INTERVIEW QUESTIONS ON SQL QUERIES

1. What is MySQL?

MySQL is a database management system for web servers. It can grow with the website as it is highly scalable. Most of the websites today are powered by MySQL.

2. What are some of the advantages of using MySQL?

- Flexibility: MySQL runs on all operating systems
- Power: MySQL focuses on performance
- Enterprise-Level SQL Features: MySQL had for some time been lacking in advanced features such as subqueries, views, and stored procedures.
- Full-Text Indexing and Searching
- Query Caching: This helps enhance the speed of MySQL greatly
- Replication: One MySQL server can be duplicated on another, providing numerous advantages
- Configuration and Security

3. What do you mean by 'databases'?

A database is a structured collection of data stored in a computer system and organized in a way to be quickly searched. With databases, information can be rapidly retrieved.

4. What does SQL in MySQL stand for?

The SQL in MySQL stands for Structured Query Language. This language is also used in other databases such as Oracle and Microsoft SQL Server. One can use commands such as the following to send requests from a database:

```
SELECT title FROM publications WHERE author = ' J. K. Rowling';
```

Note that SQL is not case sensitive. However, it is a good practice to write the SQL keywords in CAPS and other names and variables in a small case.

5. What does a MySQL database contain?

A MySQL database contains one or more tables, each of which contains records or rows. Within these rows are various columns or fields that contain the data itself.

6. How can you interact with MySQL?

There are three main ways you can interact with MySQL:

- using a command line
- via a web interface
- through a programming language

7. What are MySQL Database Queries?

A query is a specific request or a question. One can query a database for specific information and have a record returned.

8. What are some of the common MySQL commands?

Command	**Action**
ALTER	To alter a database or table
BACKUP	To back-up a table
\c	To cancel Input
CREATE	To create a database
DELETE	To delete a row from a table
DESCRIBE	To describe a table's columns
DROP	To delete a database or table
EXIT(ctrl+c)	To exit
GRANT	To change user privileges
HELP (\h, \?)	Display help
INSERT	Insert data
LOCK	Lock table(s)
QUIT(\q)	Same as EXIT
RENAME	Rename a Table
SHOW	List details about an object
SOURCE	Execute a file
STATUS (\s)	Display the current status

Command	Action
TRUNCATE	Empty a table
UNLOCK	Unlock table(s)
UPDATE	Update an existing record
USE	Use a database

9. How do you create a database in MySQL?

Use the following command to create a new database called 'books':

```
CREATE DATABASE books;
```

10. How do you create a table using MySQL?

Use the following to create a table using MySQL:

```
CREATE TABLE history (
author VARCHAR(128),
title VARCHAR(128),
type VARCHAR(16),
year CHAR(4)) ENGINE InnoDB;
```

11. How do you Insert Data Into MySQL?

The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

If we want to add values for all the columns of the table, we do not need to specify the column names in the SQL query. However, the order of the values should be in the same order as the columns in the table. The INSERT INTO syntax would be as follows:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

12. How do you remove a column from a database?

You can remove a column by using the DROP keyword:

```
ALTER TABLE classics DROP pages;
```

13. How to create an Index in MySQL?

In MySQL, there are different index types, such as a regular INDEX, a PRIMARY KEY, or a FULLTEXT index. You can achieve fast searches with the help of an index. Indexes speed up performance by either ordering the data on disk so it's quicker to find your result or, telling the SQL engine where to go to find your data.

Example: Adding indexes to the history table:

```
ALTER TABLE history ADD INDEX(author(10));

ALTER TABLE history ADD INDEX(title(10));

ALTER TABLE history ADD INDEX(category(5));

ALTER TABLE history ADD INDEX(year);

DESCRIBE history;
```

14. How to Delete Data From a MySQL Table?

In MySQL, the DELETE statement is used to delete records from a table:

```
DELETE FROM table_name

WHERE column_name = value_name
```

15. How do you view a database in MySQL?

One can view all the databases on the MySQL server host using the following command:

```
mysql> SHOW DATABASES;
```

16. What are the Numeric Data Types in MySQL?

MySQL has numeric data types for integer, fixed-point, floating-point, and bit values, as shown in the table below. Numeric types can be signed or unsigned, except BIT. A special attribute enables the automatic generation of sequential integer or floating-point column values, which is useful for applications that require a series of unique identification numbers.

Type Name	**Meaning**
TINYINT	Very Small Integer
SMALLINT	Small Integer
MEDIUMINT	Medium-sized Integer
INT	Standard Integer

Type Name	Meaning
BIGINT	Large Integer
DECIMAL	Fixed-point number
FLOAT	Single-precision floating-point number
DOUBLE	Double-precision floating-point number
BIT	Bit-field

17. What are the String Data Types in MySQL?

Type Name	Meaning
CHAR	fixed-length nonbinary(character) string
VARCHAR	variable-length nonbinary string
BINARY	fixed-length binary string
VARBINARY	variable-length binary string
TINYBLOB	Very small BLOB(binary large object)
BLOB	Small BLOB
MEDIUMBLOB	Medium-sized BLOB
LONGBLOB	Large BLOB
TINYTEXT	A very small nonbinary string
TEXT	Small nonbinary string
MEDIUMTEXT	Medium-sized nonbinary string
LONGTEXT	Large nonbinary string

Type Name	Meaning
ENUM	An enumeration; each column value is assigned, one enumeration member
SET	A set; each column value is assigned zero or more set members
NULL	NULL in SQL is the term used to represent a missing value. A NULL value in a table is a value in a field that appears to be blank. This value is different than a zero value or a field that contains spaces.

18. What are the Temporal Data Types in MySQL?

Type Name	Meaning
DATE	A date value, in ' CCYY-MM-DD ' Format
TIME	A Time value, in ' hh : mm :ss ' format
DATETIME	Date and time value, in ' CCYY-MM-DD hh : mm :ss ' format
TIMESTAMP	A timestamp value, in ' CCYY-MM-DD hh : mm :ss ' format
YEAR	A year value, in CCYY or YY format

Example: To select the records with an Order Date of "2018-11-11" from a table:

```
SELECT * FROM Orders WHERE OrderDate='2018-11-11'
```

19. What is BLOB in MySQL?

BLOB is an acronym that stands for a binary large object. It is used to hold a variable amount of data.
There are four types of BLOB:

- TINYBLOB
- BLOB
- MEDIUMBLOB
- LONGBLOB

A BLOB can hold a very large amount of data. For example - documents, images, and even videos. You could store your complete novel as a file in a BLOB if needed.

20. How to add users in MySQL?

You can add a User by using the CREATE command and specifying the necessary credentials. For example:

```
CREATE USER ‘testuser’ IDENTIFIED BY ‘sample password’;
```

21. What are MySQL “Views”?

In MySQL, a view consists of a set of rows that is returned if a particular query is executed. This is also known as a ‘virtual table’. Views make it easy to retrieve the way of making the query available via an alias.
The advantages of views are:

- Simplicity
- Security
- Maintainability

22. How do you create and execute views in MySQL?

Creating a view is accomplished with the CREATE VIEW statement. As an example:

```
CREATE
  [OR REPLACE]
  [ALGORITHM = {MERGE | TEMPTABLE | UNDEFINED }]
  [DEFINER = { user | CURRENT_USER }]
  [SQL SECURITY { DEFINER | INVOKER }]
  VIEW view_name [(column_list)]
  AS select_statement
  [WITH [CASCADED | LOCAL] CHECK OPTION]
```

23. What are MySQL Triggers?

A trigger is a task that executes in response to some predefined database event, such as after a new row is added to a particular table. Specifically, this event involves inserting, modifying, or deleting table data, and the task can occur either prior to or immediately following any such event.
Triggers have many purposes, including:

- Audit Trails
- Validation
- Referential integrity enforcement

24. How many Triggers are possible in MySQL?

There are six Triggers allowed to use in the MySQL database:

- Before Insert
- After Insert
- Before Update
- After Update
- Before Delete
- After Delete

25. What is the MySQL server?

The server, mysqld, is the hub of a MySQL installation; it performs all manipulation of databases and tables.

26. What are the MySQL clients and utilities?

Several MySQL programs are available to help you communicate with the server. For administrative tasks, some of the most important ones are listed here:

• **mysql**—An interactive program that enables you to send SQL statements to the server and to view the results. You can also use mysql to execute batch scripts (text files containing SQL statements).

• **mysqladmin**—An administrative program for performing tasks such as shutting down the server, checking its configuration, or monitoring its status if it appears not to be functioning properly.

• **mysqldump**—A tool for backing up your databases or copying databases to another server.

• **mysqlcheck and myisamchk**—Programs that help you perform table checking, analysis, and optimization, as well as repairs if tables become damaged. mysqlcheck works with MyISAM tables and to some extent with tables for other storage engines. myisamchk is for use only with MyISAM tables.

27. What are the types of relationships used in MySQL?

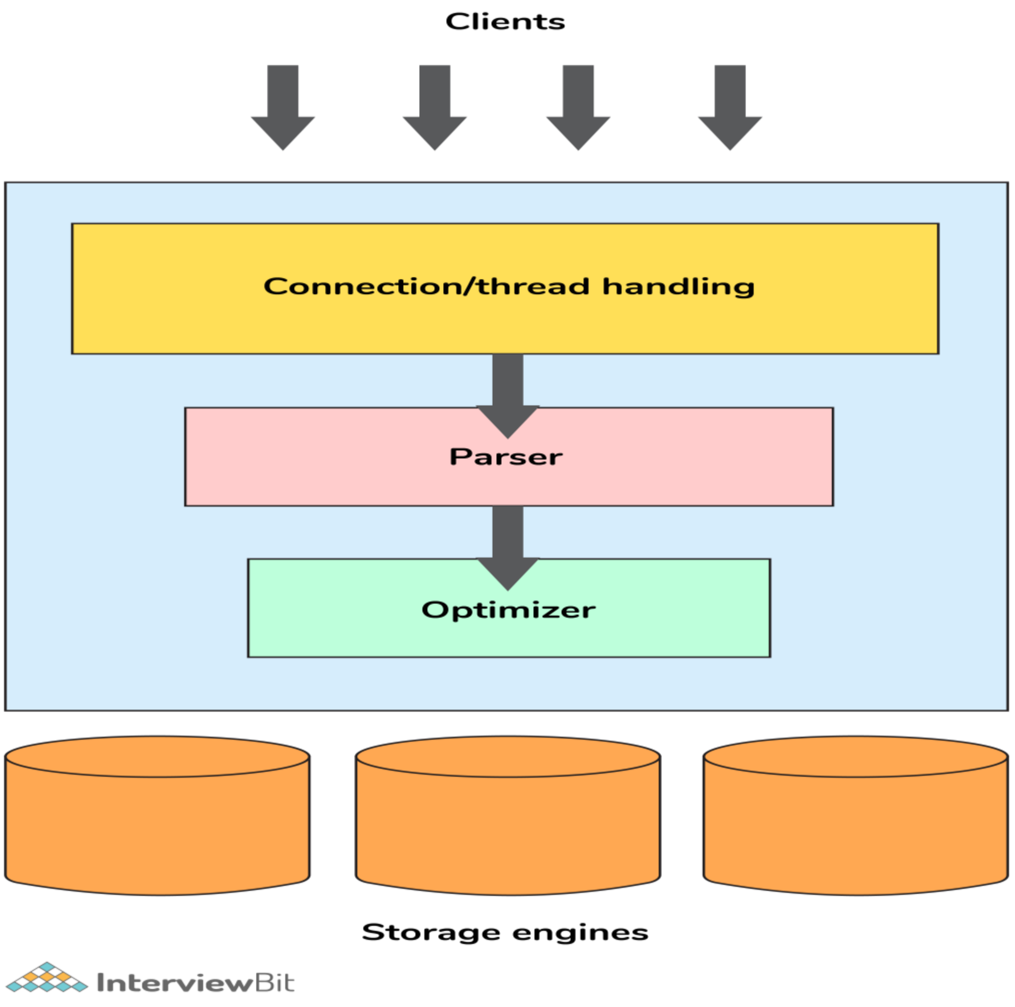
There are three categories of relationships in MySQL:

- **One-to-One**: Usually, when two items have a one-to-one relationship, you just include them as columns in the same table.
- **One-to-Many**: One-to-many (or many-to-one) relationships occur when one row in one table is linked to many rows in another table.
- **Many-to-Many**: In a many-to-many relationship, many rows in one table are linked to many rows in another table. To create this relationship, add a third table containing the same key column from each of the other tables

28. Can you explain the logical architecture of MySQL?

The top layer contains the services most network-based client/server tools or servers need such as connection handling, authentication, security, and so forth. The second layer contains much of MySQL's brains. This has the code for query parsing, analysis, optimization, caching, and all the built-in functions.

The third layer contains the storage engines that are responsible for storing and retrieving the data stored in MySQL.



29. What is Scaling in MySQL?

In MySQL, scaling capacity is actually the ability to handle the load, and it's useful to think of load from several different angles such as:

- Quantity of data
- Number of users
- User activity
- Size of related datasets
-

What is Sharding in SQL?

The process of breaking up large tables into smaller chunks (called shards) that are spread across multiple servers is called Sharding.
The advantage of Sharding is that since the sharded database is generally much smaller than the original; queries, maintenance, and all other tasks are much faster.

31. What are Transaction Storage Engines in MySQL?

To be able to use MySQL's transaction facility, you have to be using MySQL's InnoDB storage engine (which is the default from version 5.5 onward). If you are not sure which version of MySQL your code will be running on, rather than assuming InnoDB is the default engine you can force its use when creating a table, as follows

Employee Info Table:

EmpID	EmpFname	EmpLname	Department	Project	Address	DOB	Gender
1	Sanjay	Mehra	HR	P1	Hyderabad(HYD)	01/12/1976	M
2	Ananya	Mishra	Admin	P2	Delhi(DEL)	02/05/1968	F
3	Rohan	Diwan	Account	P3	Mumbai(BOM)	01/01/1980	M
4	Sonia	Kulkarni	HR	P1	Hyderabad(HYD)	02/05/1992	F
5	Ankit	Kapoor	Admin	P2	Delhi(DEL)	03/07/1994	M

Employee Position Table:

EmpID	EmpPosition	DateOfJoining	Salary
1	Manager	01/05/2022	500000
2	Executive	02/05/2022	75000
3	Manager	01/05/2022	90000
2	Lead	02/05/2022	85000
1	Executive	01/05/2022	300000

- Write a query to fetch the EmpFname from the EmployeeInfo table in the upper case and use the ALIAS name as EmpName.
- Write a query to fetch the number of employees working in the department ‘HR’.
- Write a query to get the current date.
- Write a query to retrieve the first four characters of EmpLname from the EmployeeInfo table.
- Write a query to fetch only the place name(string before brackets) from the Address column of EmployeeInfo table.
- Write a query to create a new table that consists of data and structure copied from the other table.
- Write q query to find all the employees whose salary is between 50000 to 100000.
- Write a query to find the names of employees that begin with ‘S’
- Write a query to fetch top N records.
- Write a query to retrieve the EmpFname and EmpLname in a single column as “FullName”. The first name and the last name must be separated with space.

Q1. Write a query to fetch the EmpFname from the EmployeeInfo table in upper case and use the ALIAS name as EmpName.

```
SELECT UPPER(EmpFname) AS EmpName FROM EmployeeInfo;
```

Q2. Write a query to fetch the number of employees working in the department ‘HR’.

```
SELECT COUNT(*) FROM EmployeeInfo WHERE Department = 'HR';
```

Q3. Write a query to get the current date.

You can write a query as follows in SQL Server:

```
SELECT GETDATE();
```

You can write a query as follows in MySQL:

```
SELECT SYSTDATE();
```

Q4. Write a query to retrieve the first four characters of EmpLname from the EmployeeInfo table.

```
SELECT SUBSTRING(EmpLname, 1, 4) FROM EmployeeInfo;
```

Q5. Write a query to fetch only the place name(string before brackets) from the Address column of EmployeeInfo table.

Using the MID function in MySQL

```
SELECT MID(Address, 0, LOCATE('(',Address)) FROM EmployeeInfo;
```

Using SUBSTRING

```
SELECT SUBSTRING(Address, 1, CHARINDEX('(',Address)) FROM EmployeeInfo;
```

Q6. Write a query to create a new table which consists of data and structure copied from the other table.

Using the SELECT INTO command:

```
SELECT * INTO NewTable FROM EmployeeInfo WHERE 1 = 0;
```

Using the CREATE command in MySQL:

```
CREATE TABLE NewTable AS SELECT * FROM EmployeeInfo;
```

Q7. Write q query to find all the employees whose salary is between 50000 to 100000.

```
SELECT * FROM EmployeePosition WHERE Salary BETWEEN '50000' AND '100000';
```

Q8. Write a query to find the names of employees that begin with 'S'

```
SELECT * FROM EmployeeInfo WHERE EmpFname LIKE 'S%';
```

Q9. Write a query to fetch top N records.

By using the TOP command in SQL Server:

```
SELECT TOP N * FROM EmployeePosition ORDER BY Salary DESC;
```

By using the LIMIT command in MySQL:

```
SELECT * FROM EmpPosition ORDER BY Salary DESC LIMIT N;
```

Q10. Write a query to retrieve the EmpFname and EmpLname in a single column as “FullName”. The first name and the last name must be separated with space.

```
SELECT CONCAT(EmpFname, ' ', EmpLname) AS 'FullName' FROM EmployeeInfo;
```

Q11. Write a query find number of employees whose DOB is between 02/05/1970 to 31/12/1975 and are grouped according to gender

```
SELECT COUNT(*), Gender FROM EmployeeInfo WHERE DOB BETWEEN '02/05/1970 ' AN
```

Q12. Write a query to fetch all the records from the EmployeeInfo table ordered by EmpLname in descending order and Department in the ascending order.

To order the records in ascending and descnding order, you have to use the ORDER BY statement in SQL.

```
SELECT * FROM EmployeeInfo ORDER BY EmpFname desc, Department asc;
```

Q13. Write a query to fetch details of employees whose EmpLname ends with an alphabet ‘A’ and contains five alphabets.

To fetch details mathcing a certain value, you have to use the LIKE operator in SQL.

```
SELECT * FROM EmployeeInfo WHERE EmpLname LIKE '____a';
```

Q14. Write a query to fetch details of all employees excluding the employees with first names, “Sanjay” and “Sonia” from the EmployeeInfo table.

```
SELECT * FROM EmployeeInfo WHERE EmpFname NOT IN ('Sanjay','Sonia');
```

Q15. Write a query to fetch details of employees with the address as “DELHI(DEL)”.

```
SELECT * FROM EmployeeInfo WHERE Address LIKE 'DELHI(DEL)%';
```

Q16. Write a query to fetch all employees who also hold the managerial position.

```
SELECT E.EmpFname, E.EmpLname, P.EmpPosition
FROM EmployeeInfo E INNER JOIN EmployeePosition P ON
E.EmpID = P.EmpID AND P.EmpPosition IN ('Manager');
```

Q17. Write a query to fetch the department-wise count of employees sorted by department's count in ascending order.

```
SELECT Department, count(EmpID) AS EmpDeptCount
FROM EmployeeInfo GROUP BY Department
ORDER BY EmpDeptCount ASC;
```

Q18. Write a query to calculate the even and odd records from a table.

To retrieve the even records from a table, you have to use the MOD() function as follows:

```
SELECT EmpID FROM (SELECT rowno, EmpID from EmployeeInfo) WHERE MOD(rowno,
```

Similarly, to retrieve the odd records from a table, you can write a query as follows:

```
SELECT EmpID FROM (SELECT rowno, EmpID from EmployeeInfo) WHERE MOD(rowno,
```

Q19. Write a SQL query to retrieve employee details from EmployeeInfo table who have a date of joining in the EmployeePosition table.

```
SELECT * FROM EmployeeInfo E
WHERE EXISTS
(SELECT * FROM EmployeePosition P WHERE E.EmpId = P.EmpId);
```

Q20. Write a query to retrieve two minimum and maximum salaries from the EmployeePosition table.

To retrieve two minimum salaries, you can write a query as below:

```
SELECT DISTINCT Salary FROM EmployeePosition E1
 WHERE 2 >= (SELECTCOUNT(DISTINCT Salary)FROM EmployeePosition E2
  WHERE E1.Salary >= E2.Salary) ORDER BY E1.Salary DESC;
```

To retrieve two maximum salaries, you can write a query as below:

```
SELECT DISTINCT Salary FROM EmployeePosition E1
 WHERE 2 >= (SELECTCOUNT(DISTINCT Salary) FROM EmployeePosition E2
  WHERE E1.Salary <= E2.Salary) ORDER BY E1.Salary DESC;
```

Q21. Write a query to find the Nth highest salary from the table without using TOP/limit keyword.

```
SELECT Salary
FROM EmployeePosition E1
WHERE N-1 = (
    SELECT COUNT( DISTINCT ( E2.Salary ) )
    FROM EmployeePosition E2
    WHERE E2.Salary >  E1.Salary );
```

Q22. Write a query to retrieve duplicate records from a table.

```
SELECT EmpID, EmpFname, Department COUNT(*)
FROM EmployeeInfo GROUP BY EmpID, EmpFname, Department
HAVING COUNT(*) > 1;
```

Q23. Write a query to retrieve the list of employees working in the same department.

```
Select DISTINCT E.EmpID, E.EmpFname, E.Department
FROM EmployeeInfo E, Employee E1
WHERE E.Department = E1.Department AND E.EmpID != E1.EmpID;
```

Q24. Write a query to retrieve the last 3 records from the EmployeeInfo table.

```
SELECT * FROM EmployeeInfo WHERE
EmpID <=3 UNION SELECT * FROM
(SELECT * FROM EmployeeInfo E ORDER BY E.EmpID DESC)
AS E1 WHERE E1.EmpID <=3;
```

Q25. Write a query to find the third-highest salary from the EmpPosition table.

```
SELECT TOP 1 salary
FROM(
SELECT TOP 3 salary
FROM employee_table
ORDER BY salary DESC) AS emp
ORDER BY salary ASC;
```

Q26. Write a query to display the first and the last record from the EmployeeInfo table.

To display the first record from the EmployeeInfo table, you can write a query as follows:

```
SELECT * FROM EmployeeInfo WHERE EmpID = (SELECT MIN(EmpID)
FROM EmployeeInfo);
```

To display the last record from the EmployeeInfo table, you can write a query as follows:

```
SELECT * FROM EmployeeInfo WHERE EmpID = (SELECT MAX(EmpID)
FROM EmployeeInfo);
```

Q27. Write a query to add email validation to your database

```
SELECT Email FROM EmployeeInfo WHERE NOT REGEXP
_LIKE(Email, ‘[A-Z0-9._%+-]+@[A-Z0-9.-]+.[A-Z]{2,4}’, ‘i’);
```

Q28. Write a query to retrieve Departments who have less than 2 employees working in it.

```
SELECT DEPARTMENT, COUNT(EmpID) as 'EmpNo' FROM EmployeeInfo GROUP BY
DEPARTMENT HAVING COUNT(EmpD) < 2;
```

Q29. Write a query to retrieve EmpPostion along with total salaries paid for each of them.

```
SELECT EmpPosition, SUM(Salary) from EmployeePosition GROUP BY EmpPosition;
```

Q30. Write a query to fetch 50% records from the EmployeeInfo table.

```
SELECT *
FROM EmployeeInfo WHERE
EmpID <= (SELECT COUNT(EmpID)/2 from EmployeeInfo);
```

2) In which language MySQL has been written?

MySQL is written in C and C++, and its SQL parser is written in yacc.

3) What are the technical specifications of MySQL?

MySQL has the following technical specifications -

- Flexible structure
- High performance
- Manageable and easy to use
- Replication and high availability
- Security and storage management
- Drivers
- Graphical Tools
- MySQL Enterprise Monitor
- MySQL Enterprise Security
- JSON Support
- Replication & High-Availability
- Manageability and Ease of Use
- OLTP and Transactions
- Geo-Spatial Support

4) What is the difference between MySQL and SQL?

SQL is known as the standard query language. It is used to interact with the database like MySQL. MySQL is a database that stores various types of data and keeps it safe.

A PHP script is required to store and retrieve the values inside the database.

SQL is a computer language, whereas MySQL is a software or an application

SQL is used for the creation of database management systems whereas MySQL is used to enable data handling, storing, deleting and modifying data

5. What is the difference between the database and the table?

There is a major difference between a database and a table. The differences are as follows:

- Tables are a way to represent the division of data in a database while the database is a collection of tables and data.
- Tables are used to group the data in relation to each other and create a dataset. This dataset will be used in the database. The data stored in the table in any form is a part of the database, but the reverse is not true.
- A database is a collection of organized data and features used to access them, whereas the table is a collection of rows and columns used to store the data.

6) Why do we use the MySQL database server?

First of all, the MYSQL server is free to use for developers and small enterprises.

MySQL server is open source.

MySQL's community is tremendous and supportive; hence any help regarding MySQL is resolved as soon as possible.

MySQL has very stable versions available, as MySQL has been in the market for a long time. All bugs arising in the previous builds have been continuously removed, and a very stable version is provided after every update.

The MySQL database server is very fast, reliable, and easy to use. You can easily use and modify the software. MySQL software can be downloaded free of cost from the internet.

7) What are the different tables present in MySQL?

There are many tables that remain present by default. But, MyISAM is the default database engine used in MySQL. There are five types of tables that are present:

- MyISAM
- Heap
- Merge
- INNO DB
- ISAM

9) How to check the MySQL version?

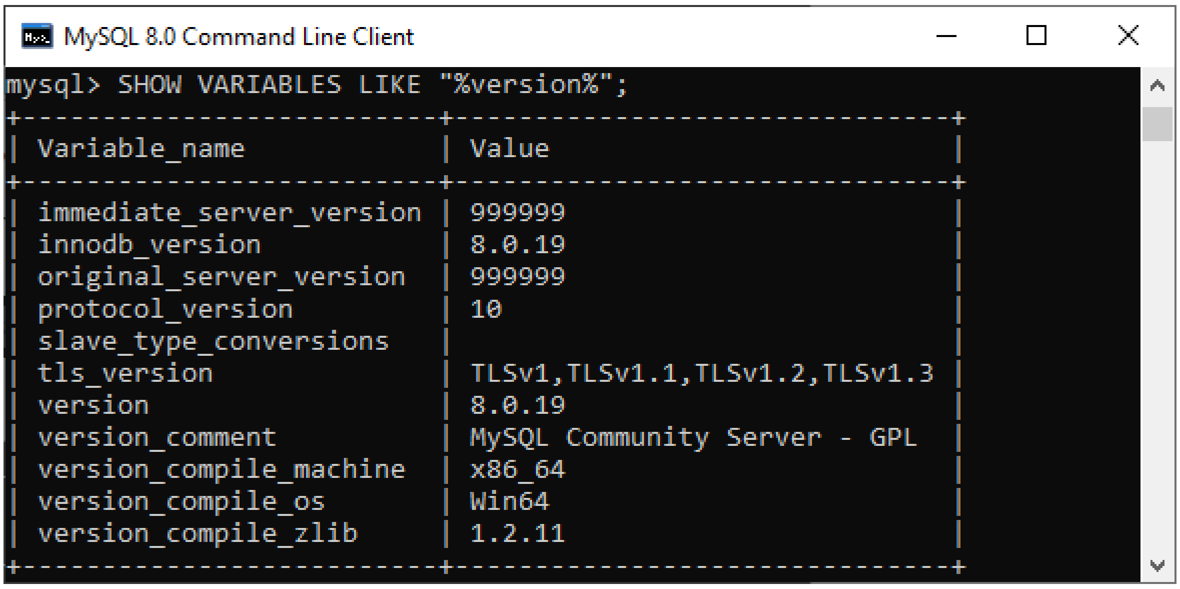
We can check the MySQL version on Linux using the below command:

1. mysql -v

If we use the MySQL in windows, opening the MySQL command-line tool displayed the version information without using any flags. If we want to know more about the server information, use the below statement:

1. SHOW VARIABLES LIKE "%version%";

It will return the output as below:



```
mysql> SHOW VARIABLES LIKE "%version%";
+--------------------------+-------------------------------+
| Variable_name            | Value                         |
+--------------------------+-------------------------------+
| immediate_server_version | 999999                        |
| innodb_version           | 8.0.19                        |
| original_server_version  | 999999                        |
| protocol_version         | 10                            |
| slave_type_conversions   |                               |
| tls_version              | TLSv1,TLSv1.1,TLSv1.2,TLSv1.3 |
| version                  | 8.0.19                        |
| version_comment          | MySQL Community Server - GPL  |
| version_compile_machine  | x86_64                        |
| version_compile_os       | Win64                         |
| version_compile_zlib     | 1.2.11                        |
+--------------------------+-------------------------------+
```

In this output, we can see the additional version information about the installed MySQL software like innodb_version, protocol_version, version_ssl_library, etc.

10) How to add columns in MySQL?

A column is a series of cells in a table that stores one value for each row in a table. We can add columns in an existing table using the ALTER TABLE statement as follows:

1. **ALTER TABLE** table_name
2. **ADD COLUMN** column_name column_definition [**FIRST**|**AFTER** existin g_column];

11) How to delete a table in MySQL?

We can delete a table in MySQL using the Drop Table statement. This statement removes the complete data of a table, including structure and definition from the database permanently. Therefore, it is required to be careful while deleting a table. After using the statement, we cannot recover the table in MySQL. The statement is as follows:

1. **DROP TABLE** table_name;

12) How to add foreign keys in MySQL?

The foreign key is used to link one or more tables together. It matches the primary key field of another table to link the two tables. It allows us to create a parent-child relationship with the tables. We can add a foreign key to a table in two ways:

- Using the CREATE TABLE Statement
- Using the ALTER TABLE Statement

Following is the syntax to define a foreign key using CREATE TABLE OR ALTER TABLE statement:

1. [**CONSTRAINT** constraint_name]
2. **FOREIGN KEY** [foreign_key_name] (col_name, ...)
3. **REFERENCES** parent_tbl_name (col_name,...) .

13) How to connect to the MySQL database?

MySQL allows us to connect with the database server in mainly two ways:

Using Command-line Tool

We can find the command-line client tool in the bin directory of the MySQL's installation folder. To invoke this program, we need to navigate the installation folder's bin directory and type the below command:

1. mysql

Next, we need to run the below command to connect to the MySQL Server:

1. shell>mysql -u root -p

Finally, type the password for the selected user account root and press Enter:

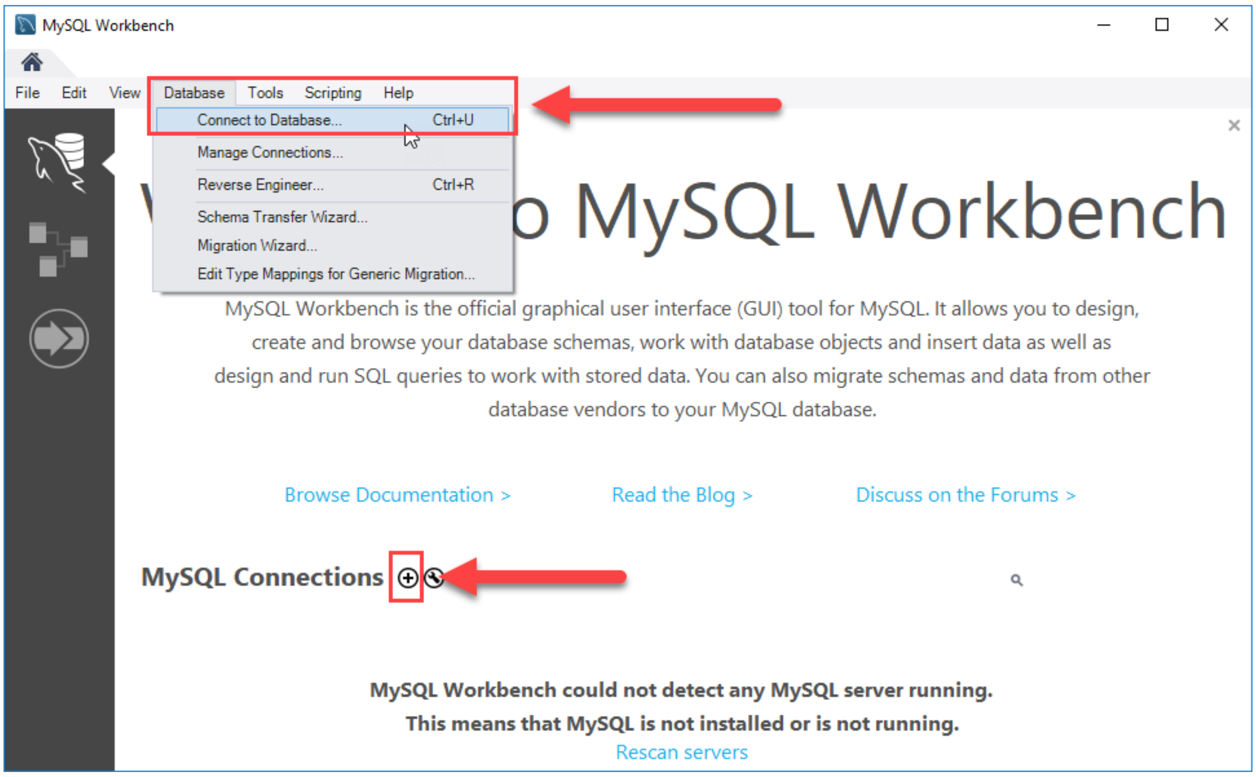
1. Enter **password**: ********

After successful connection, we can use the below command to use the:

1. USE database_name;

Using MySQL Workbench

We can make a connection with database using MySQL Workbench, simply clicking the plus (+) icon or navigating to the menu bar -> Database -> Connect to Database, the following screen appears. Now, you need to fill all the details to make a connection:



MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

MySQL Workbench could not detect any MySQL server running.
This means that MySQL is not installed or is not running.

Once we finished this setup, it will open the MySQL Workbench screen. Now, we can double click on the newly created connection to connect with the database server.

14) How to change the MySQL password?

We can change the MySQL root password using the below statement in the new notepad file and save it with an appropriate name:

1. **ALTER** USER 'root'@'localhost' IDENTIFIED **BY** 'NewPassword';

Next, open a Command Prompt and navigate to the MySQL directory. Now, copy the following folder and paste it in our DOS command and press the Enter key.

1. C:\Users\javatpoint> CD C:\Program Files\MySQL\MySQL Server 8.0\bin

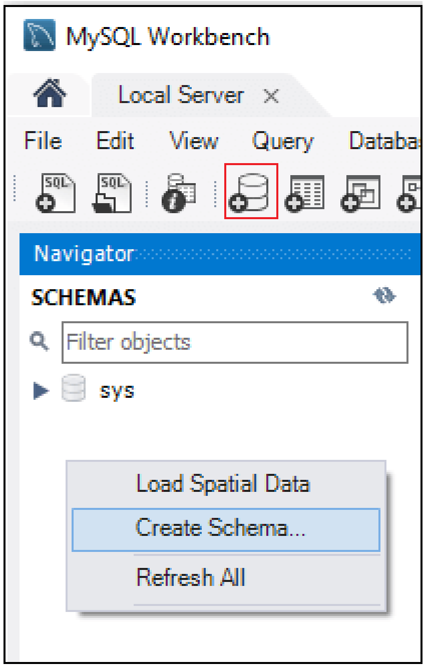
Next, enter this statement to change the password:

1. mysqld --init-file=C:\\mysql-notepadfile.txt

Finally, we can log into the MySQL server as root using this new password. After launches the MySQL server, it is to delete the C:\myswl-init.txt file to ensure the password change.

15) How to create a database in MySQL Workbench?

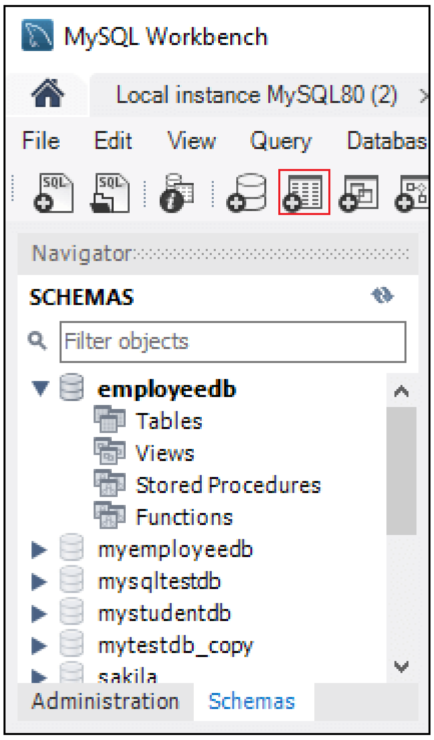
To create a new database in MySQL Workbench, we first need to launch the MySQL Workbench and log in using the username and password. Go to the Navigation tab and click on the Schema menu. Right-click under the Schema menu and select Create Schema or click the database icon (red rectangle), as shown in the following screen.



A new popup screen appears where we need to fill all the details. After entering the details, click on the Apply button and then the Finish button to complete the database creation.

16) How to create a table in MySQL Workbench?

Launch the MySQL Workbench and go to the Navigation tab and click on the Schema menu where all the previously created databases are shown. Select any database and double click on it. It will show the sub-menus where we need to select the Tables option.



Select Tables sub-menu, right-click on it and select Create Table option. We can also click on create a new table icon (shown in red rectangle) to create a table. It will open the new popup screen where we need to fill all the details to create a table. Here, we will enter the table name and column details. After entering the details, click on the Apply button and then the Finish button to complete the table creation.

17) How to change the table name in MySQL?

Sometimes our table name is non-meaningful. In that case, we need to change or rename the table name. MySQL provides the following syntax to rename one or more tables in the current database:

1. mysql> RENAME old_table **TO** new_table;

If we want to change more than one table name, use the below syntax:

1. RENAME **TABLE** old_tab1 **TO** new_tab1,
2. old_tab2 **TO** new_tab2, old_tab3 **TO** new_tab3;

18) How to change the database name in MySQL?

Sometimes we need to change or rename the database name because of its non-meaningful name. To rename the database name, we need first to create a new database into the MySQL server. Next, MySQL provides the mysqldump shell command to create a dumped copy of the selected database and then import all the data into the newly created database. The following is the syntax of using mysqldump command:

1. mysqldump -u username -p "password" -R oldDbName > oldDbName.sql

Now, use the below command to import the data into the newly created database:

1. mysql -u username -p"password" newDbName < oldDbName.sql

19) How to import a database in MySQL?

Importing database in MySQL is a process of moving data from one place to another place. It is a very useful method for backing up essential data or transferring our data between different locations. For example, we have a contact book database, which is essential to keep it in a secure place. So we need to export it in a safe place, and whenever it lost from the original location, we can restore it using import options.

In MySQL, we can import a database in mainly two ways:

- Command Line Tool
- MySQL Workbench

20) How to change the column name in MySQL?

While creating a table, we have kept one of the column names incorrectly. To change or rename an existing column name in MySQL, we need to use the ALTER TABLE and CHANGE commands together. The following are the syntax used to rename a column in MySQL:

1. **ALTER TABLE** table_name
2. CHANGE **COLUMN** old_column_name new_column_name column_definition [**FIRST**|**AFTER** existing_column];

Suppose the column's current name is S_ID, but we want to change this with a more appropriate title as Stud_ID. We will use the below statement to change its name:

1. **ALTER TABLE** Student CHANGE **COLUMN** S_ID Stud_ID **varchar**(10);

21) How to delete columns in MySQL?

We can remove, drop, or delete one or more columns in an existing table using the ALTER TABLE statement as follows:

1. **ALTER TABLE** table_name **DROP COLUMN** column_name1, column_name2....;

22) How to insert data in MySQL?

We can insert data in a MySQL table using the INSERT STATEMENT. This statement allows us to insert single or multiple rows into a table. The following is the basic syntax to insert a record into a table:

1. **INSERT INTO** table_name (field1, field2,...fieldN)
2. **VALUES** (value1, value2,...valueN);

If we want to insert more than one rows into a table, use the below syntax:

1. **INSERT INTO** **table**(field1, field2,...fieldN)
2. **VALUES**
3. (value1, value 2, ...),
4. (value1, value2, ...),
5. ...
6. (value1, value2, ...);

23) How to delete a row in MySQL?

We can delete a row from the MySQL table using the DELETE STATEMENT within the database. The following is the generic syntax of DELETE statement in MySQL to remove one or more rows from a table:

1. **DELETE FROM** table_name **WHERE** Condition_specified;

It is noted that if we have not specified the WHERE clause with the syntax, this statement will remove all the records from the given table.

24) How to join two tables in MySQL?

We can connect two or more tables in MySQL using the JOIN clause. MySQL allows various types of JOIN clauses. These clauses connect multiple tables and return only those records that match the same value and property in all tables. The following are the four easy ways to join two or more tables in MySQL:

- Inner Join
- Left Join
- Right Join
- Cross Join

25) How to join three tables in MySQL?

Sometimes we need to fetch data from three or more tables. There are two types available to do these types of joins. Suppose we have three tables named Student, Marks, and Details.

