

(DATABASE MANAGEMENT SYSTEM)

SESSION 2022-23

DBMS

Lab File

**COURSE:- BCA**

**ROLL NO :- 41221139**

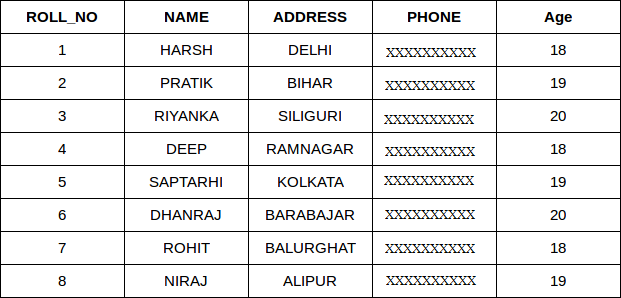
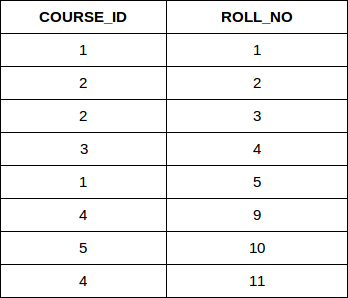
**SUBMITTED BY :- SUBMITTED TO:-**

**Sachin Rajbhar Ms. Bushra Jamal**

INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Name of practical | Date | Sign |
| 1. | Installation of MySQL Command mode and Workbench |  |  |
| 2. | Introduction to SQL   * About DDL,DML,DCL,TCL * SQL Data Types * SQL Operators |  |  |
| 3. | Write queries in context of basic database related operations in MySQL   * Create database BCA in MySQL. * Show currently selected database * Switch from default database to my database(BCA) * List all the tables in a database * Show the schema of a table(e.g.: student) |  |  |
| 4. | Design the following tables in BCA database along with keys and constraints like:   * Primary Key * Foreign key * NOT NULL, and * check constraint   student(**student\_id**, lastname, firstname, marks) employee(**emp\_id**, emp\_name, emp\_salary, dept, phone, dept\_id, date-of-joining )  department(**dept\_id**, dept\_name) |  |  |
| 5. | Write SQL statement for implementing Alter and Drop   * ALTER – perform add, modify, rename(table, column) * DROP column |  |  |
| 6. | Write SQL queries to populate tables of BCA database and display the records(all tuples) |  |  |
| 7. | Write SQL query to create copy of student table(along with data), and Drop that copy |  |  |
| 8. | Write SQL query to show use of Update Statements  ( by create table copy\_emp as select \* from employee)   * update copy\_emp set salary=salary+1000; * update copy\_emp set dept\_name=”electrical” where emp\_name = ”james”; * update copy\_emp set phone=123456, dept\_id=20 where emp\_id = 10; * select \* from copy\_emp; |  |  |
| 9. | Write SQL query to show difference between truncate and drop command. |  |  |
| 10 | Write SQL query to select only DISTINCT records from a table containing duplicate tuples. |  |  |
| 11 | Write SQL query to implement all aggregate functions: MAX(), MIN(), AVG(), COUNT(), SUM() |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 12 | Perform the following operations:   * Delete record of an employee according to his emp\_id * Select sum of total salary of all employees from employee table; * Count total names of employees from employee table; * Perform insert, update and Delete on the data from a base table (that is referred by a foreign key from another table). |  |  |
| 13 | Write SQL queries to show the use of:   * Order by * Group by, having |  |  |
| 14 | Draw ER Diagram of employee works in department in ERDplus.com |  |  |
| 15 | Write SQL query to show the use of following **operators**:   * AND, OR, NOT * Use of Null(IS NULL, and IS NOT NULL) * LIKE(with wildcards such as % and \_ ) * IN, NOT IN * BETWEEN * ANY * ALL * TOP/LIMIT clause |  |  |
| 16 | Write SQL query to display the result of UNION operator |  |  |
| 17 | Write the queries to implement the following Joins:   * Inner join * Left join * Right join * Full join, on the given course and student tables |  |  |
| 18 | For the following database schema for a customer - sale scenario of a general store, perform the given queries:  **Customer**(cust id:integer, cust\_name: string)  **Item**(item\_id: integer, item\_name: string, price: float) **Sale**(bill\_no:integer, bill\_date:date, cust\_id:integer, item\_id: integer, qty\_sold: integer)   1. Create the tables with the appropriate integrity constraints 2. Insert around 10 records in each of the tables 3. List the total Bill details with the quantity sold, price of the item and the final amount 4. List the details of the customer who have bought a product which has a price>200 |  |  |



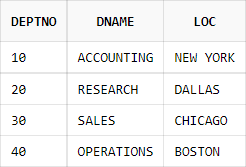
|  |  |  |  |
| --- | --- | --- | --- |
|  | 1. Give a count of how many products have been bought by each customer 2. Give a list of products bought by a customer having cust\_id as 5 3. List all the bills for the current date with the customer names and item numbers |  |  |
| 19 | Demonstrate the Application of auto increment on any of the above table in MySQL |  |  |
| 20 | Execute following functions in MySQL:   * For strings  1. To calculate the length of characters in a string 2. To concatenate strings 3. Convert a string into uppercase 4. Convert a string into lowercase 5. removes trailing spaces from a string 6. Remove leading spaces from a string 7. reverses a string and returns the result 8. Extract a substring from a string 9. Compare two strings 10. Replace all occurrences of a substring within a string, with a new substring.  * For numeric functions:  1. find the absolute value of a number 2. find ceil of a number 3. find floor of a number 4. find the greatest value from a list of argument 5. find the smallest value from the list of arguments 6. find the natural logarithm of 2 7. find the base-10 logarithm of 2 8. demonstrate integer division 9. show mod of a value(eg: 18 mod 4) 10. find the value of a number raised to the power of another number 11. Return the square root of a number 12. Return a number truncated to 3 decimal places  * For Date functions:  1. To Add 10 days to a date and return the date 2. To Add 2 seconds to a time and return the datetime 3. To Return the current date 4. To Return the number of days between two date values 5. To Return the current date and time 6. To Return the day of the month for a given date 7. To Extract a part from a given date 8. To Return the current date and time 9. To Return the day of the year for a given date 10. To Extract the last day of the month for a given date |  |  |

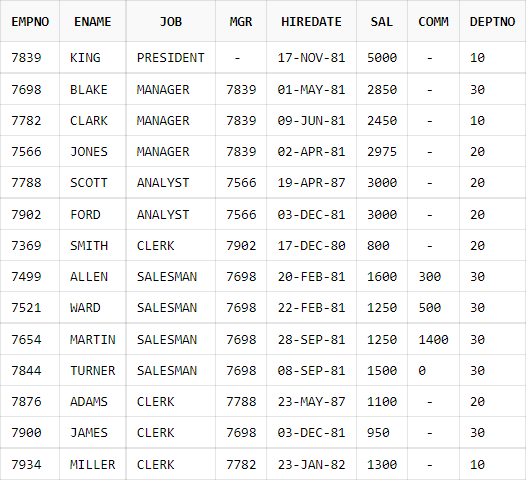
|  |  |  |  |
| --- | --- | --- | --- |
| 21 | * Write SQL query to create a view of student table * Write SQL query to create and update views   + from single table   + from multiple tables |  |  |
| 22 | Create employee and department table and provide queries to perform all basic DBMS operations as given in the task list(16 different queries in total). |  |  |
| 23 | For a given database EMPLOYEE with following attributes:  EMPLOYEE\_ID , FIRST\_NAME , LAST\_NAME , EMAIL , PHONE\_NUMBER, HIRE\_DATE, JOB\_ID , SALARY , MANAGER\_ID , DEPARTMENT\_ID   1. write a SQL query to find those employees whose salary matches the lowest salary of any of the departments. Return first name, last name and department ID. 2. write a SQL query to find those employees who receive a higher salary than the employee with ID 163. Return first name, last name. 3. write a SQL query to find out which employees have the same designation as the employee whose ID is 169. Return first name, last name, department ID and job ID. 4. write a SQL query to find those employees who report to that manager whose first name is ‘Payam’. Return first name, last name, employee ID and salary.   **Note** : show results via SQL subquery method |  |  |
| 24 | Create a Trigger |  |  |
| 25 | Create a procedure and Call it |  |  |
| 26 | Create a function and Call it |  |  |

Task list for Q22

Create employee and department tables as shown below and perform following operations:-

1. Change the name of employee with empno=7876.
2. Display the record of an employee with maximum salary.
3. Count total no of records from employee table.
4. Select all the records of an employee whose department location is “chicago”.
5. Write a query to get the records from emp table where deptno is 10 and job is “president”.
6. Write a query to get the records from emp table where deptno is 10 or salary is greater than 2000.
7. Write SQL query to select only distinct deptno from emp table.
8. Write SQL query to display the Ename starting with s,a or m in table employee.
9. Write SQL query to display name of employees in descending order.
10. Write SQL query to change the name of smith to “johnson” in employee table
11. Write SQL query to display only top 5 rows from employee table.
12. Write SQL query to select all the employees name whose deptno is 10 and 20(using IN operator)
13. Write SQL query to select the records from emp table whose salary is in the range of 1500 to 3000(using BETWEEN operator)
14. Delete the record of an employee from emp table whose name is ‘Ford’.
15. Apply all 4 joins on employee table and dept table(inner, left, right, outer) and show the resultant tables
16. Write SQL query to list the no of employees in each department(use Group By)





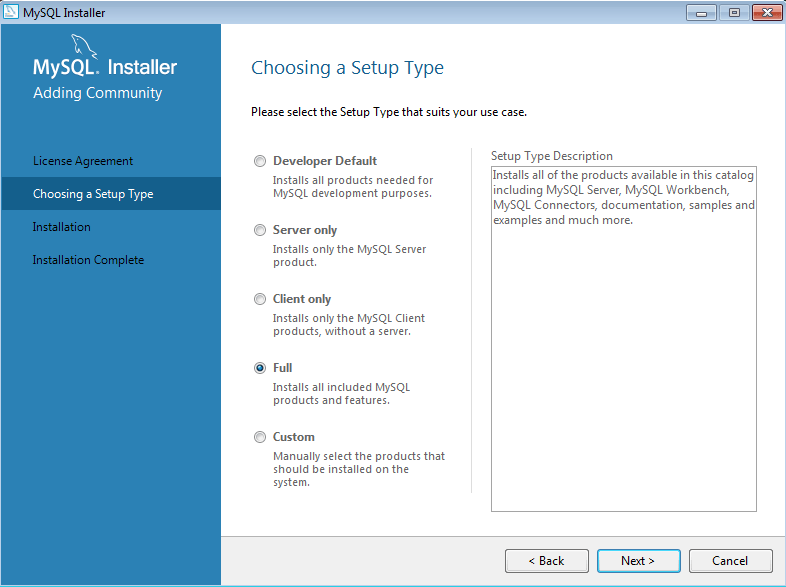
**PROGRAM 1**

**Q:** Installation of MySQL Command mode and Workbench

After downloading the MySQL community edition, double-click the installer file to begin the installation process. It’ll launch the installer window.

