NCR CAMPUS, MODINAGAR

(A Constituent of SRM University, Chennai T.N.)

Delhi-Meerut Road, Sikari Kalan, Modinagar – 201204, GHAZIABAD (U.P.)

Database Management System Lab

(18CSC303J)

Lab Record

(Jan-May 2023)

Name of Student :
Reg. No. :

Degree/Branch : B. Tech / CSE

Semester/Section : 6th Sem, (CSE-J)

Course Code : 18CSC303J

Course Title : Database Management System

Faculty In charge. : Ms. Anjali Malik

Department of Computer Science and Engineering

SRM IST, DELHI-NCR CAMPUS, MODINAGAR

Department Of Computer Science and Engineering

REGISTRATION NO.

R	**A**	**2**	**0**	**1**	**1**	**0**	**0**	**3**	**0**	**3**	**0**		

BONAFIDE CERTIFICATE

It is to be certified that the bonfide practical record submitted by _____of 6th semester for Bachelor of Technology degree in the Department of Computer Science and Engineering, Delhi-NCR Campus, SRM IST has been done for the course **Database Management System Lab (18CSC303J)** during the academic semester session January 2023 – May 2023.

---------- ----------

Dr. R. P. Mahapatra

Head of the Department
Computer Science & Engg.

Ms. Anjali Malik

Assistant Professor
Computer Science & Engg.

Submitted for the University Examination held on _____

Examiner 1

Examiner 2

INDEX

Exp. No.	Name of Experiment	Page No.	Date	Signature
01.				
02.				
03.				
04.				
05.				
06.				
07.				
08.				
09.				
10.				
11.				

12.				
13.				
14.				
15.				

EXPERIMENT 1

AIM-SQL data definition: - write at least four queries for each

- Create
- Alter
- Modify
- Drop
- Rename

CREATE

1. create table student(Stu_name varchar(20),S_id int,d_o_b, varchar(10));
2. create table subject(Sub_name varchar(10),Sub_id int);
3. create table class_level(counsellor varchar(10), level_num int);
4. create table class_rep(CR_name varchar(10), section char);

ALTER

1. create table student(Stu_name varchar(20),S_id int,d_o_b varchar(10));
   alter table student drop column_d_o_b;
2. create table subject(Sub_name varchar(10), Sub_id int);
   alter table subject drop column Sub_id;
3. create table class_level(counsellor varchar(10), level_num int);
   alter table class level drop column level_num;
4. create table class_rep(CR_name varchar(10), section char);
   alter table class_rep drop column section;

DROP

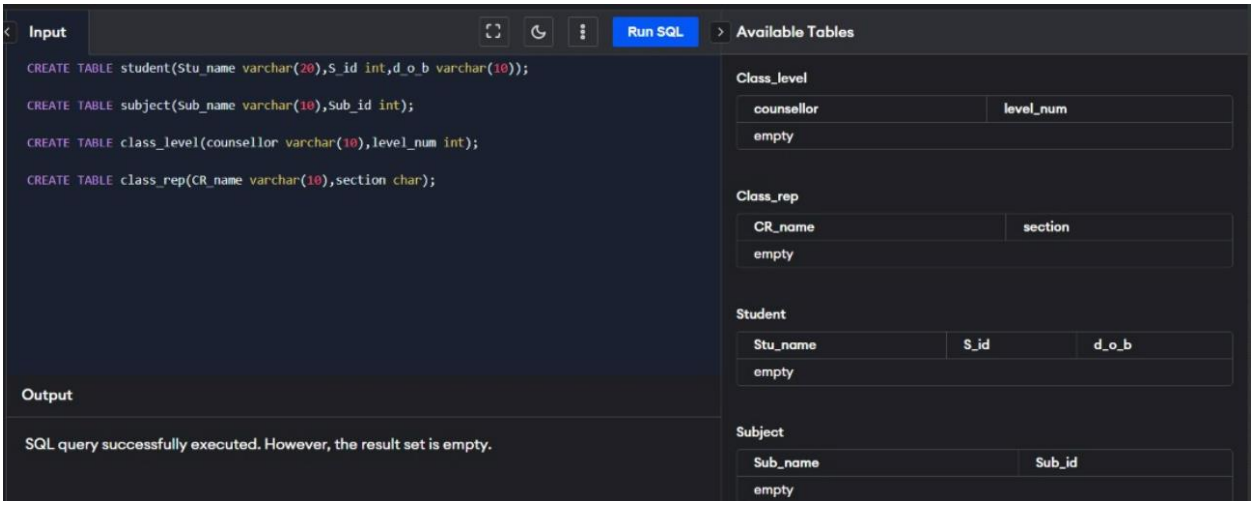
1. create table student(Stu_name varchar(20),S_id int,d_o_b varchar(10));
   drop table student;
2. create table subject(Sub_name varchar(10),Sub_id int);
   drop table subject;
3. create table class_level(counsellor varchar(10), level_num int);
   drop table class_level;
4. create table class_rep(CR_name varchar(10),section char);
   drop table class_rep;

RENAME

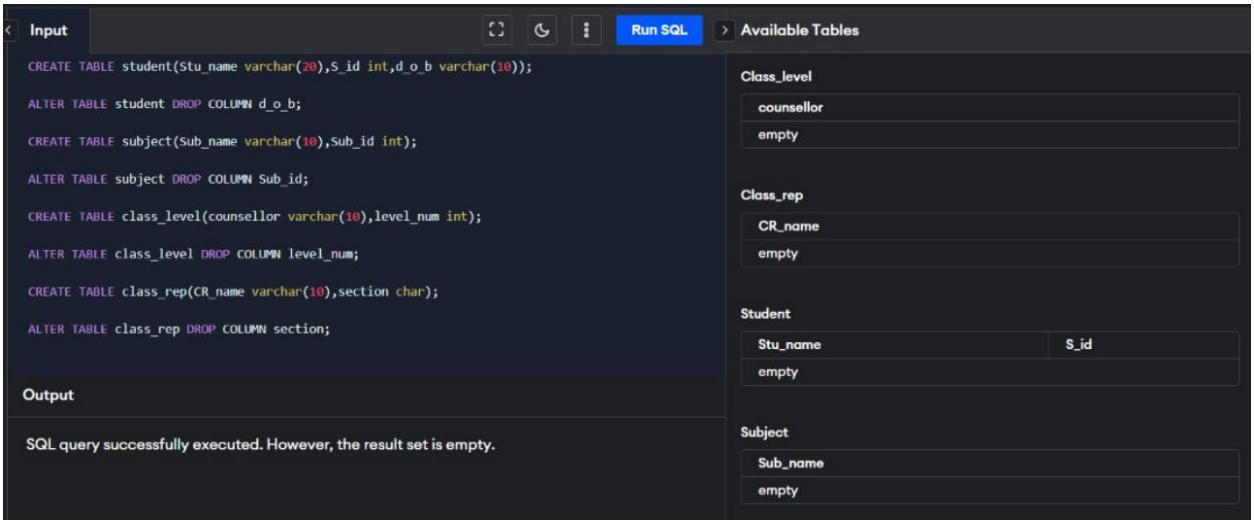
1. create table student(Stu_name varchar(20),S_id int,d_o_b varchar(10));
   rename table student to student_details;
2. create table subject(Sub_name varchar(10), Sub_id int);
   rename table student to student_details;
3. create table class_level(counsellor varchar(10), level_num int);
   rename class_level to class_details;
4. create table class_rep(CR_name varchar(10), section char);
   rename class_rep to CR details.

OUTPUT

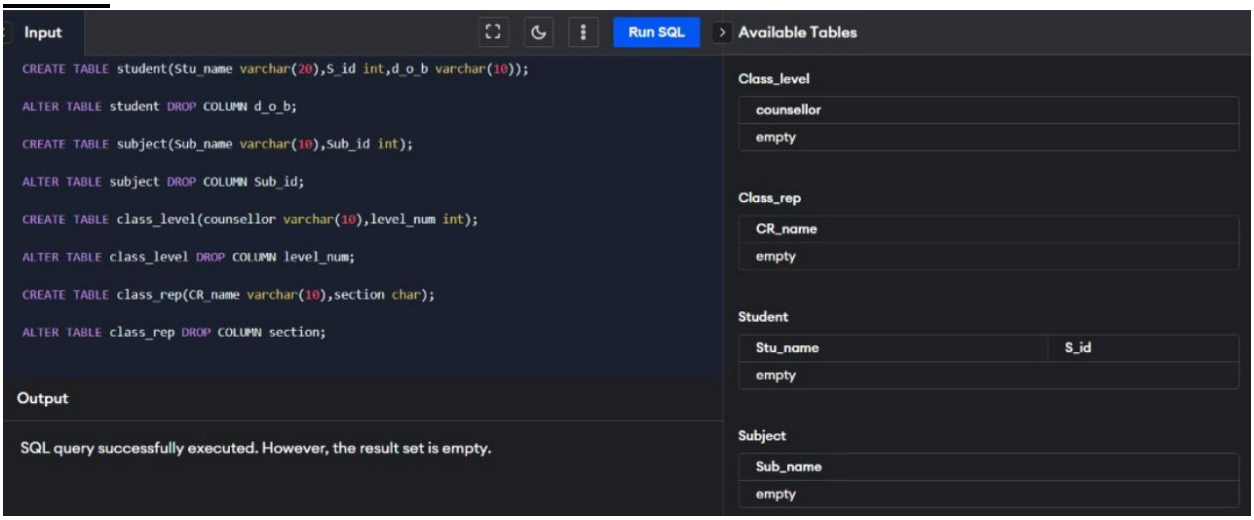
CREATE



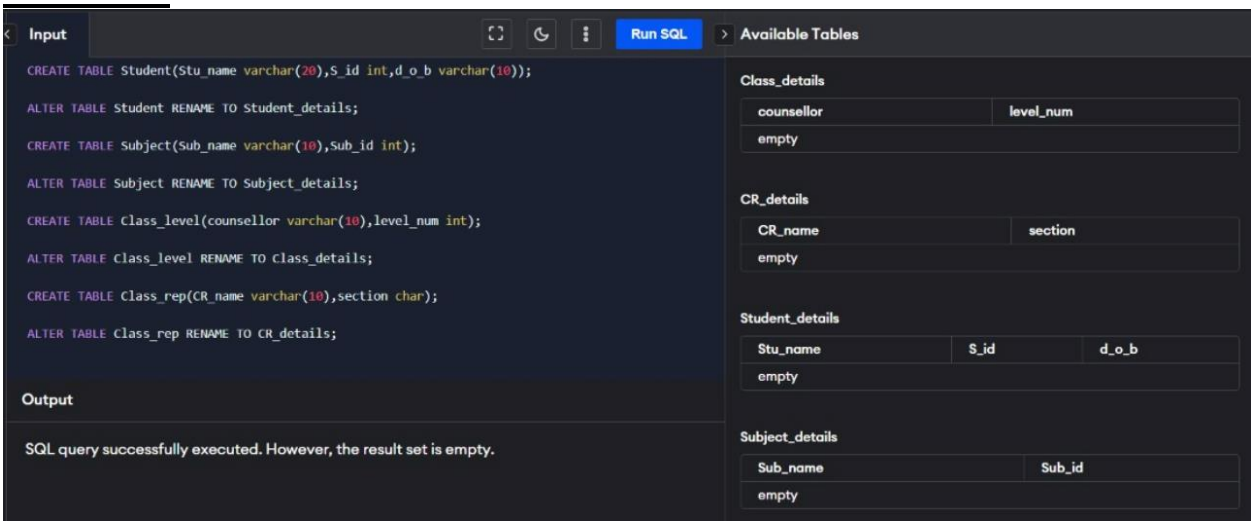
ALTER



DROP


Input
Run SQL
Available Tables
CREATE TABLE student(Stu_name varchar(20),S_id int,d_o_b varchar(10));
ALTER TABLE student DROP COLUMN d_o_b;
CREATE TABLE subject(Sub_name varchar(10),Sub_id int);
ALTER TABLE subject DROP COLUMN Sub_id;
CREATE TABLE class_level(counsellor varchar(10),level_num int);
ALTER TABLE class_level DROP COLUMN level_num;
CREATE TABLE class_rep(CR_name varchar(10),section char);
ALTER TABLE class_rep DROP COLUMN section;
Output
SQL query successfully executed. However, the result set is empty.
Class_level
counsellor
empty
Class_rep
CR_name
empty
Student
Stu_name
S_id
empty
Subject
Sub_name
empty

RENAME


Input
Run SQL
Available Tables
CREATE TABLE Student(Stu_name varchar(20),S_id int,d_o_b varchar(10));
ALTER TABLE Student RENAME TO Student_details;
CREATE TABLE Subject(Sub_name varchar(10),Sub_id int);
ALTER TABLE Subject RENAME TO Subject_details;
CREATE TABLE Class_level(counsellor varchar(10),level_num int);
ALTER TABLE Class_level RENAME TO Class_details;
CREATE TABLE Class_rep(CR_name varchar(10),section char);
ALTER TABLE Class_rep RENAME TO CR_details;
Output
SQL query successfully executed. However, the result set is empty.
Class_details
counsellor
level_num
empty
CR_details
CR_name
section
empty
Student_details
Stu_name
S_id
d_o_b
empty
Subject_details
Sub_name
Sub_id
empty

EXPERIMENT 2

AIM- SQL data manipulation: At least four queries for each

1. Insert Command
2. Update Command
3. Delete Command
4. Merge Command
5. Select

INSERT

1. create table student(Stu_name varchar(20),S_id int,d_o_b varchar(10));
   insert into student values ('Vishal', 1,'22-12-2000'), ('Alpha',2,'12-11-2000'), ('Priyanshu',3,'30-05-2000'), ('Priyanshi',4,'28-02-2000');
2. create table subject(Sub_name varchar(10), Sub_id int);
   insert into subject
   values('maths',234),(computer',303), ('database',236), ('dsa',235);
3. create table class_level(counsellor varchar(10), level_num int);
   insert into class_level values('Ms Anjali', 10), ('Mr Navin', 11), ('Mr Manas', 12);
4. create table class_rep(CR_name varchar(10), section char);

UPDATE

1. update student set Stu_name='Alice' where S_id=2;
2. update subject set Sub_id=304 where Sub_name='computer';
3. update class_level set level_num-5 where counsellor-'Mr Sahu ';
4. update class_rep set CR_name='Harsh' where section='A';

DELETE

1. delete from student where S_id=4;
2. delete from subject where Sub_id-236;
3. delete from class_level where counsellor='Mrs Babita';
4. delete from class_rep where CR_name='Anu';

MERGE

create table products (product_id int product_name varchar(100), rate int);
insert into products values (1, 'tea', 10),(2, 'coffee', 20),(3, 'muffin', 30),(4, 'biscuit', 40);
create table updated_products (product_id int, product_name

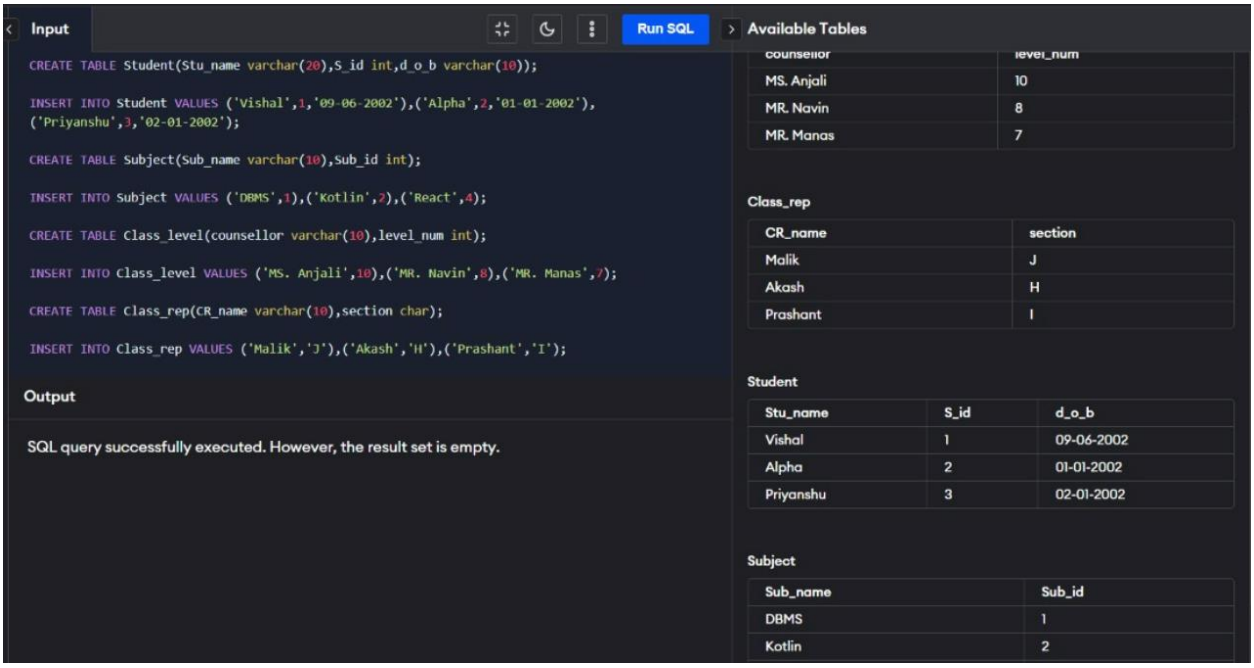
varchar(100), rate int);
insert into updated_products values (1, 'tea', 10),(2, 'coffee', 20),(3,
'muffin', 30),(5, 'pizza', 60);
select * from products;
select * from updated_products;
merge into products t using
updated_products s on (t.product_id =
s.product_id)
"
when matched then
update set t.product_name = s.product_name,
t.rate = s.rate when not matched then
insert(t.product_name,t.rate) values
(s.product_name, s.rate); select * from products;

SELECT

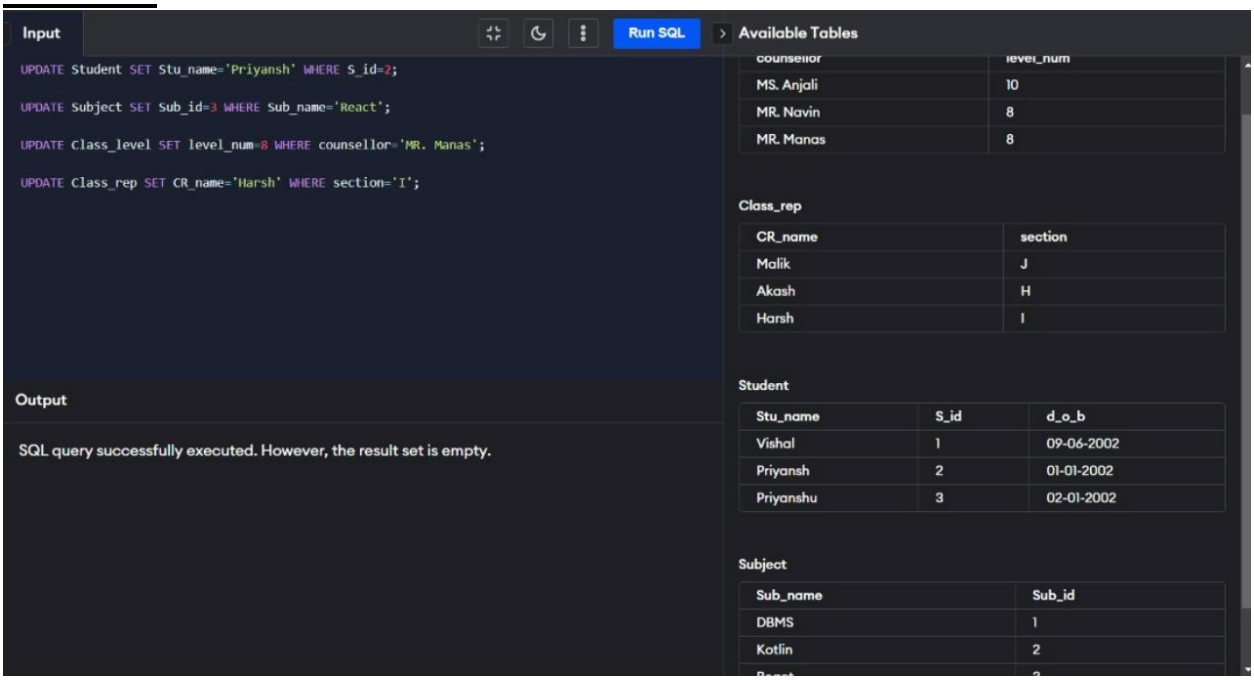
1. select from students;
2. select from subject;
3. select * from class_rep;
4. select from class_level;