Let's say Student has (stud_id, name) columns, Marks has (school_id, stud_id, scores) columns, and Details has (school_id, address, email) columns.

1. Using SQL Join Clause

This approach is similar to the way we join two tables. The following query returns result from three tables:

1. **SELECT name**, scores, address, email **FROM** Student s
2. **INNER** JOIN Marks m **on** s.stud_id = m.stud_id
3. **INNER** JOIN Details d **on** d.school_id = m.school_id;

2. Using Parent-Child Relationship

It is another approach to join more than two tables. In the above tables, we have to create a parent-child relationship. First, create column X as a primary key in one table and as a foreign key in another table. Therefore, stud_id is the primary key in the Student table and will be a foreign key in the Marks table. Next, school_id is the primary key in the Marks table and will be a foreign key in the Details table. The following query returns result from three tables:

1. **SELECT name**, scores, address, email
2. **FROM** Student s, Marks m, Details d
3. **WHERE** s.stud_id = m.stud_id AND m.school_id = d.school_id;

26) How to update the table in MySQL?

We can update existing records in a table using the UPDATE statement that comes with the SET and WHERE clauses. The SET clause changes the values of the specified column. The WHERE clause is optional, which is used to specify the condition. This statement can also use to change values in one or more columns of a single row or multiple rows at a time. Following is a generic syntax of UPDATE command to modify data into the MySQL table:

1. **UPDATE** table_name
2. **SET** field1=new-value1, field2=new-value2, ...
3. [**WHERE** Clause]

27) What is MySQL Workbench?

MySQL Workbench is a unified visual database designing or GUI tool used for working on MySQL databases. It is developed and maintained by Oracle that provides SQL development, data migration, and comprehensive administration tools for server configuration, user administration, backup, etc. We can use this Server Administration to create new physical data models, E-R diagrams, and SQL development. It is available for all major operating systems. MySQL provides supports for it from MySQL Server version v5.6 and higher.

It is mainly available in three editions, which are given below:

- Community Edition (Open Source, GPL)
- Standard Edition (Commercial)
- Enterprise Edition (Commercial)

28) How to drop the primary key in MySQL?

MySQL primary key is a single or combination of the field used to identify each record in a table uniquely. A primary key column cannot be null or empty. We can remove or delete a primary key from the table using the ALTER TABLE statement. The following syntax is used to drop the primary key:

1. **ALTER TABLE** table_name **DROP PRIMARY KEY**;

29) How to create a Stored Procedure in MySQL?

A stored procedure is a group of SQL statements that we save in the database. The SQL queries, including INSERT, UPDATE, DELETE, etc. can be a part of the stored procedure. A procedure allows us to use the same code over and over again by executing a single statement. It stores in the database data dictionary.

We can create a stored procedure using the below syntax:

1. **CREATE PROCEDURE** procedure_name [(parameter datatype [, parameter datatype])]
2. **BEGIN**
3. Body_section **of** SQL statements
4. **END**;

This statement can return one or more value through parameters or may not return any result. The following example explains it more clearly:

1. DELIMITER $$
2. **CREATE PROCEDURE** get_student_info()
3. **BEGIN**
4. **SELECT** * **FROM** Student_table;
5. **END**$$

30) How to execute a stored procedure in MySQL?

We can execute a stored procedure in MySQL by simply CALL query. This query takes the name of the stored procedure and any parameters we need to pass to it. The following is the basic syntax to execute a stored procedure:

1. CALL stored_procedure_name (argument_list);

Let's understand it with this example:

1. CALL Product_Pricing (@pricelow, @pricehigh);

Here, a stored procedure named Product_Pricing calculates and returns the lowest and highest product prices.

31) How to create a View in MySQL?

A view is a database object whose values are based on the base table. It is a **virtual table** created by a query by joining one or more tables. It is operated similarly to the base table but does not contain any data of its own. If any changes occur in the underlying table, the same changes reflected in the View also.

Following is the general syntax of creating a VIEW in MySQL:

1. **CREATE** [OR REPLACE] **VIEW** view_name **AS**
2. **SELECT** columns
3. **FROM** tables

4. [**WHERE** conditions];

32) How to create a Trigger in MySQL?

A trigger is a procedural code in a database that automatically invokes whenever certain events on a particular table or view in the database occur. It can be executed when records are inserted into a table, or any columns are being updated. We can create a trigger in MySQL using the syntax as follows:

```
CREATE TRIGGER trigger_name
    [before | after]
   {insert | update | delete}
    ON table_name [FOR EACH ROW]
    BEGIN
        --variable declarations
        --trigger code
    END;
```

33) How to clear screen in MySQL?

If we use MySQL in Windows, it is not possible to clear the screen before version 8. At that time, the Windows operating system provides the only way to clear the screen by exiting the MySQL command-line tool and then again open MySQL.

After the release of MySQL version 8, we can use the below command to clear the command line screen:

```
mysql> SYSTEM CLS;
```

34) How to create a new user in MySQL?

A USER in MySQL is a record in the USER-TABLE. It contains the login information, account privileges, and the host information for MySQL account to access and manage the databases. We can create a new user account in the database server using the MySQL Create User statement. It provides authentication, SSL/TLS, resource-limit, role, and password management properties for the new accounts.

The following is the basic syntax to create a new user in MySQL:

```
CREATE USER [IF NOT EXISTS] account_name IDENTIFIED BY 'password';
```

35) How to check USERS in MySQL?

If we want to manage a database in MySQL, it is required to see the list of all user's accounts in a database server. The following command is used to check the list of all users available in the database server:

```
1. mysql> SELECT USER FROM mysql.user;
```

36) How to import a CSV file in MySQL?

MySQL allows us to import the CSV (comma separated values) file into a database or table. A CSV is a plain text file that contains the list of data and can be saved in a tabular format. MySQL provides the LOAD DATA INFILE statement to import a CSV file. This statement is used to read a text file and import it into a database table very quickly. The full syntax to import a CSV file is given below:

```
1. LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/
   filename.csv'
2. INTO TABLE tablename
3. FIELDS TERMINATED BY ','
4. OPTIONALLY ENCLOSED BY '"'
5. LINES TERMINATED BY '\r\n'
6. IGNORE 1 ROWS;
```

37) How to insert Date in MySQL?

MySQL allows us to use the INSERT STATEMENT to add the date in MySQL table. MySQL provides several data types for storing dates such as DATE, TIMESTAMP, DATETIME, and YEAR. The default format of the date in MySQL is YYYY-MM-DD. Following is the basic syntax to insert date in MySQL table:

```
1. INSERT INTO table_name (column_name, column_date) VALUES ('DATE:
   Manual Date', '2008-7-04');
```

If we want to insert a date in the mm/dd/yyyy format, it is required to use the below statement:

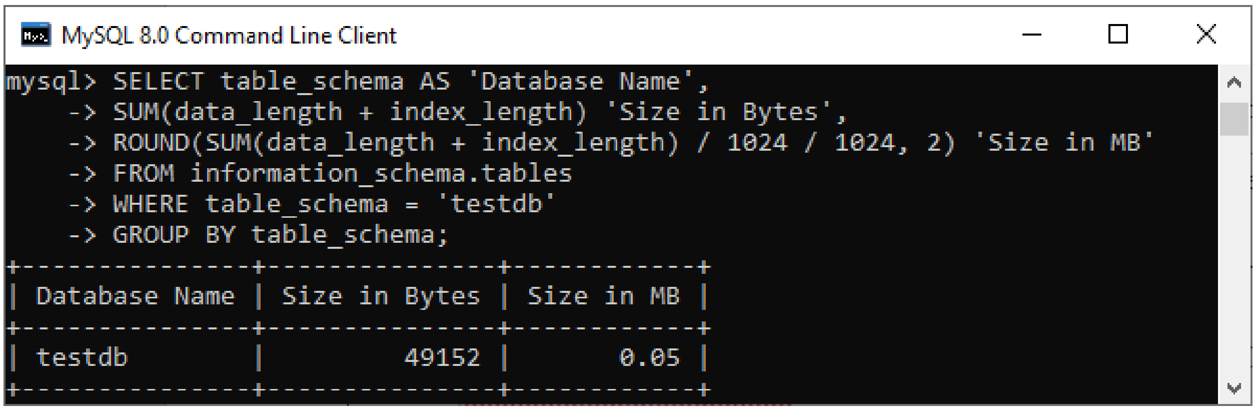
```
1. INSERT INTO table_name VALUES (STR_TO_DATE(date_value, format_s
   pecifier));
```

38) How to check database size in MySQL?

MySQL allows us to query the information_schema.tables table to get the information about the tables and databases. It will return the information about the data length, index length, collation, creation time, etc. We can check the size of the database on the server using the below syntax:

1. **SELECT** table_schema **AS** 'Database Name',
2. SUM(data_length + index_length) 'Size in Bytes',
3. ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) 'Size in MB'
4. **FROM** information_schema.tables
5. **WHERE** table_schema = 'testdb'
6. **GROUP BY** table_schema;

It will return the output as follows:



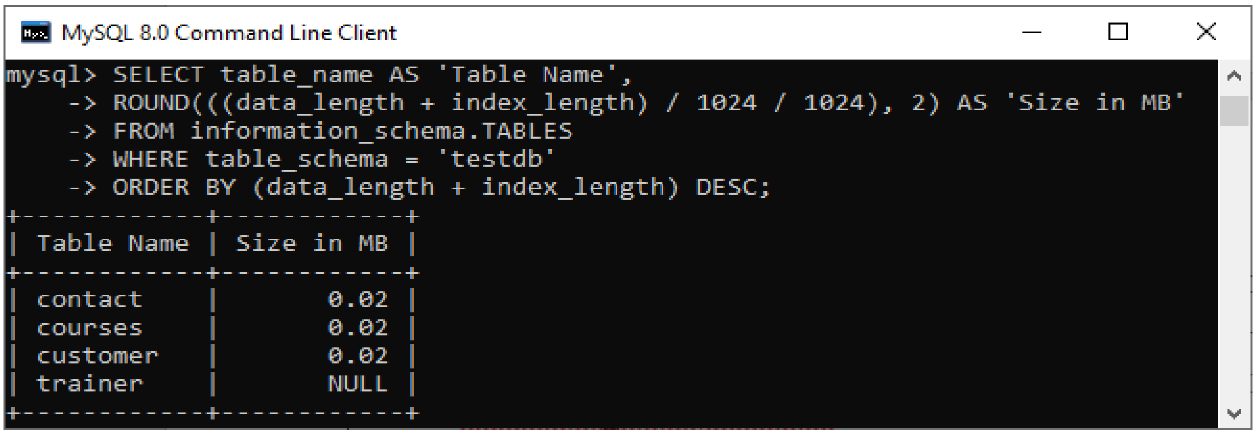
MySQL 8.0 Command Line Client

```
mysql> SELECT table_schema AS 'Database Name',
    -> SUM(data_length + index_length) 'Size in Bytes',
    -> ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) 'Size in MB'
    -> FROM information_schema.tables
    -> WHERE table_schema = 'testdb'
    -> GROUP BY table_schema;
+---------------+---------------+------------+
| Database Name | Size in Bytes | Size in MB |
+---------------+---------------+------------+
| testdb        |         49152 |       0.05 |
+---------------+---------------+------------+
```

If we want to check the size of the table in a specific database, use the following statement:

1. **SELECT** table_name **AS** 'Table Name',
2. ROUND(((data_length + index_length) / 1024 / 1024), 2) **AS** 'Size in MB'
3. **FROM** information_schema.TABLES
4. **WHERE** table_schema = 'testdb'
5. **ORDER BY** (data_length + index_length) **DESC**;

It will return the output as follows:



MySQL 8.0 Command Line Client

```
mysql> SELECT table_name AS 'Table Name',
    -> ROUND(((data_length + index_length) / 1024 / 1024), 2) AS 'Size in MB'
    -> FROM information_schema.TABLES
    -> WHERE table_schema = 'testdb'
    -> ORDER BY (data_length + index_length) DESC;
+------------+------------+
| Table Name | Size in MB |
+------------+------------+
| contact    |       0.02 |
| courses    |       0.02 |
| customer   |       0.02 |
| trainer    |       NULL |
+------------+------------+
```

39) How does indexing works in MySQL?

Indexing is a process to find an unordered list into an ordered list. It helps in maximizing the query's efficiency while searching on tables in MySQL. The working of MySQL indexing is similar to the book index.

Suppose we have a book and want to get information about, say, searching. Without indexing, it is required to go through all pages one by one, until the specific topic was not found. On the other hand, an index contains a list of keywords to find the topic mentioned on pages. Then, we can flip to those pages directly without going through all pages.

40) Who owns MySQL?

MySQL is the most popular free and open-source database software which comes under the GNU General Public License. In the beginning, it was owned and sponsored by the Swedish company MySQL AB. Now, it is bought by Sun Microsystems (now Oracle Corporation), who is responsible for managing and developing the database.

41) How to view the database in MySQL?

Working with the MySQL server, it is a common task to view or list the available databases. We can view all the databases on the MySQL server host using the following command:

1. mysql> SHOW DATABASES;

42) How to set auto increment in MySQL?

Auto Increment is a constraint that automatically generates a unique number while inserting a new record into the table. Generally, it is used for the primary key field in a table. In MySQL, we can set the value for an AUTO_INCREMENT column using the ALTER TABLE statement as follows:

1. **ALTER TABLE** table_name AUTO_INCREMENT = value;

43) How to find the second highest salary in MySQL?

MySQL uses the LIMIT keyword, which can be used to limit the result set. It will allow us to get the first few rows, last few rows, or range of rows. It can also be used to find the second, third, or nth highest salary. It ensures that you have use order by clause to sort the result set first and then print the output that provides accurate results. The following query is used to get the second highest salary in MySQL:

1. **SELECT** salary
2. **FROM** (**SELECT** salary **FROM** employees **ORDER BY** salary **DESC** LIMI T 2) **AS** Emp **ORDER BY** salary LIMIT 1;

There are some other ways to find the second highest salary in MySQL, which are given below:

This statement uses subquery and IN clause to get the second highest salary:

1. **SELECT MAX**(salary)
2. **FROM** employees
3. **WHERE** salary NOT IN (**SELECT Max**(salary) **FROM** employees);

This query uses subquery and < operator to return the second highest salary:

1. **SELECT MAX**(salary) **From** employees
2. **WHERE** salary < (**SELECT Max**(salary) **FROM** employees);

44) What is the difference between TRUNCATE and DELETE in MySQL?

- TRUNCATE is a DDL command, and DELETE is a DML command.
- It is not possible to use Where command with TRUNCATE QLbut you can use it with DELETE command.
- TRUNCATE cannot be used with indexed views, whereas DELETE can be used with indexed views.
- The DELETE command is used to delete data from a table. It only deletes the rows of data from the table while truncate is a very dangerous command and should be used carefully because it deletes every row permanently from a table.

45) How many Triggers are possible in MySQL?

There are only six Triggers allowed to use in the MySQL database.

1. Before Insert
2. After Insert
3. Before Update
4. After Update
5. Before Delete
6. After Delete

46) What is the heap table?

Tables that are present in memory is known as HEAP tables. When you create a heap table in MySQL, you should need to specify the TYPE as HEAP. These tables are commonly known as memory tables. They are used for high-speed storage on a temporary basis. They do not allow BLOB or TEXT fields.

47) What is BLOB and TEXT in MySQL?

BLOB is an acronym that stands for a large binary object. It is used to hold a variable amount of data.

There are four types of the BLOB.

1. TINYBLOB
2. BLOB
3. MEDIUMBLOB
4. LONGBLOB

The differences among all these are the maximum length of values they can hold.

TEXT is a case-insensitive BLOB. TEXT values are non-binary strings (character string). They have a character set, and values are stored and compared based on the collation of the character set.

There are four types of TEXT.

1. TINYTEXT
2. TEXT
3. MEDIUMTEXT
4. LONGTEXT

48) What is a trigger in MySQL?

A trigger is a set of codes that executes in response to some events.

49) What is the difference between the heap table and the temporary table?

Heap tables:

Heap tables are found in memory that is used for high-speed storage temporarily. They do not allow BLOB or TEXT fields.

Heap tables do not support AUTO_INCREMENT.

Indexes should be NOT NULL.

Temporary tables:

The temporary tables are used to keep the transient data. Sometimes it is beneficial in cases to hold temporary data. The temporary table is deleted after the current client session terminates.

Main differences:

The heap tables are shared among clients, while temporary tables are not shared.

Heap tables are just another storage engine, while for temporary tables, you need a special privilege (create temporary table).

50) What is the difference between FLOAT and DOUBLE?

FLOAT stores floating-point numbers with accuracy up to 8 places and allocate 4 bytes. On the other hand, DOUBLE stores floating-point numbers with accuracy up to 18 places and allocates 8 bytes.

51) What are the advantages of MySQL in comparison to Oracle?

1. MySQL is a free, fast, reliable, open-source relational database while Oracle is expensive, although they have provided Oracle free edition to attract MySQL users.
2. MySQL uses only just under 1 MB of RAM on your laptop, while Oracle 9i installation uses 128 MB.
3. MySQL is great for database enabled websites while Oracle is made for enterprises.
4. MySQL is portable.

52) What are the disadvantages of MySQL?

1. MySQL is not so efficient for large scale databases.
2. It does not support COMMIT and STORED PROCEDURES functions version less than 5.0.
3. Transactions are not handled very efficiently.
4. The functionality of MySQL is highly dependent on other addons.
5. Development is not community-driven.

53) What is the difference between CHAR and VARCHAR?

1. CHAR and VARCHAR have differed in storage and retrieval.
2. CHAR column length is fixed, while VARCHAR length is variable.
3. The maximum no. of character CHAR data types can hold is 255 characters, while VARCHAR can hold up to 4000 characters.
4. CHAR is 50% faster than VARCHAR.
5. CHAR uses static memory allocation, while VARCHAR uses dynamic memory allocation.

54) What is the difference between MySQL_connect and MySQL_pconnect?

Mysql_connect:

1. It opens a new connection to the database.
2. Every time you need to open and close the database connection, depending on the request.
3. Opens page whenever it is loaded.

Mysql_pconnect:

1. In Mysql_pconnect, "p" stands for persistent connection, so it opens the persistent connection.
2. The database connection cannot be closed.
3. It is more useful if your site has more traffic because there is no need to open and close connection frequently and whenever the page is loaded.

55) What does "i_am_a_dummy flag" do in MySQL?

The "i_am_a_dummy flag" enables the MySQL engine to refuse any UPDATE or DELETE statement to execute if the WHERE clause is not present. Hence it can save the programmer from deleting the entire table my mistake if he does not use WHERE clause.

56) How to get the current date in MySQL?

To get current date, use the following syntax:

1. **SELECT CURRENT_DATE**();

57) What are the security alerts while using MySQL?

Install antivirus and configure the operating system's firewall.

Never use the MySQL Server as the UNIX root user.

Change the root username and password Restrict or disable remote access.

58) How to change a password for an existing user via mysqladmin?

Mysqladmin -u root -p password "newpassword".

59) What is the difference between UNIX timestamps and MySQL timestamps?

Actually, both Unix timestamp and MySQL timestamp are stored as 32-bit integers, but MySQL timestamp is represented in the readable format of YYYY-MM-DD HH:MM:SS format.

60) How to display the nth highest salary from a table in a MySQL query?

Let us take a table named the employee.

To find Nth highest salary is:

select distinct(salary)from employee order by salary desc limit n-1,1

if you want to find 3rd largest salary:

select distinct(salary)from employee order by salary desc limit 2,1

61) What is the MySQL default port number?

MySQL default port number is 3306.

62) What is REGEXP?

REGEXP is a pattern match using a regular expression. The regular expression is a powerful way of specifying a pattern for a sophisticated search.

Basically, it is a special text string for describing a search pattern. To understand it better, you can think of a situation of daily life when you search for .txt files to list all text files in the file manager. The regex equivalent for .txt will be .*\.txt.

63) How many columns can you create for an index?

You can a create maximum of 16 indexed columns for a standard table.

64) What is the difference between NOW() and CURRENT_DATE()?

NOW() command is used to show current year, month, date with hours, minutes, and seconds while CURRENT_DATE() shows the current year with month and date only.

65) What is the query to display the top 20 rows?

SELECT * FROM table_name LIMIT 0,20;

66) Write a query to display the current date and time?

If you want to display the current date and time, use -

SELECT NOW();

If you want to display the current date only, use:

SELECT CURRENT_DATE();

67) What is the save point in MySQL?

A defined point in any transaction is known as savepoint.

SAVEPOINT is a statement in MySQL, which is used to set a named transaction savepoint with the name of the identifier.

68) What is SQLyog?

SQLyog program is the most popular GUI tool for admin. It is the most popular MySQL manager and admin tool. It combines the features of MySQL administrator, phpMyadmin, and others. MySQL front ends and MySQL GUI tools.

69) How do you backup a database in MySQL?

It is easy to back up data with phpMyAdmin. Select the database you want to backup by clicking the database name in the left-hand navigation bar. Then click the export button and make sure that all tables are highlighted that you want to back up. Then specify the option you want under export and save the output.

70) What are the different column comparison operators in MySQL?

The =, <>, <=, <, >=, >, <<, >>, < = >, AND, OR or LIKE operator are the comparison operators in MySQL. These operators are generally used with SELECT statement.

71) Write a query to count the number of rows of a table in MySQL.

SELECT COUNT user_id FROM users;

72) Write a query to retrieve a hundred books starting from 20th.

SELECT book_title FROM books LIMIT 20, 100;

73) Write a query to select all teams that won either 1, 3, 5, or 7 games.

SELECT team_name FROM team WHERE team_won IN (1, 3, 5, 7);

74) What is the default port of MySQL Server?

The default port of MySQL Server is 3306.

75) How is the MyISAM table stored?

MyISAM table is stored on disk in three formats.

- '.frm' file : storing the table definition
- '.MYD' (MYData): data file
- '.MYI' (MYIndex): index file

76) What is the usage of ENUMs in MySQL?

ENUMs are string objects. By defining ENUMs, we allow the end-user to give correct input as in case the user provides an input that is not part of the ENUM defined data, then the query won't execute, and an error message will be displayed which says "The wrong Query". For instance, suppose we want to take the gender of the user as an input, so we specify ENUM('male', 'female', 'other'), and hence whenever the user tries to input any string any other than these three it results in an error.

ENUMs are used to limit the possible values that go in the table:

For example:

CREATE TABLE months (month ENUM 'January', 'February', 'March'); INSERT months VALUES ('April').

77) What are the advantages of MyISAM over InnoDB?

MyISAM follows a conservative approach to disk space management and stores each MyISAM table in a separate file, which can be further compressed if required. On the other hand, InnoDB stores the tables in the tablespace. Its further optimization is difficult.

78) What are the differences between MySQL_fetch_array(), MySQL_fetch_object(), MySQL_fetch_row()?

Mysql_fetch_object is used to retrieve the result from the database as objects, while mysql_fetch_array returns result as an array. This will allow access to the data by the field names.

For example:

Using mysql_fetch_object field can be accessed as $result->name.

Using mysql_fetch_array field can be accessed as $result->[name].

Using mysql_fetch_row($result) where $result is the result resource returned from a successful query executed using the mysql_query() function.

Example:

```
$result = mysql_query("SELECT * from students");
while($row = mysql_fetch_row($result))
{
      Some statement;
}
```

79) What is the difference between mysql_connect and mysql_pconnect?

Mysql_connect() is used to open a new connection to the database, while mysql_pconnect() is used to open a persistent connection to the database. It specifies that each time the page is loaded, mysql_pconnect() does not open the database.

80) What is the use of mysql_close()?

Mysql_close() cannot be used to close the persistent connection. However, it can be used to close a connection opened by mysql_connect().

81) What is MySQL data directory?

MySQL data directory is a place where MySQL stores its data. Each subdirectory under this data dictionary represents a MySQL database. By default, the information managed my MySQL = server mysqld is stored in the data directory.

82) How do you determine the location of MySQL data directory?

The default location of MySQL data directory in windows is C:\mysql\data or C:\Program Files\MySQL\MySQL Server 5.0 \data.

83) What is the usage of regular expressions in MySQL?

In MySQL, regular expressions are used in queries for searching a pattern in a string.

- * Matches 0 more instances of the string preceding it.
- + matches one more instances of the string preceding it.
- ? Matches 0 or 1 instances of the string preceding it.
- . Matches a single character.
- [abc] matches a or b or z
- | separates strings
- ^ anchors the match from the start.
- "." Can be used to match any single character. "|" can be used to match either of the two strings
- REGEXP can be used to match the input characters with the database.

Example:

The following statement retrieves all rows where column employee_name contains the text 1000 (example salary):

1. **Select** employee_name
2. **From** employee
3. **Where** employee_name REGEXP '1000'
4. **Order by** employee_name

85) Which command is used to view the content of the table in MySQL?

The SELECT command is used to view the content of the table in MySQL.

Explain Access Control Lists.

An ACL is a list of permissions that are associated with an object. MySQL keeps the Access Control Lists cached in memory, and whenever the user tries to authenticate or execute a command, MySQL checks the permission required for the object, and if the permissions are available, then execution completes successfully.

86) What is InnoDB?

InnoDB is a storage database for SQL. The ACID-transactions are also provided in InnoDB and also includes support for the foreign key. Initially owned by InnobaseOY now belongs to Oracle Corporation after it acquired the latter since 2005.

87) What is ISAM?

It is a system for file management developed by IBM, which allows records to access sequentially or even randomly.

88) How can we run batch mode in MySQL?

To perform batch mode in MySQL, we use the following command:

1. mysql;
2. mysql mysql.**out**;

89) What are federated tables?

Federated tables are tables that point to the tables located on other databases on some other server.

90) What is the difference between primary key and candidate key?

To identify each row of a table, we will use a primary key. For a table, there exists only one primary key.

A candidate key is a column or a set of columns, which can be used to uniquely identify any record in the database without having to reference any other data.

92) What are DDL, DML, and DCL?

Majorly SQL commands can be divided into three categories, i.e., DDL, DML & DCL. Data Definition Language (DDL) deals with all the database schemas, and it defines how the data should reside in the database. Commands like CreateTABLE and ALTER TABLE are part of DDL.

Data Manipulative Language (DML) deals with operations and manipulations on the data. The commands in DML are Insert, Select, etc.

Data Control Languages (DCL) are related to the Grant and permissions. In short, the authorization to access any part of the database is defined by these.

Sample Table – Worker

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMEN
001	Monika	Arora	100000	2014-02-20 09:00:00	HR
002	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
003	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
004	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
005	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
006	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
007	Satish	Kumar	75000	2014-01-20 09:00:00	Account
008	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

Sample Table – Bonus

WORKER_REF_ID	BONUS_DATE	BONUS_AMOUNT
1	2016-02-20 00:00:00	5000
2	2016-06-11 00:00:00	3000
3	2016-02-20 00:00:00	4000
1	2016-02-20 00:00:00	4500
2	2016-06-11 00:00:00	3500

Sample Table – Title

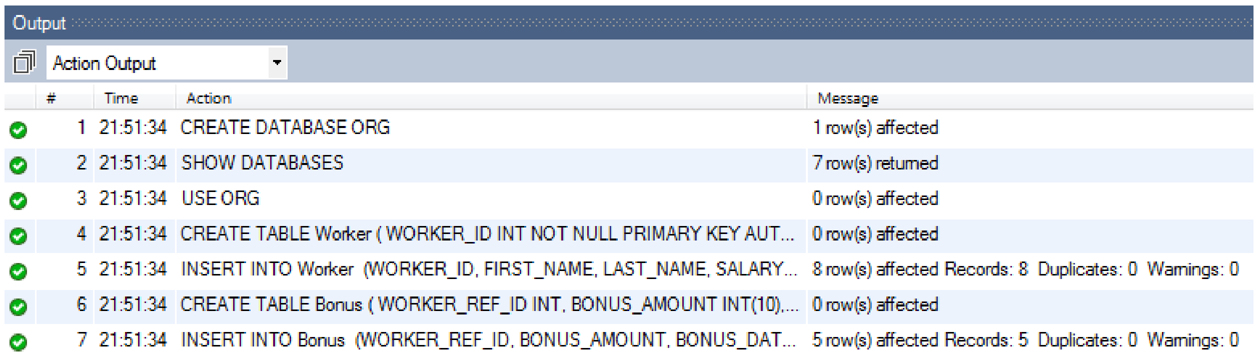
WORKER_REF_ID	WORKER_TITLE	AFFECTED_FROM
1	Manager	2016-02-20 00:00:00
2	Executive	2016-06-11 00:00:00
8	Executive	2016-06-11 00:00:00
5	Manager	2016-06-11 00:00:00
4	Asst. Manager	2016-06-11 00:00:00
7	Executive	2016-06-11 00:00:00
6	Lead	2016-06-11 00:00:00
3	Lead	2016-06-11 00:00:00

To prepare the sample data, you can run the following queries in your database query executor or on the SQL command line. We've tested them with MySQL Server 5.7 and MySQL Workbench 6.3.8 query browser. You can also download these Softwares and install them to carry on the SQL exercise.

SQL Script to Seed Sample Data.

```
CREATE DATABASE ORG;
SHOW DATABASES;
USE ORG;


CREATE TABLE Worker (
        WORKER_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
	FIRST_NAME CHAR(25),
	LAST_NAME CHAR(25),
	SALARY INT(15),
	JOINING_DATE DATETIME,
	DEPARTMENT CHAR(25)
);

INSERT INTO Worker
	(WORKER_ID, FIRST_NAME, LAST_NAME, SALARY, JOINING_DATE, DEPARTMENT) VALUES
		(001, 'Monika', 'Arora', 100000, '14-02-20 09.00.00', 'HR'),
		(002, 'Niharika', 'Verma', 80000, '14-06-11 09.00.00', 'Admin'),
		(003, 'Vishal', 'Singhal', 300000, '14-02-20 09.00.00', 'HR'),
		(004, 'Amitabh', 'Singh', 500000, '14-02-20 09.00.00', 'Admin'),
		(005, 'Vivek', 'Bhati', 500000, '14-06-11 09.00.00', 'Admin'),
		(006, 'Vipul', 'Diwan', 200000, '14-06-11 09.00.00', 'Account'),
		(007, 'Satish', 'Kumar', 75000, '14-01-20 09.00.00', 'Account'),
		(008, 'Geetika', 'Chauhan', 90000, '14-04-11 09.00.00', 'Admin');

CREATE TABLE Bonus (
	WORKER_REF_ID INT,
	BONUS_AMOUNT INT(10),
	BONUS_DATE DATETIME,
	FOREIGN KEY (WORKER_REF_ID)
		REFERENCES Worker(WORKER_ID)
      ON DELETE CASCADE
);

INSERT INTO Bonus
	(WORKER_REF_ID, BONUS_AMOUNT, BONUS_DATE) VALUES
		(001, 5000, '16-02-20'),
                    (002, 3000, '16-06-11'),
                    (003, 4000, '16-02-20'),
                    (001, 4500, '16-02-20'),
                    (002, 3500, '16-06-11');
CREATE TABLE Title (
          WORKER_REF_ID INT,
          WORKER_TITLE CHAR(25),
          AFFECTED_FROM DATETIME,
          FOREIGN KEY (WORKER_REF_ID)
                    REFERENCES Worker(WORKER_ID)
      ON DELETE CASCADE
);

INSERT INTO Title
          (WORKER_REF_ID, WORKER_TITLE, AFFECTED_FROM) VALUES
 (001, 'Manager', '2016-02-20 00:00:00'),
 (002, 'Executive', '2016-06-11 00:00:00'),
 (008, 'Executive', '2016-06-11 00:00:00'),
 (005, 'Manager', '2016-06-11 00:00:00'),
 (004, 'Asst. Manager', '2016-06-11 00:00:00'),
 (007, 'Executive', '2016-06-11 00:00:00'),
 (006, 'Lead', '2016-06-11 00:00:00'),
 (003, 'Lead', '2016-06-11 00:00:00');
```

Once above SQL would run, you'll see a result similar to the one attached below.



```
   #  Time      Action                                                          Message
✓  1  21:51:34  CREATE DATABASE ORG                                             1 row(s) affected
✓  2  21:51:34  SHOW DATABASES                                                  7 row(s) returned
✓  3  21:51:34  USE ORG                                                         0 row(s) affected
✓  4  21:51:34  CREATE TABLE Worker ( WORKER_ID INT NOT NULL PRIMARY KEY AUT...  0 row(s) affected
✓  5  21:51:34  INSERT INTO Worker  (WORKER_ID, FIRST_NAME, LAST_NAME, SALARY... 8 row(s) affected Records: 8  Duplicates: 0  Warnings: 0
✓  6  21:51:34  CREATE TABLE Bonus ( WORKER_REF_ID INT, BONUS_AMOUNT INT(10),...  0 row(s) affected
✓  7  21:51:34  INSERT INTO Bonus  (WORKER_REF_ID, BONUS_AMOUNT, BONUS_DAT...     5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0
```

Q-1. Write an SQL query to fetch "FIRST_NAME" from Worker table using the alias name as <WORKER_NAME>.

Ans.

The required query is:

```
Select FIRST_NAME AS WORKER_NAME from Worker;
```

Q-2. Write an SQL query to fetch "FIRST_NAME" from Worker table in upper case.

Ans.

The required query is:

```
Select upper(FIRST_NAME) from Worker;
```

Q-3. Write an SQL query to fetch unique values of DEPARTMENT from Worker table.

Ans.

The required query is:

```
Select distinct DEPARTMENT from Worker;
```

Q-4. Write an SQL query to print the first three characters of FIRST_NAME from Worker table.

Ans.

The required query is:

```
Select substring(FIRST_NAME,1,3) from Worker;
```

Q-5. Write an SQL query to find the position of the alphabet ('a') in the first name column 'Amitabh' from Worker table.

Ans.

The required query is:

```
Select INSTR(FIRST_NAME, BINARY'a') from Worker where FIRST_NAME =
'Amitabh';
```

Notes.

- The INSTR method is in case-sensitive by default.

- Using Binary operator will make INSTR work as the case-sensitive function.

Q-6. Write an SQL query to print the FIRST_NAME from Worker table after removing white spaces from the right side.

Ans.

The required query is:

```
Select RTRIM(FIRST_NAME) from Worker;
```

Q-7. Write an SQL query to print the DEPARTMENT from Worker table after removing white spaces from the left side.

Ans.

The required query is:

```
Select LTRIM(DEPARTMENT) from Worker;
```

Q-8. Write an SQL query that fetches the unique values of DEPARTMENT from Worker table and prints its length.

Ans.

The required query is:

```
Select distinct length(DEPARTMENT) from Worker;
```

Q-9. Write an SQL query to print the FIRST_NAME from Worker table after replacing ‘a’ with ‘A’.

Ans.

The required query is:

```
Select REPLACE(FIRST_NAME,'a','A') from Worker;
```

Q-10. Write an SQL query to print the FIRST_NAME and LAST_NAME from Worker table into a single column COMPLETE_NAME. A space char should separate them.

Ans.

The required query is:

```
Select CONCAT(FIRST_NAME, ' ', LAST_NAME) AS 'COMPLETE_NAME' from
Worker;
```

Q-11. Write an SQL query to print all Worker details from the Worker table order by FIRST_NAME Ascending.

Ans.

The required query is:

```
Select * from Worker order by FIRST_NAME asc;
```

Q-12. Write an SQL query to print all Worker details from the Worker table order by FIRST_NAME Ascending and DEPARTMENT Descending.

Ans.

The required query is:

```
Select * from Worker order by FIRST_NAME asc,DEPARTMENT desc;
```

Q-13. Write an SQL query to print details for Workers with the first name as "Vipul" and "Satish" from Worker table.

Ans.

The required query is:

```
Select * from Worker where FIRST_NAME in ('Vipul','Satish');
```

Q-14. Write an SQL query to print details of workers excluding first names, "Vipul" and "Satish" from Worker table.

Ans.

The required query is:

```
Select * from Worker where FIRST_NAME not in ('Vipul','Satish');
```

Q-15. Write an SQL query to print details of Workers with DEPARTMENT name as "Admin".

Ans.

The required query is:

```
Select * from Worker where DEPARTMENT like 'Admin%';
```

Q-16. Write an SQL query to print details of the Workers whose FIRST_NAME contains 'a'.

Ans.

The required query is:

```
Select * from Worker where FIRST_NAME like '%a%';
```

Q-17. Write an SQL query to print details of the Workers whose FIRST_NAME ends with 'a'.

Ans.

The required query is:

```
Select * from Worker where FIRST_NAME like '%a';
```

Q-18. Write an SQL query to print details of the Workers whose FIRST_NAME ends with 'h' and contains six alphabets.

Ans.

The required query is:

```
Select * from Worker where FIRST_NAME like '_____h';
```

Q-19. Write an SQL query to print details of the Workers whose SALARY lies between 100000 and 500000.

Ans.

The required query is:

```
Select * from Worker where SALARY between 100000 and 500000;
```

Q-20. Write an SQL query to print details of the Workers who have joined in Feb'2014.

Ans.

The required query is:

```
Select * from Worker where year(JOINING_DATE) = 2014 and month(JOINING_DATE) = 2;
```

Q-21. Write an SQL query to fetch the count of employees working in the department 'Admin'.

Ans.

The required query is:

```
SELECT COUNT(*) FROM worker WHERE DEPARTMENT = 'Admin';
```

Q-22. Write an SQL query to fetch worker names with salaries >= 50000 and <= 100000.

Ans.

The required query is:

```
SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME) As Worker_Name, Salary
FROM worker
WHERE WORKER_ID IN
(SELECT WORKER_ID FROM worker
WHERE Salary BETWEEN 50000 AND 100000);
```

Q-23. Write an SQL query to fetch the no. of workers for each department in the descending order.

Ans.

The required query is:

```
SELECT DEPARTMENT, count(WORKER_ID) No_Of_Workers
FROM worker
GROUP BY DEPARTMENT
ORDER BY No_Of_Workers DESC;
```

Q-24. Write an SQL query to print details of the Workers who are also Managers.

Ans.

The required query is:

```
SELECT DISTINCT W.FIRST_NAME, T.WORKER_TITLE
FROM Worker W
INNER JOIN Title T
ON W.WORKER_ID = T.WORKER_REF_ID
AND T.WORKER_TITLE in ('Manager');
```

Q-25. Write an SQL query to fetch duplicate records having matching data in some fields of a table.

Ans.

The required query is:

```
SELECT WORKER_TITLE, AFFECTED_FROM, COUNT(*)
FROM Title
GROUP BY WORKER_TITLE, AFFECTED_FROM
HAVING COUNT(*) > 1;
```

Q-26. Write an SQL query to show only odd rows from a table.

Ans.

The required query is:

```
SELECT * FROM Worker WHERE MOD (WORKER_ID, 2) <> 0;
```

Q-27. Write an SQL query to show only even rows from a table.

Ans.

The required query is:

```
SELECT * FROM Worker WHERE MOD (WORKER_ID, 2) = 0;
```

Q-28. Write an SQL query to clone a new table from another table.

Ans.

The general query to clone a table with data is:

```
SELECT * INTO WorkerClone FROM Worker;
```

The general way to clone a table without information is:

```
SELECT * INTO WorkerClone FROM Worker WHERE 1 = 0;
```

An alternate way to clone a table (for MySQL) without is:

```
CREATE TABLE WorkerClone LIKE Worker;
```

Q-29. Write an SQL query to fetch intersecting records of two tables.

Ans.

The required query is:

```
(SELECT * FROM Worker)
INTERSECT
(SELECT * FROM WorkerClone);
```

Q-30. Write an SQL query to show records from one table that another table does not have.

Ans.

The required query is:

```
SELECT * FROM Worker
MINUS
SELECT * FROM Title;
```

Q-31. Write an SQL query to show the current date and time.

Ans.

Following MySQL query returns the current date:

```
SELECT CURDATE();
```

Following MySQL query returns the current date and time:

```
SELECT NOW();
```

```
SELECT getdate();
```

Following Oracle query returns the current date and time:

```
SELECT SYSDATE FROM DUAL;
```

Q-32. Write an SQL query to show the top n (say 10) records of a table.

Ans.

Following MySQL query will return the top n records using the LIMIT method:

```
SELECT * FROM Worker ORDER BY Salary DESC LIMIT 10;
```

Following SQL Server query will return the top n records using the TOP command:

```
SELECT TOP 10 * FROM Worker ORDER BY Salary DESC;
```

Following Oracle query will return the top n records with the help of ROWNUM:

```
SELECT * FROM (SELECT * FROM Worker ORDER BY Salary DESC)
WHERE ROWNUM <= 10;
```

Q-33. Write an SQL query to determine the nth (say n=5) highest salary from a table.

Ans.

The following MySQL query returns the nth highest salary:

```
SELECT Salary FROM Worker ORDER BY Salary DESC LIMIT n-1,1;
```

The following SQL Server query returns the nth highest salary:

```
SELECT TOP 1 Salary
FROM (
 SELECT DISTINCT TOP n Salary
 FROM Worker
 ORDER BY Salary DESC
 )
ORDER BY Salary ASC;
```

Q-34. Write an SQL query to determine the 5th highest salary without using TOP or limit method.