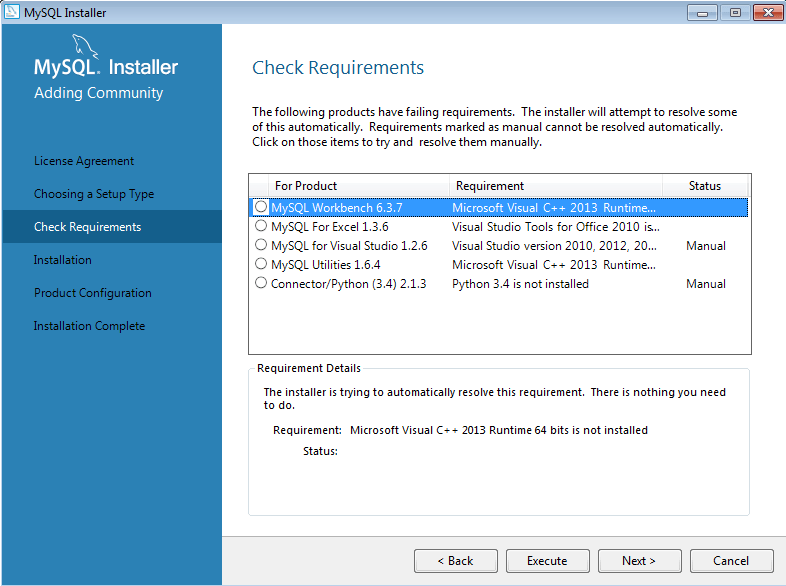
Subsequently, follow the below steps. Take help from the snapshots attached. Make sure to edit or select the option, as shown in the picture.

**Step-1.**



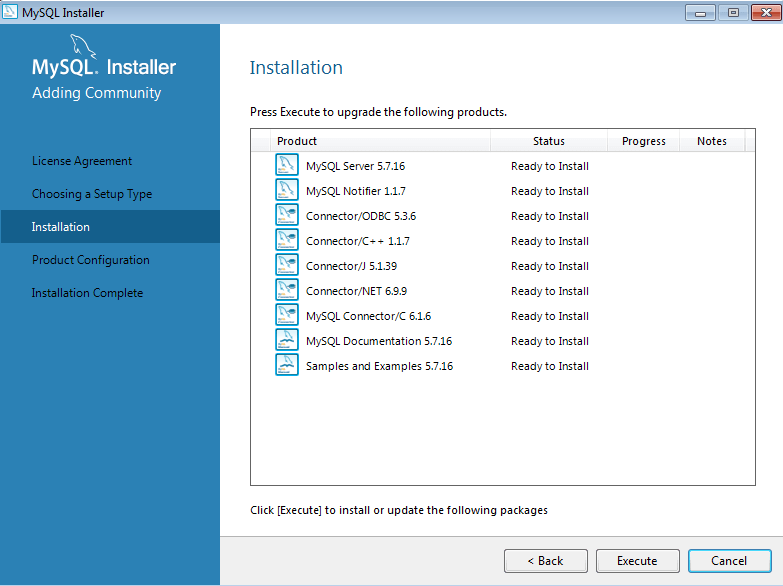
MySQL – Choose a Setup Type.

**Step-2.**



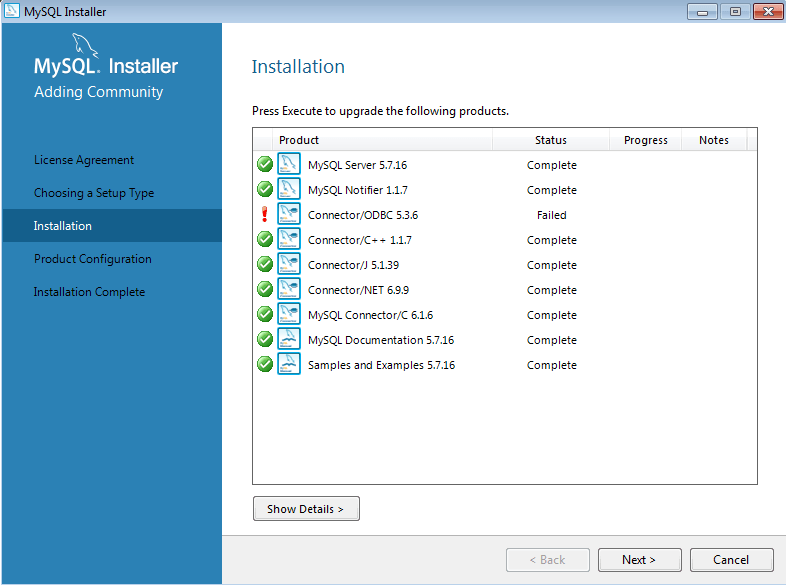
Check Requirements

**Step-3.**



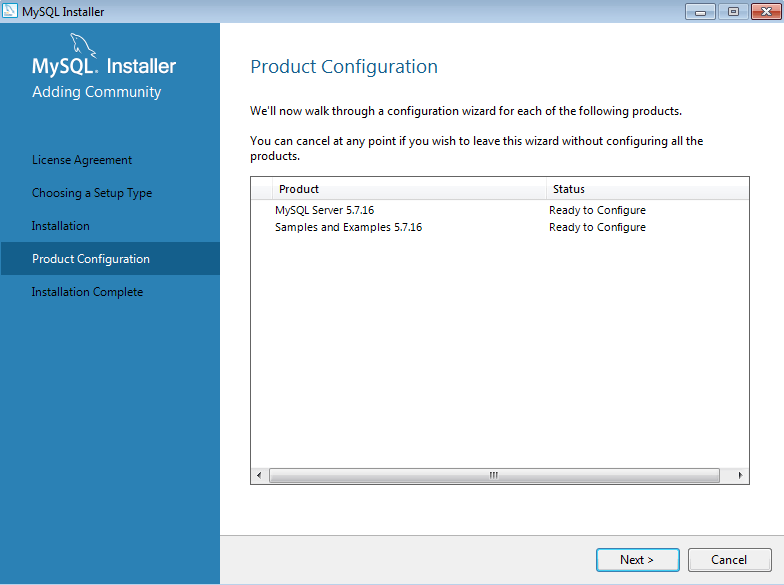
Begin Installation

**Step-4.**



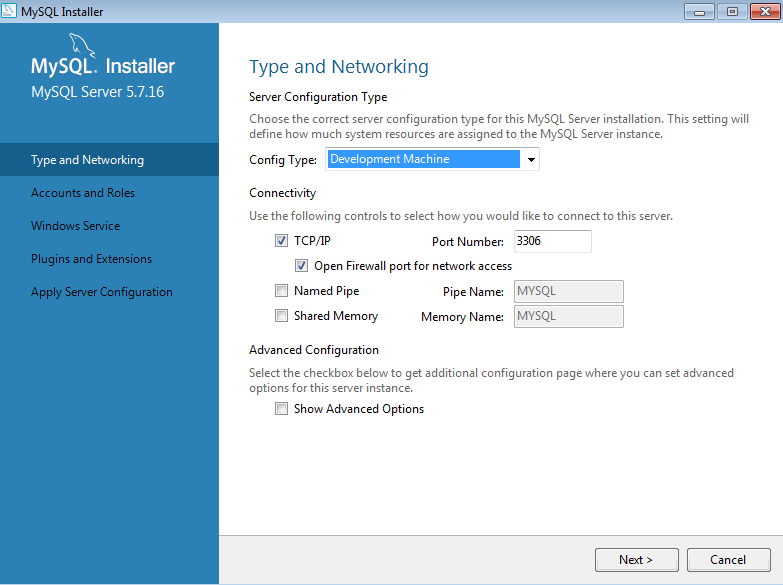
Installation Status.

**Step-5.**



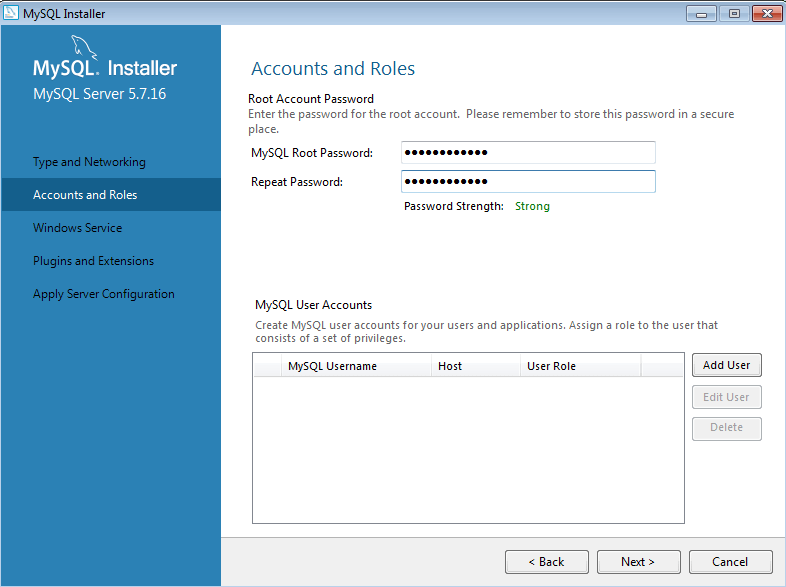
Open Configuration Screen.

**Step-6.**



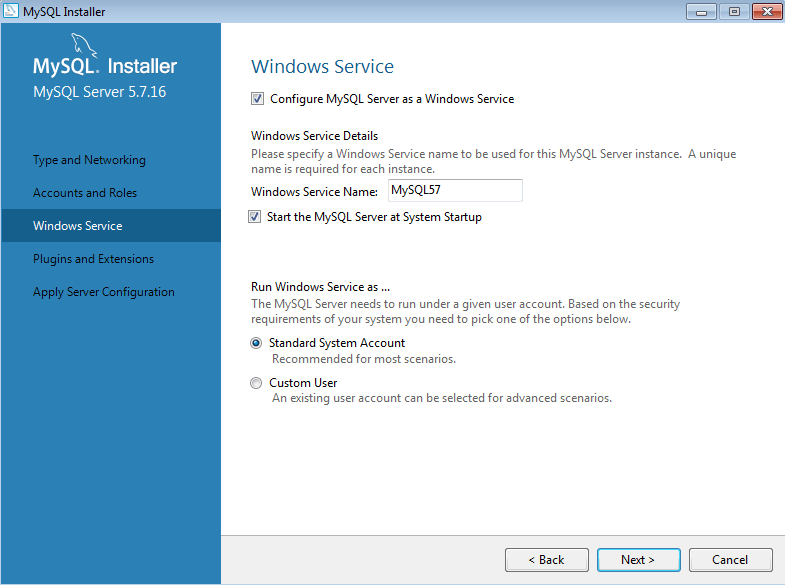
Edit Configuration.

