bruno', 'lisbon street', 'Portugal');
Query OK, 1 row affected (0.004 sec)

MariaDB [College]> select * from students;
+---------+----------+-----------+---------------+----------+
| stud_id | LastName | FirstName | Address       | City     |
+---------+----------+-----------+---------------+----------+
|       1 | solkjaer | Ole gunner | 15 Norway road | Norway   |
|       2 | fernandes | Bruno     | lisbon street  | Portugal |
+---------+----------+-----------+---------------+----------+
2 rows in set (0.001 sec)

MariaDB [College]>
```

VIVA QUESTIONS:

Que1. What is a NULL value and how does it differ from a zero value?

Ans.

Zero is a number value. It is a definite with precise mathematical properties. (You can do arithmetic on it .

NULL means the absence of any value. You can't do anything with it except test for it.

Que2. What are SQL Constraints?

Ans.

SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

The following constraints are commonly used in SQL:

• NOT NULL- Ensures that a column cannot have a NULL value
• UNIQUE- Ensures that all values in a column are different
• PRIMARY KEY- A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
• FOREIGN KEY- Uniquely identifies a row/record in another table
• CHECK- Ensures that all values in a column satisfies a specific condition
• DEFAULT- Sets a default value for a column when no value is specified
• INDEX- Ued to create and retrieve data from the database very quickly

Que3. What is the difference between CHAR and VARCHAR?

Ans.

CHAR is fixed length and VARCHAR is variable length. CHAR always uses the same amount of storage space per entry, while VARCHAR only uses the amount necessary to store the actual text
Varchar cuts off trailing spaces if the entered characters is shorter than the declared length. Char will pad spaces and will always be the length of the declared length. In in terms of efficiency, varchar is more adept .

Que 4. What is Difference between NUMBER, INTEGER and INT DataTypes?

Ans.

Number allows a decimal component Integer doesn't. If we try to store 3.43 in an

Integer, it will just store 3.Number allows for much larger values than Integer does.

EXPERIMENT - 3

DATABASE MANAGEMENT SYSTEMS LAB

Aim

Write queries for retrieving records from table using SELECT command and WHERE clause.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 3

Aim:

Write queries for retrieving records from table using SELECT command and WHERE clause.

Tools Used:

MariaDB

Procedure:

Find out the names of all the Clients

→ `SELECT NAME FROM CLIENT_MASTER;`

```
MariaDB [info]> SELECT NAME FROM CLIENT_MASTER;
+-----------------+
| NAME            |
+-----------------+
| Ivan            |
| Mamta Muzumdar  |
| Chhaya Bankar   |
| Ashwini Joshi   |
| Hansel Colaco   |
| Deepak Sharma   |
+-----------------+
6 rows in set (0.002 sec)
```

Retrieve the entire contents of the Client_Master table

→ `SELECT * FROM CLIENT_MASTER;`

```
MariaDB [info]> SELECT * FROM CLIENT_MASTER;
+----------+--------------+---------+---------+----------+---------+------------+---------+
| CLIENT_NO | NAME         | ADDRESS1 | ADDRESS2 | CITY     | PINCODE | STATE      | BAL_DUE |
+----------+--------------+---------+---------+----------+---------+------------+---------+
| C00001   | Ivan         |         |         | Mumbai   |  400054 | Maharashtra | 15000.00 |
| C00002   | Mamta Muzumdar |         |         | Madras   |  780001 | Tamil Nadu |     0.00 |
| C00003   | Chhaya Bankar |         |         | Mumbai   |  400057 | Maharashtra |  5000.00 |
| C00004   | Ashwini Joshi |         |         | Banglore |  560001 | Karnataka  |     0.00 |
| C00005   | Hansel Colaco |         |         | Mumbai   |  400060 | Maharashtra |  2000.00 |
| C00006   | Deepak Sharma |         |         | Mangalore |  560050 | Karnataka  |     0.00 |
+----------+--------------+---------+---------+----------+---------+------------+---------+
6 rows in set (0.000 sec)
```

Retrieve the list of names, cities and the states of all the clients

→ SELECT NAME, CITY, STATE FROM CLIENT_MASTER;

```
MariaDB [info]> SELECT NAME, CITY, STATE FROM CLIENT_MASTER;
+---------------+----------+-------------+
| NAME          | CITY     | STATE       |
+---------------+----------+-------------+
| Ivan          | Mumbai   | Maharashtra |
| Mamta Muzumdar | Madras  | Tamil Nadu  |
| Chhaya Bankar | Mumbai   | Maharashtra |
| Ashwini Joshi | Banglore | Karnataka   |
| Hansel Colaco | Mumbai   | Maharashtra |
| Deepak Sharma | Mangalore | Karnataka  |
+---------------+----------+-------------+
6 rows in set (0.000 sec)
```

Find names of clients from Client_Master whose Bal_Due is Rs.0

→ SELECT NAME FROM CLIENT_MASTER where BAL_DUE = ‘0’;

```
MariaDB [info]> SELECT NAME FROM CLIENT_MASTER where BAL_DUE = '0';
+---------------+
| NAME          |
+---------------+
| Mamta Muzumdar |
| Ashwini Joshi |
| Deepak Sharma |
+---------------+
3 rows in set (0.007 sec)
```

List all the clients who are located in Mumbai.

→ SELECT NAME FROM CLIENT_MASTER where CITY = ‘Mumbai’;

```
MariaDB [info]> SELECT NAME FROM CLIENT_MASTER where CITY = 'Mumbai';
+---------------+
| NAME          |
+---------------+
| Ivan          |
| Chhaya Bankar |
| Hansel Colaco |
+---------------+
3 rows in set (0.003 sec)
```

List the various products available from the Product_Mastertable

→ SELECT DESCRIPTION FROM PRODUCT_MASTER;

```
MariaDB [info]> SELECT DESCRIPTION FROM PRODUCT_MASTER;
+-------------+
| DESCRIPTION |
+-------------+
| T-shirts    |
| Shirts      |
| CottonJeans |
| Jeans       |
| Trousers    |
| PullOvers   |
| DenimShirts |
| LycraTops   |
| Skirts      |
+-------------+
9 rows in set (0.000 sec)
```

Find the names of salesman who have Sal_Amt = Rs.3000

→ SELECT SALESMAN_NAME FROM SALESMAN_MASTER where SAL_AMT = ‘3000’;

```
MariaDB [info]> SELECT SALESMAN_NAME FROM SALESMAN_MASTER where SAL_AMT = '3000';
+--------------+
| SALESMAN_NAME |
+--------------+
| Aman          |
| Omkar         |
| Raj           |
+--------------+
3 rows in set (0.000 sec)
```

List all Salesman_No who live in city Mumbai

→ SELECT SALESMAN_NO FROM SALESMAN_MASTER where CITY = ‘Mumbai’;

```
MariaDB [info]> SELECT SALESMAN_NO FROM SALESMAN_MASTER where CITY = 'Mumbai';
+-------------+
| SALESMAN_NO |
+-------------+
| S00001      |
| S00002      |
| S00003      |
| S00004      |
+-------------+
4 rows in set (0.000 sec)
```

Find Product_No whose Price = Rs.150

→ SELECT DESCRIPTION FROM PRODUCT_MASTER WHERE COST_PRICE = '150' or SELL_PRICE = '150';

```
MariaDB [info]> SELECT DESCRIPTION FROM PRODUCT_MASTER WHERE COST_PRICE = '150' or SELL_PRICE = '150';
Empty set (0.000 sec)
```

Find Product_No of T-Shirts in Product_Master table

→ SELECT PRODUCT_NO FROM PRODUCT_MASTER where DESCRIPTION = 'T-Shirts';

```
MariaDB [info]> SELECT PRODUCT_NO FROM PRODUCT_MASTER where DESCRIPTION = 'T-Shirts';
+------------+
| PRODUCT_NO |
+------------+
| P00001     |
+------------+
1 row in set (0.000 sec)
```

VIVA QUESTIONS:

Que1. What is the use of SELECT command?

Ans.

The SELECT Statement in SQL is used to retrieve or fetch data from a database. We can fetch either the entire table or according to some specified rules. The data returned is stored in a result table. This result table is also called result-set.

Que2. What is the purpose of WHERE clause?

Ans.

The SQL WHERE clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table.

Ques3. Which language supports SELECT command?

Ans.

In most applications, SELECT is the most commonly used data manipulation language (DML) command. As SQL is a declarative programming language, SELECT queries specify a result set, but do not specify how to calculate it.

Que4. How to select entire content of a table?

Ans.

SELECT * FROM table_name;

The star will return all the records available in the specified table

EXPERIMENT - 4

DATABASE MANAGEMENT SYSTEMS LAB

Aim

Write SQL commands for implementing ALTER, UPDATE and DELETE.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 4

Aim:

Write SQL commands for implementing ALTER, UPDATE and DELETE.

Tools Used:

MariaDB

Procedure:

Change the City of Client_No 'C00005' to 'Bangalore'

```
UPDATE CLIENT_MASTER
→ SET CITY = ‘Bangalore’
→ WHERE CLIENT_NO = ‘C00005’;
```

```
MariaDB [info]> UPDATE CLIENT_MASTER
    -> SET CITY = 'Bangalore'
    -> WHERE CLIENT_NO = 'C00005';
Query OK, 1 row affected (0.004 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Altered Data is: -

```
MariaDB [info]> SELECT * FROM CLIENT_MASTER;
+-----------+---------------+----------+----------+-----------+---------+-------------+----------+
| CLIENT_NO | NAME          | ADDRESS1 | ADDRESS2 | CITY      | PINCODE | STATE       | BAL_DUE  |
+-----------+---------------+----------+----------+-----------+---------+-------------+----------+
| C00001    | Ivan          |          |          | Mumbai    |  400054 | Maharashtra | 15000.00 |
| C00002    | Mamta Muzumdar |          |          | Madras    |  780001 | Tamil Nadu  |     0.00 |
| C00003    | Chhaya Bankar |          |          | Mumbai    |  400057 | Maharashtra |  5000.00 |
| C00004    | Ashwini Joshi |          |          | Bangalore |  560001 | Karnataka   |     0.00 |
| C00005    | Hansel Colaco |          |          | Bangalore |  400060 | Maharashtra |  2000.00 |
| C00006    | Deepak Sharma |          |          | Mangalore |  560050 | Karnataka   |     0.00 |
+-----------+---------------+----------+----------+-----------+---------+-------------+----------+
6 rows in set (0.000 sec)
```

Change the Bal_Due of Client_No 'C00001' to Rs.1000

```
→ UPDATE CLIENT_MASTER
→ SET BAL_DUE = ‘1000’
→ WHERE CLIENT_NO = ‘C00001’;
```

```
MariaDB [info]> UPDATE CLIENT_MASTER
    -> SET BAL_DUE = '1000'
    -> WHERE CLIENT_NO = 'C00001';
Query OK, 1 row affected (0.004 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Altered Data is: -

```
MariaDB [info]> SELECT * FROM CLIENT_MASTER;
+-----------+---------------+----------+----------+-----------+---------+-------------+---------+
| CLIENT_NO | NAME          | ADDRESS1 | ADDRESS2 | CITY      | PINCODE | STATE       | BAL_DUE |
+-----------+---------------+----------+----------+-----------+---------+-------------+---------+
| C00001    | Ivan          |          |          | Mumbai    |  400054 | Maharashtra | 1000.00 |
| C00002    | Mamta Muzumdar |          |          | Madras    |  780001 | Tamil Nadu  |    0.00 |
| C00003    | Chhaya Bankar |          |          | Mumbai    |  400057 | Maharashtra | 5000.00 |
| C00004    | Ashwini Joshi |          |          | Bangalore |  560001 | Karnataka   |    0.00 |
| C00005    | Hansel Colaco |          |          | Bangalore |  400060 | Maharashtra | 2000.00 |
| C00006    | Deepak Sharma |          |          | Mangalore |  560050 | Karnataka   |    0.00 |
+-----------+---------------+----------+----------+-----------+---------+-------------+---------+
6 rows in set (0.000 sec)
```

Change the Cost_Price of 'Trousers' to Rs.950

```
→ UPDATE PRODUCT_MASTER
→ SET COST_PRICE = ‘950
→ WHERE DESCRIPTION = ‘Trousers’;
```

```
MariaDB [info]> UPDATE PRODUCT_MASTER
    -> SET COST_PRICE = '950'
    -> WHERE DESCRIPTION = 'Trousers';
Query OK, 1 row affected (0.003 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
MariaDB [info]> SELECT * FROM PRODUCT_MASTER;
+------------+-------------+---------------+-------------+-------------+-------------+------------+------------+
| PRODUCT_NO | DESCRIPTION | PROFIT_PERCENT | UNIT_MEASURE | QTY_ON_HEAD | REORDER_LVL | SELL_PRICE | COST_PRICE |
+------------+-------------+---------------+-------------+-------------+-------------+------------+------------+
| P00001     | T-shirts    |          5.00 | Piece       |         200 |          50 |    5350.00 |     250.00 |
| P00345     | Shirts      |          6.00 | Piece       |         150 |          50 |     500.00 |     350.00 |
| P06734     | CottonJeans |          5.00 | Piece       |         100 |          20 |     600.00 |     450.00 |
| P07865     | Jeans       |          5.00 | Piece       |         100 |          20 |     750.00 |     500.00 |
| P07868     | Trousers    |          2.00 | Piece       |         150 |          50 |     850.00 |     950.00 |
| P07885     | PullOvers   |          2.50 | Piece       |          80 |          30 |     700.00 |     450.00 |
| P07965     | DenimShirts |          4.00 | Piece       |         100 |          40 |     350.00 |     250.00 |
| P07975     | LycraTops   |          5.00 | Piece       |          70 |          30 |     300.00 |     175.00 |
| P08865     | Skirts      |          5.00 | piece       |          75 |          30 |     450.00 |     300.00 |
+------------+-------------+---------------+-------------+-------------+-------------+------------+------------+
9 rows in set (0.001 sec)
```

Change the City of the Salesman to 'Pune'

```
UPDATE SALESMAN_MASTER
➔ SET CITY = ‘Pune’;
```

```
MariaDB [info]> UPDATE SALESMAN_MASTER
    -> SET CITY = 'Pune';
Query OK, 4 rows affected (0.003 sec)
Rows matched: 4  Changed: 4  Warnings: 0
```

Altered Data is: -

```
MariaDB [info]> SELECT * FROM SALESMAN_MASTER;
+------------+--------------+----------+----------+------+---------+-------------+---------+-----------+-----------+---------+
| SALESMAN_NO | SALESMAN_NAME | ADDRESS1 | ADDRESS2 | CITY | PINCODE | STATE       | SAL_AMT | TGT_TO_GET | YTD_SALES | REMARKS |
+------------+--------------+----------+----------+------+---------+-------------+---------+-----------+-----------+---------+
| S00001     | Aman         | A/14     | Worli    | Pune | 400002  | Maharashtra | 3000.00 |     100.00 |     50.00 | Good    |
| S00002     | Omkar        | 65       | Nariman  | Pune | 400002  | Maharashtra | 3000.00 |       0.00 |    100.00 | Good    |
| S00003     | Raj          | P/7      | Bandra   | Pune | 400002  | Maharashtra | 3000.00 |     200.00 |    100.00 | Good    |
| S00004     | Ashish       | A/5      | Juhu     | Pune | 400044  | Maharashtra | 3500.00 |     200.00 |    150.00 | Good    |
+------------+--------------+----------+----------+------+---------+-------------+---------+-----------+-----------+---------+
4 rows in set (0.000 sec)
```

Delete all Salesman from the Salesman_Master whosesalaries are equal to Rs. 3500

➔ DELETE FROM SALESMAN_MASTER
➔ WHERE SAL_AMT = ‘3500’;

```
MariaDB [info]> DELETE FROM SALESMAN_MASTER
    -> WHERE SAL_AMT = '3500';
Query OK, 1 row affected (0.003 sec)
```

Altered Data is: -

```
MariaDB [info]> SELECT * FROM SALESMAN_MASTER;
+------------+--------------+---------+---------+------+--------+-------------+--------+-----------+-----------+---------+
| SALESMAN_NO | SALESMAN_NAME | ADDRESS1 | ADDRESS2 | CITY | PINCODE | STATE       | SAL_AMT | TGT_TO_GET | YTD_SALES | REMARKS |
+------------+--------------+---------+---------+------+--------+-------------+--------+-----------+-----------+---------+
| S00001     | Aman         | A/14    | Worli    | Pune | 400002 | Maharashtra | 3000.00 |    100.00 |     50.00 | Good    |
| S00002     | Omkar        | 65      | Nariman  | Pune | 400002 | Maharashtra | 3000.00 |      0.00 |    100.00 | Good    |
| S00003     | Raj          | P/7     | Bandra   | Pune | 400002 | Maharashtra | 3000.00 |    200.00 |    100.00 | Good    |
+------------+--------------+---------+---------+------+--------+-------------+--------+-----------+-----------+---------+
3 rows in set (0.001 sec)
```

Delete all Products from Product_Master where the QtyOnHand = 100

➔ DELETE FROM PRODUCT_MASTER
➔ WHERE QTY_ON_HAND = ‘100’;

Altered Data is: -

```
MariaDB [info]> SELECT * FROM CLIENT_MASTER;
+-----------+---------------+----------+----------+-----------+---------+-------------+---------+
| CLIENT_NO | NAME          | ADDRESS1 | ADDRESS2 | CITY      | PINCODE | STATE       | BAL_DUE |
+-----------+---------------+----------+----------+-----------+---------+-------------+---------+
| C00001    | Ivan          |          |          | Mumbai    |  400054 | Maharashtra | 1000.00 |
| C00002    | Mamta Muzumdar |          |          | Madras    |  780001 | Tamil Nadu  |    0.00 |
| C00003    | Chhaya Bankar |          |          | Mumbai    |  400057 | Maharashtra | 5000.00 |
| C00004    | Ashwini Joshi |          |          | Bangalore |  560001 | Karnataka   |    0.00 |
| C00005    | Hansel Colaco |          |          | Bangalore |  400060 | Maharashtra | 2000.00 |
| C00006    | Deepak Sharma |          |          | Mangalore |  560050 | Karnataka   |    0.00 |
+-----------+---------------+----------+----------+-----------+---------+-------------+---------+
6 rows in set (0.000 sec)
```

Delete from Client_Master where the column state holdsthe value 'Tamil Nadu'

➔ DELETE FROM CLIENT_MASTER
➔ WHERE STATE = ‘Tamil Nadu’;