Ans.

The following query is using the correlated subquery to return the 5th highest salary:

```
SELECT Salary
FROM Worker W1
WHERE 4 = (
 SELECT COUNT( DISTINCT ( W2.Salary ) )
 FROM Worker W2
 WHERE W2.Salary >= W1.Salary
 );
```

Use the following generic method to find nth highest salary without using TOP or limit.

```
SELECT Salary
FROM Worker W1
WHERE n-1 = (
 SELECT COUNT( DISTINCT ( W2.Salary ) )
 FROM Worker W2
 WHERE W2.Salary >= W1.Salary
 );
```

Q-35. Write an SQL query to fetch the list of employees with the same salary.

Ans.

The required query is:

```
Select distinct W.WORKER_ID, W.FIRST_NAME, W.Salary
from Worker W, Worker W1
where W.Salary = W1.Salary
and W.WORKER_ID != W1.WORKER_ID;
```

Q-36. Write an SQL query to show the second highest salary from a table.

Ans.

The required query is:

```
Select max(Salary) from Worker
where Salary not in (Select max(Salary) from Worker);
```

Q-37. Write an SQL query to show one row twice in results from a table.

Ans.

The required query is:

```
select FIRST_NAME, DEPARTMENT from worker W where
W.DEPARTMENT='HR'
union all
select FIRST_NAME, DEPARTMENT from Worker W1 where
W1.DEPARTMENT='HR';
```

Q-38. Write an SQL query to fetch intersecting records of two tables.

Ans.

The required query is:

```
(SELECT * FROM Worker)
INTERSECT
(SELECT * FROM WorkerClone);
```

Q-39. Write an SQL query to fetch the first 50% records from a table.

Ans.

The required query is:

```
SELECT *
FROM WORKER
WHERE WORKER_ID <= (SELECT count(WORKER_ID)/2 from Worker);
```

Q-40. Write an SQL query to fetch the departments that have less than five people in it.

Ans.

The required query is:

```
SELECT DEPARTMENT, COUNT(WORKER_ID) as 'Number of Workers' FROM
Worker GROUP BY DEPARTMENT HAVING COUNT(WORKER_ID) < 5;
```

Q-41. Write an SQL query to show all departments along with the number of people in there.

Ans.

The following query returns the expected result:

```
SELECT DEPARTMENT, COUNT(DEPARTMENT) as 'Number of Workers'
FROM Worker GROUP BY DEPARTMENT;
```

Q-42. Write an SQL query to show the last record from a table.

Ans.

The following query will return the last record from the Worker table:

```
Select * from Worker where WORKER_ID = (SELECT max(WORKER_ID) from
Worker);
```

Q-43. Write an SQL query to fetch the first row of a table.

Ans.

The required query is:

```
Select * from Worker where WORKER_ID = (SELECT min(WORKER_ID) from
Worker);
```

Q-44. Write an SQL query to fetch the last five records from a table.

Ans.

The required query is:

```
SELECT * FROM Worker WHERE WORKER_ID <=5

UNION
```

```
SELECT * FROM (SELECT * FROM Worker W order by W.WORKER_ID DESC)
AS W1 WHERE W1.WORKER_ID <=5;
```

Q-45. Write an SQL query to print the name of employees having the highest salary in each department.

Ans.

The required query is:

```
SELECT t.DEPARTMENT,t.FIRST_NAME,t.Salary from(SELECT max(Salary) as
TotalSalary,DEPARTMENT from Worker group by DEPARTMENT) as TempNew

Inner Join Worker t on TempNew.DEPARTMENT=t.DEPARTMENT

 and TempNew.TotalSalary=t.Salary;
```

Q-46. Write an SQL query to fetch three max salaries from a table.

Ans.

The required query is:

```
SELECT distinct Salary from worker a WHERE 3 >= (SELECT count(distinct
Salary) from worker b WHERE a.Salary <= b.Salary) order by a.Salary desc;
```

Q-47. Write an SQL query to fetch three min salaries from a table.

Ans.

The required query is:

```
SELECT distinct Salary from worker a WHERE 3 >= (SELECT count(distinct
Salary) from worker b WHERE a.Salary >= b.Salary) order by a.Salary desc;
```

Q-48. Write an SQL query to fetch nth max salaries from a table.

Ans.

The required query is:

```
SELECT distinct Salary from worker a WHERE n >= (SELECT count(distinct
Salary) from worker b WHERE a.Salary <= b.Salary) order by a.Salary desc;
```

Q-49. Write an SQL query to fetch departments along with the total salaries paid for each of them.

Ans.

The required query is:

```
SELECT DEPARTMENT, sum(Salary) from worker group by DEPARTMENT;
```

Q-50. Write an SQL query to fetch the names of workers who earn the highest salary.

Ans.

The required query is:

```
SELECT FIRST_NAME, SALARY from Worker WHERE SALARY=(SELECT
max(SALARY) from Worker);
```

Q #2) What are the features of MySQL?

Answer: MySQL has several useful features that make it a popular database management software.

Some important features of MySQL are mentioned below:

- It is reliable and easy to use too.
- It is a suitable database software for both large and small applications.
- Anyone can install and use it at no cost.
- It is supported by many well-known programming languages, such as PHP, Java, C++, PERL, etc.
- It supports standard SQL (Structured Query Language).
- The open-source license of MySQL is customizable. Hence, a developer can modify it according to the requirements of the application.

Q #3) What is the default port number of MySQL?

Answer: The default port number of MySQL is 3306.

Q #4) How can you find out the version of the installed MySQL?

Answer: The version of the installed MySQL server can be found out easily by running the following command from the MySQL prompt.

mysql> SHOW VARIABLES LIKE “%version%”;

Q #5) What are the advantages and disadvantages of using MySQL?

Answer: There are several advantages of MySQL which are making it a more popular database system now.

Some significant advantages and disadvantages of MySQL are mentioned below.

Advantages:

- It is well-known for its reliable and secure database management system. Transactional tasks of the website can be done more securely by using this software.
- It supports different types of storage engines to store the data and it works faster for this feature.
- It can handle millions of queries with a high-speed transactional process.
- It supports many advanced level database features, such as multi-level transactions, data integrity, deadlock identification, etc.
- Maintenance and debugging processes are easier for this software.

Disadvantages:

- It is hard to make MySQL scalable.
- It is not suitable for a very large type of database.
- The uses of stored routines and triggers are limited to MySQL.

Q #6) What is the function of myisamchk?

Answer: myisamchk is a useful database utility tool that is used to get information about MyISAM database tables.

It is also used for checking, debugging, repairing and optimizing database tables. It is better to use this command when the server is down or when the required tables are not in use by the server.

Syntax:

myisamchk [OPTION] table_name…

The available options of this tool can be retrieved by using the following command.

myisamchk –help

To check or repair all MyISAM tables, the following command will be required for executing from the database directory location.

myisamchk *.MYI

Q #7) What are the purposes of using ENUM and SET data types?

Answer: ENUM data type is used in the MySQL database table to select any one value from the predefined list.

The value of a particular field can be restricted by defining the predefined list as the field which is declared as ENUM will not accept any value outside the list.

The SET data type is used to select one or more or all values from the predefined list. This data type can also be used to restrict the field for inserting only the predefined list of values like ENUM.

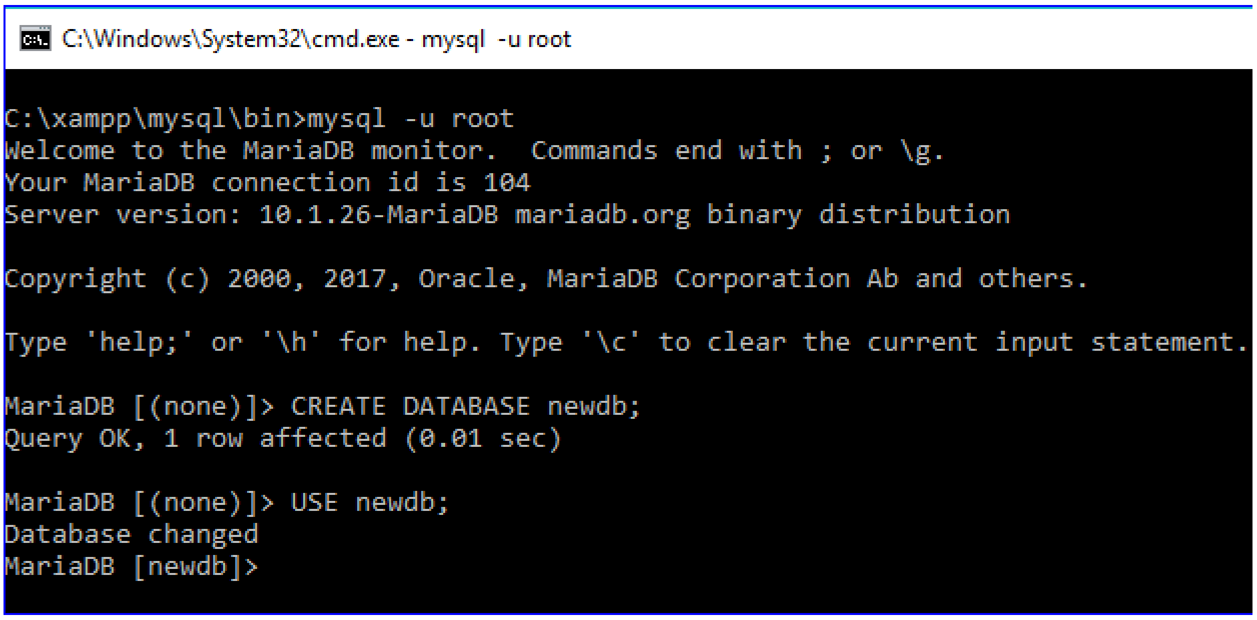
Example:

Run MySQL server from the command prompt and execute the following SQL commands to know the use of ENUM and SET data type.

The following SQL commands create a new database named '**newdb**' and select the database for use.

CREATE DATABASE newdb;

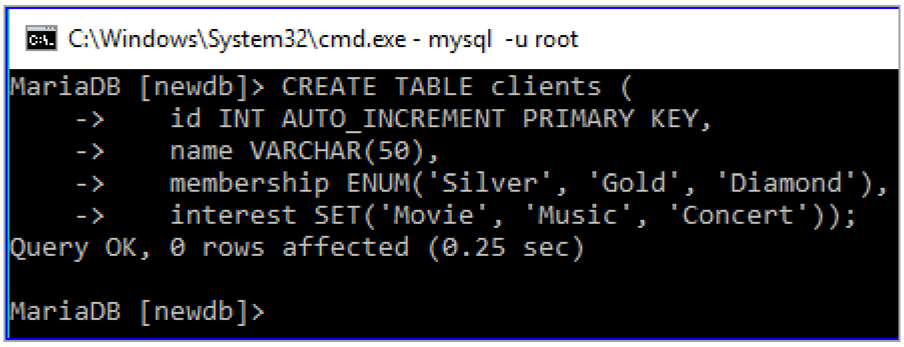
USE newdb;



```
C:\xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 104
Server version: 10.1.26-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> CREATE DATABASE newdb;
Query OK, 1 row affected (0.01 sec)

MariaDB [(none)]> USE newdb;
Database changed
MariaDB [newdb]>
```

The following SQL command will create a table named **clients** with the fields ENUM and SET data type.

```
CREATE TABLE clients (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  membership ENUM('Silver', 'Gold', 'Diamond'),
  interest SET('Movie', 'Music', 'Concert'));
```



```
MariaDB [newdb]> CREATE TABLE clients (
    ->    id INT AUTO_INCREMENT PRIMARY KEY,
    ->    name VARCHAR(50),
    ->    membership ENUM('Silver', 'Gold', 'Diamond'),
    ->    interest SET('Movie', 'Music', 'Concert'));
Query OK, 0 rows affected (0.25 sec)

MariaDB [newdb]>
```

INSERT queries will create two records in the table. ENUM field only accepts data from the defined list.

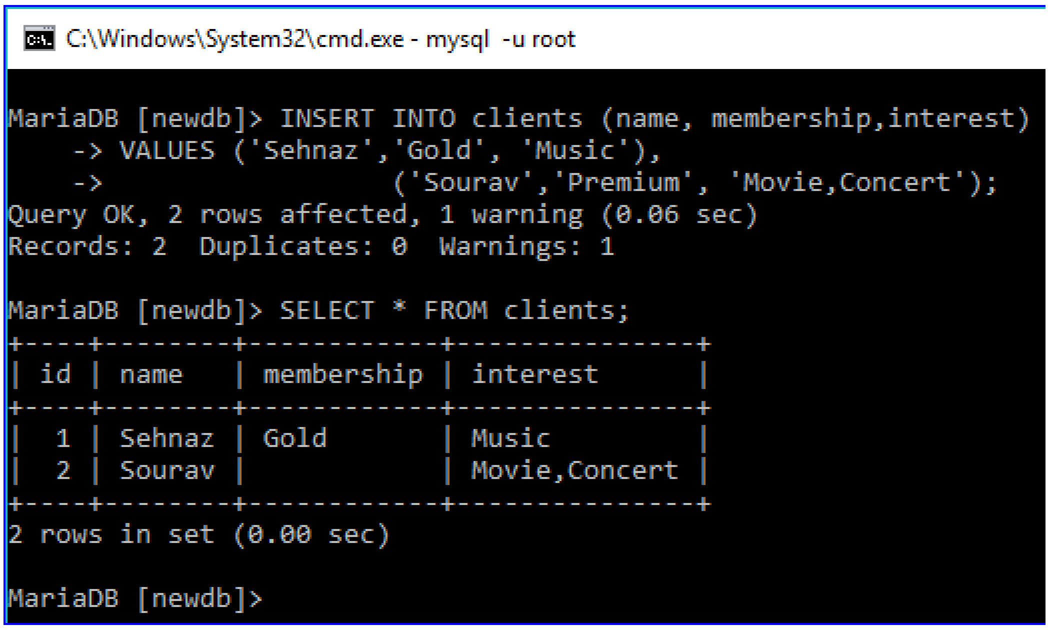
‘**Premium**’ value does not exist on the ENUM list. Hence, the value of the ENUM field will be empty for the second record. SET can accept multiple values and both the data will be inserted in the second record.

INSERT INTO clients (**name**, membership,interest)

VALUES ('Sehnaz','Gold', 'Music'),

('Sourav','Premium', 'Movie,Concert');

SELECT * **FROM** clients;



```
MariaDB [newdb]> INSERT INTO clients (name, membership,interest)
    -> VALUES ('Sehnaz','Gold', 'Music'),
    ->                  ('Sourav','Premium', 'Movie,Concert');
Query OK, 2 rows affected, 1 warning (0.06 sec)
Records: 2  Duplicates: 0  Warnings: 1

MariaDB [newdb]> SELECT * FROM clients;
+----+--------+------------+---------------+
| id | name   | membership | interest      |
+----+--------+------------+---------------+
|  1 | Sehnaz | Gold       | Music         |
|  2 | Sourav |            | Movie,Concert |
+----+--------+------------+---------------+
2 rows in set (0.00 sec)

MariaDB [newdb]>
```

Q #8) What are the differences between a primary key and a foreign key?

Answer: The database table uses a primary key to identify each row uniquely. It is necessary to declare the primary key on those tables that require to create a relationship among them. One or more fields of a table can be declared as the primary key.

When the primary key of any table is used in another table as the primary key or another field for making a database relation, then it is called a foreign key.

The differences between these two keys are mentioned below:

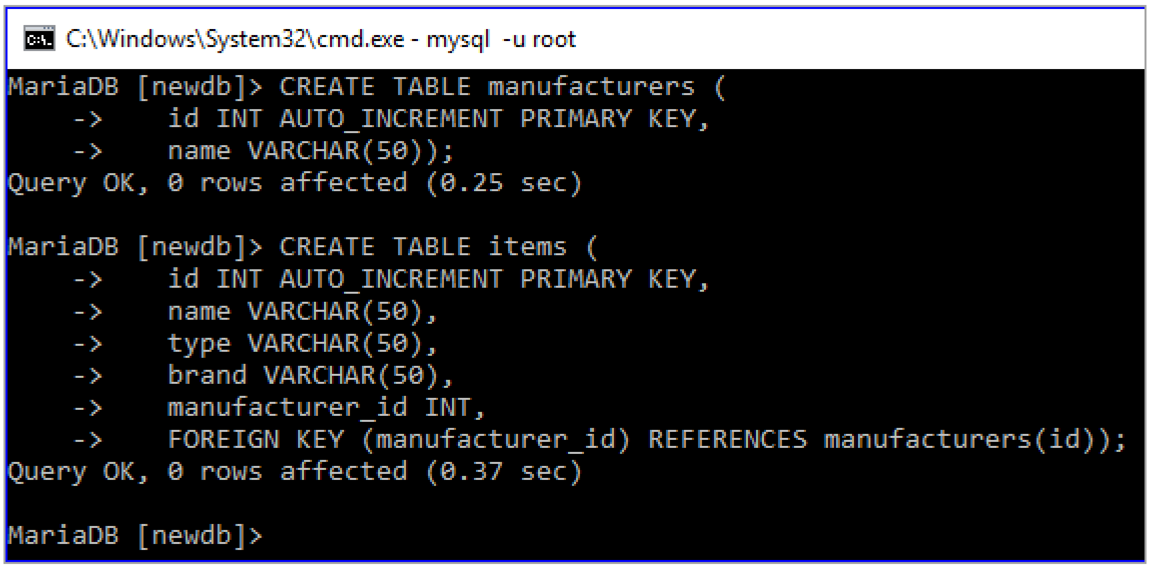
- The primary key uniquely identifies a record, whereas foreign key refers to the primary key of another table.
- The primary key can never accept a NULL value but foreign key accepts a NULL value.
- When a record is inserted in a table that contains the primary key then it is not necessary to insert the value on the table that contains this primary key field as the foreign key.
- When a record is deleted from the table that contains the primary key then the corresponding record must be deleted from the table containing the foreign key for data consistency. But any record can be deleted from the table that contains a foreign key without deleting a related record of another table.

Example:

Two tables named **manufacturers** and **items** will be created after executing the following two SQL commands.

Here, the primary key of the **manufacturer's** table is used as a foreign key in the **items** table with the field name **manufacturer_id**. Hence, the **manufacturer_id** field will contain only those values that exist in the **manufacturer's** table.

```
CREATE TABLE manufacturers (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50));
CREATE TABLE items (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  type VARCHAR(50),
  brand VARCHAR(50),
  manufacturer_id INT,
  FOREIGN KEY (manufacturer_id) REFERENCES manufacturers(id));
```



```
MariaDB [newdb]> CREATE TABLE manufacturers (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(50));
Query OK, 0 rows affected (0.25 sec)

MariaDB [newdb]> CREATE TABLE items (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(50),
    ->     type VARCHAR(50),
    ->     brand VARCHAR(50),
    ->     manufacturer_id INT,
    ->     FOREIGN KEY (manufacturer_id) REFERENCES manufacturers(id));
Query OK, 0 rows affected (0.37 sec)

MariaDB [newdb]>
```

Q #9) What are the differences between CHAR and VARCHAR data types?

Answer: Both CHAR and VARCHAR data types are used to store string data in the field of the table.

The differences between these data types are mentioned below:

- CHAR data type is used to store fixed-length string data and the VARCHAR data type is used to store variable-length string data.
- The storage size of the CHAR data type will always be the maximum length of this data type and the storage size of VARCHAR will be the length of the inserted string data. Hence, it is better to use the CHAR data type when the length of the string will be the same length for all the records.
- CHAR is used to store small data whereas VARCHAR is used to store large data.
- CHAR works faster and VARCHAR works slower.

Further Reading =>> MySQL Data Types

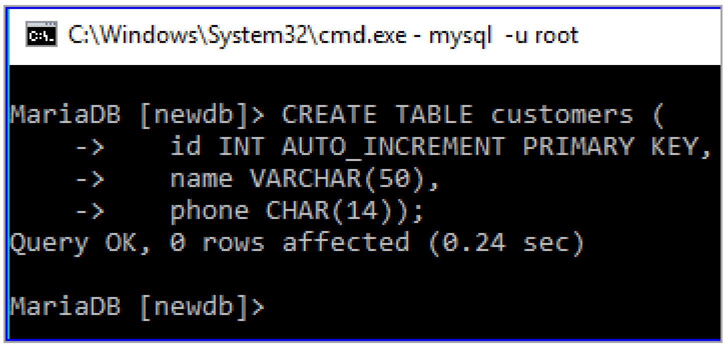
Example:

The following SQL statement will create a table named Customers. In this table, the data type of **name** field is VARCHAR and the data type of **phone** field is CHAR.

The size of the **name** field will depend on the length of the inserted value. The size of the **phone** field will always be 14 characters even if the length of the inserted value is less than 14 characters.

```
CREATE TABLE customers (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
```

phone **CHAR**(14))



C:\Windows\System32\cmd.exe - mysql -u root

```
MariaDB [newdb]> CREATE TABLE customers (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(50),
    ->     phone CHAR(14));
Query OK, 0 rows affected (0.24 sec)

MariaDB [newdb]>
```

Q #10) What is the purpose of using the TIMESTAMP data type?

Answer: A TIMESTAMP data type is used to store the combination of date and time value which is 19 characters long.

The format of TIMESTAMP is YYYY-MM-DD HH:MM: SS. It can store data from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC. By default, the current date and time of the server get inserted in the field of this data type when a new record is inserted or updated.

Q #11) What is the difference between mysql_fetch_array() and ysql_fetch_object() ?

Answer: Both mysql_fetch_array() and mysql_fetch_object() are built-in methods of PHP to retrieve records from MySQL database table.

The difference between these methods is that mysql_fetch_array() returns the result set as an array and mysql_fetch_object() returns the result set as an object.

Example:

```
$result = mysql_query("SELECT id, name FROM clients");


//using mysql_fetch_array()

while ($row = mysql_fetch_array($result, MYSQL_NUM)) {

    printf("ID: %s  Name: %s", $row[0], $row[1]);

}


//using mysql_fetch_object()

while ($row = mysql_fetch_object($result)) {

    printf("ID: %s  Name: %s", $row-&amp;gt;id, $row-&amp;gt;name);

}
```

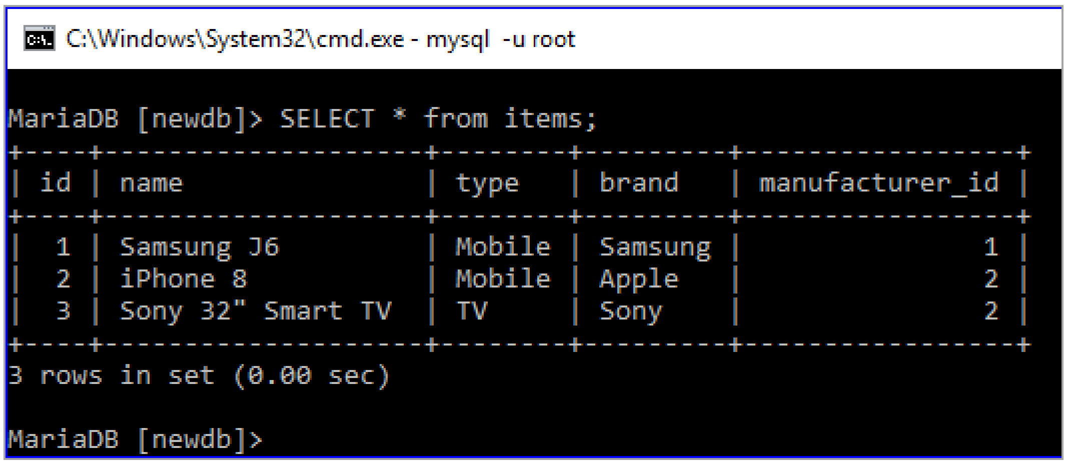
Q #12) How can you filter the duplicate data while retrieving records from the table?

Answer: A DISTINCT keyword is used to filter the duplicate data from the table while retrieving the records from a table.

Example:

The following SQL command shows all the records of the **items** table. The output shows that the table contains duplicate values in the Type field.

SELECT * **from** items;

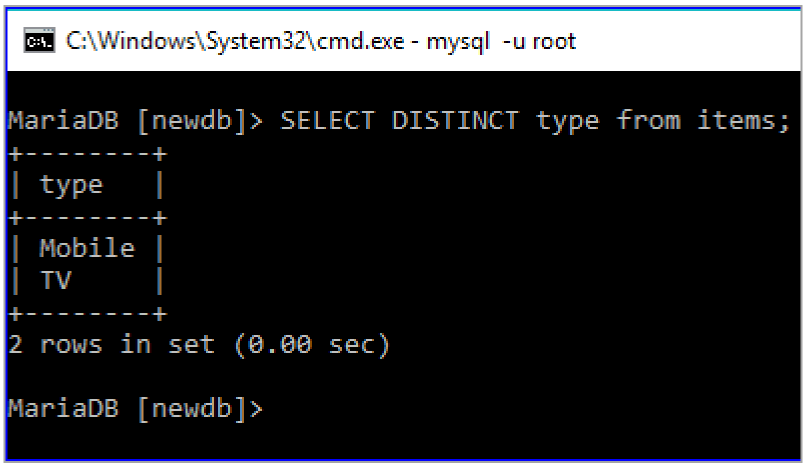


C:\Windows\System32\cmd.exe - mysql -u root

```
MariaDB [newdb]> SELECT * from items;
+----+--------------------+--------+---------+-----------------+
| id | name               | type   | brand   | manufacturer_id |
+----+--------------------+--------+---------+-----------------+
|  1 | Samsung J6         | Mobile | Samsung |               1 |
|  2 | iPhone 8           | Mobile | Apple   |               2 |
|  3 | Sony 32" Smart TV  | TV     | Sony    |               2 |
+----+--------------------+--------+---------+-----------------+
3 rows in set (0.00 sec)

MariaDB [newdb]>
```

The following SQL command will display the values of the **type** field by removing duplicate values.

SELECT DISTINCT type **from** items;



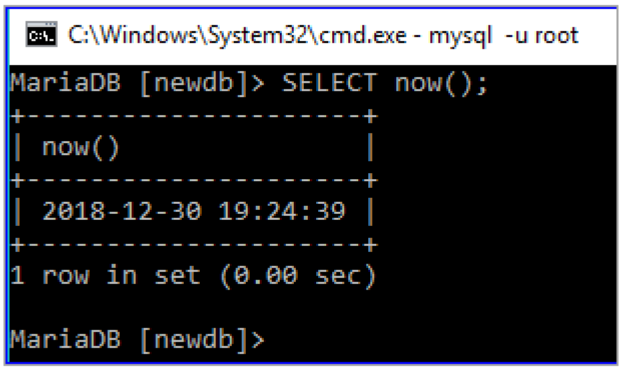
C:\Windows\System32\cmd.exe - mysql -u root

```
MariaDB [newdb]> SELECT DISTINCT type from items;
+--------+
| type   |
+--------+
| Mobile |
| TV     |
+--------+
2 rows in set (0.00 sec)

MariaDB [newdb]>
```

Q #13) What is the difference between NOW() and CURRENT_DATE()?

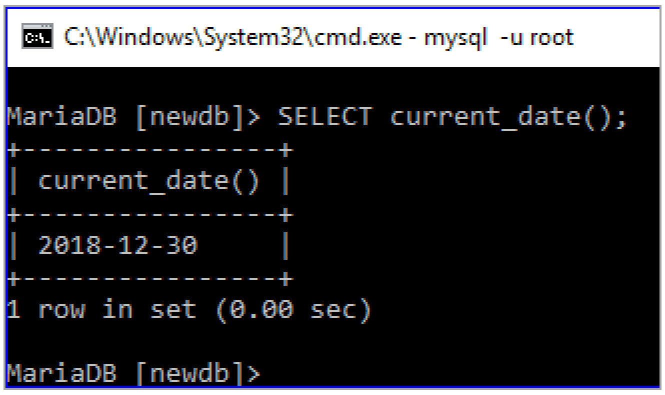
Answer: Both **NOW()** and **CURRENT_DATE()** are built-in MySQL methods. **NOW()** is used to show the current date and time of the server and **CURRENT_DATE()** is used to show only the date of the server.

SELECT now();



```
C:\Windows\System32\cmd.exe - mysql  -u root
MariaDB [newdb]> SELECT now();
+---------------------+
| now()               |
+---------------------+
| 2018-12-30 19:24:39 |
+---------------------+
1 row in set (0.00 sec)

MariaDB [newdb]>
```

SELECT current_date();



```
C:\Windows\System32\cmd.exe - mysql  -u root

MariaDB [newdb]> SELECT current_date();
+----------------+
| current_date() |
+----------------+
| 2018-12-30     |
+----------------+
1 row in set (0.00 sec)

MariaDB [newdb]>
```

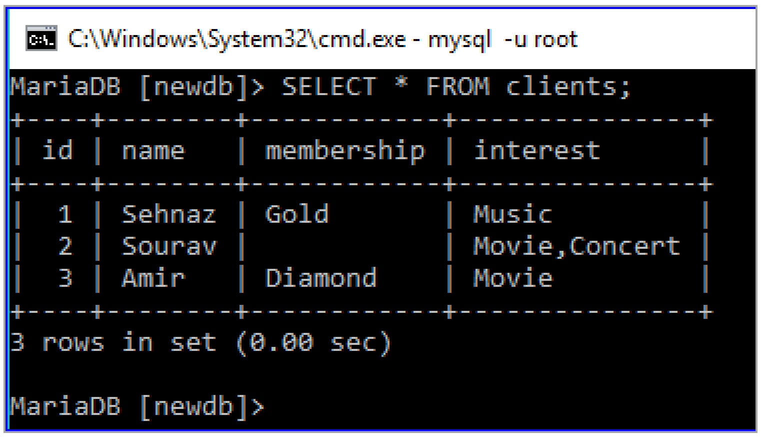
Q #14) Which statement is used in a select query for partial matching?

Answer: REGEXP and **LIKE** statements can be used in a SELECT query for partial matching. REGEXP is used to search records based on the pattern and LIKE is used to search any record by matching any string at the beginning or end or middle of a particular field value.

<u>**Example:**</u>

First, check the existing records of the '**clients'** table by executing the SELECT query.

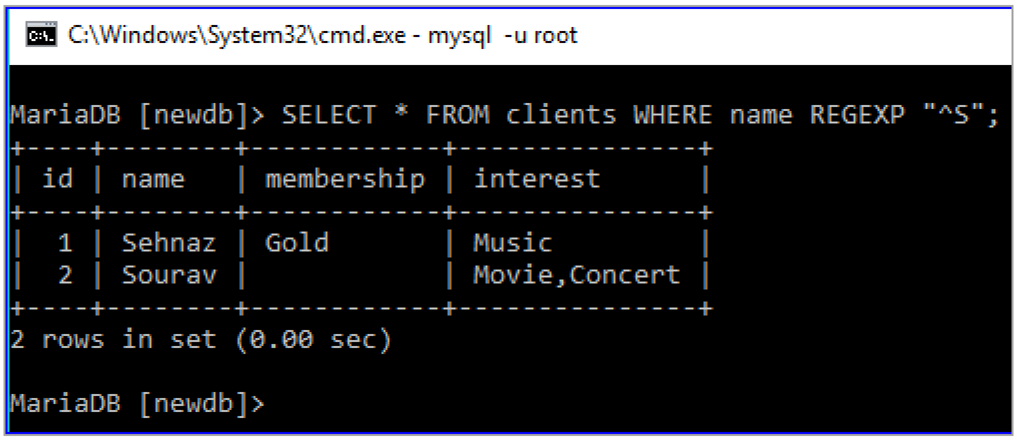
SELECT * **FROM** clients;



```
C:\Windows\System32\cmd.exe - mysql  -u root
MariaDB [newdb]> SELECT * FROM clients;
+----+--------+------------+----------------+
| id | name   | membership | interest       |
+----+--------+------------+----------------+
|  1 | Sehnaz | Gold       | Music          |
|  2 | Sourav |            | Movie,Concert  |
|  3 | Amir   | Diamond    | Movie          |
+----+--------+------------+----------------+
3 rows in set (0.00 sec)

MariaDB [newdb]>
```

Run SELECT query with REGEXP clause to search those records from the **clients** where the client name starts with '**S**'

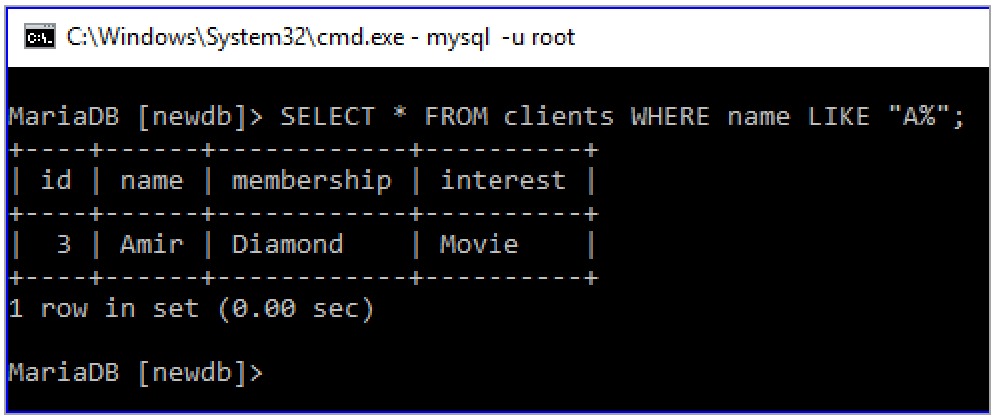
SELECT * **FROM** clients **WHERE name** REGEXP "^S";



```
C:\Windows\System32\cmd.exe - mysql  -u root

MariaDB [newdb]> SELECT * FROM clients WHERE name REGEXP "^S";
+----+--------+------------+----------------+
| id | name   | membership | interest       |
+----+--------+------------+----------------+
|  1 | Sehnaz | Gold       | Music          |
|  2 | Sourav |            | Movie,Concert  |
+----+--------+------------+----------------+
2 rows in set (0.00 sec)

MariaDB [newdb]>
```

Run SELECT query with LIKE clause to search those records from the **clients** where the client name starts with '**A**'

SELECT * **FROM** clients **WHERE name** LIKE "A%";



```
C:\Windows\System32\cmd.exe - mysql  -u root

MariaDB [newdb]> SELECT * FROM clients WHERE name LIKE "A%";
+----+------+------------+----------+
| id | name | membership | interest |
+----+------+------------+----------+
|  3 | Amir | Diamond    | Movie    |
+----+------+------------+----------+
1 row in set (0.00 sec)

MariaDB [newdb]>
```

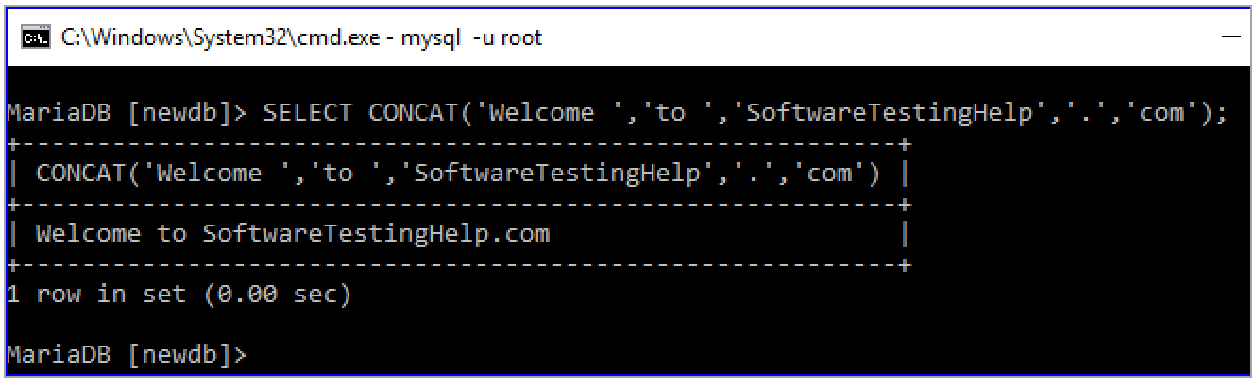
Q #15) Which MySQL function is used to concatenate string?

Answer: CONCAT() function is used to combine two or more string data. The use of this function is here with an example.

Example:

The following SELECT query with CONCAT() function will combine five words, ‘Welcome ‘, ‘to’, ‘SoftwareTestingHelp’,’.’ and ‘com’.

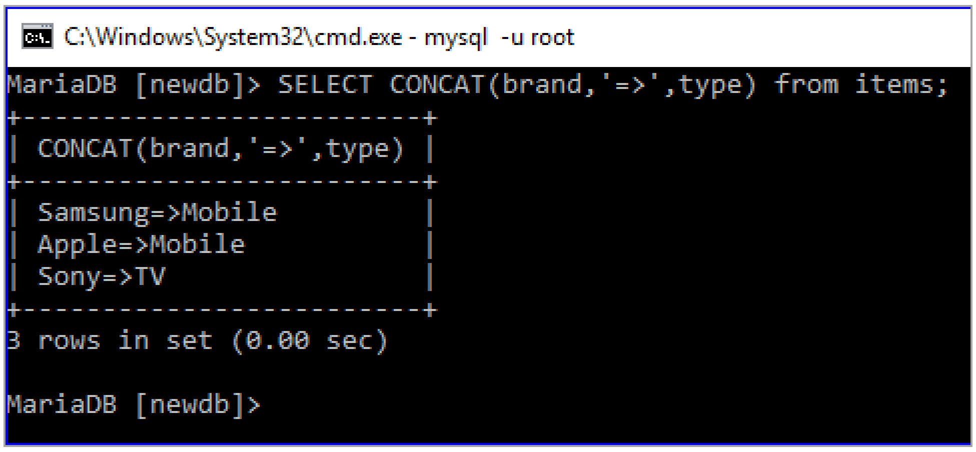
SELECT CONCAT('Welcome ',**to** ','SoftwareTestingHelp','.',com');



```
C:\Windows\System32\cmd.exe - mysql  -u root

MariaDB [newdb]> SELECT CONCAT('Welcome ','to ','SoftwareTestingHelp','.','com');
+-----------------------------------------------------------+
| CONCAT('Welcome ','to ','SoftwareTestingHelp','.','com') |
+-----------------------------------------------------------+
| Welcome to SoftwareTestingHelp.com                        |
+-----------------------------------------------------------+
1 row in set (0.00 sec)

MariaDB [newdb]>
```

CONCAT() function can be used on any table as well. The following SELECT query will show the output by combining two fields, **brand** and **type** of **items** table.

SELECT CONCAT(brand,'=>',type) **from** items;



```
C:\Windows\System32\cmd.exe - mysql  -u root
MariaDB [newdb]> SELECT CONCAT(brand,'=>',type) from items;
+-------------------------+
| CONCAT(brand,'=>',type) |
+-------------------------+
| Samsung=>Mobile         |
| Apple=>Mobile           |
| Sony=>TV                |
+-------------------------+
3 rows in set (0.00 sec)

MariaDB [newdb]>
```

Q #16) How can you change the name of any existing table by using the SQL statement?

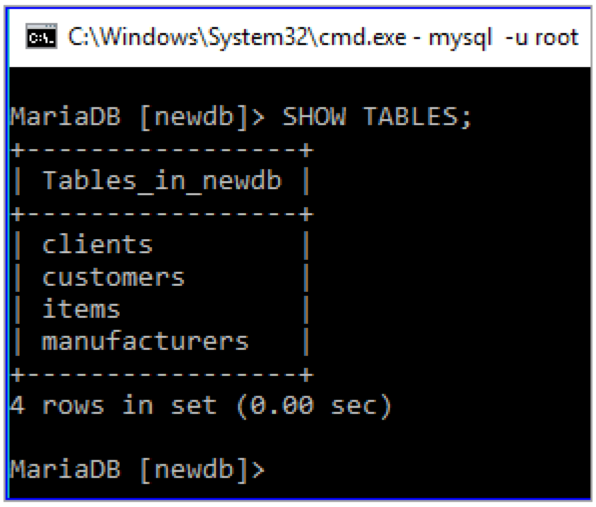
Answer: The following SQL command is used to rename an existing table of the database.

RENAME **TABLE** table_name **TO** new_name

Example:

The following command will show the table list of the **newdb** database.