**Step-7.**



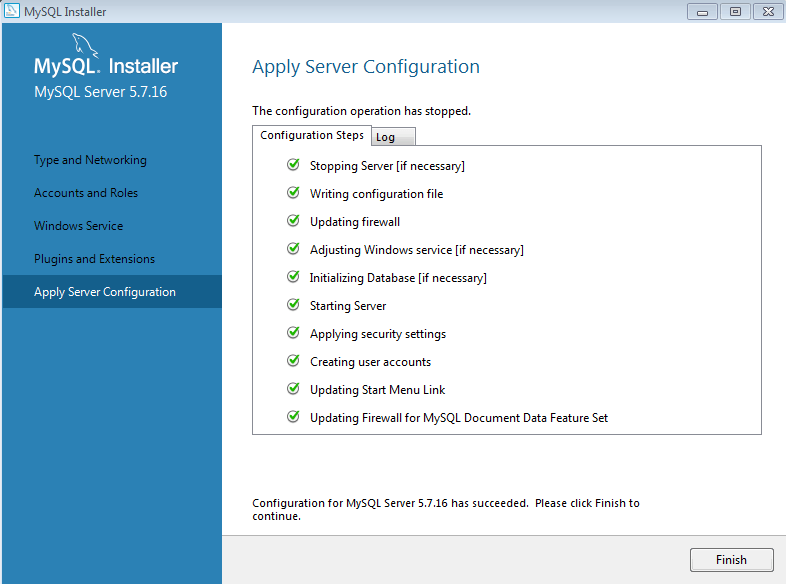
Set Root Password.

**Step-8.**



Set As Windows Service.

**Step-9.**



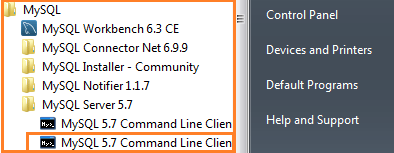
Apply Server Configuration.

**What to do After MySQL installation?**

You’ve completed the MySQL installation on your Windows system. Now, you can perform the following tasks.

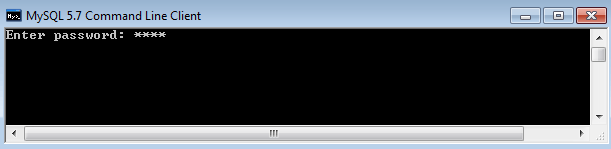
**Using MySQL CLI.**

You can access the MySQL CLI from the <All Programs> section inside the Windows start menu. See the attached screenshot and open the 2nd CLI option as shown on the screen.



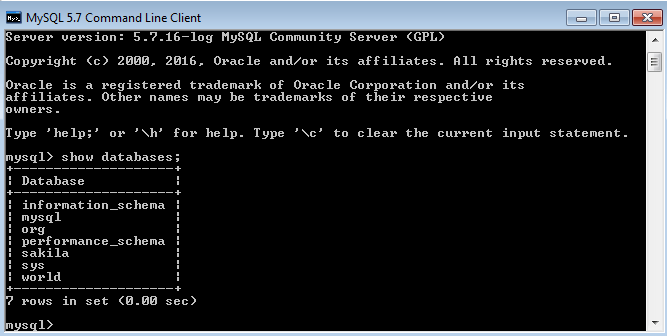
MySQL CLI – Open MySQL Command line

The CLI tool will ask for the root database password.



MySQL CLI – Enter Password**:**

Entering the correct password will open up the MySQL command prompt. Here, you can run simple SQL commands, e.g., show databases.



MySQL CLI – Show Databases

PROGRAM 2

**Q:** Introduction to SQL

* About DDL,DML,DCL,TCL
* SQL Data Types
* SQL Operators

**SQL Commands**

* SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
* SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

1. Data Definition Language (DDL)

* DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
* All the command of DDL are auto-committed that means it permanently save all the changes in the database.

List of DDL commands:

* [**CREATE**](https://www.geeksforgeeks.org/sql-create/): This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
* [**DROP**](https://www.geeksforgeeks.org/sql-drop-truncate/): This command is used to delete objects from the database.
* [**ALTER**](https://www.geeksforgeeks.org/sql-alter-add-drop-modify/)**:**This is used to alter the structure of the database.
* [**TRUNCATE**](https://www.geeksforgeeks.org/sql-drop-truncate/)**:**This is used to remove all records from a table, including all spaces allocated for the records are removed.
* [**COMMENT**](https://www.geeksforgeeks.org/sql-comments/): This is used to add comments to the data dictionary.
* [**RENAME**](https://www.geeksforgeeks.org/sql-alter-rename/)**:**This is used to rename an object existing in the database.

2. Data Manipulation Language

* DML commands are used to modify the database. It is responsible for all form of changes in the database.
* The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

* [**INSERT**](https://www.geeksforgeeks.org/sql-insert-statement/) : It is used to insert data into a table.
* [**UPDATE**](https://www.geeksforgeeks.org/sql-update-statement/)**:** It is used to update existing data within a table.
* [**DELETE**](https://www.geeksforgeeks.org/sql-delete-statement/) : It is used to delete records from a database table.
* [**LOCK:**](https://www.geeksforgeeks.org/sql-lock-table/) Table control concurrency.
* **CALL:**Call a PL/SQL or JAVA subprogram.
* **EXPLAIN PLAN:** It describes the access path to data.

3. Data Control Language

DCL commands are used to grant and take back authority from any database user.

List of  DCL commands:

* [**GRANT:**](https://www.geeksforgeeks.org/mysql-grant-revoke-privileges/)This commandgives users access privileges to the database.
* [**REVOKE:**](https://www.geeksforgeeks.org/difference-between-grant-and-revoke/)This command withdraws the user’s access privileges given by using the GRANT command.

4. Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

* [**COMMIT**](https://www.geeksforgeeks.org/sql-transactions/)**:**Commits a Transaction.
* [**ROLLBACK**](https://www.geeksforgeeks.org/sql-transactions/)**:**Rollbacks a transaction in case of any error occurs.
* [**SAVEPOINT**](https://www.geeksforgeeks.org/sql-transactions/)**:**Sets a save point within a transaction.
* [**SET TRANSACTION:**](https://www.geeksforgeeks.org/sql-transactions/)Specifies characteristics for the transaction.

## **MySQL Data Types (Version 8.0)**

Each column in a database table is required to have a name and a data type.

An SQL developer must decide what type of data that will be stored inside each column when creating a table. The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.

In MySQL there are three main data types: string, numeric, and date and time.

### String Data Types

|  |  |
| --- | --- |
| **Data type** | **Description** |
| CHAR(size) | A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1 |
| VARCHAR(size) | A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535 |
| BINARY(size) | Equal to CHAR(), but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1 |
| VARBINARY(size) | Equal to VARCHAR(), but stores binary byte strings. The size parameter specifies the maximum column length in bytes. |
| TINYBLOB | For BLOBs (Binary Large OBjects). Max length: 255 bytes |
| TINYTEXT | Holds a string with a maximum length of 255 characters |
| TEXT(size) | Holds a string with a maximum length of 65,535 bytes |
| BLOB(size) | For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data |
| MEDIUMTEXT | Holds a string with a maximum length of 16,777,215 characters |
| MEDIUMBLOB | For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data |
| LONGTEXT | Holds a string with a maximum length of 4,294,967,295 characters |
| LONGBLOB | For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data |
| ENUM(val1, val2, val3, ...) | A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them |
| SET(val1, val2, val3, ...) | A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list |

### Numeric Data Types

|  |  |
| --- | --- |
| **Data type** | **Description** |
| BIT(size) | A bit-value type. The number of bits per value is specified in size. The size parameter can hold a value from 1 to 64. The default value for size is 1. |
| TINYINT(size) | A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The size parameter specifies the maximum display width (which is 255) |
| BOOL | Zero is considered as false, nonzero values are considered as true. |
| BOOLEAN | Equal to BOOL |
| SMALLINT(size) | A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The size parameter specifies the maximum display width (which is 255) |
| MEDIUMINT(size) | A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The size parameter specifies the maximum display width (which is 255) |
| INT(size) | A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The size parameter specifies the maximum display width (which is 255) |
| INTEGER(size) | Equal to INT(size) |
| BIGINT(size) | A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The size parameter specifies the maximum display width (which is 255) |
| FLOAT(size, d) | A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions |
| FLOAT(p) | A floating point number. MySQL uses the p value to determine whether to use FLOAT or DOUBLE for the resulting data type. If p is from 0 to 24, the data type becomes FLOAT(). If p is from 25 to 53, the data type becomes DOUBLE() |
| DOUBLE(size, d) | A normal-size floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter |
| DOUBLE PRECISION(size, d) |  |
| DECIMAL(size, d) | An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. The maximum number for size is 65. The maximum number for d is 30. The default value for size is 10. The default value for d is 0. |
| DEC(size, d) | Equal to DECIMAL(size,d) |

**Note:** All the numeric data types may have an extra option: UNSIGNED or ZEROFILL. If you add the UNSIGNED option, MySQL disallows negative values for the column. If you add the ZEROFILL option, MySQL automatically also adds the UNSIGNED attribute to the column.

### Date and Time Data Types

|  |  |
| --- | --- |
| **Data type** | **Description** |
| DATE | A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31' |
| DATETIME(fsp) | A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time |
| TIMESTAMP(fsp) | A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT\_TIMESTAMP and ON UPDATE CURRENT\_TIMESTAMP in the column definition |
| TIME(fsp) | A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59' |
| YEAR | A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format. |

**SQL Operators**

Data types are used to represent the nature of the data that can be stored in the database table. For example, in a particular column of a table, if we want to store a string type of data then we will have to declare a string data type of this column.

Data types mainly classified into three categories for every database.