```
MariaDB [info]> SELECT * FROM CLIENT_MASTER;
+-----------+----------------+----------+----------+-----------+---------+-------------+---------+
| CLIENT_NO | NAME           | ADDRESS1 | ADDRESS2 | CITY      | PINCODE | STATE       | BAL_DUE |
+-----------+----------------+----------+----------+-----------+---------+-------------+---------+
| C00001    | Ivan           |          |          | Mumbai    |  400054 | Maharashtra | 1000.00 |
| C00002    | Mamta Muzumdar |          |          | Madras    |  780001 | Tamil Nadu  |    0.00 |
| C00003    | Chhaya Bankar  |          |          | Mumbai    |  400057 | Maharashtra | 5000.00 |
| C00004    | Ashwini Joshi  |          |          | Bangalore |  560001 | Karnataka   |    0.00 |
| C00005    | Hansel Colaco  |          |          | Bangalore |  400060 | Maharashtra | 2000.00 |
| C00006    | Deepak Sharma  |          |          | Mangalore |  560050 | Karnataka   |    0.00 |
+-----------+----------------+----------+----------+-----------+---------+-------------+---------+
6 rows in set (0.000 sec)

MariaDB [info]> DELETE FROM CLIENT_MASTER
    -> WHERE STATE = 'Tamil Nadu';
Query OK, 1 row affected (0.003 sec)

MariaDB [info]> SELECT * FROM CLIENT_MASTER;
+-----------+---------------+----------+----------+-----------+---------+-------------+---------+
| CLIENT_NO | NAME          | ADDRESS1 | ADDRESS2 | CITY      | PINCODE | STATE       | BAL_DUE |
+-----------+---------------+----------+----------+-----------+---------+-------------+---------+
| C00001    | Ivan          |          |          | Mumbai    |  400054 | Maharashtra | 1000.00 |
| C00003    | Chhaya Bankar |          |          | Mumbai    |  400057 | Maharashtra | 5000.00 |
| C00004    | Ashwini Joshi |          |          | Bangalore |  560001 | Karnataka   |    0.00 |
| C00005    | Hansel Colaco |          |          | Bangalore |  400060 | Maharashtra | 2000.00 |
| C00006    | Deepak Sharma |          |          | Mangalore |  560050 | Karnataka   |    0.00 |
+-----------+---------------+----------+----------+-----------+---------+-------------+---------+
5 rows in set (0.000 sec)
```

Add a column called 'Telephone' of data type 'Number'and size = '10' to the Client_Master Table

➔ ALTER TABLE CLIENT_MASTER
➔ ADD TELEPHONE INT(10);

```
MariaDB [info]> ALTER TABLE CLIENT_MASTER
    -> ADD TELEPHONE INT(10);
Query OK, 0 rows affected (0.007 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [info]> SELECT * FROM CLIENT_MASTER;
+-----------+---------------+----------+----------+-----------+---------+-------------+---------+-----------+
| CLIENT_NO | NAME          | ADDRESS1 | ADDRESS2 | CITY      | PINCODE | STATE       | BAL_DUE | TELEPHONE |
+-----------+---------------+----------+----------+-----------+---------+-------------+---------+-----------+
| C00001    | Ivan          |          |          | Mumbai    |  400054 | Maharashtra | 1000.00 |      NULL |
| C00003    | Chhaya Bankar |          |          | Mumbai    |  400057 | Maharashtra | 5000.00 |      NULL |
| C00004    | Ashwini Joshi |          |          | Bangalore |  560001 | Karnataka   |    0.00 |      NULL |
| C00005    | Hansel Colaco |          |          | Bangalore |  400060 | Maharashtra | 2000.00 |      NULL |
| C00006    | Deepak Sharma |          |          | Mangalore |  560050 | Karnataka   |    0.00 |      NULL |
+-----------+---------------+----------+----------+-----------+---------+-------------+---------+-----------+
5 rows in set (0.007 sec)
```

Change the size of Sell_Price column in Product_Master to'10.2'

➔ ALTER TABLE PRODUCT_MASTER
➔ MODIFY SELL_PRICE FLOAT(10,2);

```
MariaDB [info]> ALTER TABLE PRODUCT_MASTER
    -> MODIFY SELL_PRICE float(10,2);
Query OK, 0 rows affected (0.005 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [info]> SELECT * FROM PROSUCT_MASTER;
ERROR 1146 (42S02): Table 'info.prosuct_master' doesn't exist
MariaDB [info]> SELECT * FROM PRODUCT_MASTER;
+------------+-------------+---------------+-------------+-------------+-------------+------------+------------+
| PRODUCT_NO | DESCRIPTION | PROFIT_PERCENT | UNIT_MEASURE | QTY_ON_HAND | REORDER_LVL | SELL_PRICE | COST_PRICE |
+------------+-------------+---------------+-------------+-------------+-------------+------------+------------+
| P00001     | T-shirts    |          5.00 | Piece       |         200 |          50 |    5350.00 |     250.00 |
| P0345      | Shirts      |          6.00 | Piece       |         150 |          50 |     500.00 |     350.00 |
| P07868     | Trousers    |          2.00 | Piece       |         150 |          50 |     850.00 |     950.00 |
| P07885     | PullOvers   |          2.50 | Piece       |          80 |          30 |     700.00 |     450.00 |
| P07975     | LycraTops   |          5.00 | Piece       |          70 |          30 |     300.00 |     175.00 |
| P08865     | Skirts      |          5.00 | piece       |          75 |          30 |     450.00 |     300.00 |
+------------+-------------+---------------+-------------+-------------+-------------+------------+------------+
6 rows in set (0.007 sec)
```

Destroy the table Client_Master along with its data

➔ DROP TABLE CLIENT_MASTER;

```
MariaDB [info]> DROP TABLE CLIENT_MASTER
    -> ;
Query OK, 0 rows affected (0.013 sec)

MariaDB [info]> DESCRIBE CLIENT_MASTER;
ERROR 1146 (42S02): Table 'info.client_master' doesn't exist
```

Change the name of the Salesman_Master table toSMAN_MAST

→ RENAME TABLE SALESMAN_MASTER TO SMAN_MAST;

```
MariaDB [info]> RENAME TABLE SALESMAN_MASTER TO SMAN_MAST;
Query OK, 0 rows affected (0.010 sec)

MariaDB [info]> SELECT * FROM SMAN_MAST;
+------------+--------------+---------+---------+------+--------+-------------+---------+-----------+-----------+---------+
| SALESMAN_NO | SALESMAN_NAME | ADDRESS1 | ADDRESS2 | CITY | PINCODE | STATE       | SAL_AMT | TGT_TO_GET | YTD_SALES | REMARKS |
+------------+--------------+---------+---------+------+--------+-------------+---------+-----------+-----------+---------+
| S00001     | Aman         | A/14    | Worli   | Pune |  400002 | Maharashtra | 3000.00 |    100.00 |     50.00 | Good    |
| S00002     | Omkar        | 65      | Nariman | Pune |  400002 | Maharashtra | 3000.00 |      0.00 |    100.00 | Good    |
| S00003     | Raj          | P/7     | Bandra  | Pune |  400002 | Maharashtra | 3000.00 |    200.00 |    100.00 | Good    |
+------------+--------------+---------+---------+------+--------+-------------+---------+-----------+-----------+---------+
3 rows in set (0.001 sec)
```

Office DB

1. CREATE DATABASE Office;
2. Use Office;
3. CREATE TABLE Employee(id int NOT NULL,Name varchar(255),Role varchar(255),PRIMARY KEY (id));
4. INSERT INTO Employee(id,Name,Role)VALUES ('1','Elliot alderson','Consulting Hacker');
5. INSERT INTO Employee(id,Name,Role)VALUES ('2','saksham pathak','Systems Arch');
6. Select * FROM Employee;

```
C:\WINDOWS\System32>mysql -u root -p
Enter password: ****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 26
Server version: 10.5.9-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE Office;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> use Office;
Database changed
MariaDB [Office]> CREATE TABLE Employee(id int NOT NULL,Name varchar(255),Role varchar(255),PRIMARY KEY (id));
Query OK, 0 rows affected (0.019 sec)

MariaDB [Office]> INSERT INTO Employee(id,Name,Role)VALUES ('1','Elliot alderson','Consulting Hacker');
Query OK, 1 row affected (0.004 sec)

MariaDB [Office]> INSERT INTO Employee(id,Name,Role)VALUES ('1','saksham pathak','Systems Arch');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
MariaDB [Office]> INSERT INTO Employee(id,Name,Role)VALUES ('2','saksham pathak','Systems Arch');
Query OK, 1 row affected (0.003 sec)

MariaDB [Office]> select * FROM Employee;
+----+-----------------+------------------+
| id | Name            | Role             |
+----+-----------------+------------------+
|  1 | Elliot alderson | Consulting Hacker |
|  2 | saksham pathak  | Systems Arch     |
+----+-----------------+------------------+
2 rows in set (0.001 sec)
```

7. ALTER TABLE Employee ADD COLUMN Salary int NOT NULL;
8. Select * FROM Employee;

```
MariaDB [Office]> select * FROM Employee;
+----+----------------+------------------+
| id | Name           | Role             |
+----+----------------+------------------+
|  1 | Elliot alderson | Consulting Hacker |
|  2 | saksham pathak  | Systems Arch      |
+----+----------------+------------------+
2 rows in set (0.001 sec)

MariaDB [Office]> ALTER TABLE Employee ADD COLUMN Salary int NOT NULL;
Query OK, 0 rows affected (0.007 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [Office]> select * FROM Employee;
+----+----------------+------------------+--------+
| id | Name           | Role             | Salary |
+----+----------------+------------------+--------+
|  1 | Elliot alderson | Consulting Hacker |      0 |
|  2 | saksham pathak  | Systems Arch      |      0 |
+----+----------------+------------------+--------+
2 rows in set (0.002 sec)

MariaDB [Office]>
```

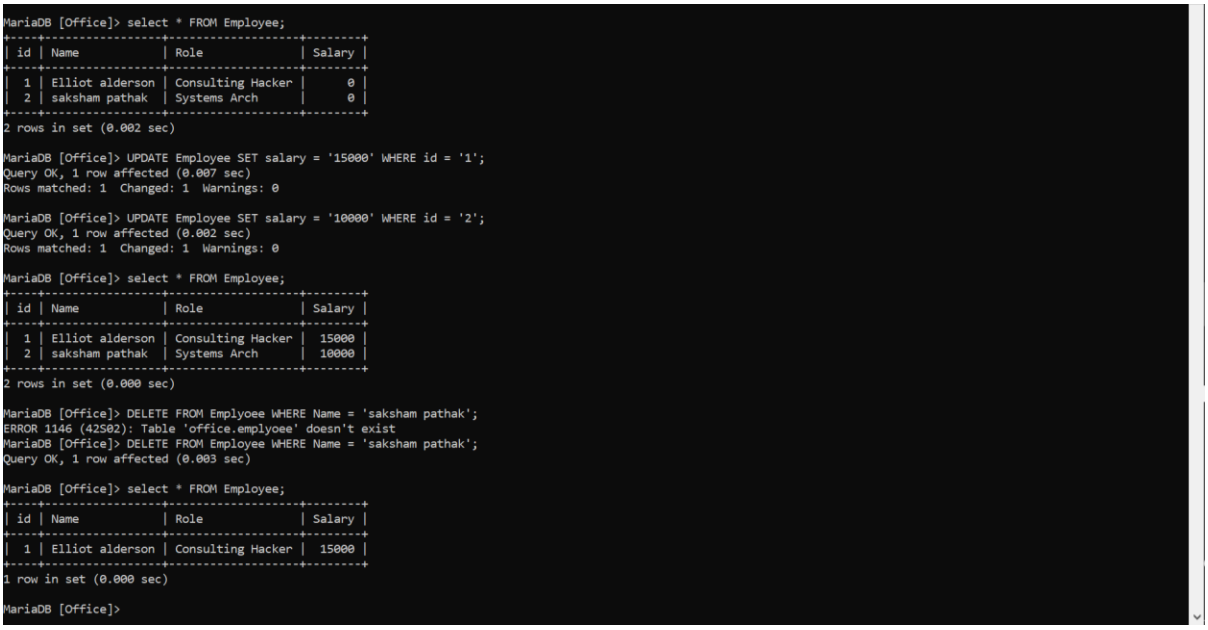
9. UPDATE Employee SET salary = '15000' WHERE id = '1';
10. UPDATE Employee SET salary = '10000' WHERE id = '2';
11. Select * FROM Employee;

```
MariaDB [Office]> UPDATE Employee SET salary = '15000' WHERE id = '1';
Query OK, 1 row affected (0.007 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [Office]> UPDATE Employee SET salary = '10000' WHERE id = '2';
Query OK, 1 row affected (0.002 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [Office]> select * FROM Employee;
+----+----------------+------------------+--------+
| id | Name           | Role             | Salary |
+----+----------------+------------------+--------+
|  1 | Elliot alderson | Consulting Hacker |  15000 |
|  2 | saksham pathak  | Systems Arch      |  10000 |
+----+----------------+------------------+--------+
2 rows in set (0.000 sec)
```

12. DELETE FROM Employee WHERE Name = 'saksham pathak';
13. Select * FROM Employee;
14.



```
MariaDB [Office]> select * FROM Employee;
+----+----------------+------------------+--------+
| id | Name           | Role             | Salary |
+----+----------------+------------------+--------+
|  1 | Elliot alderson | Consulting Hacker |      0 |
|  2 | saksham pathak  | Systems Arch      |      0 |
+----+----------------+------------------+--------+
2 rows in set (0.002 sec)

MariaDB [Office]> UPDATE Employee SET salary = '15000' WHERE id = '1';
Query OK, 1 row affected (0.007 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [Office]> UPDATE Employee SET salary = '10000' WHERE id = '2';
Query OK, 1 row affected (0.002 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [Office]> select * FROM Employee;
+----+----------------+------------------+--------+
| id | Name           | Role             | Salary |
+----+----------------+------------------+--------+
|  1 | Elliot alderson | Consulting Hacker |  15000 |
|  2 | saksham pathak  | Systems Arch      |  10000 |
+----+----------------+------------------+--------+
2 rows in set (0.000 sec)

MariaDB [Office]> DELETE FROM Emplyoee WHERE Name = 'saksham pathak';
ERROR 1146 (42S02): Table 'office.emplyoee' doesn't exist
MariaDB [Office]> DELETE FROM Employee WHERE Name = 'saksham pathak';
Query OK, 1 row affected (0.003 sec)

MariaDB [Office]> select * FROM Employee;
+----+----------------+------------------+--------+
| id | Name           | Role             | Salary |
+----+----------------+------------------+--------+
|  1 | Elliot alderson | Consulting Hacker |  15000 |
+----+----------------+------------------+--------+
1 row in set (0.000 sec)

MariaDB [Office]>
```

VIVA QUESTIONS:

Que1. What are different DML commands?

Ans.

DML is short name of Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

- SELECT- retrieve data from a database
- INSERT- insert data into a table
- UPDATE- updates existing data within a table
- DELETE- Delete all records from a database table
- MERGE - UPSERT operation (insert or update)
- CALL - call a PL/SQL or Java subprogram
- EXPLAIN PLAN - interpretation of the data access path
- LOCK TABLE - concurrency Control

Que2. What is the purpose of ALTER command .What is the Syntax?

Ans.

The ALTER TABLE statement is used to add, delete, or modify columns in an existing

table. The ALTER TABLE statement is also used to add and drop various constraints on an

existing table.

ALTER TABLE table_name

ADD column_name datatype;

Que3. What is the purpose of DELETE command?

Ans.

The DELETE Statement in SQL is used to delete existing records from a table. We can

delete a single record or multiple records depending on the condition we specify in the WHERE clause.

Basic Syntax:

DELETE FROM table_name WHERE some_condition;

Que4. What is difference between ALTER and UPDATE commands?

Ans.

The point that distinguishes both ALTER and UPDATE Command is that ALTER command is Data Definition Language (DDL). On the other hands, the UPDATE Command is a Data Manipulation Language (DML).

- ALTER Command add, delete, modify, rename the attributes of the relation whereas, the UPDATE Command modifies the values of the records in the relations.
- ALTER Command by default set values of all the tuples or record as NULL. On the other hands, the UPDATE Command set the value specified in the command to the tuples of the relation.
- ALTER command is attribute or column specific whereas, the UPDATE command is attribute value specific.

EXPERIMENT - 5

DATABASE MANAGEMENT SYSTEMS LAB

Aim

Write the queries to implement the concept of Integrity Constraints like Primary Key, Foreign Key, NOT NULL to the tables.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 5

Aim:

Write the queries to implement the concept of Integrity Constraints like Primary Key, Foreign Key, NOT NULL to the tables.

Tools Used:

MariaDB

Procedure:

Creation of Table:

1. **Table Name:** CLIENT_MASTER2
2. **Description:** Used to store Client Information

Commands used for Creating Table:

```
→ CREATE TABLE CLIENT_MASTER2 (
→ CLIENT_NO CHAR(6) PRIMARY KEY,
→ NAME VARCHAR(20) NOT NULL,
→ ADDRESS1 VARCHAR(30),
→ ADDRESS2 VARCHAR(30),
→ CITY VARCHAR(15),
→ PINCODE INT(8),
→ STATE VARCHAR(15),
→ BAL_DUE FLOAT(10,2));
```

```
MariaDB [info]> CREATE TABLE CLIENT_MASTER2 (
    -> CLIENT_NO CHAR(6) PRIMARY KEY,
    -> NAME VARCHAR(20) NOT NULL,
    -> ADDRESS1 VARCHAR(30),
    -> ADDRESS2 VARCHAR(30),
    -> CITY VARCHAR(15),
    -> PINCODE INT(8),
    -> STATE VARCHAR(15),
    -> BAL_DUE FLOAT(10,2));
Query OK, 0 rows affected (0.013 sec)
```

Describing Schema of the Table:

Commands used:

→ DESCRIBE CLIENT_MASTER2 or DESC CLIENT_MASTER2;