SHOW TABLES;

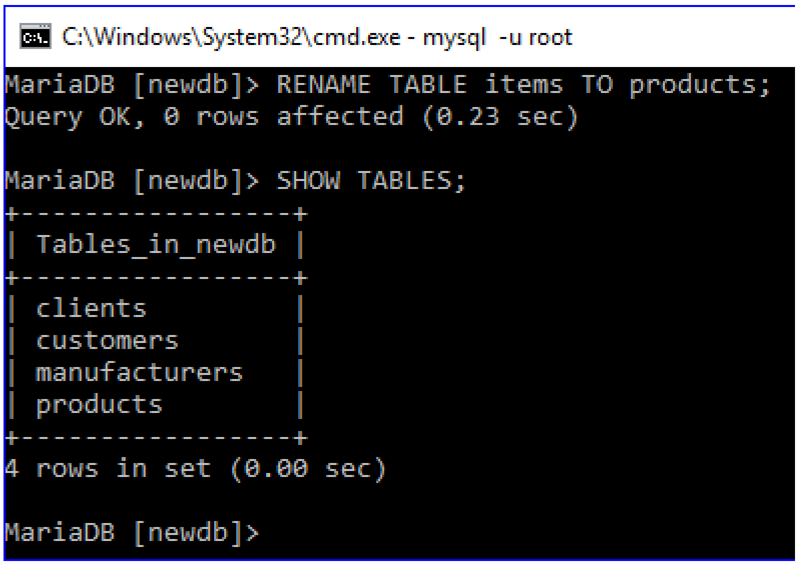


```
C:\Windows\System32\cmd.exe - mysql  -u root

MariaDB [newdb]> SHOW TABLES;
+-----------------+
| Tables_in_newdb |
+-----------------+
| clients         |
| customers       |
| items           |
| manufacturers   |
+-----------------+
4 rows in set (0.00 sec)

MariaDB [newdb]>
```

The following rename command will rename the table **items** by new name **products**.

RENAME **TABLE** items **TO** products;

SHOW TABLES;



```
C:\Windows\System32\cmd.exe - mysql  -u root
MariaDB [newdb]> RENAME TABLE items TO products;
Query OK, 0 rows affected (0.23 sec)

MariaDB [newdb]> SHOW TABLES;
+-----------------+
| Tables_in_newdb |
+-----------------+
| clients         |
| customers       |
| manufacturers   |
| products        |
+-----------------+
4 rows in set (0.00 sec)

MariaDB [newdb]>
```

Q #17) How can you retrieve a portion of any column value by using a SELECT query?

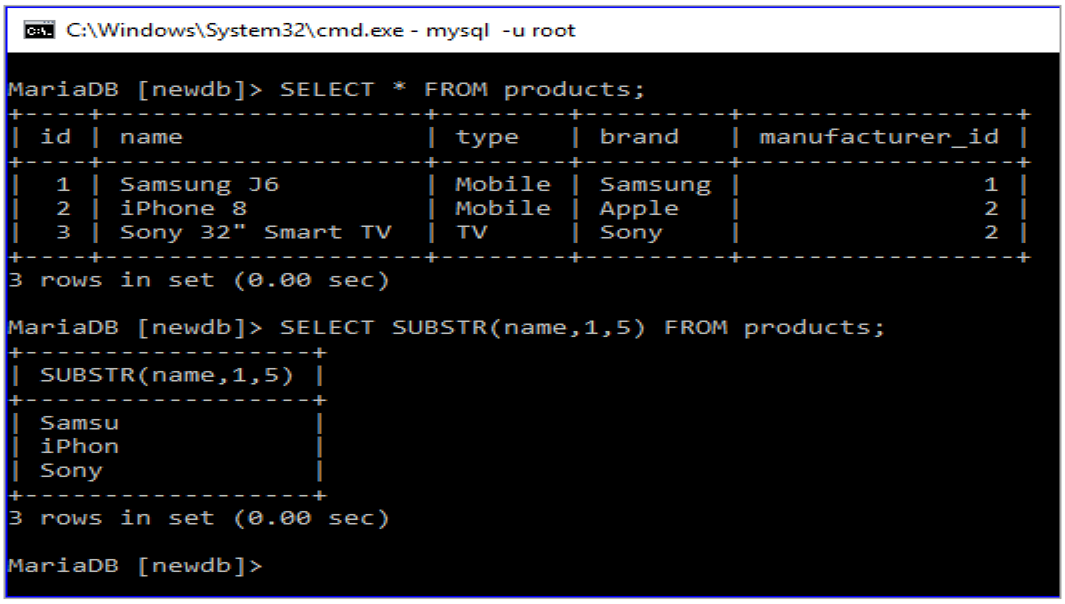
Answer: SUBSTR() function is used to retrieve the portion of any column. The use of this function is explained here with an example.

Example:

Here, the first SELECT command is used to show all the records of the Products table and the second SELECT command is executed using SUBSTR function and that prints only the first five characters of the name field.

SELECT * **FROM** products;

SELECT SUBSTR(**name**,1,5) **FROM** products;



```
C:\Windows\System32\cmd.exe - mysql  -u root

MariaDB [newdb]> SELECT * FROM products;
+----+--------------------+--------+---------+-----------------+
| id | name               | type   | brand   | manufacturer_id |
+----+--------------------+--------+---------+-----------------+
|  1 | Samsung J6         | Mobile | Samsung |               1 |
|  2 | iPhone 8           | Mobile | Apple   |               2 |
|  3 | Sony 32" Smart TV  | TV     | Sony    |               2 |
+----+--------------------+--------+---------+-----------------+
3 rows in set (0.00 sec)

MariaDB [newdb]> SELECT SUBSTR(name,1,5) FROM products;
+------------------+
| SUBSTR(name,1,5) |
+------------------+
| Samsu            |
| iPhon            |
| Sony             |
+------------------+
3 rows in set (0.00 sec)

MariaDB [newdb]>
```

Q #18) What is the purpose of using a HEAP table?

Answer: The table which uses a hashed index and stores in the memory is called the HEAP table. It works as a temporary table and it uses the indexes that make it faster than another table type.

When MySQL crashes for any reason then all the data stored in this table can be lost. It uses fixed-length data types. Hence BLOB and TEXT data types are not supported by this table. It is a useful table for those MySQL tasks where speed is the most important factor and temporary data is used.

Q #19) How can you add and remove any column of a table?

Answer: The syntax for adding any column in an existing table is shown below.

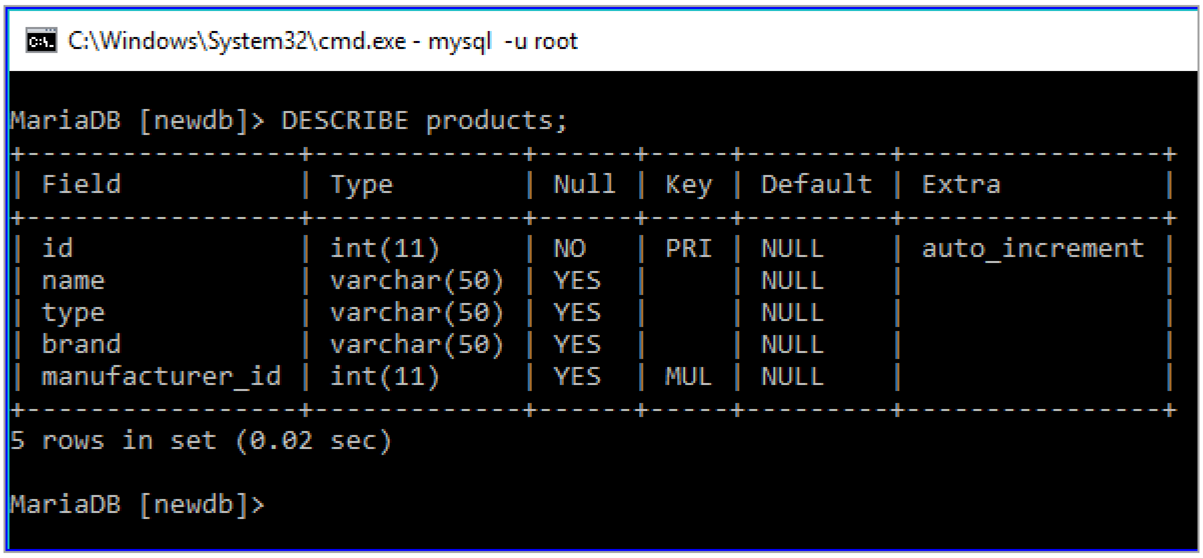
ALTER TABLE table_name **ADD COLUMN** column_name column_definition

[**FIRST**|**AFTER** existing_column]

Example:

DESCRIBE command is used to show the structure of the products table.

DESCRIBE products;



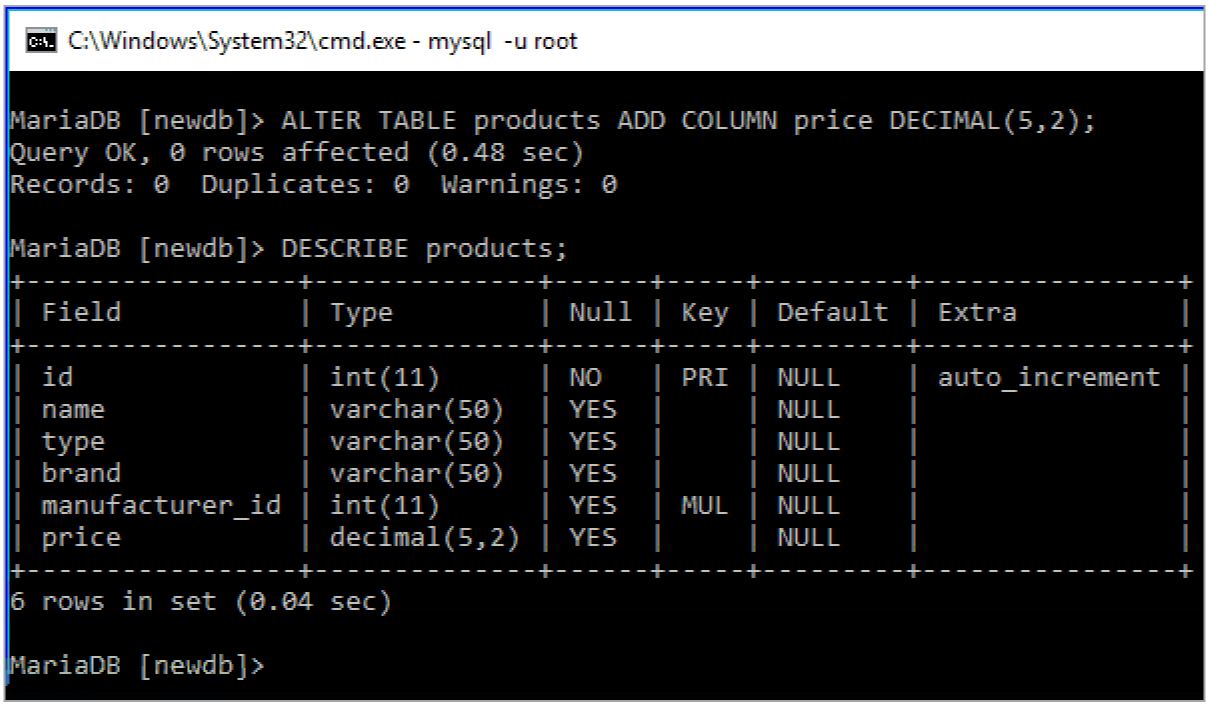
C:\Windows\System32\cmd.exe - mysql -u root

```
MariaDB [newdb]> DESCRIBE products;
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| id              | int(11)     | NO   | PRI | NULL    | auto_increment |
| name            | varchar(50) | YES  |     | NULL    |                |
| type            | varchar(50) | YES  |     | NULL    |                |
| brand           | varchar(50) | YES  |     | NULL    |                |
| manufacturer_id | int(11)     | YES  | MUL | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
5 rows in set (0.02 sec)

MariaDB [newdb]>
```

The following ALTER command with ADD COLUMN clause will add a new field named '**price**' in the table **products**.

ALTER TABLE products **ADD COLUMN** price **DECIMAL**(5,2);

DESCRIBE products;



C:\Windows\System32\cmd.exe - mysql -u root

```
MariaDB [newdb]> ALTER TABLE products ADD COLUMN price DECIMAL(5,2);
Query OK, 0 rows affected (0.48 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [newdb]> DESCRIBE products;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| id              | int(11)      | NO   | PRI | NULL    | auto_increment |
| name            | varchar(50)  | YES  |     | NULL    |                |
| type            | varchar(50)  | YES  |     | NULL    |                |
| brand           | varchar(50)  | YES  |     | NULL    |                |
| manufacturer_id | int(11)      | YES  | MUL | NULL    |                |
| price           | decimal(5,2) | YES  |     | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
6 rows in set (0.04 sec)

MariaDB [newdb]>
```

The syntax for removing any column from an existing table is shown below.

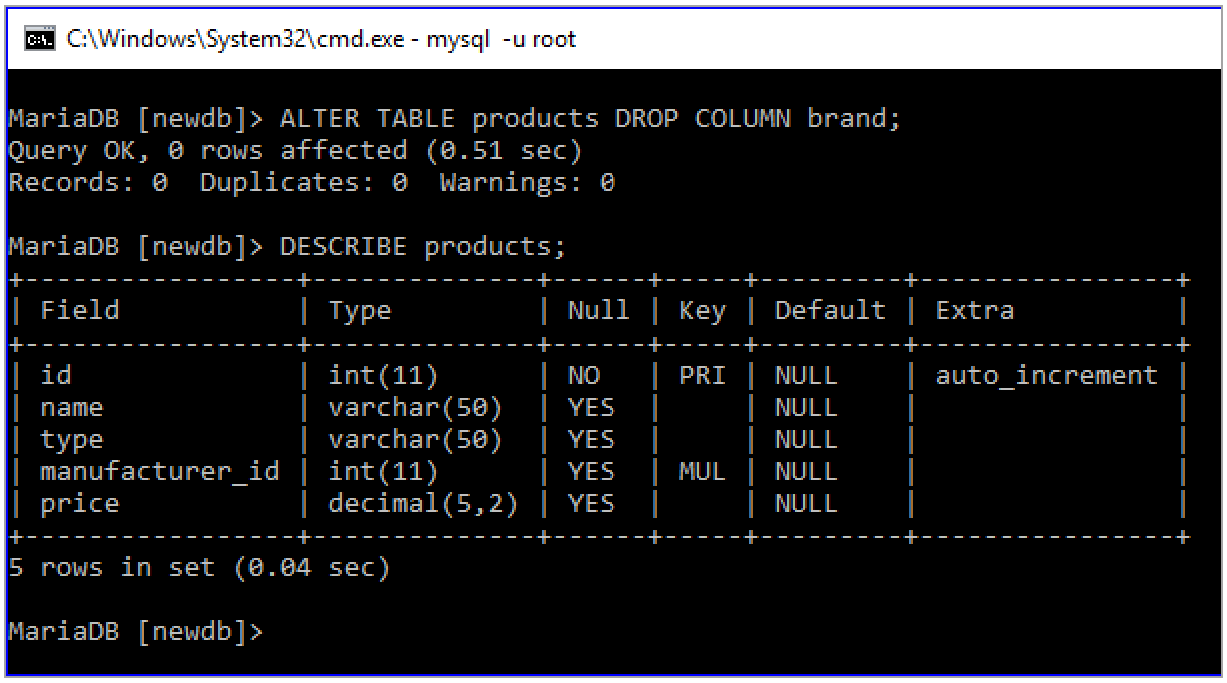
ALTER TABLE table_name **DROP COLUMN** column_name;

<u>Example:</u>

The following ALTER command with a DROP COLUMN clause will remove the field named '**brand**' in the table '**products**'.

ALTER TABLE products **DROP COLUMN** brand;

DESCRIBE products;



```
MariaDB [newdb]> ALTER TABLE products DROP COLUMN brand;
Query OK, 0 rows affected (0.51 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [newdb]> DESCRIBE products;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| id              | int(11)      | NO   | PRI | NULL    | auto_increment |
| name            | varchar(50)  | YES  |     | NULL    |                |
| type            | varchar(50)  | YES  |     | NULL    |                |
| manufacturer_id | int(11)      | YES  | MUL | NULL    |                |
| price           | decimal(5,2) | YES  |     | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
5 rows in set (0.04 sec)

MariaDB [newdb]>
```

Q #20) What is an index? How can an index be declared in MySQL?

Answer: An index is a data structure of a MySQL table that is used to speed up the queries.

It is used by the database search engine to find out the records faster. One or more fields of a table can be used as an index key. Index key can be assigned at the time of table declaration or can be assigned after creating the table.

<u>Example:</u>

username and **email** fields are set as the index in the following create table statement.

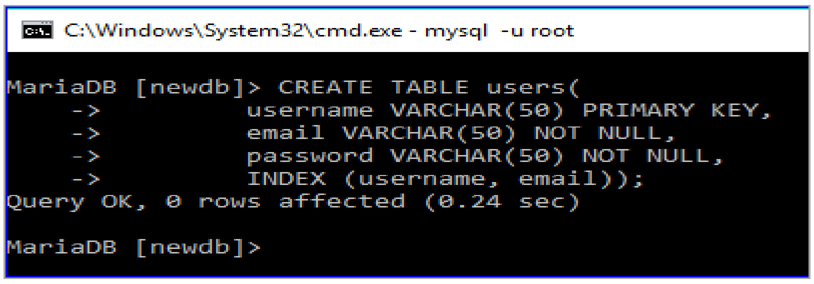
CREATE TABLE users(

username **VARCHAR**(50) **PRIMARY KEY**,

email **VARCHAR**(50) NOT NULL,

password VARCHAR(50) NOT NULL,

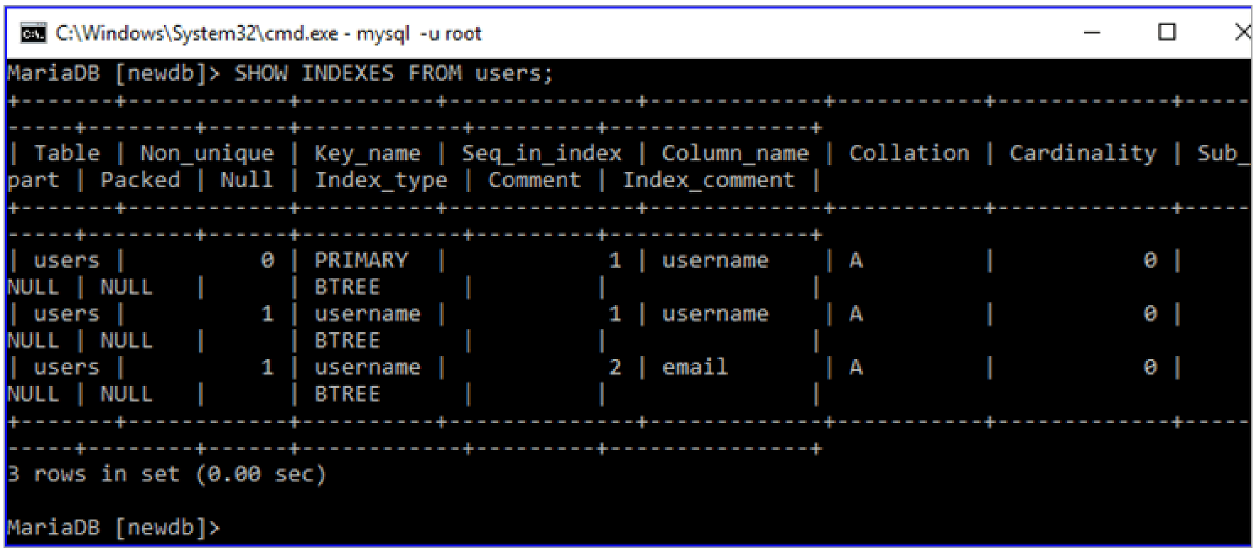
INDEX (username, email));



```
C:\Windows\System32\cmd.exe - mysql  -u root

MariaDB [newdb]> CREATE TABLE users(
    ->          username VARCHAR(50) PRIMARY KEY,
    ->          email VARCHAR(50) NOT NULL,
    ->          password VARCHAR(50) NOT NULL,
    ->          INDEX (username, email));
Query OK, 0 rows affected (0.24 sec)

MariaDB [newdb]>
```

The following command will show the index key information of the ‘**users’** table.

SHOW INDEXES **FROM** users;



```
C:\Windows\System32\cmd.exe - mysql  -u root
MariaDB [newdb]> SHOW INDEXES FROM users;
+-------+------------+----------+--------------+-------------+-----------+-------------+-----
----+--------+------+------------+---------+---------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_
part | Packed | Null | Index_type | Comment | Index_comment |
+-------+------------+----------+--------------+-------------+-----------+-------------+-----
----+--------+------+------------+---------+---------------+
| users |          0 | PRIMARY  |            1 | username    | A         |           0 |
NULL | NULL   |      | BTREE      |         |               |
| users |          1 | username |            1 | username    | A         |           0 |
NULL | NULL   |      | BTREE      |         |               |
| users |          1 | username |            2 | email       | A         |           0 |
NULL | NULL   |      | BTREE      |         |               |
+-------+------------+----------+--------------+-------------+-----------+-------------+-----
----+--------+------+------------+---------+---------------+
3 rows in set (0.00 sec)

MariaDB [newdb]>
```

Q #21) What is meant by a decimal (5,2)?

Answer: A decimal data type is used in MySQL to store the fractional data.

The meaning of decimal (5,2) means that the total length of the fractional value is 5. The field can contain 3 digits before the decimal point and 2 digits after the decimal point. If a user adds any value larger than the defined length then it will insert 999.99 in the field.

The use of this data type is explained in the following example.

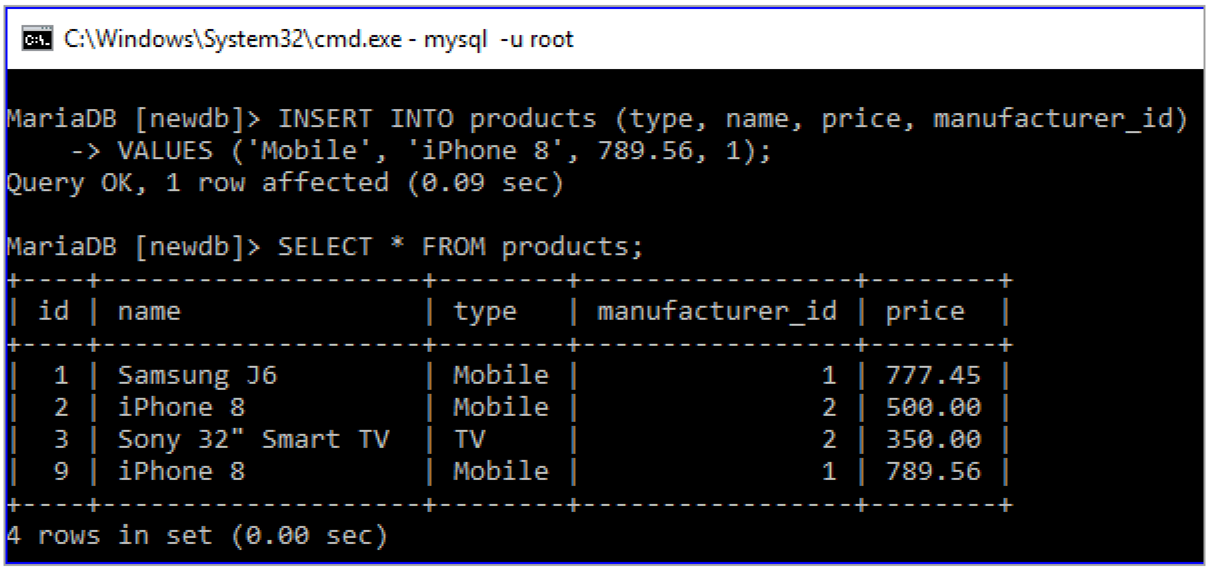
<u>**Example:**</u>

In the following insert query, **789.56** is inserted in the **price** field. This value is less than 1000 and the total digits with the fractional part are 5. So, this value is valid for this field.

INSERT INTO products (type, **name**, price, manufacturer_id)

VALUES ('Mobile', 'iPhone 8', 789.56, 1);

SELECT * **FROM** products;


C:\Windows\System32\cmd.exe - mysql -u root

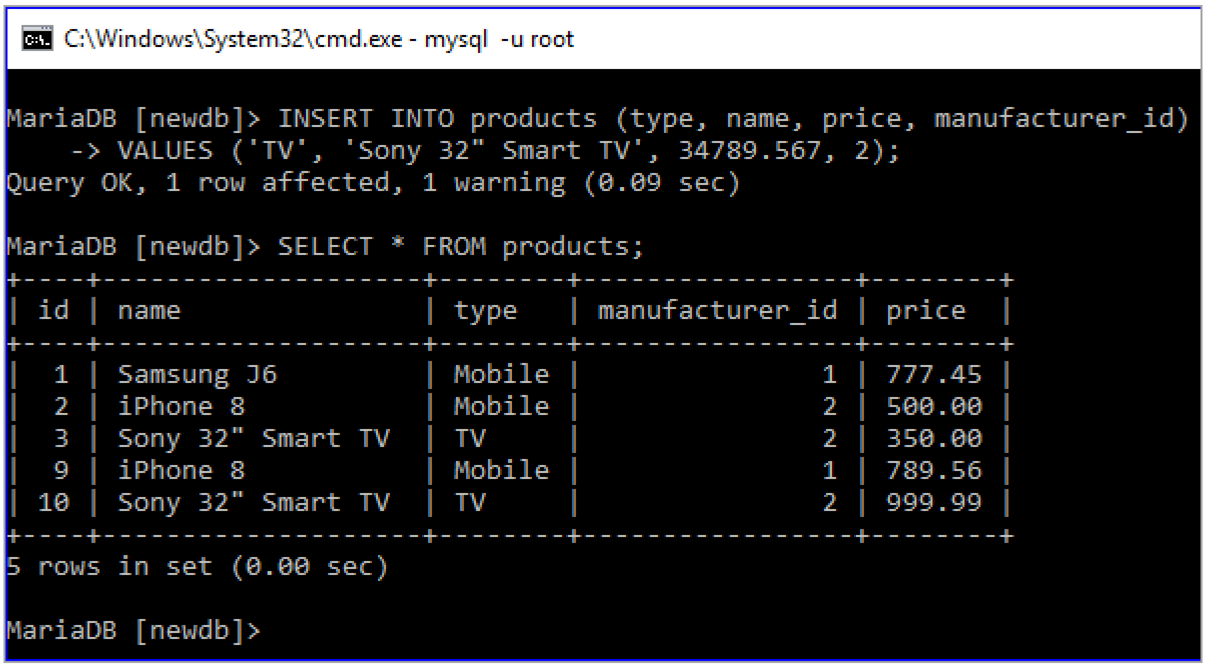
```
MariaDB [newdb]> INSERT INTO products (type, name, price, manufacturer_id)
    -> VALUES ('Mobile', 'iPhone 8', 789.56, 1);
Query OK, 1 row affected (0.09 sec)

MariaDB [newdb]> SELECT * FROM products;
+----+--------------------+--------+-----------------+--------+
| id | name               | type   | manufacturer_id | price  |
+----+--------------------+--------+-----------------+--------+
|  1 | Samsung J6         | Mobile |               1 | 777.45 |
|  2 | iPhone 8           | Mobile |               2 | 500.00 |
|  3 | Sony 32" Smart TV  | TV     |               2 | 350.00 |
|  9 | iPhone 8           | Mobile |               1 | 789.56 |
+----+--------------------+--------+-----------------+--------+
4 rows in set (0.00 sec)
```

In the following insert query, **34789.567** is set for the price field. Then this value is greater than 1000 and the total digits with fractional parts are 8. So, the default value 999.99 is inserted in the place of 34789.567**.**

INSERT INTO products (type, **name**, price, manufacturer_id)

VALUES('TV','Sony 32" Smart TV',34789.567, 2);

SELECT * **FROM** products;


C:\Windows\System32\cmd.exe - mysql -u root

```
MariaDB [newdb]> INSERT INTO products (type, name, price, manufacturer_id)
    -> VALUES ('TV', 'Sony 32" Smart TV', 34789.567, 2);
Query OK, 1 row affected, 1 warning (0.09 sec)

MariaDB [newdb]> SELECT * FROM products;
+----+--------------------+--------+-----------------+--------+
| id | name               | type   | manufacturer_id | price  |
+----+--------------------+--------+-----------------+--------+
|  1 | Samsung J6         | Mobile |               1 | 777.45 |
|  2 | iPhone 8           | Mobile |               2 | 500.00 |
|  3 | Sony 32" Smart TV  | TV     |               2 | 350.00 |
|  9 | iPhone 8           | Mobile |               1 | 789.56 |
| 10 | Sony 32" Smart TV  | TV     |               2 | 999.99 |
+----+--------------------+--------+-----------------+--------+
5 rows in set (0.00 sec)

MariaDB [newdb]>
```

Q #22) What is the view? How can you create and drop view in MySQL?

Answer: A view works as a virtual table that is used to store query and returns a result set when it is called. An updatable view is also supported by MySQL.

How a view can be created or deleted in MySQL are shown in the following examples.

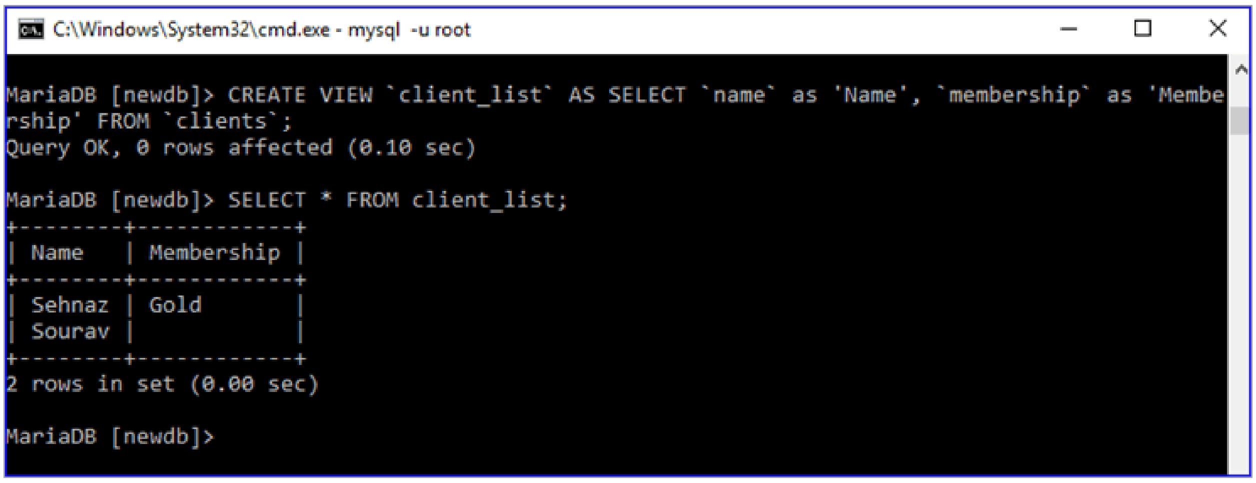
Create View Example:

The following statement will create a view file named '**client_list**' based on the table **clients**.

CREATE VIEW `client_list` **AS SELECT `name`** **as** 'Name', `membership` **as** 'Membership' **FRC**

SELECT statement will display the records of **client_list** value.

SELECT * **FROM** client_list;



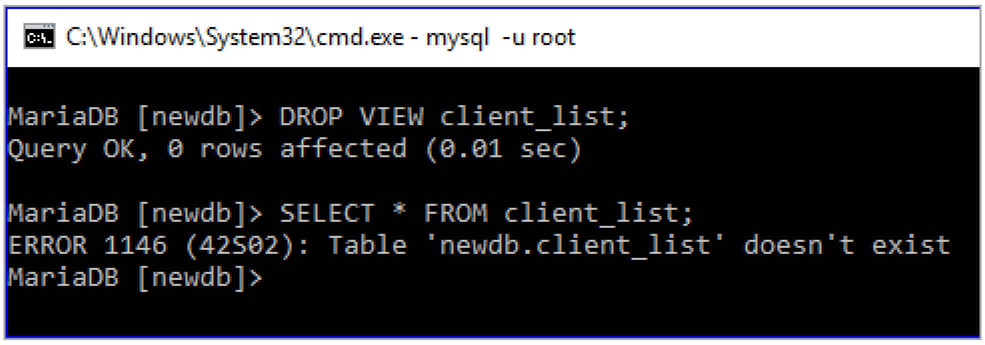
```
C:\Windows\System32\cmd.exe - mysql  -u root

MariaDB [newdb]> CREATE VIEW `client_list` AS SELECT `name` as 'Name', `membership` as 'Membe
rship' FROM `clients`;
Query OK, 0 rows affected (0.10 sec)

MariaDB [newdb]> SELECT * FROM client_list;
+--------+------------+
| Name   | Membership |
+--------+------------+
| Sehnaz | Gold       |
| Sourav |            |
+--------+------------+
2 rows in set (0.00 sec)

MariaDB [newdb]>
```

DROP View Example:

The drop view statement will delete the view file. The SELECT query will show an error after deleting the view.

DROP VIEW client_list;

SELECT * **FROM** client_list;



```
C:\Windows\System32\cmd.exe - mysql  -u root

MariaDB [newdb]> DROP VIEW client_list;
Query OK, 0 rows affected (0.01 sec)

MariaDB [newdb]> SELECT * FROM client_list;
ERROR 1146 (42S02): Table 'newdb.client_list' doesn't exist
MariaDB [newdb]>
```

Q #23) What is the function of mysqldump?

Answer: mysqldump is a useful utility tool of MySQL that is used to dump one or more or all databases from the server for backup or transfer to another database server.

Syntax:

For a single database,

mysqldump [OPTIONS] db_name [TABLES]

For multiple databases,

mysqldump [OPTIONS] –databases DB1 [DB2 DB3…]

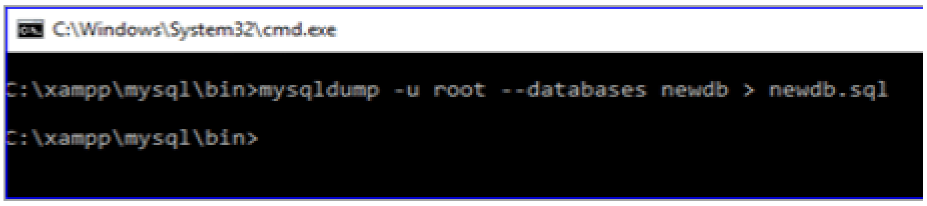
For all databases,

mysqldump [OPTIONS] –all-databases

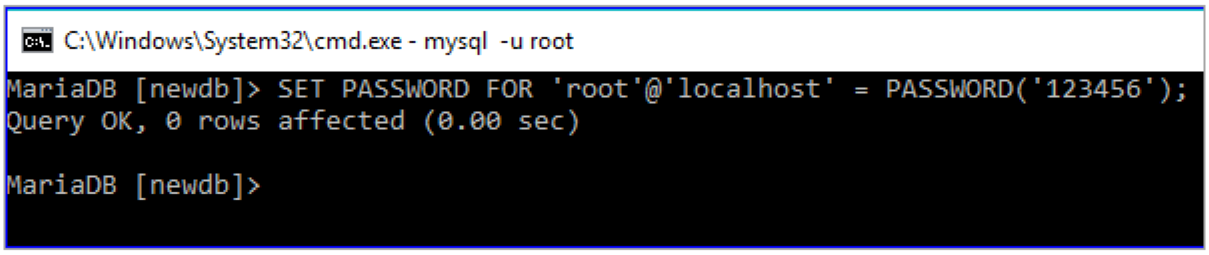
Example:

The following command will create a dump of the **'newdb'** database and export the content of the database in the file, **newdb.sql**.

mysqldump --databases newdb > newdb.sql



```
C:\xampp\mysql\bin>mysqldump -u root --databases newdb > newdb.sql

C:\xampp\mysql\bin>
```



```
MariaDB [newdb]> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('123456');
Query OK, 0 rows affected (0.00 sec)

MariaDB [newdb]>
```

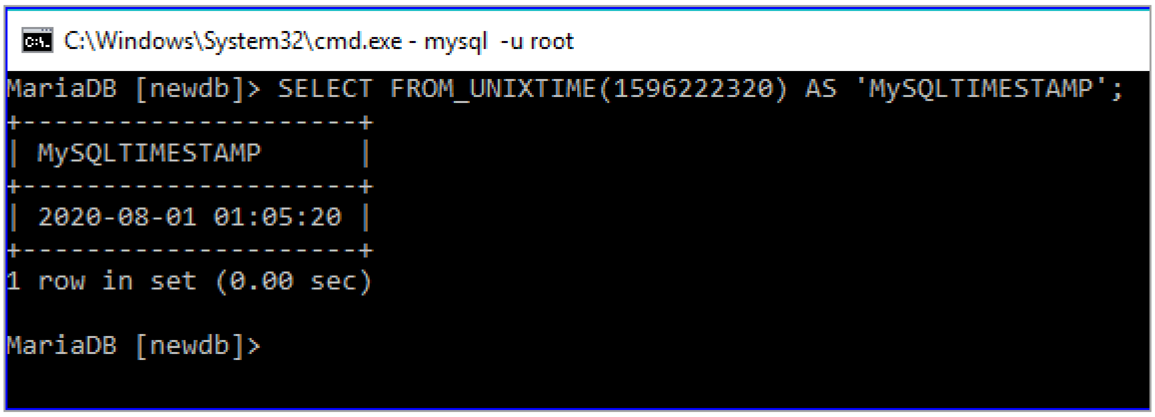
Q #25) What is the difference between UNIX TIMESTAMP and MySQL TIMESTAMP?

Answer: Both UNIX TIMESTAMP and MySQL TIMESTAMP are used to represent the date and time value. The main difference between these values is that UNIX TIMESTAMP represents the value by using 32-bits integers and MySQL TIMESTAMP represents the value in the human-readable format.

Example:

A UNIX time value is used by FROM_UNIXTIME function in the SELECT query to get the date and time value in the human-readable format.

SELECT FROM_UNIXTIME (1596222320) **AS** 'MySQLTIMESTAMP';

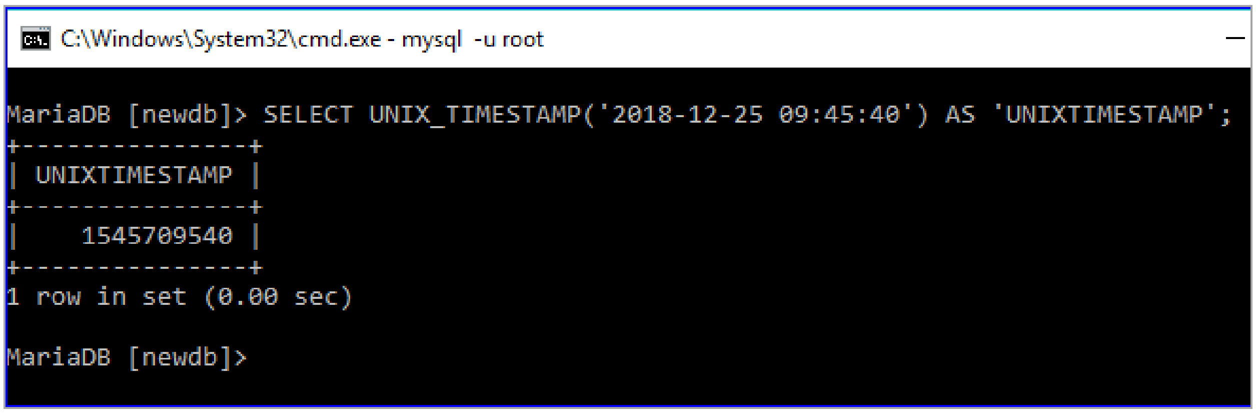


C:\Windows\System32\cmd.exe - mysql -u root

```
MariaDB [newdb]> SELECT FROM_UNIXTIME(1596222320) AS 'MySQLTIMESTAMP';
+---------------------+
| MySQLTIMESTAMP      |
+---------------------+
| 2020-08-01 01:05:20 |
+---------------------+
1 row in set (0.00 sec)

MariaDB [newdb]>
```

Date and time value is used by UNIX_TIMESTAMP function in the SELECT query to get the date and time value in the UNIX format.

SELECT UNIX_TIMESTAMP ('2018-12-25 09:45:40') **AS** 'UNIXTIMESTAMP';



C:\Windows\System32\cmd.exe - mysql -u root

```
MariaDB [newdb]> SELECT UNIX_TIMESTAMP('2018-12-25 09:45:40') AS 'UNIXTIMESTAMP';
+---------------+
| UNIXTIMESTAMP |
+---------------+
|    1545709540 |
+---------------+
1 row in set (0.00 sec)

MariaDB [newdb]>
```

Q #26) How can you import tables from a SQL file into a database by using the MySQL client?

