SRM Institute of Science and Technology
NCR Campus, Modinagar, Ghaziabad

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

DATABASE MANAGEMENT SYSTEMS (18CSC303J)

NAME	**:**	**Vatsal Goel**
REG. NO.	**:**	**RA1911028030012**
DEGREE/BRANCH/SECTION	**:**	**B.TECH/CSE/C**
YEAR/SEMESTER	**:**	**3rd/6th**
SESSION	**:**	**JAN- 2022 – MAY 2022**

SRM Institute of Science and Technology
NCR Campus, Modinagar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Reg. No.	R	A	1	9	1	1	0	2	8	0	3	0	0	1	2

BONAFIDE CERTIFICATE

It is to be certified that the bonafide practical record submitted by **Vatsal Goel** of 6th Semester/ 3rd Year for Bachelor of Technology degree in the Department of Computer Science and Engineering, NCR Campus, SRM Institute of Science and Technology, has been done for the course Database Management Systems(18CSC303J), during the academic year session JAN- 2022 – MAY 2022.

Head of the Department **Lab In-charge**

DR. R. P. MAHAPATRA(HOD)
Professor
Department of CSE

DR.ANNA ALPHY
Assistant Professor
Department of CSE

Submitted for the University Examination held on _____.

Examiner 1

Examiner 2

INDEX

Exp. No.	Name of Experiment	Page No.	Date	Signature
1.	SQL Data Definition Language Commands on sample exercise			
2.	SQL Data Manipulation Language Commands			
3.	SQL Data Control Language Commands and Transaction control commands to the sample exercises			
4.	Inbuilt functions in SQL on sample			
5.	Construct a ER Model for the application to be constructed to a database			
6.	Nested Queries on sample exercise			
7.	Join Queries on sample exercise			
8.	Set operators & Views			
9.	PL/SQL Conditional and Iterative Statements			
10.	PL/SQL Procedures on sample exercise			
11.	PL/SQL Functions			
12.	PL/SQL Cursors			
13.	PL/SQL Exception Handling			
14.	PL/SQL Trigger			
15.	Frame and exercise the appropriate PL/SQL Cursors and Exceptional Handling for the project			

Exp No:-1 Date:-

Experiment:-DDL Commands

Aim:- SQL data definition:-

Create command
Alter command
Drop
Truncate

Theory:-

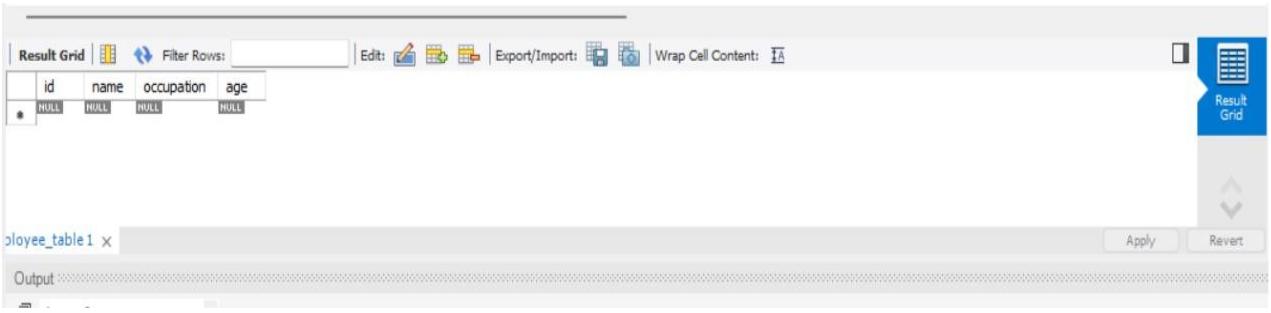
- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

a. **CREATE** It is used to create a new table in the database.

QUERY:-

```
CREATE TABLE employee_table(
   id int NOT NULL AUTO_INCREMENT,
   name varchar(45) NOT NULL,
   occupation varchar(35) NOT NULL,
   age int NOT NULL,
   PRIMARY KEY (id)
);
```

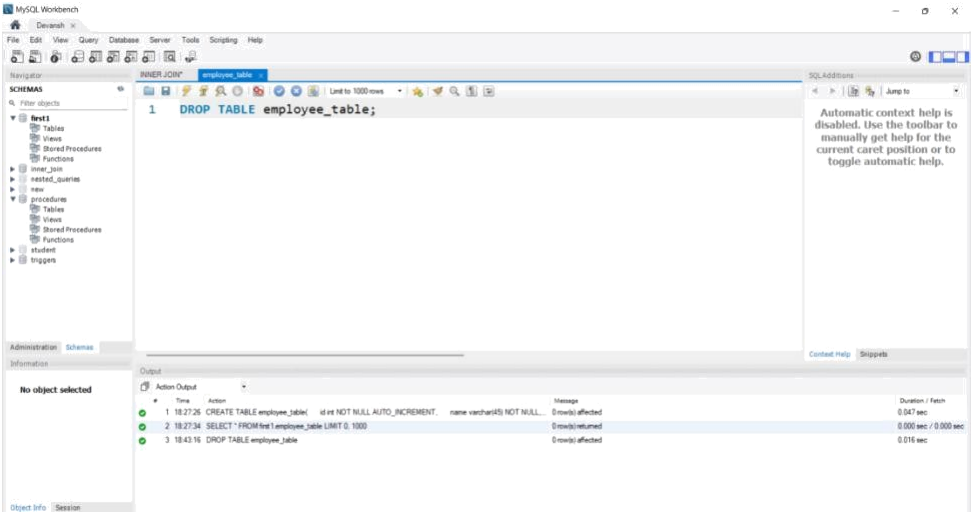
OUTPUT:-



B .DROP: It is used to delete both the structure and record stored in the table.

QUERY:-

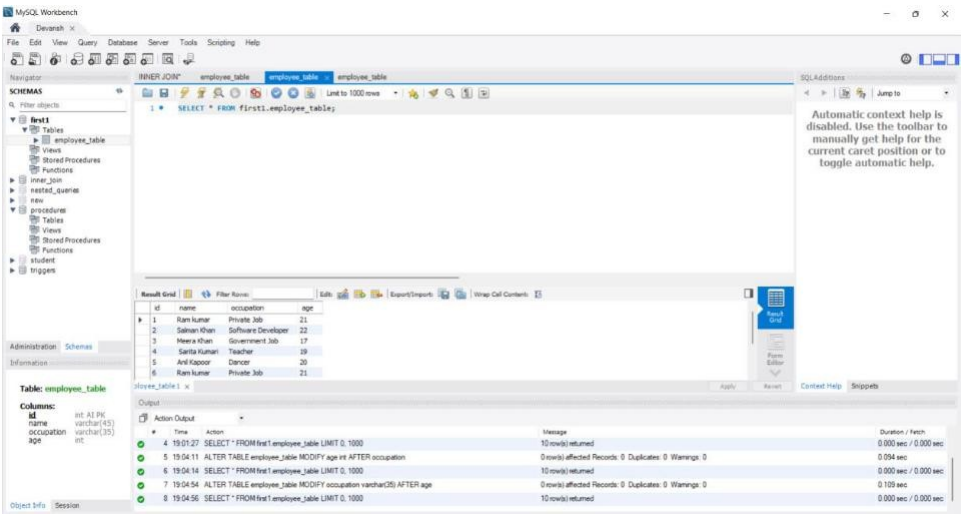
```
DROP TABLE employee_table;
```

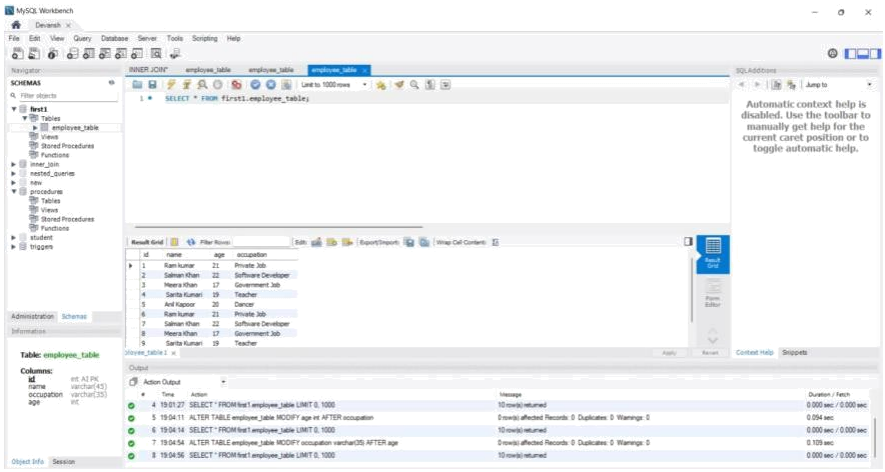


c. ALTER : It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

QUERY:-

```
ALTER TABLE employee_table
MODIFY occupation varchar(35)
AFTER age;
```

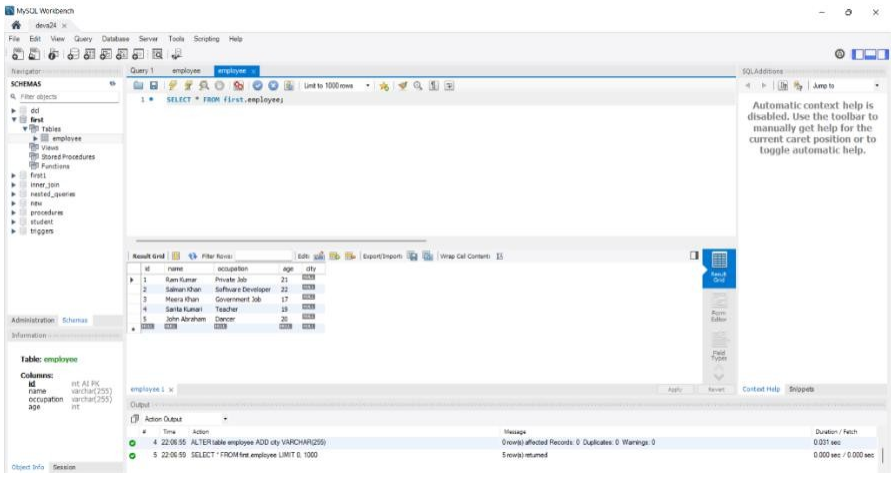




ALTER TABLE-ADD COLUMN

Query:-

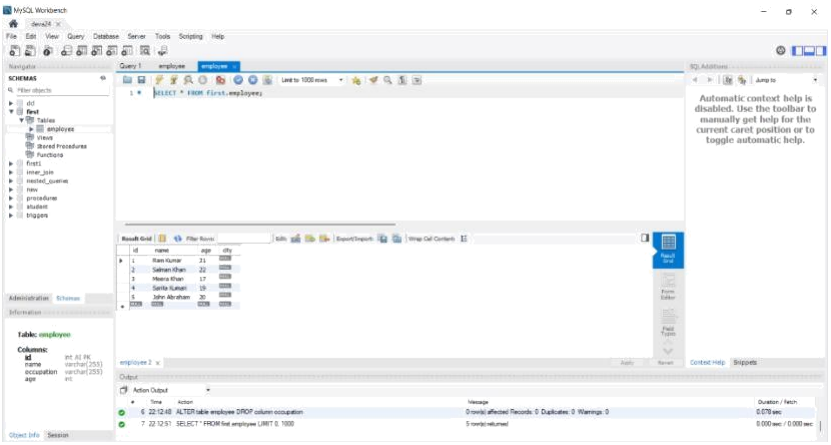
ALTER table employee
ADD city VARCHAR(255);



ALTER TABLE-DROP COLUMN:-

Query:-

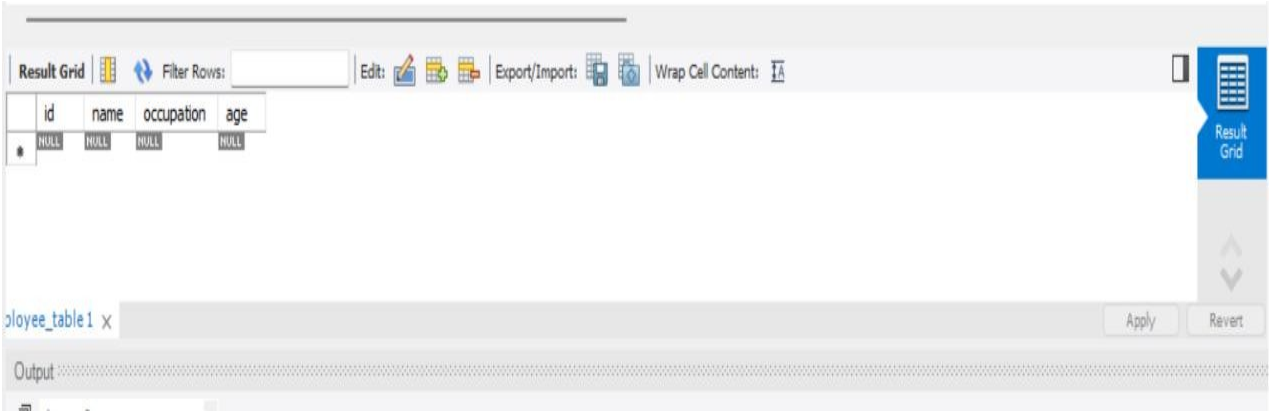
ALTER table employee
DROP column occupation;



d. TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

QUERY:-

TRUNCATE TABLE employee_table;



Exp No:-2 Date:-

EXPERIMENT:-DML Commands

Aim:- Execute DML Commands

Theory:- DML commands are the most frequently used SQL commands and is used to query and manipulate the existing database objects. Some of the commands are Insert, Select, Update, Delete.

Insert Command: This is used to add one or more rows to a table. The values are separated by commas and the data types char and date are enclosed in apostrophes. The values must be entered in the same order as they are defined.

Select Command: It is used to retrieve information from the table. It is generally referred to as querying the table. We can either display all columns in a table or only specify column from the table.

Update Command: It is used to alter the column values in a table. A single column may be updated or more than one column could be updated.

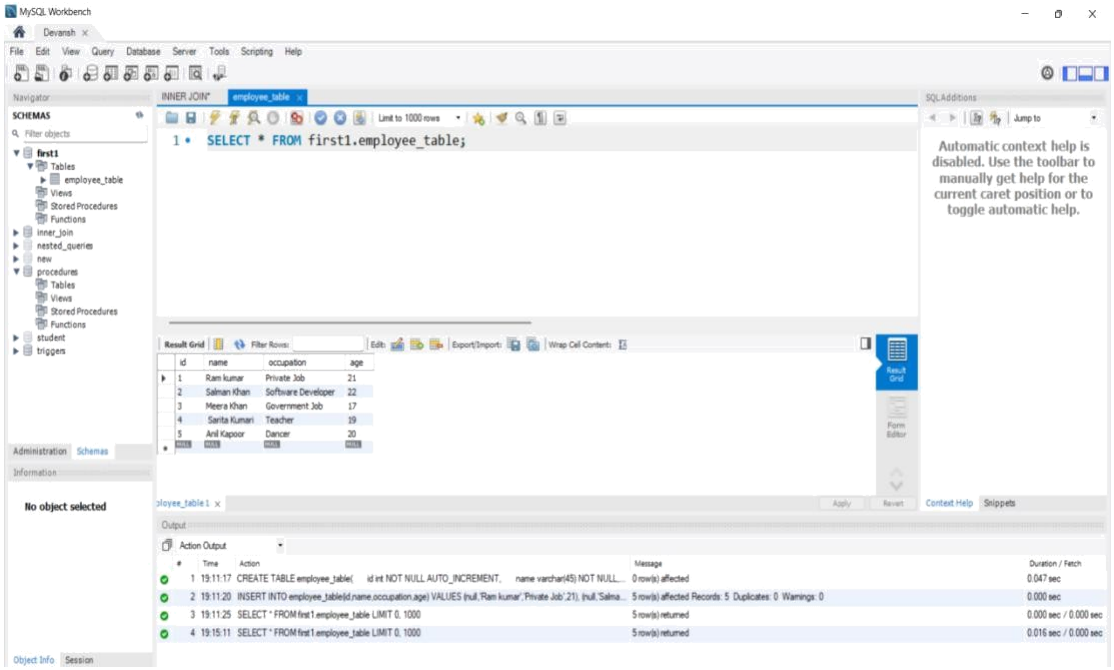
Delete command: After inserting row in a table we can also delete them if required. The delete command consists of a from clause followed by an optional where clause.

PROCEDURE

STEP 1: Start
STEP 2: Create the table with its essential attributes. STEP 3: Insert the record into table
STEP 4: Update the existing records into the table STEP 5: Delete the records in to the table.
STEP 6: Select the records

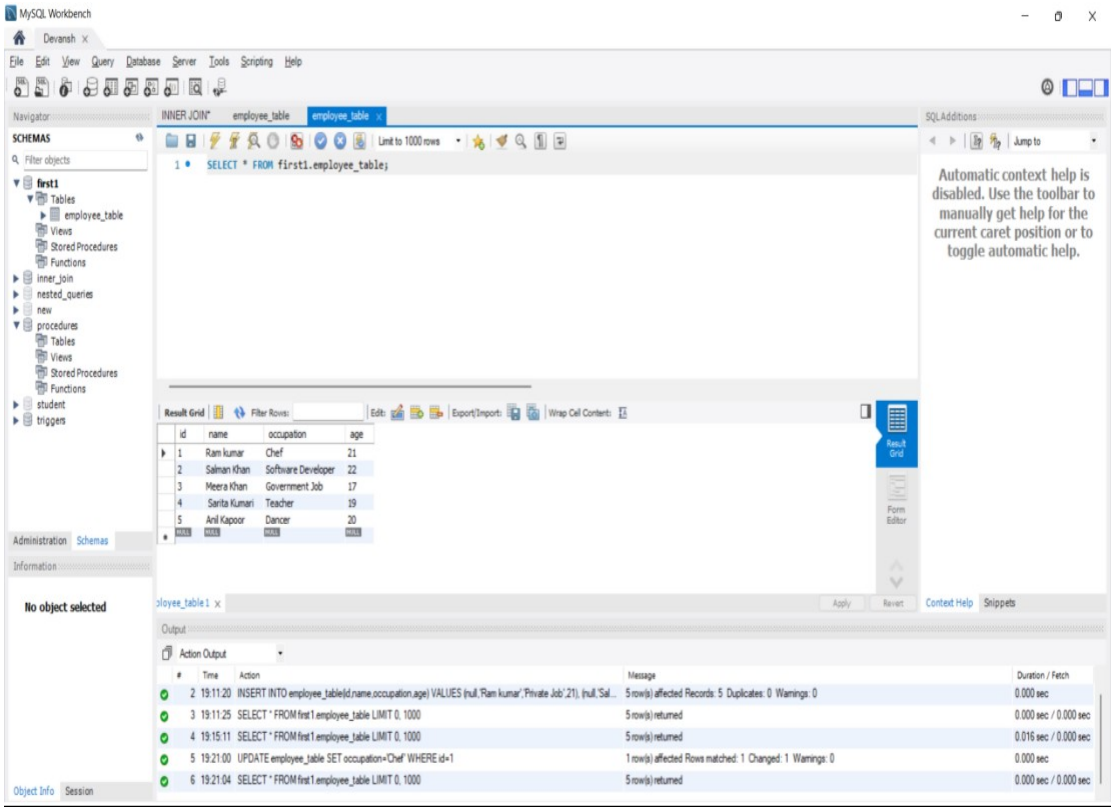
1. INSERT Command:-

```
INSERT INTO employee_table(id,name,occupation,age)
VALUES
(null,'Ram kumar','Private Job',21),
(null,'Salman Khan','Software Developer',22),
(null,'Meera Khan','Government Job',17),
(null,' Sarita Kumari','Teacher',19),
(null,'Anil Kapoor','Dancer',20);
```

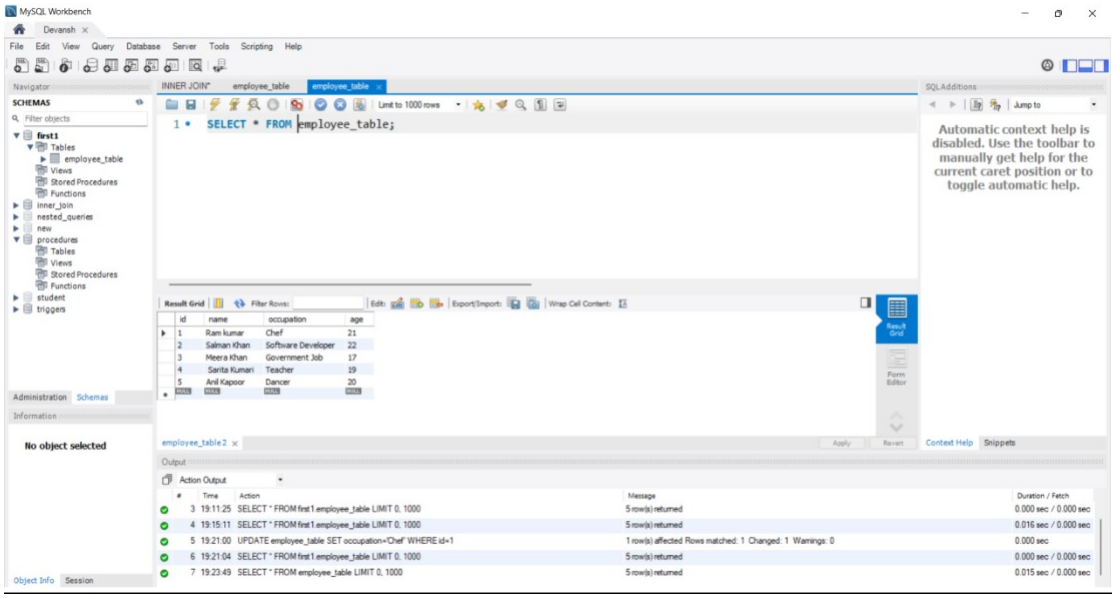


2. Update Command:-

```
UPDATE employee_table
SET occupation='Chef'
WHERE id=1;
```

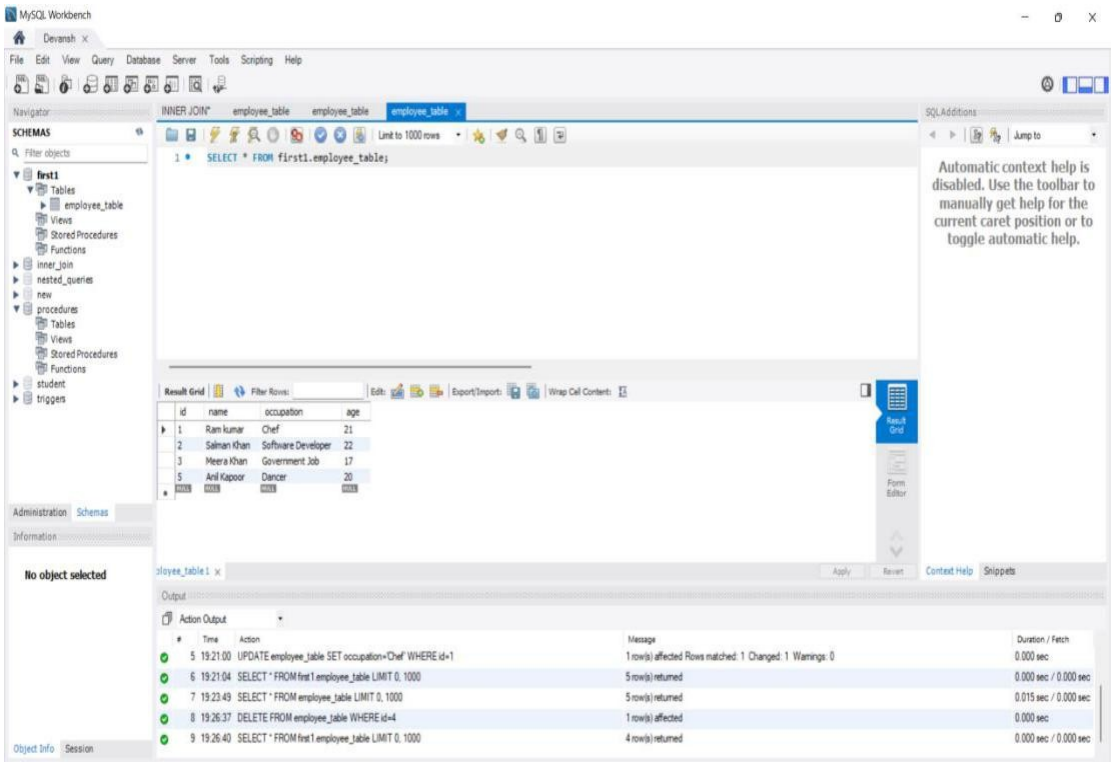


3.Select Command:-



3. Delete command:-

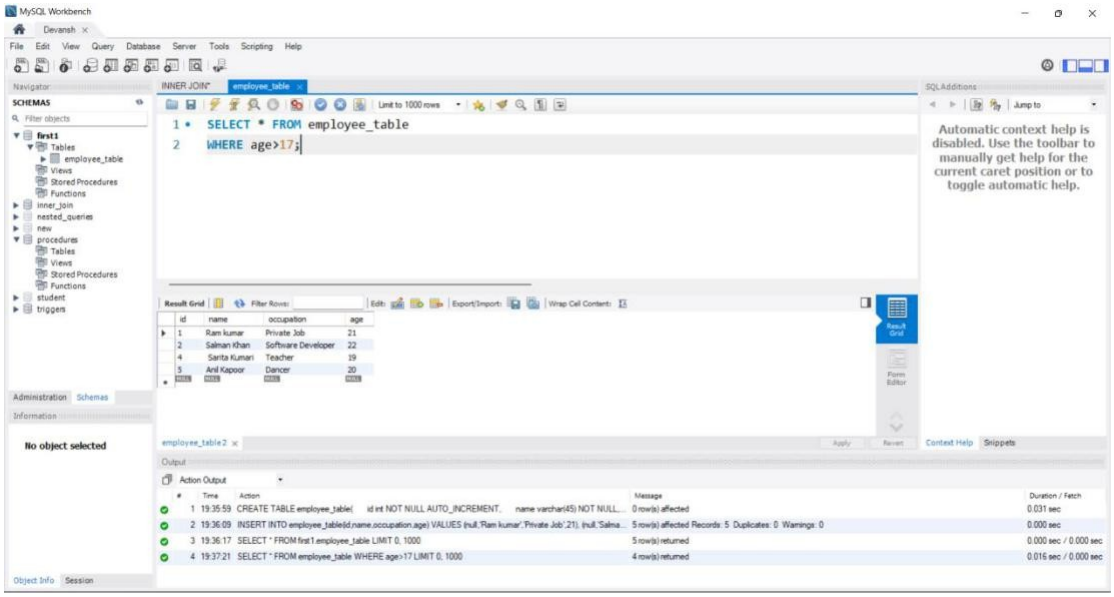
DELETE FROM employee_table
WHERE id=4;