```
MariaDB [info]> DESCRIBE CLIENT_MASTER2;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| CLIENT_NO | char(6)     | NO   | PRI | NULL    |       |
| NAME      | varchar(20) | NO   |     | NULL    |       |
| ADDRESS1  | varchar(30) | YES  |     | NULL    |       |
| ADDRESS2  | varchar(30) | YES  |     | NULL    |       |
| CITY      | varchar(15) | YES  |     | NULL    |       |
| PINCODE   | int(8)      | YES  |     | NULL    |       |
| STATE     | varchar(15) | YES  |     | NULL    |       |
| BAL_DUE   | float(10,2) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
8 rows in set (0.016 sec)
```

Inserting Data

insert into CLIENT_MASTER values('C00001', 'Aman', 'A/14', 'Worli', 'Mumbai', 400002, 'Maharashtra', 30000);

insert into CLIENT_MASTER values('C00002', 'Omkar', '65', 'Nariman', 'Mumbai', 400001, 'Maharashtra', 8000);

insert into CLIENT_MASTER values('C00003', 'Raj', 'P-7', 'Bandra', 'Mumbai', 400032, 'Maharashtra', 12000);

insert into CLIENT_MASTER values('C00004', 'Ashi', 'A/9', 'Juhu', 'Mumbai', 400044, 'Maharashtra', 0);

insert into CLIENT_MASTER values('C00005', 'Ashish', 'A/5', 'Juhu', 'Mumbai', 400044, 'Maharashtra', 3500);

insert into CLIENT_MASTER values('C00006', 'Ashutosh', 'F/5', 'Andheri', 'Mumbai', 400044, 'Maharashtra', 0);

```
MariaDB [labdb2]> insert into CLIENT_MASTER values('C00001', 'Aman', 'A/14', 'Worli', 'Mumbai', 400002, 'Maharashtra', 30000);
Query OK, 1 row affected (0.056 sec)

MariaDB [labdb2]> insert into CLIENT_MASTER values('C00002', 'Omkar', '65', 'Nariman', 'Mumbai', 400001, 'Maharashtra', 8000);
Query OK, 1 row affected (0.024 sec)

MariaDB [labdb2]> insert into CLIENT_MASTER values('C00003', 'Raj', 'P-7', 'Bandra', 'Mumbai', 400032, 'Maharashtra', 12000);
Query OK, 1 row affected (0.030 sec)

MariaDB [labdb2]> insert into CLIENT_MASTER values('C00004', 'Ashi', 'A/9', 'Juhu', 'Mumbai', 400044, 'Maharashtra', 0);
Query OK, 1 row affected (0.026 sec)

MariaDB [labdb2]> insert into CLIENT_MASTER values('C00005', 'Ashish', 'A/5', 'Juhu', 'Mumbai', 400044, 'Maharashtra', 3500);
Query OK, 1 row affected (0.023 sec)

MariaDB [labdb2]> insert into CLIENT_MASTER values('C00006', 'Ashutosh', 'F/5', 'Andheri', 'Mumbai', 400044, 'Maharashtra', 0);
Query OK, 1 row affected (0.020 sec)

MariaDB [labdb2]> select * from client_master;
+----------+----------+----------+----------+--------+---------+-------------+----------+
| clientno | name     | address1 | address2 | city   | pincode | state       | bal_due  |
+----------+----------+----------+----------+--------+---------+-------------+----------+
| C00001   | Aman     | A/14     | Worli    | Mumbai |  400002 | Maharashtra | 30000.00 |
| C00002   | Omkar    | 65       | Nariman  | Mumbai |  400001 | Maharashtra |  8000.00 |
| C00003   | Raj      | P-7      | Bandra   | Mumbai |  400032 | Maharashtra | 12000.00 |
| C00004   | Ashi     | A/9      | Juhu     | Mumbai |  400044 | Maharashtra |     0.00 |
| C00005   | Ashish   | A/5      | Juhu     | Mumbai |  400044 | Maharashtra |  3500.00 |
| C00006   | Ashutosh | F/5      | Andheri  | Mumbai |  400044 | Maharashtra |     0.00 |
+----------+----------+----------+----------+--------+---------+-------------+----------+
6 rows in set (0.013 sec)
```

Creation of Table:

1. **Table Name:** PRODUCT_MASTER2
2. **Description:** Used to store Prdouct Information

Commands for Creating Table:

```
→ CREATE TABLE PRODUCT_MASTER2(
→ PRODUCT_NO VARCHAR(6) PRIMARY KEY,
→ DESCRIPTION VARCHAR(15) NOT NULL,
→ PROFIT_PERCENT FLOAT(4,2) NOT NULL,
→ UNIT_MEASURE VARCHAR(10) NOT NULL,
→ QTY_ON_HEAD INT(8) NOT NULL,
→ REORDER_LVL INT(8) NOT NULL,
→ SELL_PRICE FLOAT(8,2) NOT NULL,
→ COST_PRICE FLOAT(8,2) NOT NULL);
```

```
MariaDB [info]> CREATE TABLE PRODUCT_MASTER2(
    -> PRODUCT_NO VARCHAR(6) PRIMARY KEY,
    -> DESCRIPTION VARCHAR(15) NOT NULL,
    -> PROFIT_PERCENT FLOAT(4,2) NOT NULL,
    -> UNIT_MEASURE VARCHAR(10) NOT NULL,
    -> QTY_ON_HAND INT(8) NOT NULL,
    -> REORDER_LVL INT(8) NOT NULL,
    -> SELL_PRICE FLOAT(8,2) NOT NULL,
    -> COST_PRICE FLOAT(8,2) NOT NULL);
Query OK, 0 rows affected (0.013 sec)
```

Describing Schema of the Table:

Commands used:

→ DESCRIBE PRODUCT_MASTER2 or DESC PRODUCT_MASTER2;

```
MariaDB [info]> DESC PRODUCT_MASTER2;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| PRODUCT_NO     | varchar(6)  | NO   | PRI | NULL    |       |
| DESCRIPTION    | varchar(15) | NO   |     | NULL    |       |
| PROFIT_PERCENT | float(4,2)  | NO   |     | NULL    |       |
| UNIT_MEASURE   | varchar(10) | NO   |     | NULL    |       |
| QTY_ON_HAND    | int(8)      | NO   |     | NULL    |       |
| REORDER_LVL    | int(8)      | NO   |     | NULL    |       |
| SELL_PRICE     | float(8,2)  | NO   |     | NULL    |       |
| COST_PRICE     | float(8,2)  | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
8 rows in set (0.011 sec)
```

Inserting Data

insert into product_master values('P00001','T-Shirts',5,'Piece',200,50,5350,250);

insert into product_master values('P0345','Shirts',6,'Piece',150,50,500,350);

insert into product_master values('P07868','Trousers',2,'Piece',150,50,850,550);

insert into product_master values('P07865','Jeans',5,'Piece',100,20,750,500);

insert into product_master values('P07965','Denim Shirts',4,'Piece',100,40,350,250);

insert into product_master values('P07885','Pull Overs',2.5,'Piece',80,30,700,450);

insert into product_master values('P08865','Skirts',5,'Piece',75,30,450,300);

insert into product_master values('P06734','Cotton Jeans',5,'Piece',100,20,600,450);

insert into product_master values('P07975','Lycra Tops',5,'Piece',70,30,300,175);

```
MariaDB [labdb2]>
MariaDB [labdb2]> insert into product_master values('P00001','T-Shirts',5,'Piece',200,50,5350,250);
Query OK, 1 row affected (1.730 sec)

MariaDB [labdb2]> insert into product_master values('P0345','Shirts',6,'Piece',150,50,500,350);
Query OK, 1 row affected (0.764 sec)

MariaDB [labdb2]> insert into product_master values('P07868','Trousers',2,'Piece',150,50,850,550);
Query OK, 1 row affected (0.075 sec)

MariaDB [labdb2]> insert into product_master values('P07865','Jeans',5,'Piece',100,20,750,500);
Query OK, 1 row affected (0.032 sec)

MariaDB [labdb2]> insert into product_master values('P07965','Denim Shirts',4,'Piece',100,40,350,250);
Query OK, 1 row affected (0.026 sec)

MariaDB [labdb2]> insert into product_master values('P07885','Pull Overs',2.5,'Piece',80,30,700,450);
Query OK, 1 row affected (0.080 sec)

MariaDB [labdb2]> insert into product_master values('P08865','Skirts',5,'Piece',75,30,450,300);
Query OK, 1 row affected (0.042 sec)

MariaDB [labdb2]> insert into product_master values('P06734','Cotton Jeans',5,'Piece',100,20,600,450);
Query OK, 1 row affected (0.023 sec)

MariaDB [labdb2]> insert into product_master values('P07975','Lycra Tops',5,'Piece',70,30,300,175);
Query OK, 1 row affected (1.176 sec)

MariaDB [labdb2]> select * from product_master;
+------------+-------------+----------------+-------------+-----------+------------+------------+------------+
| Production | Description | Profit_Percent | UnitMeasure | QTYONHAND | ReorderLvl | Sell_Price | Cost_Price |
+------------+-------------+----------------+-------------+-----------+------------+------------+------------+
| P00001     | T-Shirts    |              5 | Piece       |       200 |         50 |       5350 |        250 |
| P0345      | Shirts      |              6 | Piece       |       150 |         50 |        500 |        350 |
| P06734     | Cotton Jeans |              5 | Piece       |       100 |         20 |        600 |        450 |
| P07865     | Jeans       |              5 | Piece       |       100 |         20 |        750 |        500 |
| P07868     | Trousers    |              2 | Piece       |       150 |         50 |        850 |        550 |
| P07885     | Pull Overs  |              3 | Piece       |        80 |         30 |        700 |        450 |
| P07965     | Denim Shirts |              4 | Piece       |       100 |         40 |        350 |        250 |
| P07975     | Lycra Tops  |              5 | Piece       |        70 |         30 |        300 |        175 |
| P08865     | Skirts      |              5 | Piece       |        75 |         30 |        450 |        300 |
+------------+-------------+----------------+-------------+-----------+------------+------------+------------+
9 rows in set (0.123 sec)
```

Creation of Table:

1) **Table Name:** SALESMAN_MASTER2
2) **Description:** Used to store Salesman Information

Commands for Creating Table:

```
→ CREATE TABLE SALESMAN_MASTER (
→ SALESMAN_NO VARCHAR(6) PRIMARY KEY,
→ SALESMAN_NAME VARCHAR(20) NOT NULL,
→ ADDRESS1 VARCHAR(30) NOT NULL,
→ ADDRESS2 VARCHAR(30),
→ CITY VARCHAR(20),
→ PINCODE INT(8),
→ STATE VARCHAR(20),
→ SAL_AMT FLOAT(8,2) NOT NULL,
→ TGT_TO_GET FLOAT(6,2) NOT NULL,
→ YTD_SALES FLOAT(6,2) NOT NULL,
→ REMARKS VARCHAR(60),
→ CONSTRAINT CK_SALESMAN_NO CHECK(SALESMAN_NO LIKE "S%"),
→ CONSTRAINT CK_SAL_AMT CHECK(SAL_AMT != 0),
→ CONSTRAINT CK_TGT_TO_GET CHECK(TGT_TO_GET != 0));
```

```
MariaDB [info]> CREATE TABLE SALESMAN_MASTER2(
    -> SALESMAN_NO VARCHAR(6) PRIMARY KEY,
    -> SALESMAN_NAME VARCHAR(20) NOT NULL,
    -> ADDRESS1 VARCHAR(30) NOT NULL,
    -> ADDRESS2 VARCHAR(30),
    -> CITY VARCHAR(20),
    -> PINCODE INT(8),
    -> STATE VARCHAR(20),
    -> SAL_AMT FLOAT(8,2) NOT NULL,
    -> TGT_TO_GET FLOAT(6,2) NOT NULL,
    -> YTD_SALES FLOAT(6,2) NOT NULL,
    -> REMARKS VARCHAR(60),
    -> CONSTRAINT CK_SALESMAN_NO CHECK(SALESMAN_NO LIKE "S%"),
    -> CONSTRAINT CK_SAL_AMT CHECK(SAL_AMT != 0),
    -> CONSTRAINT CK_TGT_TO_GET CHECK(TGT_TO_GET != 0));
Query OK, 0 rows affected (0.014 sec)
```

Schema of the Table:

Commands used:

- ➔ DESCRIBE SALESMAN_MASTER2 or DESC SALESMAN_MASTER2;

```
MariaDB [info]> DESC SALESMAN_MASTER2;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| SALESMAN_NO   | varchar(6)  | NO   | PRI | NULL    |       |
| SALESMAN_NAME | varchar(20) | NO   |     | NULL    |       |
| ADDRESS1      | varchar(30) | NO   |     | NULL    |       |
| ADDRESS2      | varchar(30) | YES  |     | NULL    |       |
| CITY          | varchar(20) | YES  |     | NULL    |       |
| PINCODE       | int(8)      | YES  |     | NULL    |       |
| STATE         | varchar(20) | YES  |     | NULL    |       |
| SAL_AMT       | float(8,2)  | NO   |     | NULL    |       |
| TGT_TO_GET    | float(6,2)  | NO   |     | NULL    |       |
| YTD_SALES     | float(6,2)  | NO   |     | NULL    |       |
| REMARKS       | varchar(60) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
11 rows in set (0.026 sec)
```

Inserting Data

insert into SALESMAN_MASTER values('S00001', 'Kiran', 'A/14', 'Worli', 'Mumbai', 400002, 'Maharashtra', 30000, 100, 50, 'Good');

insert into SALESMAN_MASTER values('S00002', 'Manish', '65', 'Nariman', 'Mumbai', 400001, 'Maharashtra', 3000, 200, 100, 'Good');

insert into SALESMAN_MASTER values('S00003', 'Ravi', 'P-7', 'Bandra', 'Mumbai', 400032, 'Maharashtra', 3000, 200, 100, 'Good');

insert into SALESMAN_MASTER values('S00004', 'Ashish', 'A/9', 'Juhu', 'Mumbai', 400044, 'Maharashtra', 3000, 200, 150, 'Good');

```
MariaDB [labdb2]> insert into SALESMAN_MASTER values('S00001', 'Kiran', 'A/14', 'Worli', 'Mumbai', 400002, 'Maharashtra', 30000, 100, 50, 'Good');
Query OK, 1 row affected (0.037 sec)

MariaDB [labdb2]> insert into SALESMAN_MASTER values('S00002', 'Manish', '65', 'Nariman', 'Mumbai', 400001, 'Maharashtra', 3000, 200, 100, 'Good');
Query OK, 1 row affected (0.019 sec)

MariaDB [labdb2]> insert into SALESMAN_MASTER values('S00003', 'Ravi', 'P-7', 'Bandra', 'Mumbai', 400032, 'Maharashtra', 3000, 200, 100, 'Good');
Query OK, 1 row affected (0.076 sec)

MariaDB [labdb2]> insert into SALESMAN_MASTER values('S00004', 'Ashish', 'A/9', 'Juhu', 'Mumbai', 400044, 'Maharashtra', 3000, 200, 150, 'Good');
Query OK, 1 row affected (0.057 sec)

MariaDB [labdb2]> select * from salesman_master;
+------------+-------------+----------+----------+--------+---------+-------------+--------+----------+----------+---------+
| SALESMANNO | SALESMANNAME | ADDRESS1 | ADDRESS2 | CITY   | PINCODE | STATE       | SALAMT | TGTTOGET | YTDSALES | REMARKS |
+------------+-------------+----------+----------+--------+---------+-------------+--------+----------+----------+---------+
| S00001     | Kiran       | A/14     | Worli    | Mumbai |  400002 | Maharashtra |  30000 |      100 |       50 | Good    |
| S00002     | Manish      | 65       | Nariman  | Mumbai |  400001 | Maharashtra |   3000 |      200 |      100 | Good    |
| S00003     | Ravi        | P-7      | Bandra   | Mumbai |  400032 | Maharashtra |   3000 |      200 |      100 | Good    |
| S00004     | Ashish      | A/9      | Juhu     | Mumbai |  400044 | Maharashtra |   3000 |      200 |      150 | Good    |
+------------+-------------+----------+----------+--------+---------+-------------+--------+----------+----------+---------+
4 rows in set (0.001 sec)
```

```
MariaDB [labdb2]> CREATE TABLE SALESMAN_MASTER (
    ->     SALESMANNO varchar(6) primary key,
    ->     SALESMANNAME varchar(20),
    ->     ADDRESS1 varchar(20),
    ->     ADDRESS2 varchar(20),
    ->     CITY varchar(20),
    ->     PINCODE int(8),
    ->     STATE varchar(20),
    ->     SALAMT int,
    ->     TGTTOGET int,
    ->     YTDSALES int,
    ->     REMARKS varchar(20)
    -> );
Query OK, 0 rows affected (0.229 sec)

MariaDB [labdb2]> describe salesman_master;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| SALESMANNO   | varchar(6)  | NO   | PRI | NULL    |       |
| SALESMANNAME | varchar(20) | YES  |     | NULL    |       |
| ADDRESS1     | varchar(20) | YES  |     | NULL    |       |
| ADDRESS2     | varchar(20) | YES  |     | NULL    |       |
| CITY         | varchar(20) | YES  |     | NULL    |       |
| PINCODE      | int(8)      | YES  |     | NULL    |       |
| STATE        | varchar(20) | YES  |     | NULL    |       |
| SALAMT       | int(11)     | YES  |     | NULL    |       |
| TGTTOGET     | int(11)     | YES  |     | NULL    |       |
| YTDSALES     | int(11)     | YES  |     | NULL    |       |
| REMARKS      | varchar(20) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
11 rows in set (0.027 sec)
```

Creation of Table:

1) **Table Name:** SALES_ORDER
2) **Description:** Used to store Sales Order Information

Commands for Creating Table:

```
→ CREATE TABLE SALES_ORDER(
→ ORDER_NO CHAR(6) PRIMARY KEY,
→ CLIENT_NO CHAR(6)REFERENCES CLIENT_MASTER2,
→ ORDERDATE DATE,
→ SALESMAN_NO CHAR(6) REFERENCES SALESMAN_MASTER2,
→ DELIVTYPE CHAR(1) DEFAULT 'F',
→ BILLYN CHAR(1),
→ DELIVDATE DATE,
→ ORDERSTATUS VARCHAR(10),
→ CONSTRAINT CK_DELIVTYPE CHECK(DELIVTYPE IN('P','F')),
→ CONSTRAINT CK_DELIVDATE CHECK(DELIVDATE>ORDERDATE),
→ CONSTRAINT CK_ORDERSTATUS CHECK(ORDERSTATUS IN('In
  Process','Fulfilled','Backorder','Cancelled')));
```