OUTPUT

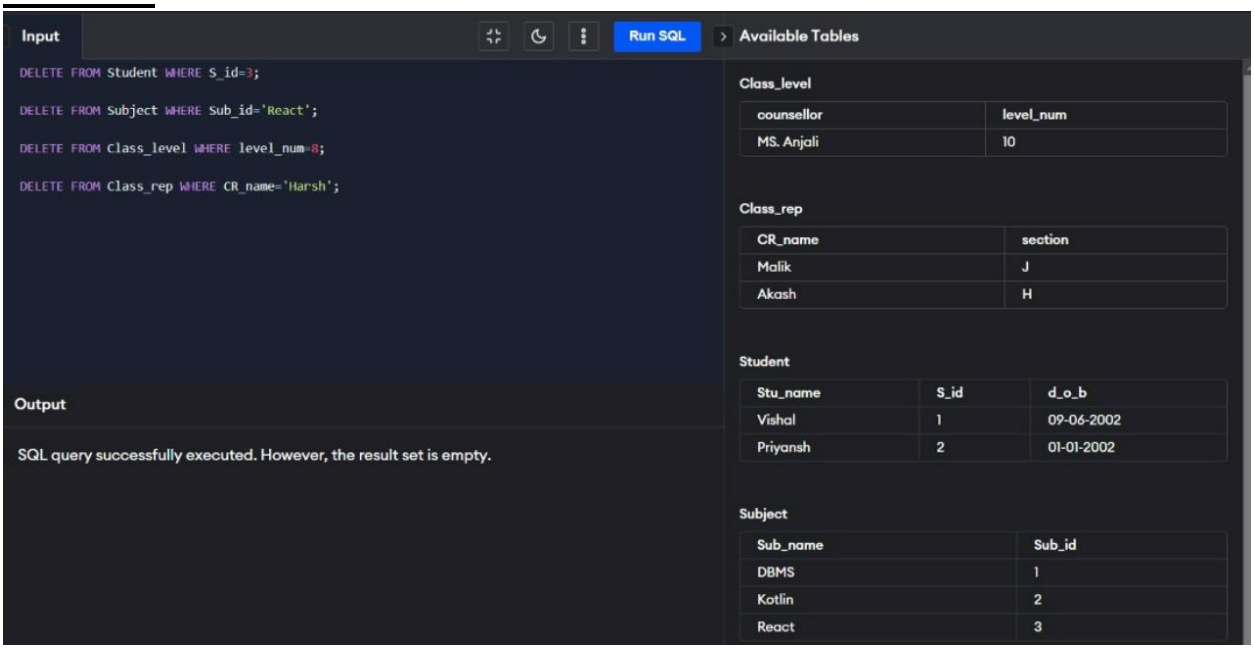
INSERT



UPDATE



DELETE



MERGE

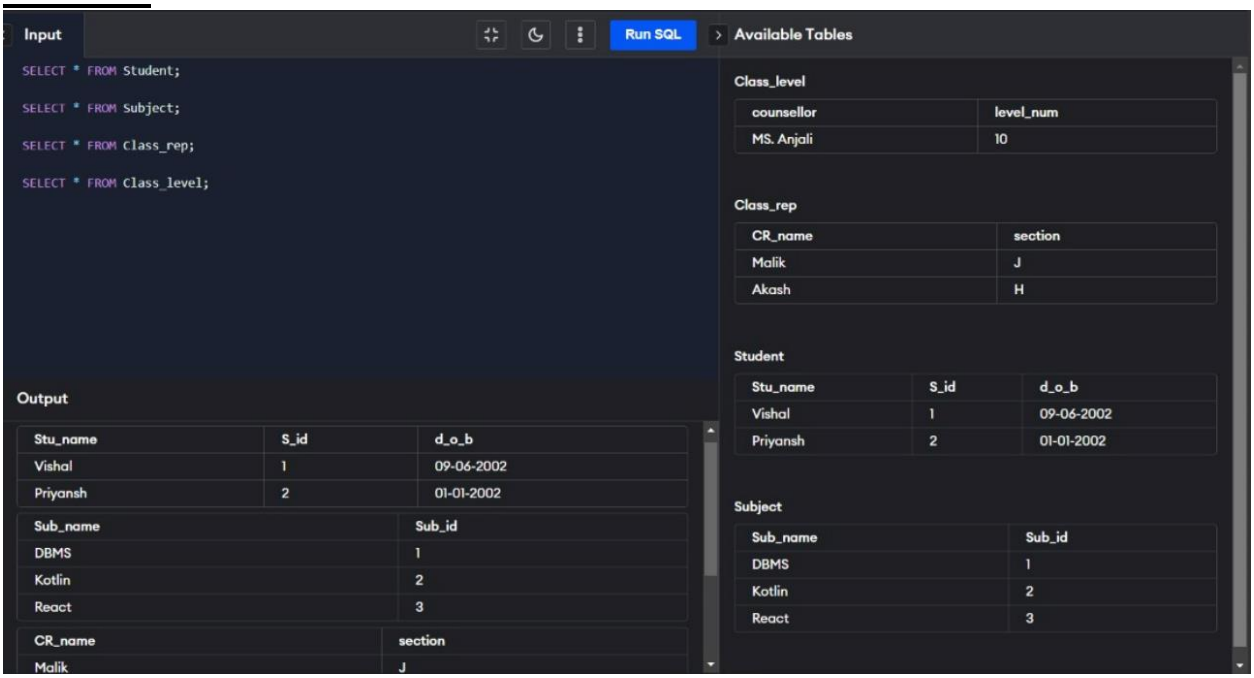
db<>fiddle SQL Server 2022 no sample DB run markdown

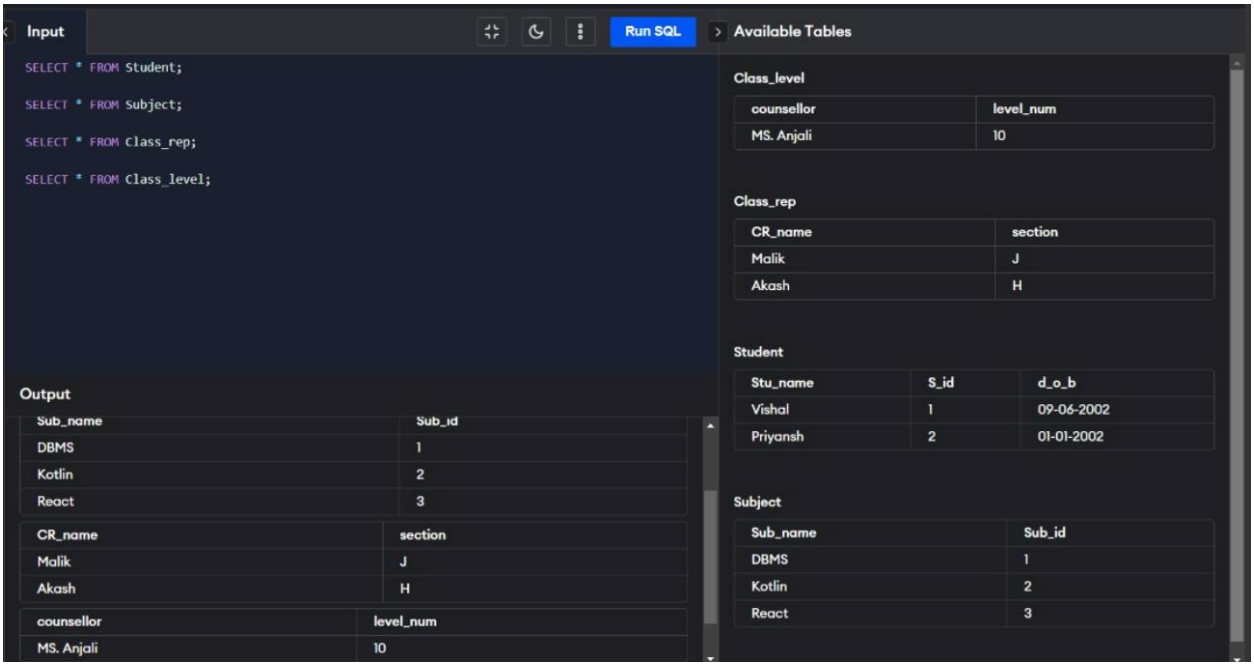
By using db<>fiddle, you agree to license everything you submit by Creative Commons CC0.

```
1  create table product(product_id int, product_name varchar(20), rate int);
2  insert into product values(1, 'Tea', 22), (2, 'Coffee', 30), (3, 'Muffin', 67);
3
4  create table updated_products (product_id int, product_name varchar(100), rate int);
5  insert into updated_products values (1, 'tea', 10), (2, 'coffee', 20), (3, 'muffin', 30);
6
7  merge into product t
8  using updated_products s on t.product_id = s.product_id
9  when matched then
10 update set
11   t.product_name = s.product_name,
12   t.rate = s.rate
13 when not matched then
14 insert (product_id, product_name, rate) values (s.product_id, s.product_name, s.rate);
15 select *from product;
```

product_id	product_name	rate
1	tea	10
2	coffee	20
3	muffin	30

SELECT




Input
Run SQL
Available Tables
SELECT * FROM Student;
SELECT * FROM Subject;
SELECT * FROM Class_rep;
SELECT * FROM Class_level;
Output
Sub_name
Sub_id
DBMS
1
Kotlin
2
React
3
CR_name
section
Malik
J
Akash
H
counsellor
level_num
MS. Anjali
10
Class_level
counsellor
level_num
MS. Anjali
10
Class_rep
CR_name
section
Malik
J
Akash
H
Student
Stu_name
S_id
d_o_b
Vishal
1
09-06-2002
Priyansh
2
01-01-2002
Subject
Sub_name
Sub_id
DBMS
1
Kotlin
2
React
3

EXPERIMENT 3

AIM – SQL data control language and transaction control Language. At least three queries for each-

a) Grant
b) Revoke
c) Commit
d) Rollback
e) Savepoint

GRANT

- create user admin1 identified by passwordl;
- grant create session to admin1;
- grant connect to admin1;

REVOKE

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

COMMIT

- grant connect to admin1;
- revoke connect from admin1;
- commit;

ROLLBACK

- create table subject(Sub_name varchar(10),Sub_id int);
  insert into subject values('maths',234), ('computer',303), ('database',236);
  select * from subject;
  delete from subject where sub_id>235;
  rollback;

- create table class_rep(CR_name varchar(10),section char);

insert into class_rep values (‘Sarthak’,‘A’), ('Sachin','B'), ('Priya','C');
delete from class_rep where section='C';
rollback;

- create table class_level(counsellor
  varchar(10),level
  num int);
  insert into class_level values('Ms
  Ruchika',10), ('Mrs Babita', 11). ('Ms
  Rani', 12);
  delete from class level where
  level_num>10;
  rollback;

SAVEPOINT

- create table a1(a_id int, a serialno int);
  insert into a1 values
  (1,123), (2,452),
  (3,526);
  savepoint al_table;

- update a1 set
  a_serialno=242 where
  a_id=2;
  savepoint a1_update;
- drop table al;
  savepoint a1_drop;