To Implement Basic Select statements

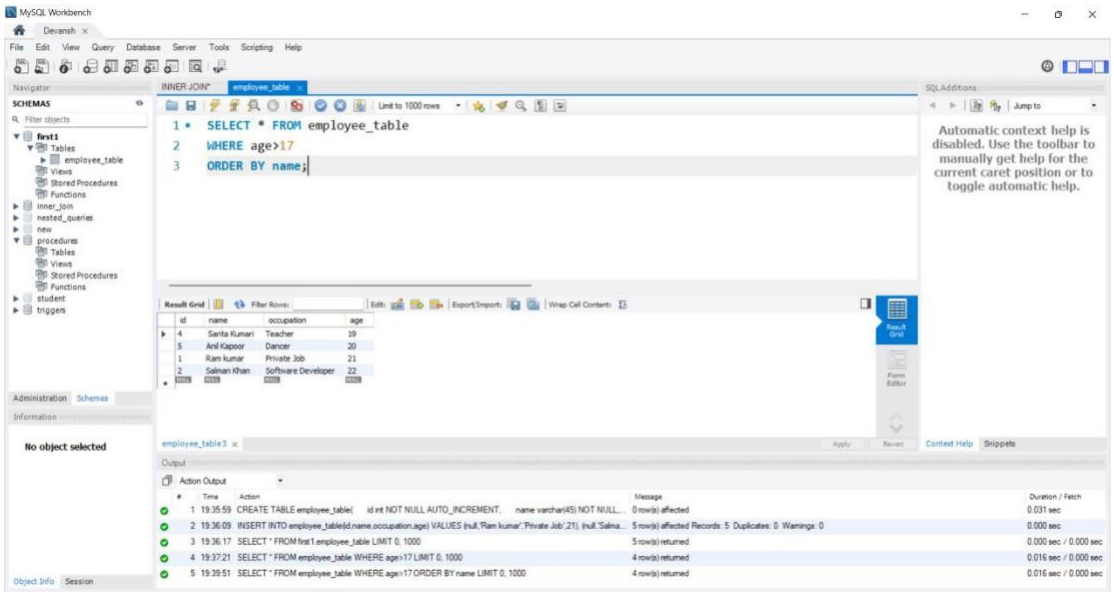
1. SELECT with WHERE Clause:-

SELECT * FROM employee_table
WHERE age>17;



2. SELECT With ORDER BY:-

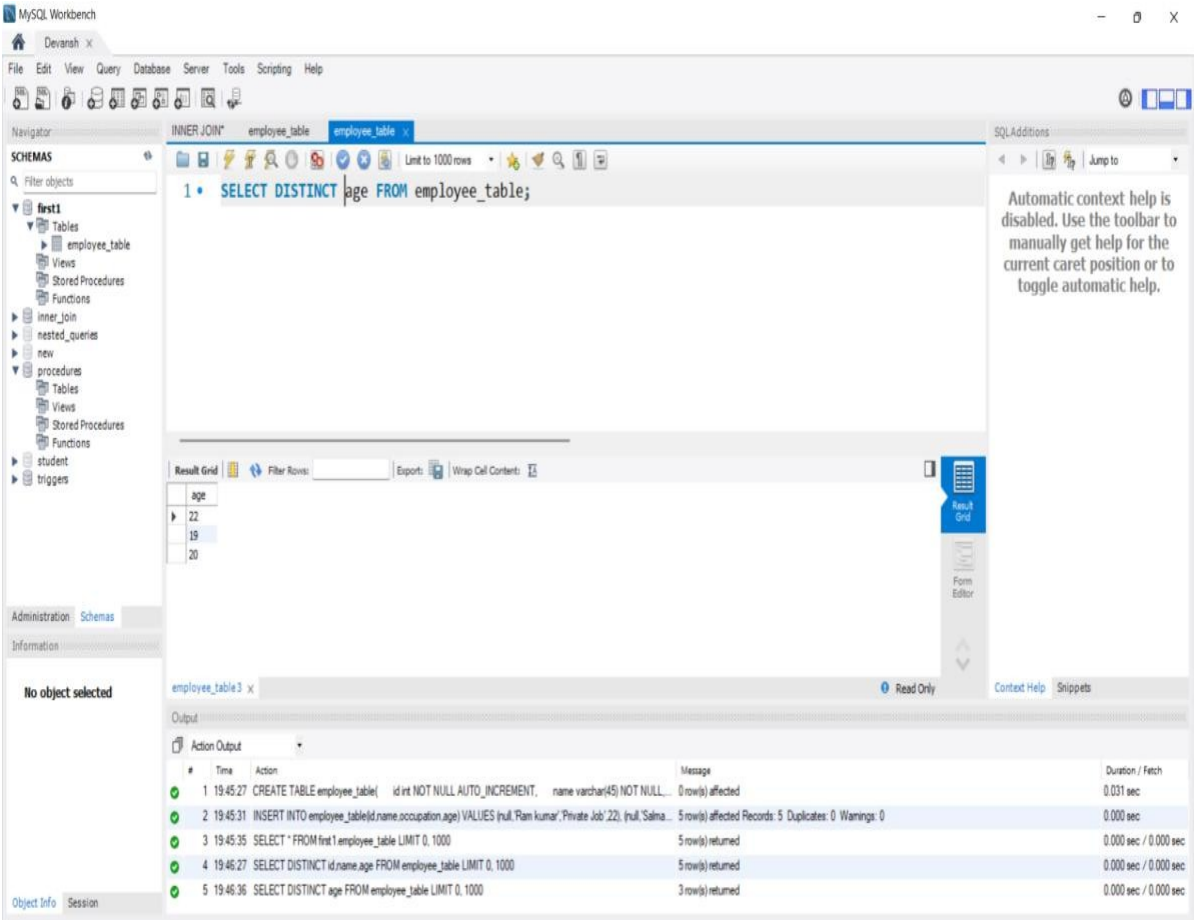
SELECT * FROM employee_table
WHERE age>17
ORDER BY name;



3. SELECT WITH DISTINCT:-

QUERY:-

SELECT DISTINCT age FROM employee_table;



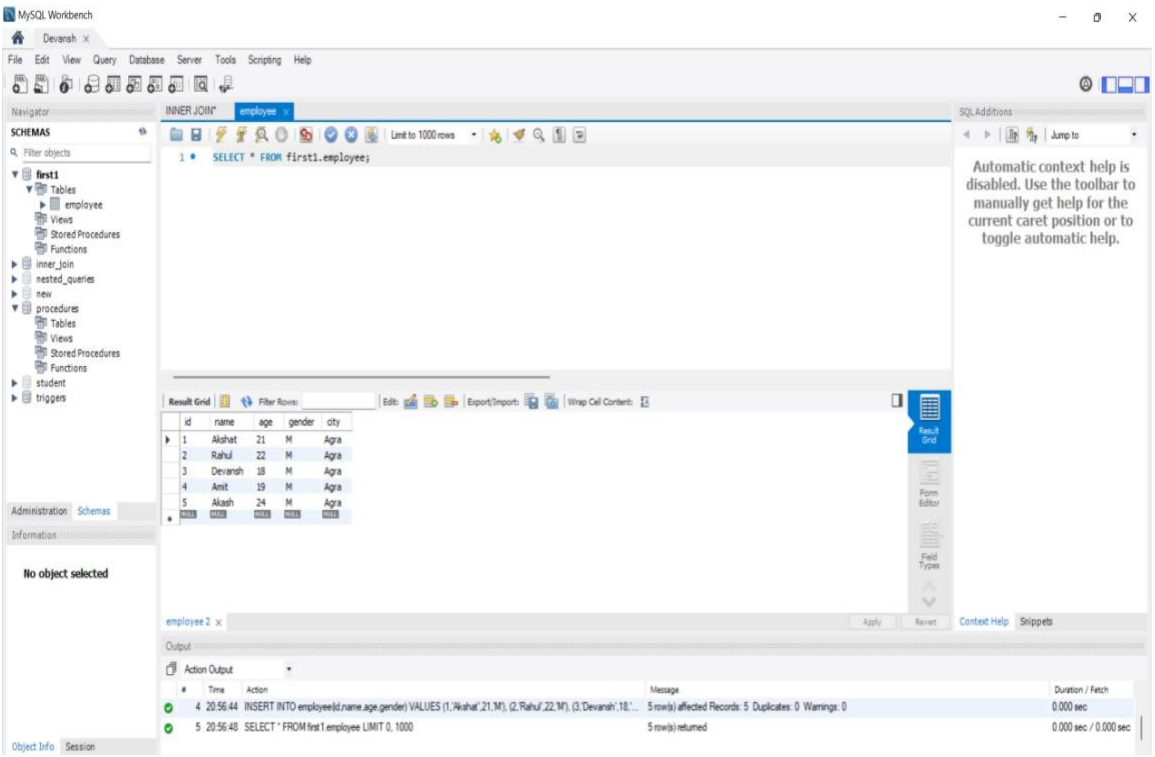
To Execute Constraints in MySQL

Theory:-

The constraint in MySQL is used to specify the rule that allows or restricts what values/data will be stored in the table. They provide a suitable method to ensure **data** accuracy and integrity inside the table. It also helps to limit the type of data that will be inserted inside the table. If any interruption occurs between the constraint and data action, the action is failed.

QUERY:-

```
CREATE TABLE employee(
id INT NOT NULL UNIQUE,
name VARCHAR(50) NOT NULL,
age INT NOT NULL CHECK(age>=18),
gender VARCHAR(10)NOT NULL,
city VARCHAR(10) NOT NULL DEFAULT
'Agra' );

INSERT INTO employee(id,name,age,gender)
VALUES
(1,'Akshat',21,'M'),
(2,'Rahul',22,'M'),
(3,'Devansh',18,'M'),
(4,'Amit',19,'M'),
(5,'Akash',24,'M');
```



Exp No:-3 Date:-

EXPERIMENT:-DCL Commands And TCL Commands

Aim:- SQL data control language and transaction control Language

Theory:- DCL (Data Control Language) includes commands like GRANT and REVOKE, which are useful to give "rights & permissions." Other permission controls parameters of the database system.

Examples of DCL commands:-

Commands that come under DCL:-

- Grant
- Revoke

GRANT

```
create user admin1 identified by password1;
grant create session to admin1;
grant connect  to admin1;
```

REVOKE

```
revoke create session from admin1;
revoke connect  from admin1;
revoke create table from admin1;
```

GRANT

```
create user admin1 identified by password1;
grant create session to admin1;
grant connect  to admin1;
```

Script Output

Task completed in 0.168 seconds

Grant succeeded.

REVOKE

```
create user admin1 identified by password1;
revoke create session from admin1;
revoke connect  from admin1;
revoke unlimited tablespace from admin1;
revoke all privileges from admin1;
```

Script Output

Task completed in 0.056 seconds

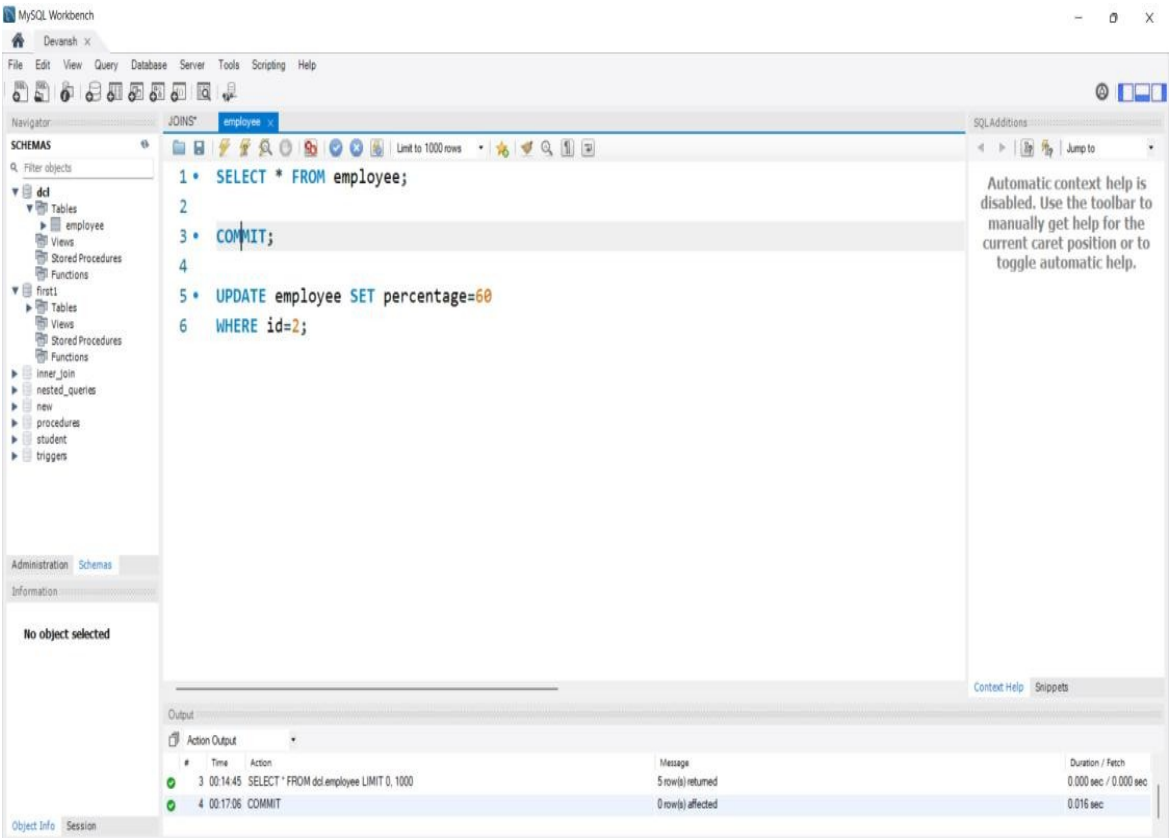
Revoke succeeded.

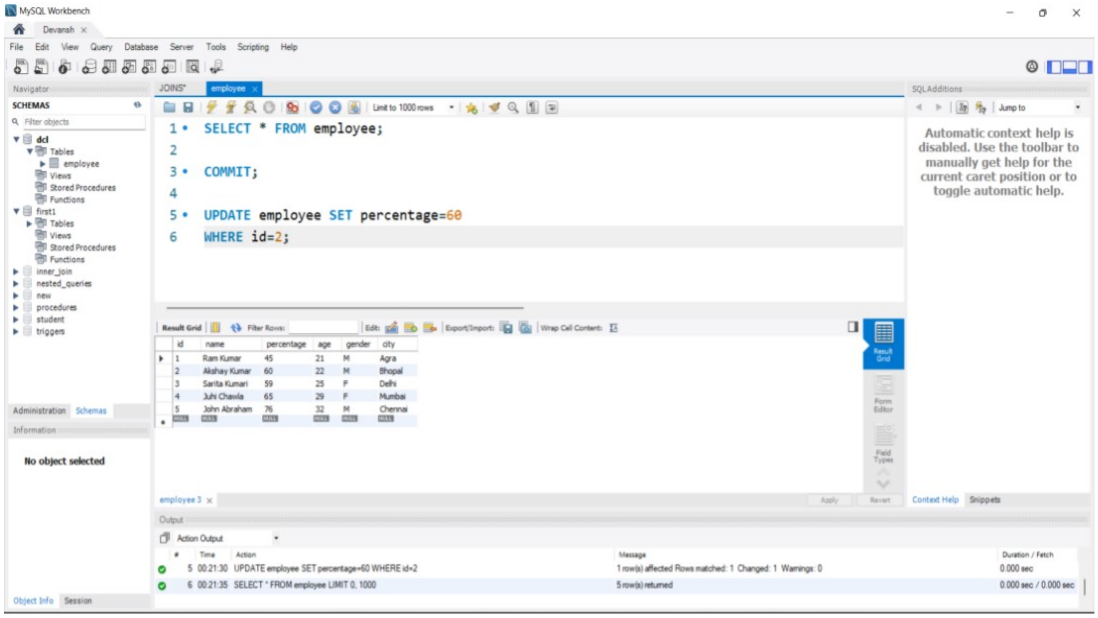
TCL Commands in SQL

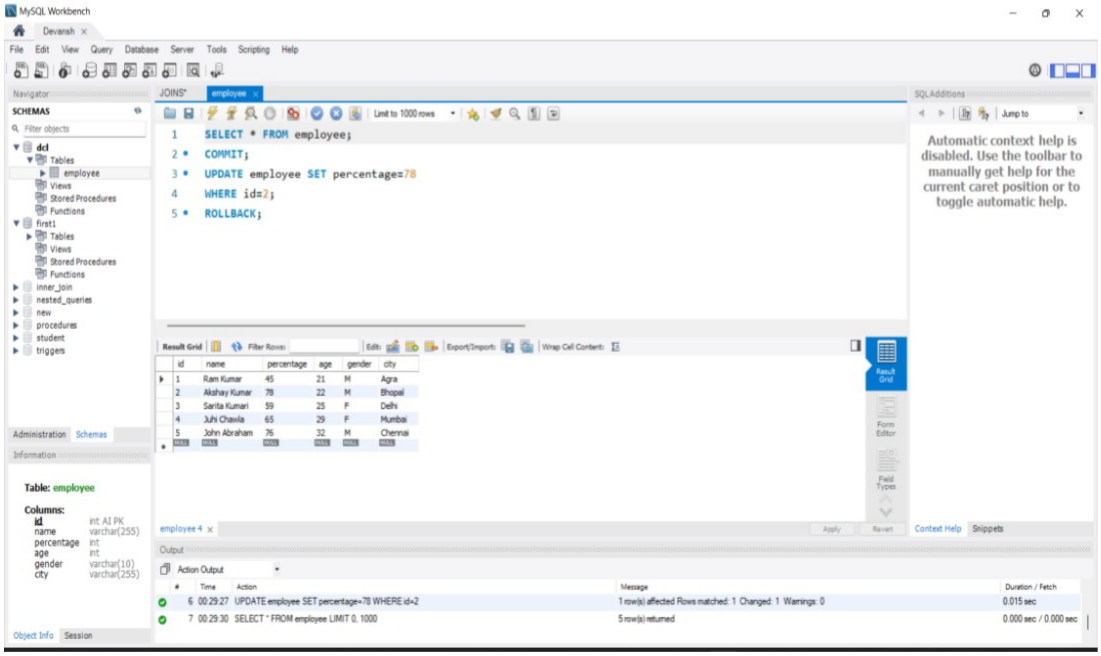
- In SQL, TCL stands for Transaction control language.
- A single unit of work in a database is formed after the consecutive execution of commands is known as a transaction.
- There are certain commands present in SQL known as TCL commands that help the user manage the transactions that take place in a database.
- COMMIT,ROLLBACK and SAVEPOINT are the most commonly used TCL commands in SQL.

1. COMMIT:-

COMMIT command in SQL is used to save all the transaction-related changes permanently to the disk. Whenever DDL commands such as INSERT, UPDATE and DELETE are used, the changes made by these commands are permanent only after closing the current session. So before closing the session, one can easily roll back the changes made by the DDL commands. Hence, if we want the changes to be saved permanently to the disk without closing the session, we will use the commit command.

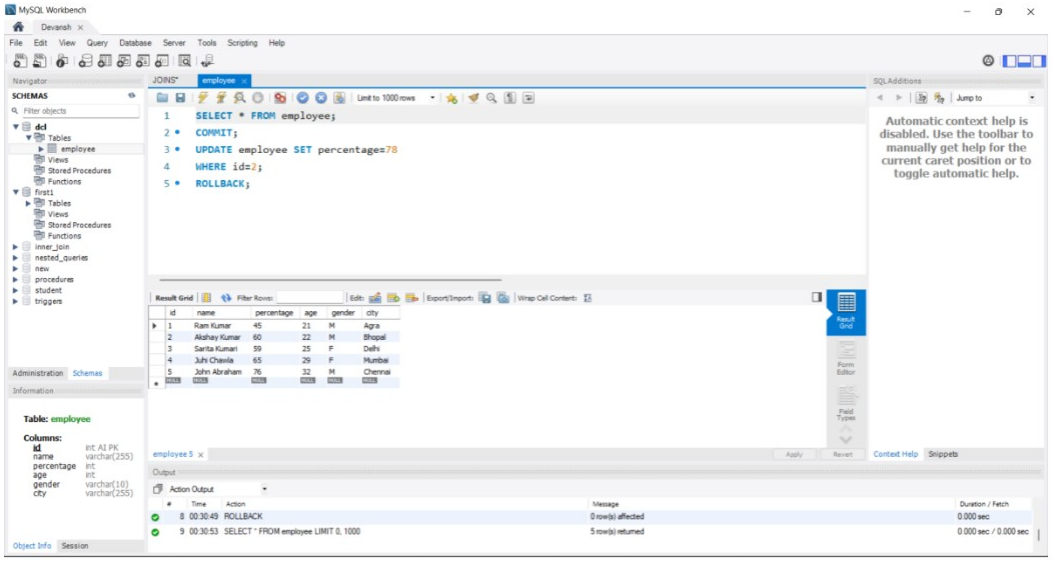


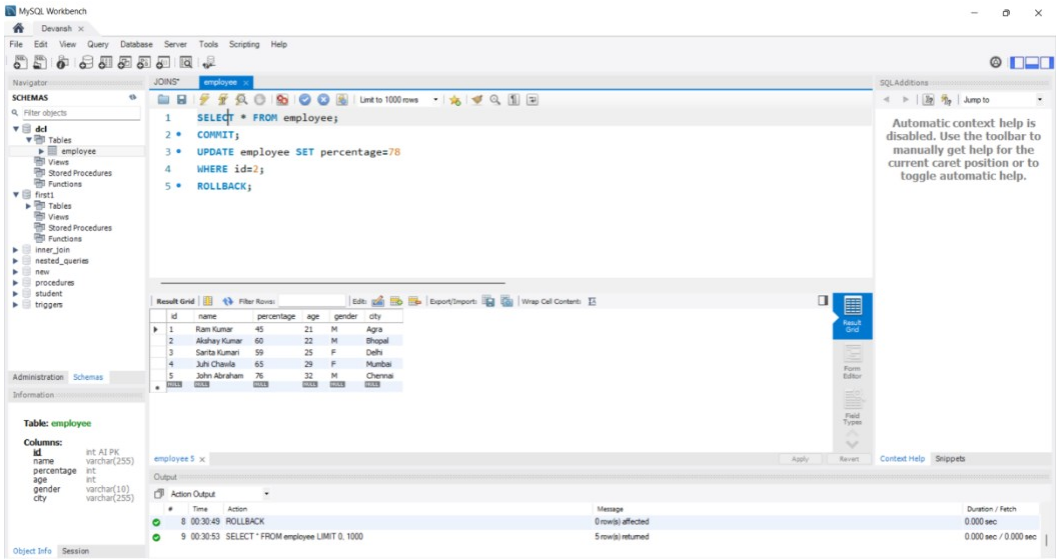




2.<u>ROLLBACK</u>

While carrying a transaction, we must create savepoints to save different parts of the transaction. According to the user's changing requirements, he/she can roll back the transaction to different savepoints. Consider a scenario: We have initiated a transaction followed by the table creation and record insertion into the table. After inserting records, we have created a savepoint INS. Then we executed a delete query, but later we thought that mistakenly we had removed the useful record. Therefore in such situations, we have an option of rolling back our transaction. In this case, we have to roll back our transaction using the ROLLBACK command to the savepoint INS, which we have created before executing the DELETE query.