* String Data types
* Numeric Data types
* Date and time Data types

SQL Server offers six categories of data types for your use which are listed below −

## *Exact Numeric Data Types*

|  |  |  |
| --- | --- | --- |
| **DATA TYPE** | **FROM** | **TO** |
| bigint | -9,223,372,036,854,775,808 | 9,223,372,036,854,775,807 |
| int | -2,147,483,648 | 2,147,483,647 |
| smallint | -32,768 | 32,767 |
| tinyint | 0 | 255 |
| bit | 0 | 1 |
| decimal | -10^38 +1 | 10^38 -1 |
| numeric | -10^38 +1 | 10^38 -1 |
| money | -922,337,203,685,477.5808 | +922,337,203,685,477.5807 |
| smallmoney | -214,748.3648 | +214,748.3647 |

AD

## *Approximate Numeric Data Types*

|  |  |  |
| --- | --- | --- |
| **DATA TYPE** | **FROM** | **TO** |
| float | -1.79E + 308 | 1.79E + 308 |
| real | -3.40E + 38 | 3.40E + 38 |

## *Date and Time Data Types*

|  |  |  |
| --- | --- | --- |
| **DATA TYPE** | **FROM** | **TO** |
| datetime | Jan 1, 1753 | Dec 31, 9999 |
| smalldatetime | Jan 1, 1900 | Jun 6, 2079 |
| date | Stores a date like June 30, 1991 | |
| time | Stores a time of day like 12:30 P.M. | |

**Note** − Here, datetime has 3.33 milliseconds accuracy where as smalldatetime has 1 minute accuracy.

AD

## *Character Strings Data Types*

|  |  |
| --- | --- |
| **Sr.No.** | **DATA TYPE & Description** |
| 1 | **char**  Maximum length of 8,000 characters.( Fixed length non-Unicode characters) |
| 2 | **varchar**  Maximum of 8,000 characters.(Variable-length non-Unicode data). |
| 3 | **varchar(max)**  Maximum length of 2E + 31 characters, Variable-length non-Unicode data (SQL Server 2005 only). |
| 4 | **text**  Variable-length non-Unicode data with a maximum length of 2,147,483,647 characters. |

## *Unicode Character Strings Data Types*

|  |  |
| --- | --- |
| **Sr.No.** | **DATA TYPE & Description** |
| 1 | **nchar**  Maximum length of 4,000 characters.( Fixed length Unicode) |
| 2 | **nvarchar**  Maximum length of 4,000 characters.(Variable length Unicode) |
| 3 | **nvarchar(max)**  Maximum length of 2E + 31 characters (SQL Server 2005 only).( Variable length Unicode) |
| 4 | **ntext**  Maximum length of 1,073,741,823 characters. ( Variable length Unicode ) |

## *Binary Data Types*

|  |  |
| --- | --- |
| **Sr.No.** | **DATA TYPE & Description** |
| 1 | **binary**  Maximum length of 8,000 bytes(Fixed-length binary data ) |
| 2 | **varbinary**  Maximum length of 8,000 bytes.(Variable length binary data) |
| 3 | **varbinary(max)**  Maximum length of 2E + 31 bytes (SQL Server 2005 only). ( Variable length Binary data) |
| 4 | **image**  Maximum length of 2,147,483,647 bytes. ( Variable length Binary Data) |

## *Misc Data Types*

|  |  |
| --- | --- |
| **Sr.No.** | **DATA TYPE & Description** |
| 1 | **sql\_variant**  Stores values of various SQL Server-supported data types, except text, ntext, and timestamp. |
| 2 | **timestamp**  Stores a database-wide unique number that gets updated every time a row gets updated |
| 3 | **uniqueidentifier**  Stores a globally unique identifier (GUID) |
| 4 | **xml**  Stores XML data. You can store xml instances in a column or a variable (SQL Server 2005 only). |
| 5 | **cursor**  Reference to a cursor object |
| 6 | **table**  Stores a result set for later processing |

PROGRAM 3

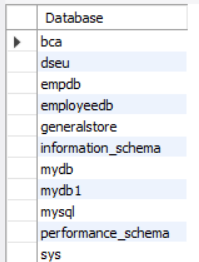
**Q:** Write queries in context of basic database related operations in MySQL

* Create database BCA in MySQL.
* Show currently selected database
* Switch from default database to my database(BCA)
* List all the tables in a database
* Show the schema of a table(e.g.: student)

1)Command:

create database BCA;

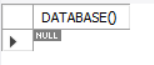
Output:



2)Command:

select DATABASE();

Output:



3)Command:

use BCA;

Output:



4)Command:

show tables;

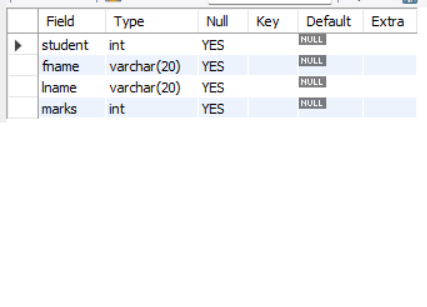
Output:



5)Command:

desc student;

Output:



PROGRAM 4

**Q:** Design the following tables in BCA database along with keys and

constraints like:

• Primary Key

• Foreign key

• NOT NULL, and

• check constraint

student(student\_id, lastname, firstname, marks)

employee(emp\_id, emp\_name, emp\_salary, dept, phone, dept\_id,

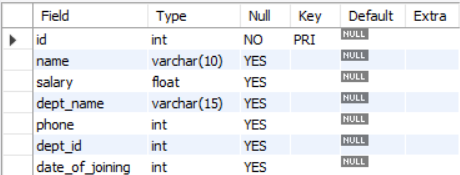
date-of-joining )

department(dept\_id, dept\_name)

1. Primary Key

Command: alter table emp modify COLUMN id int PRIMARY KEY;

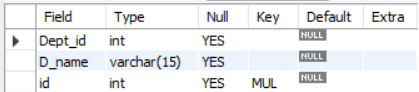
Output:



1. Foreign Key

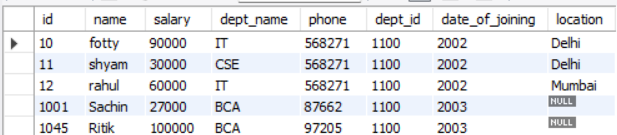
Command: alter table department ADD foreign key (id) references emp (id);

Output:



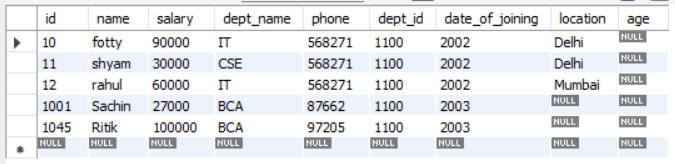
1. Not NULL

Command: select \* from emp where phone is NOT NULL;Output:



1. check constraint

Command: ALTER TABLE emp ADD age int CHECK (Age>=18);



PROGRAM 5

**Q:** Write SQL statement for implementing Alter and Drop

• ALTER – perform add, modify, rename (table, column)

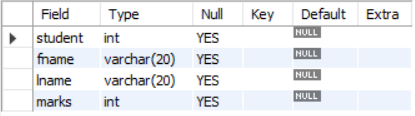
• DROP column

ALTER

1)

Command: alter table student rename to student123;

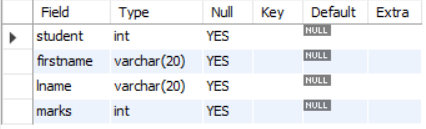
Output:



2)

Command: alter table student123 rename column fname to firstname;

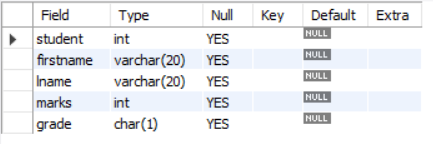
Output:



3)

Command: alter table student123 add grade char(1);

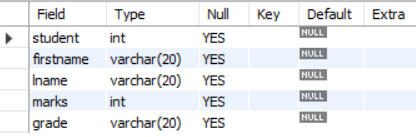
Output:



4)

Command: alter table student123 modify grade varchar(20);

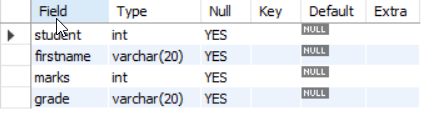
Output:



DROP

Command: alter table student123 drop column lname;

Output:



PROGRAM 6

**Q:** Write SQL queries to populate tables of BCA database and display the

Records (all tuples)

Create Table Command: create table emp (id int ,name varchar(10),salary float ,dept\_name varchar(15), phone int, dept\_id int, date\_of\_joining int);

Output:



Insert Command:

insert into emp values (1001, "Sachin", 27000, "BCA", 87662, 1100 , 2003);

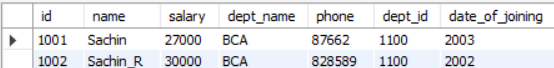
Output:



OR

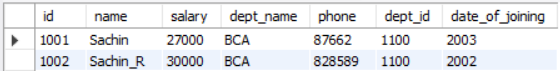
insert into emp (id,name,salary,dept\_name,phone,dept\_id,date\_of\_joining) values (1002,"Sachin\_R",30000,"BCA",828589,1100,2002);

Output:



Display Command: select \* from emp

Output:



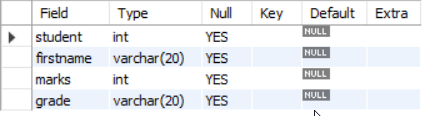
PROGRAM 7

**Q:** Write SQL query to create copy of student table(along with data), and

Drop that copy

1)Command: create table student123 as select \* from student;

Output:



2)Command: drop table student123;

Output:



PROGRAM 8

**Q:** Write SQL query to show use of Update Statements

( by create table copy\_emp as select \* from employee)

• update copy\_emp set salary=salary+1000;

• update copy\_emp set dept\_name=”electrical” where emp\_name =

”james”;

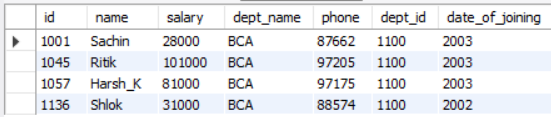
• update copy\_emp set phone=123456, dept\_id=20 where emp\_id =

10;

• select \* from copy\_emp;

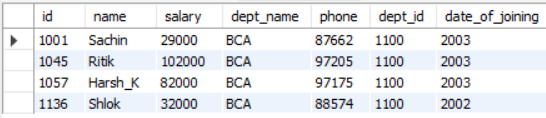
Create Copy Table:-





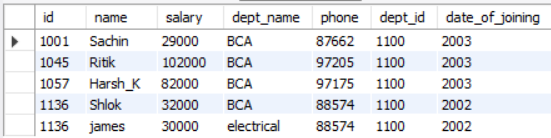
1. Command: update copy\_emp set salary=salary+1000;

Output:



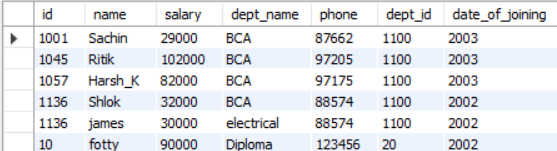
1. Command: copy\_emp set dept\_name=”electrical” where emp\_name=”james”;

Output:



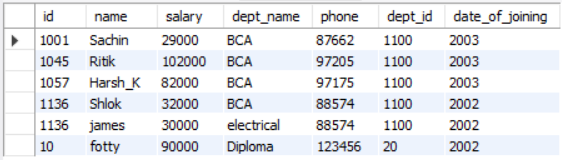
1. Command: update copy\_emp set phone=123456, dept\_id=20 where emp\_id=10;

Output:



1. Command: select \* from copy\_emp;

Output:



PROGRAM 9

**Q:** Write SQL query to show difference between truncate and drop

command.

1. Command: truncate table copy\_emp;

Output:



1. Command: drop table copy\_emp;

Output:



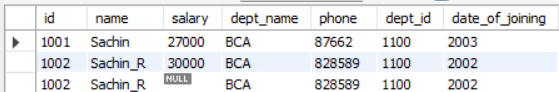
PROGRAM 10

**Q:** Write SQL query to select only DISTINCT records from a table

containing duplicate tuples.