OUTPUT

GRANT
REVOKE

```
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.

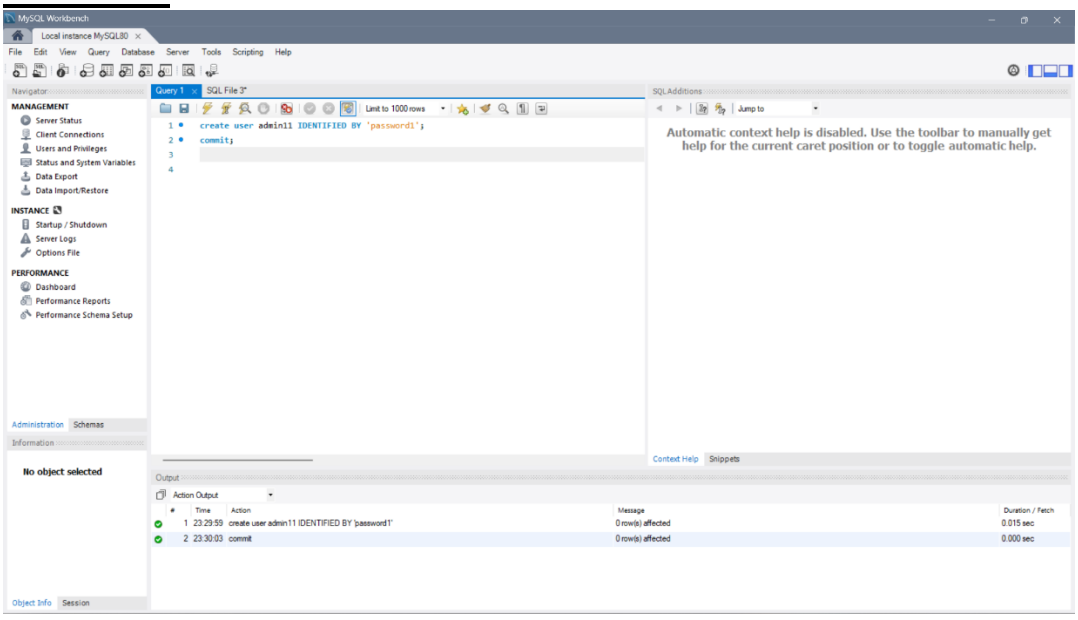
```
create user admin1 identified by password1;
revoke create session from admin1;
revoke connect  from admin1;
revoke unlimited tablespace from admin1;
revoke all privileges from admin1;
```

Script Output

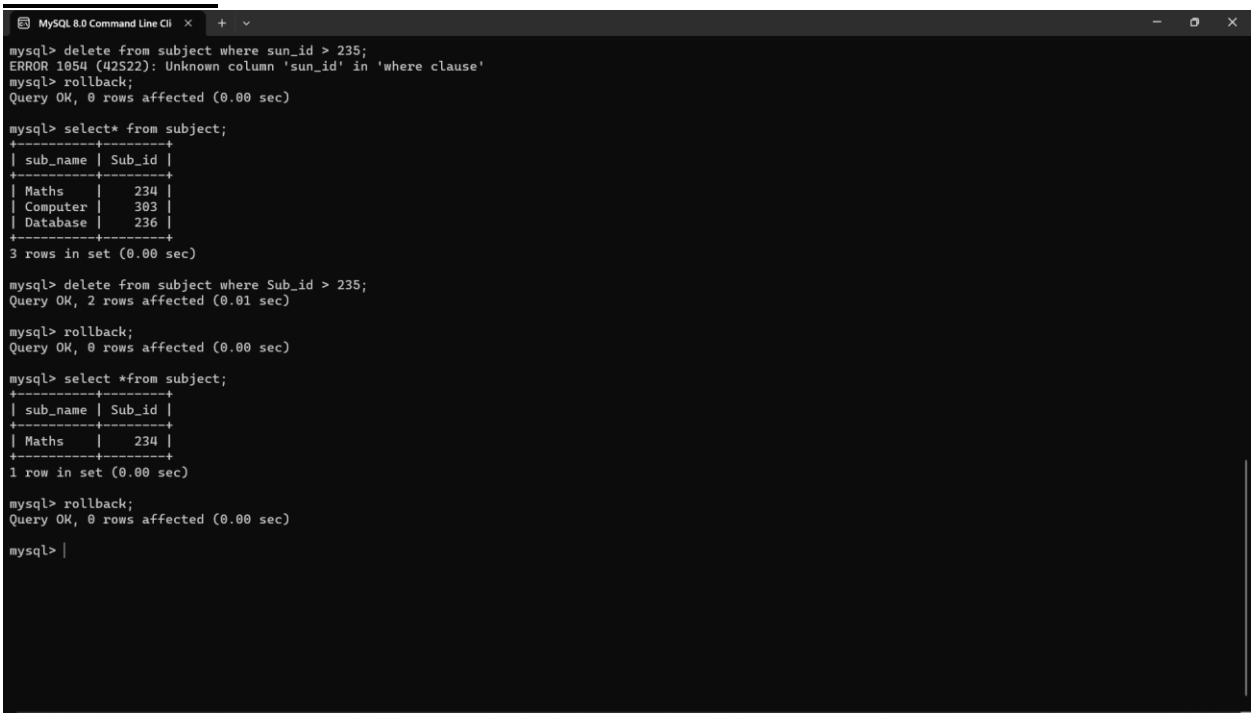
Task completed in 0.056 seconds

Revoke succeeded.

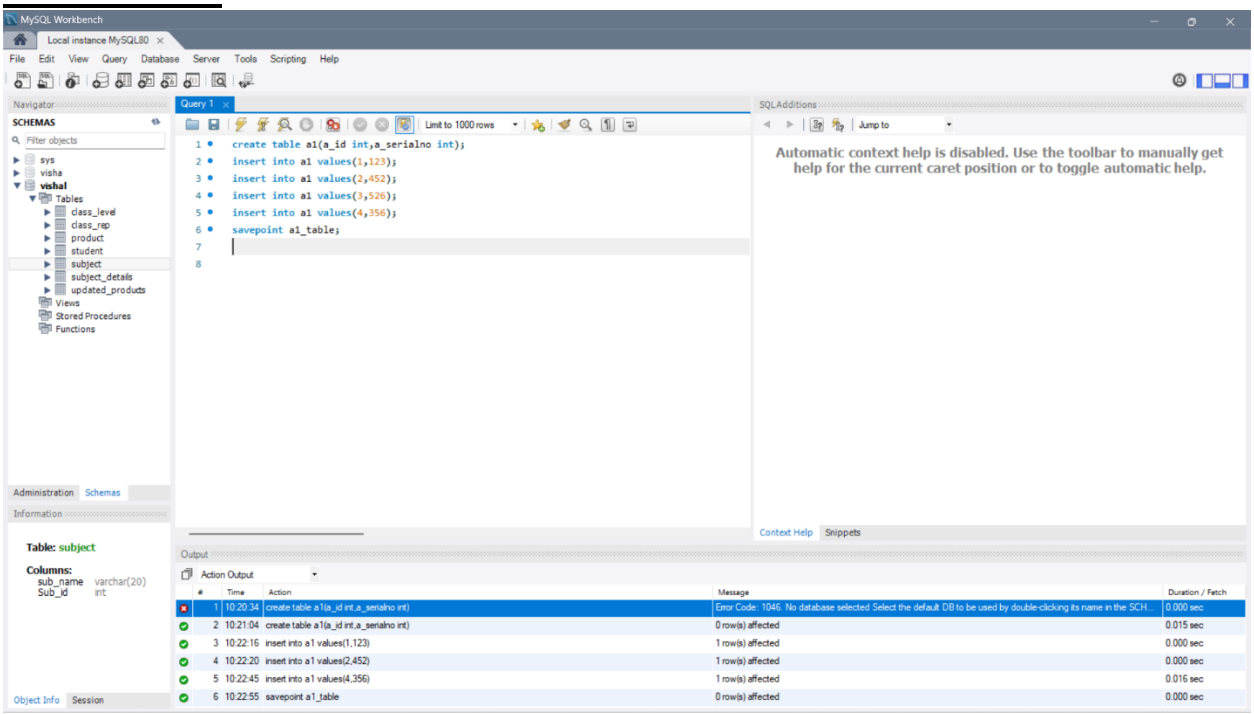
COMMIT

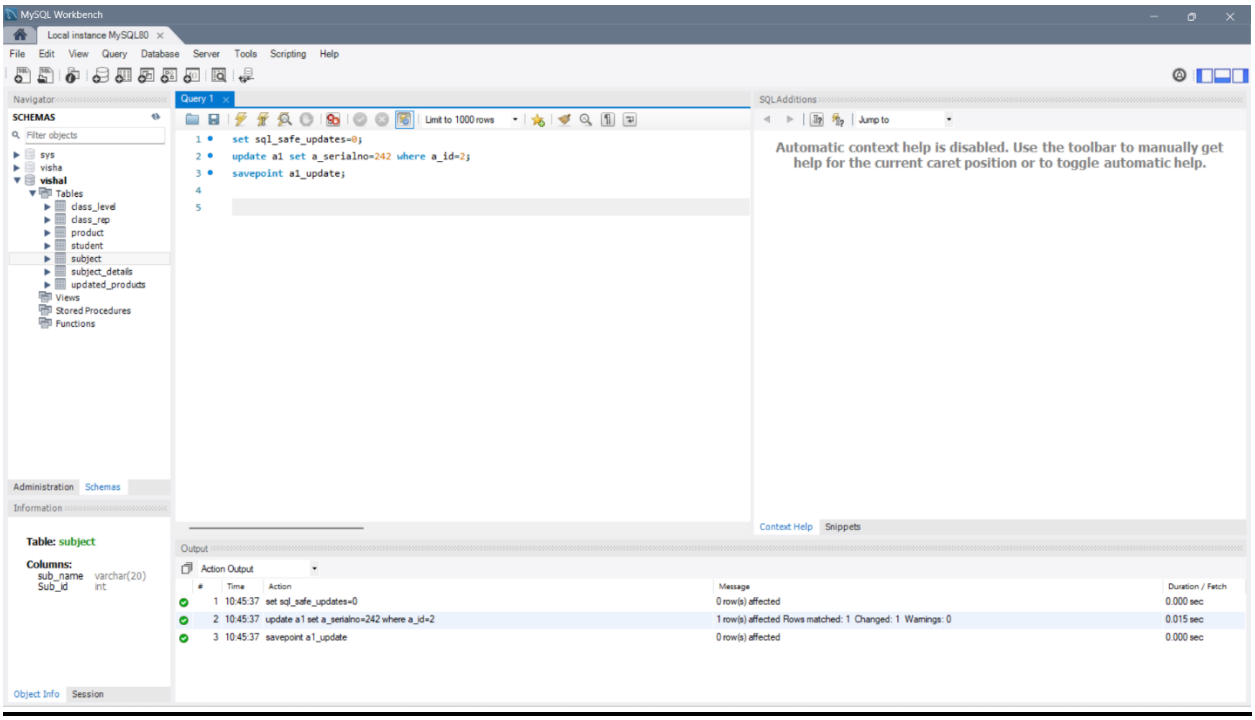


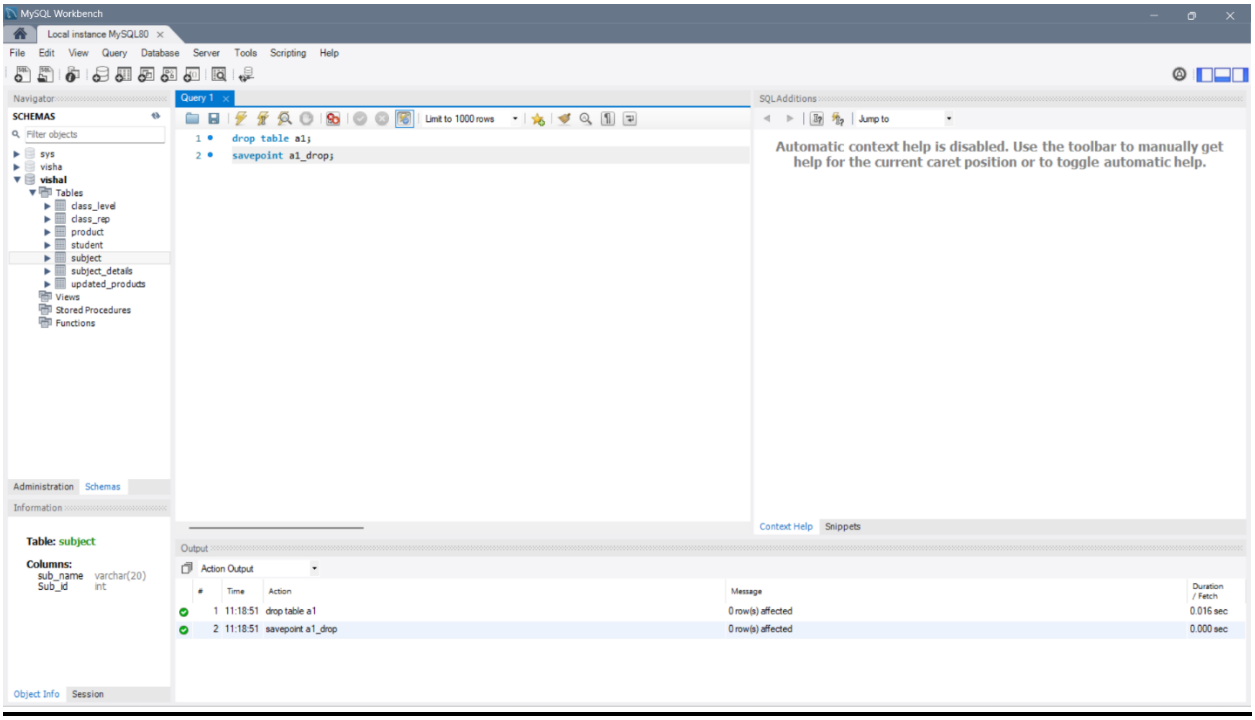
ROLLBACK



SAVEPOINT




MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
sys
visha
vishal
Tables
class_level
class_rep
product
student
subject
subject_details
updated_products
Views
Stored Procedures
Functions
Administration Schemas
Information
Table: subject
Columns:
sub_name varchar(20)
Sub_id int
Object Info Session
Query 1
Limit to 1000 rows
1 set sql_safe_updates=0;
2 update a1 set a_serialno=242 where a_id=2;
3 savepoint a1_update;
4
5
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
Output
Action Output
Time Action Message Duration / Fetch
1 10:45:37 set sql_safe_updates=0 0 row(s) affected 0.000 sec
2 10:45:37 update a1 set a_serialno=242 where a_id=2 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 0.015 sec
3 10:45:37 savepoint a1_update 0 row(s) affected 0.000 sec


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
sys
visha
vishal
Tables
class_level
class_rep
product
student
subject
subject_details
updated_products
Views
Stored Procedures
Functions
Administration Schemas
Information
Table: subject
Columns:
sub_name varchar(20)
Sub_id int
Object Info Session
Query 1
Limit to 1000 rows
1 drop table a1;
2 savepoint a1_drop;
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
Output
Action Output
Time Action Message Duration / Fetch
1 11:18:51 drop table a1 0 row(s) affected 0.016 sec
2 11:18:51 savepoint a1_drop 0 row(s) affected 0.000 sec

EXPERIMENT-4

AIM - Inbuilt functions in SQL:

a) NVL functions
b) Case function
c) String functions
d) Date functions
e) Aggregation functions
f Numeric function
g) Case manipulation function
h) Character manipulation function

Case Function

- select stu_name,
- case
- stu_id = 1 then 'good'
- when 4 then 'bad'
- else 'ok' end as student_type from student;
- select Marks,
- case Marks>90 then 'Excellent' when marks > 80 then 'Good' else 'Average' end as subject_type from subject;

String Function

- select ASCII(stu_name) as ASCII_Name from dual;
- select character_length('Vishal dubey');
- SELECT CHARACTER_LENGTH(stu_name) as Alias_Name from student_details;

Date Function

- select current_date from dual;
- select current_timestamp from dual;
- select round(date '2017-07-16','MM') as rounded_off from dual;

Aggregation Fuction

- create table stud(id int,regno int,name varchar(10),year int,marks int);
  insert into stud values(1,23, 'Vishal',3,18),(2,24,
  'Naman',3,20),(3,25,'Shalini',5,19),(7,26, 'Aneesh',3,20);
  select * from stud;
- select year,avg(marks) from stud group by year;
- select year,max(marks) from stud group by year having
  avg(marks)>19;
- select count(marks) from stud;

Numeric Functions

- select floor(4.1234) from dual;
- select log(10,1000) as log_1000_base10 from dual;
- select sin(3.1415926535897938) from dual;

Case Manipulation Function

- select * from student where lower(stu_name)= 'Priyanshu';
- select * from subject where upper(sub_name)='MATHS';
- select lower('Saturday') as łower_val from dual;

Character Manipulation Function

- select instr('this is anupriya', 'anu') location from dual;
- select doncat(concat('Anu','priya'),' Johri')as sysdba_name from
  dual;
- select trim(leading'0' from '00120341523413000')as trimmed
  from dual;

NVL Function

select nvl (100, 200) from dual;
select nvl (null,200) from dual;
create table hello (id int, code int);
insert into hello(code) values(12);
insert into hello values(1,12);
insert into hello(code)values(15);

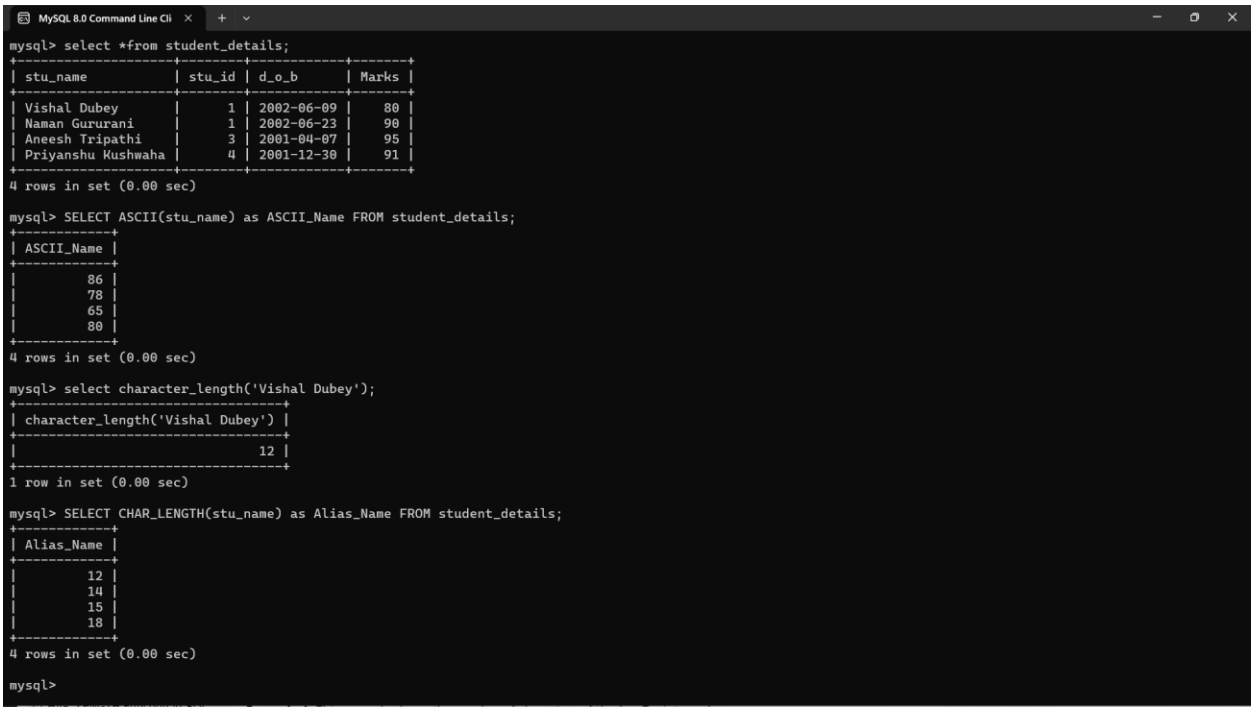
insert into hello(code) values(25);
select * from hello;
select nvl(id,code)from hello where code>20;
select nvl(id,code) from hello;

OUTPUT

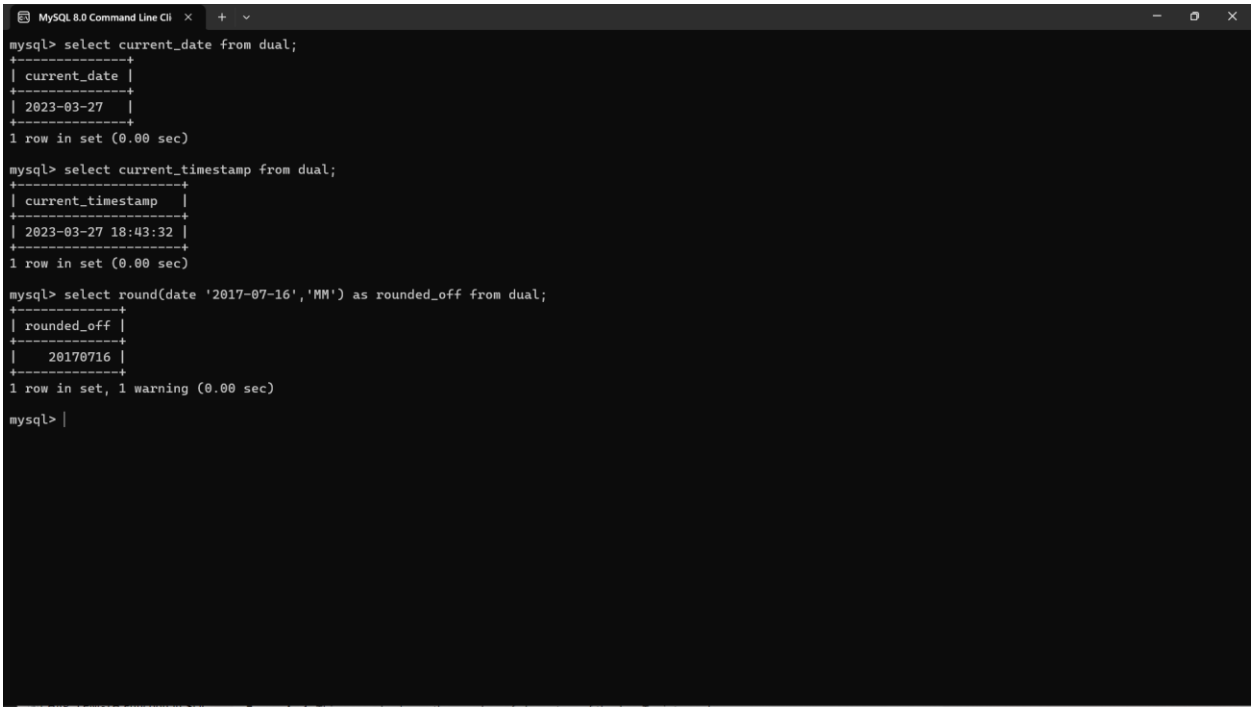
CASE FUNCTION

```
mysql> select stu_name,
    -> case
    -> when stu_id = 1 then 'good'
    -> when stu_id = 4 then 'bad'
    -> else 'ok'
    -> end as student_type
    -> from student_details;
+--------------------+--------------+
| stu_name           | student_type |
+--------------------+--------------+
| Vishal Dubey       | good         |
| Naman Gururani     | good         |
| Aneesh Tripathi    | ok           |
| Priyanshu Kushwaha | bad          |
+--------------------+--------------+
4 rows in set (0.00 sec)

mysql> select marks,
    -> case
    -> when marks > 90  then 'Excellent'
    -> when marks > 80 then 'Good'
    -> else 'Average'
    -> end as student_type
    -> from student_details;
+-------+--------------+
| marks | student_type |
+-------+--------------+
|    80 | Average      |
|    90 | Good         |
|    95 | Excellent    |
|    91 | Excellent    |
+-------+--------------+
4 rows in set (0.00 sec)