3.SAVEPOINT:-

We can divide the database operations into parts. For example, we can consider all the insert related queries that we will execute consecutively as one part of the transaction and the delete command as the other part of the transaction. Using the SAVEPOINT command in SQL, we can save these different parts of the same transaction using different names. For example, we can save all the insert related queries with the savepoint named INS. To save all the insert related queries in one savepoint, we have to execute the SAVEPOINT query followed by the savepoint name after finishing the insert command execution.

```
create table a1(a_id int, a_serialno int);
insert into a1 values (1,123);
insert into a1 values (2,452);
insert into a1 values (3,526);
insert into a1 values (4,356);
savepoint a1_table;
```


Script Output × | Query Result ×

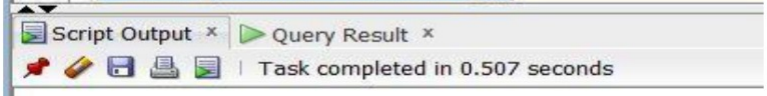
Task completed in 1.3 seconds

Savepoint created.

```
update a1 set a_serialno=242 where a_id=2;
savepoint a1_update;
```

Savepoint created.

```
drop table a1;
savepoint a1_drop;
```



Savepoint created.

Exp No:-4 Date:-

EXPERIMENT:-Inbuilt Functions

Aim:- To perform In-Built Functions In SQL.

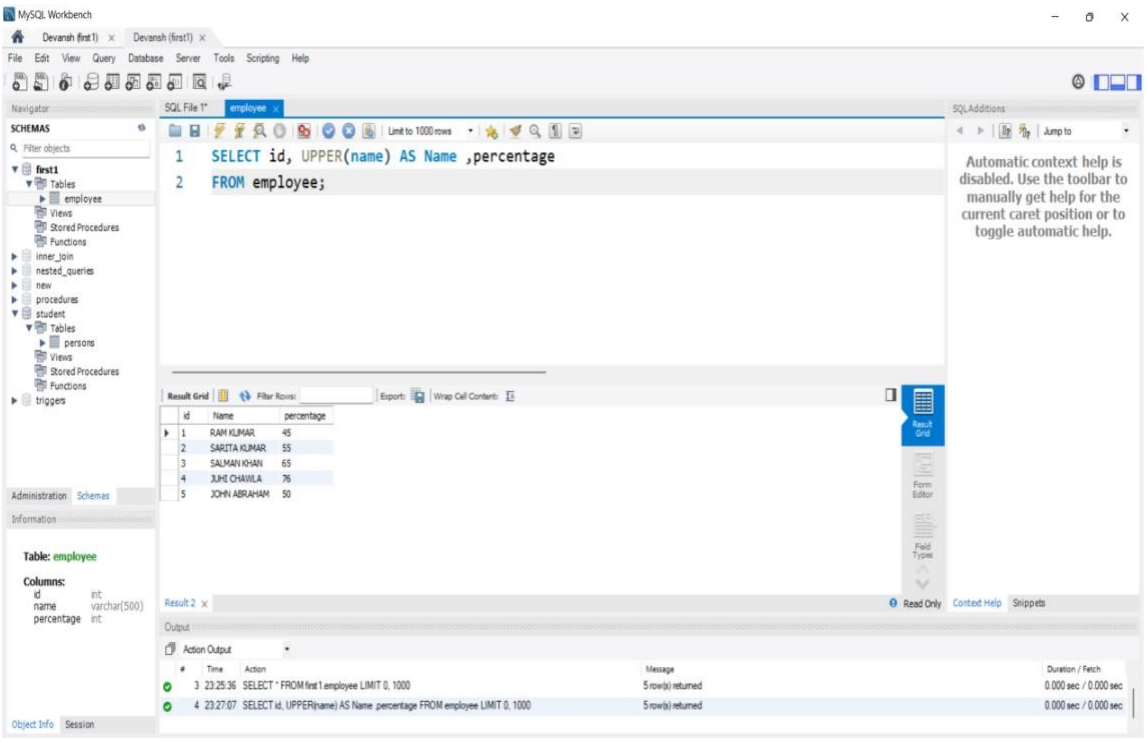
To Execute String, Date,time function

Theory:- MySQL has many built-in functions.This reference contains string, numeric, date, and some advanced functions in MySQL.

1. **UPPER()**:-

SELECT id, UPPER(name) AS Name ,percentage

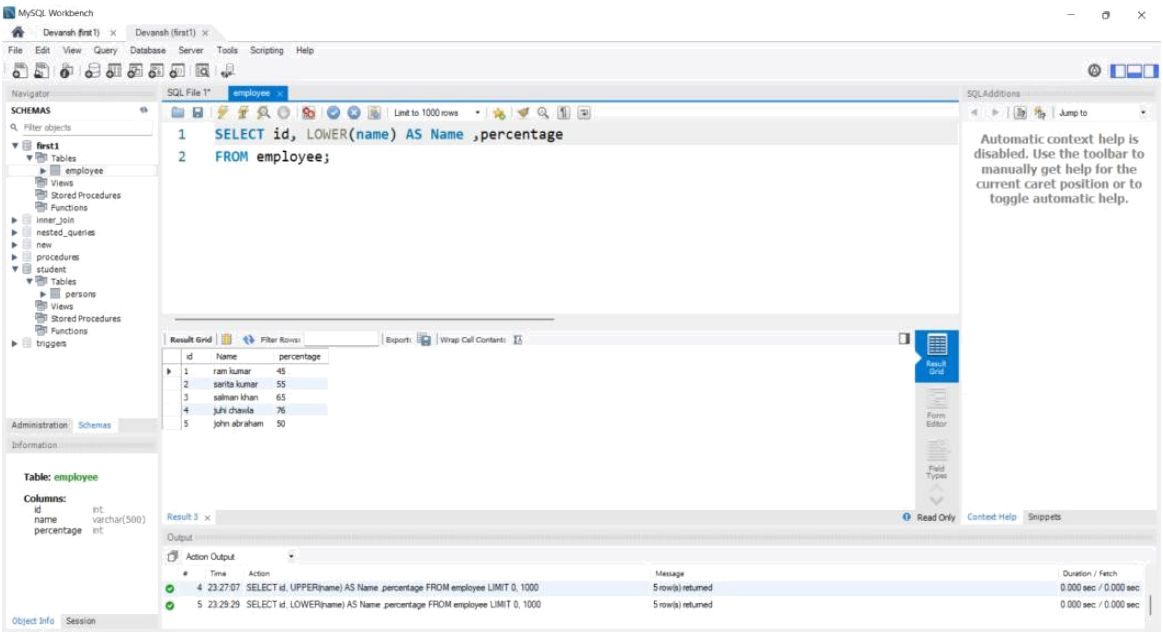
FROM employee;



2. **LOWER():-**

SELECT id, LOWER(name) AS Name ,percentage

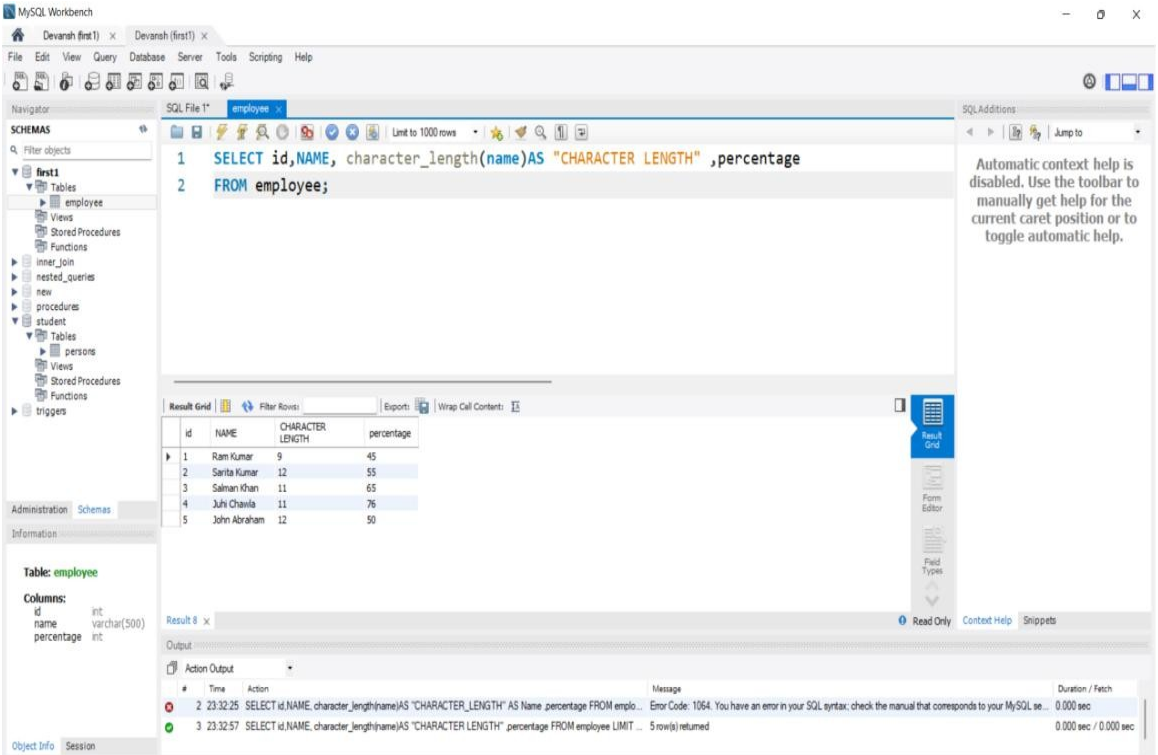
FROM employee;



3. CHARACTER_LENGTH():-

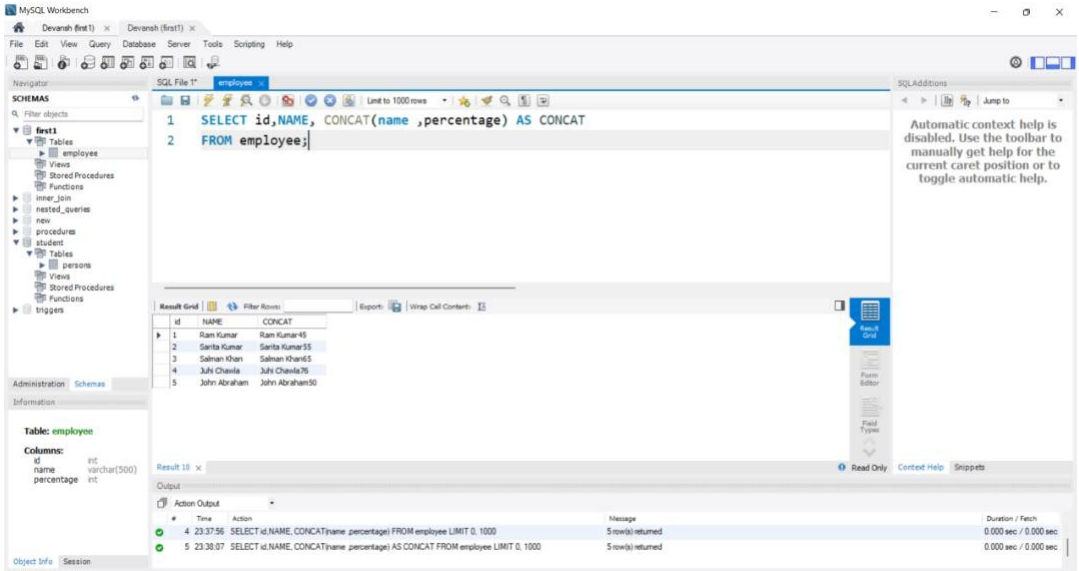
QUERY:-

SELECT id,NAME, character_length(name)AS "CHARACTER LENGTH" ,percentage
FROM employee;



4. CONCAT():-

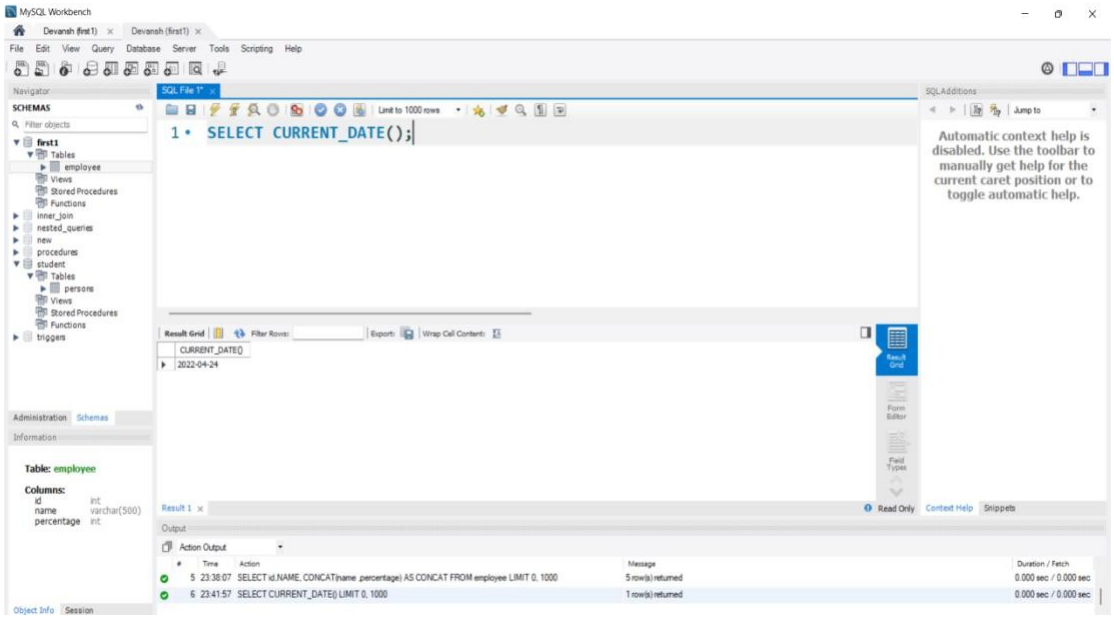
QUERY:-
SELECT id,NAME, CONCAT(name ,percentage) AS CONCAT
FROM employee;



DATE AND TIME FUNCTIONS:-

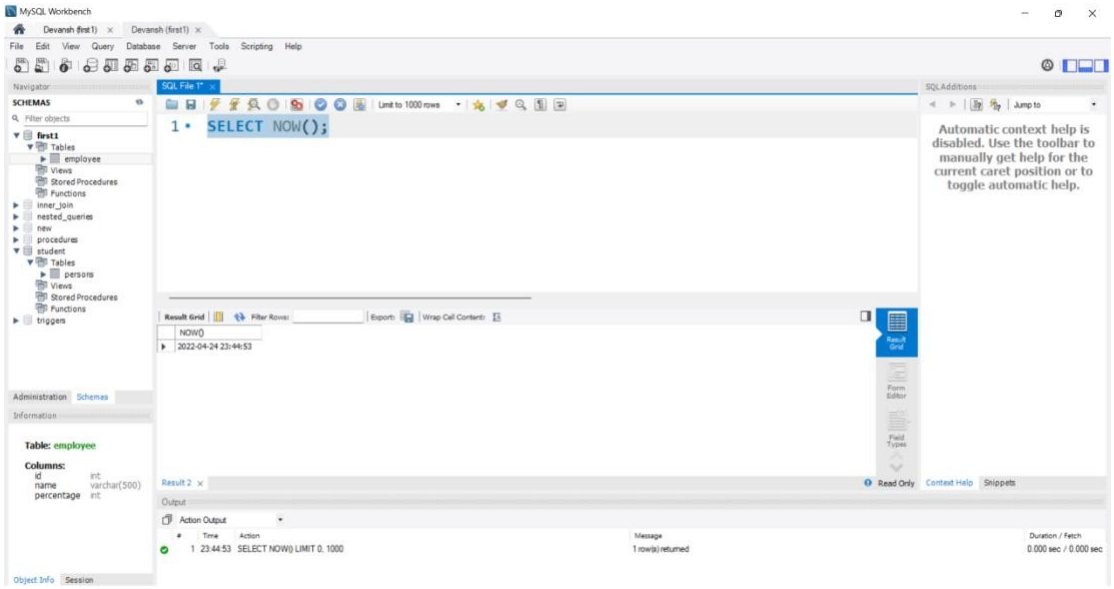
1.CURRENT_DATE():-

QUERY:-
SELECT CURRENT_DATE();



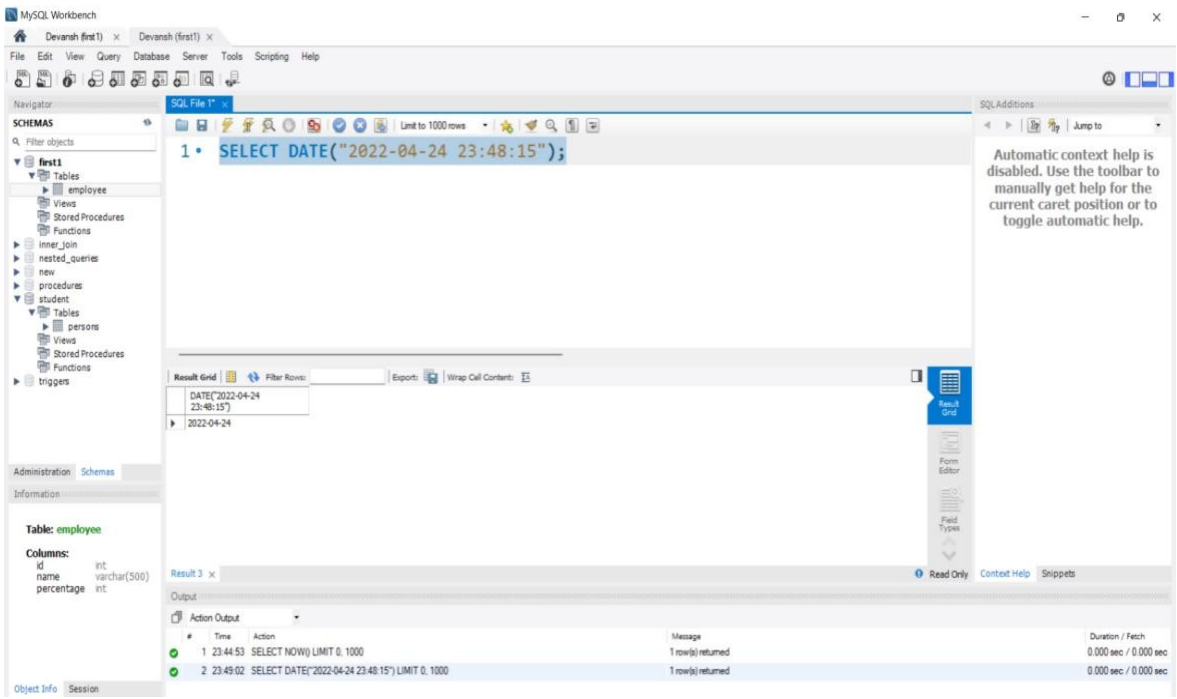
2. NOW():-

QUERY:-
SELECT NOW();



3 DATE():-

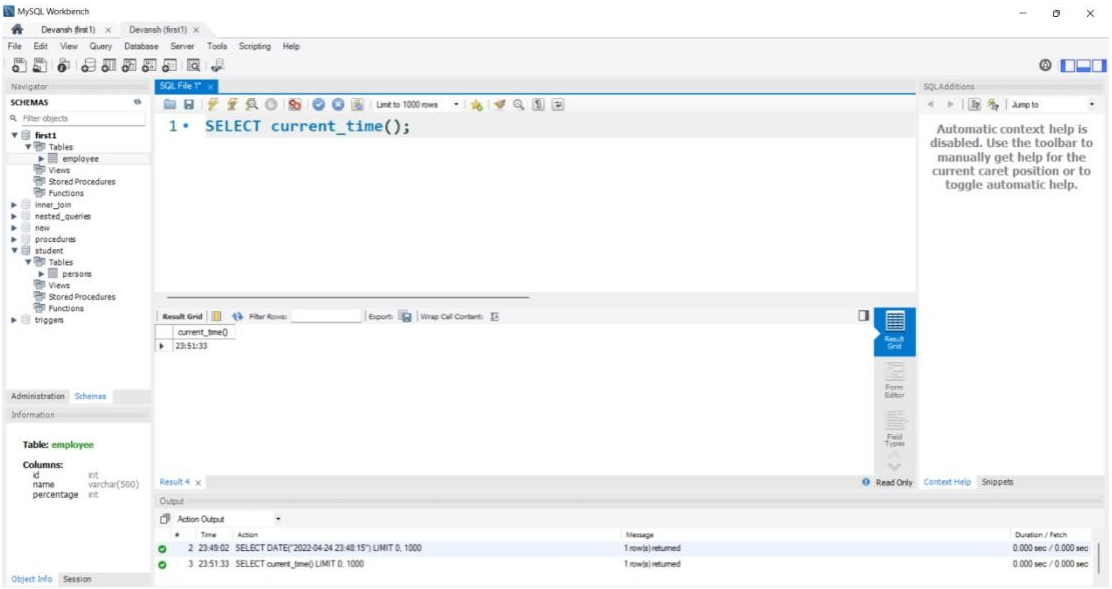
QUERY:-
SELECT DATE("2022-04-24 23:48:15");



TIME FUNCTIONS:-

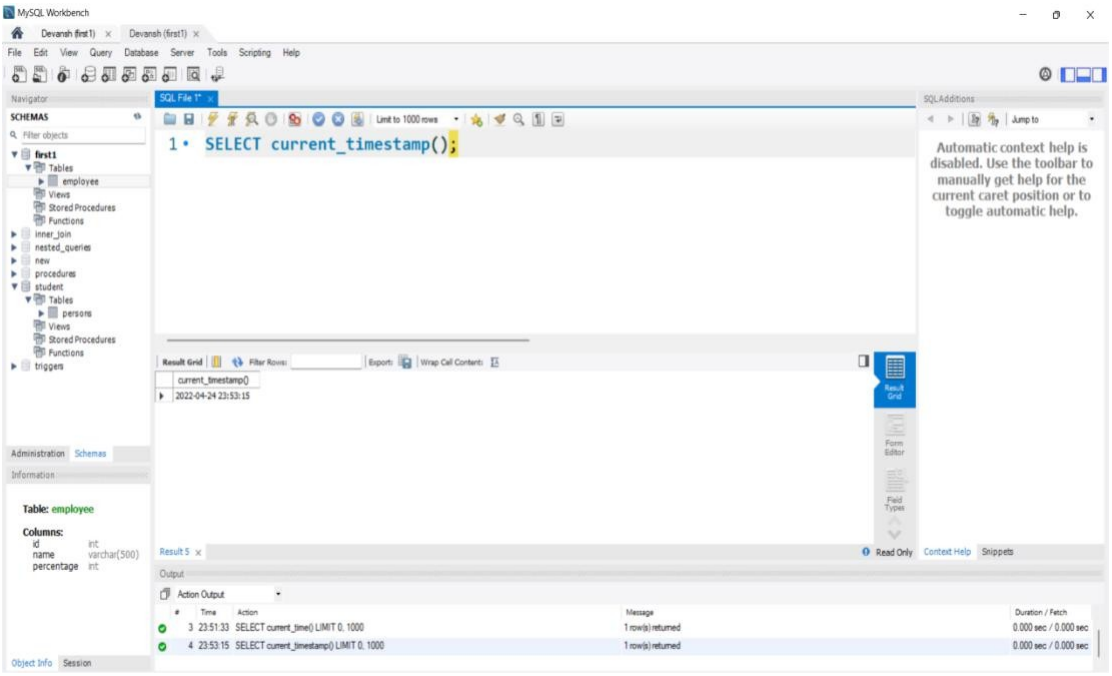
1.Current_time():-

QUERY:-
SELECT current_time();



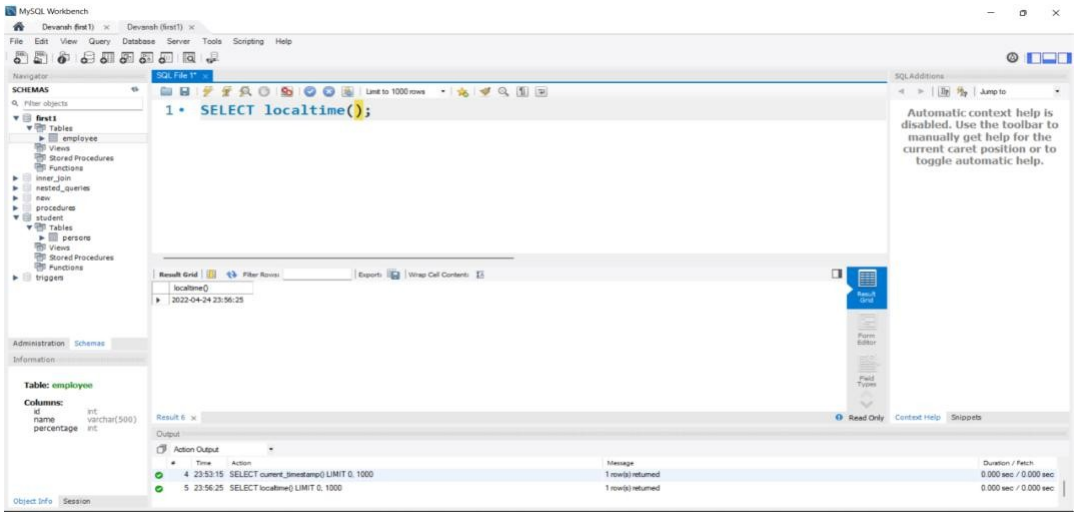
2 Current_timestamp():-

QUERY:-
SELECT current_timestamp();



3. localtime():-

QUERY:-
SELECT localtime();



To Execute Aggregate Functions.