```
MariaDB [info]> CREATE TABLE SALES_ORDER(
    -> ORDER_NO CHAR(6) PRIMARY KEY,
    -> CLIENT_NO CHAR(6)REFERENCES CLIENT_MASTER2,
    -> ORDERDATE DATE,
    -> SALESMAN_NO CHAR(6) REFERENCES SALESMAN_MASTER2,
    -> DELIVTYPE CHAR(1) DEFAULT 'F',
    -> BILLYN CHAR(1),
    -> DELIVDATE DATE,
    -> ORDERSTATUS VARCHAR(10),
    -> CONSTRAINT CK_DELIVTYPE CHECK(DELIVTYPE IN('P','F')),
    -> CONSTRAINT CK_DELIVDATE CHECK(DELIVDATE>ORDERDATE),
    -> CONSTRAINT CK_ORDERSTATUS CHECK(ORDERSTATUS IN('In Process','Fulfilled','Backorder','Cancelled')));
Query OK, 0 rows affected (0.018 sec)
```

Describing Schema of the Table:

→ DESCRIBE SALES_ORDER or DESC SALES_ORDER;

```
MariaDB [info]> DESC SALES_ORDER;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| ORDER_NO    | char(6)     | NO   | PRI | NULL    |       |
| CLIENT_NO   | char(6)     | YES  | MUL | NULL    |       |
| ORDERDATE   | date        | YES  |     | NULL    |       |
| SALESMAN_NO | char(6)     | YES  | MUL | NULL    |       |
| DELIVTYPE   | char(1)     | YES  |     | F       |       |
| BILLYN      | char(1)     | YES  |     | NULL    |       |
| DELIVDATE   | date        | YES  |     | NULL    |       |
| ORDERSTATUS | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
8 rows in set (0.011 sec)
```

Inserting Data

insert into sales_order values('O19001','C00001', '12-01-10', 'S00001', 'F', 'N', '20-01-10', 'In process');

insert into sales_order values('O19002','C00002', '25-01-10', 'S00002', 'P', 'N', '27-01-10', 'Cancelled');

insert into sales_order values('O46865','C00003', '18-02-10', 'S00003', 'F', 'Y', '20-02-10', 'Fulfilled');

insert into sales_order values('O19003','C00004', '03-04-10', 'S00001', 'F', 'Y', '07-04-10', 'Fulfilled');

insert into sales_order values('O46866','C00005', '20-05-10', 'S00002', 'P', 'N', '22-05-10', 'Cancelled');

insert into sales_order values('O19008','C00006', '24-05-10', 'S00004', 'F', 'N', '26-05-10', 'In process');

```
MariaDB [labdb2]> insert into sales_order values('O19001','C00001',  '20-01-10', 'S00001', 'F', 'N', '20-01-10', 'In process');
ERROR 4025 (23000): CONSTRAINT `ck_delivdate` failed for `labdb2`.`sales_order`
MariaDB [labdb2]> insert into sales_order values('O19002','C00002',  '25-01-10', 'S00002', 'P', 'N', '27-01-10', 'Cancelled');
Query OK, 1 row affected (0.036 sec)

MariaDB [labdb2]> insert into sales_order values('O46865','C00003',  '18-02-10', 'S00003', 'F', 'Y', '20-02-10', 'Fulfilled');
Query OK, 1 row affected (0.019 sec)

MariaDB [labdb2]> insert into sales_order values('O19003','C00004',  '03-04-10', 'S00001', 'F', 'Y', '07-04-10', 'Fulfilled');
Query OK, 1 row affected (0.037 sec)

MariaDB [labdb2]> insert into sales_order values('O46866','C00005',  '20-05-10', 'S00002', 'P', 'N', '22-05-10', 'Cancelled');
Query OK, 1 row affected (0.019 sec)

MariaDB [labdb2]> insert into sales_order values('O19008','C00006',  '24-05-10', 'S00004', 'F', 'N', '26-05-10', 'In process');
Query OK, 1 row affected (0.011 sec)

MariaDB [labdb2]> insert into sales_order values('O19001','C00001',  '12-01-10', 'S00001', 'F', 'N', '20-01-10', 'In process');
Query OK, 1 row affected (0.038 sec)

MariaDB [labdb2]> select * from sales_order;
+----------+-----------+------------+-------------+-----------+--------+------------+-------------+
| order_no | client_no | orderdate  | salesman_no | delivtype | billyn | delivdate  | orderstatus |
+----------+-----------+------------+-------------+-----------+--------+------------+-------------+
| O19001   | C00001    | 2012-01-10 | S00001      | F         | N      | 2020-01-10 | In process  |
| O19002   | C00002    | 2025-01-10 | S00002      | P         | N      | 2027-01-10 | Cancelled   |
| O19003   | C00004    | 2003-04-10 | S00001      | F         | Y      | 2007-04-10 | Fulfilled   |
| O19008   | C00006    | 2024-05-10 | S00004      | F         | N      | 2026-05-10 | In process  |
| O46865   | C00003    | 2018-02-10 | S00003      | F         | Y      | 2020-02-10 | Fulfilled   |
| O46866   | C00005    | 2020-05-10 | S00002      | P         | N      | 2022-05-10 | Cancelled   |
+----------+-----------+------------+-------------+-----------+--------+------------+-------------+
6 rows in set (0.001 sec)
```

Creation of Table:

1) **Table Name:** SALES_ORDER_DETAILS
2) **Description:** Used to store Client's Orders with details of each product.

Commands for Creating Table:

```
→ CREATE TABLE SALES_ORDER_DETAILS(
→ ORDER_NO CHAR(6) REFERENCES SALES_ORDER,
→ PRODUCT_NO CHAR(6) REFERENCES PRODUCT_MASTER2,
→ QTY_ORDERED INT,
→ QTY_DISP INT,
→ PRODUCT_RATE FLOAT(10,2));
```

```
MariaDB [info]> CREATE TABLE SALES_ORDER_DETAILS(
    -> ORDER_NO CHAR(6) REFERENCES SALES_ORDER,
    -> PRODUCT_NO CHAR(6) REFERENCES PRODUCT_MASTER2,
    -> QTY_ORDERED INT,
    -> QTY_DISP INT,
    -> PRODUCT_RATE FLOAT(10,2));
Query OK, 0 rows affected (0.015 sec)
```

Describing Schema of the Table:

Commands used:

→ DESCRIBE SALES_ORDER_DETAILS or DESC SALES_ORDER_DETAILS;

```
MariaDB [info]> DESC SALES_ORDER_DETAILS;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ORDER_NO     | char(6)     | YES  | MUL | NULL    |       |
| PRODUCT_NO   | char(6)     | YES  | MUL | NULL    |       |
| QTY_ORDERED  | int(11)     | YES  |     | NULL    |       |
| QTY_DISP     | int(11)     | YES  |     | NULL    |       |
| PRODUCT_RATE | float(10,2) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.016 sec)
```

Inserting Data

```
insert into sales_order_details values('O19001','P00001', 4, 4, 525);
insert into sales_order_details values('O19001','P07965', 2, 1, 8400);
insert into sales_order_details values('O19001','P07885', 2, 1, 5250);
insert into sales_order_details values('O19002','P00001', 10, 0, 525);
insert into sales_order_details values('O46865','P07868', 3, 3, 3150);
insert into sales_order_details values('O46865','P07885', 3, 1, 5250);
insert into sales_order_details values('O46865','P00001', 10, 10, 525);
insert into sales_order_details values('O46865','P03453', 4, 4, 1050);
insert into sales_order_details values('O19003','P03453', 2, 2, 1050);
insert into sales_order_details values('O19003','P06734', 1, 1, 12000);
insert into sales_order_details values('O04686','P07965', 1, 0, 8400);
insert into sales_order_details values('O04686','P07975', 1, 0, 1050);
insert into sales_order_details values('O19008','P00001', 10, 5, 525);
insert into sales_order_details values('O19008','P07975', 5, 3, 1050);
```

```
MariaDB [labdb2]> insert into sales_order_details values('O19001','P00001', 4, 4, 525);
Query OK, 1 row affected (0.037 sec)

MariaDB [labdb2]> insert into sales_order_details values('O19001','P07965', 2, 1, 8400);
Query OK, 1 row affected (0.019 sec)

MariaDB [labdb2]> insert into sales_order_details values('O19001','P07885', 2, 1, 5250);
Query OK, 1 row affected (0.037 sec)

MariaDB [labdb2]> insert into sales_order_details values('O19002','P00001', 10, 0, 525);
Query OK, 1 row affected (0.021 sec)

MariaDB [labdb2]> insert into sales_order_details values('O46865','P07868', 3, 3, 3150);
Query OK, 1 row affected (0.032 sec)

MariaDB [labdb2]> insert into sales_order_details values('O46865','P07885', 3, 1, 5250);
Query OK, 1 row affected (0.021 sec)

MariaDB [labdb2]> insert into sales_order_details values('O46865','P00001', 10, 10, 525);
Query OK, 1 row affected (0.019 sec)

MariaDB [labdb2]> insert into sales_order_details values('O46865','P03453', 4, 4, 1050);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`lab
Y (`productno`) REFERENCES `product_master` (`Production`))
MariaDB [labdb2]> insert into sales_order_details values('O19003','P03453', 2, 2, 1050);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`lab
Y (`productno`) REFERENCES `product_master` (`Production`))
MariaDB [labdb2]> insert into sales_order_details values('O19003','P06734', 1, 1, 12000);
Query OK, 1 row affected (0.011 sec)

MariaDB [labdb2]> insert into sales_order_details values('O04686','P07965', 1, 0, 8400);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`lab
Y (`orderno`) REFERENCES `sales_order` (`order_no`))
MariaDB [labdb2]> insert into sales_order_details values('O04686','P07975', 1, 0, 1050);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`lab
Y (`orderno`) REFERENCES `sales_order` (`order_no`))
MariaDB [labdb2]> insert into sales_order_details values('O19008','P00001', 10, 5, 525);
Query OK, 1 row affected (0.014 sec)
```

```
MariaDB [labdb2]> select * from sales_order_details;
+---------+-----------+------------+---------+-------------+
| orderno | productno | qtyordered | qtydisp | productrate |
+---------+-----------+------------+---------+-------------+
| O19001  | P00001    |          4 |       4 |      525.00 |
| O19001  | P07965    |          2 |       1 |     8400.00 |
| O19001  | P07885    |          2 |       1 |     5250.00 |
| O19002  | P00001    |         10 |       0 |      525.00 |
| O46865  | P07868    |          3 |       3 |     3150.00 |
| O46865  | P07885    |          3 |       1 |     5250.00 |
| O46865  | P00001    |         10 |      10 |      525.00 |
| O19003  | P06734    |          1 |       1 |    12000.00 |
| O19008  | P00001    |         10 |       5 |      525.00 |
| O19008  | P07975    |          5 |       3 |     1050.00 |
+---------+-----------+------------+---------+-------------+
10 rows in set (0.001 sec)
```

Database already created named college;

1) use College;
2) CREATE TABLE Teachers (teach_id int NOT NULL,LastName varchar(255),FirstName varchar(255),Address varchar(255),City varchar(255),PRIMARY KEY(teach_id);
3) INSERT INTO Teachers (teach_id,LastName,FirstName,Address,City)VALUES ('1', 'ferguson', 'sir alex', 'platama', 'Scotland');
4) INSERT INTO Teachers (teach_id,LastName,FirstName,Address,City)VALUES ('2', 'mourinhio', 'jose', 'porto', 'Portugal');

5) show tables;
6) SELECT * FROM TABLES;

```
MariaDB [college]> show tables;
+------------------+
| Tables_in_college |
+------------------+
| students          |
| teachers          |
+------------------+
2 rows in set (0.001 sec)

MariaDB [college]> select * FROM Teachers;
+----------+-----------+-----------+---------+----------+
| teach_id | LastName  | FirstName | Address | City     |
+----------+-----------+-----------+---------+----------+
|        1 | ferguson  | sir alex  | platama | Scotland |
|        2 | mourinhio | jose      | porto   | Portugal |
+----------+-----------+-----------+---------+----------+
2 rows in set (0.000 sec)
```

7) SHOW KEYS FROM Teachers WHERE Key_name = 'PRIMARY';'

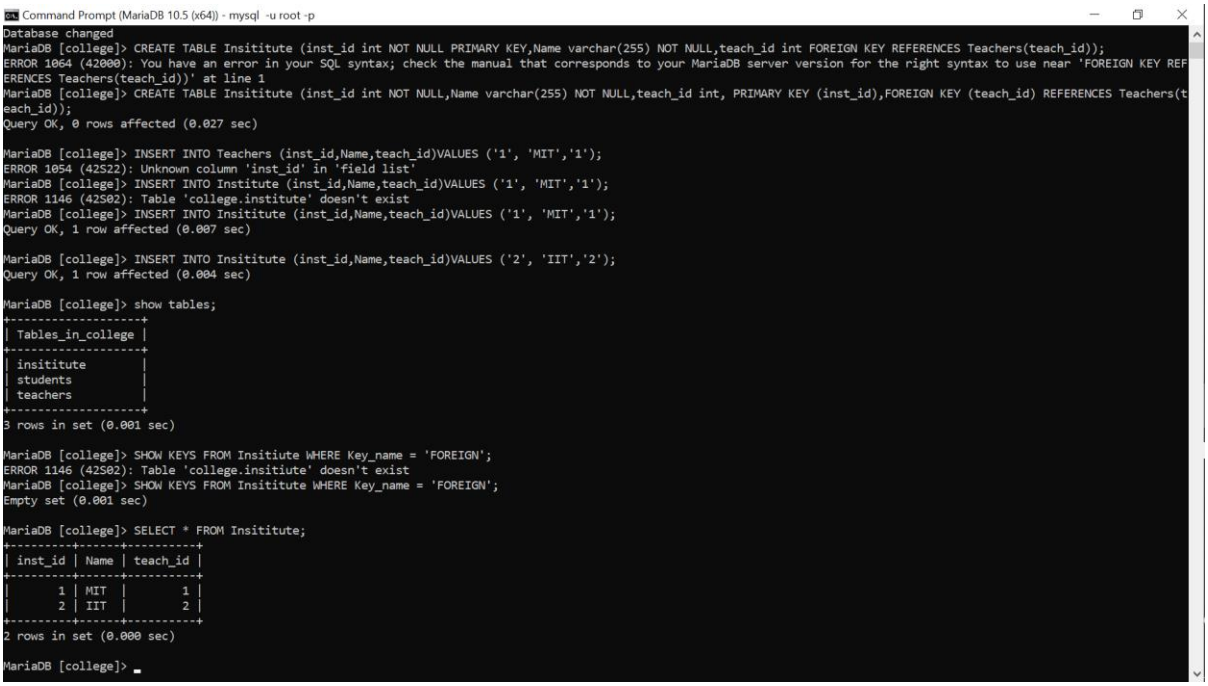
```
MariaDB [college]> select * FROM Teachers;
+----------+-----------+-----------+---------+----------+
| teach_id | LastName  | FirstName | Address | City     |
+----------+-----------+-----------+---------+----------+
|        1 | ferguson  | sir alex  | platama | Scotland |
|        2 | mourinhio | jose      | porto   | Portugal |
+----------+-----------+-----------+---------+----------+
2 rows in set (0.000 sec)

MariaDB [college]> SHOW KEYS FROM Teachers WHERE Key_name = 'PRIMARY';
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table    | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| teachers |          0 | PRIMARY  |            1 | teach_id    | A         |           2 |     NULL | NULL   |      | BTREE      |         |               |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
1 row in set (0.019 sec)
```

Here we can see The implementation of PRIMARY KEY AND NOT NULL CONSTRAINS

Now For FOREIGN KEY

8) CREATE TABLE Insititute (inst_id int NOT NULL,Name varchar(255) NOT NULL,teach_id int, PRIMARY KEY (inst_id),FOREIGN KEY (teach_id) REFERENCES Teachers(teach_id));
9) INSERT INTO Teachers (inst_id,Name,teach_id)VALUES ('1', 'MIT','1');
10) INSERT INTO Insititute (inst_id,Name,teach_id)VALUES ('2', 'IIT','2');
11) SELECT * FROM Insititute;


Command Prompt (MariaDB 10.5 (x64)) - mysql -u root -p
Database changed
MariaDB [college]> CREATE TABLE Insititute (inst_id int NOT NULL PRIMARY KEY,Name varchar(255) NOT NULL,teach_id int FOREIGN KEY REFERENCES Teachers(teach_id));
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'FOREIGN KEY REF
ERENCES Teachers(teach_id))' at line 1
MariaDB [college]> CREATE TABLE Insititute (inst_id int NOT NULL,Name varchar(255) NOT NULL,teach_id int, PRIMARY KEY (inst_id),FOREIGN KEY (teach_id) REFERENCES Teachers(t
each_id));
Query OK, 0 rows affected (0.027 sec)
MariaDB [college]> INSERT INTO Teachers (inst_id,Name,teach_id)VALUES ('1', 'MIT','1');
ERROR 1054 (42S22): Unknown column 'inst_id' in 'field list'
MariaDB [college]> INSERT INTO Institute (inst_id,Name,teach_id)VALUES ('1', 'MIT','1');
ERROR 1146 (42S02): Table 'college.institute' doesn't exist
MariaDB [college]> INSERT INTO Insititute (inst_id,Name,teach_id)VALUES ('1', 'MIT','1');
Query OK, 1 row affected (0.007 sec)
MariaDB [college]> INSERT INTO Insititute (inst_id,Name,teach_id)VALUES ('2', 'IIT','2');
Query OK, 1 row affected (0.004 sec)
MariaDB [college]> show tables;
+------------------+
| Tables_in_college |
+------------------+
| insititute |
| students |
| teachers |
+------------------+
3 rows in set (0.001 sec)
MariaDB [college]> SHOW KEYS FROM Insitiute WHERE Key_name = 'FOREIGN';
ERROR 1146 (42S02): Table 'college.insitiute' doesn't exist
MariaDB [college]> SHOW KEYS FROM Insititute WHERE Key_name = 'FOREIGN';
Empty set (0.001 sec)
MariaDB [college]> SELECT * FROM Insititute;
+---------+------+----------+
| inst_id | Name | teach_id |
+---------+------+----------+
| 1 | MIT | 1 |
| 2 | IIT | 2 |
+---------+------+----------+
2 rows in set (0.000 sec)
MariaDB [college]>

VIVA QUESTIONS:

Que1. What are different Constraints in SQL?

Ans.

-NOT NULL Constraint restricts a column from having a NULL value. Once NOT NULL constraint is applied to a column, you cannot pass a null value to that column.

-UNIQUE Constraint ensures that a field or column will only have unique values. A

-UNIQUE constraint field will not have duplicate data.

-Primary key Constraint uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain null value.

-Foreign key Constraint is also used to restrict actions that would destroy links between tables. Foreign key is used to relate two tables.

-CHECK Constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database.

-Default Constraint is used to assign a default value to a column when no value is specified. Index Constraint is used to create and retrieve data from the database very quickly. An Index can be created by using a single or group of columns in a table.

Que2. What is the purpose of Null Constraint?

Ans.

This implies that the column need not receive any value during insert or update operations. a column can hold NULL values. The NULL constraint is logically equivalent to omitting the NOT NULL constraint from the column definition. Once NULL constraint is applied to a column, you can pass a null value to that column.

Que 3. What is Index Constraint?

Ans.

Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries. When the index is created, it is assigned a ROWID for each row before it sorts out the data. Proper indexes are good for performance in large databases.

Que 4. What is the purpose of Default Constraint?

Ans.

The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value. It is used to provide a default value for a column. The default value will be added to all new records IF no other value is specified.

EXPERIMENT - 6

DATABASE MANAGEMENT SYSTEMS LAB

Aim

Write the queries for implementing the following functions: MAX(), MIN(), AVG(), COUNT() and other logical pattern matching operations.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 6

Aim:

Write the queries for implementing the following functions: MAX(), MIN(), AVG(), COUNT() and other logical pattern matching operations.

Tools Used:

MariaDB

Procedure/ Queries:

1. List the names of all clients having 'a' as the second letter in their names.