mysql>
```

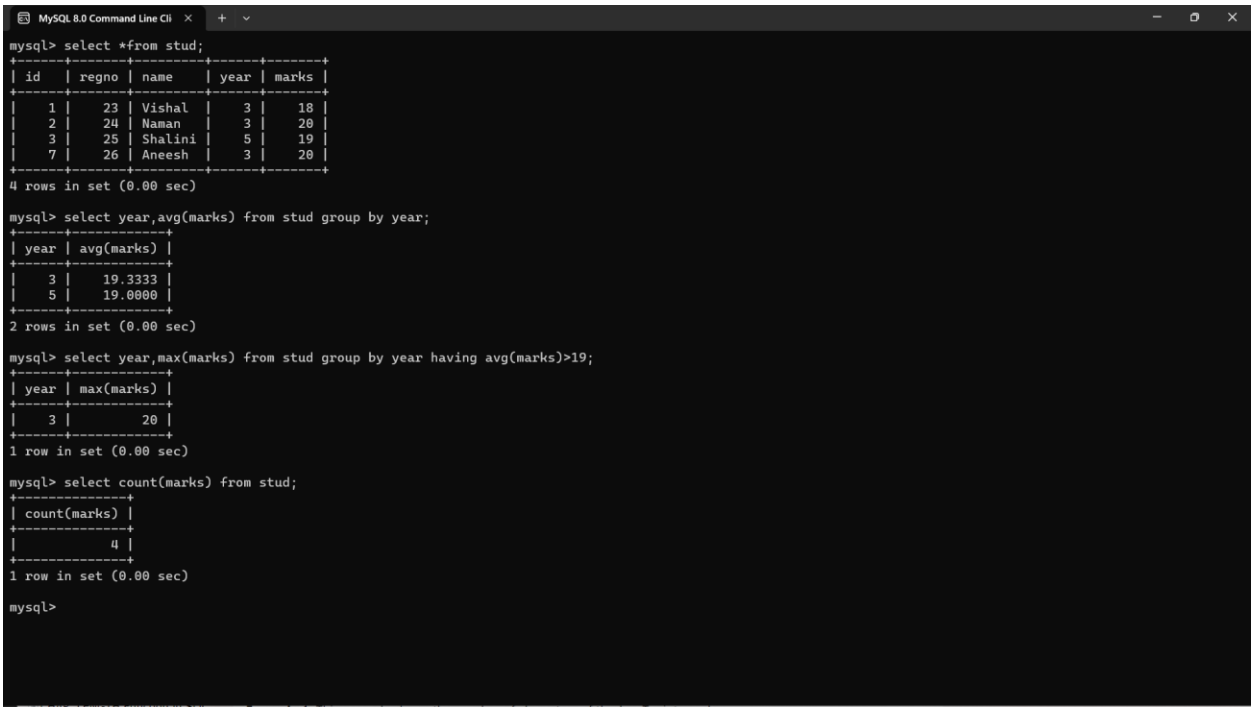
STRING FUNCTION



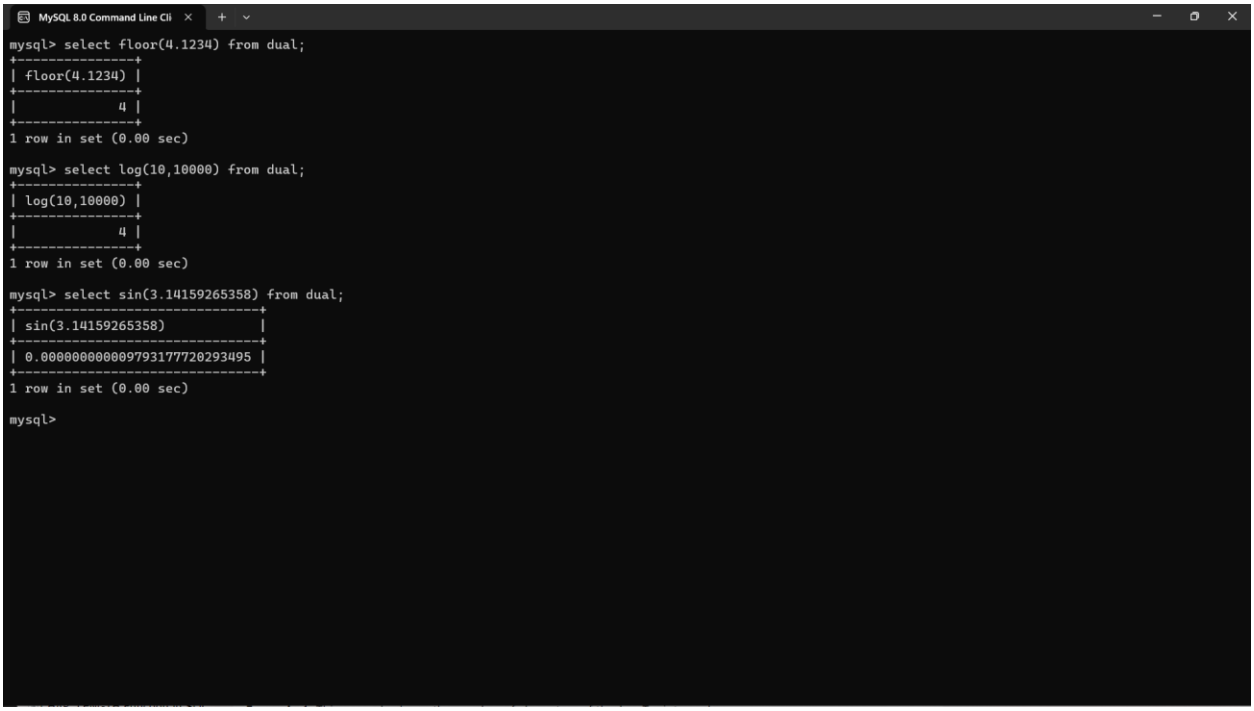
DATE FUNCTION



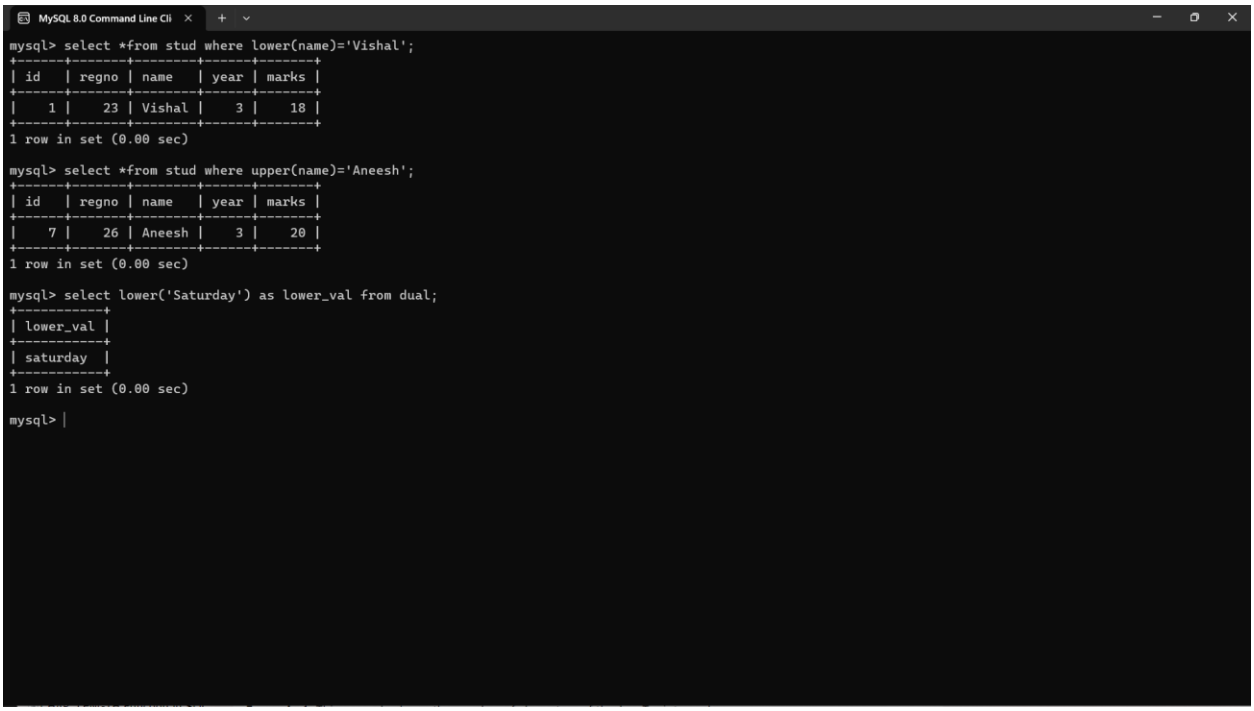
AGGREGATION FUNCTION



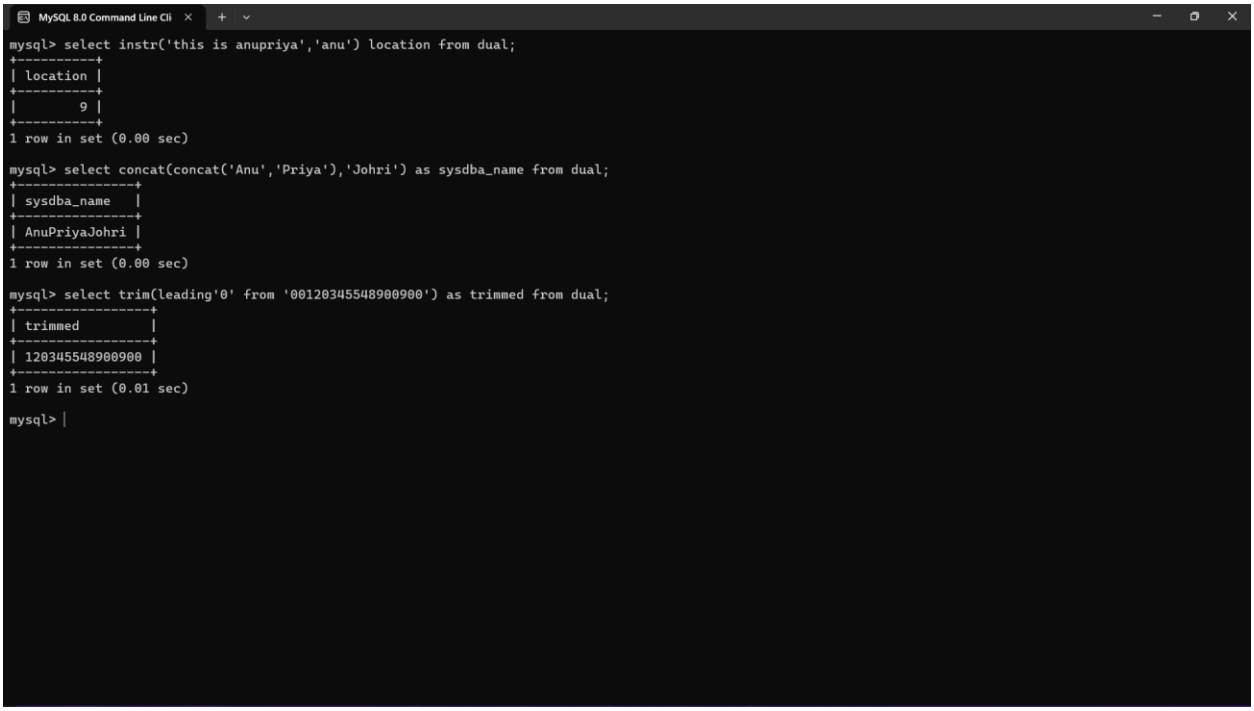
NUMERIC FUNCTION



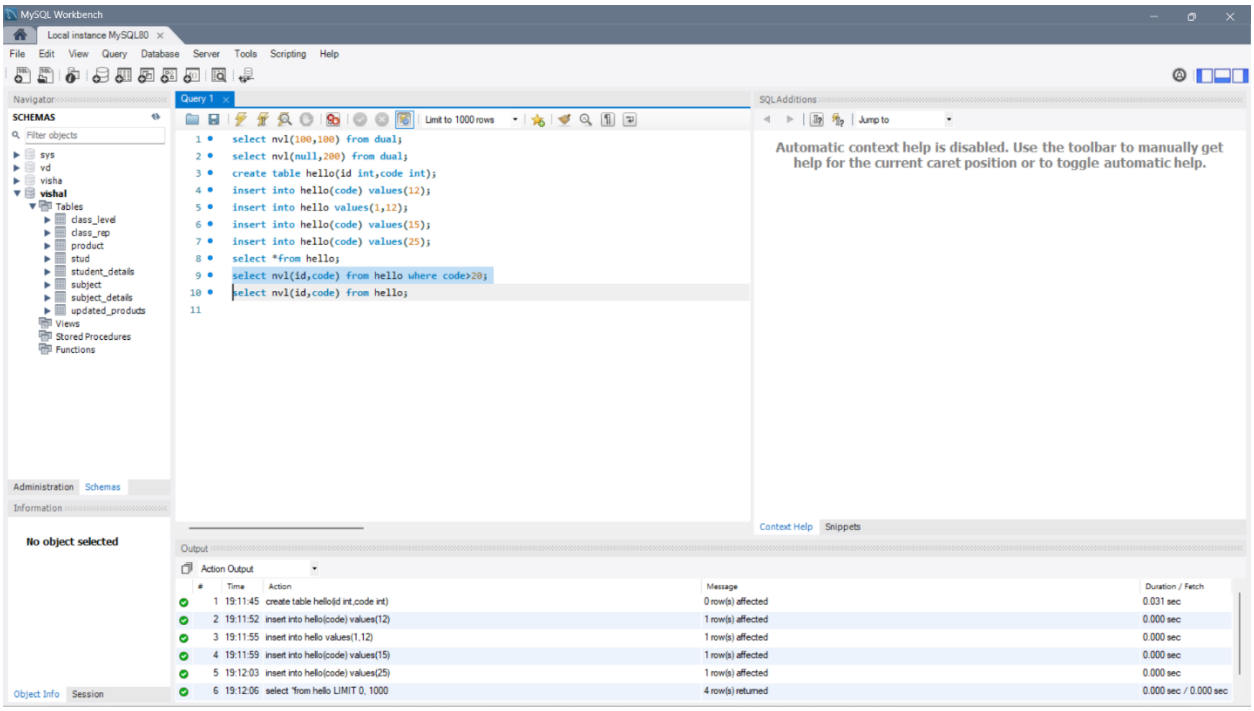
CASE MANIPULATION FUNCTION



CHARACTER MAINIPULATION FUNCTION



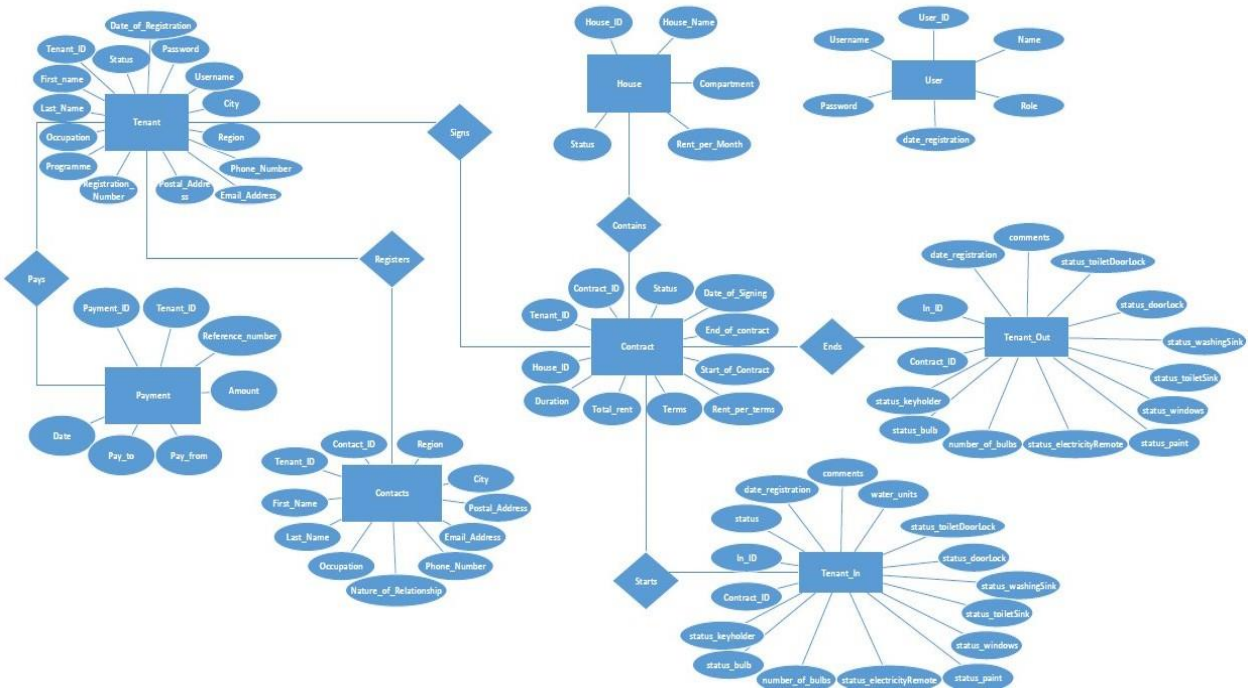
NVL FUNCTION



EXPERIMENT-05

AIM- Construct an 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



EXPERIMENT-06

AIM- Nested queries:
a) Copying rows from another table
b) Updating two columns with a sub query
c) Deleting rows based on another table
d) Using a subquery in an insert statement
e) Creating table using subquery

COPYING ROWS FROM ANOTHER TABLE

CREATING TABLES

```
create table empl(name varchar(10),id int,salary int, holiday int, dept varchar(10));
create table admin_emp(name varchar(10),id int, email varchar(20));
insert into empl
values('Aneesh', 1,23412,12,MGT),('Vishal',2,43322,10,'Shalini'), ('Riya',4,65000,
15, 'RND'),
insert into admin_emp
values('Naman', 5,'naman21@gmail.com'), ('Manas',6,'Manascool@gmail.com'),
('Garvit',7,'garvithot2@gmail.com'), ('suraj',8,'surajboom@gmail.com');
select * from emp1;
select from admin_emp;
```

COPYING ROWS FROM ANOTHER TABLE

```
insert into empl (name,id)select name,id from admin_emp where id>6;
select * from empl;
```

UPDATING TWO COLUMNS WITH A SUBQUERY

```
update empl set salary-70000, holiday-11 where id in(select id
from admin_emp where name='Garvit');
update empl set salary-60000, holiday-11 where id in(select id
from admin_emp where name='Suraj');
select * from empl;
```

DELETING ROWS BASED ON ANOTHER TABLE

delete from emp1 where id in(select id from admin_emp where id>=6);
select * from empl;

USING SUBQUERY TO INSERT STATEMENT

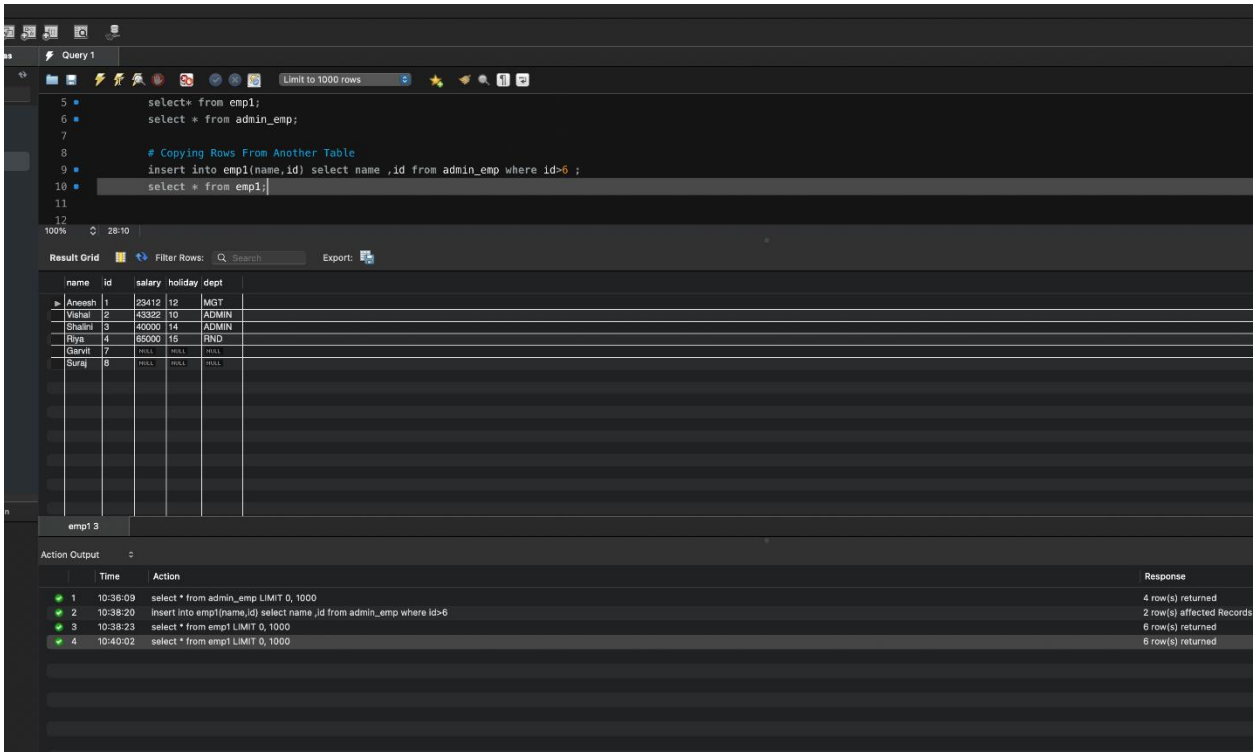
insert into emp1(id)(select id from admin_emp where id>=6);
select * from empl;

CREATING TABLE USING SUBQUERY

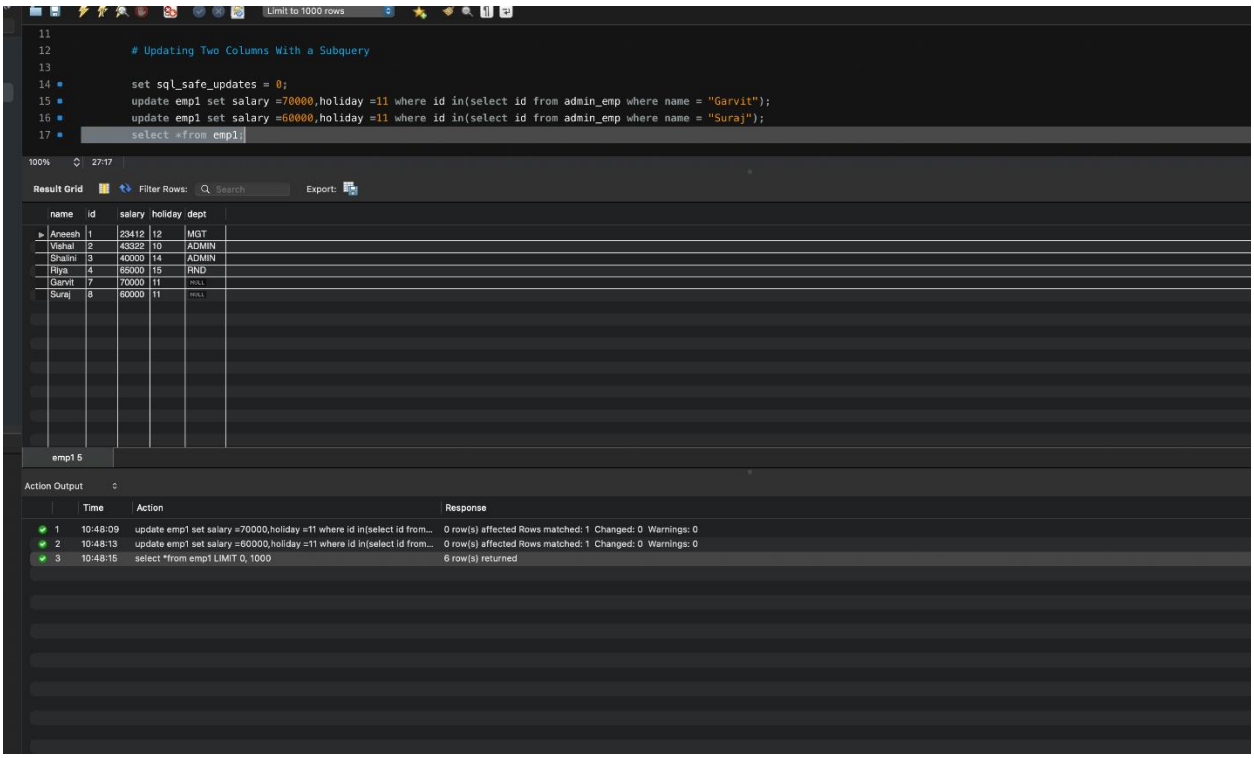
create table emp_salary as select id,salary from emp1 where id >=1
and id<=4; select * from emp_salary;

OUTPUT

COPYING ROWS FROM ANOTHER TABLE



UPDATING TWO COLUMNS WITH A SUBQUERY