Command: select distinct \* from emp;

Output:



**OR**

Command: select distinct id,name from emp;

Output:



PROGRAM 11

**Q:** Write SQL query to implement all aggregate functions: MAX(),

MIN(), AVG(), COUNT(), SUM()

1. min

Command:

select min(salary) as salary\_min from emp ;

Output:



1. max

Command: select max(salary) as salary\_max from emp ;

Output:



1. sum

Command: select sum(salary) as salary\_sum from emp ;

Output:



1. avg

Command: select avg(salary) as salary\_avg from emp ;

Output:



1. count

Command: select count(salary) as salary\_count from emp;

Output:



PROGRAM 12

**Q:** Perform the following operations:

• Delete record of an employee according to his emp\_id

• Select sum of total salary of all employees from employee table;

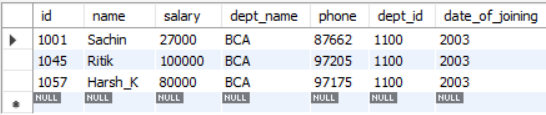
• Count total names of employees from employee table;

• Perform insert, update and Delete on the data from a base table

(that is referred by a foreign key from another table).

1)Command: delete from emp where id=1136;

Output:



2) Command: select sum(salary) as salary\_sum from emp ;

Output:



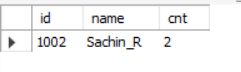
3) Command: select count(\*) from emp;

Output:



Command: select id, name, count(\*) as cnt from emp group by id, name having count(\*)>1;

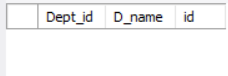
Output:



4) Command: delete from department where id=1057;

delete from emp where id=1057;

Output:

PROGRAM 13

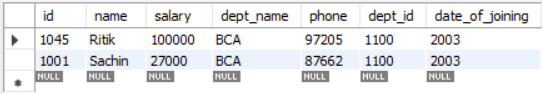
**Q:** Write SQL queries to show the use of:

• Order by

• Group by, having

Command: select \* from emp order by name asc;

Output:

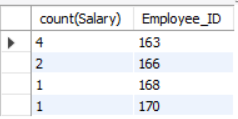


The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate function

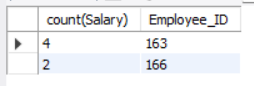
Command: SELECT count(Salary),Employee\_ID FROM emp GROUP BY Salary;

Output:



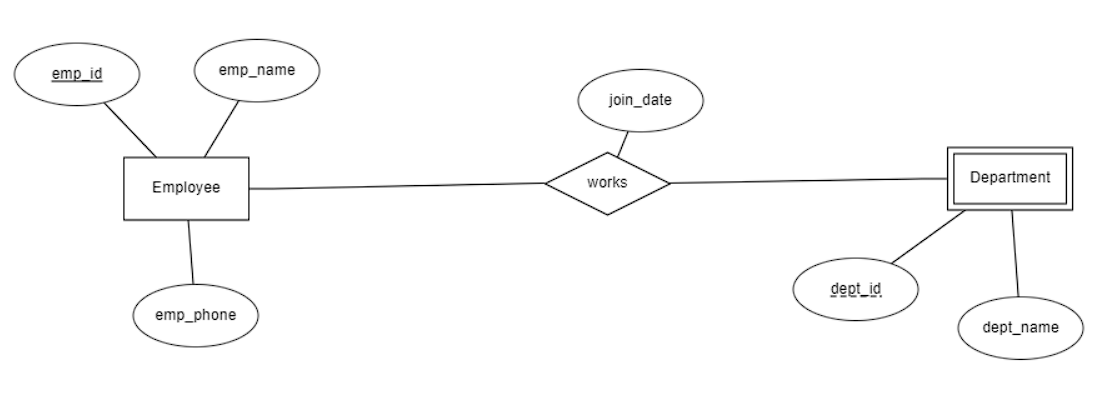
Command: SELECT count(Salary),Employee\_ID FROM emp GROUP BY Salary HAVING count(Salary)>1;

Output:



PROGRAM 14

**Q:** Draw ER Diagram of employee works in department in ERDplus.com



PROGRAM 15

**Q:** Write SQL query to show the use of following operators:

• AND, OR, NOT

• Use of Null (IS NULL, and IS NOT NULL)

• LIKE (with wildcards such as % and \_)

• IN, NOT IN

• BETWEEN

• ANY

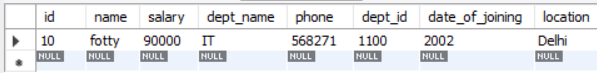
• ALL

• TOP/LIMIT clause

1)AND, OR, NOT

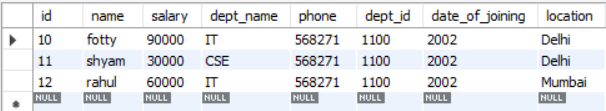
Command: select \* from emp where dept\_name = "IT" AND location = "Delhi";

Output:



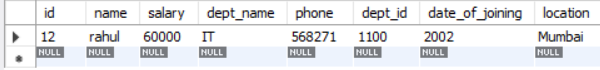
Command: select \* from emp where dept\_name = "IT" OR location = "Delhi";

Output:



Command: select \* from emp where dept\_name = "IT" NOT location = "Delhi";

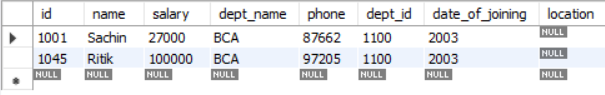
Output:



2)Use of NULL (IS NULL, IS NOT NULL)

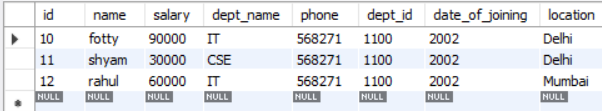
Command: select \* from emp where location IS NULL;

Output:



Command: select \* from emp where location IS NOT NULL;

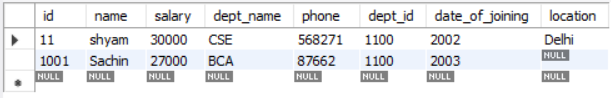
Output:



3)Use of LIKE operators with wildcards (\_,%)

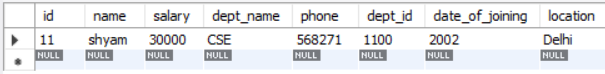
Command: select \* from emp where name LIKE 's%';

Output:



Command: select \* from emp where name LIKE '\_h%';

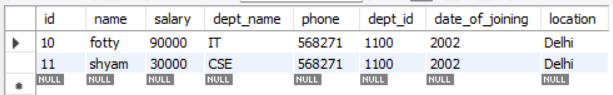
Output:



4) Use of IN, NOT IN operator

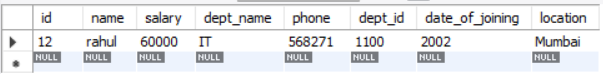
Command: select \* from emp where location IN ("delhi");

Output:



Command: select \* from emp where location NOT IN ("delhi");

Output:



5)BETWEEN

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

Command: SELECT \* FROM emp WHERE Salary BETWEEN 30000 AND 50000;

Output:

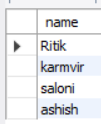


6)ANY

ANY means that the condition will be true if the operation is true for any of the values in the range.

Command: SELECT name FROM student WHERE roll\_no = ANY(SELECT roll\_no FROM course WHERE age=19);

Output:

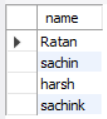


7)ALL

ALL means that the condition will be true only if the operation is true for all values in the range.

Command: SELECT name FROM student WHERE roll\_no = ALL(SELECT roll\_no FROM course WHERE age=19);

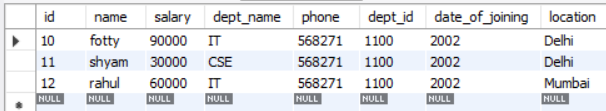
Output:



8) Use of TOP/LIMIT clause

Command: select \* from emp LIMIT 3;

Output:

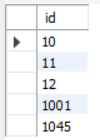


PROGRAM 16

**Q:** Write SQL query to display the result of UNION operator

Command: SELECT id FROM emp UNION SELECT D\_name FROM Department;

Output:



PROGRAM 17

**Q:** Write the queries to implement the following Joins:

➢ Inner join

➢ Left join

➢ Right join

➢ Full join, on the given course and student tables

Command:

SELECT course.course\_id, student.name, student.phone\_no FROM course INNER JOIN student ON course.course\_id = student.roll\_no;

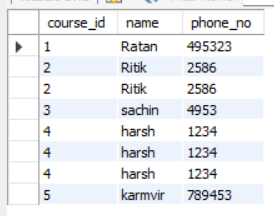
Output(Inner Join):



Command(Left Join):

SELECT course.course\_id, student.name, student.phone\_no FROM course LEFT JOIN student ON course.course\_id = student.roll\_no;

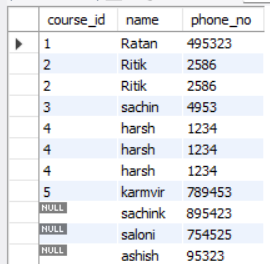
Output:



Command(Right Join):

SELECT course.course\_id, student.name, student.phone\_no FROM course RIGHT JOIN student ON course.course\_id = student.roll\_no;

Output:



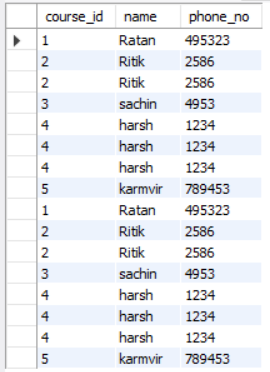
Command(FULL JOIN):

SELECT course.course\_id, student.name, student.phone\_no FROM course LEFT JOIN student ON course.course\_id = student.roll\_no

UNION ALL

SELECT course.course\_id, student.name, student.phone\_no FROM course RIGHT JOIN student ON course.course\_id = student.roll\_no;

Output:



PROGRAM 18

**Q:** For the following database schema for a customer - sale scenario of a

general store, perform the given queries:

Customer(cust id:integer, cust\_name: string)

Item(item\_id: integer, item\_name: string, price: float)

Sale(bill\_no:integer, bill\_date:date, cust\_id:integer, item\_id: integer,

qty\_sold: integer)

1) Create the tables with the appropriate integrity constraints

2) Insert around 10 records in each of the tables

3) List the total Bill details with the quantity sold, price of the item

and the final amount

4) List the details of the customer who have bought a product which

has a price>200

5) Give a count of how many products have been bought by each

customer

6) Give a list of products bought by a customer having cust\_id as 5

7) List all the bills for the current date with the customer names and

item number

1) Create the tables with the appropriate integrity constraints

Command:

create database GeneralStore;

use GeneralStore;

create table Customer

(

Cust\_id int Primary key,

Cust\_name varchar(30)