```
MariaDB [labdb2]> SELECT * FROM client_master
    -> WHERE name LIKE '_a%';
+----------+------+----------+----------+--------+---------+-------------+----------+
| clientno | name | address1 | address2 | city   | pincode | state       | bal_due  |
+----------+------+----------+----------+--------+---------+-------------+----------+
| C00003   | Raj  | P-7      | Bandra   | Mumbai |  400032 | Maharashtra | 12000.00 |
+----------+------+----------+----------+--------+---------+-------------+----------+
1 row in set (0.545 sec)
```

2. List the client who stay in the city whose first letter is 'M.

```
MariaDB [labdb2]> SELECT * FROM client_master
    -> WHERE city LIKE 'M%';
+----------+----------+----------+----------+--------+---------+-------------+----------+
| clientno | name     | address1 | address2 | city   | pincode | state       | bal_due  |
+----------+----------+----------+----------+--------+---------+-------------+----------+
| C00001   | Aman     | A/14     | Worli    | Mumbai |  400002 | Maharashtra | 30000.00 |
| C00002   | Omkar    | 65       | Nariman  | Mumbai |  400001 | Maharashtra |  8000.00 |
| C00003   | Raj      | P-7      | Bandra   | Mumbai |  400032 | Maharashtra | 12000.00 |
| C00004   | Ashi     | A/9      | Juhu     | Mumbai |  400044 | Maharashtra |     0.00 |
| C00005   | Ashish   | A/5      | Juhu     | Mumbai |  400044 | Maharashtra |  3500.00 |
| C00006   | Ashutosh | F/5      | Andheri  | Mumbai |  400044 | Maharashtra |     0.00 |
+----------+----------+----------+----------+--------+---------+-------------+----------+
6 rows in set (0.009 sec)
```

3. List all clients who stay in 'Manglore' or ‘Banglore'

```
MariaDB [labdb2]> SELECT * FROM client_master
    -> WHERE city IN ('Manglore', 'Banglore');
Empty set (0.028 sec)
```

4. List all the clients whose bal due=8,000.

```
MariaDB [labdb2]> SELECT * FROM client_master
    -> WHERE bal_due=8000;
+----------+-------+----------+----------+--------+---------+-------------+---------+
| clientno | name  | address1 | address2 | city   | pincode | state       | bal_due |
+----------+-------+----------+----------+--------+---------+-------------+---------+
| C00002   | Omkar | 65       | Nariman  | Mumbai |  400001 | Maharashtra | 8000.00 |
+----------+-------+----------+----------+--------+---------+-------------+---------+
1 row in set (0.485 sec)
```

5. List all the information frome sales order for orders placed in the month of June October.

```
MariaDB [labdb2]> SELECT * FROM sales_order
    -> WHERE orderdate LIKE '____-__-10';
+----------+-----------+------------+-------------+-----------+--------+------------+-------------+
| order_no | client_no | orderdate  | salesman_no | delivtype | billyn | delivdate  | orderstatus |
+----------+-----------+------------+-------------+-----------+--------+------------+-------------+
| O19001   | C00001    | 2012-01-10 | S00001      | F         | N      | 2020-01-10 | In process  |
| O19002   | C00002    | 2025-01-10 | S00002      | P         | N      | 2027-01-10 | Cancelled   |
| O19003   | C00004    | 2003-04-10 | S00001      | F         | Y      | 2007-04-10 | Fulfilled   |
| O19008   | C00006    | 2024-05-10 | S00004      | F         | N      | 2026-05-10 | In process  |
| O46865   | C00003    | 2018-02-10 | S00003      | F         | Y      | 2020-02-10 | Fulfilled   |
| O46866   | C00005    | 2020-05-10 | S00002      | P         | N      | 2022-05-10 | Cancelled   |
+----------+-----------+------------+-------------+-----------+--------+------------+-------------+
6 rows in set (0.011 sec)
```

6. List the order information for the client number 'C00001' and ' C00002

7. List the products who selling price is greater than 500 and less than or equal to 750.

```
MariaDB [labdb2]> SELECT * FROM product_master
    -> WHERE sell_price BETWEEN 500 AND 750;
+------------+-------------+---------------+-------------+----------+-----------+------------+------------+
| Production | Description | Profit_Percent | UnitMeasure | QTYONHAND | ReorderLvl | Sell_Price | Cost_Price |
+------------+-------------+---------------+-------------+----------+-----------+------------+------------+
| P0345      | Shirts      |             6 | Piece       |      150 |        50 |        500 |        350 |
| P06734     | Cotton Jeans |             5 | Piece       |      100 |        20 |        600 |        450 |
| P07865     | Jeans       |             5 | Piece       |      100 |        20 |        750 |        500 |
| P07885     | Pull Overs  |             3 | Piece       |       80 |        30 |        700 |        450 |
+------------+-------------+---------------+-------------+----------+-----------+------------+------------+
4 rows in set (0.627 sec)
```

8. List products who's selling price is more than 500. Calculate a new Selling Price as original selling price multiplied by 0.15.

```
MariaDB [labdb2]> SELECT description,
    -> Sell_price*0.15
    -> FROM product_master;
+--------------+-----------------+
| description  | Sell_price*0.15 |
+--------------+-----------------+
| T-Shirts     |          802.50 |
| Shirts       |           75.00 |
| Cotton Jeans |           90.00 |
| Jeans        |          112.50 |
| Trousers     |          127.50 |
| Pull Overs   |          105.00 |
| Denim Shirts |           52.50 |
| Lycra Tops   |           45.00 |
| Skirts       |           67.50 |
+--------------+-----------------+
9 rows in set (0.254 sec)
```

9. Rename the new column in the output of above query as new_price.

```
MariaDB [labdb2]> SELECT description,
    -> Sell_price*0.15 as new_price
    -> FROM product_master;
+--------------+-----------+
| description  | new_price |
+--------------+-----------+
| T-Shirts     |    802.50 |
| Shirts       |     75.00 |
| Cotton Jeans |     90.00 |
| Jeans        |    112.50 |
| Trousers     |    127.50 |
| Pull Overs   |    105.00 |
| Denim Shirts |     52.50 |
| Lycra Tops   |     45.00 |
| Skirts       |     67.50 |
+--------------+-----------+
9 rows in set (0.018 sec)
```

10. List the name city of clients who are not in the state of Maharashtra'.

```
MariaDB [labdb2]> SELECT name,city FROM client_master
    -> WHERE state NOT IN ('Maharashtra');
Empty set (0.001 sec)
```

11. Count the total number of orders.

```
MariaDB [labdb2]> SELECT COUNT(order_no) FROM sales_order;
+-----------------+
| COUNT(order_no) |
+-----------------+
|               6 |
+-----------------+
1 row in set (0.181 sec)
```

12. Calculate the average price of all the products.

```
MariaDB [labdb2]> SELECT AVG(sell_price) FROM product_master;
+-----------------+
| AVG(sell_price) |
+-----------------+
|       1094.4444 |
+-----------------+
1 row in set (0.015 sec)
```

13. Determine the maximum and minimum product prices. Rename the output as max price and min_price respectively.

```
MariaDB [labdb2]> SELECT MAX(sell_price) as max_price, MIN(sell_price) as min_price FROM product_master;
+-----------+-----------+
| max_price | min_price |
+-----------+-----------+
|      5350 |       300 |
+-----------+-----------+
1 row in set (0.020 sec)
```

14. Count the number of products having price less than or equal to 500

15. List the products whose qtyonhand is less than 3 order level.

```
SELECT * FROM client_master
WHERE name LIKE '_a%';

SELECT * FROM client_master
WHERE city LIKE 'M%';

SELECT * FROM client_master
WHERE city IN ('Manglore', 'Banglore');

SELECT * FROM client_master
WHERE bal_due=8000;

SELECT * FROM sales_order
WHERE orderdate LIKE '____-__-10';
```

```
SELECT * FROM sales_order_details
WHERE clientno IN ('C00001', 'C00002');
```

```
SELECT * FROM client_master
WHERE city IN ('Manglore', 'Banglore');
```

```
SELECT * FROM product_master
WHERE sell_price BETWEEN 500 AND 750;
```

```
SELECT * FROM product_master
WHERE sell_price BETWEEN 500 AND 750;
```

```
SELECT description,
Sell_price*0.15
FROM product_master;
```

```
SELECT description,
Sell_price*0.15 as new_price
FROM product_master;
```

```
SELECT name,city FROM client_master
WHERE state NOT IN (‘Maharashtra’);
```

```
SELECT COUNT(order_no) FROM sales_order;
```

```
SELECT AVG(sell_price) FROM product_master;
```

```
SELECT MAX(sell_price) as max_price, MIN(sell_price) as min_price
FROM product_master;
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> drop database student;
Query OK, 2 rows affected, 2 warnings (0.007 sec)

MariaDB [(none)]> create database student;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use student;
Database changed
MariaDB [student]> create table employee(id int,name varchar(20), age int,primary key(id));
Query OK, 0 rows affected (0.024 sec)

MariaDB [student]> insert into employee values(01,'Tushar',10);
Query OK, 1 row affected (0.002 sec)

MariaDB [student]> insert into employee values(02,'Aman',11);
Query OK, 1 row affected (0.001 sec)

MariaDB [student]> insert into employee values(03,'Deepanshu',12);
Query OK, 1 row affected (0.001 sec)

MariaDB [student]> insert into employee values(03,'Gaurav',13);
ERROR 1062 (23000): Duplicate entry '3' for key 'PRIMARY'
MariaDB [student]> insert into employee values(w,'Gaurav',13);
ERROR 1054 (42S22): Unknown column 'w' in 'field list'
MariaDB [student]> insert into employee values('w','Gaurav',13);
ERROR 1366 (22007): Incorrect integer value: 'w' for column `student`.`employee`.`id` at row 1
MariaDB [student]> insert into employee values(nill,'Gaurav',13);
ERROR 1054 (42S22): Unknown column 'nill' in 'field list'
MariaDB [student]> insert into employee values(null,'Gaurav',13);
ERROR 1048 (23000): Column 'id' cannot be null
MariaDB [student]> select * from employee;
+----+-----------+------+
| id | name      | age  |
+----+-----------+------+
|  1 | Tushar    |   10 |
|  2 | Aman      |   11 |
|  3 | Deepanshu |   12 |
+----+-----------+------+
3 rows in set (0.001 sec)

MariaDB [student]>
```

1. SELECT COUNT(*) AS `Count` FROM Client_master;

```
MariaDB [sales]> SELECT COUNT(*) AS `Count` FROM Client_master;
+-------+
| Count |
+-------+
|     6 |
+-------+
1 row in set (0.001 sec)
```

reeha@Reeha: ~

```
MariaDB [sales]> CREATE TABLE IF NOT EXISTS products (
    ->          productID    INT UNSIGNED  NOT NULL AUTO_INCREMENT,
    ->          productCode  CHAR(3)       NOT NULL DEFAULT '',
    ->          name         VARCHAR(30)   NOT NULL DEFAULT '',
    ->          quantity     INT UNSIGNED  NOT NULL DEFAULT 0,
    ->          price        DECIMAL(7,2)  NOT NULL DEFAULT 99999.99,
    ->          PRIMARY KEY  (productID)
    ->        );
Query OK, 0 rows affected (0.046 sec)

MariaDB [sales]>  INSERT INTO products VALUES (1001, 'PEN', 'Pen Red', 5000, 1.23);
Query OK, 1 row affected (0.019 sec)

MariaDB [sales]>  INSERT INTO products VALUES
    ->          (NULL, 'PEN', 'Pen Blue',  8000, 1.25),
    ->          (NULL, 'PEN', 'Pen Black', 2000, 1.25);
Query OK, 2 rows affected (0.003 sec)
Records: 2  Duplicates: 0  Warnings: 0

MariaDB [sales]> INSERT INTO products (productCode, name, quantity, price) VALUES
    ->          ('PEC', 'Pencil 2B', 10000, 0.48),
    ->          ('PEC', 'Pencil 2H', 8000, 0.49);
Query OK, 2 rows affected (0.007 sec)
Records: 2  Duplicates: 0  Warnings: 0

MariaDB [sales]> INSERT INTO products (productCode, name) VALUES ('PEC', 'Pencil HB');
Query OK, 1 row affected (0.007 sec)

MariaDB [sales]> SELECT * FROM products;
+-----------+-------------+-----------+----------+----------+
| productID | productCode | name      | quantity | price    |
+-----------+-------------+-----------+----------+----------+
|      1001 | PEN         | Pen Red   |     5000 |     1.23 |
|      1002 | PEN         | Pen Blue  |     8000 |     1.25 |
|      1003 | PEN         | Pen Black |     2000 |     1.25 |
|      1004 | PEC         | Pencil 2B |    10000 |     0.48 |
|      1005 | PEC         | Pencil 2H |     8000 |     0.49 |
|      1006 | PEC         | Pencil HB |        0 | 99999.99 |
+-----------+-------------+-----------+----------+----------+
6 rows in set (0.001 sec)

MariaDB [sales]> DELETE FROM products WHERE productID = 1006;
```

```
MariaDB [sales]> SELECT COUNT(*) AS `Count` FROM products;
+-------+
| Count |
+-------+
|     5 |
+-------+
1 row in set (0.001 sec)
```

```
MariaDB [sales]> SELECT productCode, COUNT(*) FROM products GROUP BY productCode;
+-------------+----------+
| productCode | COUNT(*) |
+-------------+----------+
| PEC         |        2 |
| PEN         |        3 |
+-------------+----------+
2 rows in set (0.006 sec)
```

```
MariaDB [sales]> SELECT productCode, COUNT(*) AS count
    ->         FROM products
    ->         GROUP BY productCode
    ->         ORDER BY count DESC;
+--------------+-------+
| productCode | count |
+--------------+-------+
| PEN          |     3 |
| PEC          |     2 |
+--------------+-------+
2 rows in set (0.001 sec)
```

```
MariaDB [sales]>  SELECT MAX(price), MIN(price), AVG(price), STD(price), SUM(quantity)
    ->        FROM products;
+------------+------------+------------+------------+---------------+
| MAX(price) | MIN(price) | AVG(price) | STD(price) | SUM(quantity) |
+------------+------------+------------+------------+---------------+
|       1.25 |       0.48 |   0.940000 |   0.371591 |         33000 |
+------------+------------+------------+------------+---------------+
1 row in set (0.001 sec)
```

```
MariaDB [sales]>  SELECT productCode, MAX(price) AS `Highest Price`, MIN(price) AS `L
owest Price`
    ->        FROM products
    ->        GROUP BY productCode;
+-------------+---------------+--------------+
| productCode | Highest Price | Lowest Price |
+-------------+---------------+--------------+
| PEC         |          0.49 |         0.48 |
| PEN         |          1.25 |         1.23 |
+-------------+---------------+--------------+
2 rows in set (0.001 sec)
```

```
MariaDB [sales]> SELECT productCode, MAX(price), MIN(price),
    ->              CAST(AVG(price) AS DECIMAL(7,2)) AS `Average`,
    ->              CAST(STD(price) AS DECIMAL(7,2)) AS `Std Dev`,
    ->              SUM(quantity)
    ->        FROM products
    ->        GROUP BY productCode;
+-------------+-----------+-----------+---------+---------+---------------+
| productCode | MAX(price) | MIN(price) | Average | Std Dev | SUM(quantity) |
+-------------+-----------+-----------+---------+---------+---------------+
| PEC         |       0.49 |       0.48 |    0.49 |    0.01 |         18000 |
| PEN         |       1.25 |       1.23 |    1.24 |    0.01 |         15000 |
+-------------+-----------+-----------+---------+---------+---------------+
2 rows in set (0.001 sec)
```

```
MariaDB [sales]> SELECT
    ->              productCode AS `Product Code`,
    ->              COUNT(*) AS `Count`,
    ->              CAST(AVG(price) AS DECIMAL(7,2)) AS `Average`
    ->         FROM products
    ->         GROUP BY productCode
    ->         HAVING Count >=3;
+--------------+-------+---------+
| Product Code | Count | Average |
+--------------+-------+---------+
| PEN          |     3 |    1.24 |
+--------------+-------+---------+
1 row in set (0.001 sec)
```

```
MariaDB [sales]> SELECT
    ->           productCode,
    ->           MAX(price),
    ->           MIN(price),
    ->           CAST(AVG(price) AS DECIMAL(7,2)) AS `Average`,
    ->           SUM(quantity)
    ->        FROM products
    ->        GROUP BY productCode
    ->        WITH ROLLUP;
+-------------+------------+------------+---------+---------------+
| productCode | MAX(price) | MIN(price) | Average | SUM(quantity) |
+-------------+------------+------------+---------+---------------+
| PEC         |       0.49 |       0.48 |    0.49 |         18000 |
| PEN         |       1.25 |       1.23 |    1.24 |         15000 |
| NULL        |       1.25 |       0.48 |    0.94 |         33000 |
+-------------+------------+------------+---------+---------------+
3 rows in set (0.001 sec)