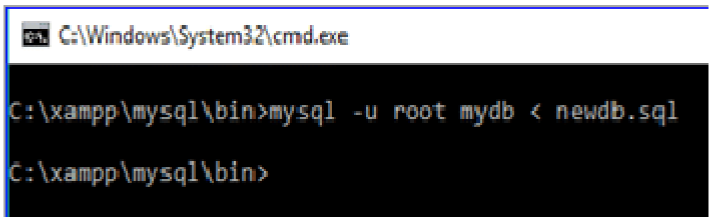
Answer: Database tables can be imported into a database from a SQL file by using the following MySQL statement.

mysql -u username -p database_name < sql_filename

<u>Example:</u>

If the root user's password is empty, then the following command will import tables from 'newdb.sql' file into the database **\`mydb\`.**

mysql -u root mydb < newdb.sql


C:\Windows\System32\cmd.exe

```
C:\xampp\mysql\bin>mysql -u root mydb < newdb.sql

C:\xampp\mysql\bin>
```

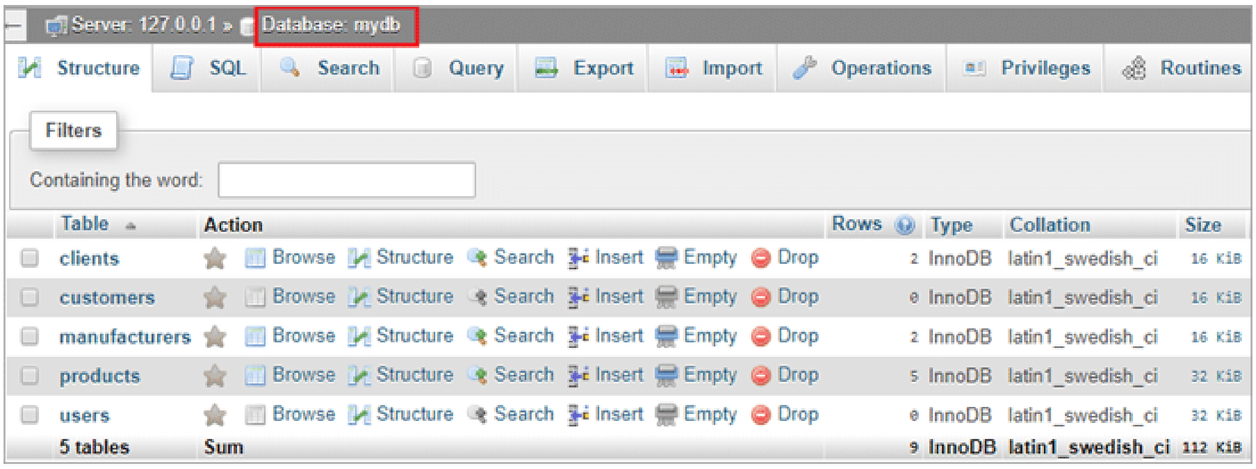


Table	Action	Rows	Type	Collation	Size
clients	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	16 KiB
customers	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	16 KiB
manufacturers	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	16 KiB
products	Browse Structure Search Insert Empty Drop	5	InnoDB	latin1_swedish_ci	32 KiB
users	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	32 KiB
5 tables	Sum	9	InnoDB	latin1_swedish_ci	112 KiB

Q #27) What is the difference between the Primary key and the Unique key?

Answer: Unique data is stored in the primary key and unique key fields. The primary key field never accepts NULL value but a unique key field accepts a NULL value.

Example:

In the **users'** table, the **id** field is the **primary key** and the **email** field is a **unique key**. Two records are inserted in the table where the email field is NULL for the 2nd record. The records are inserted properly as the unique field supports a NULL value.

INSERT INTO users (username, email, **password**)

VALUES('admin', 'admin@example.com', '7890'),

            ('staff', 'NULL', '1234');

SELECT * **FROM** users;

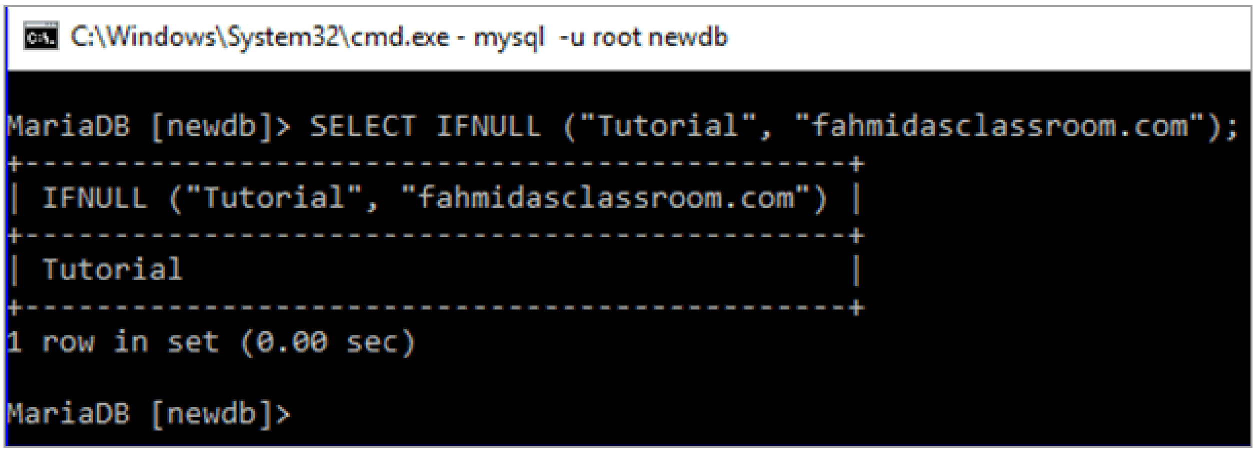
Q #28) What is the purpose of using the IFNULL() function?

Answer: **IFNULL() function** takes two arguments. It returns the first argument value if the value of the first argument is not NULL and it returns the second argument if the value of the first argument is NULL.

Example:

Here, the first argument of the IFNULL function is not NULL. So, the output is the first argument value.

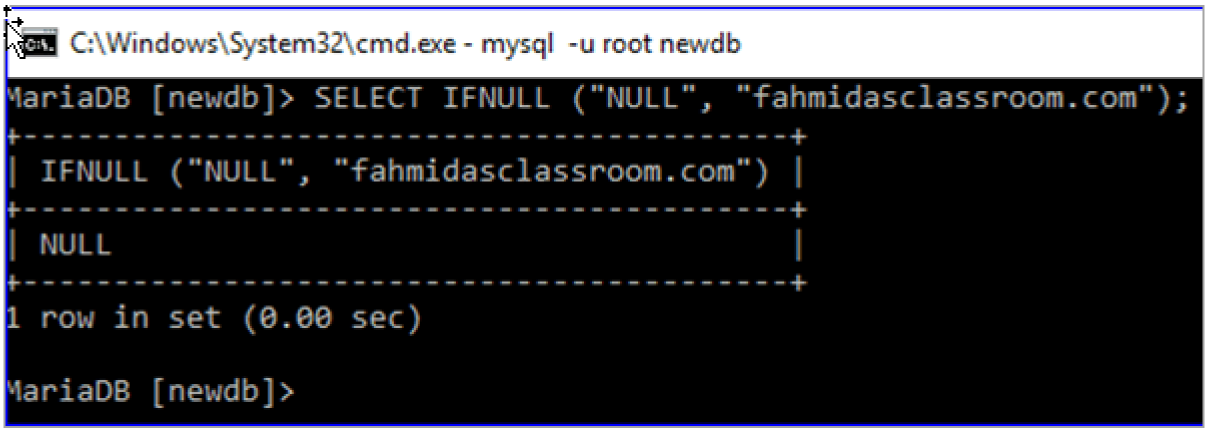
SELECT IFNULL ("Tutorial", "fahmidasclassroom.com");



```
C:\Windows\System32\cmd.exe - mysql  -u root newdb

MariaDB [newdb]> SELECT IFNULL ("Tutorial", "fahmidasclassroom.com");
+-------------------------------------------+
| IFNULL ("Tutorial", "fahmidasclassroom.com") |
+-------------------------------------------+
| Tutorial                                  |
+-------------------------------------------+
1 row in set (0.00 sec)

MariaDB [newdb]>
```

Here, the first argument of the IFNULL function is NULL. So, the output is NULL.

SELECT IFNULL ("NULL", "fahmidasclassroom.com");



```
C:\Windows\System32\cmd.exe - mysql  -u root newdb
MariaDB [newdb]> SELECT IFNULL ("NULL", "fahmidasclassroom.com");
+---------------------------------------+
| IFNULL ("NULL", "fahmidasclassroom.com") |
+---------------------------------------+
| NULL                                  |
+---------------------------------------+
1 row in set (0.00 sec)

MariaDB [newdb]>
```

Q #29) What is a join? Explain the different types of MySQL joins.

Answer: The SQL statement that is used to make a connection between two or more tables based on the matching columns is called a join. It is mainly used for complex queries.

Different types of SQL joins are mentioned below:

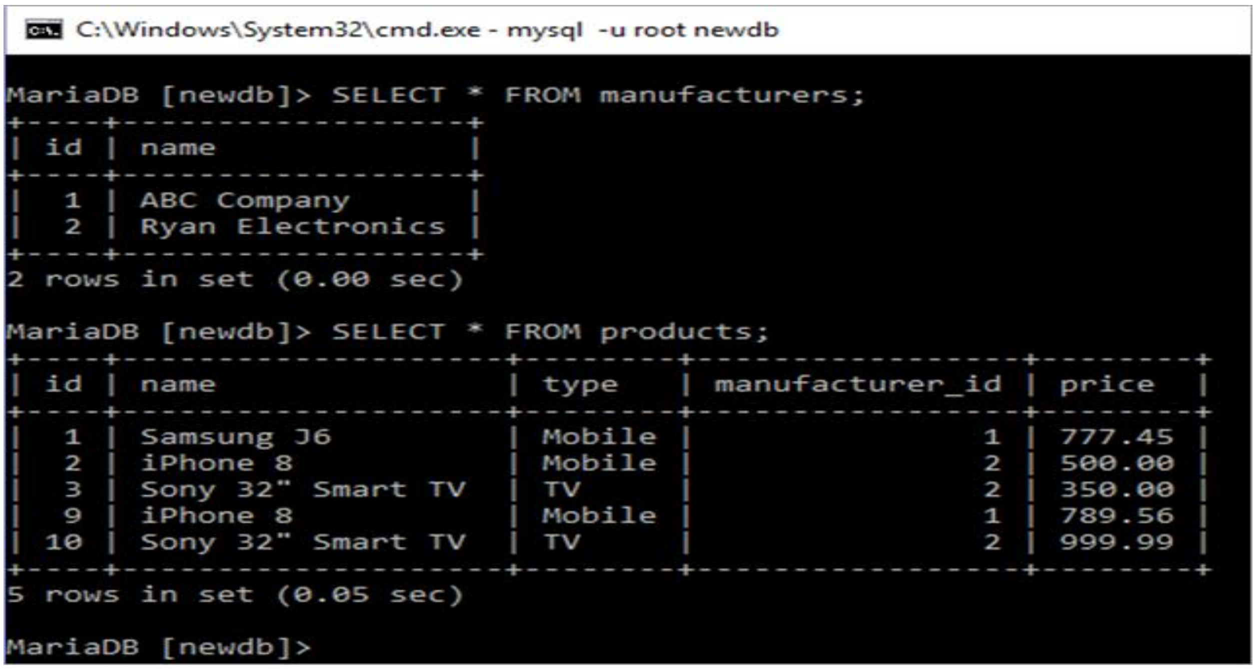
- **Inner Join**: It is a default join. It returns records when the values match in the joining tables.
- **Left Outer Join**: It returns all the records from the left table based on the matched records from the right table.
- **Right Outer Join**: It returns all the records from the right table based on the matched records from the left table.
- **Full Outer Join**: It returns all the records that match from the left or right table.

Example:

Two tables, **manufacturers** and **products** are used in this example to show the use of INNER JOIN. Here, SELECT queries are used to show the current records of these two tables.

SELECT * **FROM** manufacturers;

SELECT * **FROM** products;



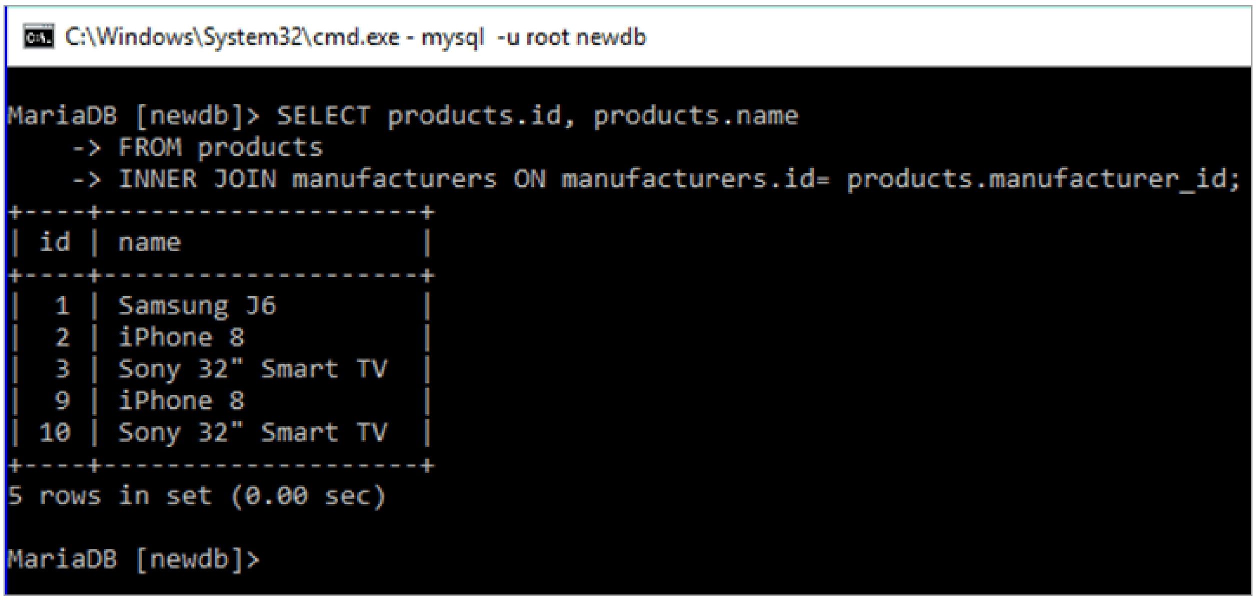
```
C:\Windows\System32\cmd.exe - mysql  -u root newdb

MariaDB [newdb]> SELECT * FROM manufacturers;
+----+-----------------+
| id | name            |
+----+-----------------+
|  1 | ABC Company     |
|  2 | Ryan Electronics |
+----+-----------------+
2 rows in set (0.00 sec)

MariaDB [newdb]> SELECT * FROM products;
+----+--------------------+--------+-----------------+--------+
| id | name               | type   | manufacturer_id | price  |
+----+--------------------+--------+-----------------+--------+
|  1 | Samsung J6         | Mobile |               1 | 777.45 |
|  2 | iPhone 8           | Mobile |               2 | 500.00 |
|  3 | Sony 32" Smart TV  | TV     |               2 | 350.00 |
|  9 | iPhone 8           | Mobile |               1 | 789.56 |
| 10 | Sony 32" Smart TV  | TV     |               2 | 999.99 |
+----+--------------------+--------+-----------------+--------+
5 rows in set (0.05 sec)

MariaDB [newdb]>
```

INNER JOIN is used in the following SELECT query where all the id and name of the products table will be displayed based on matching **manufacturer_id** of the **products** with an **id** of the **manufacturer's** table.

SELECT products.id, products.**name**

FROM products

INNER JOIN manufacturers **ON** manufacturers.id= products.manufacturer_id;



C:\Windows\System32\cmd.exe - mysql -u root newdb

```
MariaDB [newdb]> SELECT products.id, products.name
    -> FROM products
    -> INNER JOIN manufacturers ON manufacturers.id= products.manufacturer_id;
+----+-------------------+
| id | name              |
+----+-------------------+
|  1 | Samsung J6        |
|  2 | iPhone 8          |
|  3 | Sony 32" Smart TV |
|  9 | iPhone 8          |
| 10 | Sony 32" Smart TV |
+----+-------------------+
5 rows in set (0.00 sec)

MariaDB [newdb]>
```

Q #30) How can you retrieve a particular number of records from a table?

Answer: LIMIT clause is used with the SQL statement to retrieve a particular number of records from a table. From which record and how many records will be retrieved are defined by the LIMIT clause.

Syntax:

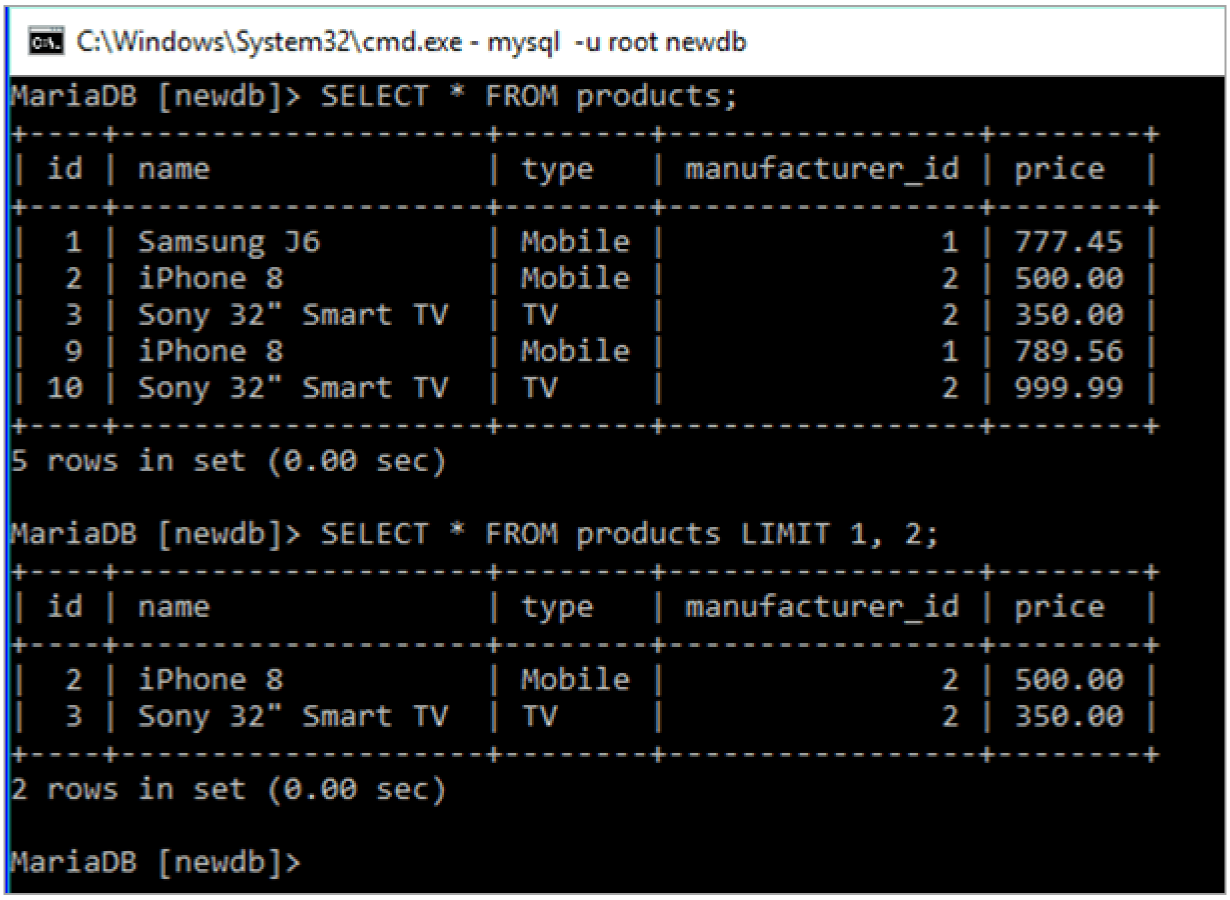
LIMIT starting_number, number_of_rows

Example:

Products table has 5 records which are displayed by the first SELECT query and the second SELECT query is used to display the records from 2nd to 3rd by using LIMIT 1, 2.

SELECT * **FROM** products;

SELECT * **FROM** products LIMIT 1, 2;



```
C:\Windows\System32\cmd.exe - mysql  -u root newdb
MariaDB [newdb]> SELECT * FROM products;
+----+--------------------+--------+-----------------+--------+
| id | name               | type   | manufacturer_id | price  |
+----+--------------------+--------+-----------------+--------+
|  1 | Samsung J6         | Mobile |               1 | 777.45 |
|  2 | iPhone 8           | Mobile |               2 | 500.00 |
|  3 | Sony 32" Smart TV  | TV     |               2 | 350.00 |
|  9 | iPhone 8           | Mobile |               1 | 789.56 |
| 10 | Sony 32" Smart TV  | TV     |               2 | 999.99 |
+----+--------------------+--------+-----------------+--------+
5 rows in set (0.00 sec)

MariaDB [newdb]> SELECT * FROM products LIMIT 1, 2;
+----+--------------------+--------+-----------------+--------+
| id | name               | type   | manufacturer_id | price  |
+----+--------------------+--------+-----------------+--------+
|  2 | iPhone 8           | Mobile |               2 | 500.00 |
|  3 | Sony 32" Smart TV  | TV     |               2 | 350.00 |
+----+--------------------+--------+-----------------+--------+
2 rows in set (0.00 sec)

MariaDB [newdb]>
```

Q #31) How can you export the table as an XML file in MySQL?

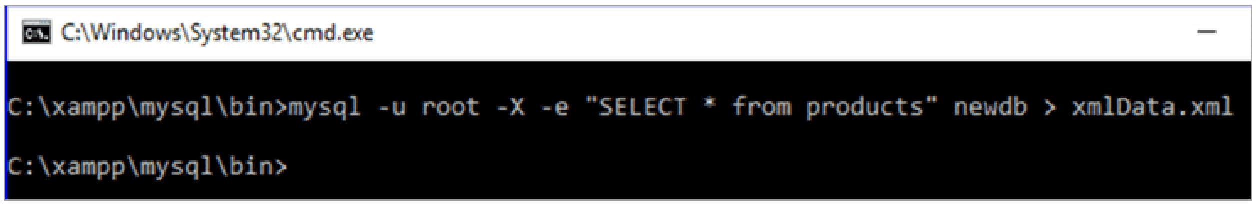
Answer: '-X' option is used with `mysql` command for exporting the file as XML. The following statement will export any table from a database as an XML file.

mysql -u username -X -e "SELECT query" database_name

<u>Example:</u>

The following command will export the data of the **items** table into **an xmlData.xml** file.

mysql -u root -X -e "SELECT * from products" newdb > xmlData.xml



```
C:\Windows\System32\cmd.exe
C:\xampp\mysql\bin>mysql -u root -X -e "SELECT * from products" newdb > xmlData.xml

C:\xampp\mysql\bin>
```

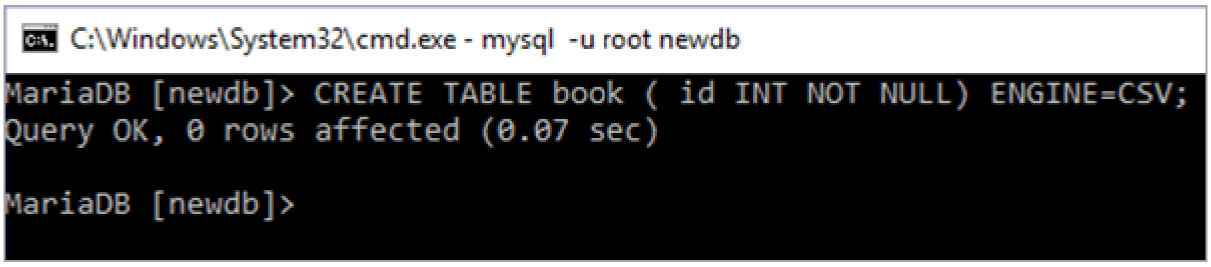
Q #32) What is a CSV table?

Answer: MySQL table that uses the CSV storage engine is called a CSV table. Data are stored as comma-separated values in the CSV table. MySQL server creates a data file with an extension '.csv' to store the content of the CSV table.

Example:

The following create statement will create a CSV file named book.

CREATE TABLE book (id **INT** NOT NULL) ENGINE=CSV;



C:\Windows\System32\cmd.exe - mysql -u root newdb

```
MariaDB [newdb]> CREATE TABLE book ( id INT NOT NULL) ENGINE=CSV;
Query OK, 0 rows affected (0.07 sec)

MariaDB [newdb]>
```

Q #33) How can you calculate the sum of any column of a table?

Answer: SUM() function is used to calculate the sum of any column.

Syntax:

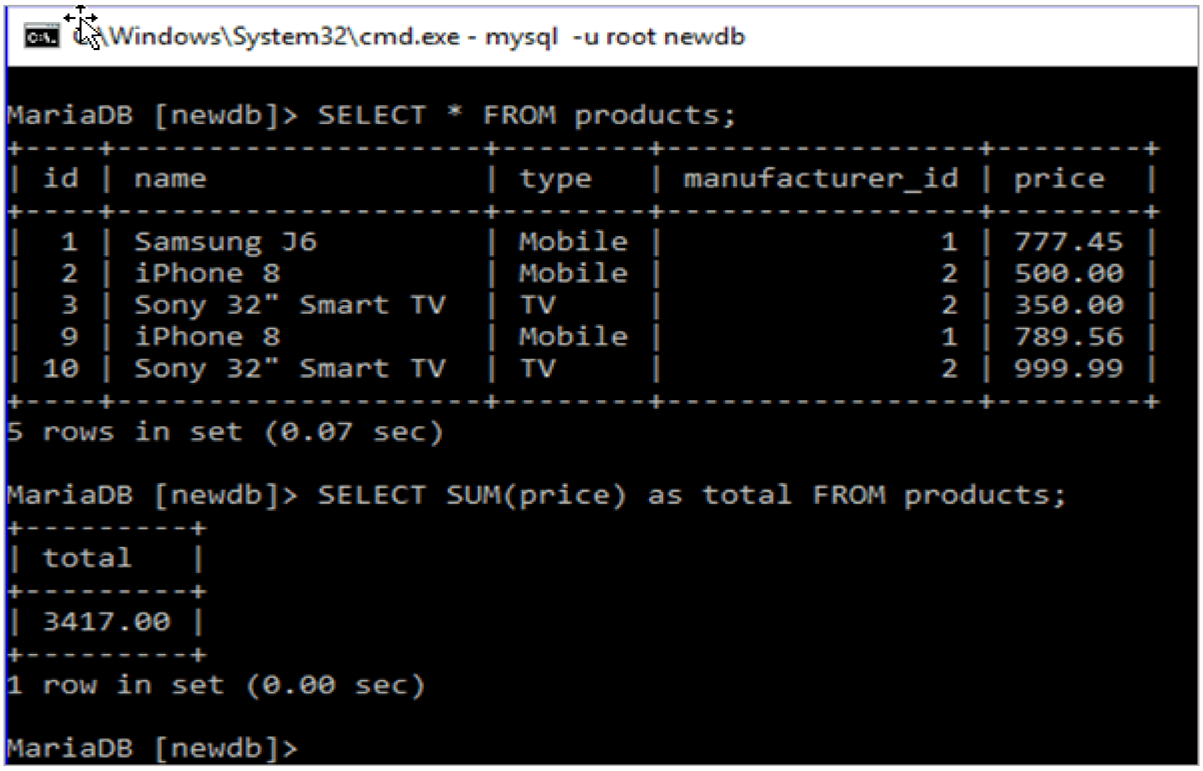
SUM(DISTINCT expression)

Example:

Products table has a numeric field named, price. In this example, the **SUM() function** is used to calculate the total value of the **price** field.

SELECT * **FROM** products;

SELECT SUM(price) **as** total **FROM** products;



C:\Windows\System32\cmd.exe - mysql -u root newdb

```
MariaDB [newdb]> SELECT * FROM products;
+----+--------------------+--------+-----------------+--------+
| id | name               | type   | manufacturer_id | price  |
+----+--------------------+--------+-----------------+--------+
|  1 | Samsung J6         | Mobile |               1 | 777.45 |
|  2 | iPhone 8           | Mobile |               2 | 500.00 |
|  3 | Sony 32" Smart TV  | TV     |               2 | 350.00 |
|  9 | iPhone 8           | Mobile |               1 | 789.56 |
| 10 | Sony 32" Smart TV  | TV     |               2 | 999.99 |
+----+--------------------+--------+-----------------+--------+
5 rows in set (0.07 sec)

MariaDB [newdb]> SELECT SUM(price) as total FROM products;
+---------+
| total   |
+---------+
| 3417.00 |
+---------+
1 row in set (0.00 sec)

MariaDB [newdb]>
```

Q #34) How can you count the total number of records of any table?

Answer: COUNT() function is used to count the total number of records of any table.

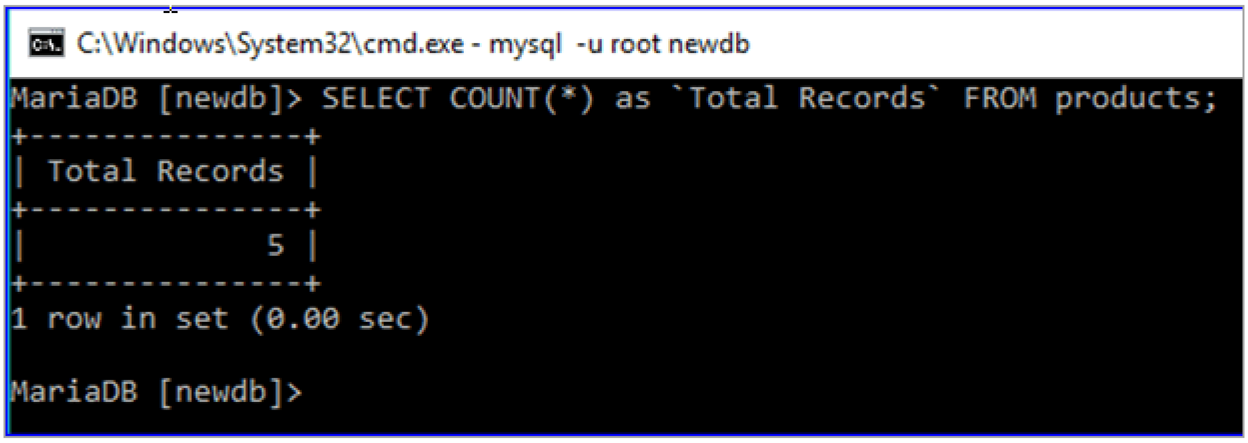
Syntax:

COUNT(expression)

Example:

The following SELECT query is used to count the total number of records of the **products** table.

SELECT COUNT(*) **as** `Total Records` **FROM** products;



C:\Windows\System32\cmd.exe - mysql -u root newdb

```
MariaDB [newdb]> SELECT COUNT(*) as `Total Records` FROM products;
+---------------+
| Total Records |
+---------------+
|             5 |
+---------------+
1 row in set (0.00 sec)

MariaDB [newdb]>
```

Q #35) Explain the difference between DELETE and TRUNCATE.

Answer: Both DELETE and TRUNCATE commands are used to delete the records from any database table. However, there are some significant differences between these commands. If the table contains the AUTO_INCREMENT PRIMARY KEY field then the effect of these commands can be shown properly.

Two differences between these commands are mentioned below.

1. DELETE command is used to delete a single or multiple or all the records from the table. The TRUNCATE command is used to delete all the records from the table or make the table empty.
2. When DELETE command is used to delete all the records from the table then it doesn't re-initialize the table. So, the AUTO_INCREMENT field does not count from one when the user inserts any record.

But when all the records of any table are deleted by using TRUNCATE command then it re-initializes the table and a new record will start from one for the AUTO_INCREMENT field.

Example:

The previously created user table is used in this example.

First, the SELECT query will show all the records of the user's table. DELETE query will delete all the records from the user's table. INSERT query will insert a new record into the user's table. After insert, if the SELECT query executes again then it will be shown that a new **id** is calculated after the deleted **id**.

SELECT * **FROM** users;

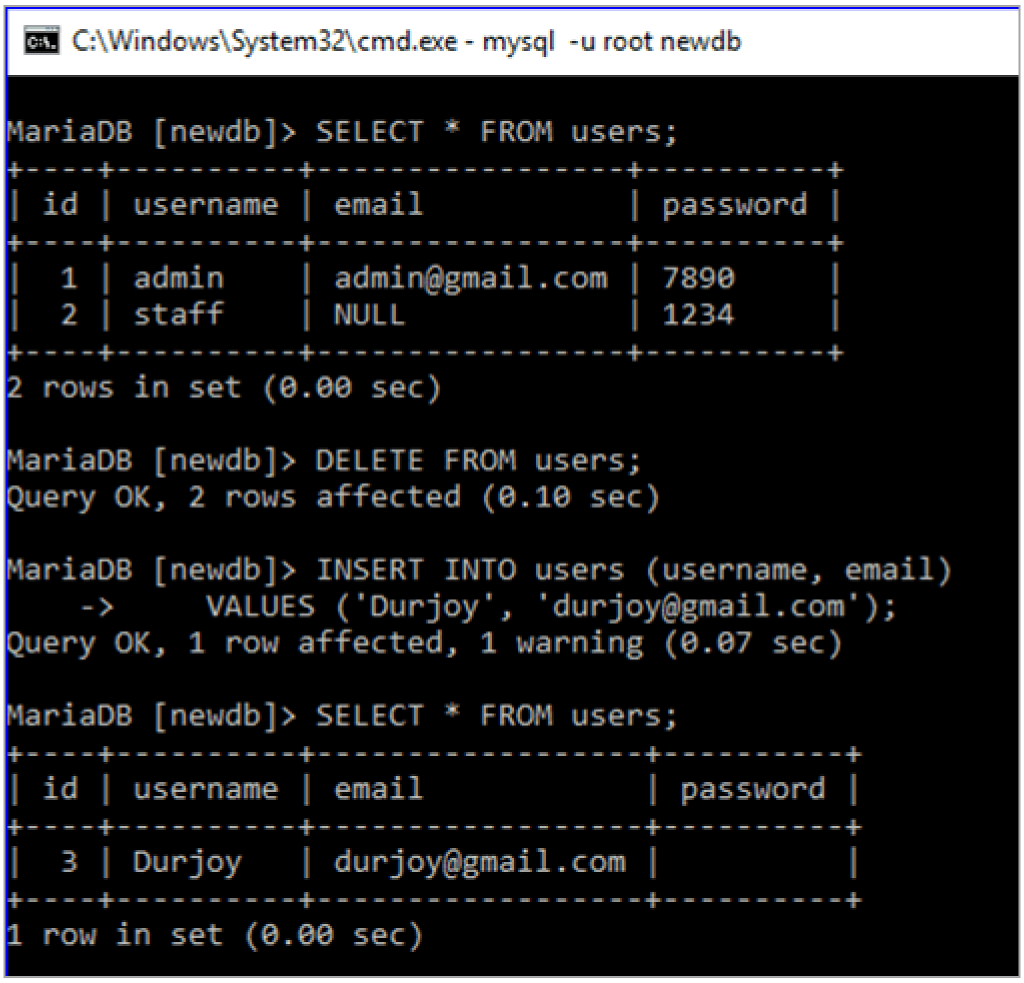
DELETE FROM users;

INSERT INTO users (username, email)

VALUES ('Durjoy', 'durjoy@gmail.com');

SELECT * **FROM** users;

Currently, there are two records in the user's table and when a new record is inserted after deleting all the records then the new id is 3, and not 1.



```
C:\Windows\System32\cmd.exe - mysql  -u root newdb

MariaDB [newdb]> SELECT * FROM users;
+----+----------+-----------------+----------+
| id | username | email           | password |
+----+----------+-----------------+----------+
|  1 | admin    | admin@gmail.com | 7890     |
|  2 | staff    | NULL            | 1234     |
+----+----------+-----------------+----------+
2 rows in set (0.00 sec)

MariaDB [newdb]> DELETE FROM users;
Query OK, 2 rows affected (0.10 sec)

MariaDB [newdb]> INSERT INTO users (username, email)
    ->      VALUES ('Durjoy', 'durjoy@gmail.com');
Query OK, 1 row affected, 1 warning (0.07 sec)

MariaDB [newdb]> SELECT * FROM users;
+----+----------+------------------+----------+
| id | username | email            | password |
+----+----------+------------------+----------+
|  3 | Durjoy   | durjoy@gmail.com |          |
+----+----------+------------------+----------+
1 row in set (0.00 sec)
```

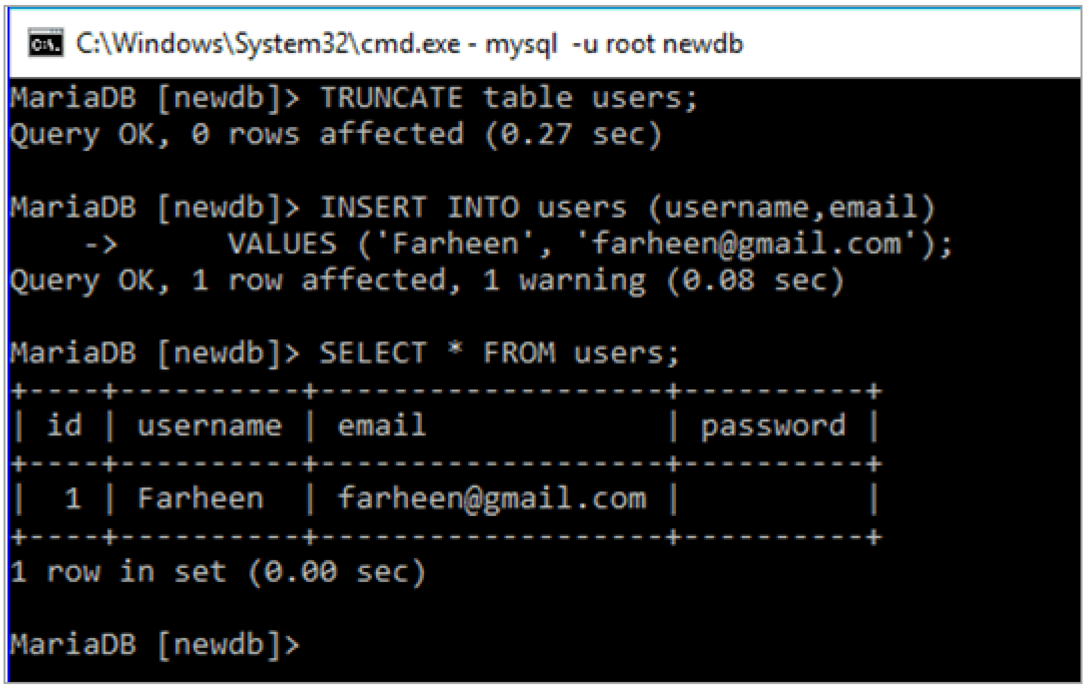
The same queries are executed in this part, just used the TRUNCATE statement in place of DELETE. It is shown that the id value of the new record is 1.

TRUNCATE table users;

INSERT INTO users (username, email)

VALUES ('Farheen', 'farheen@gmail.com');

SELECT * **FROM** users;



C:\Windows\System32\cmd.exe - mysql -u root newdb

```
MariaDB [newdb]> TRUNCATE table users;
Query OK, 0 rows affected (0.27 sec)

MariaDB [newdb]> INSERT INTO users (username,email)
    ->      VALUES ('Farheen', 'farheen@gmail.com');
Query OK, 1 row affected, 1 warning (0.08 sec)

MariaDB [newdb]> SELECT * FROM users;
+----+----------+-------------------+----------+
| id | username | email             | password |
+----+----------+-------------------+----------+
|  1 | Farheen  | farheen@gmail.com |          |
+----+----------+-------------------+----------+
1 row in set (0.00 sec)

MariaDB [newdb]>
```

Q #36) What is a storage engine? What are the differences between InnoDB and MyISAM engines?

Answer: One of the major components of the MySQL server is the storage engine for doing different types of database operations. Each database table created is based on the specific storage engine.

MySQL supports two types of storage engines i.e **transactional and non-transactional**. InnoDB is the default storage engine of MySQL which is transactional. MyISAM storage engine is a non-transactional storage engine.

The differences between InnoDB and MyISAM storage engines are discussed below:

- MyISAM supports the FULLTEXT index but InnoDB doesn’t support the FULLTEXT index.
- MyISAM is faster and InnoDB is slower.
- InnoDB supports ACID (Atomicity, Consistency, Isolation, and Durability) property but MyISAM doesn’t.
- InnoDB supports row-level locking and MyISAM supports table-level locking.
- InnoDB is suitable for large database and MyISAM is suitable for a small database.

Q #37) What is a transaction? Describe MySQL transaction properties.

Answer: When a group of database operations is done as a single unit then it is called a transaction. If any task of the transactional tasks remains incomplete then the transaction will not succeed. Hence, it is mandatory to complete all the tasks of a transaction to make the transaction successful.

A transaction has four properties which are known as ACID property. These properties are described below.

- **Atomicity:** It ensures that all the tasks of a transaction will be completed successfully otherwise all the completed tasks will be rolled back to the previous state for any failure.
- **Consistency:** It ensures that the database state must be changed accurately for the committed transaction.
- **Isolation:** It ensures that all the tasks of a transaction will be done independently and transparently.
- **Durability:** It ensures that all the committed transaction is consistent for any type of system failure.

Q #38) What are the functions of commit and rollback statements?

Answer: Commit is a transaction command that executes when all the tasks of a transaction are completed successfully. It will modify the database permanently to confirm the transaction.

Syntax:

```
COMMIT;
```

Rollback is another transactional command that executes when any of the transactional tasks become unsuccessful and undoes all the changes that are made by any transactional task to make the transaction unsuccessful.