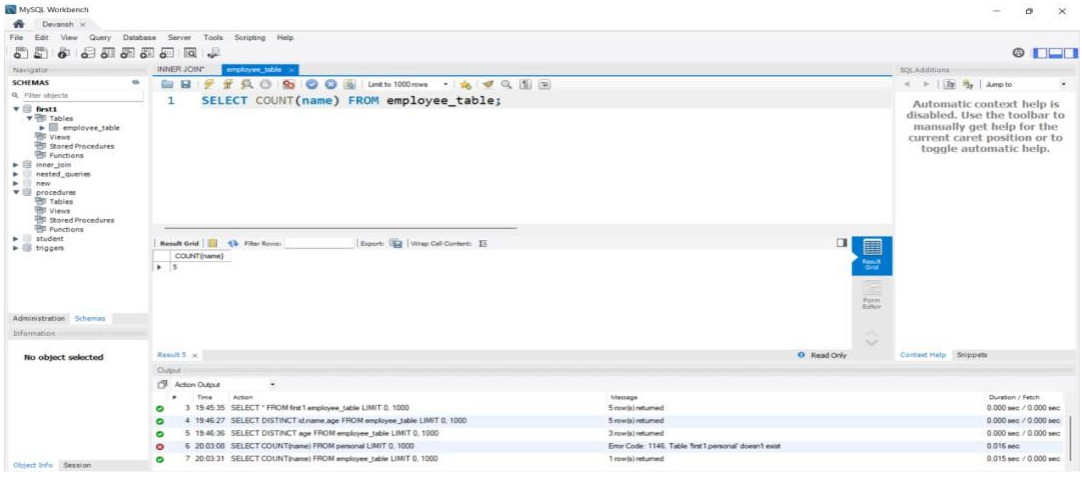
Theory:-SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.It is also used to summarize the data.

1. COUNT FUNCTION:-

COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.

QUERY:-

SELECT COUNT(name) FROM employee_table;

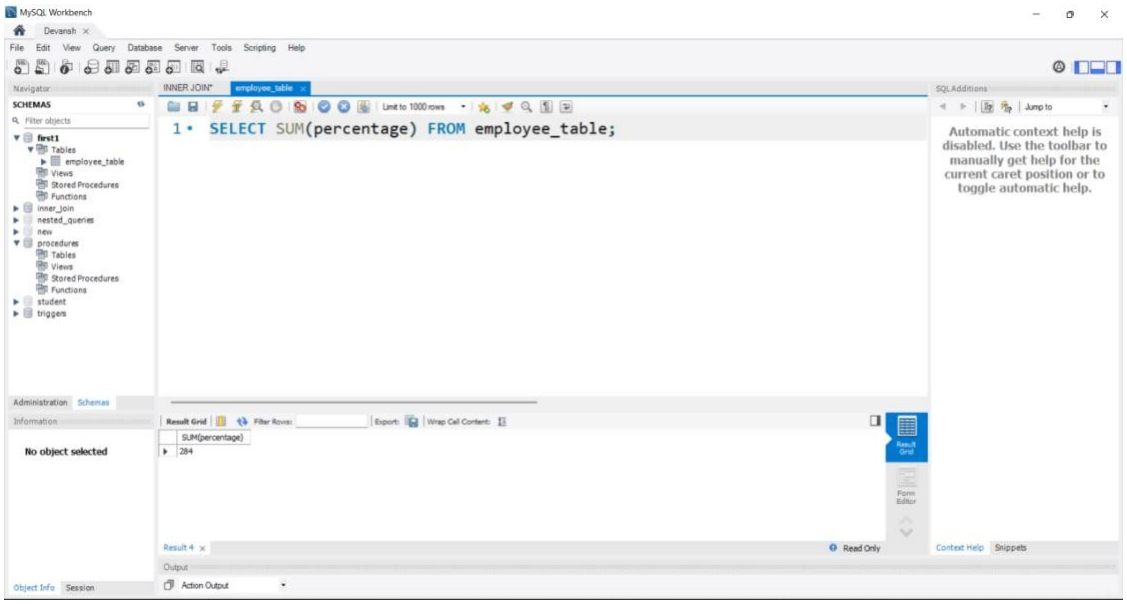


2. SUM Function:-

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

QUERY:-

SELECT SUM(percentage) FROM employee_table;

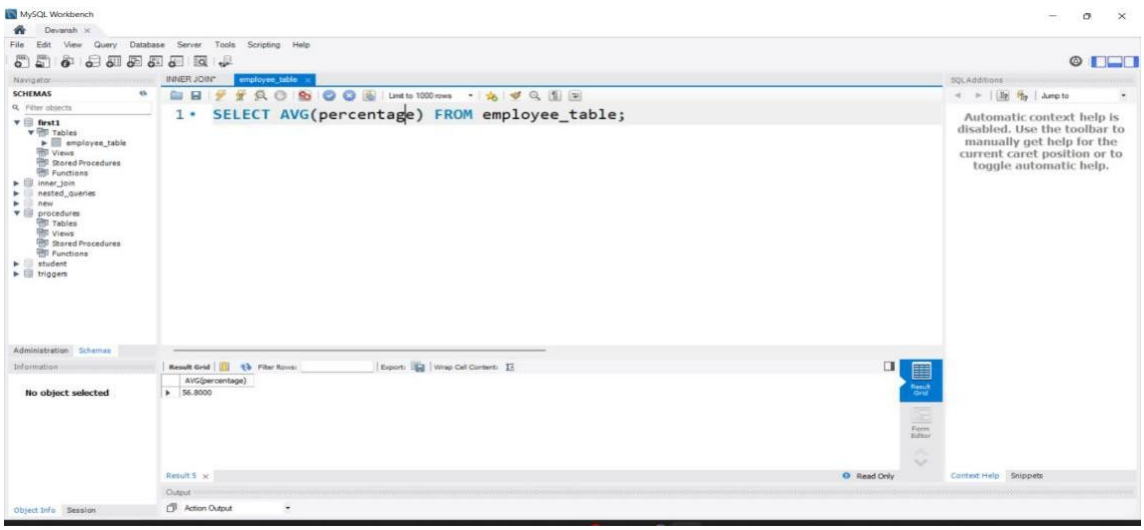


3. AVG function:-

The AVG function is used to calculate the average value of the numeric type.

QUERY:-

SELECT AVG(percentage) FROM employee_table;

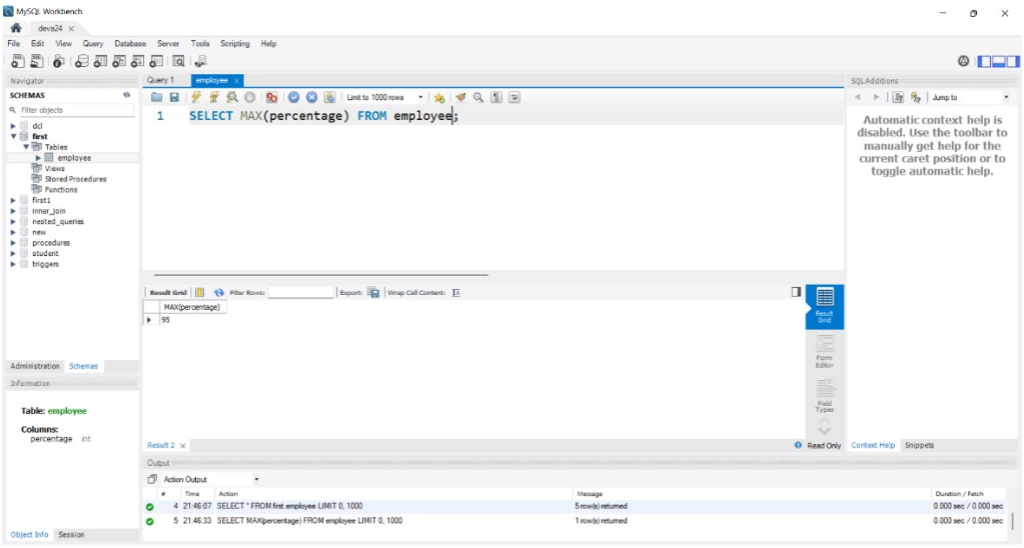


4. MAX Function:-

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

QUERY:-

SELECT MAX(percentage) FROM employee;

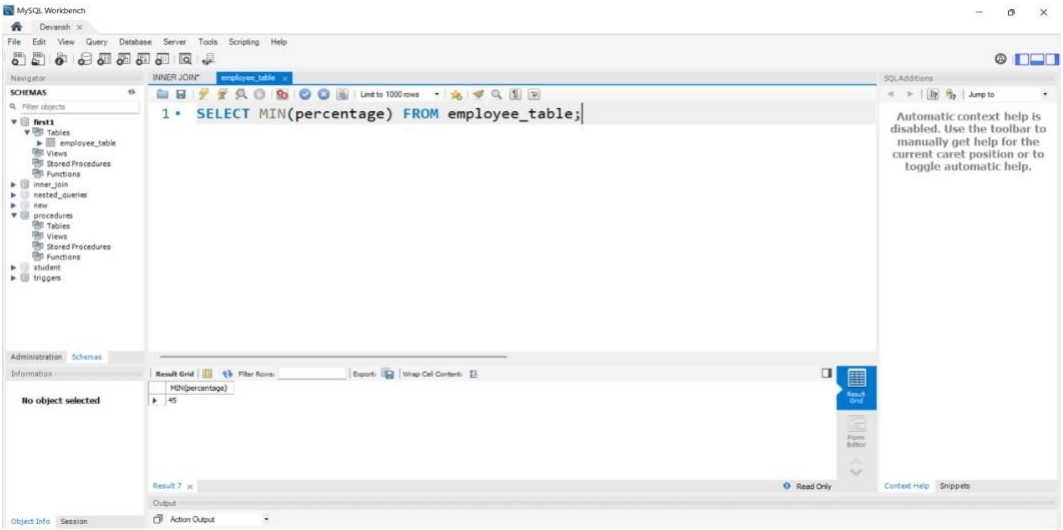


5.MIN Function:-

MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

QUERY:-

SELECT MIN(percentage) FROM employee_table;



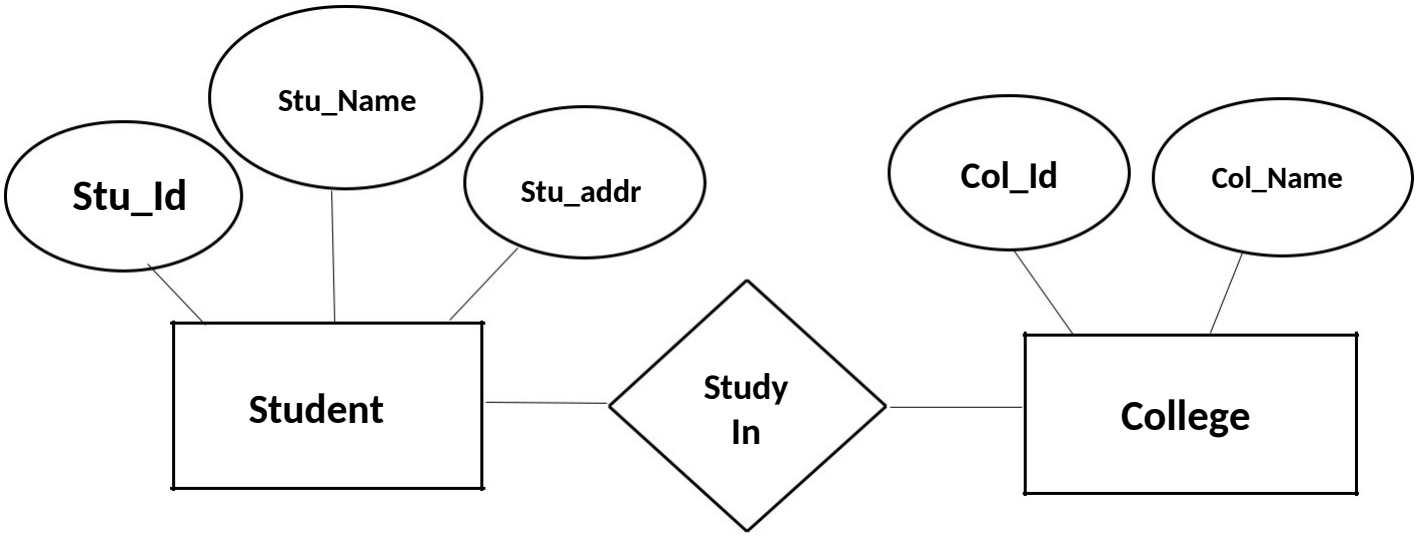
Exp No:-5 Date:-

EXPERIMENT:-ER Model

Aim:-Construct a e- r model for the application to be constructed to a Database:

a) E-R model for Property Management Database Project
b) E-R model for Water Supply Management System
c) E-R model for Home renting system database
d) E-R model for Complaint management system database
e) E-R model for Employee performance review system database
f) E-R model for Employee track and report system database

Theory:- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.



Exp No:-6 Date:-

EXPERIMENT:-Nested Queries

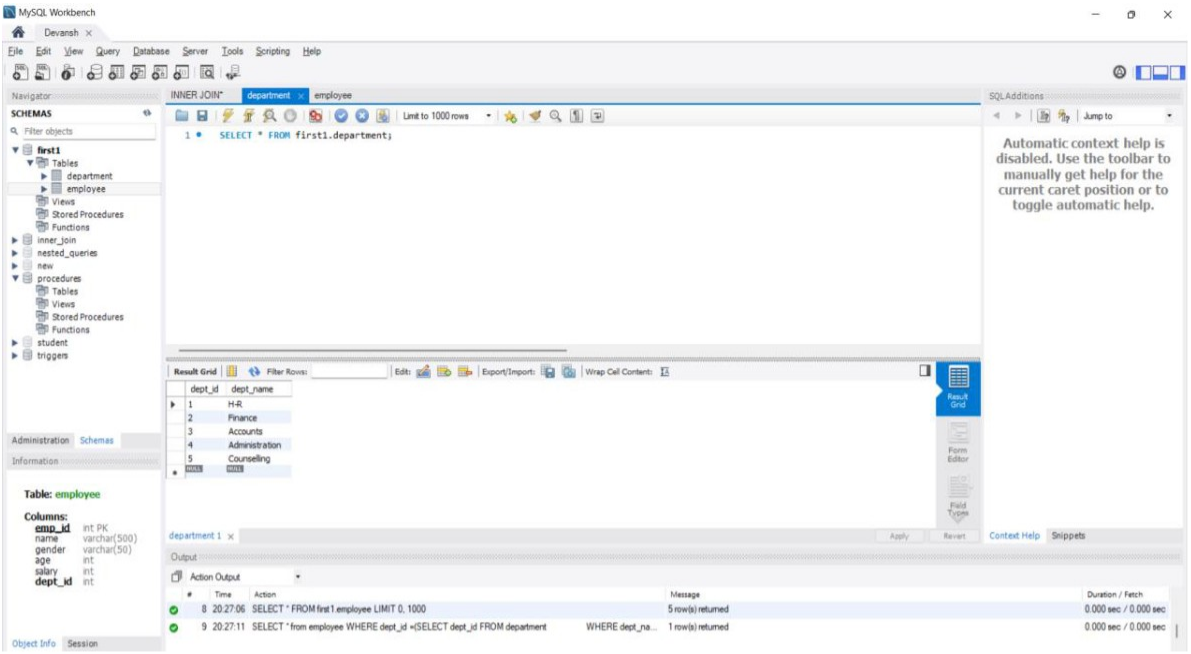
Aim:- To Execute Subqueries

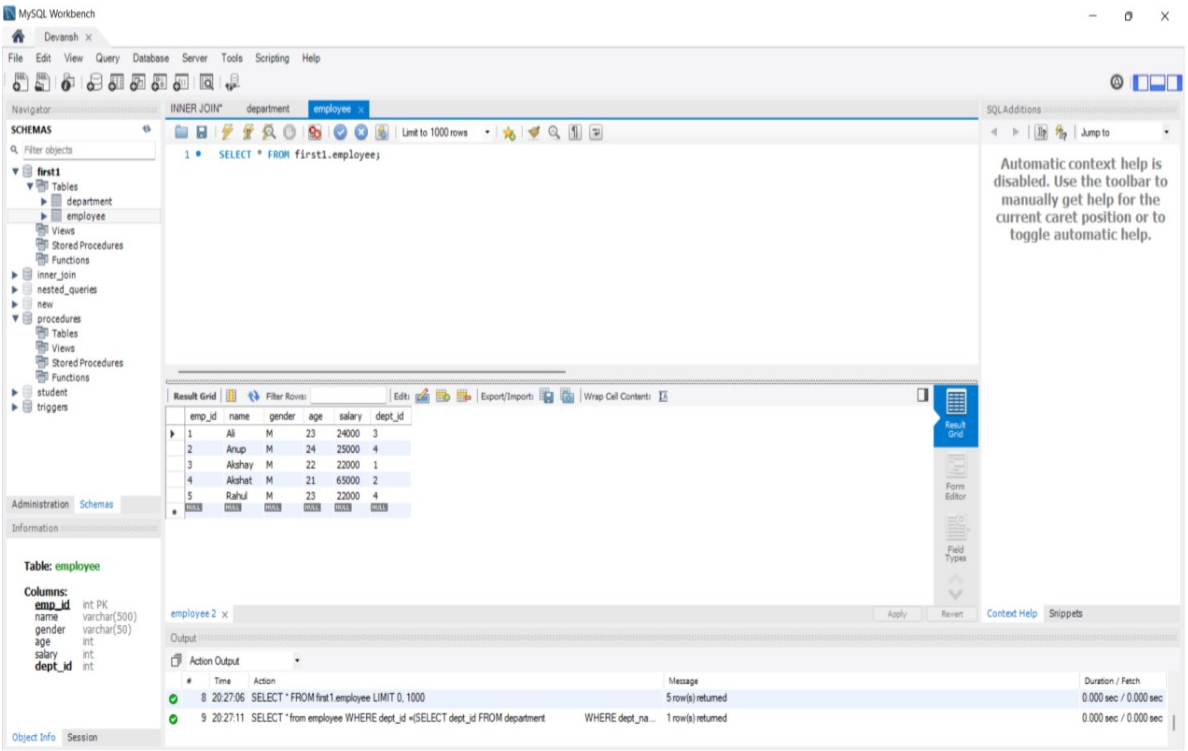
Theory:-A subquery in MySQL is a query, which is nested into another SQL query and embedded with SELECT, INSERT, UPDATE or DELETE statement along with the various operators. We can also nest the subquery with another subquery. A subquery is known as the inner query, and the query that contains subquery is known as the outer query. The inner query executed first gives the result to the outer query, and then the main/outer query will be performed. MySQL allows us to use subquery anywhere, but it must be closed within parenthesis. All subquery forms and operations supported by the SQL standard will be supported in MySQL also.

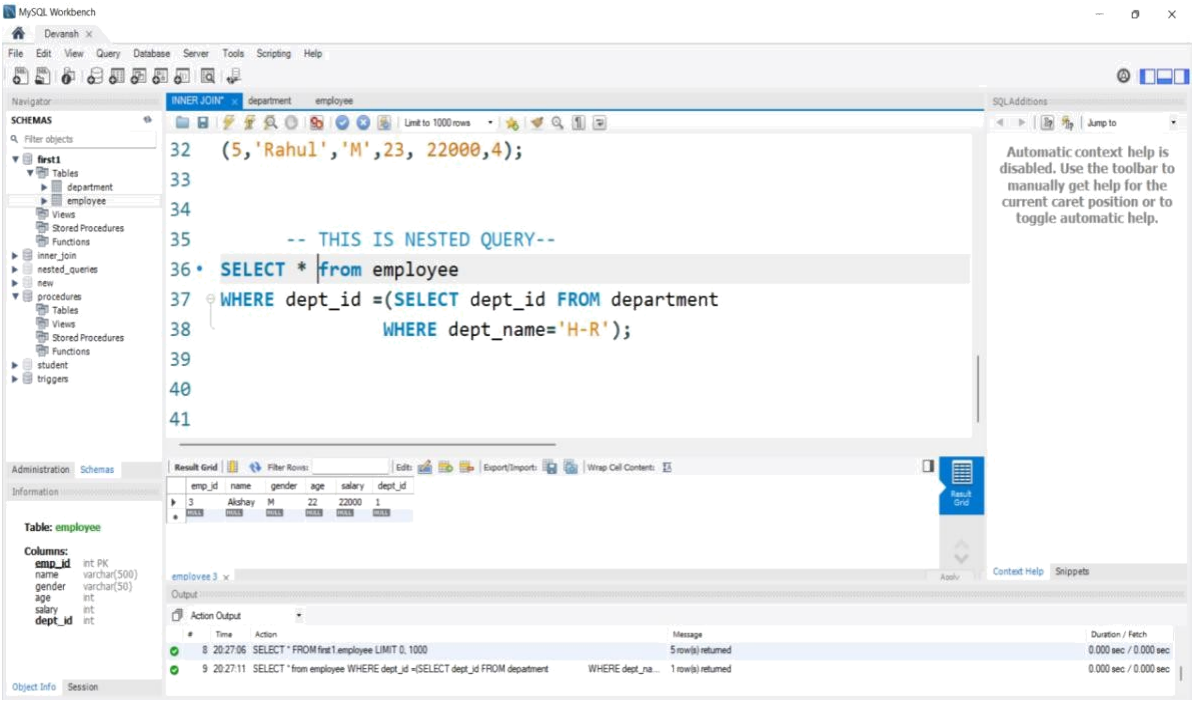
QUERY:-

```
CREATE TABLE department(
dept_id INT PRIMARY KEY,
dept_name VARCHAR(50)
);

INSERT INTO department
VALUES
(1,'H-R'),
(2,'Finance'),
(3,'Accounts'),
(4,'Administration'),
(5,'Counselling');


CREATE TABLE employee(
emp_id INT PRIMARY KEY,
name VARCHAR(500),
gender VARCHAR(50),
age INT,
salary INT,
dept_id INT,
FOREIGN KEY(dept_id)
REFERENCES department(dept_id)
);

INSERT INTO employee
VALUES
(1,'Ali','M',23,24000,3),
(2,'Anup','M',24,25000,4),
(3,'Akshay','M',22,22000,1),
(4,'Akshat','M',21,65000,2),
(5,'Rahul','M',23, 22000,4);
```

```
 -- THIS IS NESTED QUERY--
SELECT * from employee
WHERE dept_id =(SELECT dept_id FROM
            department WHERE dept_name='H-R');
```