MariaDB [sales]>
```

VIVA VOCE QUESTIONS

Q.1 What are pattern matching operation?

Ans

SQL pattern matching allows you to search for patterns in data if you don't know the exact word or phrase you are seeking. This kind of SQL query uses wild card characters to match a pattern rather than specifying it exactly.

Q.2 What are different variants of like command?

Ans.

LIKE Operator	DESCRIPTION
WHERE CustomerName LIKE ‘a%’	Finds any value that starts with “a”
WHERE CustomerName LIKE ‘%a’	Finds any value that ends with “a”
WHERE CustomerName LIKE ‘%or%’	Finds any value that has “or” in any position
WHERE CustomerName LIKE ‘_r%’	Finds any value that has “r” in the second position.
WHERE CustomerName LIKE ‘a_%_%’	Finds any value that starts with “a” and are at least 3 characters in length.
WHERE ContactName LIKE ‘a%o’	Finds any value that starts with “a” and ends with “o”.

Q.3 What are different Logical operations?

Ans.

OPERATOR	**DESCRIPTION**
ALL	True if all of the subquery values meet the condition
AND	True if all the conditions separated by AND is true.

ANY	True if any of the subquery meets the condition.
BETWEEN	True if operand is within the range of comparisons.
EXISTS	True if the subquery returns one or more records.
IN	True if the operand is equal to one of a list of expressions.
NOT	Displays a record if the condition(s) is NOT TRUE
OR	True if any of the conditions separated by OR is true.

Q.4 What is difference between IN and BETWEEN command?

Ans.

The IN command allows you to specify multiple values in a WHERE clause. it is a shorthand for multiple OR conditions.

Whereas BETWEEN command select values within a given range the values can be numbers, text or dates. The BETWEEN operator is inclusive: begin an end value are included.

EXPERIMENT - 7

DATABASE MANAGEMENT SYSTEMS LAB

Aim

Write the SQL queries to implement DATE and MONTH commands.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 7

Aim:

Write the SQL queries to implement DATE and MONTH commands.

Tools Used:

MariaDB

Procedure/ Queries:

In SQL, we have many data types available, which we can use as the date in our table. Some of them popularly being- 'YYYY-MM-DD' and 'DD-MM-YYYY'.

In some scenarios, we also have time stored in our database with the date, in such cases, we need tools to separately access the time and the date. This is where the SQL time and functions come in handy.

Also, as a beginner, one should be very careful when using date or DateTime in the database as these are very likely to give exceptions if not dealt with properly.

Formats of Date Time in SQL:

DATE – YYYY-MM-DD
DATETIME – YYYY-MM-DD HH:MI:SS
TIMESTAMP – YYYY-MM-DD HH:MI:SS
YEAR – YYYY or YY

Why do we need Date and Time Functions?

We have plenty of functions available for date and time in SQL. These are provided to make sure smooth access of the date and time module while making and accessing a SQL database.

Some of the most popular date and time functions are as follows:

Sr.No	Function	Description
1	NOW( )	Displays the current date and time.
2	CURDATE( )	Displays the current date.

3	CURTIME( )	Displays the current time.
4	DATE( )	Displays the date from the Date/DateTime expression.
5	EXTRACT( )	Displays selected part i.e. date/time.
6	DAY( )	Displays the day from the given date.
7	MONTH( )	Displays the month from the given date.
8	YEAR( )	Displays the year from the given date.
9	DATE_ADD( )	Displays date after adding the given interval.
10	DATE_SUB( )	Displays date after subtracting the given interval.
11	DATEDIFF( )	Displays the interval between two dates.
12	DATE_FORMAT( )	Displays the date/time data in various formats available.

DATE

1. CREATE TABLE employees (dates DATE);

```
MariaDB [sales]> CREATE TABLE employees (dates DATE);
Query OK, 0 rows affected (0.047 sec)
```

2. INSERT INTO employees VALUES ("2010-01-12"), ("2011-2-28"), ('120314'),('13*04*21');

```
MariaDB [sales]> INSERT INTO employees VALUES ("2010-01-12"), ("2011-2-28"), ('120314'),('13*04*21');
Query OK, 4 rows affected (0.020 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

3. select * from employees;

```
MariaDB [sales]> select * from employees;
+------------+
| dates      |
+------------+
| 2010-01-12 |
| 2011-02-28 |
| 2012-03-14 |
| 2013-04-21 |
+------------+
4 rows in set (0.001 sec)
```

MONTH

1. SELECT MONTH('2014-05-19');

```
MariaDB [sales]> SELECT MONTH('2014-05-19');
+----------------------+
| MONTH('2014-05-19') |
+----------------------+
|                    5 |
+----------------------+
1 row in set (0.005 sec)
```

2. SELECT MONTH('2014-05-19 09:04:05');

```
MariaDB [sales]> SELECT MONTH('2014-05-19 09:04:05');
+------------------------------+
| MONTH('2014-05-19 09:04:05') |
+------------------------------+
|                            5 |
+------------------------------+
1 row in set (0.000 sec)
```

3. SELECT MONTH('2013-11-23');

```
MariaDB [sales]> SELECT MONTH('2013-11-23');
+----------------------+
| MONTH('2013-11-23') |
+----------------------+
|                   11 |
+----------------------+
1 row in set (0.000 sec)
```

4. SELECT MONTH(CURDATE());

```
MariaDB [sales]> SELECT MONTH(CURDATE());
+------------------+
| MONTH(CURDATE()) |
+------------------+
|                6 |
+------------------+
1 row in set (0.000 sec)
```

```
SELECT '2008-12-31 23:59:59' + INTERVAL 1 SECOND;
```

```
MariaDB [sales]> SELECT '2008-12-31 23:59:59' + INTERVAL 1 SECOND;
+-----------------------------------------+
| '2008-12-31 23:59:59' + INTERVAL 1 SECOND |
+-----------------------------------------+
| 2009-01-01 00:00:00                     |
+-----------------------------------------+
1 row in set (0.005 sec)
```

SELECT NOW();

```
MariaDB [sales]> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2021-06-18 00:49:29 |
+---------------------+
1 row in set (0.001 sec)
```

SELECT CURDATE();

```
MariaDB [sales]> SELECT CURDATE();
+------------+
| CURDATE()  |
+------------+
| 2021-06-18 |
+------------+
1 row in set (0.000 sec)
```

SELECT CURTIME();

```
1 row in set (0.000 sec)

MariaDB [sales]> SELECT CURTIME();
+-----------+
| CURTIME() |
+-----------+
| 00:49:50  |
+-----------+
1 row in set (0.000 sec)
```

SELECT DATEDIFF('2017-01-13','2017-01-03') AS DateDiff;

```
MariaDB [sales]> SELECT DATEDIFF('2017-01-13','2017-01-03') AS DateDiff;
+----------+
| DateDiff |
+----------+
|       10 |
+----------+
1 row in set (0.000 sec)
```

SELECT DATE_FORMAT(NOW(),'%d %b %y');

```
MariaDB [sales]> SELECT DATE_FORMAT(NOW(),'%d %b %y')
    -> ;
+-------------------------------+
| DATE_FORMAT(NOW(),'%d %b %y') |
+-------------------------------+
| 18 Jun 21                     |
+-------------------------------+
1 row in set (0.000 sec)

MariaDB [sales]>
```

VIVA QUESTIONS

Q1. What is DATE command?

Ans.

MySQL DATE() takes the date part out from a datetime expression.

The **DATE**() **function** extracts the **date** part from a **datetime** expression.

Q2). What is month Function?

Ans.

The **MONTH**() function returns an integer value which represents the **month** of a specified date. The **MONTH**() function takes an argument which can be a literal date value or an expression that can resolve to a TIME , DATE , SMALLDATETIME , DATETIME , DATETIME2 , or DATETIMEOFFSET value.

Q3). What is syntax of Date Command?

Ans.

SQL Server comes with the following data types for storing a **date** or a **date**/time value in the database: **DATE** - format YYYY-MM-DD. **DATETIME** - format: YYYY-MM-DD HH:MI:SS. SMALLDATETIME - format: YYYY-MM-DD HH:MI:SS. TIMESTAMP - format: a unique number.

OrderId	ProductName	OrderDate
1	Geitost	2008-11-11
2	Camembert Pierrot	2008-11-09
3	Mozzarella di Giovanni	2008-11-11
4	Mascarpone Fabioli	2008-10-29

Now we want to select the records with an OrderDate of "2008-11-11" from the table above.

We use the following `SELECT` statement:

```
SELECT * FROM Orders WHERE OrderDate='2008-11-11'
```

The result-set will look like this:

OrderId	ProductName	OrderDate
1	Geitost	2008-11-11
3	Mozzarella di Giovanni	2008-11-11

Q4). Write function for month command.

Ans.

SQL Server **MONTH**() Function

The **MONTH**() function returns the **month** part for a specified date (a number from 1 to 12).

Q5). What is difference between DATETIME and TIMESTAMP command?

Ans.

TIMESTAMP is four bytes vs eight bytes **for DATETIME**. **Timestamps** are also lighter on the database and indexed faster. The **DATETIME** type is used when you need values that contain both date and time information. MySQL retrieves and displays **DATETIME** values in 'YYYY-MM-DD HH:MM:SS' **format**.

TIMESTAMP value converts from the current time to UTC and vice-versa while **DATETIME** does not do any conversion. TIMESTAMPdiffers with current time zone settings while **DATETIME** remains constant. **TIMESTAMP** data can be indexed while the **DATETIME** data cannot.

EXPERIMENT - 8

DATABASE MANAGEMENT SYSTEMS LAB

Aim

Write the SQL queries to implement Having and Group By Clauses on table.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 8

Aim:

Write the SQL queries to implement Having and Group By Clauses on table.

Tools Used:

MariaDB

Procedure/ Queries:

What is the SQL Group by Clause?

The GROUP BY clause is a SQL command that is used to **group rows that have the same values**. The GROUP BY clause is used in the SELECT statement. Optionally it is used in conjunction with aggregate functions to produce summary reports from the database.

That's what it does, **summarizing data** from the database.

The queries that contain the GROUP BY clause are called grouped queries and only return a single row for every grouped item.

SQL GROUP BY Syntax

Now that we know what the SQL GROUP BY clause is, let's look at the syntax for a basic group by query.

SELECT statements... GROUP BY column_name1[,column_name2,...] [HAVING condition];

- "SELECT statements..." is the standard SQL SELECT command query.
- "**GROUP BY** *column_name1*" is the clause that performs the grouping based on column_name1.
- "[,column_name2,...]" is optional; represents other column names when the grouping is done on more than one column.
- "[HAVING condition]" is optional; it is used to restrict the rows affected by the GROUP BY clause. It is similar to the WHERE clause.

In this experiment we will be working on clauses like HAVING and GROUP BY to be applied on a table.

HAVING Clause: The HAVING clause was added to SQL because the WHERE keyword could not be used with the aggregate functions.

SYNTAX:

SELECT columnname(s)

From tablename

Where condition

GROUP BY columnname(s)

Having condition

ORDER BY columnname(s);

GROUP BY: The GROUP BY statement is often used with the aggregate functions(COUNT,MAX,MIN,SUM,AVG) to group the result set by one or more columns.

SYNTAX:

SELECT columnname(s)

FROM tablename

Where condition

GROUP BY columnname(s)

ORDER BY columnname(s);

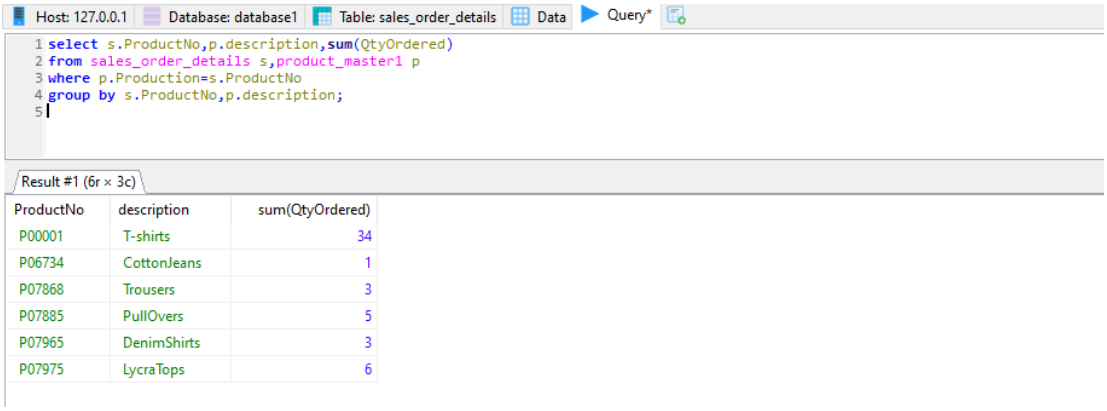
QUERIES:

Print the description and total qty sold for each product.

Input Query:

```
select s.ProductNo,p.description,sum(QtyOrdered)
from sales_order_details s,product_master1 p
where p.Production=s.ProductNo
group by s.ProductNo,p.description;
```

OUTPUT:



Find the value of each product sold.

Input Query:

```
SELECT SOD.ProductNo, PM.DESCRIPTION, SUM(SOD.QTYDISP * SOD.PRODUCTRATE)
SALESPERPRODUCT
FROM SALES_ORDER_DETAILS SOD, product_master1 PM
WHERE PM.PRODUCTION = SOD.ProductNo
GROUP BY SOD.ProductNo, PM.PRODUCTION;
```

OUTPUT:

Host: 127.0.0.1 | Database: database1 | Table: sales_order_details | Data | Query* | Query #2*

```
1 SELECT SOD.ProductNo, PM.DESCRIPTION, SUM(SOD.QTYDISP * SOD.PRODUCTRATE) SALESPERPRODUCT
2 FROM SALES_ORDER_DETAILS SOD, product_master1 PM
3 WHERE PM.PRODUCTION = SOD.ProductNo
4 GROUP BY SOD.ProductNo, PM.PRODUCTION;
```

SALES_ORDER_DETAILS (6r × 3c)

ProductNo	DESCRIPTION	SALESPERPRODUCT
P00001	T-shirts	9,987
P06734	CottonJeans	12,000
P07868	Trousers	9,450
P07885	PullOvers	10,500
P07965	DenimShirts	8,400
P07975	LycraTops	3,150

Calculate the average qty sold for each client that has a maximum order value for 15000.00

Input Query:

```
SELECT CM.CLIENTNO, AVG(SOD.QTYDISP) AVGSALES
FROM SALES_ORDER_DETAILS SOD, CLIENT_MASTER CM ,SALES_ORDER SO
WHERE CM.CLIENTNO = SO.clientNo AND SO.OrderNo = SOD.ORDERNO
GROUP BY CM.CLIENTNO
HAVING MAX(SOD.QTYOrdered * SOD.ProductRate) > 15000;
```

OUTPUT:

```
1 SELECT CM.CLIENTNO, AVG(SOD.QTYDISP) AVGSALES
2 FROM SALES_ORDER_DETAILS SOD, CLIENT_MASTER CM ,SALES_ORDER SO
3 WHERE CM.CLIENTNO = SO.clientNo AND SO.OrderNo = SOD.ORDERNO
4 GROUP BY CM.CLIENTNO
5 HAVING MAX(SOD.QTYOrdered * SOD.ProductRate) > 15000;
6
```

SALES_ORDER_DETAILS (2r × 2c)

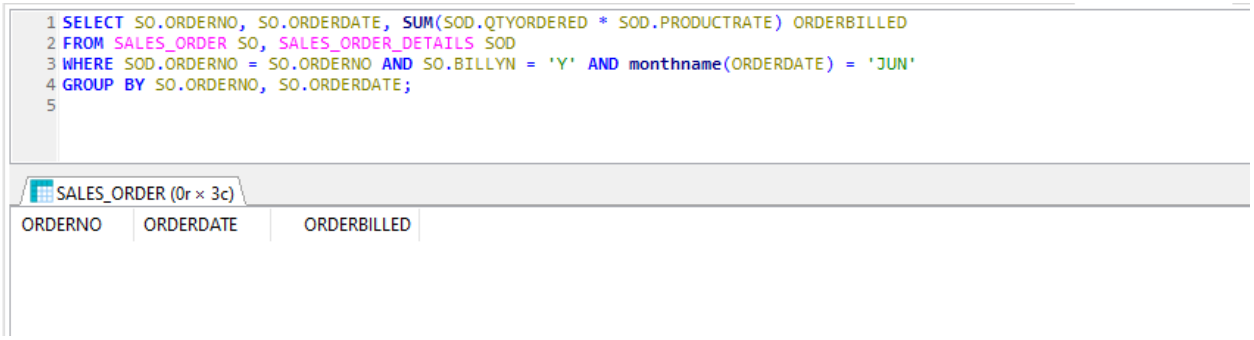
CLIENTNO	AVGSALES
C00001	1.8
C00003	4.5

Find out the total of all the billed orders for the month of June.

Input Query:

```
SELECT SO.ORDERNO, SO.ORDERDATE, SUM(SOD.QTYORDERED * SOD.PRODUCTRATE)
ORDERBILLED
   FROM SALES_ORDER SO, SALES_ORDER_DETAILS SOD
   WHERE SOD.ORDERNO = SO.ORDERNO AND SO.BILLYN = 'Y' AND
monthname(ORDERDATE) = 'JUN'
   GROUP BY SO.ORDERNO, SO.ORDERDATE;
```

OUTPUT:



VIVA QUESTIONS

Q.1: Explain HAVING and GROUP BY clause.

Ans.

HAVING clause: The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions. The HAVING clause is used instead of WHERE with aggregate functions. The having clause is used with the WHERE clause in order to find rows with certain conditions. The having clause is always used after the Group By clause.

GROUP BY: The GROUP BY Statement in SQL is used to arrange identical data into groups with the help of some functions, that is, if a particular column has same values in different rows then it will arrange these rows in a group. The GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups. This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

Q.2: What is the syntax of HAVING clause?

Ans.

HAVING Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Q.3: What is the syntax of GROUP BY clause?

Ans:

GROUP BY Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Q.4: What is the difference between HAVING and WHERE clauses?

Ans:

S.NO	WHERE CLAUSE	HAVING CLAUSE
1.	WHERE Clause is used to filter the records from the table based on the specified condition	HAVING Clause is used to filter record from the groups based on the specified condition.
2.	WHERE CLAUSE can be used without GROUP BY CLAUSE.	HAVING CLAUSE cannot be used without GROUP BY clause
3.	WHERE Clause implements in row operations	HAVING Clause implements in column operation
4.	WHERE CLAUSE cannot contain aggregate functions.	HAVING CLAUSE can contain aggregate functions.
5.	WHERE Clause can be used with SELECT, UPDATE, DELETE statement.	HAVING CLAUSE is used after GROUP BY CLAUSE
6.	WHERE CLAUSE is used before GROUP BY CLAUSE	HAVING CLAUSE is used after GROUP BY CLAUSE

7.	WHERE Clause is used with single row function like UPPER, LOWER etc.	HAVING Clause is used with multiple row function like SUM, COUNT etc.