```
# Updating Two Columns With a Subquery

set sql_safe_updates = 0;
update emp1 set salary =70000,holiday =11 where id in(select id from admin_emp where name = "Garvit");
update emp1 set salary =60000,holiday =11 where id in(select id from admin_emp where name = "Suraj");
select *from emp1;
```

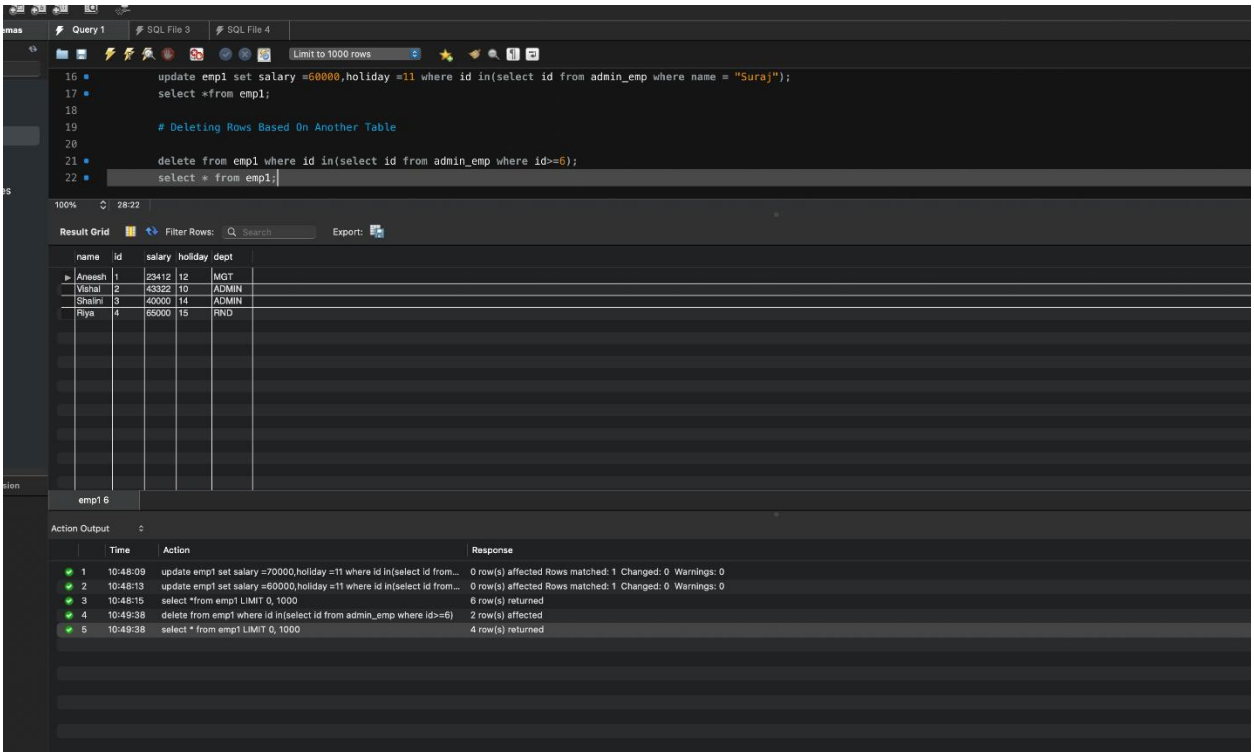
name	id	salary	holiday	dept
Aneesh	1	23412	12	MGT
Vishal	2	43322	10	ADMIN
Shalini	3	40000	14	ADMIN
Riya	4	65000	15	RND
Garvit	7	70000	11	NULL
Suraj	8	60000	11	NULL

emp1 5

Action Output

	Time	Action	Response
1	10:48:09	update emp1 set salary =70000,holiday =11 where id in(select id from...	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
2	10:48:13	update emp1 set salary =60000,holiday =11 where id in(select id from...	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
3	10:48:15	select *from emp1 LIMIT 0, 1000	6 row(s) returned

DELETING ROWS BASED ON ANOTHER TABLE



```
update emp1 set salary =60000,holiday =11 where id in(select id from admin_emp where name = "Suraj");
select *from emp1;

# Deleting Rows Based On Another Table

delete from emp1 where id in(select id from admin_emp where id>=6);
select * from emp1;
```

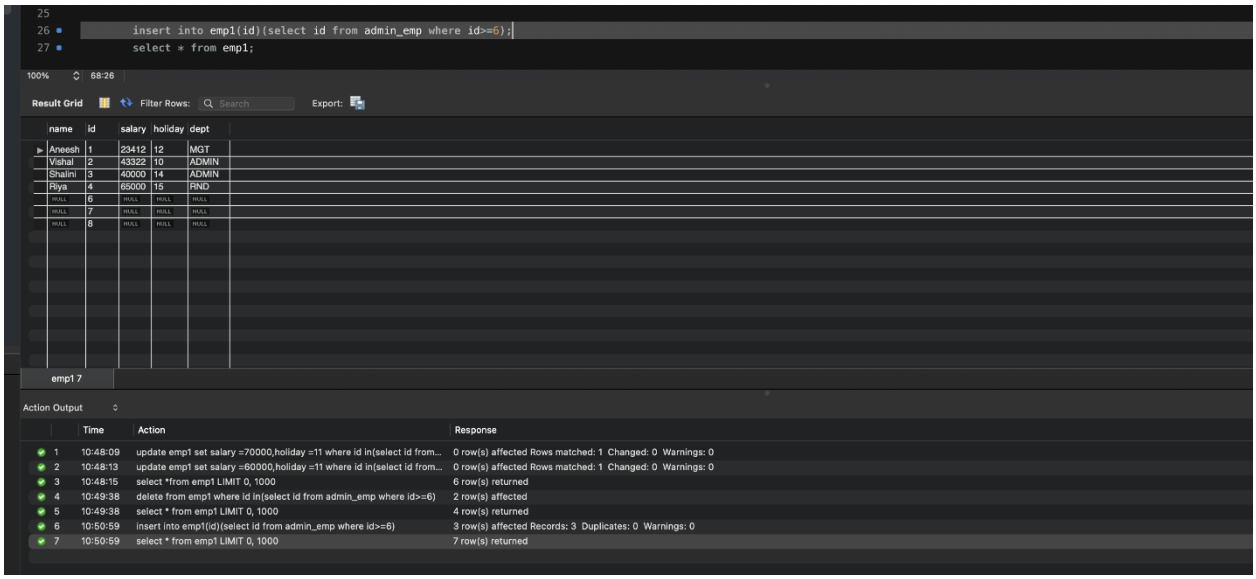
name	id	salary	holiday	dept
Aneesh	1	23412	12	MGT
Vishal	2	43322	10	ADMIN
Shalini	3	40000	14	ADMIN
Riya	4	65000	15	RND

emp1 6

Action Output

	Time	Action	Response
1	10:48:09	update emp1 set salary =70000,holiday =11 where id in(select id from...	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
2	10:48:13	update emp1 set salary =60000,holiday =11 where id in(select id from...	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
3	10:48:15	select *from emp1 LIMIT 0, 1000	6 row(s) returned
4	10:49:38	delete from emp1 where id in(select id from admin_emp where id>=6)	2 row(s) affected
5	10:49:38	select * from emp1 LIMIT 0, 1000	4 row(s) returned

USING SUBQUERY TO INSERT STATEMENT



```
25
26 •        insert into emp1(id)(select id from admin_emp where id>=6);
27 •        select * from emp1;
```

100% 68:26

Result Grid | Filter Rows: Search | Export:

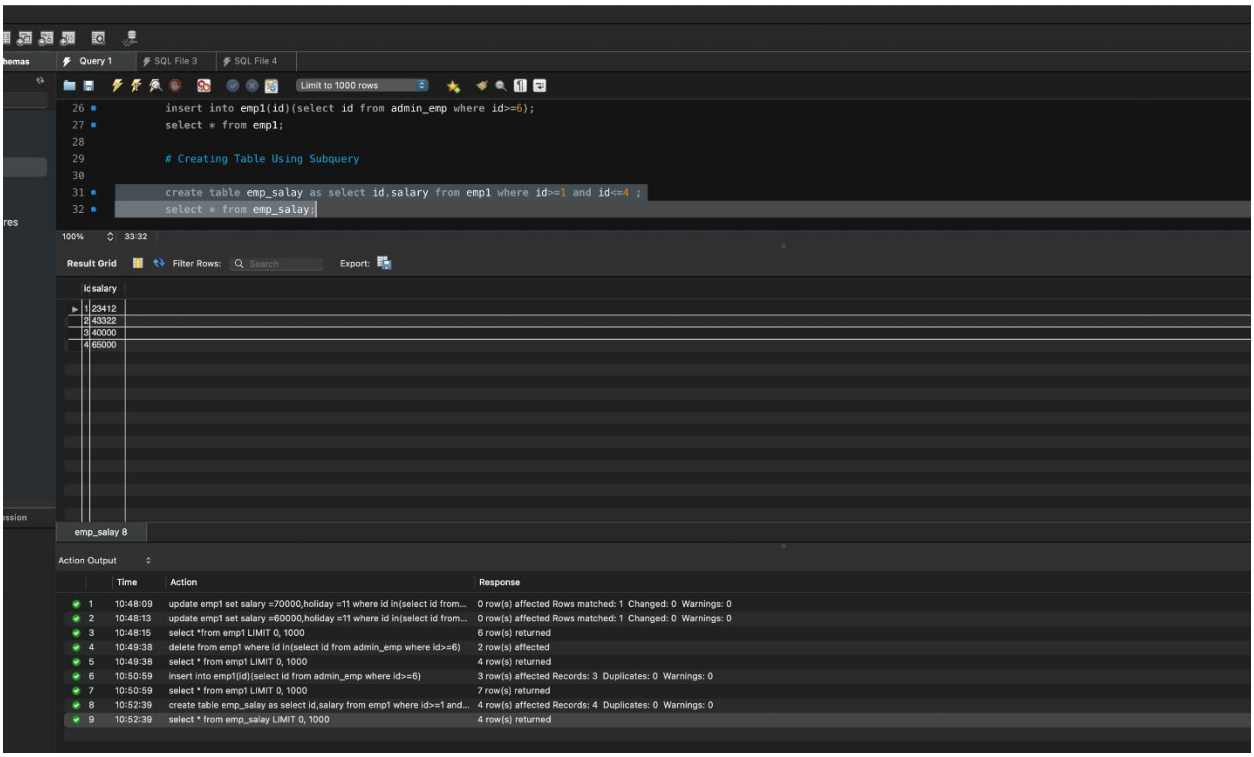
name	id	salary	holiday	dept
Aneesh	1	23412	12	MGT
Vishal	2	43322	10	ADMIN
Shalini	3	40000	14	ADMIN
Riya	4	65000	15	RND
NULL	6	NULL	NULL	NULL
NULL	7	NULL	NULL	NULL
NULL	8	NULL	NULL	NULL

emp1 7

Action Output

	Time	Action	Response
1	10:48:09	update emp1 set salary =70000,holiday =11 where id in(select id from...	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
2	10:48:13	update emp1 set salary =60000,holiday =11 where id in(select id from...	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
3	10:48:15	select *from emp1 LIMIT 0, 1000	6 row(s) returned
4	10:49:38	delete from emp1 where id in(select id from admin_emp where id>=6)	2 row(s) affected
5	10:49:38	select * from emp1 LIMIT 0, 1000	4 row(s) returned
6	10:50:59	insert into emp1(id)(select id from admin_emp where id>=6)	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
7	10:50:59	select * from emp1 LIMIT 0, 1000	7 row(s) returned

CREATING TABLE USING SUBQUERY



Query 1 | SQL File 3 | SQL File 4

Limit to 1000 rows

```
26 •        insert into emp1(id)(select id from admin_emp where id>=6);
27 •        select * from emp1;
28
29          # Creating Table Using Subquery
30
31 •        create table emp_salay as select id,salary from emp1 where id>=1 and id<=4 ;
32 •        select * from emp_salay;
```

100% 33:32

Result Grid | Filter Rows: Search | Export:

id	salary
1	23412
2	43322
3	40000
4	65000

emp_salay 8

Action Output

	Time	Action	Response
1	10:48:09	update emp1 set salary =70000,holiday =11 where id in(select id from...	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
2	10:48:13	update emp1 set salary =60000,holiday =11 where id in(select id from...	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
3	10:48:15	select *from emp1 LIMIT 0, 1000	6 row(s) returned
4	10:49:38	delete from emp1 where id in(select id from admin_emp where id>=6)	2 row(s) affected
5	10:49:38	select * from emp1 LIMIT 0, 1000	4 row(s) returned
6	10:50:59	insert into emp1(id)(select id from admin_emp where id>=6)	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
7	10:50:59	select * from emp1 LIMIT 0, 1000	7 row(s) returned
8	10:52:39	create table emp_salay as select id,salary from emp1 where id>=1 and...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0
9	10:52:39	select * from emp_salay LIMIT 0, 1000	4 row(s) returned

EXPERIMENT-07

AIM-Join queries:
a) Equi-join
b) Non Equi-join
c) Outer join
d) Left outer join
e) Right outer join
f) Self join

CREATE DATABASE

```
create table stud(id int,regno int,name varchar(10),year int,marks int);
insert into stud values(1,23, 'Vishal',3,18), (2,24,'Naman',3,20),
(3,25,'Shalini',5,19), (7,26, 'Priyanshu',2,20), (8,27, 'Aneesh',3,20);
select * from stud;
create table subject_details(id int,subject_name varchar(10), year int);
insert into subject_details values(1,'maths',3), (2,'eng',3), (3,'CD',5), (4,'dbms',5),
(5,COA',3), (6,'CN',5), (7,'OS',3);
select * from subject_details;
```

EQUI JOIN

```
select * from stud,subject_details where
subject_details year = stud.year;
```

NON EQUI JOIN

```
select * from stud,subject_details where stud.marks>18;
```

OUTER JOIN

```
select * from stud,subject_details where subject_details.id = stud.id;
```

LEFT OUTER JOIN