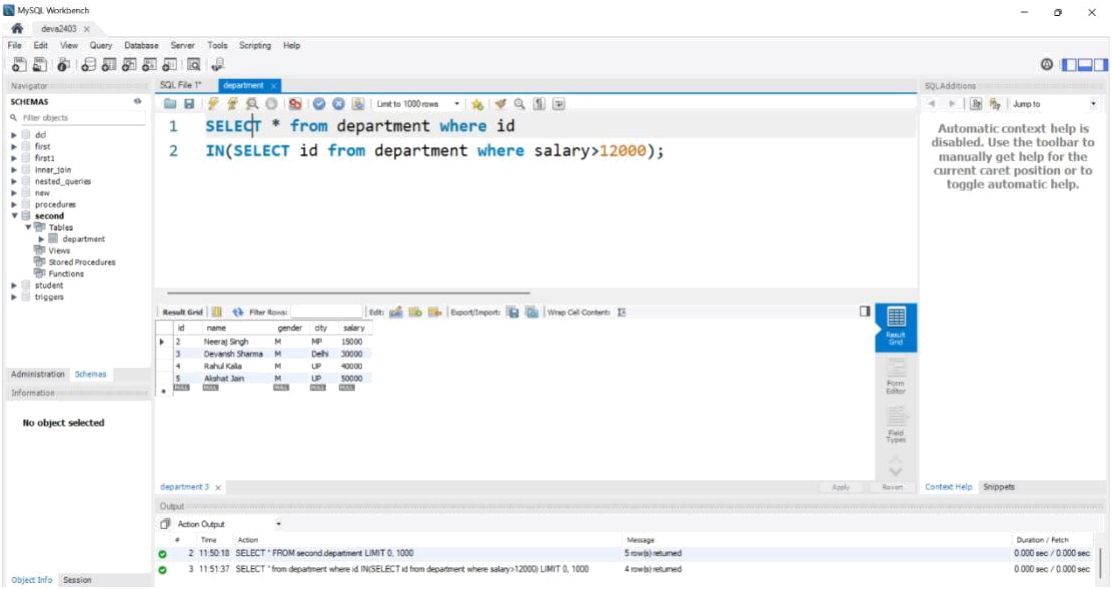
Subqueries:-

CREATING TABLE FOR SUBQUERY

```
CREATE TABLE department(
 id INT primary key,
 name varchar(100) NOT NULL,
 gender varchar(50) NOT NULL,
 city varchar(20) NOT NULL,
 salary int NOT NULL
 );

INSERT INTO department(id,name,gender,city,salary)
VALUES
(1,'Ram Kumar','M','Rajasthan',12000),
(2,'Neeraj Singh','M','MP',15000),
(3,'Devansh Sharma','M','Delhi',30000),
(4,'Rahul Kalia','M','UP',40000),
(5,'Akshat Jain','M','UP',50000);
```

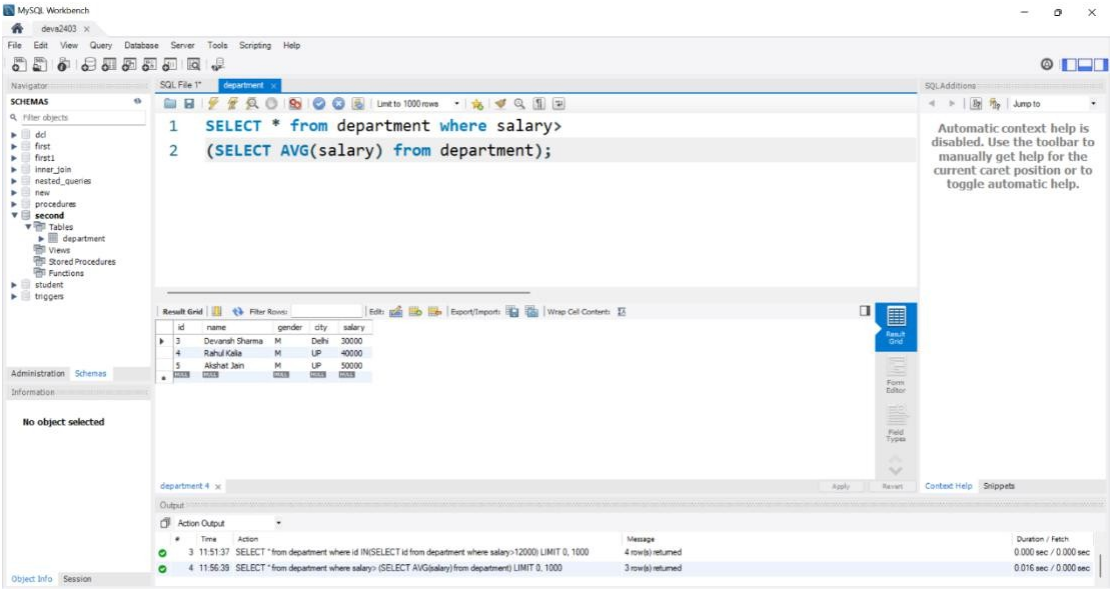
QUERY 1:-

```
SELECT * from department where id
IN(SELECT id from department where salary>12000);
```



QUERY 1:-

SELECT * from department where salary>
(SELECT AVG(salary) from department);



Find the name and salary of the employee with maximum salary:-

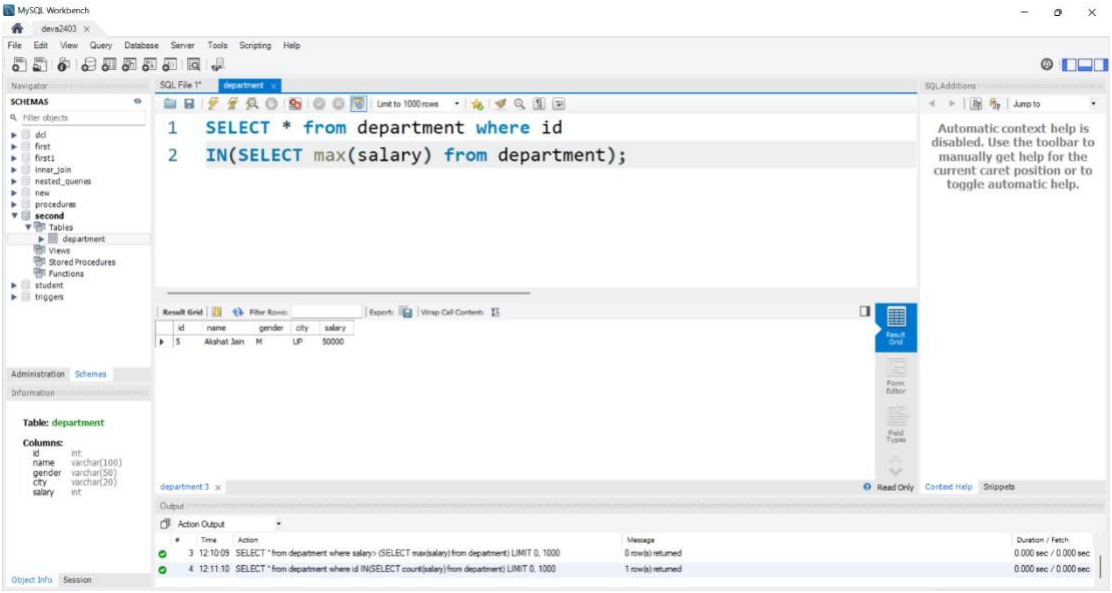
CREATING TABLE

```
CREATE TABLE department(
 id INT,
 name varchar(100),
 gender varchar(50),
 city varchar(20),
 salary int
 );
```

INSERT INTO department(id,name,gender,city,salary)
VALUES
(1,'Ram Kumar','M','Rajasthan',12000),
(2,'Neeraj Singh','M','MP',15000),
(3,'Devansh Sharma','M','Delhi',30000),
(4,'Rahul Kalia','M','UP',40000),
(5,'Akshat Jain','M','UP',50000);

QUERY:-

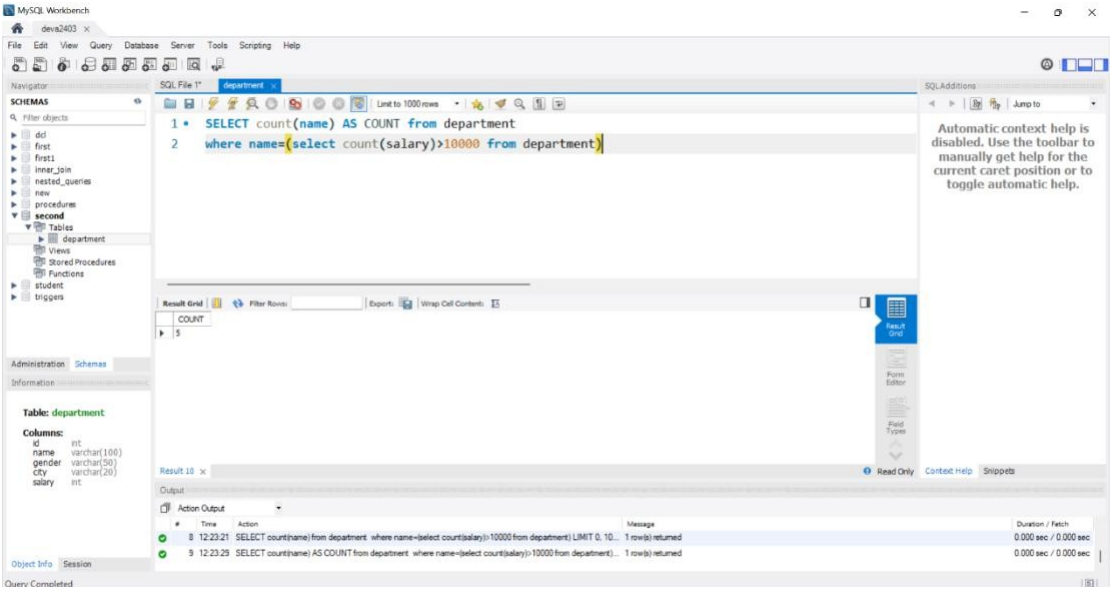
SELECT * from department where id
IN(SELECT max(salary) from department);



Find the count of employees for each job so that at least two of the employees had salary greater than 10000:-

QUERY:-

SELECT count(name) AS COUNT from department where name=(select count(salary)>10000 from department)



Exp No:-7 Date:-

EXPERIMENT:-Join Queries

Aim:-To Execute Joins in MySQL

Theory:-

MySQL JOINS are used with SELECT statement. It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.

There are three types of MySQL joins:

- MySQL INNER JOIN (or sometimes called simple join)
- MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
- MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)

1.MySQL Inner JOIN (Simple Join):-

The MySQL INNER JOIN is used to return all rows from multiple tables where the join condition is satisfied. It is the most common type of join.

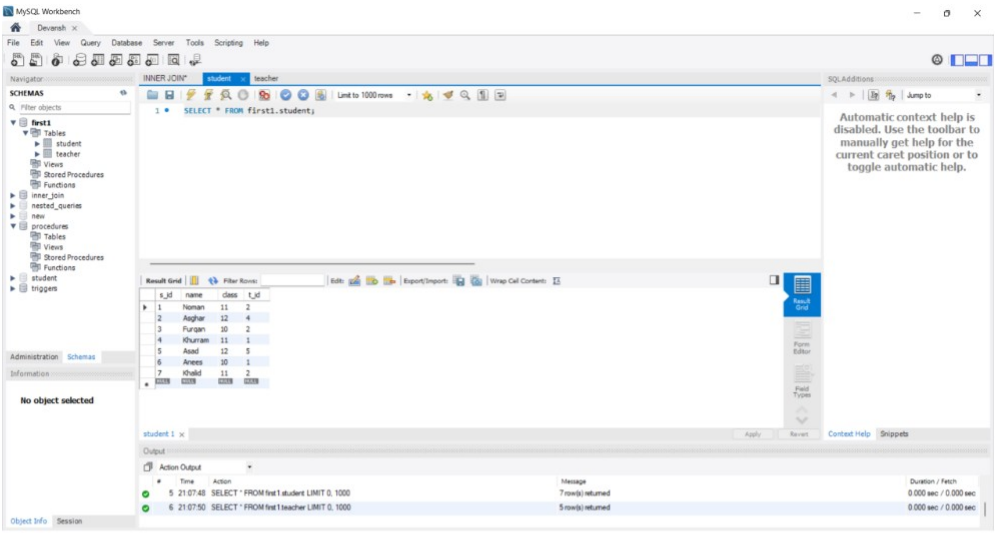
QUERY:-

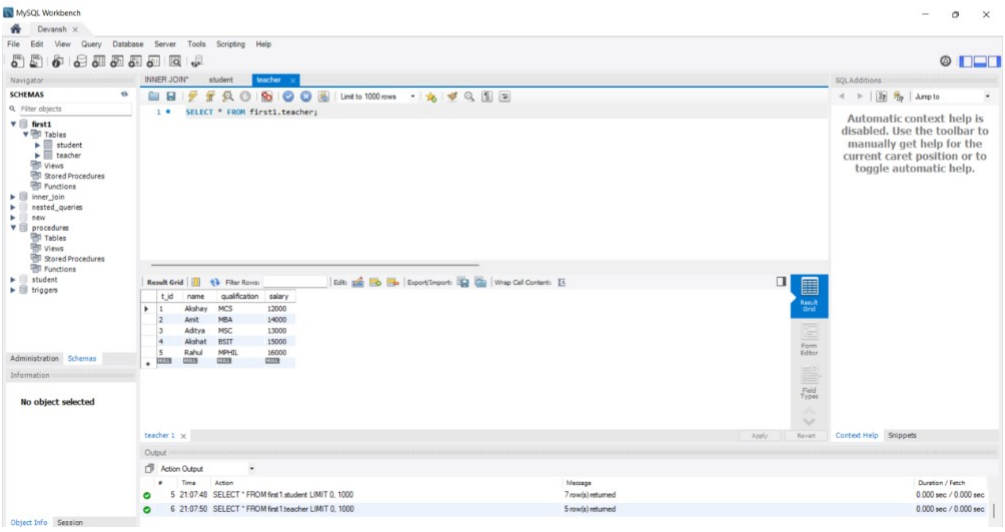
```
CREATE TABLE teacher(
t_id int primary key,
name varchar(50) not null,
qualification varchar(50) not null,
salary int not null
);

INSERT INTO teacher
VALUES
(1,'Akshay','MCS',12000),
(2,'Amit','MBA',14000),
(3,'Aditya','MSC',13000),
(4,'Akshat','BSIT',15000),
(5,'Rahul','MPHIL',16000);

CREATE TABLE student(
s_id int primary key,
name varchar(50) not null,
class int not null,
t_id int not null
);
```

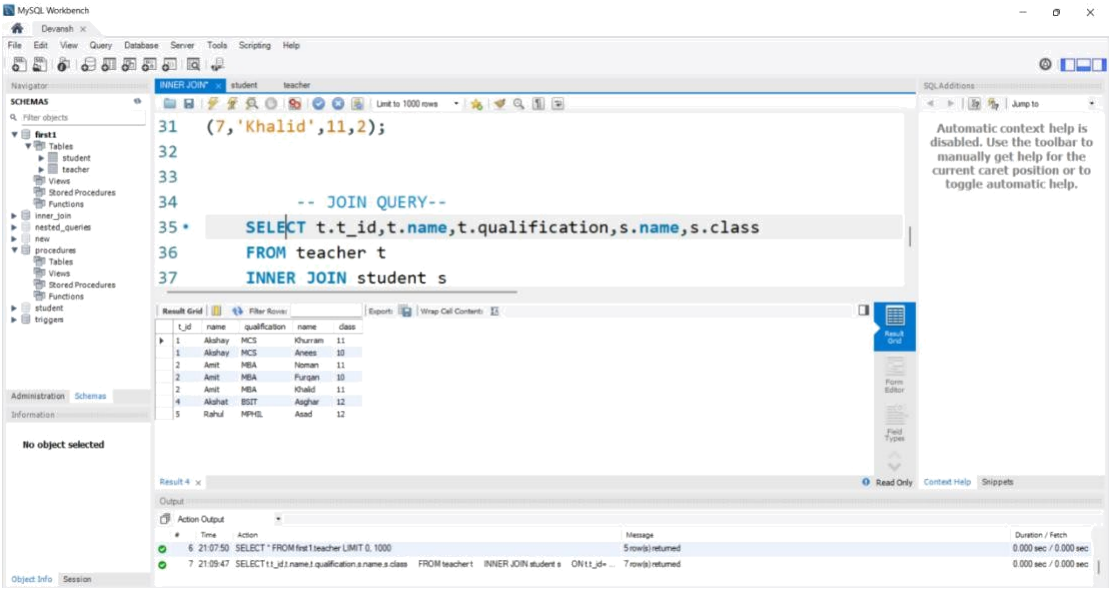
```
INSERT INTO student
VALUES
(1,'Noman',11,2),
(2,'Asghar',12,4),
(3,'Furqan',10,2),
(4,'Khurram',11,1),
(5,'Asad',12,5),
(6,'Anees',10,1),
(7,'Khalid',11,2);
```

-- INNER JOIN QUERY--

```
SELECT t.t_id,t.name,t.qualification,s.name,s.class
FROM teacher t
INNER JOIN student s
ON t.t_id= s.t_id
ORDER BY t_id,t.name;
```







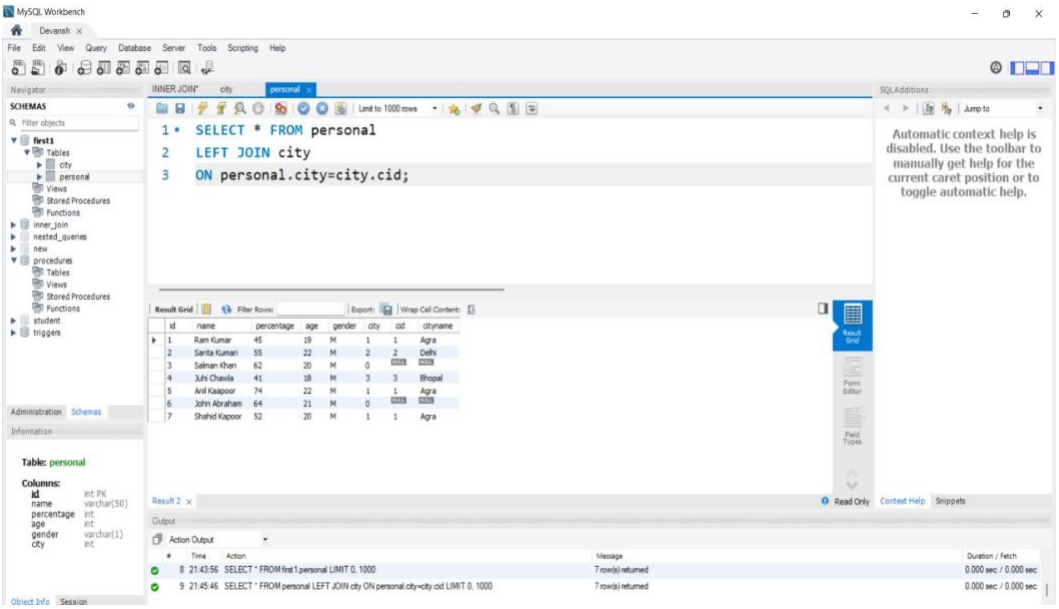
2.MySQL Left Outer Join:-

The LEFT OUTER JOIN returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

QUERY:-

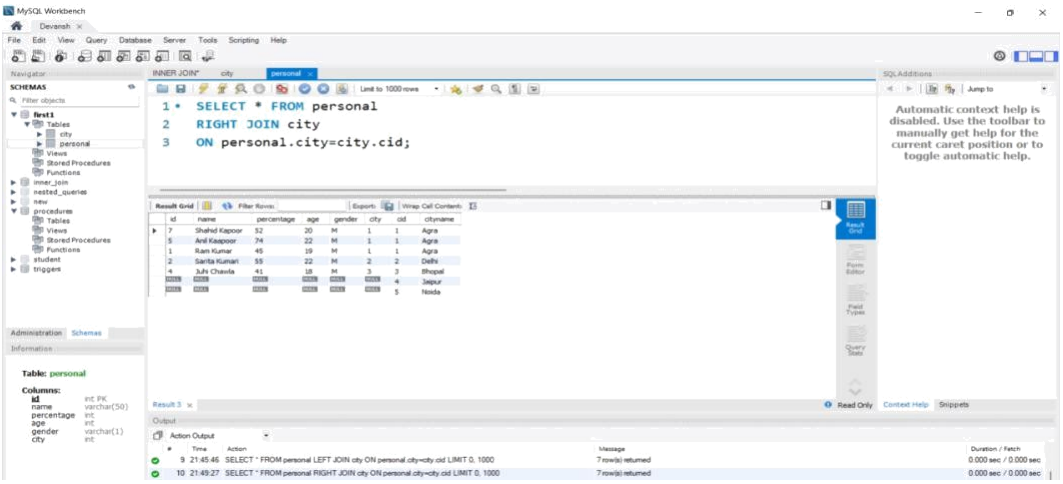
```
CREATE TABLE city(
cid INT NOT NULL AUTO_INCREMENT,
cityname VARCHAR(50) NOT NULL,
PRIMARY KEY(cid)
);

INSERT INTO city(cityname)
VALUES
('Agra'),
('Delhi'),
('Bhopal'),
('Jaipur'),
('Noida');


CREATE TABLE personal(
id INT NOT NULL,
name VARCHAR(50) NOT NULL,
percentage INT NOT NULL,
age INT NOT NULL,
gender VARCHAR(1) NOT NULL,
city INT NOT NULL,
PRIMARY KEY(id),
```

```
FOREIGN KEY(city) REFERENCES City(cid)
);

INSERT INTO personal(id,name,percentage,age,gender,city)
VALUES
(1,'Ram Kumar',45,19,"M",1),
(2,'Sarita Kumari',55,22,"M",2),
(3,'Salman Khan',62,20,"M",1),
(4,'Juhi Chawla',41,18,"M",3),
(5,'Anil Kaapoor',74,22,"M",1),
(6,'John Abraham',64,21,"M",2),
(7,'Shahid Kapoor',52,20,"M",1);


SELECT * FROM personal
LEFT JOIN city
ON personal.city=city.cid;
```



3.MySQL Right Outer Join:-

The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled.

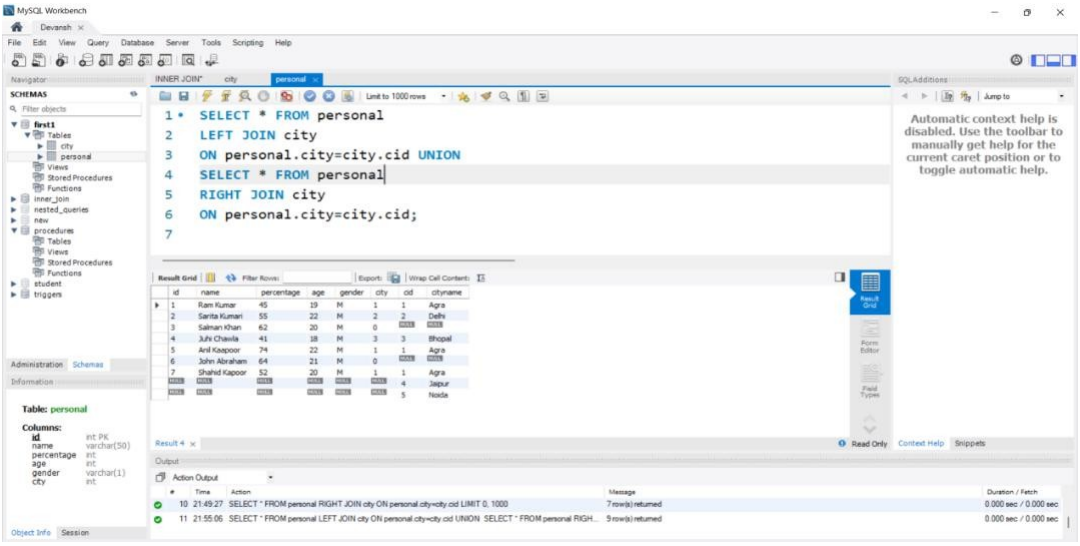


4.SQL FULL OUTER JOIN Keyword:-

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

QUERY:-

```
SELECT * FROM personal
LEFT JOIN city
ON personal.city=city.cid UNION
SELECT * FROM personal
RIGHT JOIN city
ON personal.city=city.cid;
```

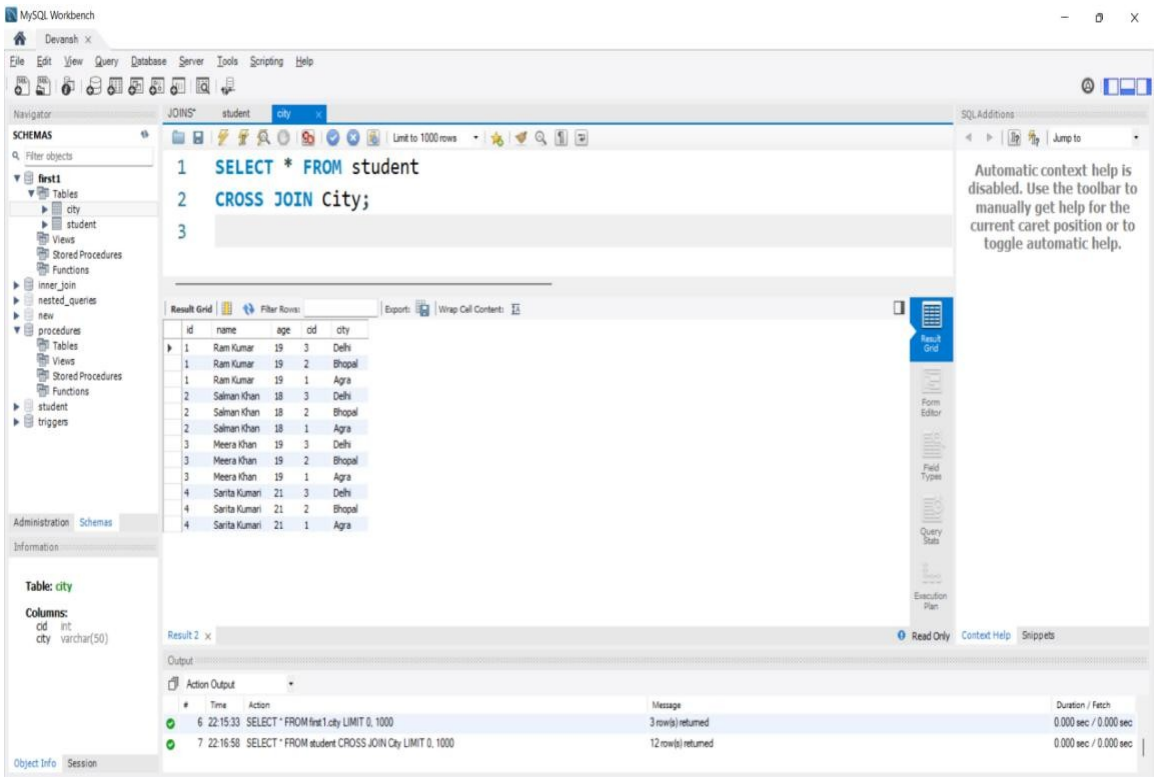


5.SQL CROSS JOIN Keyword

The CROSS JOIN keyword returns all records from both tables (table1 and table2).