Syntax:

```
ROLLBACK;
```

Q #39) What is the difference between MyISAM Static and MyISAM Dynamic?

Answer: MyISAM Static and MyISAM dynamic are the variations of the MyISAM storage engine. The differences between these tables are mentioned below.

- All the fields of MyISAM static table are of a fixed length and MyISAM dynamic table accepts variable length fields such as BLOB, TEXT, etc.
- After data corruption, it is easier to restore the MyISAM static table than MyISAM dynamic table.

Q #40) What is a trigger? How you can create a trigger in MySQL?

Answer: One of the important features of the MySQL database is a trigger that executes automatically when a particular database event occurs.

It fires after or before the execution of an insert or update or deletes a statement. It is a very useful option when a database user wants to do some database operations automatically.

Trigger Example:

If you want to delete the items of a supplier from the **items** table automatically after deleting the entry of the particular supplier from the '**suppliers'** table then write the trigger in the following way.

Example:

This is an example of **after delete trigger** that will fire automatically when any record is removed from the **manufacturer** table and deletes all the records from the **products** table where the deleted **id** of the **manufacturer** table matches with the **manufacturer_id** field of the **products** table.

DELIMITER //

CREATE TRIGGER manufacturer_after_delete

AFTER DELETE

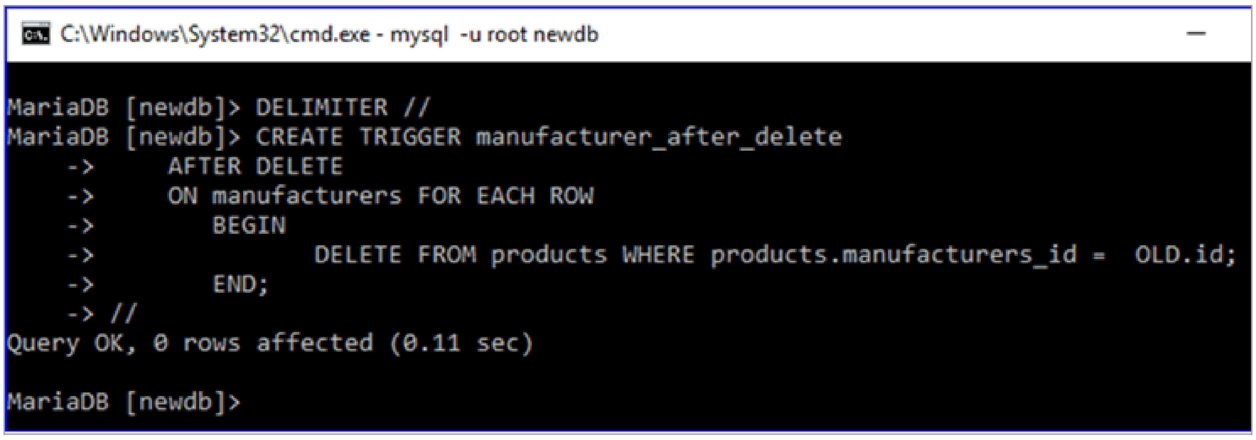
ON manufacturers **FOR** EACH ROW

BEGIN

DELETE FROM products **WHERE** products.manufacturers_id = OLD.id;

END;

//



C:\Windows\System32\cmd.exe - mysql -u root newdb

```
MariaDB [newdb]> DELIMITER //
MariaDB [newdb]> CREATE TRIGGER manufacturer_after_delete
    ->     AFTER DELETE
    ->     ON manufacturers FOR EACH ROW
    ->        BEGIN
    ->              DELETE FROM products WHERE products.manufacturers_id =  OLD.id;
    ->        END;
    -> //
Query OK, 0 rows affected (0.11 sec)

MariaDB [newdb]>
```

Table – EmployeeDetails

EmpId	FullName	ManagerId	DateOfJoining	City
121	John Snow	321	01/31/2014	Toronto
321	Walter White	986	01/30/2015	California
421	Kuldeep Rana	876	27/11/2016	New Delhi

Table – EmployeeSalary

EmpId	Project	Salary	Variable
121	P1	8000	500
321	P2	10000	1000
421	P1	12000	0

For your convenience, I have compiled the top 10 questions for you. You can try solving these questions and click on the links to go to their respective answers.

1. SQL Query to fetch records that are present in one table but not in another table.
2. SQL query to fetch all the employees who are not working on any project.
3. SQL query to fetch all the Employees from EmployeeDetails who joined in the Year 2020.
4. Fetch all employees from EmployeeDetails who have a salary record in EmployeeSalary.
5. Write an SQL query to fetch project-wise count of employees.
6. Fetch employee names and salary even if the salary value is not present for the employee.
7. Write an SQL query to fetch all the Employees who are also managers.

8. Write an SQL query to fetch duplicate records from EmployeeDetails.
9. Write an SQL query to fetch only odd rows from the table.
10. Write a query to find the 3rd highest salary from a table without top or limit keyword.

Ques.1. Write an SQL query to fetch the EmpId and FullName of all the employees working under Manager with id – '986'.
Ans. We can use the EmployeeDetails table to fetch the employee details with a where clause for the manager-

```
SELECT  EmpId, FullName

FROM EmployeeDetails

WHERE ManagerId = 986;
```

Ques.2. Write an SQL query to fetch the different projects available from the EmployeeSalary table.
Ans. While referring to the EmployeeSalary table, we can see that this table contains project values corresponding to each employee, or we can say that we will have duplicate project values while selecting Project values from this table.

So, we will use the distinct clause to get the unique values of the Project.

```
SELECT DISTINCT(Project)

FROM EmployeeSalary;
```

Ques.3. Write an SQL query to fetch the count of employees working in project 'P1'.
Ans. Here, we would be using aggregate function count() with the SQL **where** clause-

```
SELECT COUNT(*)

FROM EmployeeSalary

WHERE Project = 'P1';
```

Ques.4. Write an SQL query to find the maximum, minimum, and average salary of the employees.
Ans. We can use the aggregate function of SQL to fetch the max, min, and average values-

```
SELECT Max(Salary),

Min(Salary),

AVG(Salary)

FROM EmployeeSalary;
```

Ques.5. Write an SQL query to find the employee id whose salary lies in the range of 9000 and 15000.
Ans. Here, we can use the ‘Between’ operator with a where clause.

SELECT EmpId, Salary

FROM EmployeeSalary

WHERE Salary BETWEEN 9000 AND 15000;

Ques.6. Write an SQL query to fetch those employees who live in Toronto and work under manager with ManagerId – 321.
Ans. Since we have to satisfy both the conditions – employees living in ‘Toronto’ and working in Project ‘P2’. So, we will use AND operator here-

SELECT EmpId, City, ManagerId

FROM EmployeeDetails

WHERE City='Toronto' AND ManagerId='321';

Ques.7. Write an SQL query to fetch all the employees who either live in California or work under a manager with ManagerId – 321.
Ans. This interview question requires us to satisfy either of the conditions – employees living in ‘California’ and working under Manager with ManagerId ‘321’. So, we will use the OR operator here-

SELECT EmpId, City, ManagerId

FROM EmployeeDetails

WHERE City='California' OR ManagerId='321';

Ques.8. Write an SQL query to fetch all those employees who work on Project other than P1.
Ans. Here, we can use the NOT operator to fetch the rows which are not satisfying the given condition.

SELECT EmpId

FROM EmployeeSalary

WHERE NOT Project='P1';

Or using the not equal to operator-

SELECT EmpId

FROM EmployeeSalary

WHERE Project <> 'P1';

For the difference between NOT and <> SQL operators, check this link – Difference between the NOT and != operators.

Ques.9. Write an SQL query to display the total salary of each employee adding the Salary with Variable value.
Ans. Here, we can simply use the '+' operator in SQL.

```
SELECT EmpId,

Salary+Variable as TotalSalary

FROM EmployeeSalary;
```

Ques.10. Write an SQL query to fetch the employees whose name begins with any two characters, followed by a text "hn" and ending with any sequence of characters.
Ans. For this question, we can create an SQL query using like operator with '_' and '%' wild card characters, where '_' matches a single character and '%' matches '0 or multiple characters'.

```
SELECT FullName

FROM EmployeeDetails

WHERE FullName LIKE '__hn%';
```

Ques.11. Write an SQL query to fetch all the EmpIds which are present in either of the tables – 'EmployeeDetails' and 'EmployeeSalary'.
Ans. In order to get unique employee ids from both the tables, we can use Union clause which can combine the results of the two SQL queries and return unique rows.

```
SELECT EmpId FROM EmployeeDetails

UNION

SELECT EmpId FROM EmployeeSalary;
```

Ques.12. Write an SQL query to fetch common records between two tables.
Ans. SQL Server – Using INTERSECT operator-

```
SELECT * FROM EmployeeSalary

INTERSECT

SELECT * FROM ManagerSalary;
```

MySQL – Since MySQL doesn't have INTERSECT operator so we can use the sub query-

```
SELECT *
```

```
FROM EmployeeSalary
WHERE EmpId IN
(SELECT EmpId from ManagerSalary);
```

Ques.13. Write an SQL query to fetch records that are present in one table but not in another table.

Ans. SQL Server – Using MINUS- operator-

```
SELECT * FROM EmployeeSalary
MINUS
SELECT * FROM ManagerSalary;
```

MySQL – Since MySQL doesn’t have MINUS operator so we can use LEFT join-

```
SELECT EmployeeSalary.*
FROM EmployeeSalary
LEFT JOIN
ManagerSalary USING (EmpId)
WHERE ManagerSalary.EmpId IS NULL;
```

Ques.14. Write an SQL query to fetch the EmpIds that are present in both the tables – ‘EmployeeDetails’ and ‘EmployeeSalary.

Ans. Using sub query-

```
SELECT EmpId FROM
EmployeeDetails
where EmpId IN
(SELECT EmpId FROM EmployeeSalary);
```

Ques.15. Write an SQL query to fetch the EmpIds that are present in EmployeeDetails but not in EmployeeSalary.

Ans. Using sub query-

```
SELECT EmpId FROM
EmployeeDetails
where EmpId Not IN
(SELECT EmpId FROM EmployeeSalary);
```

Ques.16. Write an SQL query to fetch the employee full names and replace the space with '-'.
Ans. Using 'Replace' function-

```
SELECT REPLACE(FullName, ' ', '-')

FROM EmployeeDetails;
```

Ques.17. Write an SQL query to fetch the position of a given character(s) in a field.
Ans. Using 'Instr' function-

```
SELECT INSTR(FullName, 'Snow')

FROM EmployeeDetails;
```

Ques.18. Write an SQL query to display both the EmpId and ManagerId together.
Ans. Here we can use the CONCAT command.

```
SELECT CONCAT(EmpId, ManagerId) as NewId

FROM EmployeeDetails;
```

Ques.19. Write a query to fetch only the first name(string before space) from the FullName column of the EmployeeDetails table.
Ans. In this question, we are required to first fetch the location of the space character in the FullName field and then extract the first name out of the FullName field.

For finding the location we will use the LOCATE method in MySQL and CHARINDEX in SQL SERVER and for fetching the string before space, we will use the SUBSTRING OR MID method.

MySQL – using MID

```
SELECT MID(FullName, 1, LOCATE(' ',FullName))

FROM EmployeeDetails;
```

SQL Server – using SUBSTRING

```
SELECT SUBSTRING(FullName, 1, CHARINDEX(' ',FullName))

FROM EmployeeDetails;
```

Ques.20. Write an SQL query to upper case the name of the employee and lower case the city values.
Ans. We can use SQL Upper and Lower functions to achieve the intended results.

```
SELECT UPPER(FullName), LOWER(City)

FROM EmployeeDetails;
```

Ques.21. Write an SQL query to find the count of the total occurrences of a particular character – 'n' in the FullName field.
Ans. Here, we can use the 'Length' function. We can subtract the total length of the FullName field with a length of the FullName after replacing the character – 'n'.

```
SELECT FullName,

LENGTH(FullName) - LENGTH(REPLACE(FullName, 'n', ''))

FROM EmployeeDetails;
```

Ques.22. Write an SQL query to update the employee names by removing leading and trailing spaces.
Ans. Using the 'Update' command with the 'LTRIM' and 'RTRIM' function.

```
UPDATE EmployeeDetails

SET FullName = LTRIM(RTRIM(FullName));
```

Ques.23. Fetch all the employees who are not working on any project.
Ans. This is one of the very basic interview questions in which the interviewer wants to see if the person knows about the commonly used – Is NULL operator.

```
SELECT EmpId

FROM EmployeeSalary

WHERE Project IS NULL;
```

Ques.24. Write an SQL query to fetch employee names having a salary greater than or equal to 5000 and less than or equal to 10000.
Ans. Here, we will use BETWEEN in the 'where' clause to return the EmpId of the employees with salary satisfying the required criteria and then use it as subquery to find the fullName of the employee from EmployeeDetails table.

```
SELECT FullName

FROM EmployeeDetails

WHERE EmpId IN

(SELECT EmpId FROM EmployeeSalary

WHERE Salary BETWEEN 5000 AND 10000);
```

Ques.25. Write an SQL query to find the current date-time.
Ans. MySQL-

SELECT NOW();

SQL Server-

SELECT getdate();

Oracle-

SELECT SYSDATE FROM DUAL;

Ques.26. Write an SQL query to fetch all the Employees details from EmployeeDetails table who joined in the Year 2020.
Ans. Using BETWEEN for the date range ’01-01-2020′ AND ’31-12-2020′-

SELECT * FROM EmployeeDetails

WHERE DateOfJoining BETWEEN '2020/01/01'

AND '2020/12/31';

Also, we can extract year part from the joining date (using YEAR in mySQL)-

SELECT * FROM EmployeeDetails

WHERE YEAR(DateOfJoining) = '2020';

Ques.27. Write an SQL query to fetch all employee records from EmployeeDetails table who have a salary record in EmployeeSalary table.
Ans. Using ‘Exists’-

SELECT * FROM EmployeeDetails E

WHERE EXISTS

(SELECT * FROM EmployeeSalary S

WHERE E.EmpId = S.EmpId);

Ques.28. Write an SQL query to fetch project-wise count of employees sorted by project’s count in descending order.
Ans. The query has two requirements – first to fetch the project-wise count and then to sort the result by that count.

For project-wise count, we will be using the GROUP BY clause and for sorting, we will use the ORDER BY clause on the alias of the project-count.

SELECT Project, count(EmpId) EmpProjectCount

FROM EmployeeSalary

```
GROUP BY Project
ORDER BY EmpProjectCount DESC;
```

Ques.29. Write a query to fetch employee names and salary records. Display the employee details even if the salary record is not present for the employee.
Ans. This is again one of the very common interview questions in which the interviewer just wants to check the basic knowledge of SQL JOINS.

Here, we can use left join with EmployeeDetail table on the left side of the EmployeeSalary table.

```
SELECT E.FullName, S.Salary
FROM EmployeeDetails E
LEFT JOIN
EmployeeSalary S
ON E.EmpId = S.EmpId;
```

Ques.30. Write an SQL query to join 3 tables.
Ans. Considering 3 tables TableA, TableB, and TableC, we can use 2 joins clauses like below-

```
SELECT column1, column2
FROM TableA
JOIN TableB ON TableA.Column3 = TableB.Column3
JOIN TableC ON TableA.Column4 = TableC.Column4;
```

Here is the list of some of the most frequently asked SQL query interview questions for experienced professionals. These questions cover SQL queries on advanced SQL JOIN concepts, fetching duplicate rows, odd and even rows, nth highest salary, etc.

Ques. 31. Write an SQL query to fetch all the Employees who are also managers from the EmployeeDetails table.
Ans. Here, we have to use Self-Join as the requirement wants us to analyze the EmployeeDetails table as two tables. We will use different aliases ‘E’ and ‘M’ for the same EmployeeDetails table.

```
SELECT DISTINCT E.FullName
FROM EmployeeDetails E
INNER JOIN EmployeeDetails M
ON E.EmpID = M.ManagerID;
```

Ques.32. Write an SQL query to fetch duplicate records from EmployeeDetails (without considering the primary key – EmpId).
Ans. In order to find duplicate records from the table, we can use GROUP BY on all the fields and then use the HAVING clause to return only those fields whose count is greater than 1 i.e. the rows having duplicate records.

```
SELECT FullName, ManagerId, DateOfJoining, City, COUNT(*)

FROM EmployeeDetails

GROUP BY FullName, ManagerId, DateOfJoining, City

HAVING COUNT(*) > 1;
```

Ques.33. Write an SQL query to remove duplicates from a table without using a temporary table.
Ans. Here, we can use delete with alias and inner join. We will check for the equality of all the matching records and them remove the row with higher EmpId.

```
DELETE E1 FROM EmployeeDetails E1

INNER JOIN EmployeeDetails E2

WHERE E1.EmpId > E2.EmpId

AND E1.FullName = E2.FullName

AND E1.ManagerId = E2.ManagerId

AND E1.DateOfJoining = E2.DateOfJoining

AND E1.City = E2.City;
```

Ques.34. Write an SQL query to fetch only odd rows from the table.
Ans. In case we have an auto-increment field e.g. EmpId then we can simply use the below query-

```
SELECT * FROM EmployeeDetails

WHERE MOD (EmpId, 2) <> 0;
```

In case we don't have such a field then we can use the below queries.

Using Row_number in SQL server and checking that the remainder when divided by 2 is 1-

```
SELECT E.EmpId, E.Project, E.Salary

FROM (

   SELECT *, Row_Number() OVER(ORDER BY EmpId) AS RowNumber

   FROM EmployeeSalary
) E
WHERE E.RowNumber % 2 = 1;
```

Using a user defined variable in MySQL-

```
SELECT *
FROM (
      SELECT *, @rowNumber := @rowNumber+ 1 rn
      FROM EmployeeSalary
      JOIN (SELECT @rowNumber:= 0) r
     ) t
WHERE rn % 2 = 1;
```

Ques.35. Write an SQL query to fetch only even rows from the table.
Ans. In case we have an auto-increment field e.g. EmpId then we can simply use the below query-

```
SELECT * FROM EmployeeDetails
WHERE MOD (EmpId, 2) = 0;
```

In case we don’t have such a field then we can use the below queries.

Using Row_number in SQL server and checking that the remainder when divided by 2 is 1-

```
SELECT E.EmpId, E.Project, E.Salary
FROM (
    SELECT *, Row_Number() OVER(ORDER BY EmpId) AS RowNumber
    FROM EmployeeSalary
) E
WHERE E.RowNumber % 2 = 0;
```

Using a user defined variable in MySQL-

```
SELECT *
FROM (
      SELECT *, @rowNumber := @rowNumber+ 1 rn
      FROM EmployeeSalary
      JOIN (SELECT @rowNumber:= 0) r
    ) t
WHERE rn % 2 = 0;
```

Ques.36. Write an SQL query to create a new table with data and structure copied from another table.
Ans.

```
CREATE TABLE NewTable
SELECT * FROM EmployeeSalary;
```

Ques.37. Write an SQL query to create an empty table with the same structure as some other table.
Ans. Here, we can use the same query as above with False 'WHERE' condition-

```
CREATE TABLE NewTable
SELECT * FROM EmployeeSalary where 1=0;
```

Ques.38. Write an SQL query to fetch top n records?
Ans. In MySQL using LIMIT-

```
SELECT *
FROM EmployeeSalary
ORDER BY Salary DESC LIMIT N;
```

In SQL server using TOP command-

```
SELECT TOP N *
FROM EmployeeSalary
ORDER BY Salary DESC;
```

Ques.39. Write an SQL query to find the nth highest salary from table.
Ans, Using Top keyword (SQL Server)-

```
SELECT TOP 1 Salary
FROM (
      SELECT DISTINCT TOP N Salary
      FROM Employee
      ORDER BY Salary DESC
```

```
    )
ORDER BY Salary ASC;
```

Using limit clause(MySQL)-

```
SELECT Salary
FROM Employee
ORDER BY Salary DESC LIMIT N-1,1;
```

Ques.40. Write SQL query to find the 3rd highest salary from a table without using the TOP/limit keyword.

Ans. This is one of the most commonly asked interview questions. For this, we will use a correlated subquery.

In order to find the 3rd highest salary, we will find the salary value until the inner query returns a count of 2 rows having the salary greater than other distinct salaries.

```
SELECT Salary
FROM EmployeeSalary Emp1
WHERE 2 = (
            SELECT COUNT( DISTINCT ( Emp2.Salary ) )
            FROM EmployeeSalary Emp2
            WHERE Emp2.Salary > Emp1.Salary
        )
```

For nth highest salary-

```
SELECT Salary
FROM EmployeeSalary Emp1
WHERE N-1 = (
            SELECT COUNT( DISTINCT ( Emp2.Salary ) )
            FROM EmployeeSalary Emp2
            WHERE Emp2.Salary > Emp1.Salary
        )
```

2. Why is MySQL so popular?

First of all, MySQL is open-source. Second, it is widely adopted, so a lot of code is already available. Even entire developed systems are there that can be referred to for the upcoming projects. MySQL has relational databases; hence it makes it have methodical storage rather than a big dump of unorganized mess. And finally, as said earlier, MySQL is quick and robust.

3. What are the tables in MySQL? Explain the types.

This is a must-know **MySQL interview question**. Let's see the answer-

MySQL stores everything in logical tables. Tables can be thought of as the core storage structure of MySQL. And hence tables are also known as storage engines. Here are the storage engines provided by MySQL:

· **MyISAM** – MyISAM is the default storage engine for MySQL. It extends the former ISAM storage engine. MyISAM offers big storage, up to 256TB! The tables can also be compressed to get extra storage. MyISAM tables are not transaction-safe.

· **MERGE** – A MERGE table is a virtual table that consolidates different MyISAM tables that have a comparable structure to one table. MERGE tables use the indexes of the base tables, as they do not have indexes of their own.

· **ARCHIVE** – As the name suggests, Archive helps in archiving the tables by compressing them, in-turn reducing the storage space. Hence, you can store a lot of records with the Archive. It uses the compression-decompression procedure while writing and reading the table records. It is done using the Zlib library.

· **CSV** – This is more like a storage format. CSV engine stores the values in the Comma-separated values (CSV) format. This engine makes it easier to migrate the tables into a non-SQL pipeline.

· **InnoDB** – InnoDB is the most optimal while choosing an engine to drive performance. InnoDB is a transaction-safe engine. Hence it is ACID-compliant and can efficiently restore your database to the most stable state in case of a crash.

· **Memory**– Memory tables were formerly known as HEAP. With memory tables, there can be a performance boost as the tables are stored in the memory. But it does not work with large data tables due to the same reason.

· **Federated** – Federated tables allow accessing remote MySQL server tables. It can be done without any third-party integration or cluster technology.

Read: SQL for Data Science: Why SQL, List of Benefits & Commands

4. Write a query for a column addition in MySQL

For this, an ALTER TABLE query is required. Once invoked, simply mention the column and its definition. Something like this:

```
ALTER TABLE cars

ADD COLUMN engine VARCHAR(80) AFTER colour;
```

5. What is a foreign key? Write a query to implement the same in MySQL.

A foreign key is used to connect two tables. A FOREIGN KEY is a field (or assortment of it) in one table that alludes to the PRIMARY KEY in another table. The FOREIGN KEY requirement is utilised to forestall activities that would crush joins between tables.

To assign a foreign key, it is important to mention it while creating the table. It can be assigned by invoking the FOREIGN KEY query. Something like this:

FOREIGN KEY (Any_ID) REFERENCES Table_to_reference(Any_ID)

7. How does database import/export work in MySQL?

It can be done in two ways. One is to use phpMyAdmin, and the second is to use the command line access of MySQL. The latter can be done by using the command named mysqldump. It goes something like this:

· mysqldump -u username -p databasename > dbsample.sql

To import a database into MySQL, only a sign change is required, with a command of MySQL. The command goes something like this:

· mysql -u username -p databasename < dbsample.sql

8. How can we delete a column or a row in MySQL?

Now dropping a column can be simply done by using the ALTER TABLE command and then using the DROP command. It goes something like this:

ALTER TABLE table_name DROP column name;

To drop a row, first, an identification for the row is required. Once that is handy, use the DELETE command in conjunction with the conditional WHERE command. Something like this:

DELETE FROM cars WHERE carID = 3;

9. What are the different ways to join tables in MySQL?

Join is used to link one or more tables together, with the common column's values in both tables. Primarily there are four types of joins:

1. Inner Join – Inner join uses a join predicate, which is a condition used to make the join. Here is the syntax:

SELECT something FROM tablename INNER JOIN another table ON condition;

2. Left Join – Left join also requires a join condition. The left join chooses information beginning from the left table. For each entry in the left table, the left compares each entry in the right table. Here is the syntax:

SELECT something FROM tablename LEFT JOIN another table ON condition;

3. Right Join – Opposite to left join and, with one difference in the query, that is the name of join. Here care should be taken about the order of tables. Here is the syntax:

SELECT something FROM tablename LEFT JOIN another table ON condition;

4. Cross Join – Cross join has no join condition. It makes a cartesian of rows of both the tables. Here is the syntax:

SELECT something FROM tablename CROSS JOIN another table;

Note: While dealing with just one table, self-join is also possible.

It is one of the most dealt with **MySQL interview questions**. Interviewers do like to see if the candidate understands the basics or not and join one of the core concepts.

Read: PHP Interview Questions & Answers

10. Can a primary key be dropped in MySQL? If yes, how?

Yes, it is possible to drop the primary key from a table. The command to use is again, the ALTER TABLE followed by DROP. It goes like this:

ALTER TABLE table_name DROP PRIMARY KEY;

11. What are Procedures in MySQL?

Procedures (or stored procedures) are subprograms, just like in a regular language, embedded in the database. A stored procedure consists of a name, SQL statement(s) and parameters. It utilises the caching in MySQL and hence saves time and memory, just like the prepared statements.

12. What is a trigger in MySQL?

A trigger is a table-associated database object in MySQL. It is activated when a specified action takes place.

A trigger can be invoked after or before the event takes place. It can be used on INSERT, DELETE, and UPDATE. It uses the respective syntax to define the triggers. For example, BEFORE INSERT, AFTER DELETE, etc.

13. How to add users in MySQL?

To simply put, the user can be added by using the CREATE command and specifying the necessary credentials. First, log in to the MySQL account and then apply the syntax. Something like this:

CREATE USER ‘testuser’ IDENTIFIED BY ‘sample password’;

Users can be granted permissions, by the following commands:

GRANT SELECT ON * . * TO ‘testuser’;

14. What is the core difference between Oracle and MySQL?

The core difference is that MySQL works on a single-model database. That means it can only work with one base structure, while Oracle is a multi-model database. It means it can support various data models like graph, document, key-value, etc.

Another fundamental difference is that Oracle's support comes with a price tag for industrial solutions. While MySQL is open-source.

Now this question is one of the **MySQL interview questions** that should be understood carefully. Because it directly deals with the industry standards and what the company wants.

15. What is CHAR and VARCHAR in MySQL?

Both of them define a string. The core difference is that CHAR is a fixed-length while VARCHAR is variable length. For example, if CHAR(5) is defined, then it needs exactly five characters. If VARCHAR(5) is defined, then it can take at most five characters. VARCHAR can be said to have more efficiency in the usage of memory as it can have dynamic memory allocations.

17. What is a LIKE statement? Explain % and _ in LIKE.

While using filters in commands like SELECT, UPDATE, and DELETE, conditions might require a pattern to detect. LIKE is used to do just that. LIKE has two wildcard characters, namely % (percentage) and _ (underscore). Percentage(%) matches a string of characters, while underscore matches a single character.

For example, %t will detect trees and tea both. However, _t will only detect one extra character, i.e., strings like ti or te.

18. How to convert timestamps to date in MySQL?

It is a rather simple question that requires knowledge on two commands, like DATE_FORMAT and FROM_UNIXTIME.

DATE_FORMAT(FROM_UNIXTIME(`date_in_timestamp`), '%e %b %Y') AS 'date_formatted'

19. Can a query be written in any case in MySQL?

This MySQL interview question often confuses people who are just getting started with MySQL. Although most of the time, the queries are written in capital or some in small letters, there is no such case sensitivity to MySQL queries.

For example, both create table tablename and CREATE TABLE tablename, works fine.

However, if required, it is possible to make the query case sensitive by using the keyword BINARY.

This **MySQL interview question** can be tricky, especially when asked to make the query case-sensitive explicitly.

20. How to save images in MySQL?

Images can be stored in the MySQL database by converting them to BLOBS. But it is not preferred due to the large overhead it creates. Plus, it puts unnecessary load on the RAM while loading the entire database. It is hence preferred to store the paths in the database and store the images on disk.

21. How to get multiple condition results from data in MySQL?

There are two ways to do so. The first is to use the keyword OR while using the WHERE condition. The other is to use a list of values to check and use IN with WHERE.

22. What are the different file formats used by MyISAM?

Typically, a MyISAM table is stored using three files on disk. The data file and the index file, which are defined with extensions .MYD and .MYI, respectively. There is a table definition file that has .frm extension.

23. How does DISTINCT work in MySQL?

DISTINCT is used to avoid the problem of duplicity while fetching the results of a particular query. DISTINCT is used to make sure the results do not contain repeated values. DISTINCT can be used with the SELECT clause. Here is the syntax for it:

SELECT DISTINCT something FROM tablename;

24. Is there any upper limit for the number of columns in a table?

Although the exact size limitation depends on many factors, MySQL has a hard limit on max size to be 4096 columns. But as said, for a given table, the effective-maximum may be less.

25. What are Access Control Lists or ACLs, in accordance with MySQL?

The ACLs or Access control lists are used in a way to give a guideline for security in the MySQL database. MySQL provides security based on ACLs for all the tasks performed by users like connection requests, queries, and any other operation.

26. How to make connections persistent in MySQL?

While making a connection request, if Mysql_pconnect is used rather than mysql_connect, then it can make the connection persistent. Here ‘p’ means persistent. The database connection is not closed every time.

27. Explain the SAVEPOINT statement in MySQL.

SAVEPOINT is a way of making sub-transactions in MySQL, which are also known as nested transactions.

SAVEPOINT marks a point in a regular transaction. It indicates a point to which the system can rollback.

1. What is Database?

A database is an organized collection of data, stored and retrieved digitally from a remote or local computer system. Databases can be vast and complex, and such databases are developed using fixed design and modeling approaches.

2. What is DBMS?

DBMS stands for Database Management System. DBMS is a system software responsible for the creation, retrieval, updation, and management of the database. It ensures that our data is consistent, organized, and is easily accessible by serving as an interface between the database and its end-users or application software.

3. What is RDBMS? How is it different from DBMS?

RDBMS stands for Relational Database Management System. The key difference here, compared to DBMS, is that RDBMS stores data in the form of a collection of tables, and relations can be defined between the common fields of these tables. Most modern database management systems like MySQL, Microsoft SQL Server, Oracle, IBM DB2, and Amazon Redshift are based on RDBMS.

4. What is SQL?

SQL stands for Structured Query Language. It is the standard language for relational database management systems. It is especially useful in handling organized data comprised of entities (variables) and relations between different entities of the data.

5. What is the difference between SQL and MySQL?

SQL is a standard language for retrieving and manipulating structured databases. On the contrary, MySQL is a relational database management system, like SQL Server, Oracle or IBM DB2, that is used to manage SQL databases.

6. What are Tables and Fields?

A table is an organized collection of data stored in the form of rows and columns. Columns can be categorized as vertical and rows as horizontal. The columns in a table are called fields while the rows can be referred to as records.

7. What are Constraints in SQL?

Constraints are used to specify the rules concerning data in the table. It can be applied for single or multiple fields in an SQL table during the creation of the table or after creating using the ALTER TABLE command. The constraints are:

- **NOT NULL -** Restricts NULL value from being inserted into a column.
- **CHECK -** Verifies that all values in a field satisfy a condition.
- **DEFAULT -** Automatically assigns a default value if no value has been specified for the field.
- **UNIQUE -** Ensures unique values to be inserted into the field.
- **INDEX -** Indexes a field providing faster retrieval of records.
- **PRIMARY KEY -** Uniquely identifies each record in a table.
- **FOREIGN KEY -** Ensures referential integrity for a record in another table.

8. What is a Primary Key?

The PRIMARY KEY constraint uniquely identifies each row in a table. It must contain UNIQUE values and has an implicit NOT NULL constraint.
A table in SQL is strictly restricted to have one and only one primary key, which is comprised of single or multiple fields (columns).

```
CREATE TABLE Students (   /* Create table with a single field as primary key */
  ID INT NOT NULL
  Name VARCHAR(255)
  PRIMARY KEY (ID)
);


CREATE TABLE Students (   /* Create table with multiple fields as primary key */
  ID INT NOT NULL
  LastName VARCHAR(255)
  FirstName VARCHAR(255) NOT NULL,
  CONSTRAINT PK_Student
  PRIMARY KEY (ID, FirstName)
);


ALTER TABLE Students   /* Set a column as primary key */
```

```
ADD PRIMARY KEY (ID);
ALTER TABLE Students   /* Set multiple columns as primary key */
ADD CONSTRAINT PK_Student   /*Naming a Primary Key*/
PRIMARY KEY (ID, FirstName);
```

9. What is a UNIQUE constraint?

A UNIQUE constraint ensures that all values in a column are different. This provides uniqueness for the column(s) and helps identify each row uniquely. Unlike primary key, there can be multiple unique constraints defined per table. The code syntax for UNIQUE is quite similar to that of PRIMARY KEY and can be used interchangeably.

```
CREATE TABLE Students (   /* Create table with a single field as unique */
   ID INT NOT NULL UNIQUE
   Name VARCHAR(255)
);

CREATE TABLE Students (   /* Create table with multiple fields as unique */
   ID INT NOT NULL
   LastName VARCHAR(255)
   FirstName VARCHAR(255) NOT NULL
   CONSTRAINT PK_Student
   UNIQUE (ID, FirstName)
);

ALTER TABLE Students   /* Set a column as unique */
ADD UNIQUE (ID);
ALTER TABLE Students   /* Set multiple columns as unique */
ADD CONSTRAINT PK_Student   /* Naming a unique constraint */
UNIQUE (ID, FirstName);
```

10. What is a Foreign Key?

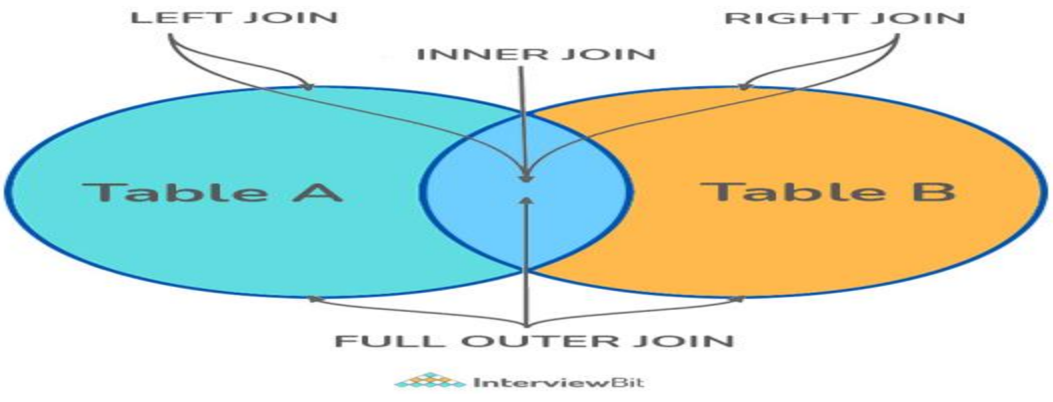
A FOREIGN KEY comprises of single or collection of fields in a table that essentially refers to the PRIMARY KEY in another table. Foreign key constraint ensures referential integrity in the relation between two tables.

The table with the foreign key constraint is labeled as the child table, and the table containing the candidate key is labeled as the referenced or parent table.

```
CREATE TABLE Students (   /* Create table with foreign key - Way 1 */
  ID INT NOT NULL
  Name VARCHAR(255)
  LibraryID INT
  PRIMARY KEY (ID)
  FOREIGN KEY (Library_ID) REFERENCES Library(LibraryID)
);

CREATE TABLE Students (   /* Create table with foreign key - Way 2 */
  ID INT NOT NULL PRIMARY KEY
  Name VARCHAR(255)
  LibraryID INT FOREIGN KEY (Library_ID) REFERENCES Library(LibraryID)
);

ALTER TABLE Students   /* Add a new foreign key */
ADD FOREIGN KEY (LibraryID)
REFERENCES Library (LibraryID);
```

11. What is a Join? List its different types.

The SQL Join clause is used to combine records (rows) from two or more tables in a SQL database based on a related column between the two.



There are four different types of JOINs in SQL:

- **(INNER) JOIN:** Retrieves records that have matching values in both tables involved in the join. This is the widely used join for queries.

```
SELECT *
FROM Table_A
JOIN Table_B;
SELECT *
FROM Table_A
INNER JOIN Table_B;
```

- **LEFT (OUTER) JOIN:** Retrieves all the records/rows from the left and the matched records/rows from the right table.

```
SELECT *
FROM Table_A A
LEFT JOIN Table_B B
ON A.col = B.col;
```

- **RIGHT (OUTER) JOIN:** Retrieves all the records/rows from the right and the matched records/rows from the left table.

```
SELECT *
FROM Table_A A
RIGHT JOIN Table_B B
ON A.col = B.col;
```

- **FULL (OUTER) JOIN:** Retrieves all the records where there is a match in either the left or right table.

```
SELECT *
FROM Table_A A
FULL JOIN Table_B B
ON A.col = B.col;
```

12. What is a Self-Join?

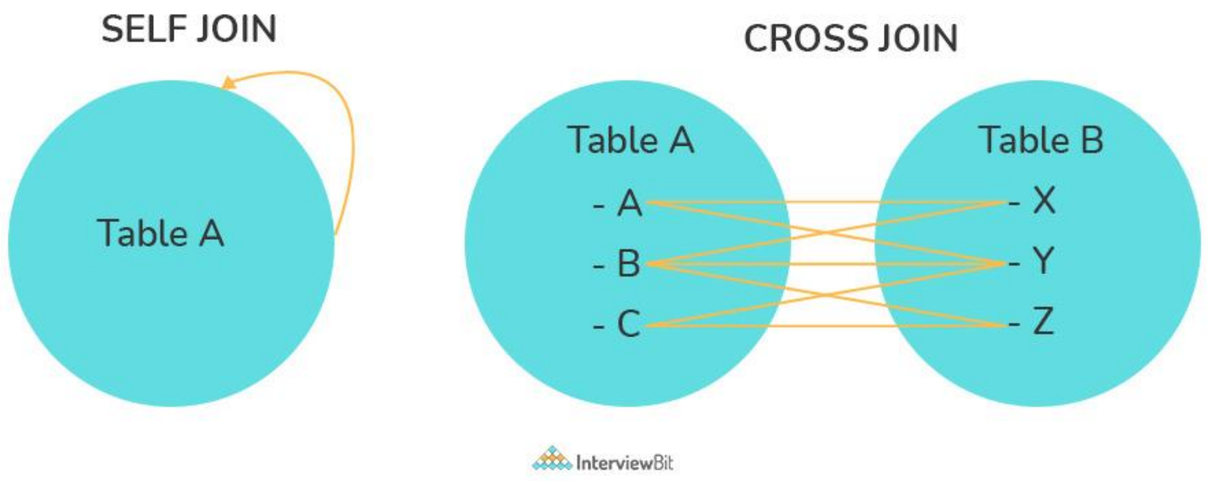
A **self JOIN** is a case of regular join where a table is joined to itself based on some relation between its own column(s). Self-join uses the INNER JOIN or LEFT JOIN clause and a table alias is used to assign different names to the table within the query.

```
SELECT A.emp_id AS "Emp_ID",A.emp_name AS "Employee",
B.emp_id AS "Sup_ID",B.emp_name AS "Supervisor"
FROM employee A, employee B
WHERE A.emp_sup = B.emp_id;
```

13. What is a Cross-Join?

Cross join can be defined as a cartesian product of the two tables included in the join. The table after join contains the same number of rows as in the cross-product of the number of rows in the two tables. If a WHERE clause is used in cross join then the query will work like an INNER JOIN.

```
SELECT stu.name, sub.subject
FROM students AS stu
CROSS JOIN subjects AS sub;
```



14. What is an Index? Explain its different types.

A database index is a data structure that provides a quick lookup of data in a column or columns of a table. It enhances the speed of operations accessing data from a database table at the cost of additional writes and memory to maintain the index data structure.

```
CREATE INDEX index_name   /* Create Index */
ON table_name (column_1, column_2);
DROP INDEX index_name;   /* Drop Index */
```

There are different types of indexes that can be created for different purposes:

- **Unique and Non-Unique Index:**

Unique indexes are indexes that help maintain data integrity by ensuring that no two rows of data in a table have identical key values. Once a unique index has been defined for a table, uniqueness is enforced whenever keys are added or changed within the index.

```
CREATE UNIQUE INDEX myIndex
ON students (enroll_no);
```

Non-unique indexes, on the other hand, are not used to enforce constraints on the tables with which they are associated. Instead, non-unique indexes are used solely to improve query performance by maintaining a sorted order of data values that are used frequently.