);

create table Item

(

Item\_id int Primary key,

Item\_name varchar(30),

Price float

);

create table Sale

(

bill\_no int Primary key,

bill\_date date,

cust\_id int,

item\_id int,

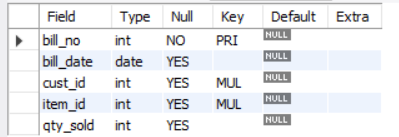
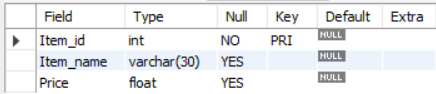
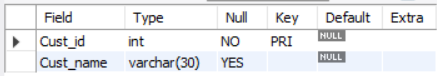
qty\_sold int,

foreign key(cust\_id) references customer(cust\_id),

foreign key(item\_id) references item(item\_id)

);

Output:

2) Insert around 10 records in each of the tables

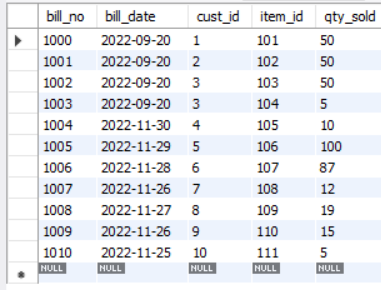
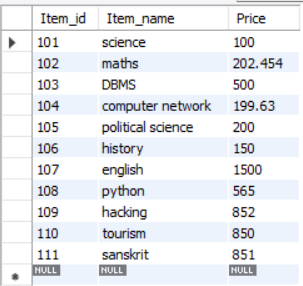
Command:

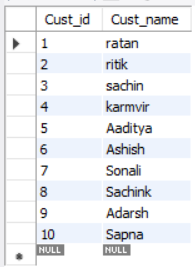
insert into sale values (1000,'2022-09-20',1,101,50),(1001,'2022-09-20',2,102,50),(1002,'2022-09-20',3,103,50),(1003,'2022-09-20',3,104,5),(1004,'2022-11-30',4,105,10),(1005,'2022-11-29',5,106,100),(1006,'2022-11-28',6,107,87),(1007,'2022-11-26',7,108,12),(1008,'2022-11-27',8,109,19),(1009,'2022-11-26',9,110,15),(1010,'2022-11-25',10,111,5);

insert into item values (101,"science",100),(102,"maths",202.454),(103,"DBMS",500),(104,"computer network",199.63),(105,"political science",200),(106,"history",150),(107,"english",1500),(108,"python",565),(109,"hacking",852),(110,"tourism",850),(111,"sanskrit",851);

insert into customer values (1,"ratan"),(2,"ritik"),(3,"sachin"),(4,"karmvir"),(5,"Aaditya"),(6,"Ashish"),(7,"Sonali"),(8,"Sachink"),(9,"Adarsh"),(10,"Sapna");

Output:

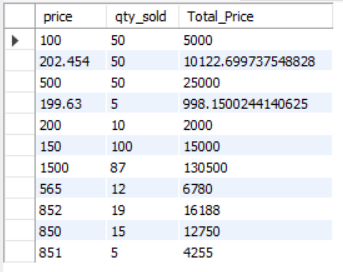


3) List the total Bill details with the quantity sold, price of the item and the final amount

Command:

SELECT price, qty\_sold, (price\*qty\_sold) Total\_Price FROM item, sale WHERE item.Item\_id=sale.Item\_id;

Output:

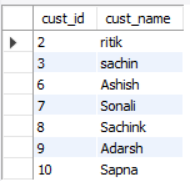


4) List the details of the customer who have bought a product which has a price>200

Command:

SELECT customer.cust\_id, Customer.cust\_name FROM Customer, Item, sale WHERE Item.Price>200 AND customer.cust\_id=sale.cust\_id AND Item.Item\_id=sale.item\_id;

Output:

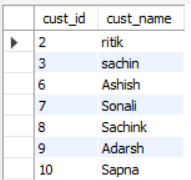


5) Give a count of how many products have been bought by each customer

Command:

SELECT cust\_id, count(Item\_id) FROM sale GROUP BY cust\_id;

Output:

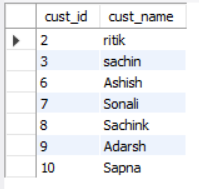


6) Give a list of products bought by a customer having cust\_id as 5

Command:

SELECT i.Item\_Name FROM item i, sale s WHERE s.cust\_id=5 AND i.item\_id-s.item\_id;

Output:



7) List all the bills for the current date with the customer names and item numbers

Command:

SELECT Item.item\_id,Item.item\_name FROM item, sale WHERE Item.item\_id=sale.item\_id AND sale.bill\_date=curdate();

Output:



PROGRAM 19

**Q:** Demonstrate the Application of auto increment on any of the above table in MySQL

Command:

CREATE TABLE Persons (

Personid int NOT NULL AUTO\_INCREMENT PRIMARY KEY,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int

);

INSERT INTO Persons (FirstName,LastName)

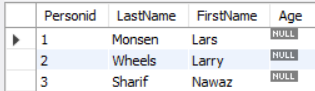
VALUES ('Lars','Monsen');

INSERT INTO Persons (FirstName,LastName)

VALUES ('Larry','Wheels'),('Nawaz','Sharif');

SELECT \* FROM Persons;

Output:



PROGRAM 20

**Q:** Execute following functions for strings , numeric functions & date functions in MySQL

Strings Command:

SELECT CHAR\_LENGTH("SQL Lab");

SELECT CONCAT("SQL ","Lab ","is ","fun ");

SELECT UCASE("sql lab");

SELECT LCASE("SQL LAB");

SELECT RTRIM("SQL Lab ");

SELECT LTRIM(" SQL Lab");

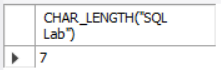
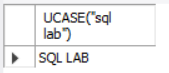
SELECT REVERSE("SQL Lab");

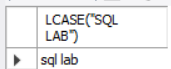
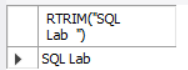
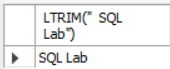
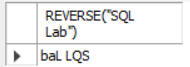
SELECT SUBSTR("SQL Lab",3);

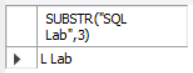
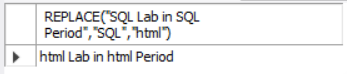
SELECT STRCMP("SQL","sql");

SELECT REPLACE("SQL Lab in SQL Period","SQL","html");

Output:

Numeric Functions Command:

SELECT ABS(-245.563);

SELECT CEIL(2.7);

SELECT FLOOR(2.3);

SELECT GREATEST(3,2,4,5,6);

SELECT LEAST(3,2,4,5,6);

SELECT LOG(2);

SELECT LOG10(2);

SELECT 12.1 DIV 3;

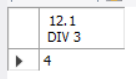
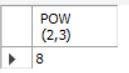
SELECT 18 MOD 4;

SELECT POW (2,3);

SELECT SQRT(9);

SELECT TRUNCATE(3.144444,3);

Output:

Date functions Command: 

SELECT DATEADD(day, 10, CAST(GETDATE() AS DATE)) As DateAdd;

SELECT GETDATE() 'Now', DATEADD(ss,2,GETDATE()) 'Now + 2 seconds';

SELECT CURRENT\_DATE();

SELECT DATEDIFF(day, '2023/01/01', '2023/01/11') AS NoOfDays;

SELECT GETDATE();

SELECT DAY('2023/01/11 09:08') AS DayOfMonth;

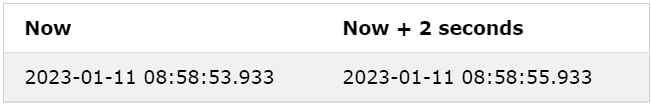
SELECT EXTRACT(WEEK FROM "2023-01-11");

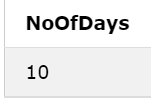
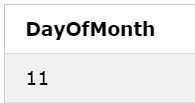
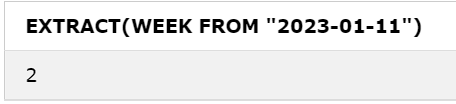
SELECT sysdate();

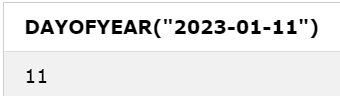
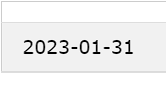
SELECT DAYOFYEAR("2023-01-11");

SELECT CAST(eomonth(GETDATE()) AS date);

Output:



PROGRAM 21

**Q:** • Write SQL query to create a view of student table

• Write SQL query to create and update views

o from single table

o from multiple tables

i) from single table

Command:

CREATE VIEW viewDept\_id AS

SELECT Dept\_id

FROM department

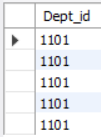
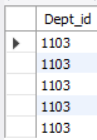
WHERE Dept\_id = 1100;

UPDATE viewDept\_id

SET Dept\_id = 1103

WHERE Dept\_id = 1101;

Output:

ii) from multiple tables

Command:

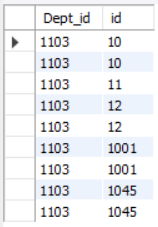
CREATE VIEW Deptid\_Empid AS

SELECT department.Dept\_id, emp.id

FROM department,emp

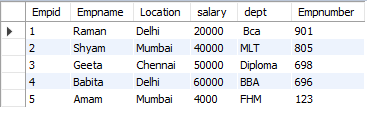
WHERE department.D\_name = emp.dept\_name;

Output:



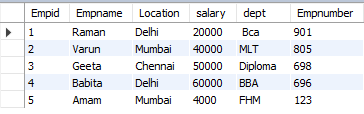
PROGRAM 22

**Q:** Create employee and department table and provide queries to perform all basic DBMS operations as given in the task list (16 different queries in total).



1. Change the name of employee with empno-805.

update employee set Empname="Varun" where Empnumber=805;



1. Display the record of an employee with maximum salary.

SELECT \*FROM employee HAVING max(salary);



1. Count total no of records from employee table.

SELECT COUNT(Empid) FROM employee;



1. Select all the records of an employee whose department location is "Delhi".

select \*from employee where location ="Delhi";



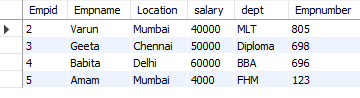
1. Write a query to get the records from emp table where location= "Delhi" and department is "BBA".

select \* from employee where location= "Delhi" and dept="BBA";



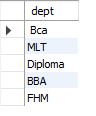
1. Write a query to get the records from emp table where deptno is 10 or salary is greater than 2000.

select \* from employee where empnumber=123 or salary>20000 ;



1. Write SQL query to select only distinct dept from emp table.

SELECT DISTINCT dept FROM employee;



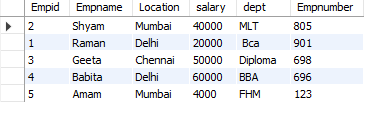
1. Write SQL query to display the Empname starting with s.a or m in table employee.