QUERY:-

```
SELECT * FROM student
CROSS JOIN City;
```



Exp No:-8 Date:-

EXPERIMENT:-Set Operators & Views

Aim:- To execute Set Operators and Views in Mysql

Theory:- SQL supports few Set operations which can be performed on the table data. These are used to get meaningful results from data stored in the table, under different special conditions.

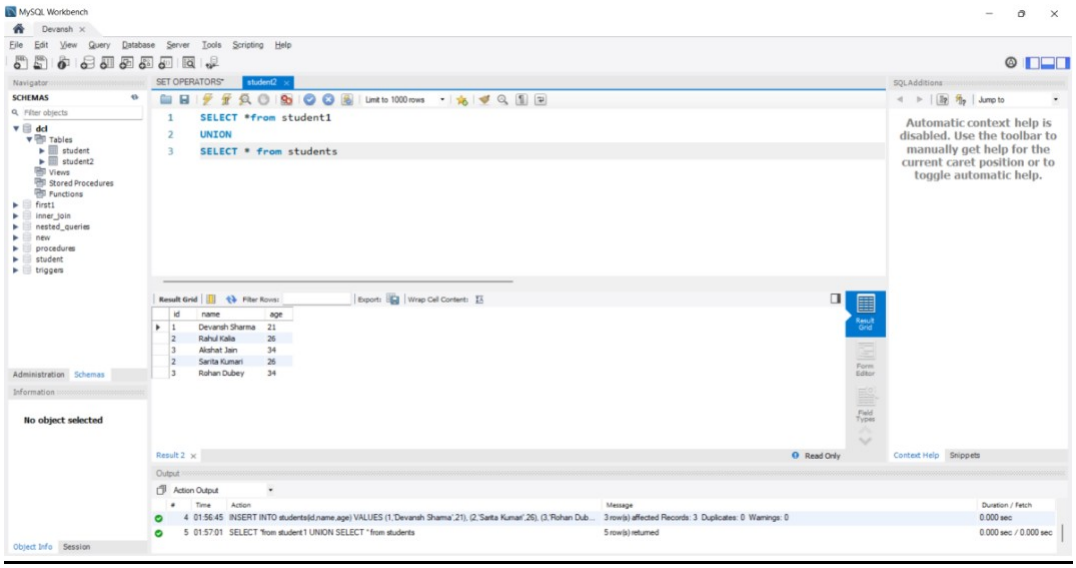
SET OPERATORS:-
a) Union
b) Union all
c) Intersect
d) Minus

CREATING TABLES FOR SET OPERATORS(UNION and UNION ALL):-

```
CREATE TABLE student1(
id INT,
name VARCHAR(255),
age INT
);

INSERT INTO student1(id,name,age)
VALUES
(1,'Devansh Sharma',21),
(2,'Rahul Kalia',26),
(3,'Akshat Jain',34);

CREATE TABLE students(
id INT,
name VARCHAR(255),
age INT
);

INSERT INTO students(id,name,age)
VALUES
(1,'Devansh Sharma',21),
(2,'Sarita Kumari',26),
(3,'Rohan Dubey',34);
```

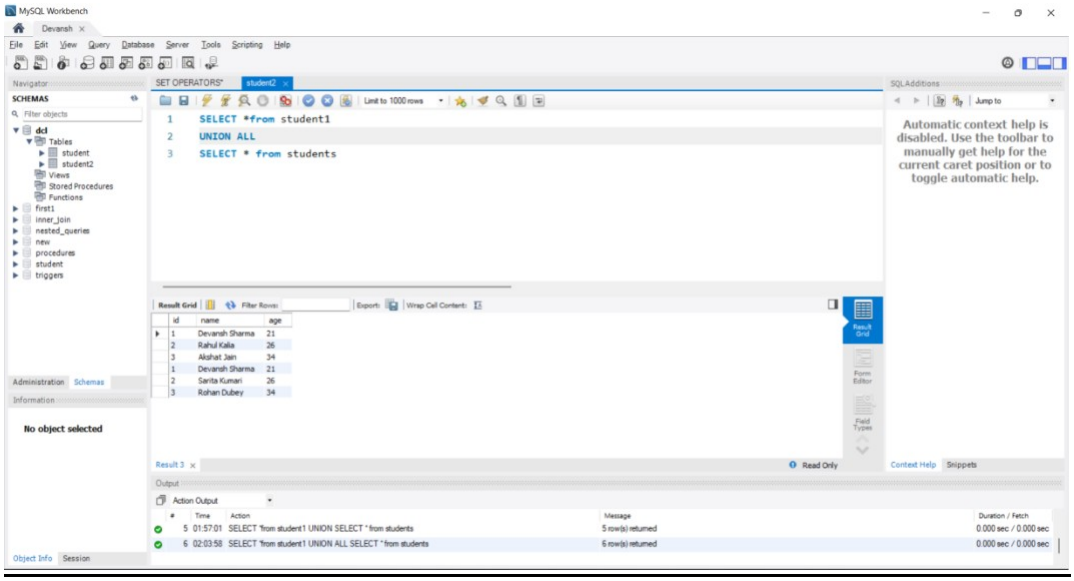
a) Union:-
Query:-

```
SELECT *from student1
UNION
SELECT * from students
```



b) <u>Union All:-</u>

SELECT *from student1
UNION ALL
SELECT * from students



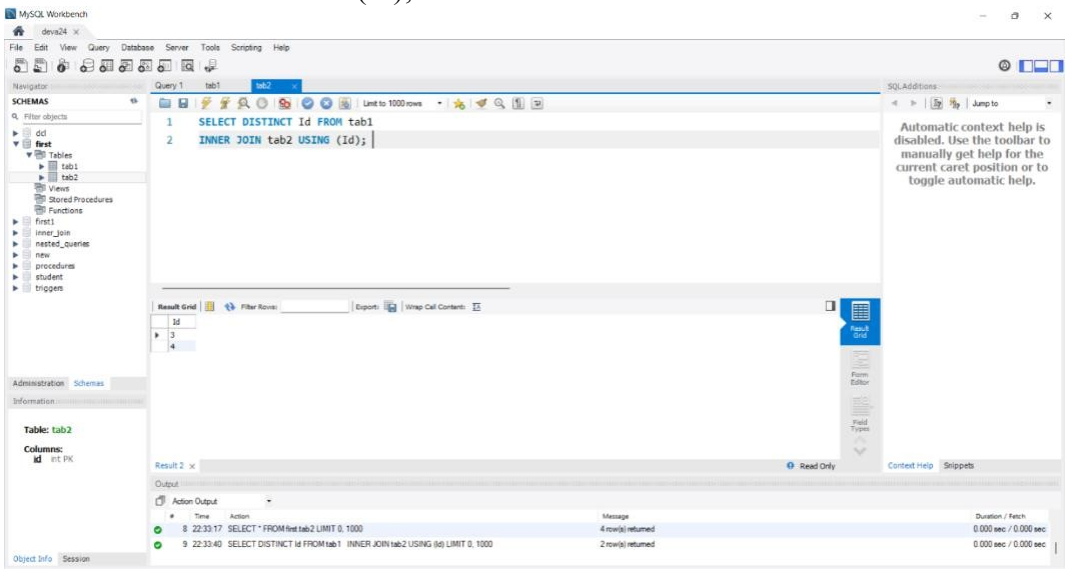
c) INTERSECT:-
<u>Query</u>:-
FOR CREATING TABLE

```
CREATE TABLE tab1 (
    Id INT PRIMARY KEY
);
   INSERT INTO tab1 VALUES (1), (2), (3), (4);
```

```
  CREATE TABLE tab2 (
   id INT PRIMARY KEY
);
  INSERT INTO tab2 VALUES (3), (4), (5), (6);
```

FOR EXECUTION

```
SELECT DISTINCT Id FROM tab1
INNER JOIN tab2 USING (Id);
```



d) MINUS:-

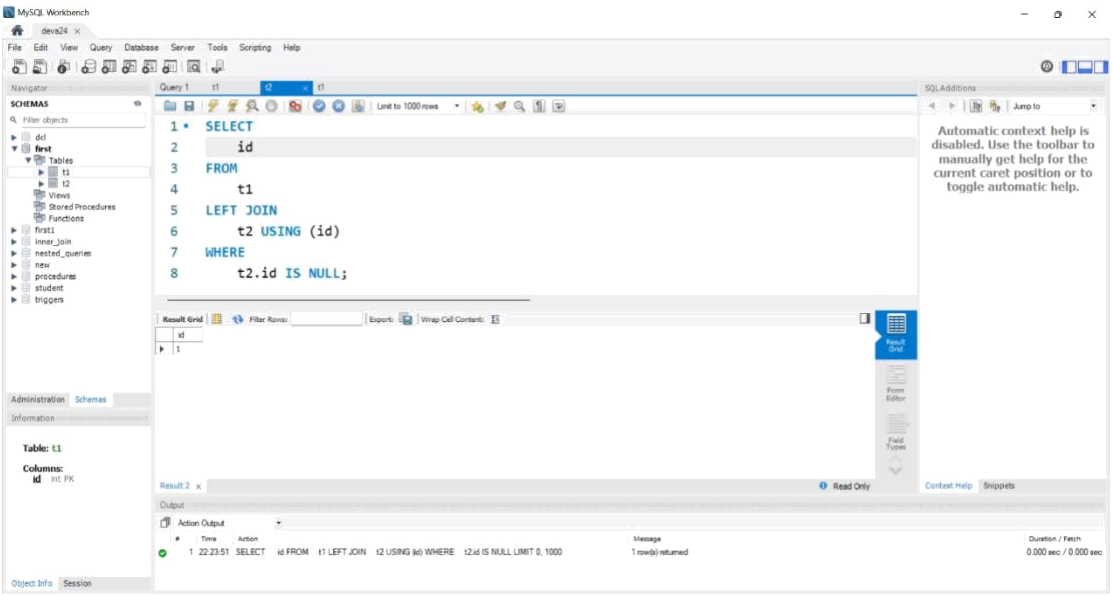
Query:-

FOR CREATING TABLE

```
CREATE TABLE t1 (
   id INT PRIMARY KEY
);
CREATE TABLE t2 (
   id INT PRIMARY KEY
);
INSERT INTO t1 VALUES (1),(2),(3);
INSERT INTO t2 VALUES (2),(3),(4);
```

FOR EXECUTION

```
SELECT
   id
FROM
   t1
LEFT JOIN
   t2 USING (id)
WHERE
   t2.id IS NULL;
```



VIEWS:-

Theory:-

In SQL, a view 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.

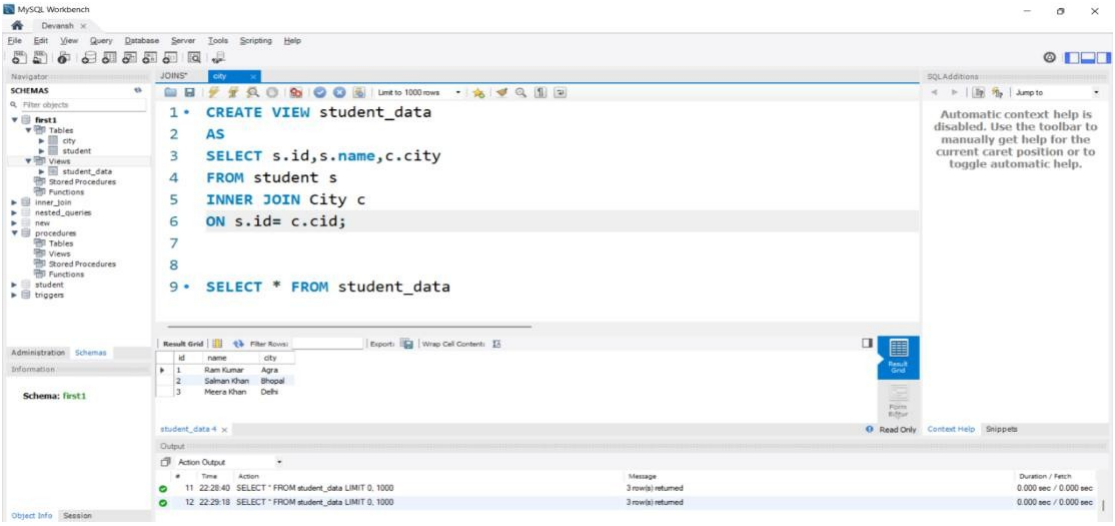
You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

A view is created with the CREATE VIEW statement.

QUERY:-

```
CREATE VIEW student_data

AS

SELECT s.id,s.name,c.city

FROM student s

INNER JOIN City c

ON s.id= c.cid;
```

SELECT * FROM student_data

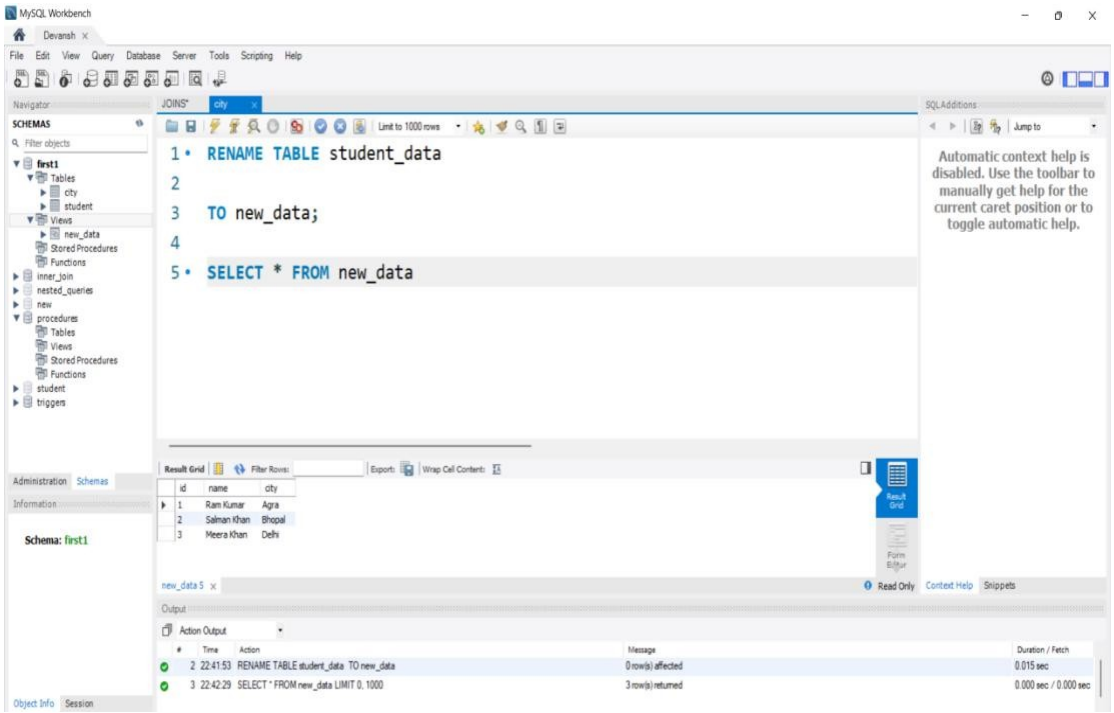


IF we rename the existing view then we use RENAME Command:-

RENAME TABLE student_data

TO new_data;

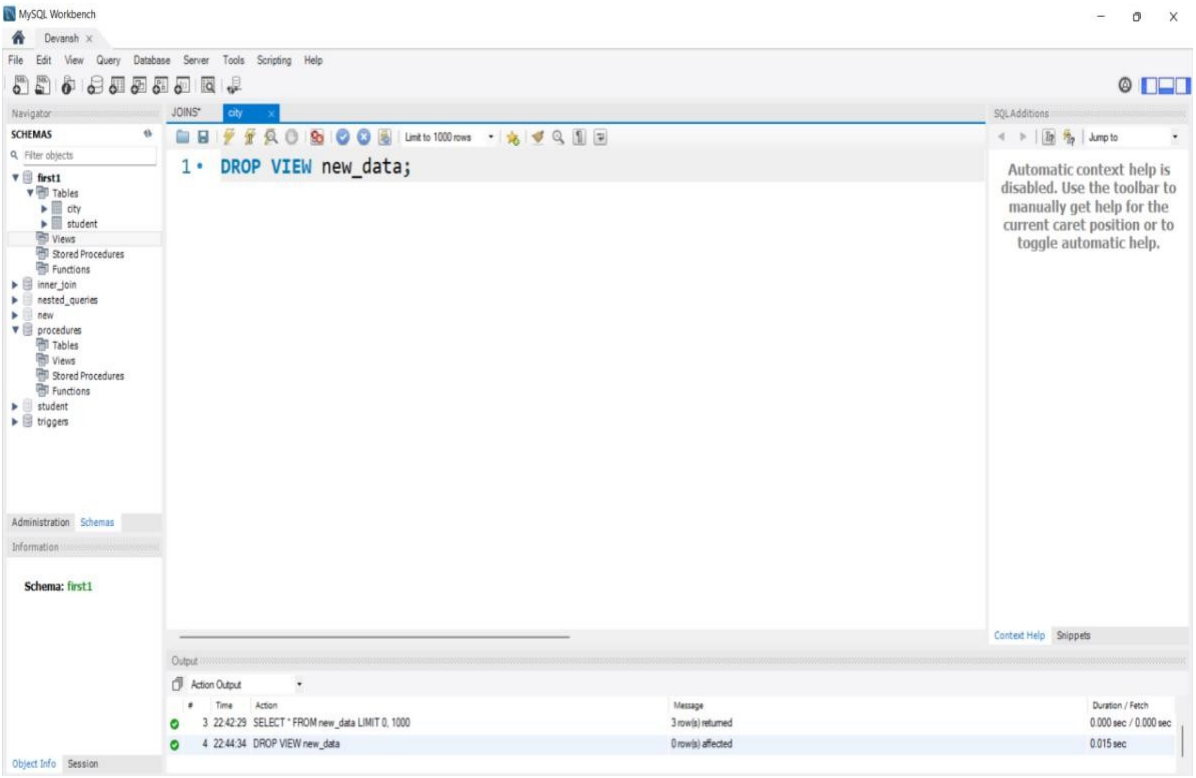
SELECT * FROM new_data



IF we DELETE the VIEW then we use DROP VIEW:-

QUERY:-

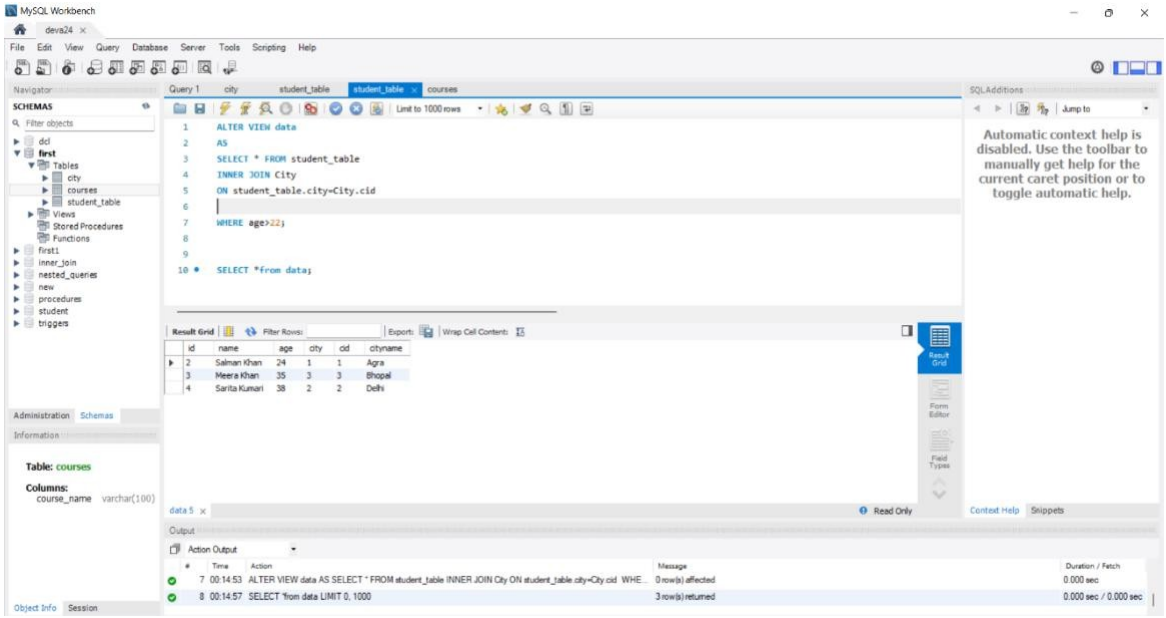
DROP VIEW new_data;



ALTER IN VIEWS

QUERY:-

```
ALTER VIEW data
AS
SELECT * FROM student_table
INNER JOIN City
ON student_table.city=City.cid
WHERE age>22;
SELECT *from data;
```



Exp No:-9 Date:-

EXPERIMENT:-Conditional &Iterative Statements

AIM – PL/SQL conditional and iterative statements.

Theory:- A program that supports a user interface may run a main loop that waits for, and then processes, user keystrokes (this doesn't apply to stored programs, however).Many mathematical algorithms can be implemented only by loops in computer programs.When processing a file, a program may loop through each record in the file and perform computations.A database program may loop through the rows returned by a SELECT statement.