- **Clustered and Non-Clustered Index:**

Clustered indexes are indexes whose order of the rows in the database corresponds to the order of the rows in the index. This is why only one clustered index can exist in a given table, whereas, multiple non-clustered indexes can exist in the table.

The only difference between clustered and non-clustered indexes is that the database manager attempts to keep the data in the database in the same order as the corresponding keys appear in the clustered index.

Clustering indexes can improve the performance of most query operations because they provide a linear-access path to data stored in the database.

15. What is the difference between Clustered and Non-clustered index?

As explained above, the differences can be broken down into three small factors -

- Clustered index modifies the way records are stored in a database based on the indexed column. A non-clustered index creates a separate entity within the table which references the original table.
- Clustered index is used for easy and speedy retrieval of data from the database, whereas, fetching records from the non-clustered index is relatively slower.
- In SQL, a table can have a single clustered index whereas it can have multiple non-clustered indexes.

16. What is Data Integrity?

Data Integrity is the assurance of accuracy and consistency of data over its entire life-cycle and is a critical aspect of the design, implementation, and usage of any system which stores, processes, or retrieves data. It also defines integrity constraints to enforce business rules on the data when it is entered into an application or a database.

17. What is a Query?

A query is a request for data or information from a database table or combination of tables. A database query can be either a select query or an action query.

```
SELECT fname, lname    /* select query */
FROM myDb.students
WHERE student_id = 1;
UPDATE myDB.students    /* action query */
SET fname = 'Captain', lname = 'America'
WHERE student_id = 1;
```

18. What is a Subquery? What are its types?

A subquery is a query within another query, also known as a **nested query** or **inner query**. It is used to restrict or enhance the data to be queried by the main query, thus restricting or enhancing the output of the main query respectively. For example, here we fetch the contact information for students who have enrolled for the maths subject:

```
SELECT name, email, mob, address
FROM myDb.contacts
WHERE roll_no IN (
 SELECT roll_no
 FROM myDb.students
 WHERE subject = 'Maths');
```

There are two types of subquery **- Correlated** and **Non-Correlated**.

- A **correlated** subquery cannot be considered as an independent query, but it can refer to the column in a table listed in the FROM of the main query.
- A **non-correlated** subquery can be considered as an independent query and the output of the subquery is substituted in the main query.

19. What is the SELECT statement?

SELECT operator in SQL is used to select data from a database. The data returned is stored in a result table, called the result-set.

```
SELECT * FROM myDB.students;
```

20. What are some common clauses used with SELECT query in SQL?

Some common SQL clauses used in conjuction with a SELECT query are as follows:

- **WHERE** clause in SQL is used to filter records that are necessary, based on specific conditions.
- **ORDER BY** clause in SQL is used to sort the records based on some field(s) in ascending (**ASC**) or descending order (**DESC)**.

```
SELECT *
```

```
FROM myDB.students

WHERE graduation_year = 2019

ORDER BY studentID DESC;
```

- **GROUP BY** clause in SQL is used to group records with identical data and can be used in conjunction with some aggregation functions to produce summarized results from the database.
- **HAVING** clause in SQL is used to filter records in combination with the GROUP BY clause. It is different from WHERE, since the WHERE clause cannot filter aggregated records.

```
SELECT COUNT(studentId), country

FROM myDB.students

WHERE country != "INDIA"

GROUP BY country

HAVING COUNT(studentID) > 5;
```

21. What are UNION, MINUS and INTERSECT commands?

The **UNION** operator combines and returns the result-set retrieved by two or more SELECT statements.
The **MINUS** operator in SQL is used to remove duplicates from the result-set obtained by the second SELECT query from the result-set obtained by the first SELECT query and then return the filtered results from the first.
The **INTERSECT** clause in SQL combines the result-set fetched by the two SELECT statements where records from one match the other and then returns this intersection of result-sets.

Certain conditions need to be met before executing either of the above statements in SQL -

- Each SELECT statement within the clause must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement should necessarily have the same order

```
SELECT name FROM Students   /* Fetch the union of queries */

UNION

SELECT name FROM Contacts;

SELECT name FROM Students   /* Fetch the union of queries with duplicates*/

UNION ALL
```

```
SELECT name FROM Contacts;
SELECT name FROM Students   /* Fetch names from students */
MINUS     /* that aren't present in contacts */
SELECT name FROM Contacts;
SELECT name FROM Students   /* Fetch names from students */
INTERSECT    /* that are present in contacts as well */
SELECT name FROM Contacts;
```

22. What is Cursor? How to use a Cursor?

A database cursor is a control structure that allows for the traversal of records in a database. Cursors, in addition, facilitates processing after traversal, such as retrieval, addition, and deletion of database records. They can be viewed as a pointer to one row in a set of rows.

Working with SQL Cursor:

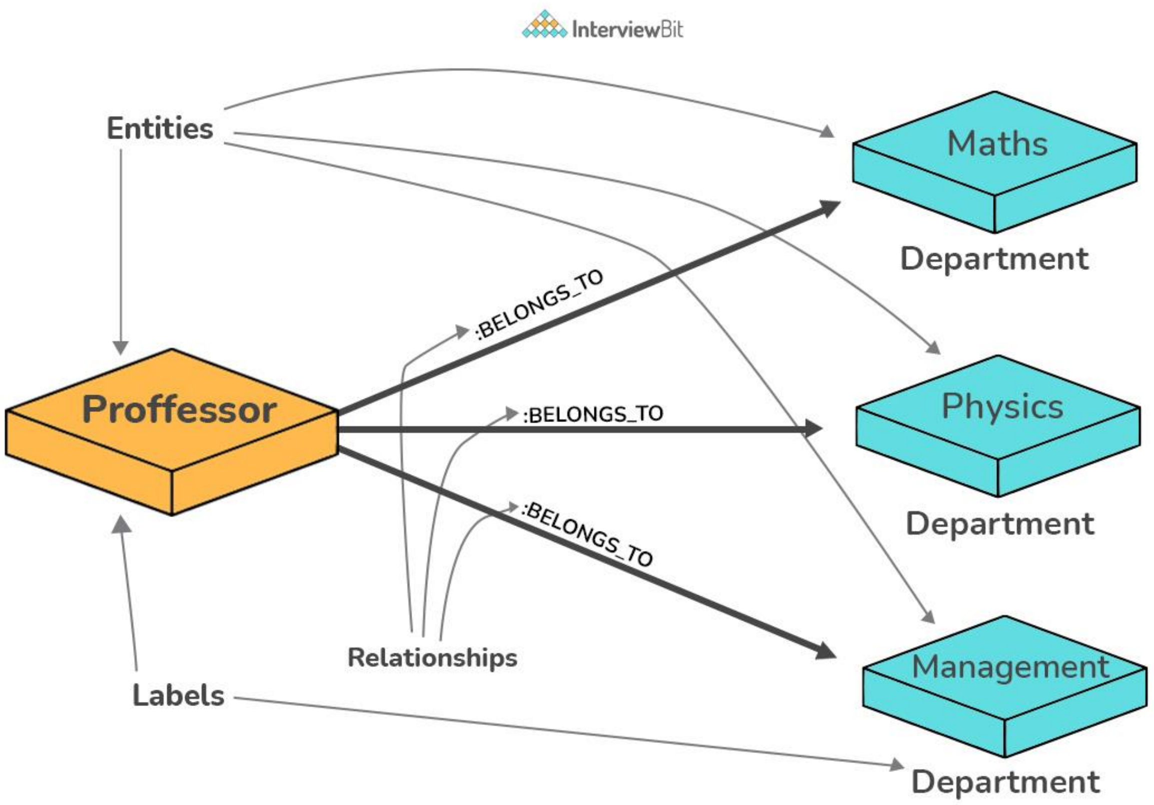
1. **DECLARE** a cursor after any variable declaration. The cursor declaration must always be associated with a SELECT Statement.
2. Open cursor to initialize the result set. The **OPEN** statement must be called before fetching rows from the result set.
3. **FETCH** statement to retrieve and move to the next row in the result set.
4. Call the **CLOSE** statement to deactivate the cursor.
5. Finally use the **DEALLOCATE** statement to delete the cursor definition and release the associated resources.

```
DECLARE @name VARCHAR(50)   /* Declare All Required Variables */
DECLARE db_cursor CURSOR FOR   /* Declare Cursor Name*/
SELECT name
FROM myDB.students
WHERE parent_name IN ('Sara', 'Ansh')
OPEN db_cursor   /* Open cursor and Fetch data into @name */
FETCH next
FROM db_cursor
INTO @name
CLOSE db_cursor   /* Close the cursor and deallocate the resources */
DEALLOCATE db_cursor
```

23. What are Entities and Relationships?

Entity: An entity can be a real-world object, either tangible or intangible, that can be easily identifiable. For example, in a college database, students, professors, workers, departments, and projects can be referred to as entities. Each entity has some associated properties that provide it an identity.

Relationships: Relations or links between entities that have something to do with each other. For example - The employee's table in a company's database can be associated with the salary table in the same database.



24. List the different types of relationships in SQL.

- **One-to-One -** This can be defined as the relationship between two tables where each record in one table is associated with the maximum of one record in the other table.
- **One-to-Many & Many-to-One -** This is the most commonly used relationship where a record in a table is associated with multiple records in the other table.
- **Many-to-Many -** This is used in cases when multiple instances on both sides are needed for defining a relationship.
- **Self-Referencing Relationships -** This is used when a table needs to define a relationship with itself.

25. What is an Alias in SQL?

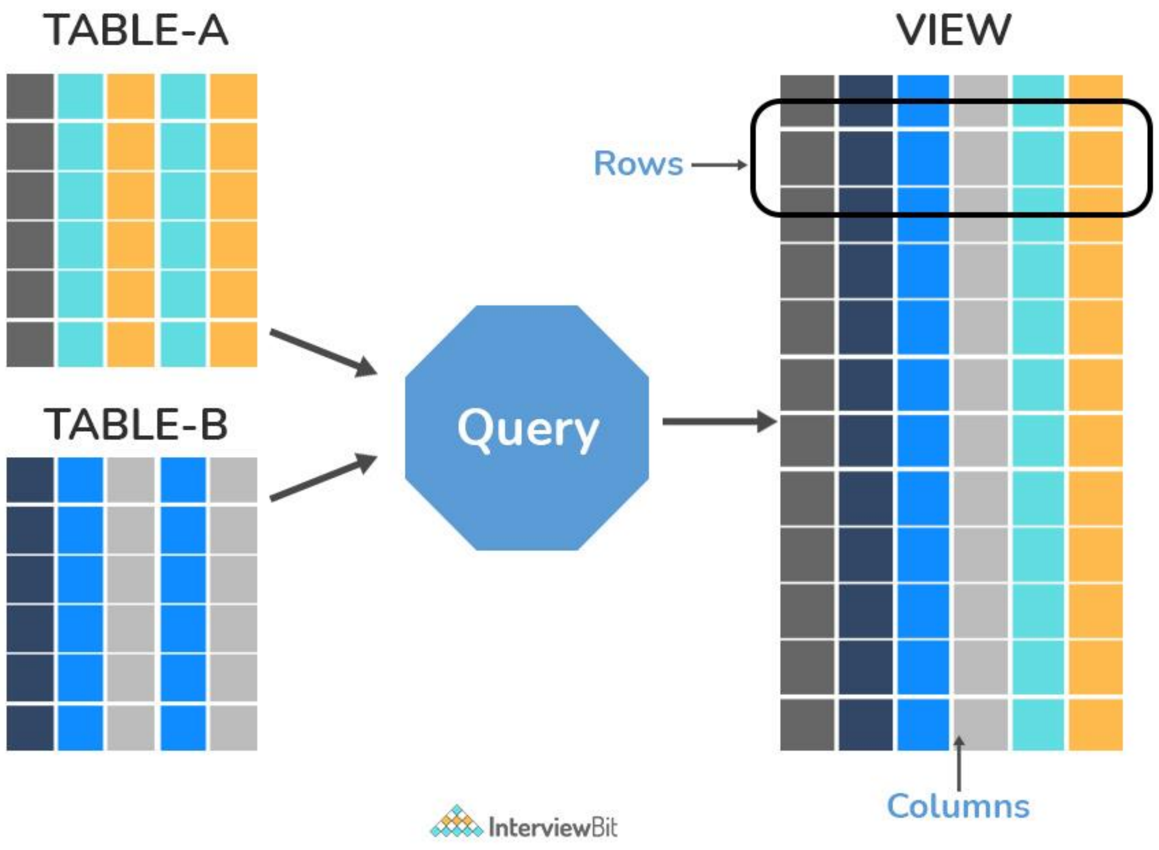
An alias is a feature of SQL that is supported by most, if not all, RDBMSs. It is a temporary name assigned to the table or table column for the purpose of a particular SQL query. In addition, aliasing can be employed as an obfuscation technique to secure the real names of database fields. A table alias is also called a correlation name.

An alias is represented explicitly by the AS keyword but in some cases, the same can be performed without it as well. Nevertheless, using the AS keyword is always a good practice.

```
SELECT A.emp_name AS "Employee"  /* Alias using AS keyword */
B.emp_name AS "Supervisor"
FROM employee A, employee B   /* Alias without AS keyword */
WHERE A.emp_sup = B.emp_id;
```

26. What is a View?

A view in SQL is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.



27. What is Normalization?

Normalization represents the way of organizing structured data in the database efficiently. It includes the creation of tables, establishing relationships between them, and defining rules for those relationships. Inconsistency and redundancy can be kept in check based on these rules, hence, adding flexibility to the database.

28. What is Denormalization?

Denormalization is the inverse process of normalization, where the normalized schema is converted into a schema that has redundant information. The performance is improved by using redundancy and keeping the redundant data consistent. The reason for performing denormalization is the overheads produced in the query processor by an over-normalized structure.

29. What are the various forms of Normalization?

Normal Forms are used to eliminate or reduce redundancy in database tables. The different forms are as follows:

- **First Normal Form:**
  A relation is in first normal form if every attribute in that relation is a **single-valued attribute**. If a relation contains a composite or multi-valued attribute, it violates the first normal form. Let's consider the following **students** table. Each student in the table, has a name, his/her address, and the books they issued from the public library -

Students Table

Student	**Address**	**Books Issued**	**Salutation**
Sara	Amanora Park Town 94	Until the Day I Die (Emily Carpenter), Inception (Christopher Nolan)	Ms.
Ansh	62nd Sector A-10	The Alchemist (Paulo Coelho), Inferno (Dan Brown)	Mr.
Sara	24th Street Park Avenue	Beautiful Bad (Annie Ward), Woman 99 (Greer Macallister)	Mrs.
Ansh	Windsor Street 777	Dracula (Bram Stoker)	Mr.

As we can observe, the Books Issued field has more than one value per record, and to convert it into 1NF, this has to be resolved into separate individual records for each book issued. Check the following table in 1NF form -

Students Table (1st Normal Form)

Student	Address	Books Issued	Salutation
Sara	Amanora Park Town 94	Until the Day I Die (Emily Carpenter)	Ms.
Sara	Amanora Park Town 94	Inception (Christopher Nolan)	Ms.
Ansh	62nd Sector A-10	The Alchemist (Paulo Coelho)	Mr.
Ansh	62nd Sector A-10	Inferno (Dan Brown)	Mr.
Sara	24th Street Park Avenue	Beautiful Bad (Annie Ward)	Mrs.
Sara	24th Street Park Avenue	Woman 99 (Greer Macallister)	Mrs.
Ansh	Windsor Street 777	Dracula (Bram Stoker)	Mr.

- **Second Normal Form:**

A relation is in second normal form if it satisfies the conditions for the first normal form and does not contain any partial dependency. A relation in 2NF has **no partial dependency**, i.e., it has no non-prime attribute that depends on any proper subset of any candidate key of the table. Often, specifying a single column Primary Key is the solution to the problem. Examples -

Example 1 - Consider the above example. As we can observe, the Students Table in the 1NF form has a candidate key in the form of [Student, Address] that can uniquely identify all records in the table. The field Books Issued (non-prime attribute) depends partially on the Student field. Hence, the table is not in 2NF. To convert it into the 2nd Normal Form, we will partition the tables into two while specifying a new **Primary Key** attribute to identify the individual records in the Students table. The **Foreign Key** constraint will be set on the other table to ensure referential integrity.

Students Table (2nd Normal Form)

Student_ID	Student	Address	Salutation
1	Sara	Amanora Park Town 94	Ms.
2	Ansh	62nd Sector A-10	Mr.
3	Sara	24th Street Park Avenue	Mrs.

Student_ID	Student	Address	Salutation
4	Ansh	Windsor Street 777	Mr.

Books Table (2nd Normal Form)

Student_ID	Book Issued
1	Until the Day I Die (Emily Carpenter)
1	Inception (Christopher Nolan)
2	The Alchemist (Paulo Coelho)
2	Inferno (Dan Brown)
3	Beautiful Bad (Annie Ward)
3	Woman 99 (Greer Macallister)
4	Dracula (Bram Stoker)

Example 2 - Consider the following dependencies in relation to R(W,X,Y,Z)

WX -> Y [W **and** X together determine Y]

XY -> Z [X **and** Y together determine Z]

Here, WX is the only candidate key and there is no partial dependency, i.e., any proper subset of WX doesn't determine any non-prime attribute in the relation.

- **Third Normal Form**

A relation is said to be in the third normal form, if it satisfies the conditions for the second normal form and there is **no transitive dependency** between the non-prime attributes, i.e., all non-prime attributes are determined only by the candidate keys of the relation and not by any other non-prime attribute.

Example 1 - Consider the Students Table in the above example. As we can observe, the Students Table in the 2NF form has a single candidate key Student_ID (primary key) that can uniquely identify all records in the table. The field Salutation (non-prime attribute), however, depends on the Student Field rather than the candidate key. Hence, the table is not in 3NF. To convert it into the 3rd Normal Form, we will once again partition the tables into two while specifying a new **Foreign Key** constraint to identify the salutations for individual records in the Students table. The **Primary**

Key constraint for the same will be set on the Salutations table to identify each record uniquely.

Students Table (3rd Normal Form)

Student_ID	**Student**	**Address**	**Salutation_ID**
1	Sara	Amanora Park Town 94	1
2	Ansh	62nd Sector A-10	2
3	Sara	24th Street Park Avenue	3
4	Ansh	Windsor Street 777	1

Books Table (3rd Normal Form)

Student_ID	**Book Issued**
1	Until the Day I Die (Emily Carpenter)
1	Inception (Christopher Nolan)
2	The Alchemist (Paulo Coelho)
2	Inferno (Dan Brown)
3	Beautiful Bad (Annie Ward)
3	Woman 99 (Greer Macallister)
4	Dracula (Bram Stoker)

Salutations Table (3rd Normal Form)

Salutation_ID	Salutation
1	Ms.
2	Mr.
3	Mrs.

Example 2 - Consider the following dependencies in relation to R(P,Q,R,S,T)

```
P -> QR     [P together determine C]
RS -> T     [B and C together determine D]
Q -> S
T -> P
```

For the above relation to exist in 3NF, all possible candidate keys in the above relation should be {P, RS, QR, T}.

- **Boyce-Codd Normal Form**

A relation is in Boyce-Codd Normal Form if satisfies the conditions for third normal form and for every functional dependency, Left-Hand-Side is super key. In other words, a relation in BCNF has non-trivial functional dependencies in form X –> Y, such that X is always a super key. For example - In the above example, Student_ID serves as the sole unique identifier for the Students Table and Salutation_ID for the Salutations Table, thus these tables exist in BCNF. The same cannot be said for the Books Table and there can be several books with common Book Names and the same Student_ID.

30. What are the TRUNCATE, DELETE and DROP statements?

DELETE statement is used to delete rows from a table.

```
DELETE FROM Candidates
WHERE CandidateId > 1000;
```

TRUNCATE command is used to delete all the rows from the table and free the space containing the table.

```
TRUNCATE TABLE Candidates;
```

DROP command is used to remove an object from the database. If you drop a table, all the rows in the table are deleted and the table structure is removed from the database.

```
DROP TABLE Candidates;
```

31. What is the difference between DROP and TRUNCATE statements?

If a table is dropped, all things associated with the tables are dropped as well. This includes - the relationships defined on the table with other tables, the integrity checks and constraints, access privileges and other grants that the table has. To create and use the table again in its original form, all these relations, checks, constraints, privileges and relationships need to be redefined. However, if a table is truncated, none of the above problems exist and the table retains its original structure.

32. What is the difference between DELETE and TRUNCATE statements?

The **TRUNCATE** command is used to delete all the rows from the table and free the space containing the table.
The **DELETE** command deletes only the rows from the table based on the condition given in the where clause or deletes all the rows from the table if no condition is specified. But it does not free the space containing the table.

33. What are Aggregate and Scalar functions?

An aggregate function performs operations on a collection of values to return a single scalar value. Aggregate functions are often used with the GROUP BY and HAVING clauses of the SELECT statement. Following are the widely used SQL aggregate functions:

- **AVG() -** Calculates the mean of a collection of values.
- **COUNT() -** Counts the total number of records in a specific table or view.
- **MIN() -** Calculates the minimum of a collection of values.
- **MAX() -** Calculates the maximum of a collection of values.
- **SUM() -** Calculates the sum of a collection of values.
- **FIRST() -** Fetches the first element in a collection of values.
- **LAST() -** Fetches the last element in a collection of values.

Note: All aggregate functions described above ignore NULL values except for the COUNT function.

A scalar function returns a single value based on the input value. Following are the widely used SQL scalar functions:

- **LEN() -** Calculates the total length of the given field (column).
- **UCASE() -** Converts a collection of string values to uppercase characters.
- **LCASE() -** Converts a collection of string values to lowercase characters.
- **MID() -** Extracts substrings from a collection of string values in a table.
- **CONCAT() -** Concatenates two or more strings.
- **RAND() -** Generates a random collection of numbers of a given length.

- **ROUND()** - Calculates the round-off integer value for a numeric field (or decimal point values).
- **NOW()** - Returns the current date & time.
- **FORMAT()** - Sets the format to display a collection of values.

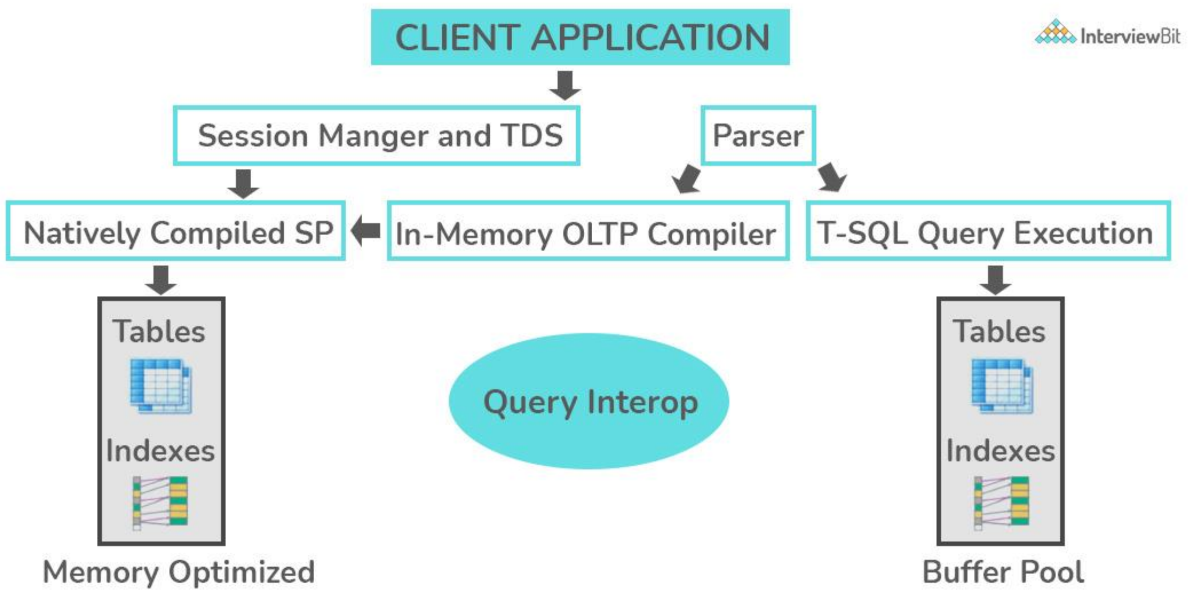
34. What is User-defined function? What are its various types?

The user-defined functions in SQL are like functions in any other programming language that accept parameters, perform complex calculations, and return a value. They are written to use the logic repetitively whenever required. There are two types of SQL user-defined functions:

- Scalar Function: As explained earlier, user-defined scalar functions return a single scalar value.
- Table-Valued Functions: User-defined table-valued functions return a table as output.
    - **Inline:** returns a table data type based on a single SELECT statement.
    - **Multi-statement:** returns a tabular result-set but, unlike inline, multiple SELECT statements can be used inside the function body.

35. What is OLTP?

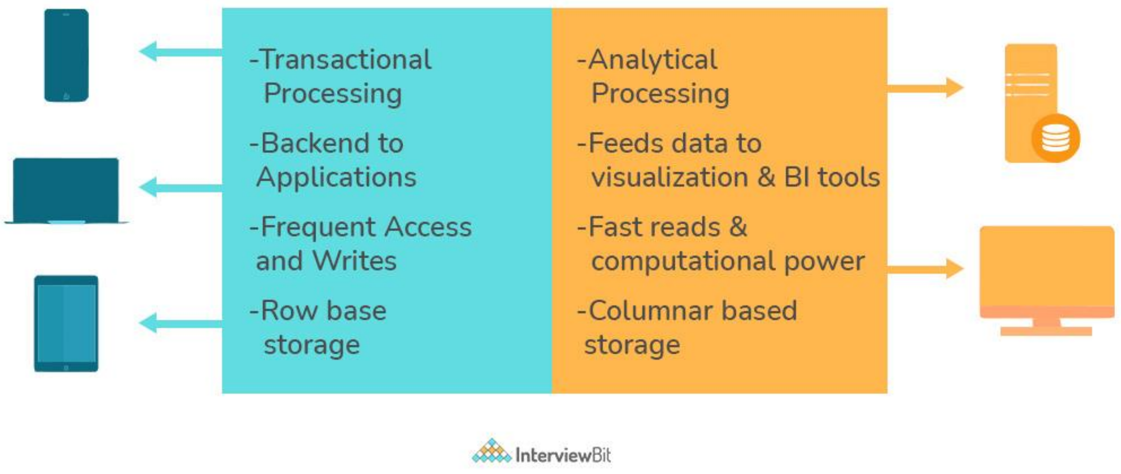
OLTP stands for Online Transaction Processing, is a class of software applications capable of supporting transaction-oriented programs. An essential attribute of an OLTP system is its ability to maintain concurrency. To avoid single points of failure, OLTP systems are often decentralized. These systems are usually designed for a large number of users who conduct short transactions. Database queries are usually simple, require sub-second response times, and return relatively few records. Here is an insight into the working of an OLTP system [Note - The figure is not important for interviews] -



36. What are the differences between OLTP and OLAP?

OLTP stands for **Online Transaction Processing**, is a class of software applications capable of supporting transaction-oriented programs. An important attribute of an OLTP system is its ability to maintain concurrency. OLTP systems often follow a decentralized architecture to avoid single points of failure. These systems are generally designed for a large audience of end-users who conduct short transactions. Queries involved in such databases are generally simple, need fast response times, and return relatively few records. A number of transactions per second acts as an effective measure for such systems.

OLAP stands for **Online Analytical Processing**, a class of software programs that are characterized by the relatively low frequency of online transactions. Queries are often too complex and involve a bunch of aggregations. For OLAP systems, the effectiveness measure relies highly on response time. Such systems are widely used for data mining or maintaining aggregated, historical data, usually in multi-dimensional schemas.


-Transactional Processing
-Backend to Applications
-Frequent Access and Writes
-Row base storage
-Analytical Processing
-Feeds data to visualization & BI tools
-Fast reads & computational power
-Columnar based storage
InterviewBit

37. What is Collation? What are the different types of Collation Sensitivity?

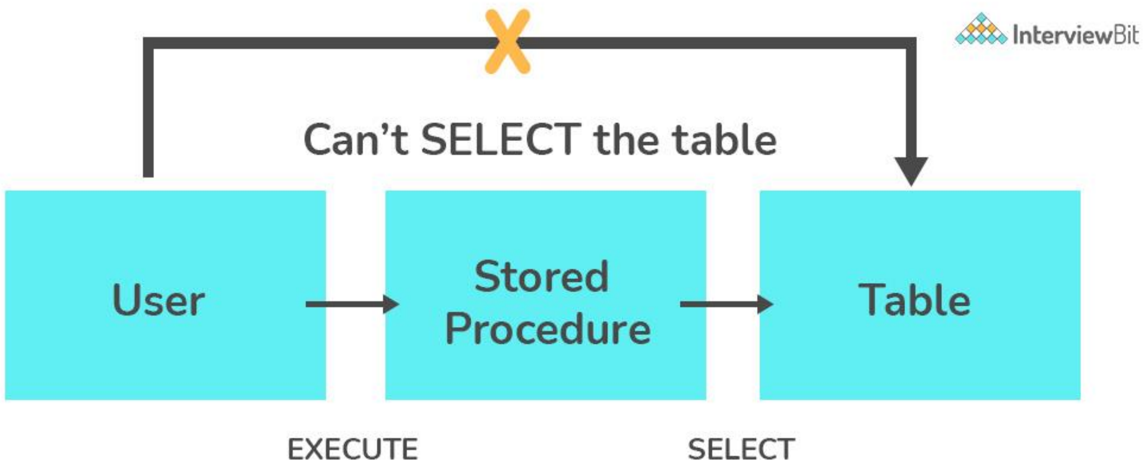
Collation refers to a set of rules that determine how data is sorted and compared. Rules defining the correct character sequence are used to sort the character data. It incorporates options for specifying case sensitivity, accent marks, kana character types, and character width. Below are the different types of collation sensitivity:

- **Case sensitivity: A** and **a** are treated differently.
- **Accent sensitivity: a** and **á** are treated differently.
- **Kana sensitivity:** Japanese kana characters Hiragana and Katakana are treated differently.
- **Width sensitivity:** Same character represented in single-byte (half-width) and double-byte (full-width) are treated differently.

38. What is a Stored Procedure?

A stored procedure is a subroutine available to applications that access a relational database management system (RDBMS). Such procedures are stored in the database data dictionary. The sole disadvantage of stored procedure is that it can be executed nowhere except in the database and occupies more memory in the database server. It also provides a sense of security and functionality as users who can't access the data directly can be granted access via stored procedures.

```
DELIMITER $$
CREATE PROCEDURE FetchAllStudents()
BEGIN
SELECT *  FROM myDB.students;
END $$
DELIMITER ;
```



39. What is a Recursive Stored Procedure?

A stored procedure that calls itself until a boundary condition is reached, is called a recursive stored procedure. This recursive function helps the programmers to deploy the same set of code several times as and when required. Some SQL programming languages limit the recursion depth to prevent an infinite loop of procedure calls from causing a stack overflow, which slows down the system and may lead to system crashes.

```
DELIMITER $$     /* Set a new delimiter => $$ */
CREATE PROCEDURE calctotal( /* Create the procedure */
   IN number INT,   /* Set Input and Ouput variables */
   OUT total INT
) BEGIN
```

```
DECLARE score INT DEFAULT NULL;   /* Set the default value => "score" */
SELECT awards FROM achievements   /* Update "score" via SELECT query */
WHERE id = number INTO score;
IF score IS NULL THEN SET total = 0;   /* Termination condition */
ELSE
CALL calctotal(number+1);   /* Recursive call */
SET total = total + score;   /* Action after recursion */
END IF;
END $$     /* End of procedure */
DELIMITER ;     /* Reset the delimiter */
```

40. How to create empty tables with the same structure as another table?

Creating empty tables with the same structure can be done smartly by fetching the records of one table into a new table using the INTO operator while fixing a WHERE clause to be false for all records. Hence, SQL prepares the new table with a duplicate structure to accept the fetched records but since no records get fetched due to the WHERE clause in action, nothing is inserted into the new table.

```
SELECT * INTO Students_copy
FROM Students WHERE 1 = 2;
```

41. What is Pattern Matching in SQL?

SQL pattern matching provides for pattern search in data if you have no clue as to what that word should be. This kind of SQL query uses wildcards to match a string pattern, rather than writing the exact word. The LIKE operator is used in conjunction with **SQL Wildcards** to fetch the required information.

- **Using the % wildcard to perform a simple search**

The % wildcard matches zero or more characters of any type and can be used to define wildcards both before and after the pattern. Search a student in your database with first name beginning with the letter K:

```
SELECT *
FROM students
WHERE first_name LIKE 'K%'
```

- **Omitting the patterns using the NOT keyword**

Use the NOT keyword to select records that don't match the pattern. This query returns all students whose first name does not begin with K.

```
SELECT *
```

```
FROM students
WHERE first_name NOT LIKE 'K%'
```

- **Matching a pattern anywhere using the % wildcard twice**

Search for a student in the database where he/she has a K in his/her first name.

```
SELECT *
FROM students
WHERE first_name LIKE '%Q%'
```

- **Using the _ wildcard to match pattern at a specific position**

The _ wildcard matches exactly one character of any type. It can be used in conjunction with % wildcard. This query fetches all students with letter K at the third position in their first name.

```
SELECT *
FROM students
WHERE first_name LIKE '__K%'
```

- **Matching patterns for a specific length**

The _ wildcard plays an important role as a limitation when it matches exactly one character. It limits the length and position of the matched results. For example -

```
SELECT *   /* Matches first names with three or more letters */
FROM students
WHERE first_name LIKE '___%'


SELECT *   /* Matches first names with exactly four characters */
FROM students
WHERE first_name LIKE '____'
```

Q #1) What is SQL?

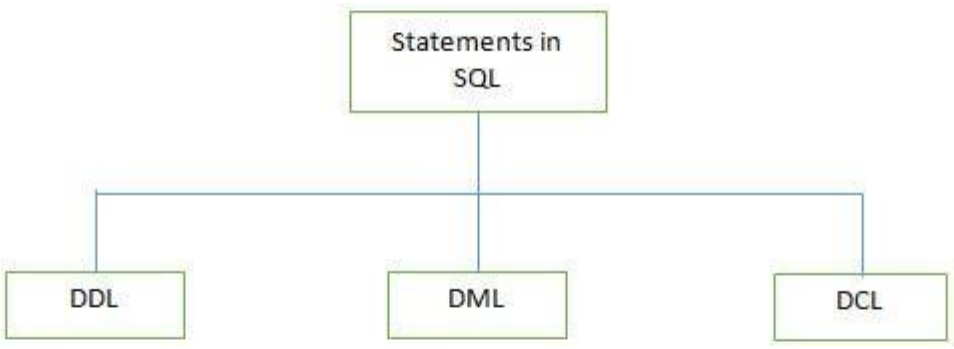
Answer: Structured Query Language SQL is a database tool that is used to create and access the database to support software applications.

Q #2) What are tables in SQL?

Answer: The table is a collection of record and its information at a single view.

Q #3) What are the different types of statements supported by SQL?

Answer:



There are 3 types of SQL statements:

a) DDL (Data Definition Language): It is used to define the database structure such as tables. It includes three statements such as CREATE, ALTER, and DROP.

Also read =>> MySQL Create Table Tutorial

Some of the DDL Commands are listed below:

CREATE: It is used for creating the table.

CREATE TABLE table_name

column_name1 data_type(**size**),

column_name2 data_type(**size**),

column_name3 data_type(**size**),

ALTER: The ALTER table is used for modifying the existing table object in the database.

ALTER TABLE table_name

ADD column_name datatype

OR

ALTER TABLE table_name

DROP COLUMN column_name

b) DML (Data Manipulation Language): These statements are used to manipulate the data in records. Commonly used DML statements are INSERT, UPDATE, and DELETE.

The SELECT statement is used as a partial DML statement, used to select all or relevant records in the table.

c) DCL (Data Control Language): These statements are used to set privileges such as GRANT and REVOKE database access permission to the specific user**.**

Q #4) How do we use the DISTINCT statement? What is its use?

Answer: The DISTINCT statement is used with the SELECT statement. If the record contains duplicate values then the DISTINCT statement is used to select different values among duplicate records.

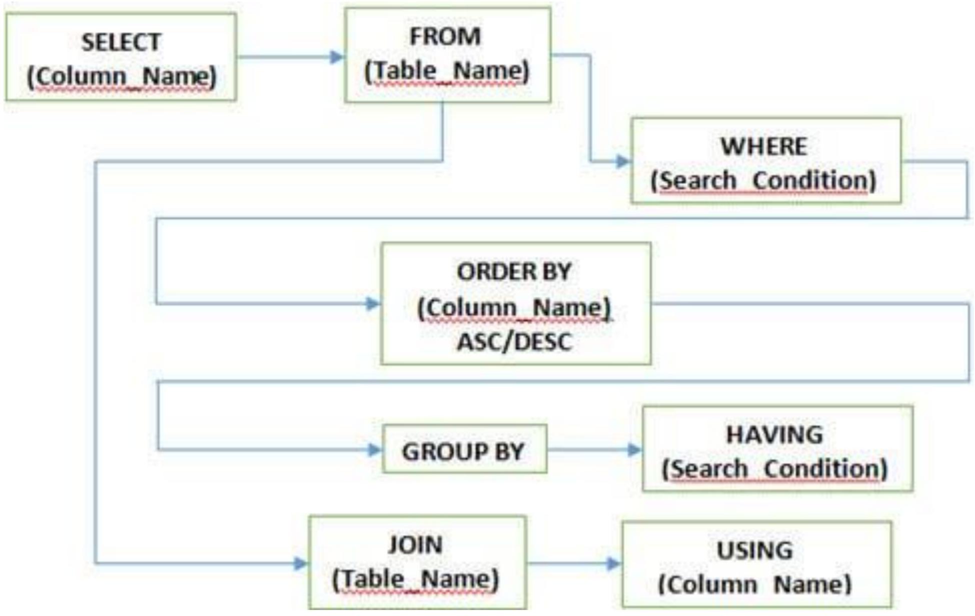
Syntax:

SELECT DISTINCT column_name(s)

FROM table_name;

Q #5) What are the different Clauses used in SQL?

Answer:



WHERE Clause: This clause is used to define the condition, extract and display only those records which fulfill the given condition.

Syntax:

SELECT column_name(s)

FROM table_name

WHERE condition;

GROUP BY Clause: It is used with SELECT statement to group the result of the executed query using the value specified in it. It matches the value with the column name in tables and groups the end result accordingly.

Syntax:

SELECT column_name(s)

FROM table_name

GROUP BY column_name;

HAVING clause: This clause is used in association with the GROUP BY clause. It is applied to each group of results or the entire result as a single group. It is much similar as WHERE clause but the only difference is you cannot use it without GROUP BY clause

Syntax:

SELECT column_name(s)

FROM table_name

GROUP BY column_name

HAVING condition;

ORDER BY clause: This clause is used to define the order of the query output either in ascending (ASC) or in descending (DESC). Ascending (ASC) is set as the default one but descending (DESC) is set explicitly.

Syntax:

SELECT column_name(s)

FROM table_name

WHERE condition

ORDER BY column_name **ASC|DESC**;

USING clause: USING clause comes in use while working with SQL JOIN. It is used to check equality based on columns when tables are joined. It can be used instead of the ON clause in JOIN.

Syntax:

SELECT column_name(s)

FROM table_name

JOIN table_name

USING (column_name);

Q #6) Why do we use SQL constraints? Which constraints we can use while creating a database in SQL?

Answer: Constraints are used to set the rules for all records in the table. If any constraints get violated then it can abort the action that caused it.

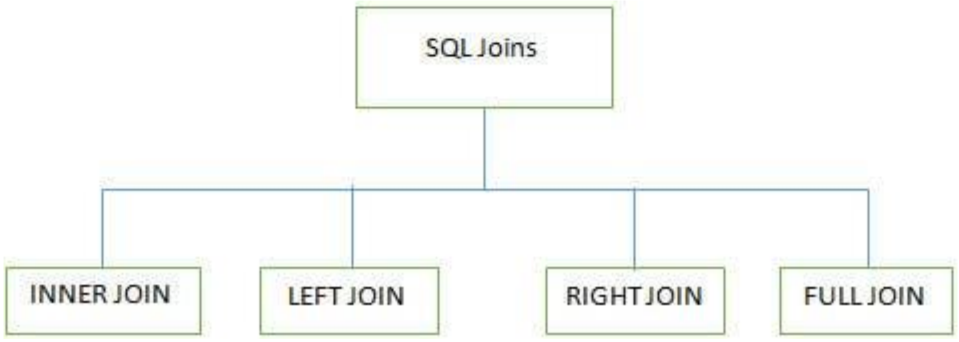
Constraints are defined while creating the database itself with the CREATE TABLE statement or even after the table is created once with the ALTER TABLE statement.