```
select * from stud LEFT OUTER JOIN subject_details on stud.id=
subject_details.id;
```

RIGHT OUTER JOIN

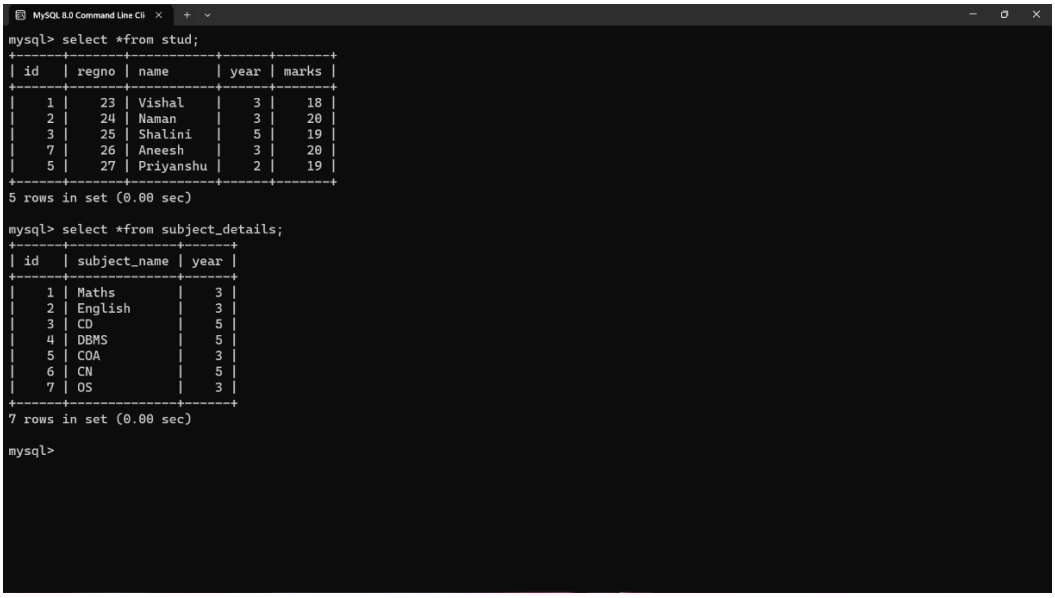
```
select * from stud RIGHT OUTER JOIN subject_details on stud.id=
subject_details.id;
```

SELF JOIN

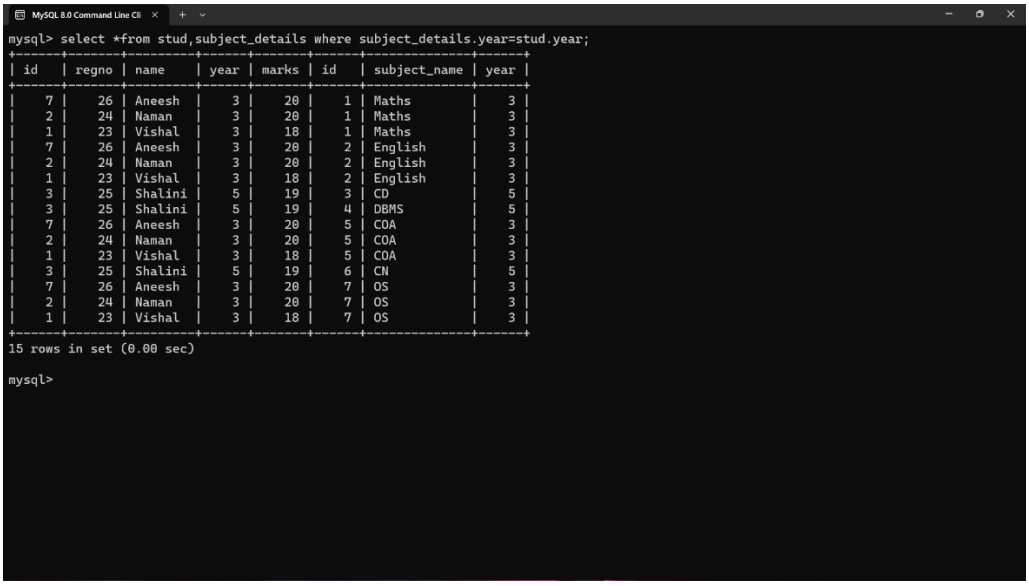
- select a.id,a.name,b.id,b.name from stud a, stud b where a.id=byear;
- select a.id,a.subject_name.b.id,b.subject_name from subject_details a, subject_details b where a.id=b.year;

OUTPUT

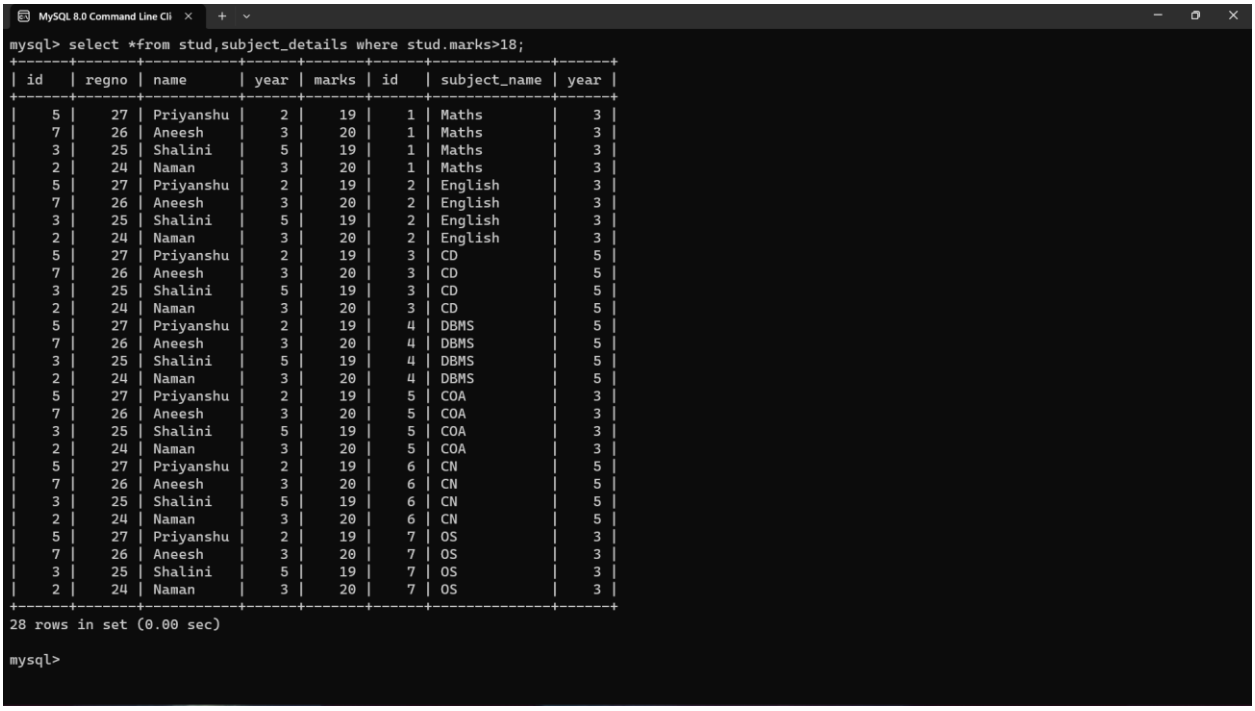
CREATE DATABASE



EQUI JOIN

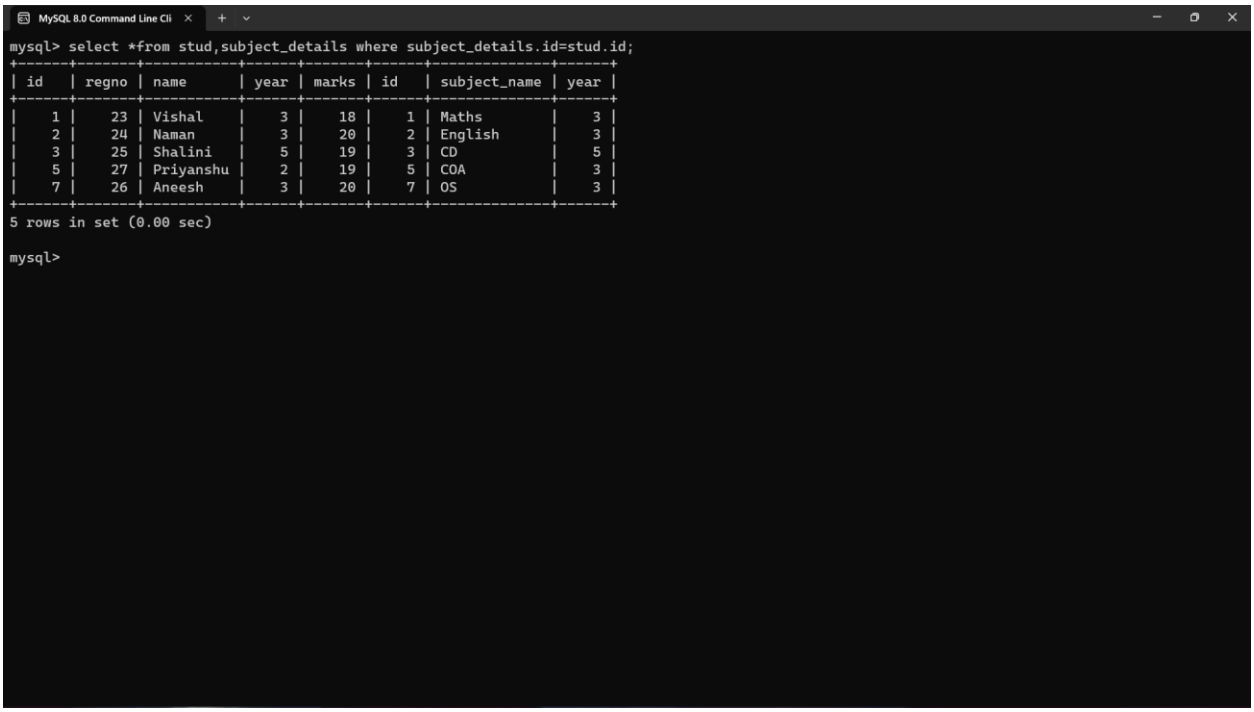


NON EQUI JOIN



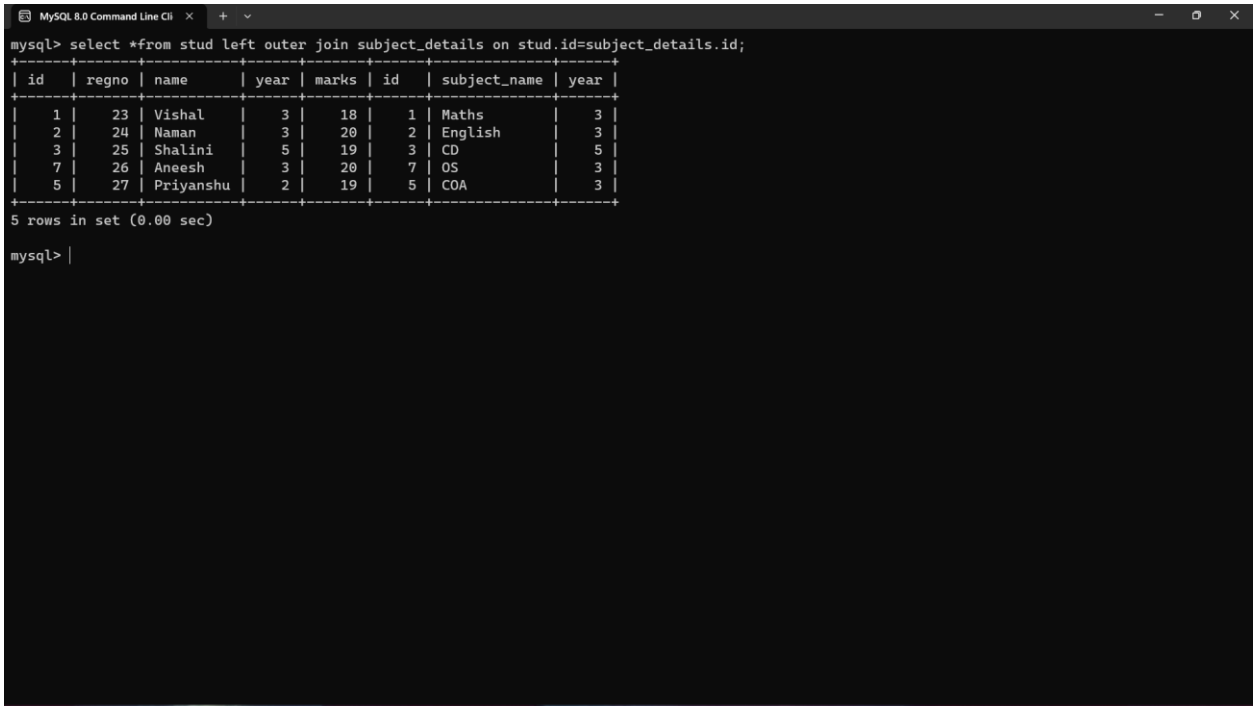
```
MySQL 8.0 Command Line Cli

mysql> select *from stud,subject_details where stud.marks>18;
+------+-------+-----------+------+-------+------+--------------+------+
| id   | regno | name      | year | marks | id   | subject_name | year |
+------+-------+-----------+------+-------+------+--------------+------+
|    5 |    27 | Priyanshu |    2 |    19 |    1 | Maths        |    3 |
|    7 |    26 | Aneesh    |    3 |    20 |    1 | Maths        |    3 |
|    3 |    25 | Shalini   |    5 |    19 |    1 | Maths        |    3 |
|    2 |    24 | Naman     |    3 |    20 |    1 | Maths        |    3 |
|    5 |    27 | Priyanshu |    2 |    19 |    2 | English      |    3 |
|    7 |    26 | Aneesh    |    3 |    20 |    2 | English      |    3 |
|    3 |    25 | Shalini   |    5 |    19 |    2 | English      |    3 |
|    2 |    24 | Naman     |    3 |    20 |    2 | English      |    3 |
|    5 |    27 | Priyanshu |    2 |    19 |    3 | CD           |    5 |
|    7 |    26 | Aneesh    |    3 |    20 |    3 | CD           |    5 |
|    3 |    25 | Shalini   |    5 |    19 |    3 | CD           |    5 |
|    2 |    24 | Naman     |    3 |    20 |    3 | CD           |    5 |
|    5 |    27 | Priyanshu |    2 |    19 |    4 | DBMS         |    5 |
|    7 |    26 | Aneesh    |    3 |    20 |    4 | DBMS         |    5 |
|    3 |    25 | Shalini   |    5 |    19 |    4 | DBMS         |    5 |
|    2 |    24 | Naman     |    3 |    20 |    4 | DBMS         |    5 |
|    5 |    27 | Priyanshu |    2 |    19 |    5 | COA          |    3 |
|    7 |    26 | Aneesh    |    3 |    20 |    5 | COA          |    3 |
|    3 |    25 | Shalini   |    5 |    19 |    5 | COA          |    3 |
|    2 |    24 | Naman     |    3 |    20 |    5 | COA          |    3 |
|    5 |    27 | Priyanshu |    2 |    19 |    6 | CN           |    5 |
|    7 |    26 | Aneesh    |    3 |    20 |    6 | CN           |    5 |
|    3 |    25 | Shalini   |    5 |    19 |    6 | CN           |    5 |
|    2 |    24 | Naman     |    3 |    20 |    6 | CN           |    5 |
|    5 |    27 | Priyanshu |    2 |    19 |    7 | OS           |    3 |
|    7 |    26 | Aneesh    |    3 |    20 |    7 | OS           |    3 |
|    3 |    25 | Shalini   |    5 |    19 |    7 | OS           |    3 |
|    2 |    24 | Naman     |    3 |    20 |    7 | OS           |    3 |
+------+-------+-----------+------+-------+------+--------------+------+
28 rows in set (0.00 sec)

mysql>
```

OUTER JOIN

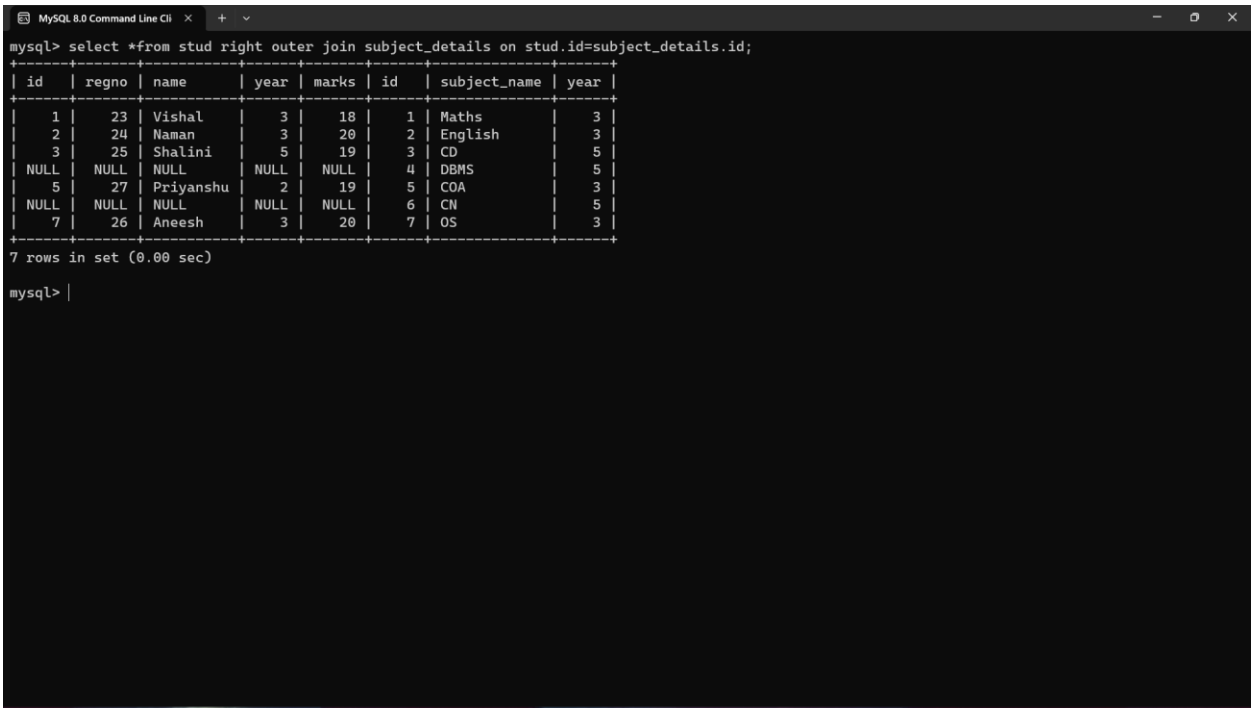


```
MySQL 8.0 Command Line Cli

mysql> select *from stud,subject_details where subject_details.id=stud.id;
+------+-------+-----------+------+-------+------+--------------+------+
| id   | regno | name      | year | marks | id   | subject_name | year |
+------+-------+-----------+------+-------+------+--------------+------+
|    1 |    23 | Vishal    |    3 |    18 |    1 | Maths        |    3 |
|    2 |    24 | Naman     |    3 |    20 |    2 | English      |    3 |
|    3 |    25 | Shalini   |    5 |    19 |    3 | CD           |    5 |
|    5 |    27 | Priyanshu |    2 |    19 |    5 | COA          |    3 |
|    7 |    26 | Aneesh    |    3 |    20 |    7 | OS           |    3 |
+------+-------+-----------+------+-------+------+--------------+------+
5 rows in set (0.00 sec)

mysql>
```

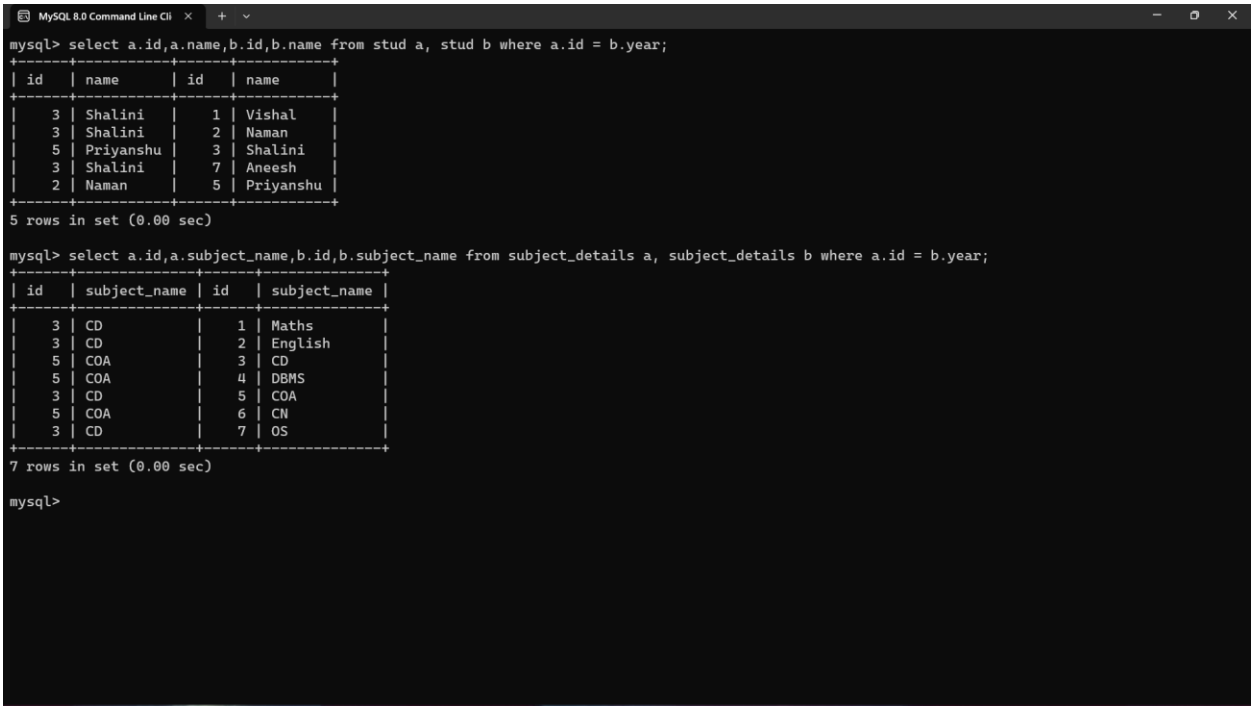
LEFT OUTER JOIN



RIGHT OUTER JOIN



SELF JOIN


MySQL 8.0 Command Line Cli
mysql> select a.id,a.name,b.id,b.name from stud a, stud b where a.id = b.year;
+------+-----------+------+-----------+
| id | name | id | name |
+------+-----------+------+-----------+
3	Shalini	1	Vishal
3	Shalini	2	Naman
5	Priyanshu	3	Shalini
3	Shalini	7	Aneesh
2	Naman	5	Priyanshu
+------+-----------+------+-----------+
5 rows in set (0.00 sec)
mysql> select a.id,a.subject_name,b.id,b.subject_name from subject_details a, subject_details b where a.id = b.year;
+------+--------------+------+--------------+
| id | subject_name | id | subject_name |
+------+--------------+------+--------------+
3	CD	1	Maths
3	CD	2	English
5	COA	3	CD
5	COA	4	DBMS
3	CD	5	COA
5	COA	6	CN
3	CD	7	OS
+------+--------------+------+--------------+
7 rows in set (0.00 sec)
mysql>

EXPERIMENT-08

AIM - Set operators and views:

a) Union
b) Union all
c) Intersect
d) Minus
e) Creation of views
f) Modifications of views
g) Deletion of view
h) Rownum query

UNION