QUERY

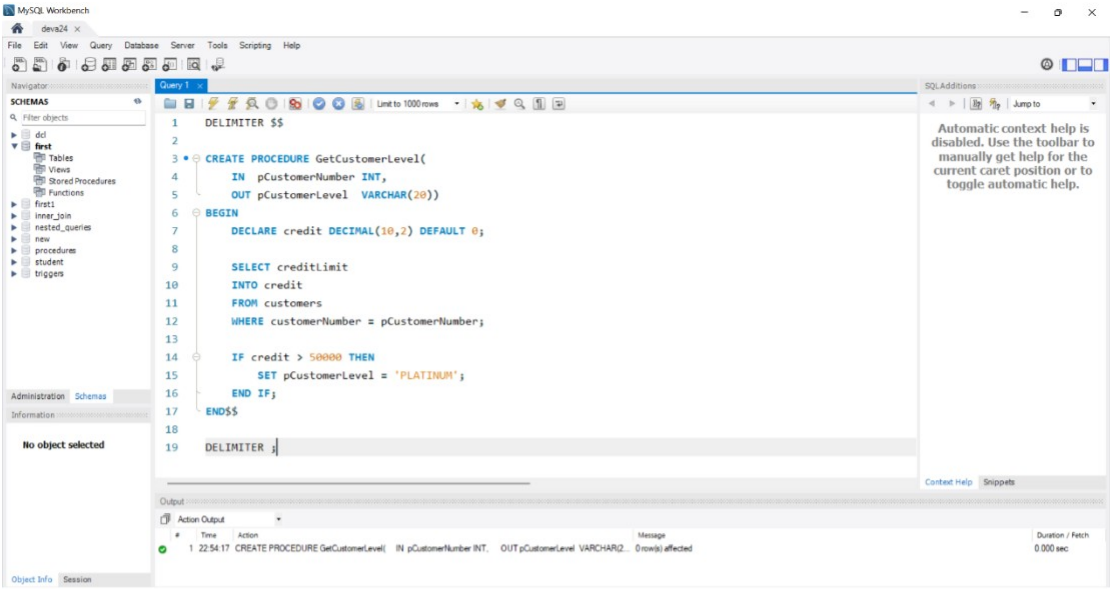
a)Conditional Statement

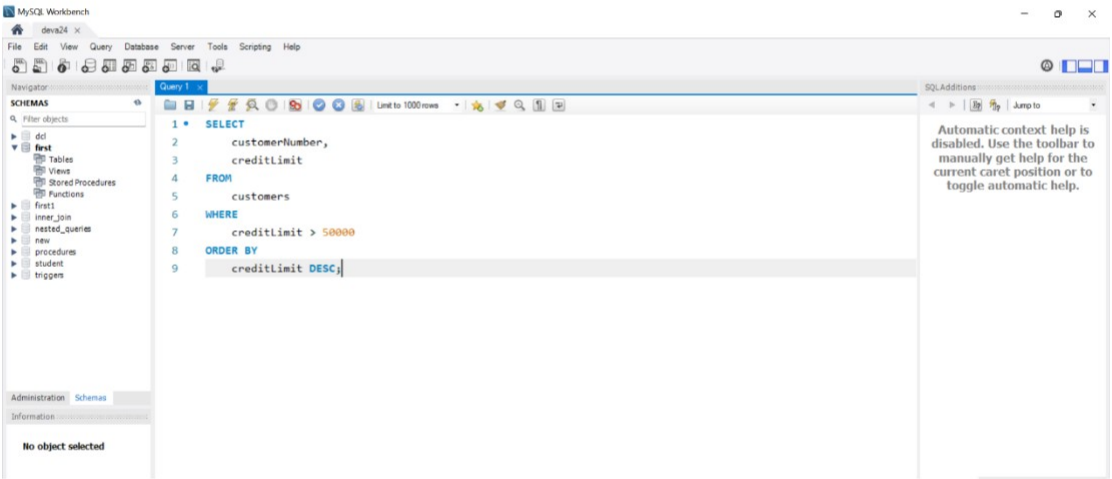
FOR CREATING PROCEDURE

```
DELIMITER $$
CREATE PROCEDURE
GetCustomerLevel(
   IN pCustomerNumber INT,
   OUT pCustomerLevel  VARCHAR(20))
BEGIN
   DECLARE credit DECIMAL(10,2)
DEFAULT 0;
SELECT creditLimit
   INTO credit
   FROM customers
   WHERE customerNumber =
pCustomerNumber;
IF credit > 50000 THEN
      SET pCustomerLevel = 'PLATINUM';
   END IF;
END$$
DELIMITER ;
```

FOR EXECUTION

```
SELECT
   customerNumber,
   creditLimit
FROM
   customers
WHERE
   creditLimit > 50000
ORDER BY
   creditLimit DESC;
```





customerNumber	creditLimit
141	227600.00
124	210500.00
298	141300.00
151	138500.00
187	136800.00
146	123900.00
286	123700.00
386	121400.00
227	120800.00
259	120400.00

b)Iterative Statement

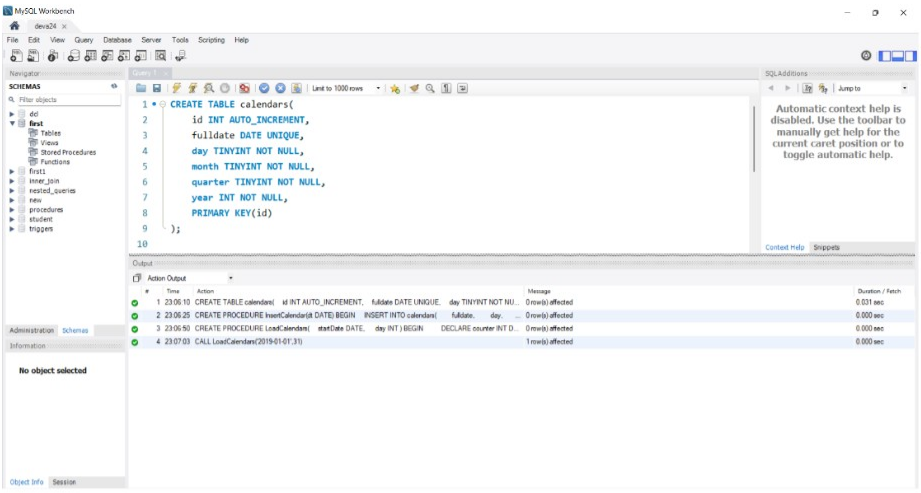
```
CREATE TABLE calendars(
   id INT AUTO_INCREMENT,
   fulldate DATE UNIQUE,
   day TINYINT NOT NULL,
   month TINYINT NOT NULL,
   quarter TINYINT NOT NULL,
   year INT NOT NULL,
   PRIMARY KEY(id)
);
```

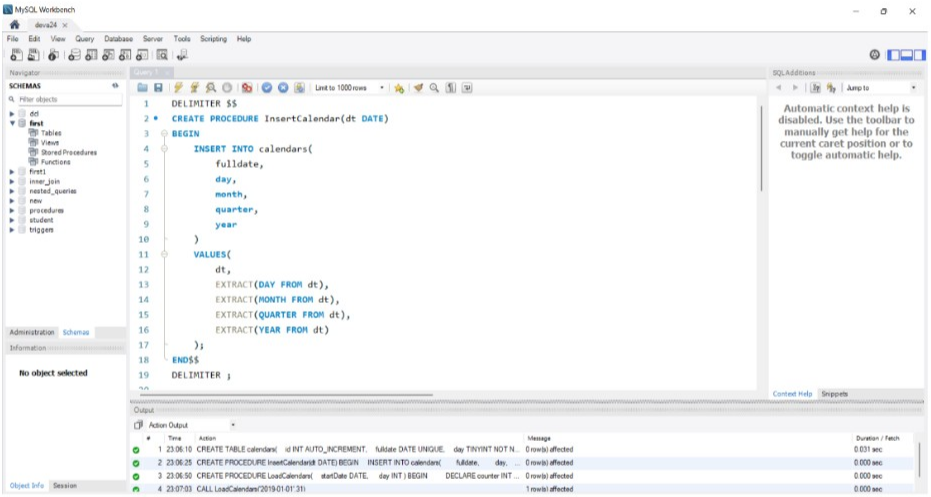
FOR CREATING PROCEDURE

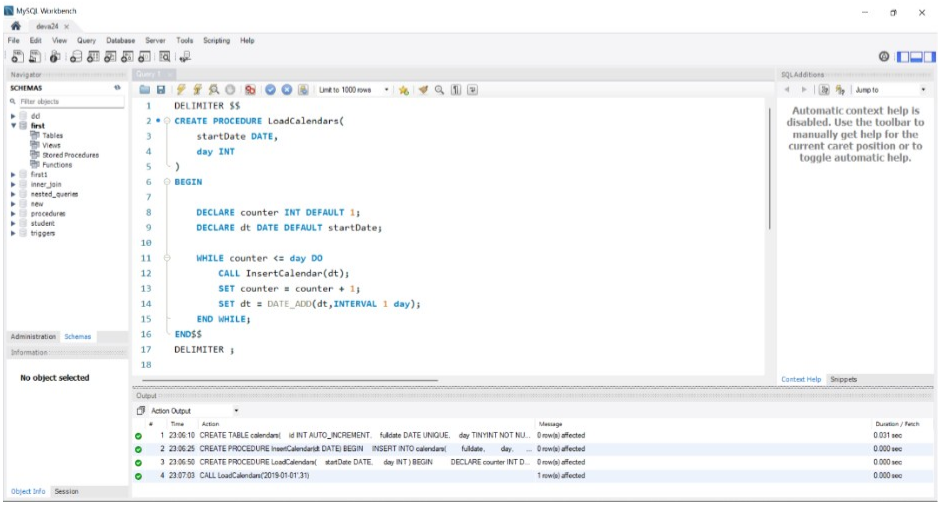
```
DELIMITER $$
CREATE PROCEDURE InsertCalendar(dt
DATE)
BEGIN
   INSERT INTO calendars(
      fulldate,
      day,
      month,
      quarter,
      year
   )
   VALUES(
      dt,
      EXTRACT(DAY FROM dt),
      EXTRACT(MONTH FROM dt),
      EXTRACT(QUARTER FROM dt),
      EXTRACT(YEAR FROM dt)
   );
END$$
DELIMITER ;

DELIMITER $$
CREATE PROCEDURE LoadCalendars(
   startDate DATE,
   day INT
)
BEGIN
   DECLARE counter INT DEFAULT 1;
   DECLARE dt DATE DEFAULT
startDate;
WHILE counter <= day DO
      CALL InsertCalendar(dt);
      SET counter = counter + 1;
      SET dt =
DATE_ADD(dt,INTERVAL 1 day);
END WHILE;
END$$
DELIMITER ;
```

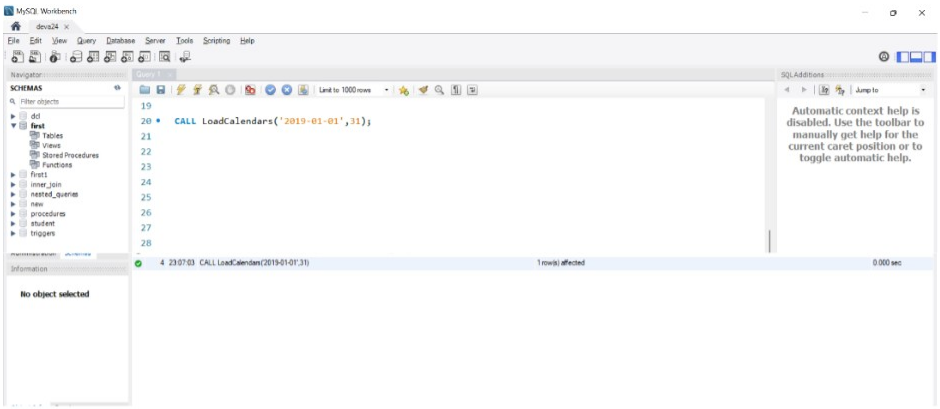
FOR EXECUTION

CALL LoadCalendars('2019-01-01',31);









id	fulldate	day	month	quarter	year
1	2019-01-01	1	1	1	2019
2	2019-01-02	2	1	1	2019
3	2019-01-03	3	1	1	2019
4	2019-01-04	4	1	1	2019
5	2019-01-05	5	1	1	2019
6	2019-01-06	6	1	1	2019
7	2019-01-07	7	1	1	2019
8	2019-01-08	8	1	1	2019
9	2019-01-09	9	1	1	2019
10	2019-01-10	10	1	1	2019
11	2019-01-11	11	1	1	2019
12	2019-01-12	12	1	1	2019
13	2019-01-13	13	1	1	2019
14	2019-01-14	14	1	1	2019
15	2019-01-15	15	1	1	2019

Exp No:-10 Date:-

EXPERIMENT:-Procedures

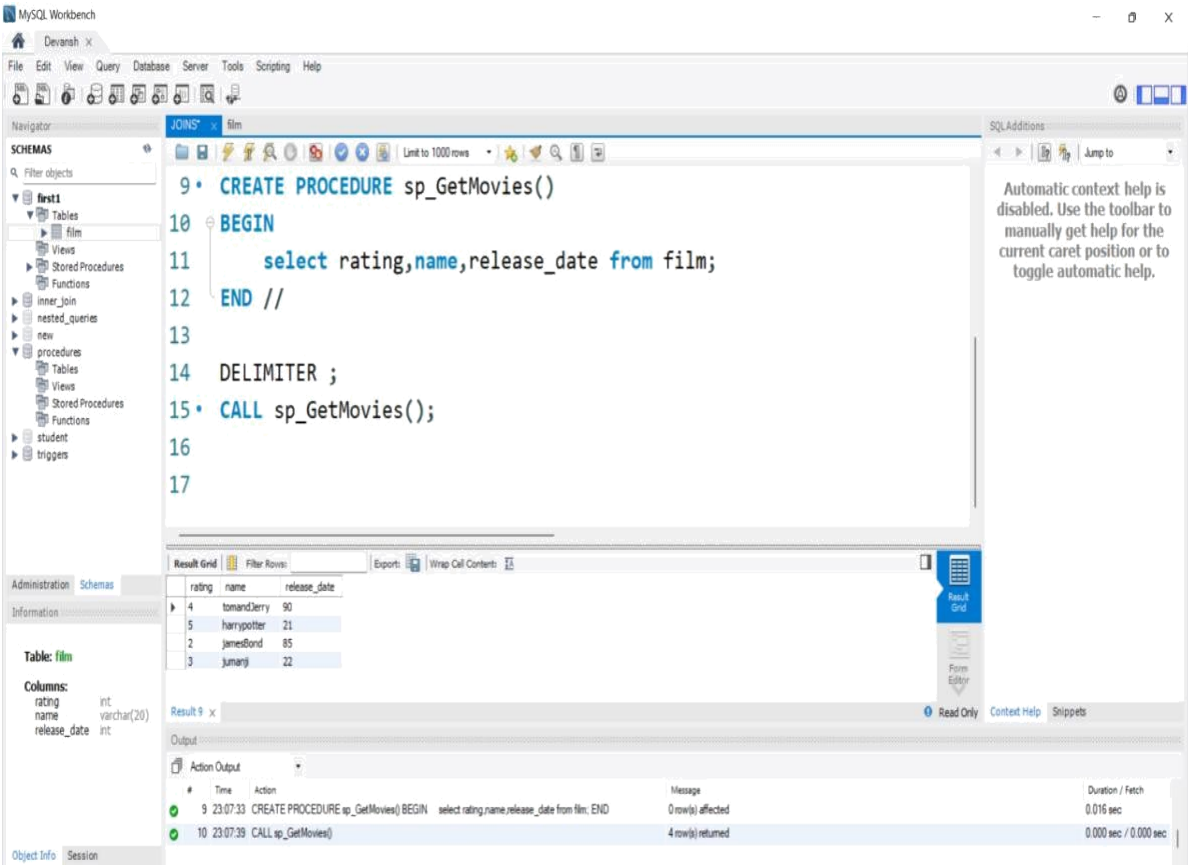
Aim:- To Execute Procedure in MySql

Theory:- A procedure (often called a stored procedure) is a collection of pre-compiled SQL statements stored inside the database. It is a subroutine or a subprogram in the regular computing language. A procedure always contains a name, parameter lists, and SQL statements. We can invoke the procedures by using triggers, other procedures and applications such as Java, Python, PHP, etc. It was first introduced in MySQL version 5. Presently, it can be supported by almost all relational database systems.

If we consider the enterprise application, we always need to perform specific tasks such as database cleanup, processing payroll, and many more on the database regularly. Such tasks involve multiple SQL statements for executing each task. This process might easy if we group these tasks into a single task. We can fulfill this requirement in MySQL by creating a stored procedure in our database.

Query:-

```
create table film(rating int ,name varchar(20),release_date int);
insert into film values(4,'tomandJerry',90); insert into film
values(5,'harrypotter',21);
insert into film values(2,'jamesBond',85);
insert into film values(3,'jumanji',22);


DELIMITER //
CREATE PROCEDURE sp_GetMovies()
BEGIN
   select rating,name,release_date from film;
END //

DELIMITER ;

CALL sp_GetMovies();
```


MySQL Workbench
Devansh
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
first1
Tables
film
Views
Stored Procedures
Functions
inner_join
nested_queries
new
procedures
Tables
Views
Stored Procedures
Functions
student
triggers
Administration Schemas
Information
Table: film
Columns:
rating int
name varchar(20)
release_date int
Object Info Session
JOINS* film
Limit to 1000 rows
9 • CREATE PROCEDURE sp_GetMovies()
10 BEGIN
11 select rating,name,release_date from film;
12 END //
13
14 DELIMITER ;
15 • CALL sp_GetMovies();
16
17
Result Grid Filter Rows: Export: Wrap Cell Content:
rating name release_date
4 tomandJerry 90
5 harrypotter 21
2 jamesBond 85
3 jumanji 22
Result 9
Read Only
Output
Action Output
Time Action Message Duration / Fetch
9 23:07:33 CREATE PROCEDURE sp_GetMovies() BEGIN select rating,name,release_date from film; END 0 row(s) affected 0.016 sec
10 23:07:39 CALL sp_GetMovies() 4 row(s) returned 0.000 sec / 0.000 sec
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets

Exp No:-11 Date:-

EXPERIMENT:-PL/SQL Functions

AIM - PL/SQL functions

Theory:- A function can be used as a part of SQL expression i.e. we can use them with select/update/merge commands. One most important characteristic of a function is that unlike procedures, it must return a value.

QUERY

CREATING TABLE FOR FUNCTION

```
CREATE TABLE employee(
emp_id INT,
fname varchar(50),
lname varchar(50),
start_date date
);
INSERT INTO
employee(emp_id,fname,lname,start_date)
VALUES
(1,'Michael','Smith','2001-06-22'),
(2,'Susan', 'Barker','2002-09-12'),
(3,'Robert','Tvler','2000-02-09'),
(4,'Susan','Hawthorne','2002-04-24');
```

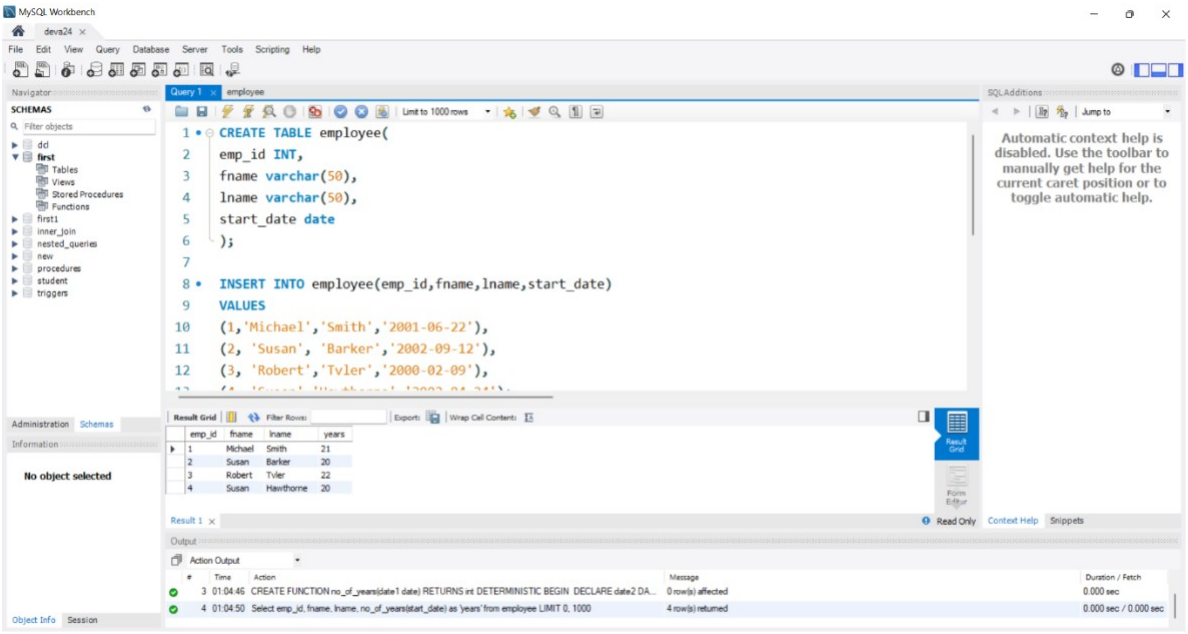
CREATING FUNCTION

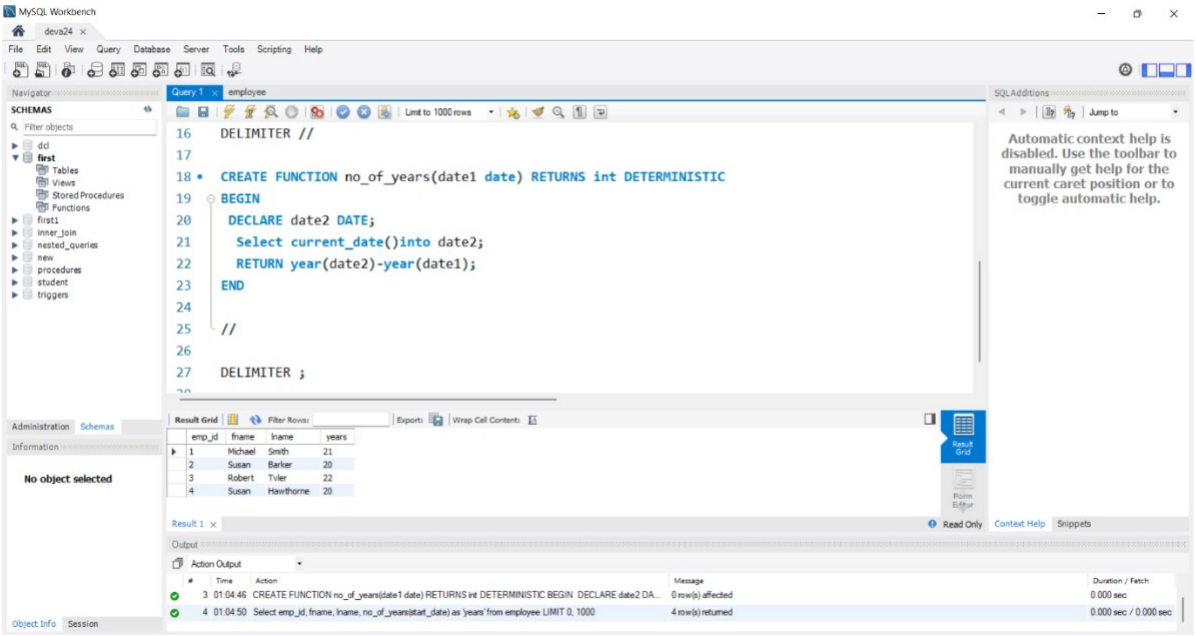
```
DELIMITER //
CREATE FUNCTION no_of_years(date1 date) RETURNS int DETERMINISTIC
BEGIN
 DECLARE date2 DATE;
  Select current_date()into date2;
  RETURN year(date2)-year(date1);
END //
DELIMITER ;
```

CALLING FUNCTION

```
Select emp_id, fname, lname, no_of_years(start_date) as 'years' from employee;
```

OUTPUT:-


MySQL Workbench
deva24
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
dcl
first
Tables
Views
Stored Procedures
Functions
first1
inner_join
nested_queries
new
procedures
student
triggers
Administration Schemas
Information
No object selected
Object Info Session
Query 1 employee
Limit to 1000 rows
1 • CREATE TABLE employee(
2 emp_id INT,
3 fname varchar(50),
4 lname varchar(50),
5 start_date date
6);
7
8 • INSERT INTO employee(emp_id,fname,lname,start_date)
9 VALUES
10 (1,'Michael','Smith','2001-06-22'),
11 (2, 'Susan', 'Barker','2002-09-12'),
12 (3, 'Robert','Tvler','2000-02-09'),
Result Grid Filter Rows: Export: Wrap Cell Content:
emp_id fname lname years
1 Michael Smith 21
2 Susan Barker 20
3 Robert Tvler 22
4 Susan Hawthorne 20
Result 1
Read Only
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
3 01:04:46 CREATE FUNCTION no_of_years(date1 date) RETURNS int DETERMINISTIC BEGIN DECLARE date2 DA... 0 row(s) affected 0.000 sec
4 01:04:50 Select emp_id, fname, lname, no_of_years(start_date) as 'years' from employee LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.