There are 5 major constraints are used in SQL, such as

- **NOT NULL:** That indicates that the column must have some value and cannot be left NULL.
- **UNIQUE:** This constraint is used to ensure that each row and column has a unique value and no value is being repeated in any other row or column.
- **PRIMARY KEY:** This constraint is used in association with NOT NULL and UNIQUE constraints such as on one or the combination of more than one column to identify the particular record with a unique identity.
- **FOREIGN KEY:** It is used to ensure the referential integrity of data in the table. It matches the value in one table with another using the PRIMARY KEY.
- **CHECK:** It ensures whether the value in columns fulfills the specified condition.

Q #7) What are different JOINS used in SQL?

Answer:



4 major types of Joins are used while working on multiple tables in SQL databases:

INNER JOIN: It is also known as SIMPLE JOIN which returns all rows from BOTH tables when it has at least one matching column.

Syntax:

SELECT column_name(s)

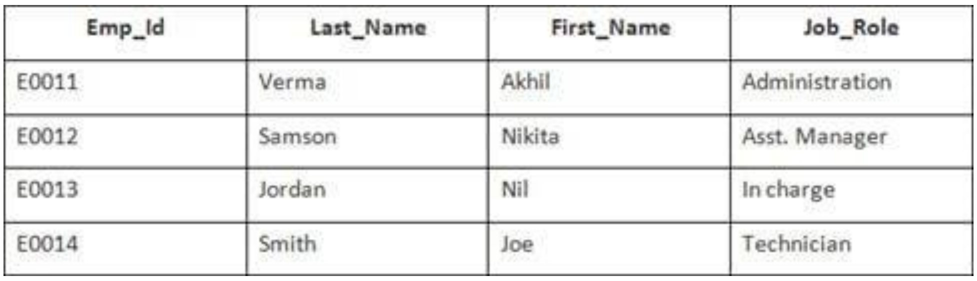
FROM table_name1&nbsp;

INNER JOIN table_name2

ON column_name1=column_name2;

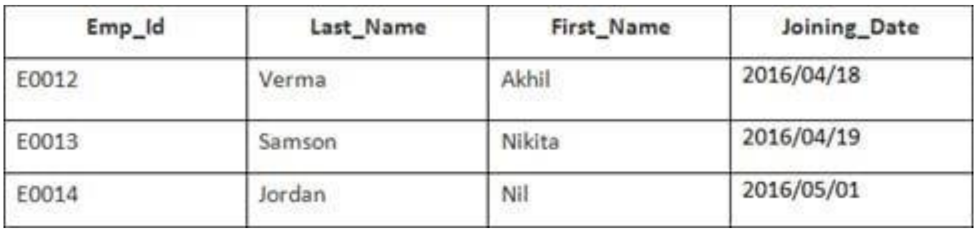
<u>**For Example,**</u>

In this example, we have a table **Employee** with the following data:



Emp_Id	Last_Name	First_Name	Job_Role
E0011	Verma	Akhil	Administration
E0012	Samson	Nikita	Asst. Manager
E0013	Jordan	Nil	In charge
E0014	Smith	Joe	Technician

The second table's name is **Joining.**



Emp_Id	Last_Name	First_Name	Joining_Date
E0012	Verma	Akhil	2016/04/18
E0013	Samson	Nikita	2016/04/19
E0014	Jordan	Nil	2016/05/01

Enter the following SQL statement:

SELECT Employee.Emp_id, Joining.Joining_Date

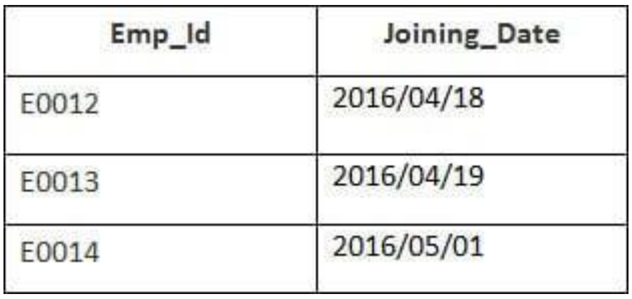
FROM Employee

INNER JOIN Joining

ON Employee.Emp_id = Joining.Emp_id

ORDER BY Employee.Emp_id;

There will be 4 records selected. **Results are:**



Emp_Id	Joining_Date
E0012	2016/04/18
E0013	2016/04/19
E0014	2016/05/01

Employee and **Orders** tables have a matching customer_id value.

LEFT JOIN (LEFT OUTER JOIN): This join returns all rows from the LEFT table and its matched rows from a RIGHT table**.**

Syntax:

SELECT column_name(s)

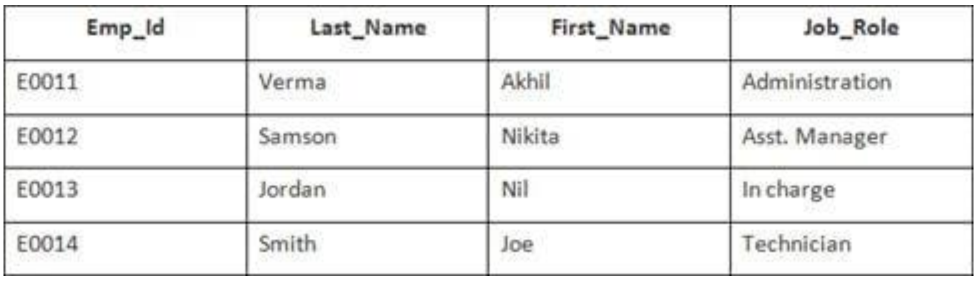
FROM table_name1

LEFT JOIN table_name2

ON column_name1=column_name2;

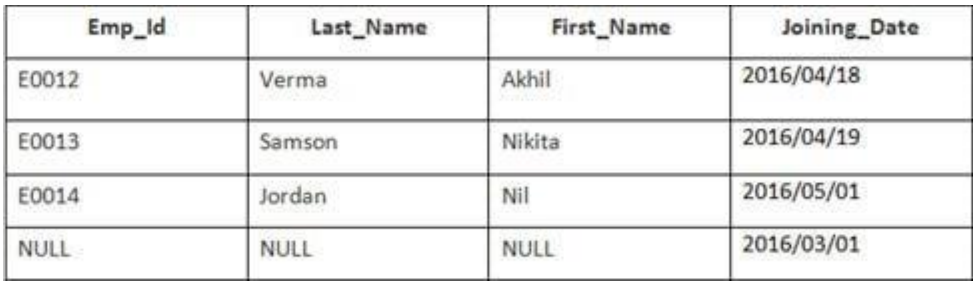
<u>For Example,</u>

In this example, we have a table **Employee** with the following data:



Emp_Id	Last_Name	First_Name	Job_Role
E0011	Verma	Akhil	Administration
E0012	Samson	Nikita	Asst. Manager
E0013	Jordan	Nil	In charge
E0014	Smith	Joe	Technician

The second table's name is **Joining.**



Emp_Id	Last_Name	First_Name	Joining_Date
E0012	Verma	Akhil	2016/04/18
E0013	Samson	Nikita	2016/04/19
E0014	Jordan	Nil	2016/05/01
NULL	NULL	NULL	2016/03/01

Enter the following SQL statement:

SELECT Employee.Emp_id, Joining.Joining_Date

FROM Employee

LEFT OUTER JOIN Joining

ON Employee.Emp_id = Joining.Emp_id

ORDER BY Employee.Emp_id;

There will be 4 records selected. **You will see the following results:**

Emp_Id	Joining_Date
NULL	*NULL*
E0012	2016/04/18
E0013	2016/04/19
E0014	2016/05/01

RIGHT JOIN (RIGHT OUTER JOIN): This joins returns all rows from the RIGHT table and its matched rows from the LEFT table**.**

Syntax:

SELECT column_name(s)

FROM table_name1

RIGHT JOIN table_name2

ON column_name1=column_name2;

<u>For Example,</u>

In this example, we have a table **Employee** with the following data:

Emp_Id	Last_Name	First_Name	Job_Role
E0011	Verma	Akhil	Administration
E0012	Samson	Nikita	Asst. Manager
E0013	Jordan	Nil	In charge
E0014	Smith	Joe	Technician

The second table's name is **Joining.**

Emp_Id	Last_Name	First_Name	Joining_Date
E0012	Verma	Akhil	2016/04/18
E0013	Samson	Nikita	2016/04/19
E0014	Jordan	Nil	2016/05/01
NULL	NULL	NULL	2016/03/01

Enter the following SQL statement:

SELECT Employee.Emp_id, Joining.Joining_Date **FROM** Employee

RIGHT JOIN Joining

ON Employee.Emp_id = Joining.Emp_id

ORDER BY Employee.Emp_id;

Output:

Emp_id	**Joining_Date**
E0012	2016/04/18
E0013	2016/04/19
E0014	2016/05/01

FULL JOIN (FULL OUTER JOIN): This joins returns all results when there is a match either in the RIGHT table or in the LEFT table**.**

Syntax:

SELECT column_name(s)

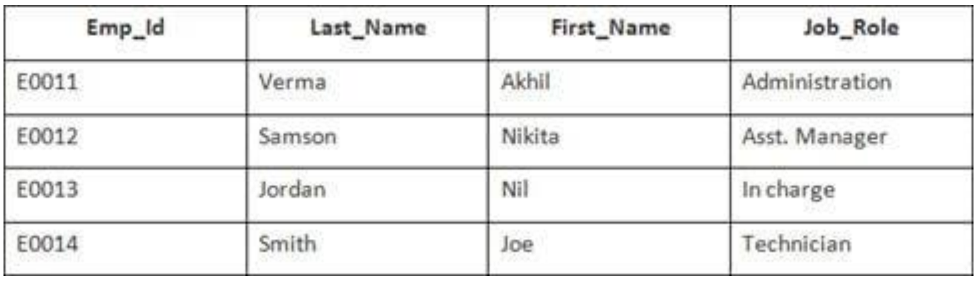
FROM table_name1

FULL OUTER JOIN table_name2

ON column_name1=column_name2;

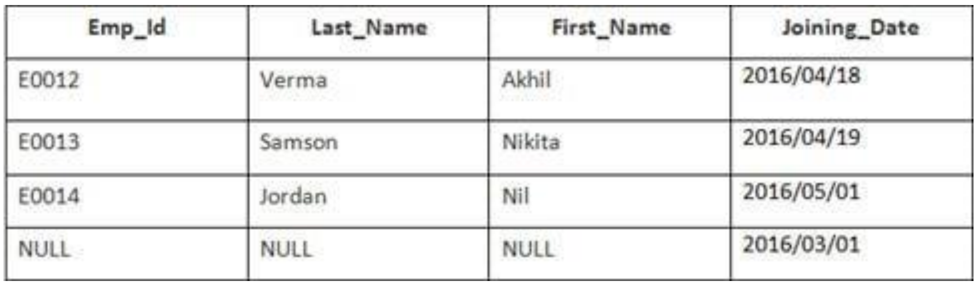
<u>**For Example,**</u>

In this example, we have a table **Employee** with the following data:



Emp_Id	Last_Name	First_Name	Job_Role
E0011	Verma	Akhil	Administration
E0012	Samson	Nikita	Asst. Manager
E0013	Jordan	Nil	In charge
E0014	Smith	Joe	Technician

The second table's name is **Joining.**



Emp_Id	Last_Name	First_Name	Joining_Date
E0012	Verma	Akhil	2016/04/18
E0013	Samson	Nikita	2016/04/19
E0014	Jordan	Nil	2016/05/01
NULL	NULL	NULL	2016/03/01

Enter the following SQL statement:

SELECT Employee.Emp_id, Joining.Joining_Date

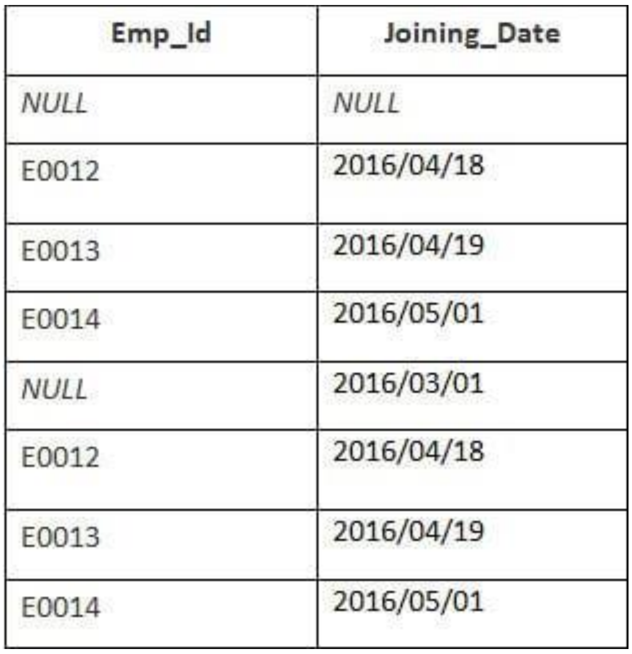
FROM Employee

FULL OUTER JOIN Joining

ON Employee.Emp_id = Joining.Emp_id

ORDER BY Employee.Emp_id;

There will be 8 records selected. **These are the results that you should see.**



Emp_Id	Joining_Date
NULL	*NULL*
E0012	2016/04/18
E0013	2016/04/19
E0014	2016/05/01
NULL	2016/03/01
E0012	2016/04/18
E0013	2016/04/19
E0014	2016/05/01

Q #8) What are transactions and their controls?

Answer: A transaction can be defined as the sequence task that is performed on databases in a logical manner to gain certain results. Operations like Creating, updating, deleting records performed in the database come from transactions.

In simple words, we can say that a transaction means a group of SQL queries executed on database records.

There are 4 transaction controls such as

- **COMMIT**: It is used to save all changes made through the transaction.

- **ROLLBACK**: It is used to roll back the transaction. All changes made by the transaction are reverted back and the database remains as before.
- **SET TRANSACTION**: Set the name of the transaction.
- **SAVEPOINT:** It is used to set the point where the transaction is to be rolled back.

Q #9) What are the properties of the transaction?

Answer: Properties of the transaction are known as ACID properties. These are:

- **Atomicity**: Ensures the completeness of all transactions performed. Checks whether every transaction is completed successfully or not. If not, then the transaction is aborted at the failure point and the previous transaction is rolled back to its initial state as changes are undone.
- **Consistency**: Ensures that all changes made through successful transactions are reflected properly on the database.
- **Isolation**: Ensures that all transactions are performed independently and changes made by one transaction are not reflected on others.
- **Durability**: Ensures that the changes made in the database with committed transactions persist as it is even after a system failure.

Q #10) How many Aggregate functions are available in SQL?

Answer: SQL Aggregate functions determine and calculate values from multiple columns in a table and return a single value.

There are 7 aggregate functions in SQL:

- **AVG():** Returns the average value from specified columns.
- **COUNT():** Returns number of table rows.
- **MAX():** Returns the largest value among the records.
- **MIN():** Returns smallest value among the records.
- **SUM():** Returns the sum of specified column values.
- **FIRST():** Returns the first value.
- **LAST():** Returns last value.

Q #11) What are Scalar functions in SQL?

Answer: Scalar functions are used to return a single value based on the input values.

Scalar Functions are as follows:

- **UCASE():** Converts the specified field in the upper case.
- **LCASE():** Converts the specified field in lower case.

- **MID():** Extracts and returns character from the text field.
- **FORMAT():** Specifies the display format.
- **LEN():** Specifies the length of the text field.
- **ROUND():** Rounds up the decimal field value to a number.

Q #12) What are triggers?

Answer: Triggers in SQL is kind of stored procedures used to create a response to a specific action performed on the table such as INSERT, UPDATE or DELETE. You can invoke triggers explicitly on the table in the database.

Action and Event are two main components of SQL triggers. When certain actions are performed, the event occurs in response to that action.

Syntax:

CREATE TRIGGER name {BEFORE|**AFTER**} (event [OR..]}

ON table_name [**FOR** [EACH] {ROW|STATEMENT}]

EXECUTE PROCEDURE functionname {arguments}

Q #13) What is View in SQL?

Answer: A View can be defined as a virtual table that contains rows and columns with fields from one or more tables.

Syntax:

CREATE VIEW view_name **AS**

SELECT column_name(s)

FROM table_name

WHERE condition

Q #14) How we can update the view?

Answer: SQL CREATE and REPLACE can be used for updating the view.

Execute the below query to update the created view.

Syntax:

CREATE OR REPLACE **VIEW** view_name **AS**

SELECT column_name(s)

FROM table_name

WHERE condition

Q #15) Explain the working of SQL Privileges?

Answer: SQL GRANT and REVOKE commands are used to implement privileges in SQL multiple user environments. The administrator of the database can grant or revoke privileges to or from users of database objects by using commands like SELECT, INSERT, UPDATE, DELETE, ALL, etc.

GRANT Command: This command is used to provide database access to users other than the administrator.

Syntax:

GRANT privilege_name

ON object_name

TO {user_name|**PUBLIC**|role_name}

[**WITH GRANT OPTION**];

In the above syntax, the GRANT option indicates that the user can grant access to another user too.

REVOKE command: This command is used to provide database deny or remove access to database objects.

Syntax:

REVOKE privilege_name

ON object_name

FROM {user_name|**PUBLIC**|role_name};

Q #16) How many types of Privileges are available in SQL?

Answer: There are two types of privileges used in SQL, such as

- **System privilege:** System privilege deals with the object of a particular type and provides users the right to perform one or more actions on it. These actions include performing administrative tasks, ALTER ANY INDEX, ALTER ANY CACHE GROUP CREATE/ALTER/DELETE TABLE, CREATE/ALTER/DELETE VIEW etc.
- **Object privilege:** This allows to perform actions on an object or object of another user(s) viz. table, view, indexes etc. Some of the object privileges are EXECUTE, INSERT, UPDATE, DELETE, SELECT, FLUSH, LOAD, INDEX, REFERENCES etc.

Q #17) What is SQL Injection?

Answer: SQL Injection is a type of database attack technique where malicious SQL statements are inserted into an entry field of database in a way that once it is executed, the database is exposed to an attacker for the attack. This technique is usually used for attacking data-driven applications to have access to sensitive data and perform administrative tasks on databases.

For Example,

SELECT column_name(s) **FROM** table_name **WHERE** condition;

Q #18) What is SQL Sandbox in SQL Server?

Answer: SQL Sandbox is a safe place in the SQL server environment where untrusted scripts are executed. There are 3 types of SQL sandbox:

- **Safe Access Sandbox:** Here a user can perform SQL operations such as creating stored procedures, triggers etc. but cannot have access to the memory as well as cannot create files.
- **External Access Sandbox:** Users can access files without having the right to manipulate the memory allocation.
- **Unsafe Access Sandbox:** This contains untrusted codes where a user can have access to memory.

Q #19) What is the difference between SQL and PL/SQL?

Answer: SQL is a Structured Query Language to create and access databases whereas PL/SQL comes with procedural concepts of programming languages.

Q #20) What is the difference between SQL and MySQL?

Answer: SQL is a Structured Query Language that is used for manipulating and accessing the relational database. On the other hand, MySQL itself is a relational database that uses SQL as the standard database language.

Q #21) What is the use of the NVL function?

Answer: NVL function is used to convert the null value to its actual value.

Q #22) What is the Cartesian product of the table?

Answer: The output of Cross Join is called a Cartesian product. It returns rows combining each row from the first table with each row of the second table. **For Example,** if we join two tables having 15 and 20 columns the Cartesian product of two tables will be 15×20=300 rows.

Q #23) What do you mean by Subquery?

Answer: Query within another query is called as Subquery. A subquery is called inner query which returns output that is to be used by another query.

Q #24) How many row comparison operators are used while working with a subquery?

Answer: There are 3-row comparison operators that are used in subqueries such as IN, ANY and ALL.

Q #25) What is the difference between clustered and non-clustered indexes?

Answer: The differences between the two are as follows:

- One table can have only one clustered index but multiple non-clustered indexes.
- Clustered indexes can be read rapidly rather than non-clustered indexes.
- Clustered indexes store data physically in the table or view whereas, non-clustered indexes do not store data in the table as it has separate structure from the data row.

Q #26) What is the difference between DELETE and TRUNCATE?

Answer: The differences are:

- The basic difference in both is DELETE command is DML command and the TRUNCATE command is DDL.
- DELETE command is used to delete a specific row from the table whereas the TRUNCATE command is used to remove all rows from the table.
- We can use the DELETE command with WHERE clause but cannot use the TRUNCATE command with it.

Q #27) What is the difference between DROP and TRUNCATE?

Answer: TRUNCATE removes all rows from the table which cannot be retrieved back, DROP removes the entire table from the database and it also cannot be retrieved back.

**Q #28) How to write a query to show the details of a student from Students table whose
name start with K?**

Answer: Query:

SELECT * **FROM** Student **WHERE** Student_Name like ‘K%’;

Here ‘like’ operator is used to perform pattern matching.

Q #29) What is the difference between Nested Subquery and Correlated Subquery?

Answer: Subquery within another subquery is called Nested Subquery. If the output of a subquery depends on column values of the parent query table then the query is called Correlated Subquery.

SELECT adminid(SELEC Firstname+' '+Lastname&nbsp;&nbsp;**FROM** Employee **WHERE** empid=emp. adminid)**AS** EmpAdminId **FROM** Employee;

The result of the query is the details of an employee from the Employee table.

Q #30) What is Normalization? How many Normalization forms are there?

Answer: Normalization is used to organize the data in such a manner that data redundancy will never occur in the database and avoid insert, update and delete anomalies.

There are 5 forms of Normalization:

- **First Normal Form (1NF):** It removes all duplicate columns from the table. It creates a table for related data and identifies unique column values.
- **First Normal Form (2NF):** Follows 1NF and creates and places data subsets in an individual table and defines the relationship between tables using the primary key.
- **Third Normal Form (3NF):** Follows 2NF and removes those columns which are not related through the primary key.
- **Fourth Normal Form (4NF):** Follows 3NF and does not define multi-valued dependencies. 4NF is also known as BCNF.

Q #31) What is a Relationship? How many types of Relationships are there?

Answer: The relationship can be defined as the connection between more than one table in the database.

There are 4 types of relationships:

- One to One Relationship
- Many to One Relationship
- Many to Many Relationship
- One to Many Relationship

Q #32) What do you mean by Stored Procedures? How do we use it?

Answer: A stored procedure is a collection of SQL statements that can be used as a function to access the database. We can create these stored procedures earlier before using it and can execute them wherever required by applying some conditional logic to it. Stored procedures are also used to reduce network traffic and improve performance.

Syntax:

CREATE Procedure Procedure_Name

(

//Parameters

)

AS

BEGIN

SQL statements in stored procedures **to update**/retrieve records

END

Q #33) State some properties of Relational databases?

Answer: Properties are as follows:

- In relational databases, each column should have a unique name.
- The sequence of rows and columns in relational databases is insignificant.
- All values are atomic and each row is unique.

Q #34) What are Nested Triggers?

Answer: Triggers may implement data modification logic by using INSERT, UPDATE, and DELETE statements. These triggers that contain data modification logic and find other triggers for data modification are called Nested Triggers.

Q #35) What is a Cursor?

Answer: A cursor is a database object which is used to manipulate data in a row-to-row manner.

Cursor follows steps as given below:

- Declare Cursor
- Open Cursor
- Retrieve row from the Cursor
- Process the row
- Close Cursor
- Deallocate Cursor

Q #36) What is Collation?

Answer: Collation is a set of rules that check how the data is sorted by comparing it. Such as character data is stored using correct character sequence along with case sensitivity, type, and accent.

Q #37) What do we need to check in Database Testing?

Answer: In Database testing, the following thing is required to be tested:

- Database connectivity
- Constraint check
- Required application field and its size
- Data Retrieval and processing with DML operations
- Stored Procedures
- Functional flow

Q #38) What is Database White Box Testing?

Answer: Database White Box testing involves:

- Database Consistency and ACID properties
- Database triggers and logical views
- Decision Coverage, Condition Coverage, and Statement Coverage
- Database Tables, Data Model, and Database Schema
- Referential integrity rules

Q #39) What is Database Black Box Testing?

Answer: Database Black Box testing involves:

- Data Mapping
- Data stored and retrieved
- Use of Black Box testing techniques such as Equivalence Partitioning and Boundary Value Analysis (BVA)

Q #40) What are Indexes in SQL?

Answer: The index can be defined as the way to retrieve the data more quickly. We can define indexes using CREATE statements.

Syntax:

CREATE INDEX index_name

ON table_name (column_name)

Further, we can also create a Unique Index using the following syntax:

CREATE UNIQUE INDEX index_name

ON table_name (column_name)

UPDATE: We have added few more short questions for practice.

Q #41) What does SQL stand for?

Answer: SQL stands for Structured Query Language.

Q #42) How to select all records from the table?

Answer: To select all the records from the table we need to use the following syntax:

Select * **from** table_name;

Q #43) Define join and name different types of joins?

Answer: Join keyword is used to fetch data from two or more related tables. It returns rows where there is at least one match in both the tables included in the join. Read more here.
Type of joins are:

1. Right join
2. Outer join
3. Full join
4. Cross join
5. Self join.

Q #44) What is the syntax to add a record to a table?

Answer: To add a record in a table INSERT syntax is used.

For Example,

INSERT into table_name **VALUES** (value1, value2..);

Q #45) How do you add a column to a table?

Answer: To add another column in the table, use the following command:

ALTER TABLE table_name **ADD** (column_name);

Recommended reading =>> How to add a column to a table in MySQL

Q #46) Define the SQL DELETE statement.

Answer: DELETE is used to delete a row or rows from a table based on the specified condition.
The basic syntax is as follows:

DELETE FROM table_name

WHERE &lt;Condition&gt;

Q #47) Define COMMIT?

Answer: COMMIT saves all changes made by DML statements.

Q #48) What is the Primary key?

Answer: A Primary key is a column whose values uniquely identify every row in a table. Primary key values can never be reused.

Q #49) What are Foreign keys?

Answer: When a table's primary key field is added to related tables in order to create the common field which relates the two tables, it called a foreign key in other tables. Foreign key constraints enforce referential integrity.

Q #50) What is CHECK Constraint?

Answer: A CHECK constraint is used to limit the values or type of data that can be stored in a column. They are used to enforce domain integrity.

Q #51) Is it possible for a table to have more than one foreign key?

Answer: Yes, a table can have many foreign keys but only one primary key.

Q #52) What are the possible values for the BOOLEAN data field?

Answer: For a BOOLEAN data field, two values are possible: -1(true) and 0(false).

Q #53) What is a stored procedure?

Answer: A stored procedure is a set of SQL queries that can take input and send back output.

Q #54) What is identity in SQL?

Answer: An identity column in where SQL automatically generates numeric values. We can define a start and increment value of the identity column.

Q #55) What is Normalization?

Answer: The process of table design to minimize the data redundancy is called normalization. We need to divide a database into two or more table and define the relationship between them.

Q #56) What is a Trigger?

Answer: The Trigger allows us to execute a batch of SQL code when table event occurs (INSERT, UPDATE or DELETE commands are executed against a specific table).

Q #57) How to select random rows from a table?

Answer: Using a SAMPLE clause we can select random rows.

For Example,

SELECT * **FROM** table_name SAMPLE(10);

Q #58) Which TCP/IP port does SQL Server run?

Answer: By default SQL Server runs on port 1433.

Q #59) Write a SQL SELECT query that only returns each name only once from a table?

Answer: To get the result as each name only once, we need to use the DISTINCT keyword.

SELECT DISTINCT name FROM table_name;

Q #60) Explain DML and DDL?

Answer: DML stands for Data Manipulation Language. INSERT, UPDATE and DELETE are DML statements.

DDL stands for Data Definition Language. CREATE, ALTER, DROP, RENAME are DDL statements.

Q #61) Can we rename a column in the output of the SQL query?

Answer: Yes, using the following syntax we can do this.

SELECT column_name **AS** new_name **FROM** table_name;

Q #62) Give the order of SQL SELECT?

Answer: Order of SQL SELECT clauses is: SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY. Only the SELECT and FROM clauses are mandatory.

Q #63) Suppose a Student column has two columns, Name and Marks. How to get names and marks of the top three students.
Answer: SELECT Name, Marks FROM Student s1 where 3 <= (SELECT COUNT(*) FROM Students s2 WHERE s1.marks = s2.marks)

Q #64) What is SQL comments?

Answer: SQL comments can be inserted by adding two consecutive hyphens (–).

Q #65) Difference between TRUNCATE, DELETE and DROP commands?

Answer:

- **DELETE** removes some or all rows from a table based on the condition. It can be rolled back.
- **TRUNCATE** removes ALL rows from a table by de-allocating the memory pages. The operation cannot be rolled back

- **DROP** command removes a table from the database completely.

Q #66) What are the properties of a transaction?

Answer: Generally, these properties are referred to as ACID properties. They are:

1. Atomicity
2. Consistency
3. Isolation
4. Durability.

Q #67) What do you mean by ROWID?

Answer: It's an 18 character long pseudo column attached with each row of a table.

Q #68) Define UNION, MINUS, UNION ALL, INTERSECT?

Answer:

- **MINUS** – returns all distinct rows selected by the first query but not by the second.
- **UNION** – returns all distinct rows selected by either query
- **UNION ALL** – returns all rows selected by either query, including all duplicates.
- **INTERSECT** – returns all distinct rows selected by both queries.

Q #69) What is a transaction?

Answer: A transaction is a sequence of code that runs against a database. It takes the database from one consistent state to another.

Q #70) What is the difference between UNIQUE and PRIMARY KEY constraints?

Answer: The differences are as follows:

- A table can have only one PRIMARY KEY whereas there can be any number of UNIQUE keys.
- The primary key cannot contain Null values whereas the Unique key can contain Null values.

Q #71) What is a composite primary key?

Answer: The primary key created on more than one column is called composite primary key.

Q #72) What is an Index?

Answer: An Index is a special structure associated with a table to speed up the performance of queries. The index can be created on one or more columns of a table.

Q #73) What is the Subquery?

Answer: A Subquery is a subset of select statements whose return values are used in filtering conditions of the main query.

Q #74) What do you mean by query optimization?

Answer: Query optimization is a process in which a database system compares different query strategies and select the query with the least cost.

Q #75) What is Collation?

Answer: Set of rules that define how data is stored, how case-sensitivity and Kana character can be treated etc.

Q #76) What is Referential Integrity?

Answer: Set of rules that restrict the values of one or more columns of the tables based on the values of the primary key or unique key of the referenced table.

Q #77) What is the Case function?

Answer: Case facilitates if-then-else type of logic in SQL. It evaluates a list of conditions and returns one of the multiple possible result expressions.

Q #78) Define a temp table?

Answer: A temp table is a temporary storage structure to store the data temporarily.

Q #79) How can we avoid duplicating records in a query?

Answer: By using the DISTINCT keyword, duplication of records in a query can be avoided.

Q #80) Explain the difference between Rename and Alias?

Answer: Rename is a permanent name given to a table or column whereas Alias is a temporary name given to a table or column.

Q #81) What is a View?

Answer: A view is a virtual table that contains data from one or more tables. Views restrict data access of the table by selecting only required values and make complex queries easy.

Q #82) What are the advantages of Views?

Answer: Advantages of Views are:

- Views restrict access to the data because the view can display selective columns from the table.
- Views can be used to make simple queries to retrieve the results of complicated queries. **For Example,** views can be used to query information from multiple tables without the user knowing.

Q #83) List the various privileges that a user can grant to another user?

Answer: SELECT, CONNECT, RESOURCES.

Q #84) What is schema?

Answer: A schema is a collection of database objects of a User.

Q #85) What is a Table?

Answer: A table is the basic unit of data storage in the database management system. Table data is stored in rows and columns.

Q #86) Does View contain Data?

Answer: No, Views are virtual structures.

Q #87) Can a View based on another View?

Answer: Yes, A View is based on another View.

Q #88) What is the difference between the HAVING clause and WHERE clause?

Answer: Both specify a search condition but Having clause is used only with the SELECT statement and typically used with GROUP BY clause.
If GROUP BY clause is not used then Having behaved like WHERE clause only.

Q #89) What is the difference between Local and Global temporary tables?

Answer: If defined inside a compound statement a local temporary table exists only for the duration of that statement but a global temporary table exists permanently in the DB but its rows disappear when the connection is closed.

Q #90) What is CTE?

Answer: A CTE or common table expression is an expression that contains temporary result set which is defined in a SQL statement.

What is the difference between SQL and MySQL or SQL Server?

SQL or Structured Query Language is a language; language that communicates with a relational database thus providing ways of manipulating and creating databases. MySQL and Microsoft's SQL Server both are relational database management systems that use SQL as their standard relational database language.

What is the difference between SQL and PL/SQL?

PL/SQL is a dialect of SQL that adds procedural features of programming languages in SQL. It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL.

What are various DDL commands in SQL? Give brief description of their purposes.

Following are various DDL or Data Definition Language commands in SQL −

CREATE − it creates a new table, a view of a table, or other object in database.

ALTER − it modifies an existing database object, such as a table.

DROP − it deletes an entire table, a view of a table or other object in the database.

What are various DML commands in SQL? Give brief description of their purposes.

Following are various DML or Data Manipulation Language commands in SQL −

SELECT − it retrieves certain records from one or more tables.

INSERT − it creates a record.

UPDATE − it modifies records.

DELETE − it deletes records.

What are various DCL commands in SQL? Give brief description of their purposes.

Following are various DCL or Data Control Language commands in SQL −

GRANT − it gives a privilege to user.

REVOKE − it takes back privileges granted from user.

Can you sort a column using a column alias?

Yes. A column alias could be used in the ORDER BY clause.

Is a NULL value same as zero or a blank space? If not then what is the difference?

A NULL value is not same as zero or a blank space. A NULL value is a value which is 'unavailable, unassigned, unknown or not applicable'. Whereas, zero is a number and blank space is a character.

Say True or False. Give explanation if False.

If a column value taking part in an arithmetic expression is NULL, then the result obtained would be NULLM.

True.

If a table contains duplicate rows, does a query result display the duplicate values by default? How can you eliminate duplicate rows from a query result?

A query result displays all rows including the duplicate rows. To eliminate duplicate rows in the result, the DISTINCT keyword is used in the SELECT clause.

What is the purpose of the condition operators BETWEEN and IN?

The BETWEEN operator displays rows based on a range of values. The IN condition operator checks for values contained in a specific set of values.

How do you search for a value in a database table when you don't have the exact value to search for?

In such cases, the LIKE condition operator is used to select rows that match a character pattern. This is also called 'wildcard' search.

What is the default ordering of data using the ORDER BY clause? How could it be changed?

The default sorting order is ascending. It can be changed using the DESC keyword, after the column name in the ORDER BY clause.

What are the specific uses of SQL functions?

SQL functions have the following uses −

Performing calculations on data

Modifying individual data items

Manipulating the output

Formatting dates and numbers and Converting data types

What are the case manipulation functions of SQL?

LOWER, UPPER, INITCAP

Which function returns the remainder in a division operation?

The MOD function returns the remainder in a division operation.

What is the purpose of the NVL function?

The NVL function converts a NULL value to an actual value.

What is the difference between the NVL and the NVL2 functions?

The NVL(exp1, exp2) function converts the source expression (or value) exp1 to the target expression (or value) exp2, if exp1 contains NULL. The return value has the same data type as that of exp1.

The NVL2(exp1, exp2, exp3) function checks the first expression exp1, if it is not null then, the second expression exp2 is returned. If the first expression exp1 is null, then the third expression exp3 is returned.

What is the use of the NULLIF function?

The NULLIF function compares two expressions. If they are equal, the function returns null. If they are not equal, the first expression is returned.

Discuss the syntax and use of the COALESCE function?

The COALESCE function has the expression COALESCE(exp1, exp2, …. expn)

It returns the first non-null expression given in the parameter list.

Which expressions or functions allow you to implement conditional processing in a SQL statement?

There are two ways to implement conditional processing or IF-THEN-ELSE logic in a SQL statement.

Using CASE expression

Using the DECODE function

You want to display a result query from joining two tables with 20 and 10 rows respectively. Erroneously you forget to write the WHERE clause. What would be the result?

The result would be the Cartesian product of two tables with 20 x 10 = 200 rows.

What is the difference between cross joins and natural joins?

The cross join produces the cross product or Cartesian product of two tables. The natural join is based on all the columns having same name and data types in both the tables.

What is the purpose of the group functions in SQL? Give some examples of group functions.

Group functions in SQL work on sets of rows and returns one result per group. Examples of group functions are AVG, COUNT, MAX, MIN, STDDEV, SUM, VARIANCE.

Say True or False. Give explanation if False.

By default the group functions consider only distinct values in the set.

By default, group functions consider all values including the duplicate values.

Say True or False. Give explanation if False.

The DISTINCT keyword allows a function consider only non-duplicate values.

True.

Say True or False. Give explanation if False.

All group functions ignore null values.

True.

Say True or False. Give explanation if False.

COUNT(*) returns the number of columns in a table.

False. COUNT(*) returns the number of rows in a table.

What's wrong in the following query?

```
SELECT subject_code, count(name)

FROM students;
```

It doesn't have a GROUP BY clause. The subject_code should be in the GROUP BY clause.

```
SELECT subject_code, count(name)

FROM students

GROUP BY subject_code;
```

What’s wrong in the following query?

```
SELECT subject_code, AVG (marks)
FROM students
WHERE AVG(marks) > 75
GROUP BY subject_code;
```

The WHERE clause cannot be used to restrict groups. The HAVING clause should be used.

```
SELECT subject_code, AVG (marks)
FROM students
HAVING AVG(marks) > 75
GROUP BY subject_code;
```

Say True or False. Give explanation if False.

Group functions cannot be nested.

False. Group functions can be nested to a depth of two.

What do you understand by a subquery? When is it used?

A subquery is a SELECT statement embedded in a clause of another SELECT statement. It is used when the inner query, or the subquery returns a value that is used by the outer query. It is very useful in selecting some rows in a table with a condition that depends on some data which is contained in the same table.

Say True or False. Give explanation if False.

A single row subquery returns only one row from the outer SELECT statement

False. A single row subquery returns only one row from the inner SELECT statement.

Say True or False. Give explanation if False.

A multiple row subquery returns more than one row from the inner SELECT statement.

True.

Say True or False. Give explanation if False.

Multiple column subqueries return more than one column from the inner SELECT statement.

True.

What’s wrong in the following query?

```
SELECT student_code, name
FROM students
WHERE marks =
        (SELECT MAX(marks)
          FROM students
          GROUP BY subject_code);
```

Here a single row operator = is used with a multiple row subquery.

What are the various multiple row comparison operators in SQL?

IN, ANY, ALL.

What is the pupose of DML statements in SQL?

The DML statements are used to add new rows to a table, update or modify data in existing rows, or remove existing rows from a table.

Which statement is used to add a new row in a database table?

The INSERT INTO statement.

Say True or False. Give explanation if False.

While inserting new rows in a table you must list values in the default order of the columns.

True.

How do you insert null values in a column while inserting data?

Null values can be inserted into a table by one of the following ways −

Implicitly by omitting the column from the column list.

Explicitly by specifying the NULL keyword in the VALUES clause.

Say True or False. Give explanation if False.

INSERT statement does not allow copying rows from one table to another.

False. INSERT statement allows to add rows to a table copying rows from an existing table.

How do you copy rows from one table to another?

The INSERT statement can be used to add rows to a table by copying from another table. In this case, a subquery is used in the place of the VALUES clause.

What happens if you omit the WHERE clause in the UPDATE statement?

All the rows in the table are modified.

Can you modify the rows in a table based on values from another table? Explain.

Yes. Use of subqueries in UPDATE statements allow you to update rows in a table based on values from another table.

Say True or False. Give explanation if False.

The DELETE statement is used to delete a table from the database.

False. The DELETE statement is used for removing existing rows from a table.

What happens if you omit the WHERE clause in a delete statement?

All the rows in the table are deleted.

Can you remove rows from a table based on values from another table? Explain.

Yes, subqueries can be used to remove rows from a table based on values from another table.

Say True or False. Give explanation if False.

Attempting to delete a record with a value attached to an integrity constraint, returns an error.

True.

Say True or False. Give explanation if False.

You can use a subquery in an INSERT statement.

True.

What is the purpose of the MERGE statement in SQL?

The MERGE statement allows conditional update or insertion of data into a database table. It performs an UPDATE if the rows exists, or an INSERT if the row does not exist.

Say True or False. Give explanation if False.

A DDL statement or a DCL statement is automatically committed.

True.

What is the difference between VARCHAR2 AND CHAR datatypes?

VARCHAR2 represents variable length character data, whereas CHAR represents fixed length character data.

Say True or False. Give explanation if False.

A DROP TABLE statement can be rolled back.

False. A DROP TABLE statement cannot be rolled back.

Which SQL statement is used to add, modify or drop columns in a database table?

The ALTER TABLE statement.

What is a view? Why should you use a view?

A view is a logical snapshot based on a table or another view. It is used for −

Restricting access to data;

Making complex queries simple;

Ensuring data independency;

Providing different views of same data.

Say True or False. Give explanation if False.

A view doesn’t have data of its own.

True.