SELECT \* FROM employee WHERE empname LIKE 's%' ;



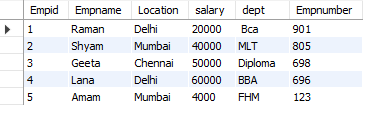
1. Write SQL query to display name of employees in descending order.

SELECT \* from employee order by Empname Desc ;



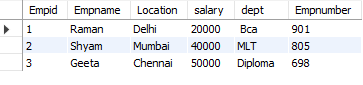
1. Write SQL query to change the name of Babite to Lana in employee table.

update employee set Empname="Lana" where Empname="Babita";



1. Write SQL query to display only top 3 rows from employee table.

select \* from employee limit 3;



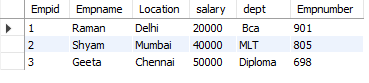
1. Write SQL query to select all the employees name whose deptno is 10 and 20(using IN operator)

SELECT \* FROM employee WHERE empnumber IN ('805', '901');



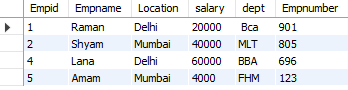
1. Write SQL query to select the records from emp table whose salary is in the range of 1500 to 3000(using BETWEEN operator)

SELECT \* FROM employee WHERE salary BETWEEN 20000 AND 50000;



1. Delete the record of an employee from emp table whose name is 'geeta'.

DELETE FROM employee WHERE empname="geeta";

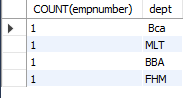


1. Apply all 4 joins on employee table and dept table(inner, left, right, outer) and show the resultant tables

***Joins Already done in practical 17***

1. Write SQL query to list the no of employees in each department(use Group By)

SELECT COUNT(empnumber),dept FROM employee GROUP BY dept;



PROGRAM 23

**Q: For a given database with following attributes:**

**EMPLOYEE ID, FIRST NAME, LAST NAME EMAIL PHONE NUMBER, HIRE\_DATE, JOB\_ID, SALARY, MANAGER\_ID, DEPARTMENT\_ID**

****

1)write a SQL query to find those employees whose salary matches the lowest salary of any of the departments. Return first name, last name and department ID.

Command:

select First\_name,Last\_name,Department\_ID from emp where salary=(select min(Salary) from emp) group by Department\_ID;

Output:

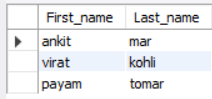


2)write a SQL query to find those employees who receive a higher salary than the employee with ID 163. Return first name, last name.

Command:

select First\_name,Last\_name from emp where salary > (select salary from emp where Employee\_id=163);

Output:



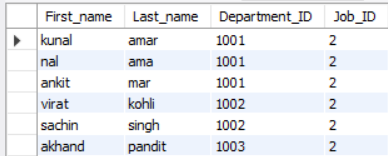
3) write a SQL query to find out which employees have the same designation as the

employee whose ID is 169. Return first name, last name, department ID and job ID.

Command:

select First\_name,Last\_name,Department\_ID,Job\_ID from emp where Job\_ID=(select Job\_ID from emp where Employee\_ID=169);

Output:



4) write a SQL query to find those employees who report to that manager whose first name

is 'Payam'. Return first name, last name, employee ID and salary.

Command:

select First\_name,Last\_name,Employee\_ID,Salary from emp where First\_name=(select First\_name from emp where Manager\_ID=31 && First\_name='payam');

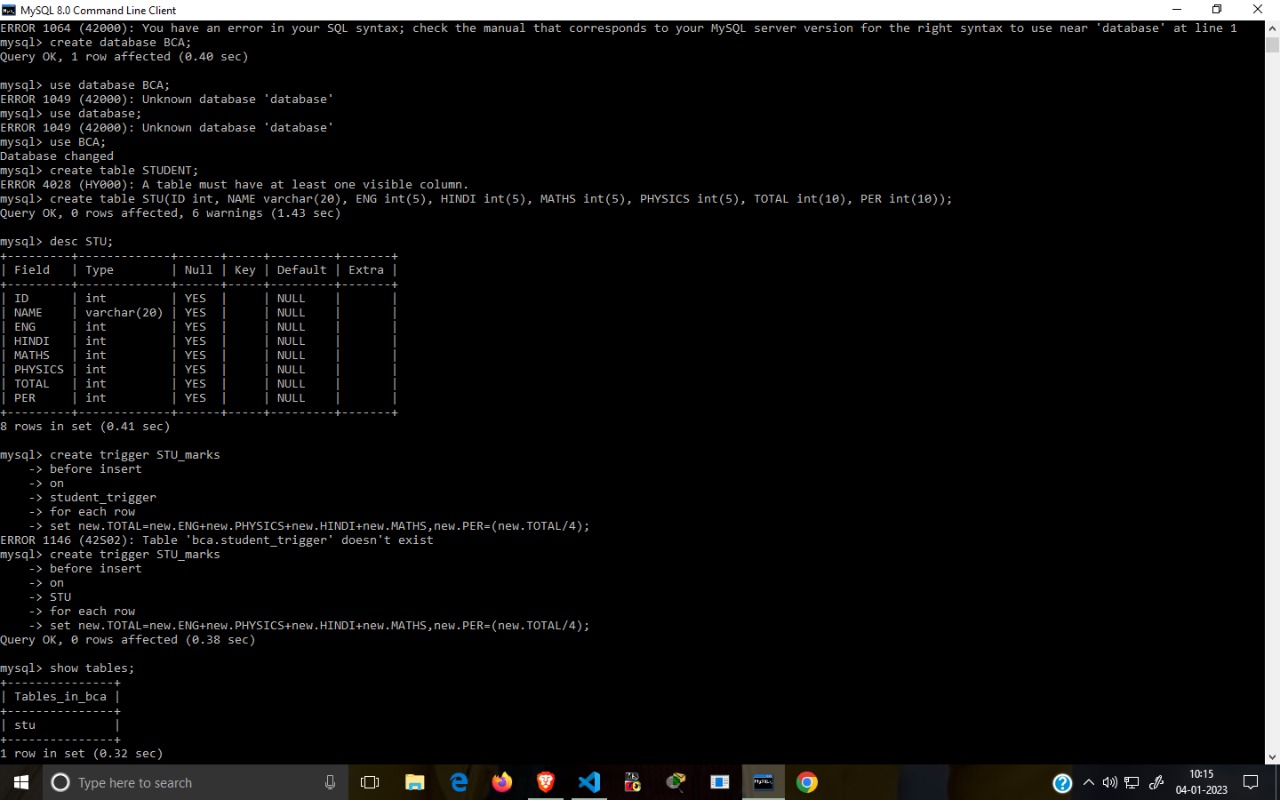
Output:

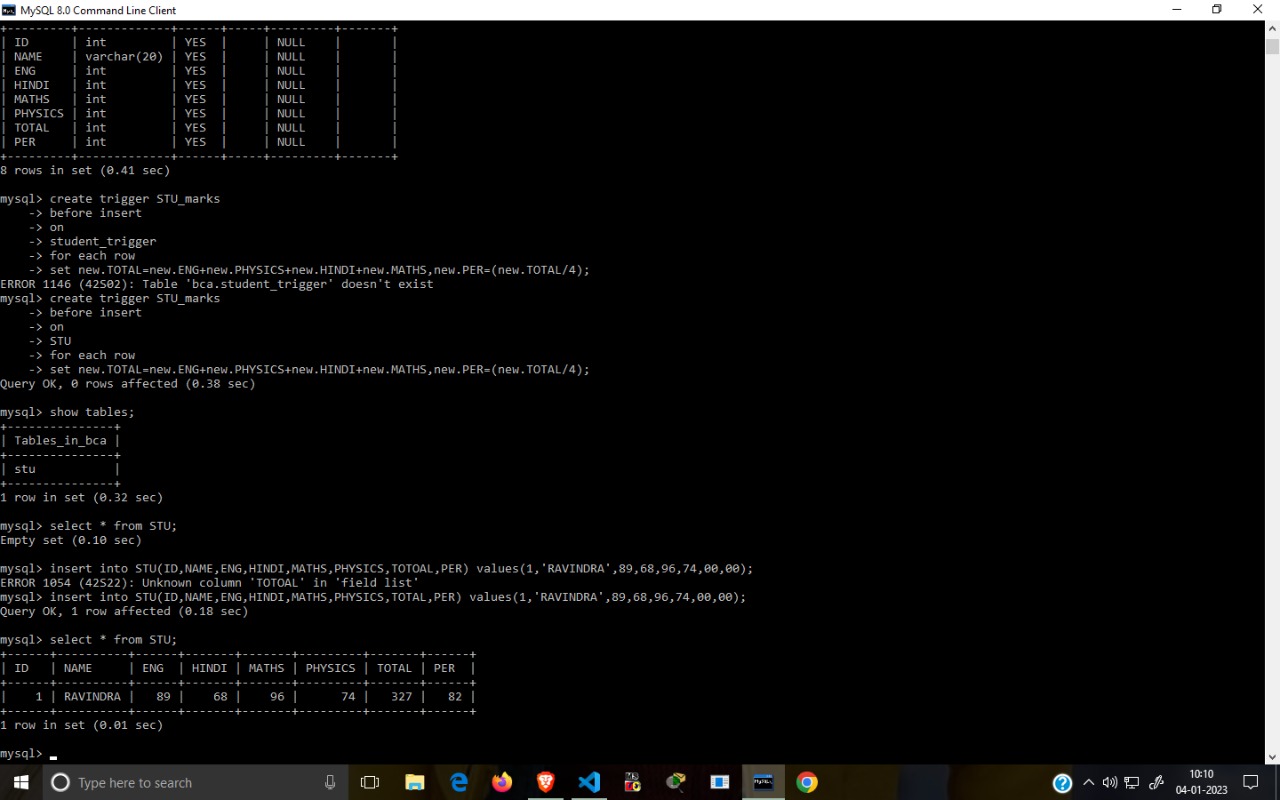


PROGRAM 24

**Q: Create a Trigger**

A trigger is **a special type of stored procedure that automatically runs when an event occurs in the database server**. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.