Q.5: What is the difference between GROUP BY and ORDER BY clause?

Ans:

S.NO	GROUP BY	ORDER BY
1.	Group by statement is used to group the rows that have the same value.	Whereas Order by statement sort the result-set either in ascending or in descending order.
2.	It many be allowed in CREATE VIEW statement.	It is not allowed in CREATE VIEW statement.
3.	In select statement, it is always used before the order by keyword.	While in select statement, it is always used after the group by keyword.
4.	In Group By statement ,attribute cannot be in aggregate function.	In Order By statement ,attribute can be in aggregate function.
5.	In group by clause, the tuples are grouped based on the similarity between the attribute values of tuples	Whereas in order by clause, the result-set is sorted based on ascending or descending order.
6.	It controls the presentation of tuples.	It controls the presentation of attributes.

EXPERIMENT - 8

DATABASE MANAGEMENT SYSTEMS LAB

Aim

Write the SQL queries to implement the joins.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 9

Aim:

Write the SQL queries to implement the joins.

Tools Used:

MariaDB

Procedure/ Queries:

SQL joins are used to fetch / retrieve data from two or more data tables, based on join condition. A join condition is a relationship among some columns in the data tables that take part in SQL join. Basically data tables are related to each other with keys. We use keys relationship in SQL joins.

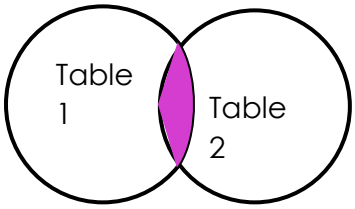
Types of Joins: in SQL Server we have only three types of joints. Using these joins we fetched the data from multiple tables based on condition.

1. **INNER JOIN**: Inner join returns only those records/rows that match/exists in both the tables. Syntax for inner join is as:

Select * from table_1 as t1

Inner join table_2 as t2

On t1.IDcol=t2.IDccol



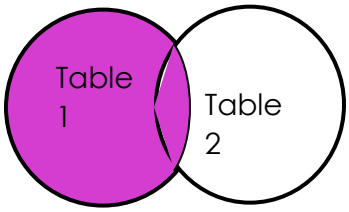
2. **OUTER JOIN:** We have three types of outer joins:

   a) **Left Outer Join:** Left outer join returns all records/rows from left table and from right table returns only matched records. If there are no columns matched in the right table, it returns NULL values. Syntax for Left outer Join is as:

Select * from table1 as t1

Left outer join table2 as t2

On t1.IDcol=t2.IDcol

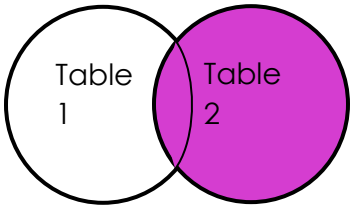


b) **Right Outer Join:** Right outer join returns all records/rows from right table and from left table returns only matched records. If there are no columns matched in the right table, it returns NULL values. Syntax for Left outer Join is as:

Select * from table1 as t1

Right outer join table2 as t2

On t1.IDcol=t2.IDcol

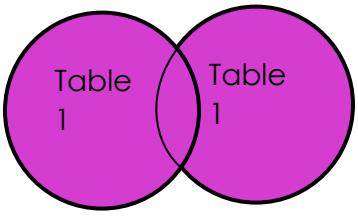


c) **Full outer Join:** full outer join combines left outer join and right outer join. This joint returns all records/rows from both the tables. If there are no columns matching in both the tables, it returns NULL values. Syntax for full outer join is as:

Select * from table1 as t1

Full outer join table2 as t2

On t1.IDcol=t2.IDcol



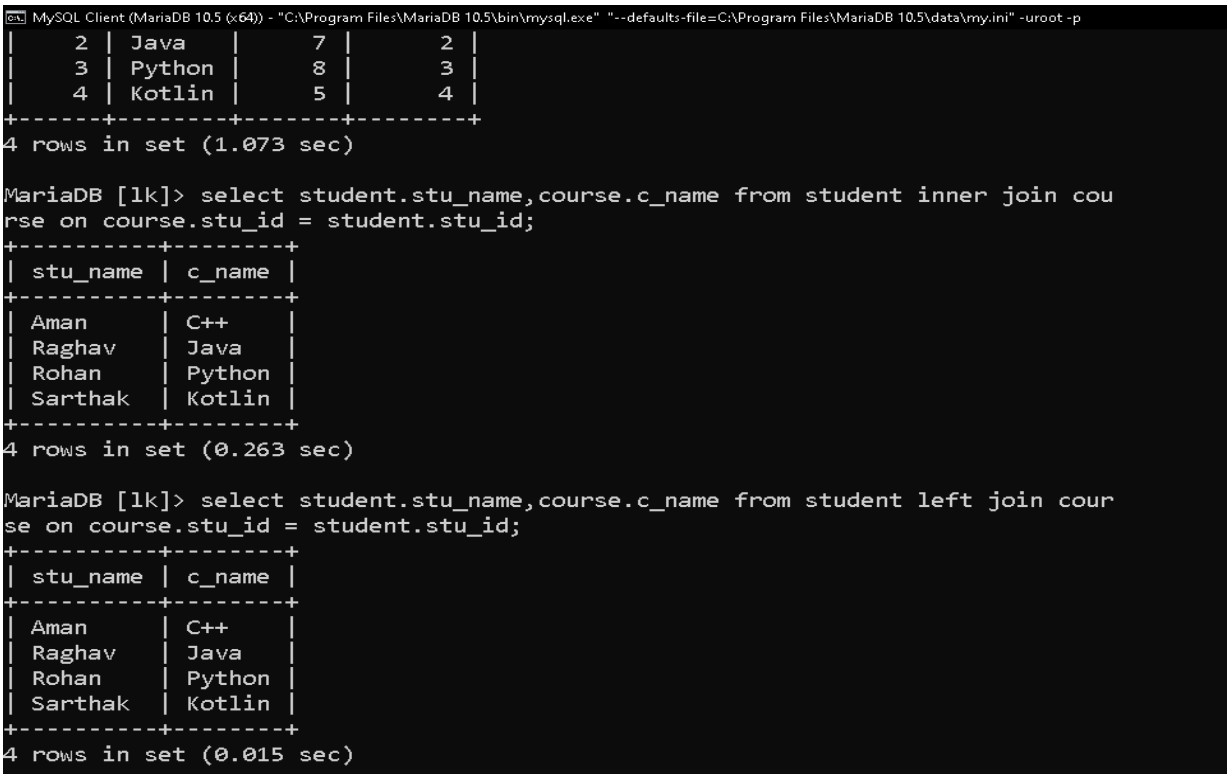
3. **CROSS JOIN:** cross join is the cartesian product of both tables. This joint does not need any condition to join two tables. This joint returns records/rows that are multiplication of records number from both the tables means each row on left table will be related to each row of right table. Syntax for cross join is:

Select*from table1

Cross join table2

4. **SELF JOIN:** Self join is used to join a data base table to itself, particularly when the table has a foreign key that references it's own primary key. Basically we have only three types of joins: inner join, outer join and cross join. We use any of these three joints to join a table to itself. Hence self join is not a type of SQL join.

INNER JOIN & LEFT JOIN :



2. RIGHT JOIN:

```
MariaDB [(none)]> use lk;
Database changed
MariaDB [lk]> select student.stu_name ,course.c_name
    -> from student RIGHT JOIN course on course.stu_id = student.stu_id;
+----------+--------+
| stu_name | c_name |
+----------+--------+
| Aman     | C++    |
| Raghav   | Java   |
| Rohan    | Python |
| Sarthak  | Kotlin |
+----------+--------+
4 rows in set (0.133 sec)
```

QUERIES:

Find out the products which have been sold to 'Ivan Bayross'.

Input Query:

```
select c.name,p.production,p. description,s.orderno
from client_master c,product_master1 p,sales_order s,sales_order_details so
where c.name="Ivan" and c.clientno=s.clientno and s.orderno=so.orderno
and so.productno=p.production;
```

OUTPUT:

```
1 select c.name,p.production,p. description,s.orderno
2 from client_master c,product_master1 p,sales_order s,sales_order_details so
3 where c.name="Ivan" and c.clientno=s.clientno and s.orderno=so.orderno
4   and so.productno=p.production;
5
```

Result #1 (4r × 4c)

name	production	description	orderno
Ivan	P00001	T-shirts	O19001
Ivan	P07965	DenimShirts	O19001
Ivan	P07885	PullOvers	O19001
Ivan	P06734	CottonJeans	O19003

Find out the products and their quantities that we have to deliver in the current month.

Input Query:

```
SELECT SOD.PRODUCTNO, PM.DESCRIPTION, SUM(SOD.QTYORDERED)
FROM SALES_ORDER_DETAILS SOD, SALES_ORDER SO, product_master1 PM
WHERE PM.PRODUCTION = SOD.PRODUCTNO AND SO.ORDERNO = SOD.ORDERNO
AND monthname(SO.OrderDate)=monthname(curdate())
GROUP BY SOD.PRODUCTNO, PM.DESCRIPTION;
```

OUTPUT:

```
1 SELECT SOD.PRODUCTNO, PM.DESCRIPTION, SUM(SOD.QTYORDERED)
2 FROM SALES_ORDER_DETAILS SOD, SALES_ORDER SO, product_master1 PM
3 WHERE PM.PRODUCTION = SOD.PRODUCTNO AND SO.ORDERNO = SOD.ORDERNO
4 AND monthname(SO.OrderDate)=monthname(curdate())
5 GROUP BY SOD.PRODUCTNO, PM.DESCRIPTION;
6
```

SALES_ORDER_DETAILS (0r × 3c)

PRODUCTNO	DESCRIPTION	SUM(SOD.QTYORDERED)

list the product number and the description of the constantly sold(i.e rapidly moving) products.

Input Query:

```
select s.productno,p.description
from sales_order_details s,product_master1 p
where s.productno=p.production
and ProductRate=(select max(ProductRate) from sales_order_details);
```

OUTPUT:

```
1 select s.productno,p.description
2 from sales_order_details s,product_master1 p
3 where s.productno=p.production
4 and ProductRate=(select max(ProductRate) from sales_order_details);
5
```

Result #1 (1r × 2c)

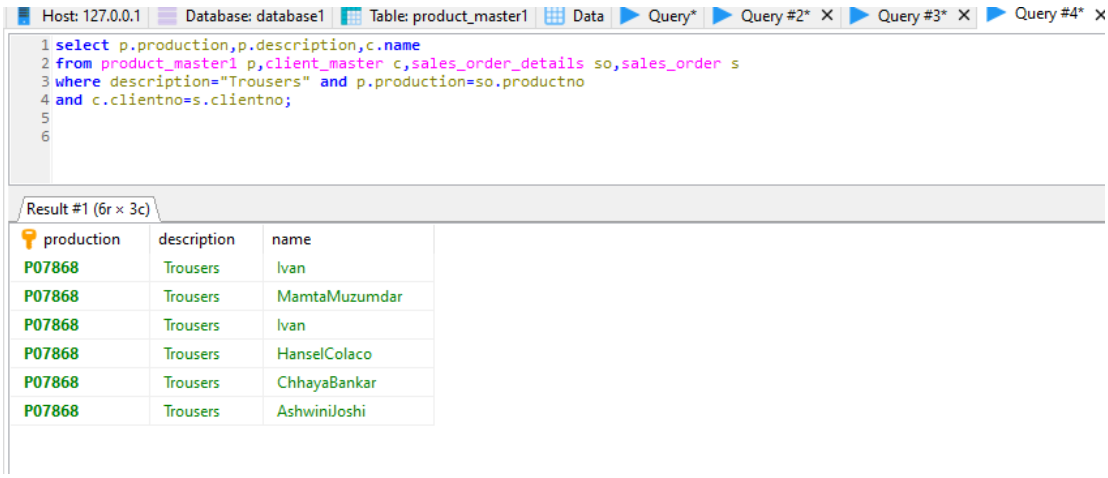
productno	description
P06734	CottonJeans

find the names of clients who have purchased ‘Trousers’.

Input Query:

```
select p.production,p.description,c.name
from product_master1 p,client_master c,sales_order_details so,sales_order s
where description="Trousers" and p.production=so.productno
and c.clientno=s.clientno;
```

OUTPUT:

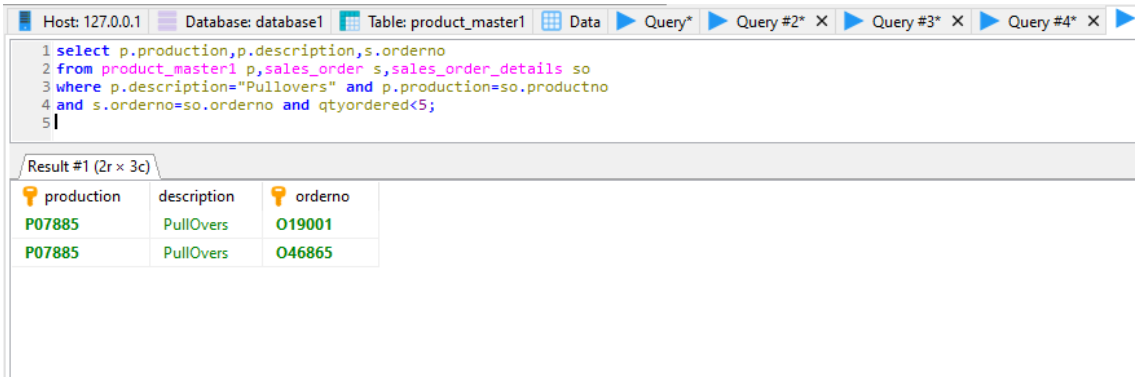


list of products and orders from customers who have ordered less than five units of 'Pullovers'.

Input Query:

```
select p.production,p.description,s.orderno
from product_master1 p,sales_order s,sales_order_details so
where p.description="Pullovers" and p.production=so.productno
and s.orderno=so.orderno and qtyordered<5;
```

OUTPUT:

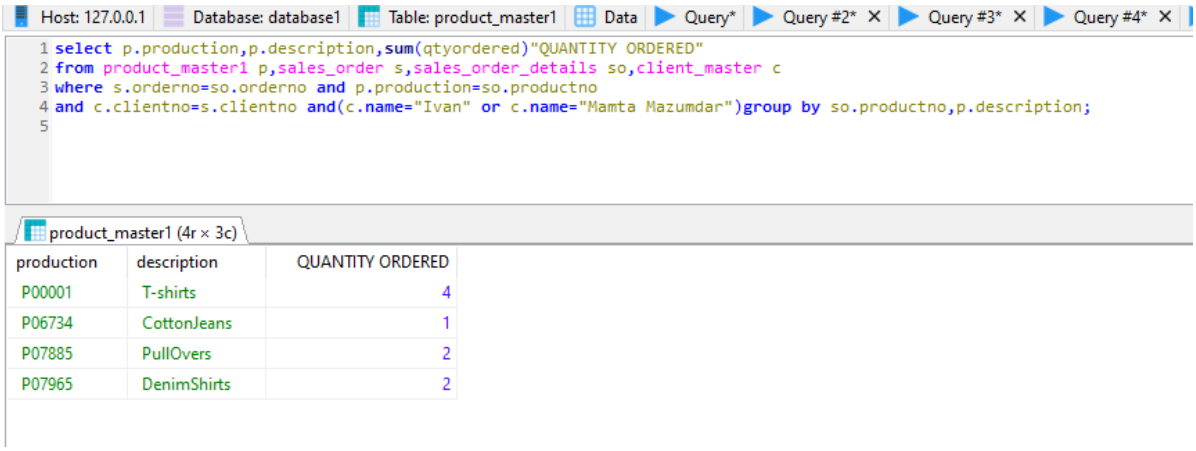


Find the products and their quantities for the orders placed by 'Ivan Bayross' and 'Mamta Muzumdar'.

Input Query:

```
select p.production,p.description,sum(qtyordered)"QUANTITY ORDERED"
from product_master1 p,sales_order s,sales_order_details so,client_master c
where s.orderno=so.orderno and p.production=so.productno
and c.clientno=s.clientno and(c.name="Ivan" or c.name="Mamta Mazumdar")
group by so.productno,p.description;
```

OUTPUT:



Find the products and their quantities for the orders placed by client number 'C00001' and 'C00002'.