```
select * from
A_table union
select * from B_table;
```

UNION ALL

```
select * from A_table union
all
select * from B_table;
```

INTERSECT

```
select * from A_table Pp
intersect
select * from B_table;
```

MINUS

```
select* from A
minus
select* from B_table;
```

CREATION OF VIEWS

```
create view academics as
select stu_name from
student;
```

MODIFICATION OF VIEWS

alter view academics compile;

DELETION OF VIEW

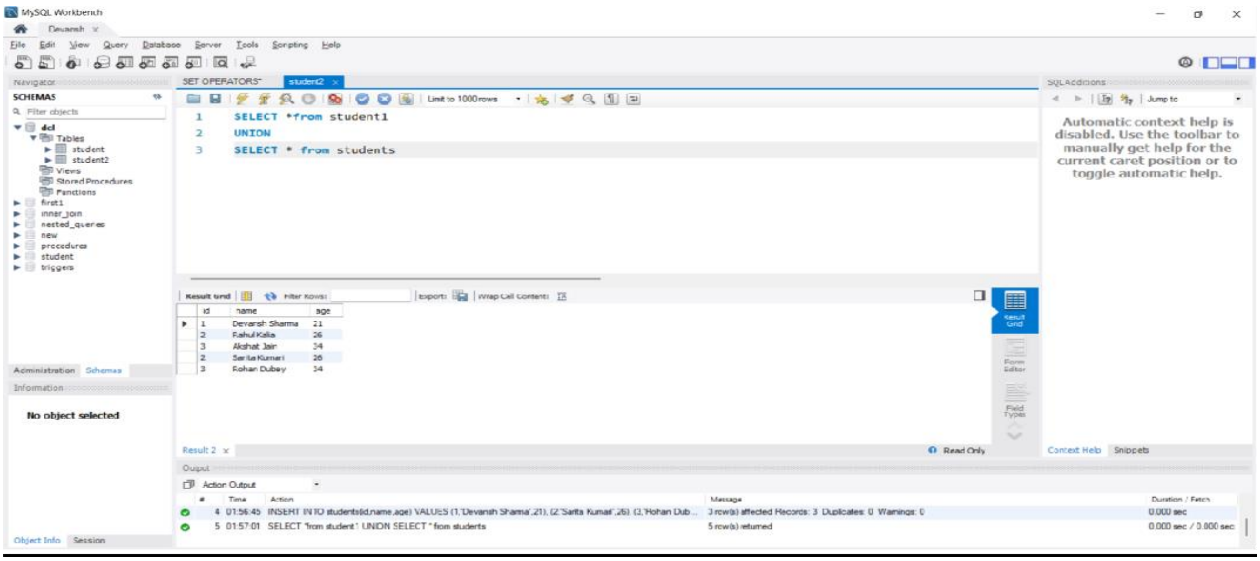
drop view academics;

ROWNUM QUERY

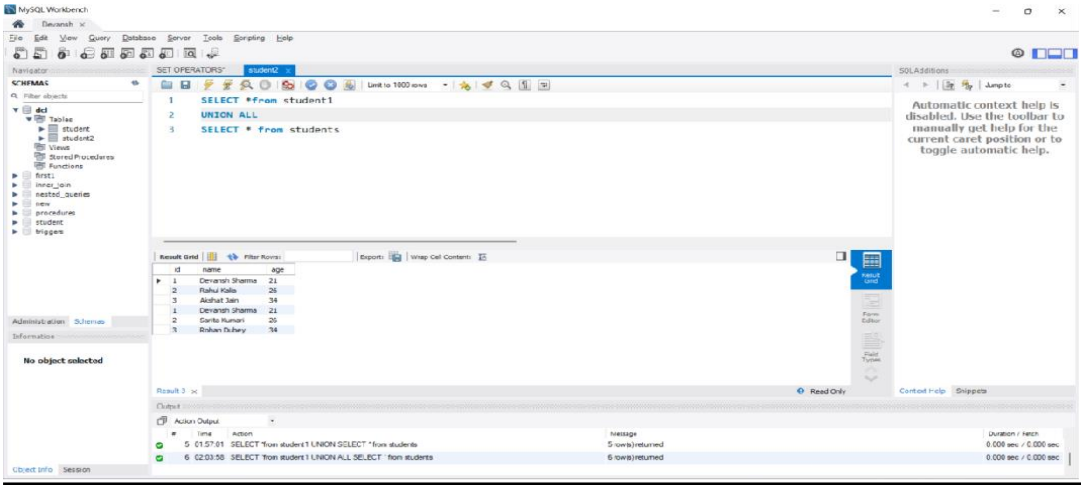
select from stud where rownum<10;
select from stud where rownum<4 order by name,
update stud set id-rownum;
select * from stud;

OUTPUT

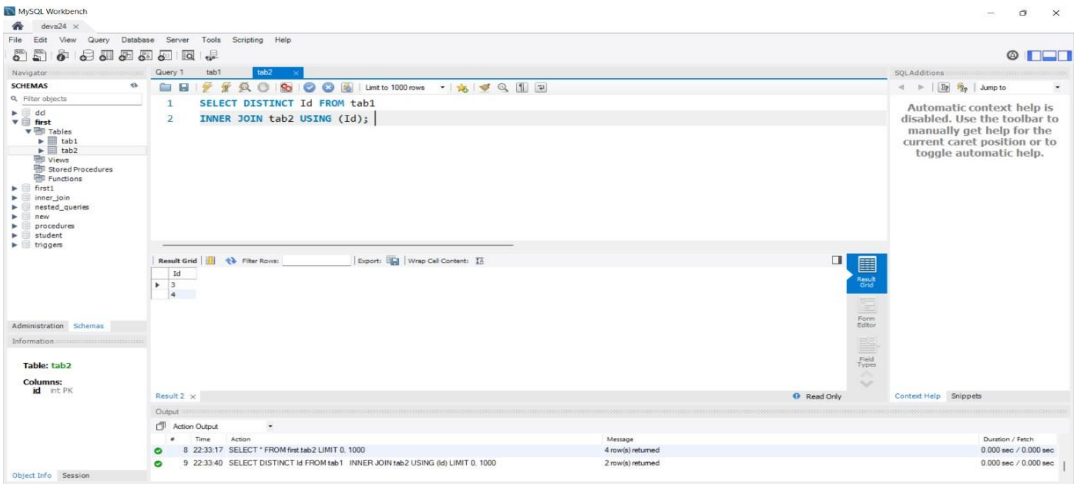
UNION



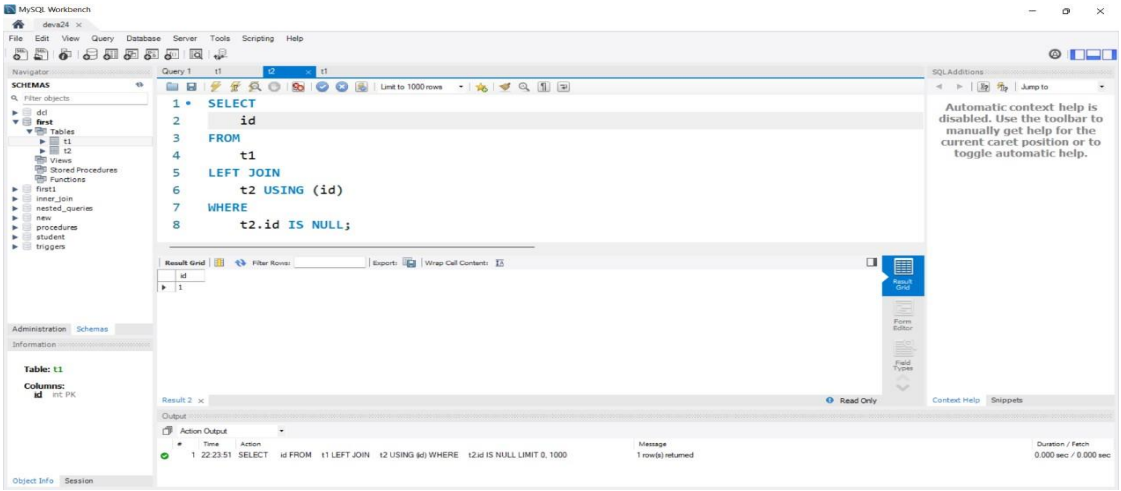
UNION ALL



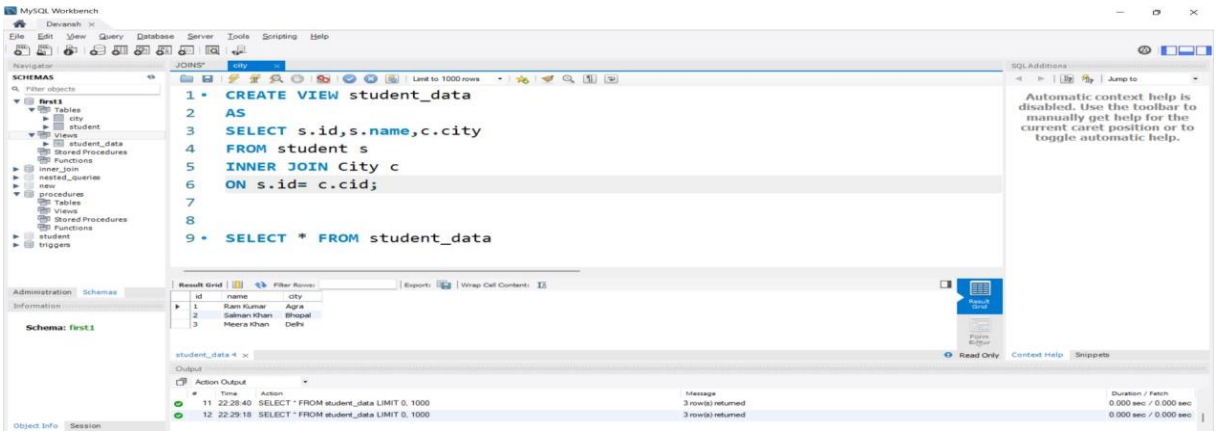
INTERSECT



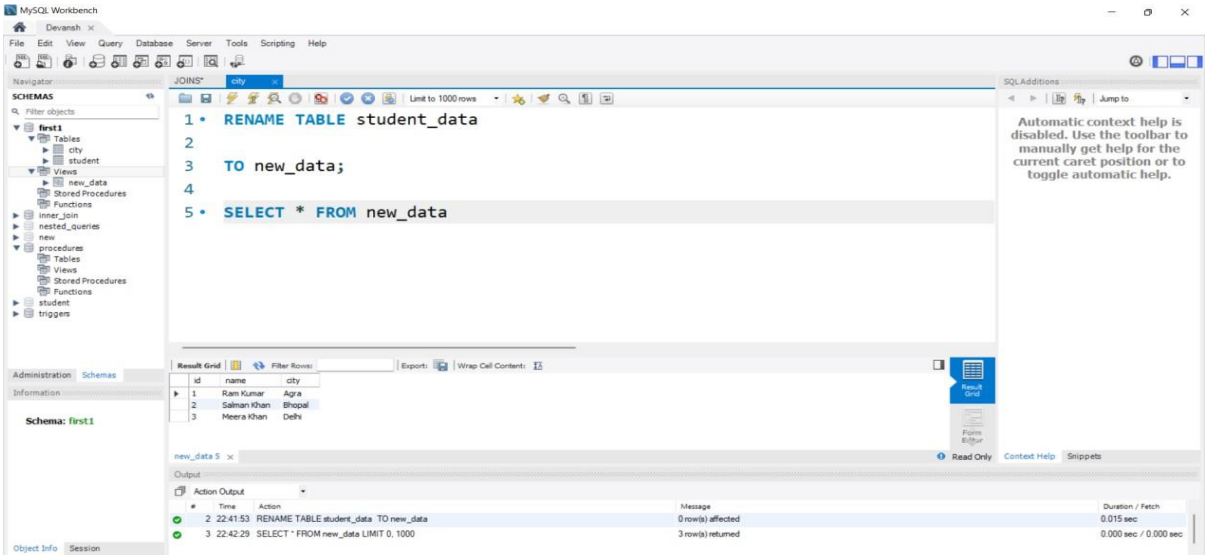
MINUS



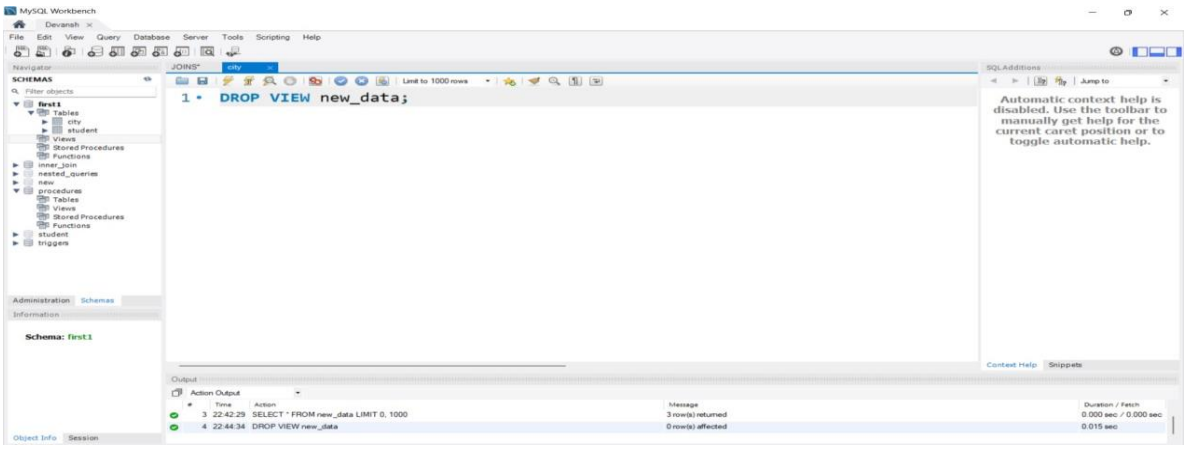
CREATION OF VIEW



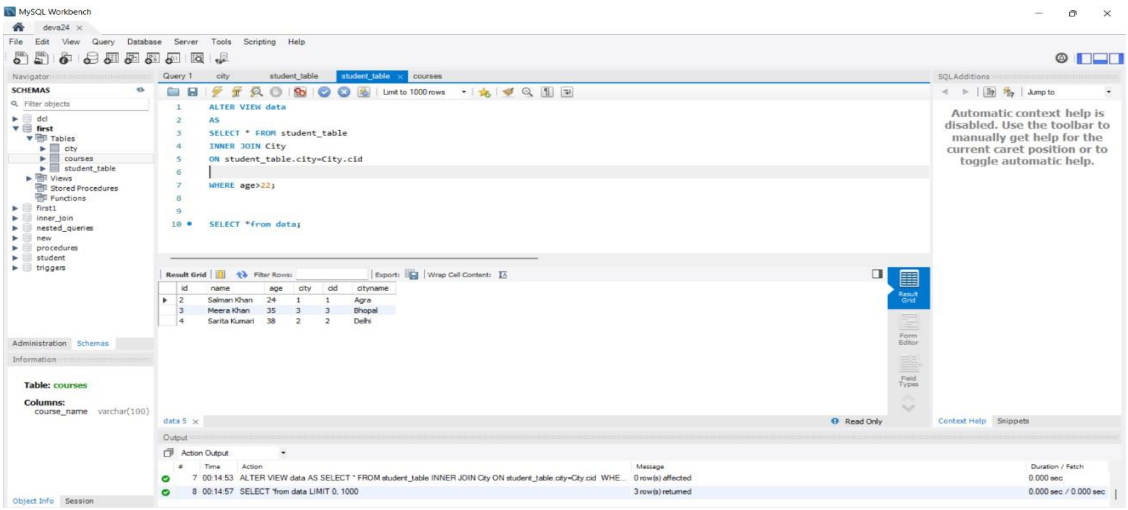
MODIFICATION OF VIEW



DELETION OF VIEW



ROWNUM QUERY



EXPERIMENT -09
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:

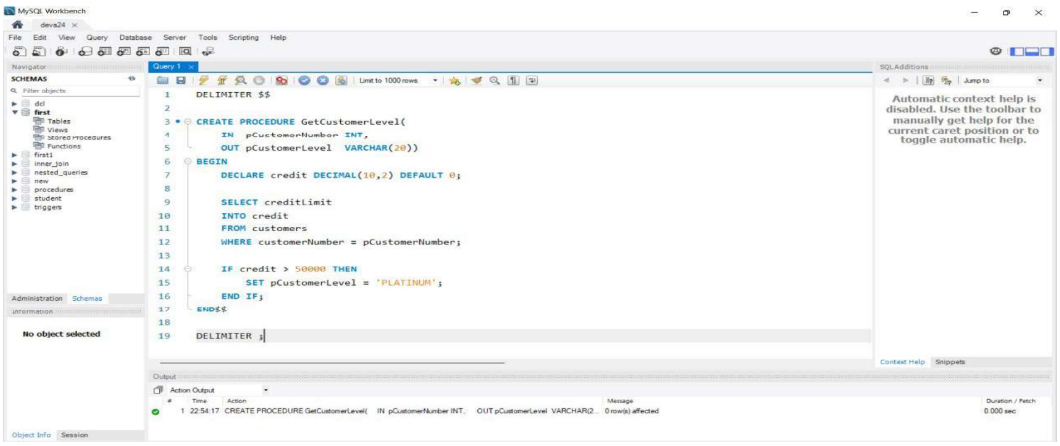
a) Conditional Statement

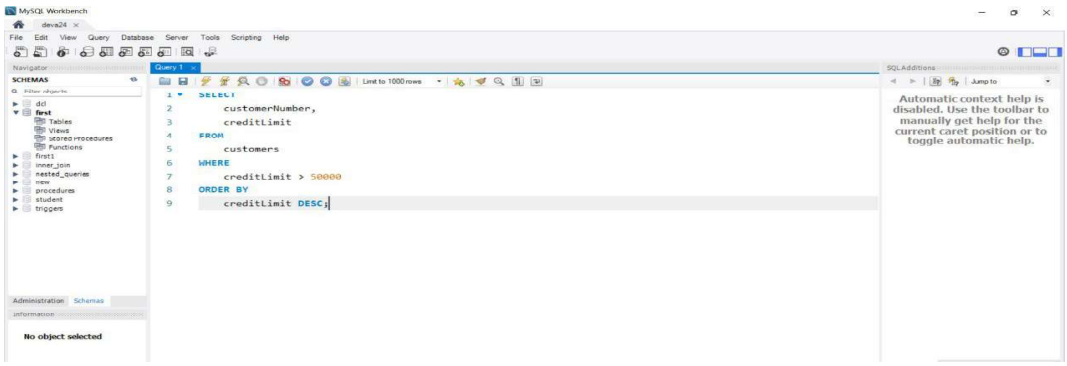
FOR CREATING PROCEDURE

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

FOR EXECUTION

```
SELECT
   customerNum
   ber,
   creditLimit
FROM
   custo
mers
WHERE
   creditLimit >
50000ORDER 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

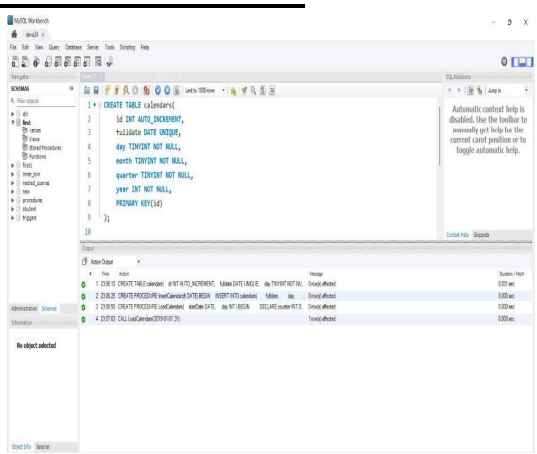
Iterative Statement

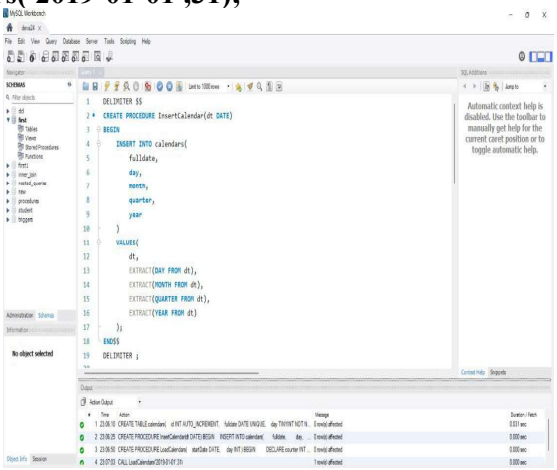
```
CREATE TABLE calendars (
id INT AUTO_INCREMENT,
fulldate DATE UNIQUE, day TINYINT NOT NULL,
month TINYINT NOT NULL,
quarter TINYINT NOT NULL,year INT NOT NULL, PRIMARY KEY(id));
```