MySQL Workbench
deva24
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
dcl
first
Tables
Views
Stored Procedures
Functions
first1
inner_join
nested_queries
new
procedures
student
triggers
Administration Schemas
Information
No object selected
Object Info Session
Query 1 employee
Limit to 1000 rows
16 DELIMITER //
17
18 • CREATE FUNCTION no_of_years(date1 date) RETURNS int DETERMINISTIC
19 BEGIN
20 DECLARE date2 DATE;
21 Select current_date()into date2;
22 RETURN year(date2)-year(date1);
23 END
24
25 //
26
27 DELIMITER ;
Result Grid Filter Rows: Export: Wrap Cell Content:
emp_id fname lname years
1 Michael Smith 21
2 Susan Barker 20
3 Robert Tvler 22
4 Susan Hawthorne 20
Result 1
Read Only
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
3 01:04:46 CREATE FUNCTION no_of_years(date1 date) RETURNS int DETERMINISTIC BEGIN DECLARE date2 DA... 0 row(s) affected 0.000 sec
4 01:04:50 Select emp_id, fname, lname, no_of_years(start_date) as 'years' from employee LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Exp No:-12 Date:-

EXPERIMENT:-PL/SQL Cursors

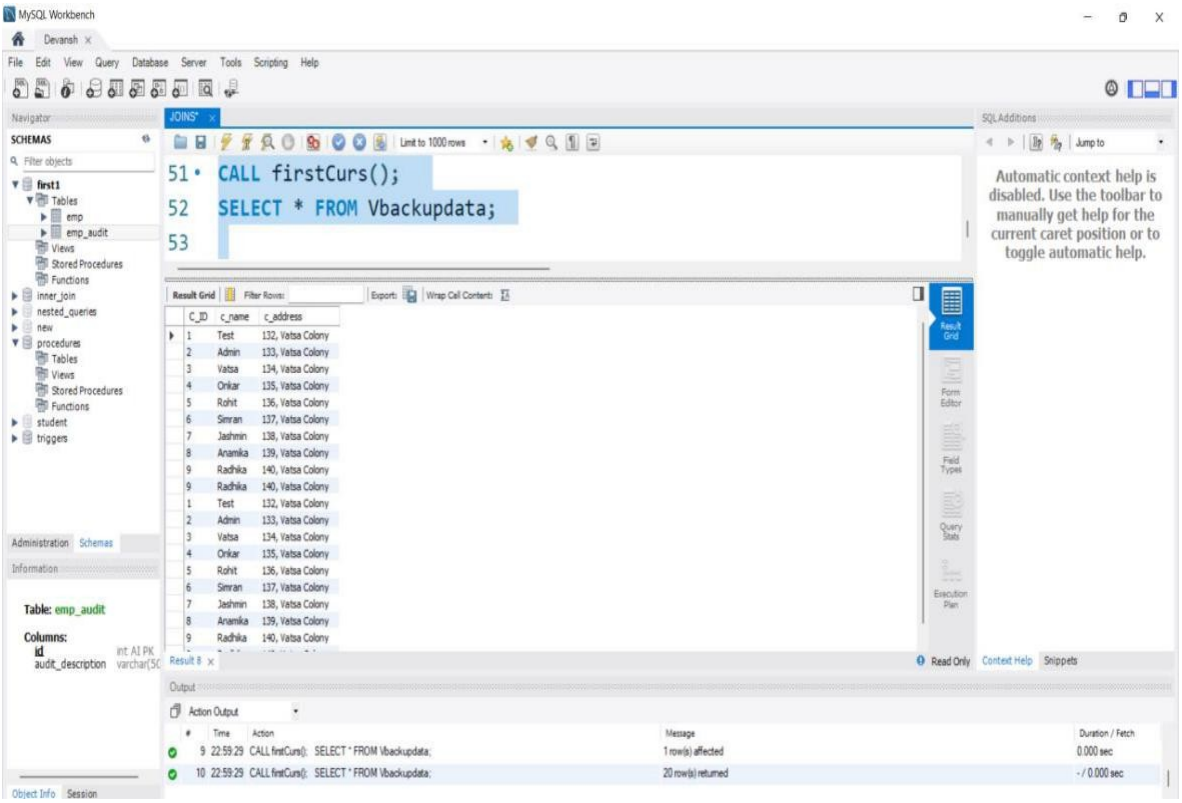
Aim:- To Execute Cursors in MySql

Theory:- A cursor allows you to iterate a set of rows returned by a query and process each row individually. MySQL cursor is read-only, non-scrollable and asensitive. Read-only: you cannot update data in the underlying table through the cursor.

QUERY:-

```
CREATE TABLE GetVatsaCursor(
C_ID INT PRIMARY KEY AUTO_INCREMENT,
c_name VARCHAR(50),
c_address VARCHAR(200));
CREATE TABLE Vbackupdata(
C_ID INT,
 c_name VARCHAR(50),
 c_address VARCHAR(200));
 INSERT INTO GetVatsaCursor(c_name, c_address) VALUES('Test', '132, Vatsa
Colony'),
  ('Admin', '133, Vatsa Colony'),
  ('Vatsa', '134, Vatsa Colony'),
  ('Onkar', '135, Vatsa Colony'),
  ('Rohit', '136, Vatsa Colony'),
  ('Simran', '137, Vatsa Colony'),
  ('Jashmin', '138, Vatsa Colony'),
  ('Anamika', '139, Vatsa Colony'),
  ('Radhika', '140, Vatsa Colony');
  SELECT * FROM GetVatsaCursor;
  SELECT * FROM Vbackupdata;
  delimiter //
  CREATE PROCEDURE firstCurs()
  BEGIN
  DECLARE d INT DEFAULT 0;
  DECLARE c_id INT;
  DECLARE c_name, c_address VARCHAR(20);
  DECLARE Get_cur CURSOR FOR SELECT * FROM GetVatsaCursor;
  DECLARE CONTINUE HANDLER FOR SQLSTATE
  '02000' SET d = 1;
  DECLARE CONTINUE HANDLER FOR SQLSTATE '23000'
  SET d = 1;
  OPEN Get_cur;
  lbl: LOOP
  IF d = 1 THEN
  LEAVE lbl;
```

```
END IF;
IF NOT d = 1 THEN
FETCH Get_cur INTO c_id, c_name, c_address;
INSERT INTO Vbackupdata VALUES(c_id, c_name, c_address);
END IF;
END LOOP;
CLOSE Get_cur;
END;
CALL firstCurs();
SELECT * FROM Vbackupdata
```



Exp No:-13 Date:-

EXPERIMENT:-PL/SQL Exception Handling

Aim- PL/SQL exception handling.

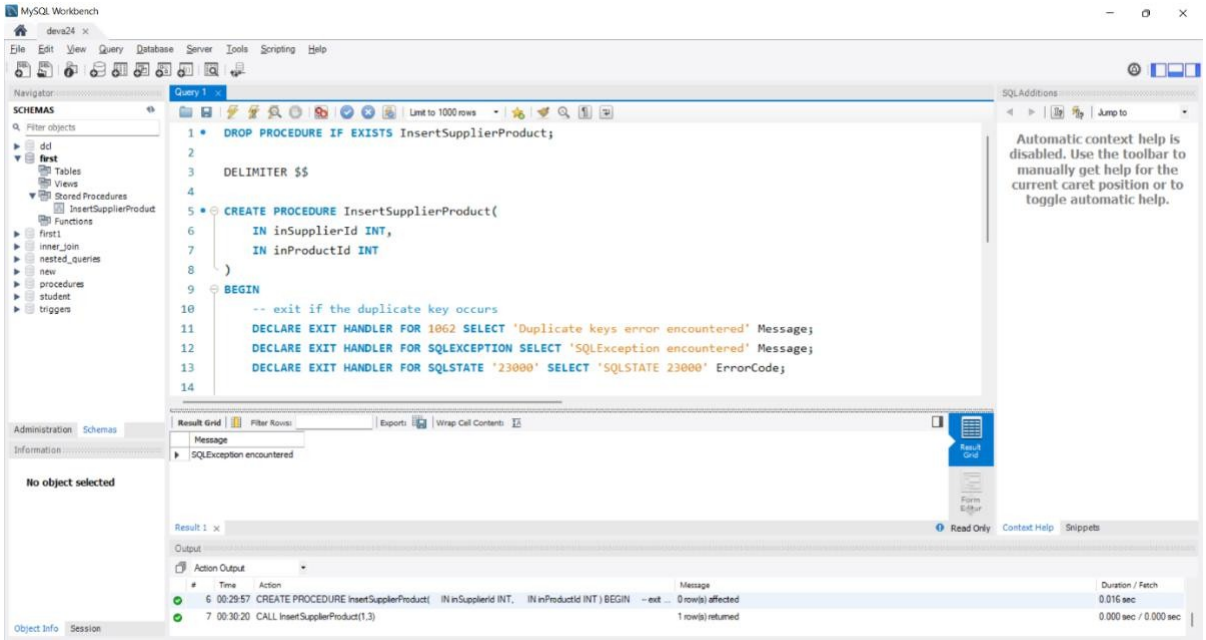
Theory:-When an error occurs inside a stored procedure, it is important to handle it appropriately, such as continuing or exiting the current code block's execution, and issuing a meaningful error message.MySQL provides an easy way to define handlers that handle from general conditions such as warnings or exceptions to specific conditions e.g., specific error codes.

QUERY:-

```
DROP PROCEDURE IF EXISTS InsertSupplierProduct;

DELIMITER $$

CREATE PROCEDURE InsertSupplierProduct(

   IN inSupplierId INT,

   IN inProductId INT
)
BEGIN

   -- exit if the duplicate key occurs

   DECLARE EXIT HANDLER FOR 1062 SELECT

'Duplicate keys error encountered' Message;
DECLARE EXIT HANDLER FOR SQLEXCEPTION

SELECT 'SQLException encountered' Message;
DECLARE EXIT HANDLER FOR SQLSTATE
'23000' SELECT 'SQLSTATE 23000' ErrorCode;

   -- insert a new row into the SupplierProducts

   INSERT INTO SupplierProducts(supplierId,productId)

   VALUES(inSupplierId,inProductId);

   -  return the products supplied by the supplier

   id SELECT COUNT(*)
```

```
FROM SupplierProducts

WHERE supplierId = inSupplierId;

END$$

DELIMITER ;
```

OUTPUT



Exp No:-14 Date:-

EXPERIMENT:-PL/SQL Trigger

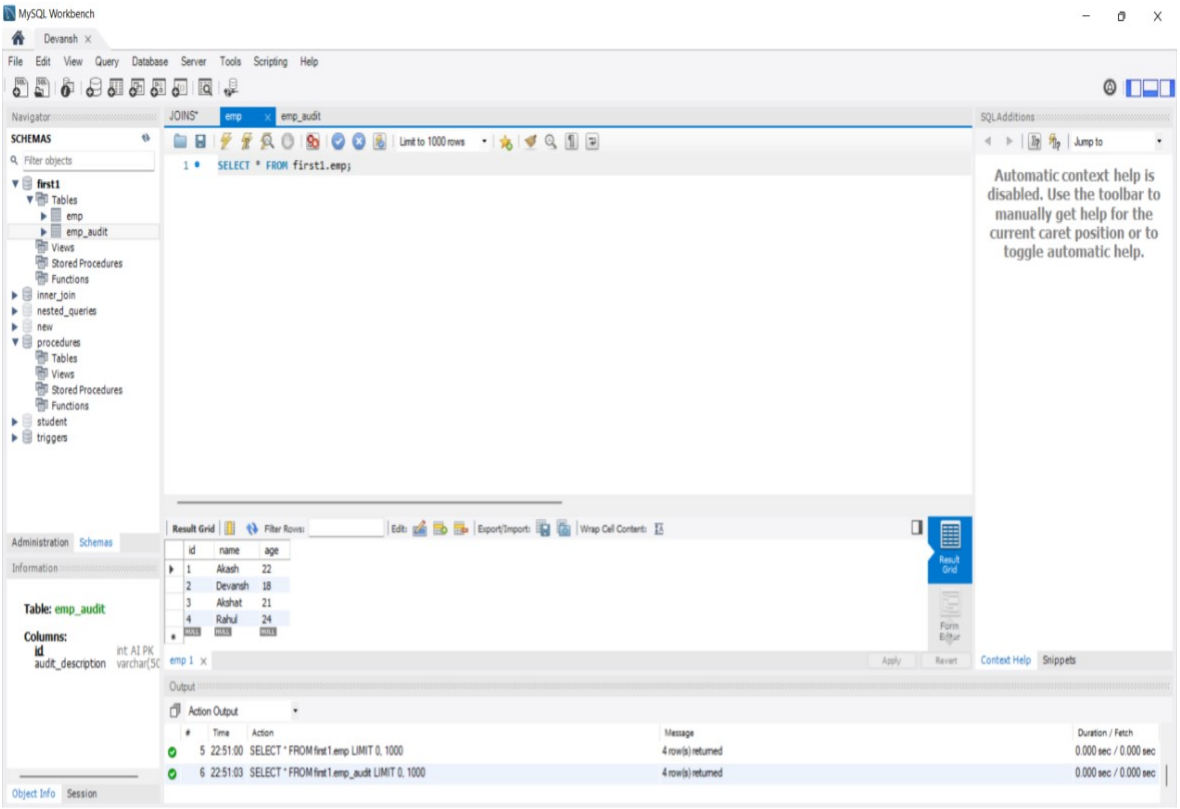
Aim:- To Execute Triggers in MYsql

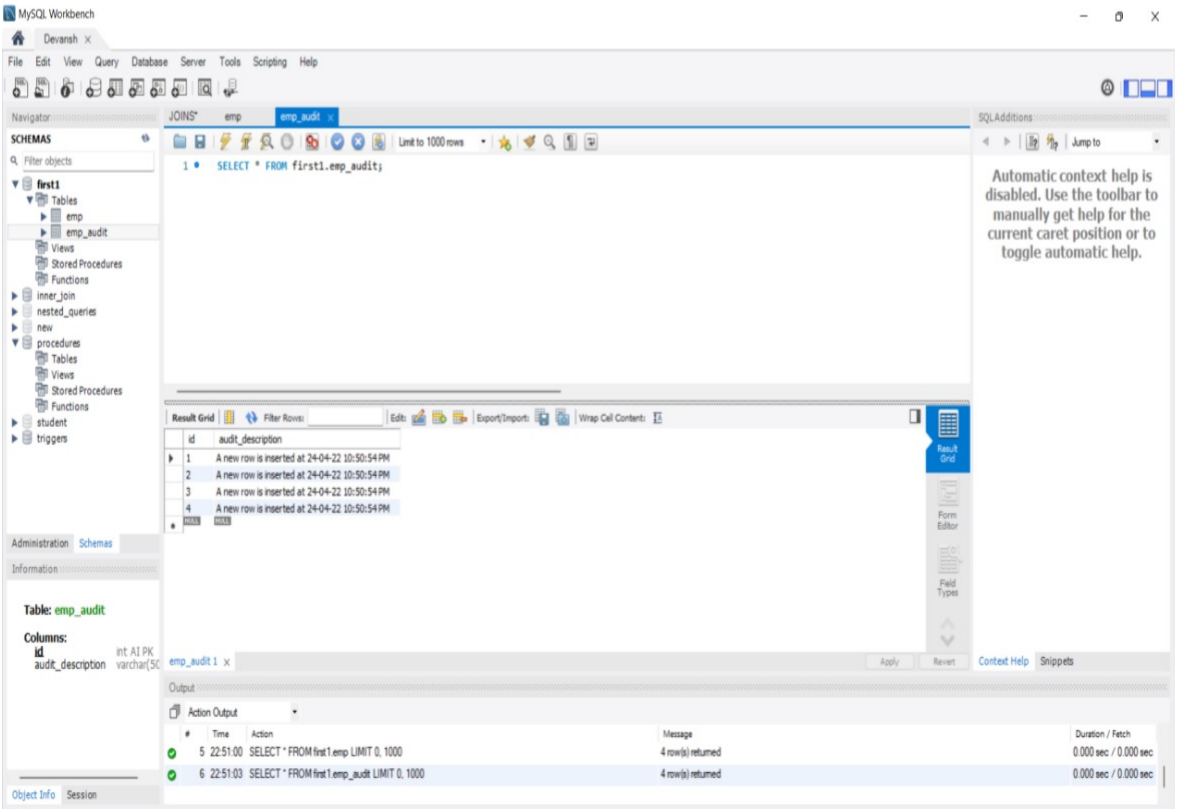
Theory:- A trigger in MySQL is a set of SQL statements that reside in a system catalog. It is a special type of stored procedure that is invoked automatically in response to an event. Each trigger is associated with a table, which is activated on any DML statement such as INSERT, UPDATE, or DELETE.

A trigger is called a special procedure because it cannot be called directly like a stored procedure. The main difference between the trigger and procedure is that a trigger is called automatically when a data modification event is made against a table. In contrast, a stored procedure must be called explicitly.

QUERY:-

```
CREATE TABLE emp(
id INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(50),
age INT
);
CREATE TABLE emp_audit(
id INT PRIMARY KEY AUTO_INCREMENT,
audit_description VARCHAR(500)
);
DELIMITER //
CREATE TRIGGER tr_AfterInsetEmp
AFTER INSERT
ON emp
FOR EACH ROW
BEGIN
INSERT INTO emp_audit

VALUES(null,concat('newrow',date_format(now(),'%d-%m-%y %h:%i:%s %p')));
END//
DELIMITER ;
INSERT INTO emp
VALUES
(null,'Akash',22),
(null,'Devansh',18),
(null,'Akshat',21),
(null,'Rahul',24);
```


MySQL Workbench
Devansh
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
first1
Tables
emp
emp_audit
Views
Stored Procedures
Functions
inner_join
nested_queries
new
procedures
Tables
Views
Stored Procedures
Functions
student
triggers
Administration Schemas
Information
Table: emp_audit
Columns:
id int AI PK
audit_description varchar(5
Object Info Session
JOINS* emp emp_audit
Limit to 1000 rows
1 SELECT * FROM first1.emp;
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
id name age
1 Akash 22
2 Devansh 18
3 Akshat 21
4 Rahul 24
emp 1
Apply Revert
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
5 22:51:00 SELECT * FROM first1.emp LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec
6 22:51:03 SELECT * FROM first1.emp_audit LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec


MySQL Workbench
Devansh
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
first1
Tables
emp
emp_audit
Views
Stored Procedures
Functions
inner_join
nested_queries
new
procedures
Tables
Views
Stored Procedures
Functions
student
triggers
Administration Schemas
Information
Table: emp_audit
Columns:
id int AI PK
audit_description varchar(5
Object Info Session
JOINS* emp emp_audit
Limit to 1000 rows
1 SELECT * FROM first1.emp_audit;
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
id audit_description
1 A new row is inserted at 24-04-22 10:50:54 PM
2 A new row is inserted at 24-04-22 10:50:54 PM
3 A new row is inserted at 24-04-22 10:50:54 PM
4 A new row is inserted at 24-04-22 10:50:54 PM
emp_audit 1
Apply Revert
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
5 22:51:00 SELECT * FROM first1.emp LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec
6 22:51:03 SELECT * FROM first1.emp_audit LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec

Exp No:-15 Date:-

Experiment:-Frame And Execute PL/SQL Cursor & Exceptional Handling

AIM –Frame and execute all queries for a project: Home renting system database

INTRODUCTION:-The Home Rental System is Searching in Based on the Apartment House for rent in metropolitan cities. The Home Rental System is Based on the Owners and the Customers. The Owner is updated on the Apartment details, and rent details. The Customer is details about the Room space, Room rent and the Address Details also.

PROBLEM STATEMENT

There is no properly allocate home and the system is not easily arranges according to their user interest. And also the home rental management system almost is done through the manual system.

The administrative system doesn't have the facility to make home rental management system through online and the most time the work done through illegal intermediate personwithout awareness of the administrative and this make more complex and more cost to find home for the customer. This leads to customer in to more trouble, cost, dishonest and time wastage.

The problem found in the current system:

- Complexity of finding home is not easy and more tedious.
- And also Extra money to find home.
- The system needs more human power.
- The user cannot get information about home when they need.
- There is too match time consumption find home

OBJECTIVE

The main objective of the system is to develop online home rental management system for wolkite city

Specific objectives

In order to attain the general objective, the following are the list of specific objectives:

- To facilitate home record keeping for who wants home and for administrative management system.
- Prepare an online home rental system for the home finders
- To reduce the travel costs and other unnecessary expenses of the buyer.
- Reduce the role of the broker, thus, providing protection from frauds related to papers.

MODULES

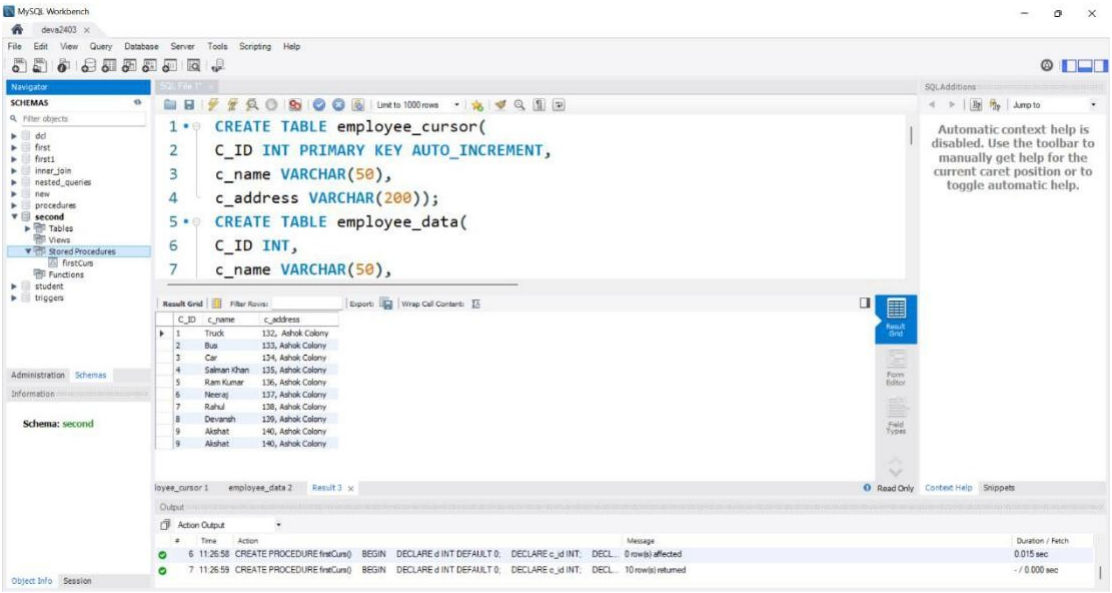
- Tenant
- Owner
- Contract
- Payment

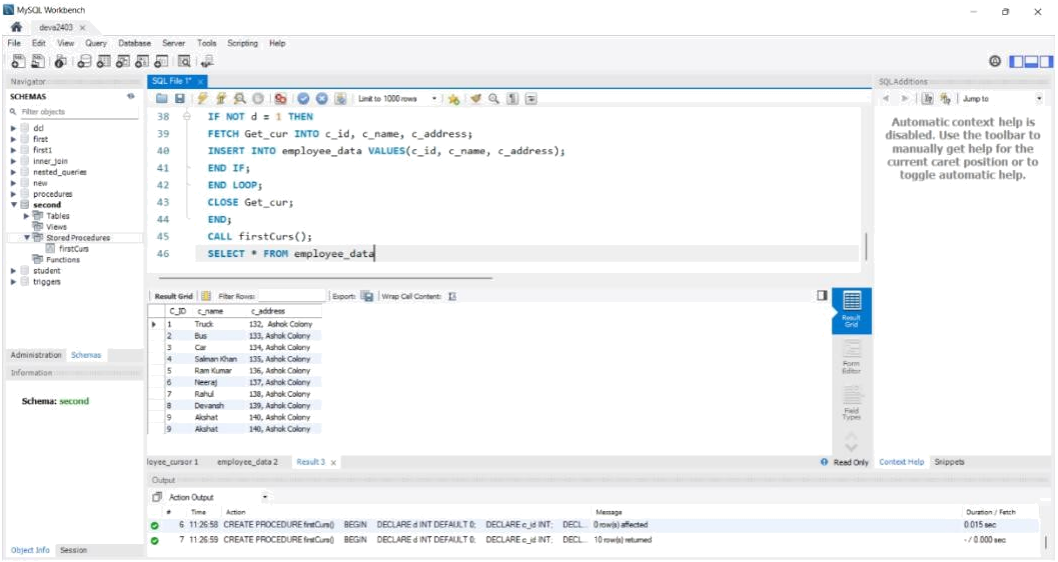
OPERATION

- First will be the login page where the User will login using their ID and Password.
- Next the user will enter the details like the requirements of the house he needs and contact details
- The other ebd user who wants to rent their house can also give their details
- He can also search the houses available in a particular area.
- Whoever has the required spcifications of the house can contact using the details provided.
- There is also a chatting option if they want to communicate with each other. There is also an option of video calling.
- After providing the details in the website, a contract will be generated which they can download.

OUTPUT:-

a)Cursor Output





b)Exception Handling