****

PROGRAM 25

**Q: Create a procedure and Call it**

Command:

**DECLARE**

**a number;**

**b number;**

**c number;**

**PROCEDURE findMin(x IN number, y IN number, z OUT number) IS**

**BEGIN**

**IF x < y THEN**

**z:= x;**

**ELSE**

**z:= y;**

**END IF;**

**END;**

**BEGIN**

**a:= 23;**

**b:= 45;**

**findMin(a, b, c);**

**dbms\_output.put\_line(' Minimum of (23, 45) : ' || c);**

**END;**

Output:



PROGRAM 26

**Q: Create a function and Call it**

1)using if-else loop-

Command:

**DECLARE**

**num number;**

**factorial number;**

**FUNCTION fact(x number)**

**RETURN number**

**IS**

**f number;**

**BEGIN**

**IF x=0 THEN**

**f := 1;**

**ELSE**

**f := x \* fact(x-1);**

**END IF;**

**RETURN f;**

**END;**

**BEGIN**

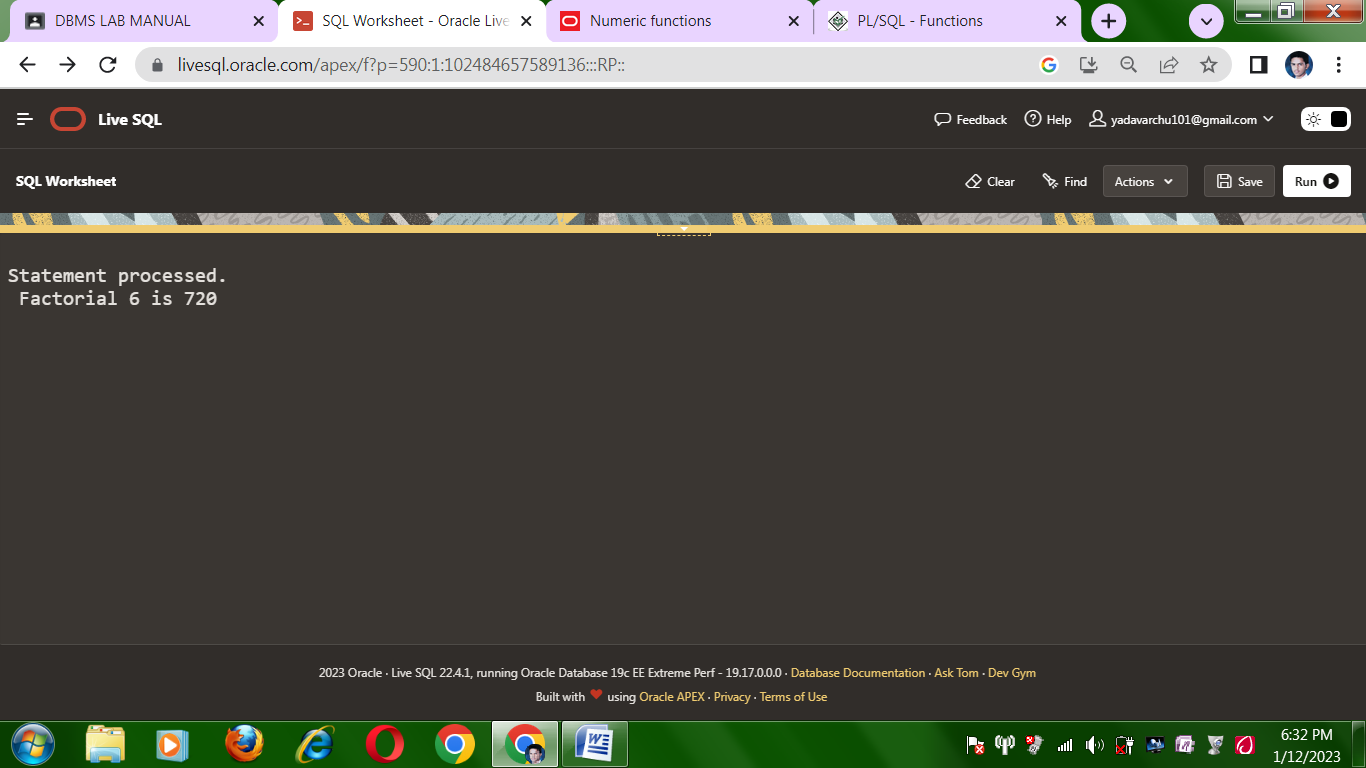
**num:= 6;**

**factorial := fact(num);**

**dbms\_output.put\_line(' Factorial '|| num || ' is ' || factorial);**

**END;**

Output:



2)using while loop

**declare**

**n integer:=1;**

**begin**

**while n<=5**

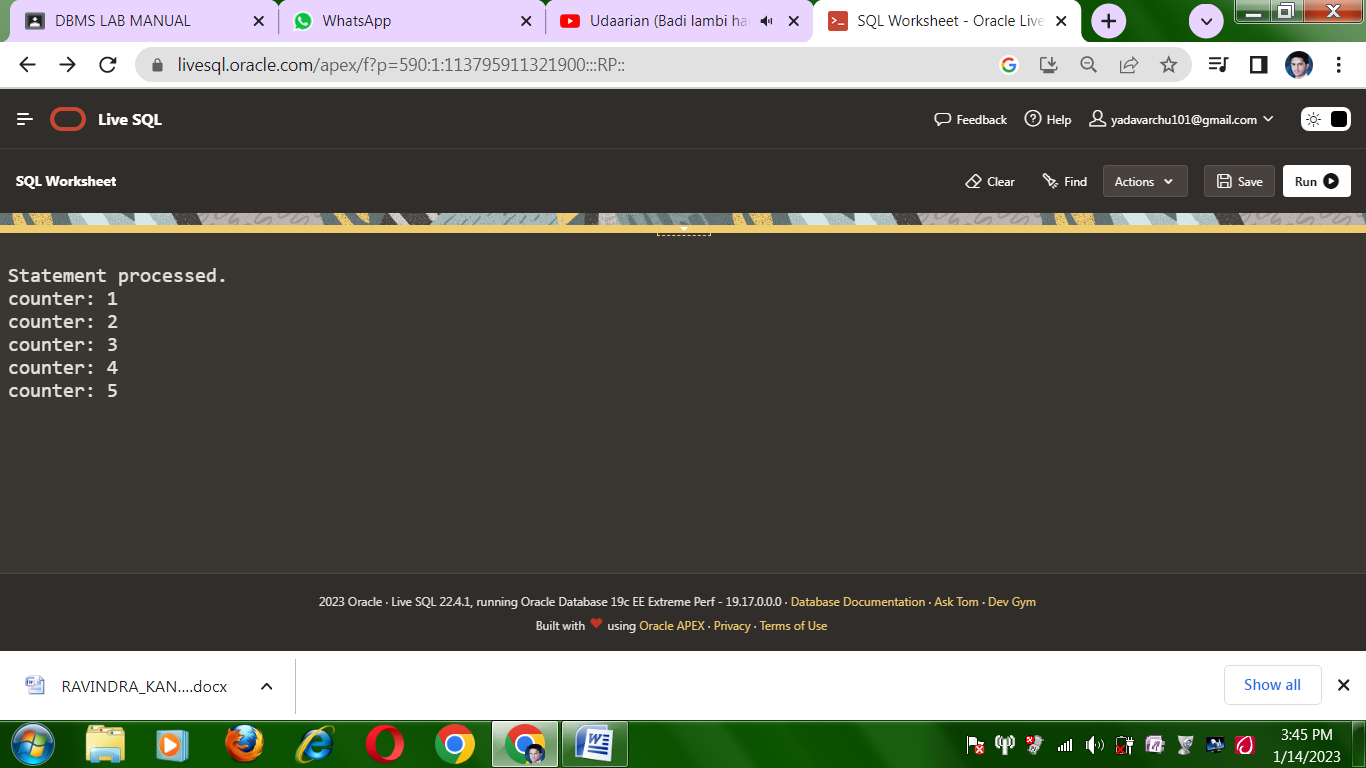
**loop**

**dbms\_output.put\_line('counter: '|| n);**

**n:=n+1;**

**end loop;**

**end;**



3)using for loop-

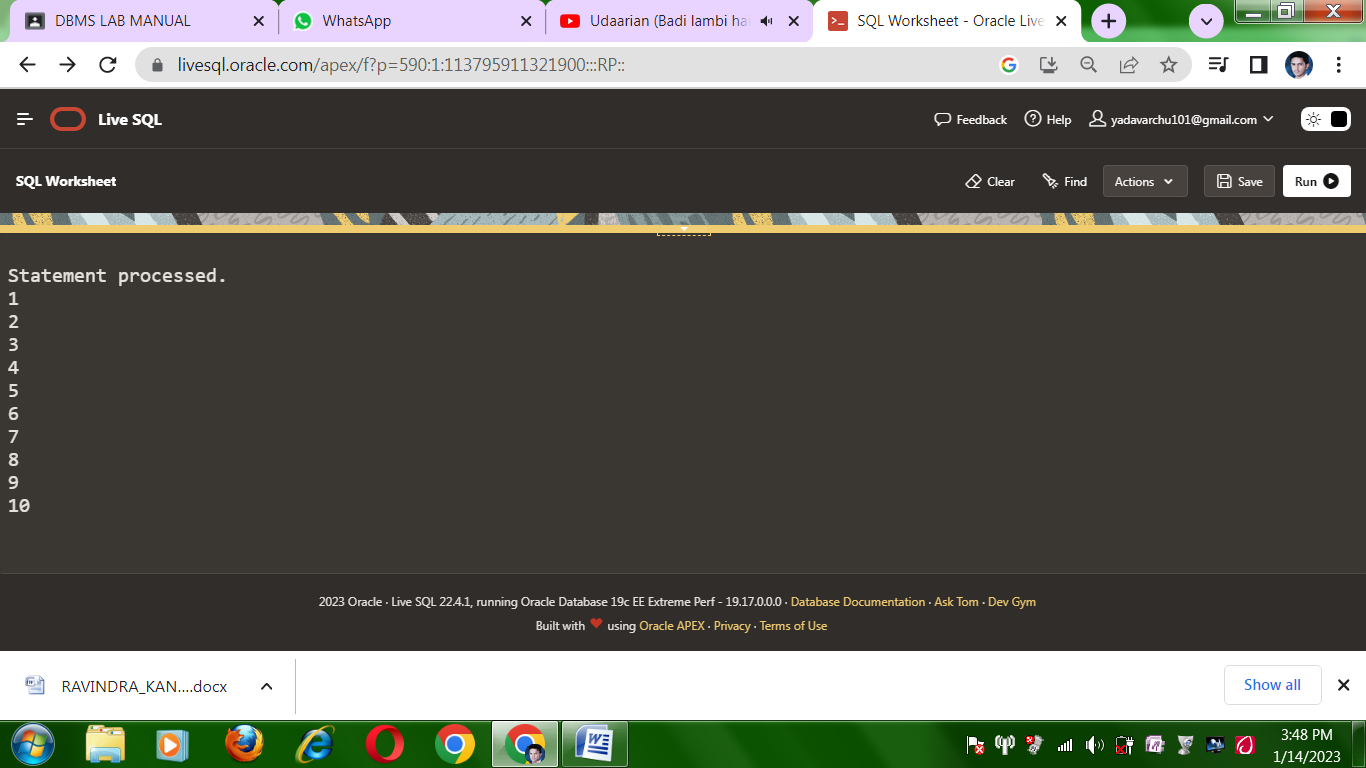
**begin**

**for i in 1..10 loop**

**dbms\_output.put\_line(i);**

**end loop;**

**end;**

****