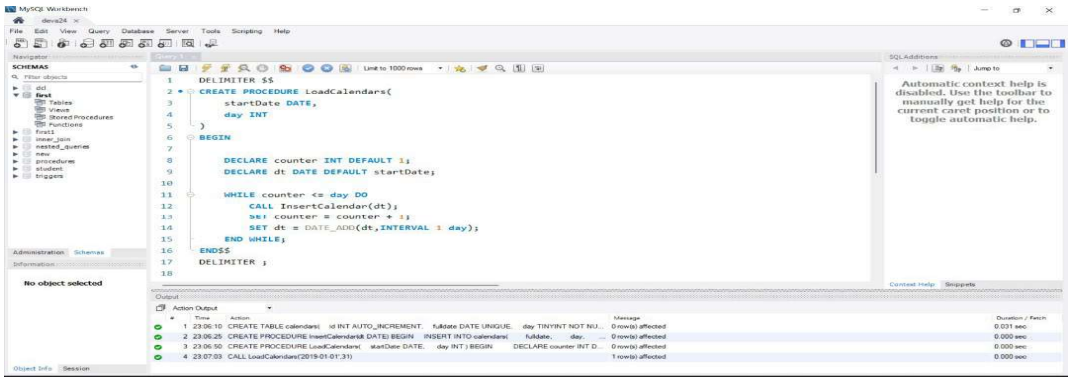
FOR CREATING PROCEDURE

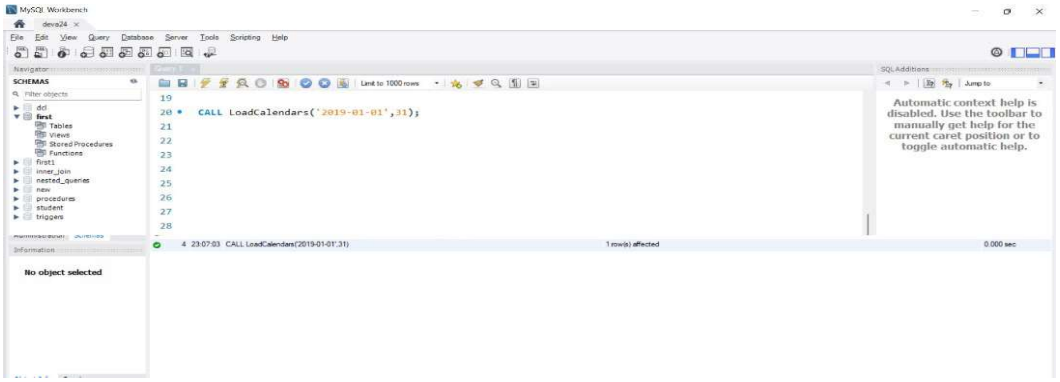
```
DELIMITER $$
CREATE PROCEDURE InsertCalendar(dtDATE)
BEGIN
INSERT INTO calendars(fulldate,
day, month, quarter,year) VALUES(
dt,
EXTRACT(DAY FROM dt), EXTRACT(MONTH FROM dt), EXTRACT(QUARTER FROM dt),
EXTRACT(YEAR FROM dt)
);
END$$ DELIMITER ;

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

RESULT: The program for PL/SQL conditional and iterative statements in RDBMS is implemented successfully and output is verified.

EXPERIMENT 10

PL/SQL Procedures on sample exercise.

Aim: To Execute Procedure in MySql

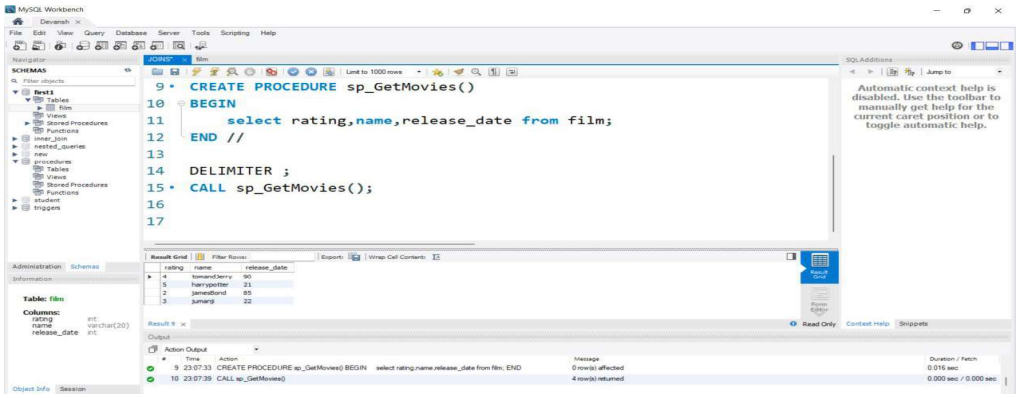
Theory:- A procedure (often called a stored procedure) is a collection of pre- compiled SQL statements stored inside the database. It is a subroutine or a subprogram in the regular computing language. A procedure always contains a name, parameter lists, and SQL statements. We can invoke the procedures 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 //
PROCEDURE sp_GetMovies()BEGIN
select rating,name,release_date from film;END //

DELIMITER ;

CALL sp_GetMovies();
```



RESULT: The program for PL/SQL Procedures on sample exercise in RDBMS is implemented successfully and the output is verified.

EXPERIMENT 11

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 functionis 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_da
te)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');
```

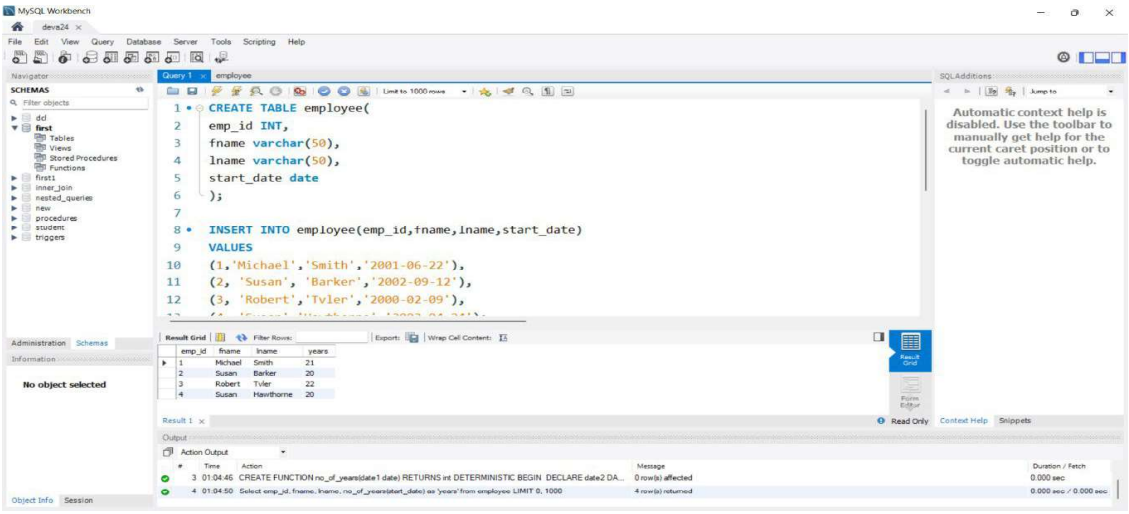
CREATING FUNCTION

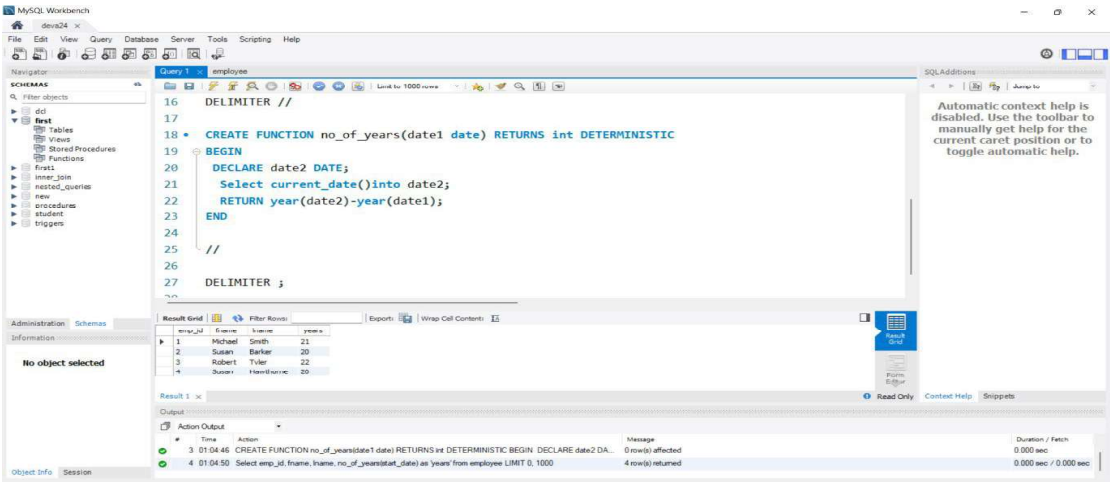
```
DELIMITER //

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:-





RESULT: The program for PL/SQL PL/SQL Functions in RDBMS is implemented successfully and output is verified.

EXPERIMENT 12

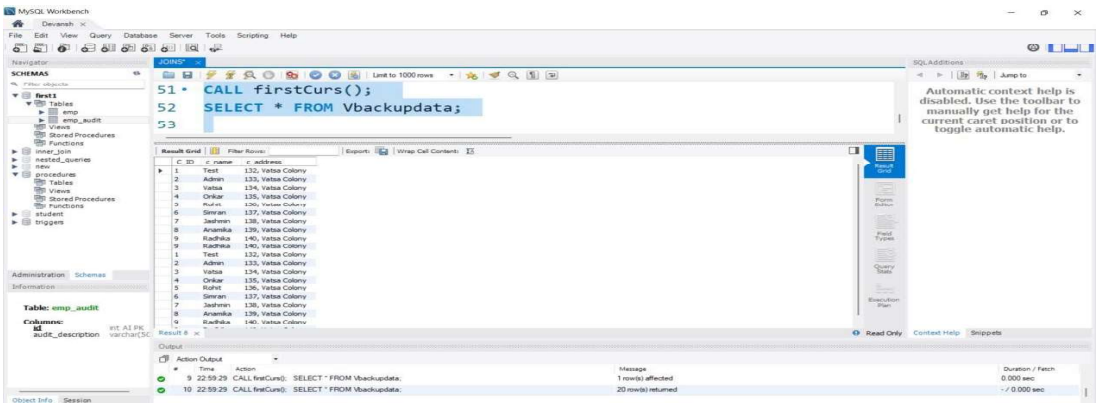
PL/SQL Cursors

Aim:- To Execute Cursors in MySql

Theory:- A cursor allows you to iterate a set of rows returned by a query and processeach 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, VatsaColony'),
('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
```



RESULT: The program for **PL/SQL Cursors** in RDBMS is implemented successfully and output is verified.

EXPERIMENT 13

PL/SQL Exception Handling

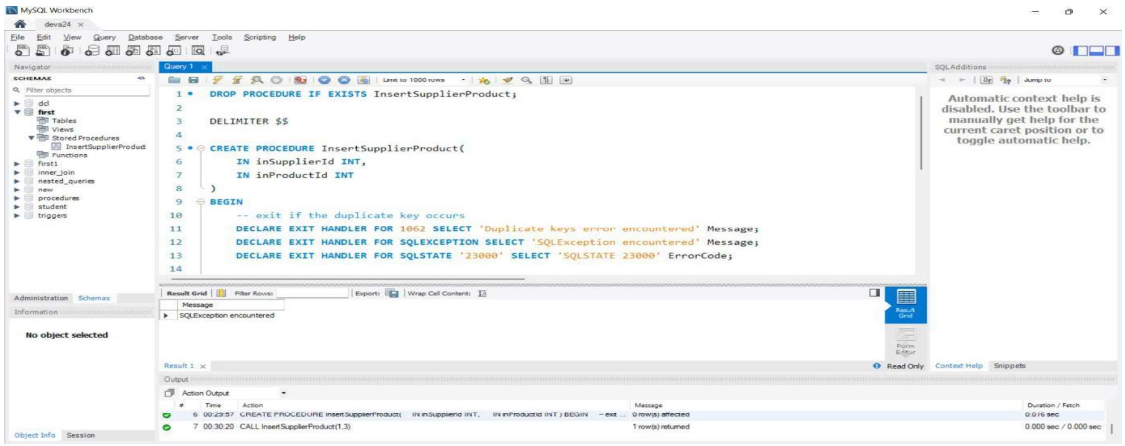
Aim- PL/SQL exception handling.

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 handlersthat 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



RESULT: The program for **PL/SQL EXCEPTION handling** in RDBMS is implemented successfully and output is verified.

EXPERIMENT 14

PL/SQL Trigger

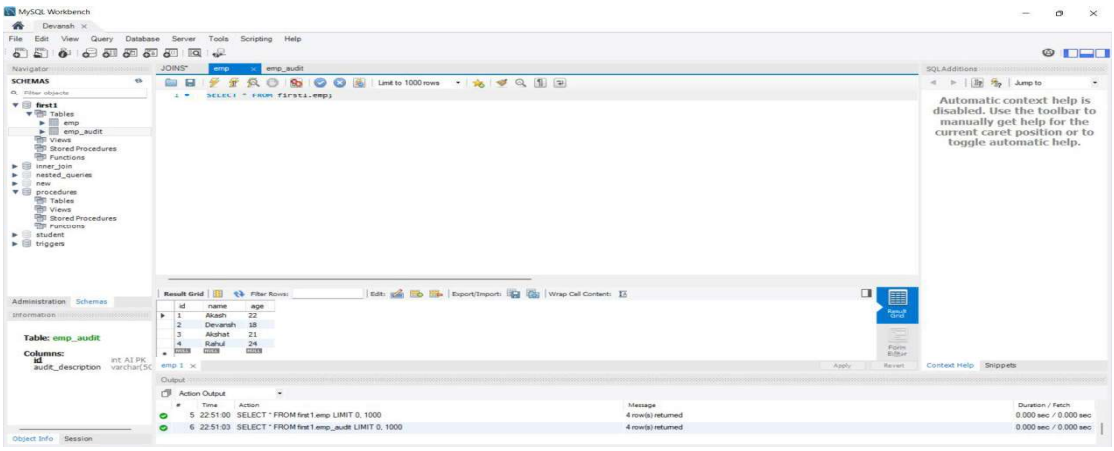
Aim:- To Execute Triggers in MYsql

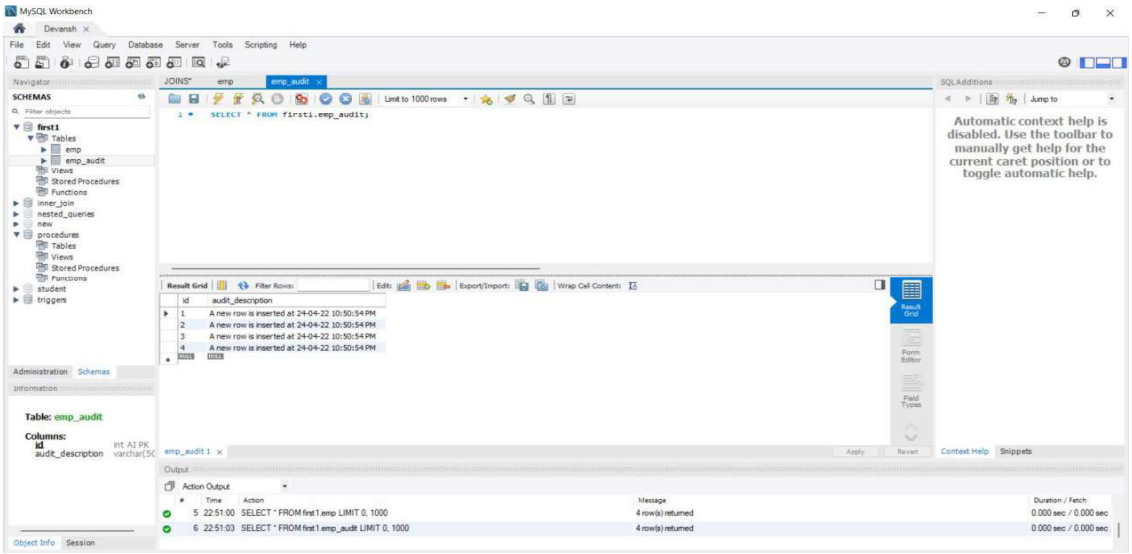
Theory:- A trigger in MySQL is a set of SQL statements that reside in a system catalog. It is a special type of stored procedure that is invoked automatically in response to an event. Each trigger is associated with a table, which is activated on anyDML statement such as INSERT, UPDATE, or DELETE.

A trigger is called a special procedure because it cannot be called directly like a storedprocedure. 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_AfterInsetEmpAFTER
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);
```





RESULT: The program for **PL/SQL trigger** in RDBMS is implemented successfully and output is verified.

EXPERIMENT 15

Frame and Execute PL/SQL Cursor & ExceptionalHandling

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 Basedon 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 managementsystem 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 administrativemanagement 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 topapers.

MODULES

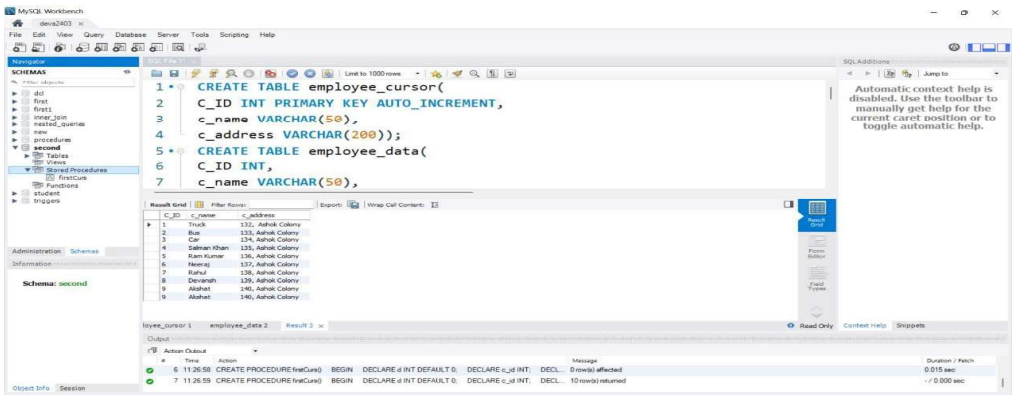
- Tenant
- Owner
- Contract
- Payment

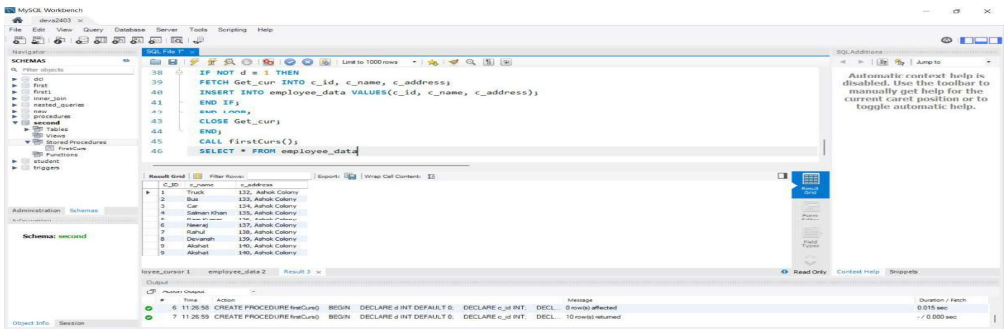
OPERATION:

First will be the login page where the User will login using their ID andPassword.
Next the user will enter the details like the requirements of the house he needsand 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 thedetails 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 whichthey can download.

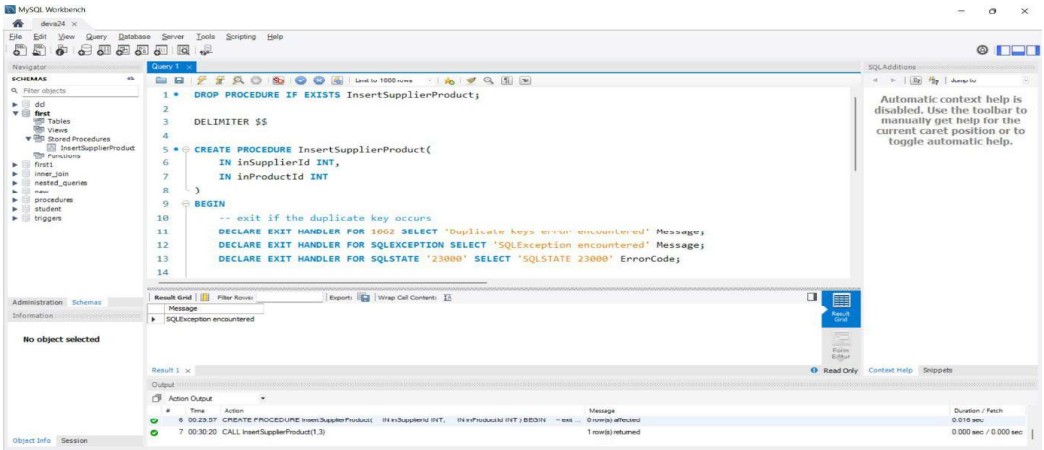
OUTPUT:-

a) Cursor Output





b) Exception Handling



RESULT: The program for Frame and Execute PL/SQL Cursor & ExceptionalHandling in RDBMS is implemented successfully and output is verified.