Input Query:

SELECT SO.CLIENTNO, SOD.PRODUCTNO, PM.DESCRIPTION, **SUM**(qTYORDERED) UNITSORDERED

FROM SALES_ORDER SO, SALES_ORDER_DETAILS SOD, product_master1 PM, CLIENT_MASTER CM

WHERE SO.ORDERNO = SOD.ORDERNO **AND** SOD.PRODUCTNO = PM.PRODUCTION **AND** SO.CLIENTNO = CM.CLIENTNO

GROUP BY SO.CLIENTNO, SOD.PRODUCTNO, PM.DESCRIPTION

HAVING SO.CLIENTNO = 'C00001' **OR** SO.CLIENTNO = 'C00002';

OUTPUT:

Host: 127.0.0.1 | Database: database1 | Table: product_master1 | Data | Query* | Query #3* | Query #4* | Query #5*

```
1 SELECT SO.CLIENTNO, SOD.PRODUCTNO, PM.DESCRIPTION, SUM(qTYORDERED) UNITSORDERED
2 FROM SALES_ORDER SO, SALES_ORDER_DETAILS SOD, product_master1 PM, CLIENT_MASTER CM
3 WHERE SO.ORDERNO = SOD.ORDERNO AND SOD.PRODUCTNO = PM.PRODUCTION AND SO.CLIENTNO = CM.CLIENTNO
4 GROUP BY SO.CLIENTNO, SOD.PRODUCTNO, PM.DESCRIPTION
5 HAVING SO.CLIENTNO = 'C00001' OR SO.CLIENTNO = 'C00002';
6
```

SALES_ORDER (5r × 4c)

CLIENTNO	PRODUCTNO	DESCRIPTION	UNITSORDERED
C00001	P00001	T-shirts	4
C00001	P06734	CottonJeans	1
C00001	P07885	PullOvers	2
C00001	P07965	DenimShirts	2
C00002	P00001	T-shirts	10

```
MariaDB [(none)]> create database students;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use students;
Database changed
MariaDB [students]> create table info(student_id int, student_name varchar(20), student_address varchar(50));
Query OK, 0 rows affected (0.040 sec)

MariaDB [students]> insert into info vlues(100, 'Tushar', '9/54 bagichi gali');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'vlues(100, 'Tushar', '9/54 bagichi gali')' at l
ine 1
MariaDB [students]> insert into info values(100, 'Tushar', '9/54 bagichi gali');
Query OK, 1 row affected (0.026 sec)

MariaDB [students]> insert into info values(101, 'Aayush', '8/54 bagichi gali');
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into info values(102, 'Sonu', '7/54 bagichi gali');
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into info values(103, 'vinay', '6/54 bagichi gali');
Query OK, 1 row affected (0.002 sec)

MariaDB [students]> insert into info values(104, 'nimit', '5/54 bagichi gali');
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> create table marks(student_id int, total_marks int);
Query OK, 0 rows affected (0.029 sec)

MariaDB [students]> insert into marks values(100,90);
Query OK, 1 row affected (0.002 sec)

MariaDB [students]> insert into marks values(101,99);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(102,98);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(103,97);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(104,96);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(103,95);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(106,89);
Query OK, 1 row affected (0.002 sec)

MariaDB [students]> insert into marks values(107,85);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(108,84);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(109,81);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> select info.student_name, marks.total_marks from info
    -> inner join marks on info.student_id = marks.student_id;
+--------------+-------------+
```

```
MariaDB [(none)]> create database students;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use students;
Database changed
MariaDB [students]> create table info(student_id int, student_name varchar(20), student_address varchar(50));
Query OK, 0 rows affected (0.040 sec)

MariaDB [students]> insert into info vlues(100, 'Tushar', '9/54 bagichi gali');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'vlues(100, 'Tushar', '9/54 bagichi gali')' at l
ine 1
MariaDB [students]> insert into info values(100, 'Tushar', '9/54 bagichi gali');
Query OK, 1 row affected (0.026 sec)

MariaDB [students]> insert into info values(101, 'Aayush', '8/54 bagichi gali');
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into info values(102, 'Sonu', '7/54 bagichi gali');
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into info values(103, 'vinay', '6/54 bagichi gali');
Query OK, 1 row affected (0.002 sec)

MariaDB [students]> insert into info values(104, 'nimit', '5/54 bagichi gali');
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> create table marks(student_id int, total_marks int);
Query OK, 0 rows affected (0.029 sec)

MariaDB [students]> insert into marks values(100,90);
Query OK, 1 row affected (0.002 sec)

MariaDB [students]> insert into marks values(101,99);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(102,98);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(103,97);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(104,96);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(103,95);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(106,89);
Query OK, 1 row affected (0.002 sec)

MariaDB [students]> insert into marks values(107,85);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(108,84);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(109,81);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> select info.student_name, marks.total_marks from info
    -> inner join marks on info.student_id = marks.student_id;
+--------------+-------------+
```

VIVA QUESTIONS:

Q.1: What is a JOIN Operation?

Ans:

A join is an SQL operation performed to establish a connection between two or more database tables based on matching columns, thereby creating a relationship between the tables. A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by ⋈.

Q.2: What is the difference between JOIN and PRODUCT operations?

Ans:

S.no	**JOIN Operation**	**PRODUCT Operations**
1.	Natural join is denoted by '⋈'. It is applicable only when there is atleast one attribute common between 2 relations.	Product operation is denoted by X *Y and returns a relation on tuples, whose schema contains all fields of X followed by all fields of Y.
2.	If you will use join, you will have keys which will be able to join rows from first table with rows of second table and depending on your relation, number of rows may vary.	Cartesian product means you will have rows wi each record from one table matched with all rows of second table.
3.	JOIN operation is applied only with a WHERE clause.	In product operations, WHERE clause is not required.

Q.3: What are different types of JOIN operations?

Ans:

There are mainly 3 types of joins in SQL: Inner Join, Outer Join and Cross Join.

- **(INNER) JOIN:** Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN:** Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN:** Returns all records when there is a match in either left or right table.
- **CROSS JOIN:** This join returns records/rows that are multiplication of record number from both the tables means each row on the left table will relate to each row of right table.

Q.4: Difference between RIGHT OUTER JOIN and LEFT OUTER JOIN?

Ans:

S.NO	RIGHT OUTER JOIN	LEFT OUTER JOIN
1.	Fetches all the rows from the table on the right,regardless of whether there are any matching columns in the "left" table.	Fetches all the rows from the table on the left, regardless of whether there are any matching columns in the "right" table.
2.	The result of Right Outer Join can be seen as: Inner Join +all the unmatched rows from the right table.	The result of Left Outer Join can be seen as: Inner Join +all the unmatched rows from the left table.
3.	Unmatched data of the left table is lost.	Unmatched data of the right table is lost.
4.	If left table doesn't have the matching record then for such records left table column will have NULL value in the result.	If right table doesn't have the matching record then for such records right table column will have NULL value in the result.

Q.5: What is an INNER Join?

Ans:

Inner Join returns only those records/rows that match/exists in both the tables. Inner Join clause in SQL creates a new table (not physical) by combining rows that have matching values in two or more tables. This join is based on a logical relationship (or a common field) between the tables and is used to retrieve data that appears in both tables. Syntax for Inner Join is as follows:

```
Select* from table_1 as t1
 inner join table_2 as t2
 on t1.IDcol=t2.IDcol;
```

EXPERIMENT – 10

DATABASE MANAGEMENT SYSTEMS LAB

Aim

Write the SQL queries to create the views.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 10

Aim:

Write the SQL queries to create the views.

Tools Used:

MariaDB

Procedure/ Queries:

A VIEW is a virtual table, through which a selective portion off the data from one or more tables can be seen will stop views do not contain data of their own will stop they are used to restrict access to the database or to hide data complexity. A view is stored as a SELECT statement in the database.DML operations on a view like INSERT, UPDATE, DELETE affects the data in the original table upon which the view is based.

Syntax:

1.

CREATE VIEW view_name AS

SELECT column1,column2 ,..

FROM table_name

Where condition;

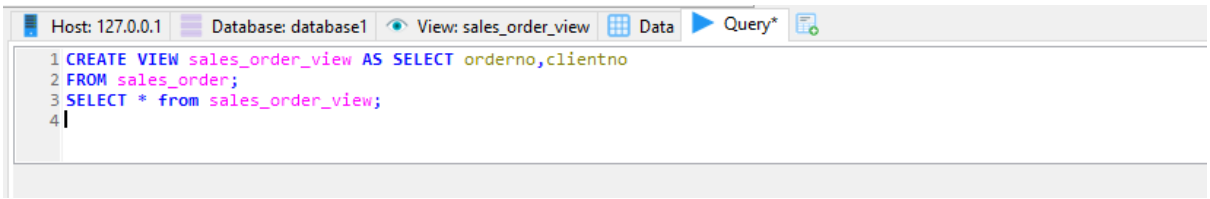
view_name is the name of VIEW.

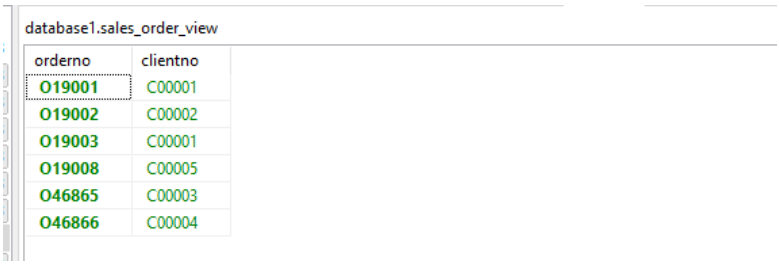
The SELECT statement is used to define the columns and rows that you want to display in the view.

CREATE VIEW sales_order_view **AS SELECT** orderno,clientno

FROM sales_order;

SELECT * **from** sales_order_view;



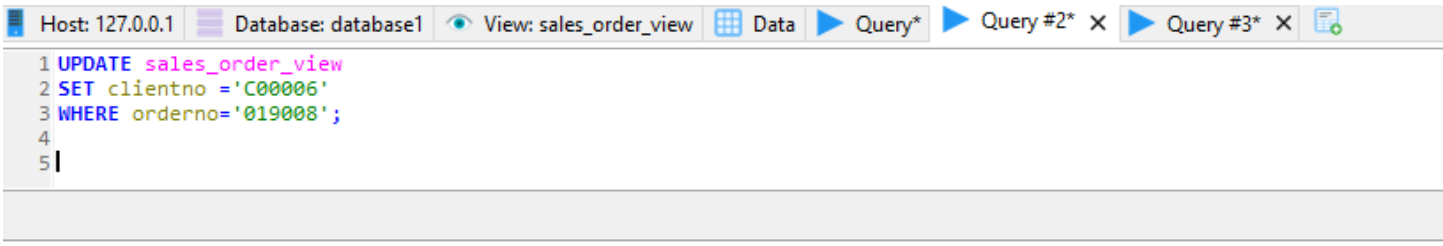


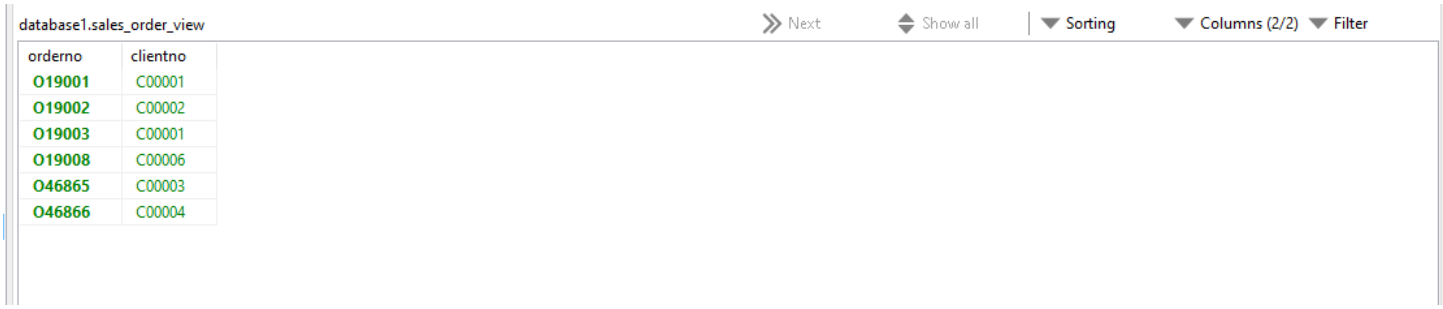
2.

UPDATE sales_order_view

SET clientno ='C00006'

WHERE orderno='019008';





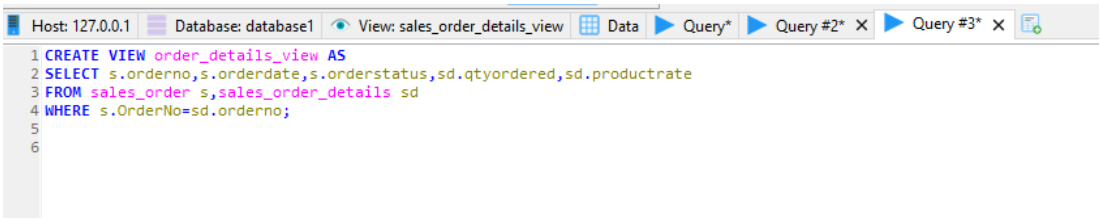
3.

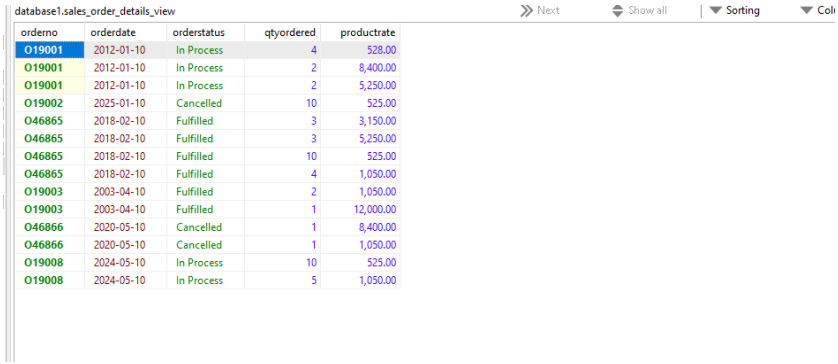
CREATE VIEW_order_details_view **AS**

SELECT s.orderno,s.orderdate,s.orderstatus,sd.qtyordered,sd.productrate

FROM sales_order s,sales_order_details sd

WHERE s.OrderNo=sd.orderno;





orderno	orderdate	orderstatus	qtyordered	productrate
O19001	2012-01-10	In Process	4	528.00
O19001	2012-01-10	In Process	2	8,400.00
O19001	2012-01-10	In Process	2	5,250.00
O19002	2025-01-10	Cancelled	10	525.00
O46865	2018-02-10	Fulfilled	3	3,150.00
O46865	2018-02-10	Fulfilled	3	5,250.00
O46865	2018-02-10	Fulfilled	10	525.00
O46865	2018-02-10	Fulfilled	4	1,050.00
O19003	2003-04-10	Fulfilled	2	1,050.00
O19003	2003-04-10	Fulfilled	1	12,000.00
O46866	2020-05-10	Cancelled	1	8,400.00
O46866	2020-05-10	Cancelled	1	1,050.00
O19008	2024-05-10	In Process	10	525.00
O19008	2024-05-10	In Process	5	1,050.00

Views in SQL are considered as a virtual table. A view also contains rows and columns.

To create the view, we can select the fields from one or more tables present in the database.

A view can either have specific rows based on certain condition or all the rows of a table.

1. Creating view

A view can be created using the **CREATE VIEW** statement. We can create a view from a single table or multiple tables.

Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;
```

2. Creating View from a single table

In this example, we create a View named DetailsView from the table Student_Detail.

Query:

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM Student_Details
WHERE STU_ID < 4;
```

Just like table query, we can query the view to view the data.

```
SELECT * FROM DetailsView;
```

3. Creating View from multiple tables

View from multiple tables can be created by simply include multiple tables in the SELECT statement.

In the given example, a view is created named MarksView from two tables Student_Detail and Student_Marks.

Query:

```
CREATE VIEW MarksView AS
SELECT Student_Detail.NAME, Student_Detail.ADDRESS, Student_Marks.MARKS
FROM Student_Detail, Student_Mark
WHERE Student_Detail.NAME = Student_Marks.NAME;
```

To display data of View MarksView:

```
SELECT * FROM MarksView;
```

4. Deleting View

A view can be deleted using the Drop View statement.

Syntax

```
DROP VIEW view_name;
```

Example:

If we want to delete the View **MarksView**, we can do this as:

```
DROP VIEW MarksView;
```

- **OUTPUT :**

1. Creating a view :

```
MariaDB [lk]> select * from student;
+--------+----------+-----------+---------+-----------+----------+
| stu_id | stu_name | stu_branch | stu_age | Math_marks | hobbies  |
+--------+----------+-----------+---------+-----------+----------+
|      1 | Aman     | CSE        |      20 |         97 | cricket  |
|      2 | Raghav   | Me         |      21 |         87 | football |
|      3 | Rohan    | Cse        |      22 |         90 | singing  |
|      4 | Sarthak  | EEE        |      21 |         77 | dancing  |
|      5 | kiran    | CSE        |      21 |         94 | skating  |
+--------+----------+-----------+---------+-----------+----------+
5 rows in set (0.000 sec)

MariaDB [lk]> create view student_view as
    -> select stu_name,stu_branch,stu_age from student where stu_id<4;
Query OK, 0 rows affected (0.092 sec)
```

```
1
MariaDB [lk]> select * from student_view;
+----------+------------+---------+
| stu_name | stu_branch | stu_age |
+----------+------------+---------+
| Aman     | CSE        |      20 |
| Raghav   | Me         |      21 |
| Rohan    | Cse        |      22 |
+----------+------------+---------+
3 rows in set (0.044 sec)

MariaDB [lk]>
```

2. . Creating View from multiple tables

```
mysql> CREATE VIEW Trainer
    -> AS SELECT c.course_name, c.trainer, t.email
    -> FROM courses c, contact t
    -> WHERE c.id= t.id;
Query OK, 0 rows affected (0.29 sec)

mysql> SELECT * FROM Trainer;
+-------------+---------+------------------------+
| course_name | trainer | email                  |
+-------------+---------+------------------------+
| Java        | Mike    | mike@javatpoint.com    |
| Python      | James   | james@javatpoint.com   |
| Android     | Robin   | robin@javatpoint.com   |
| Hadoop      | Stephen | stephen@javatpoint.com |
| Testing     | Micheal | micheal@javatpoint.com |
+-------------+---------+------------------------+
5 rows in set (0.00 sec)
```

3. Deleting View

```
+----------+------------+----------+
| stu_name | stu_branch | stu_age |
+----------+------------+----------+
| Aman     | CSE        |       20 |
| Raghav   | Me         |       21 |
| Rohan    | Cse        |       22 |
+----------+------------+----------+
3 rows in set (0.232 sec)

MariaDB [lk]> drop student_view;
ERROR 1064 (42000): You have an error in your S
ew' at line 1
MariaDB [lk]> drop view student_view;
Query OK, 0 rows affected (0.016 sec)
```

VIVA QUESTIONS

Q.1: What is a VIEW?

Ans:

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition. They are used to restrict access to the database or to hide data complexity. A view is stored as a SELECT statement in the database. DML operations on a view like INSERT, UPDATE, DELETE affects the data in the original table upon which the view is based.

Q.2: How we create a VIEW?

Ans:

A view is a virtual table based on the result set of an SQL statement. The CREATE VIEW command creates a view. At first, we need to specify the **CREATE VIEW** statement and then we have to give a name to the view. In the second step, we define the **SELECT** statement after the **AS** keyword. Views can be created from a single table, multiple tables or another view. The basic CREATE VIEW syntax is as follows −

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

Q.3: What is the purpose of creating a VIEW?

Ans:

Views are used to limit the visibility of data of the table to just those specific tasks. Also, the view is used for combining the data from multiple tables into a logical table. Views can be used to aggregate rows (using GROUP BY and HAVING) of a table with better detail. The view is used to summarize the data from multiple tables so that views can be used to generate reports. Views can simplify support legacy code. If you need to refactor a table that would break a lot of code, you can replace the table with a view of the same name. The view provides the exact same schema as the original table, while the actual schema has changed. This keeps the legacy code that references the table from breaking, allowing you to change the legacy code at your leisure. Views are used for security purposes because they provide encapsulation of the name of the table. Data is in the virtual table, not stored permanently. Views display only selected data.

Q.4: What is the syntax of creating a VIEW?

Ans:

The basic CREATE VIEW syntax is as follows −

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

Q.5: How many VIEWS can be created for a table?

Ans:

Database users can create multiple VIEWS from a table, thus one view can aggregate data from other views.