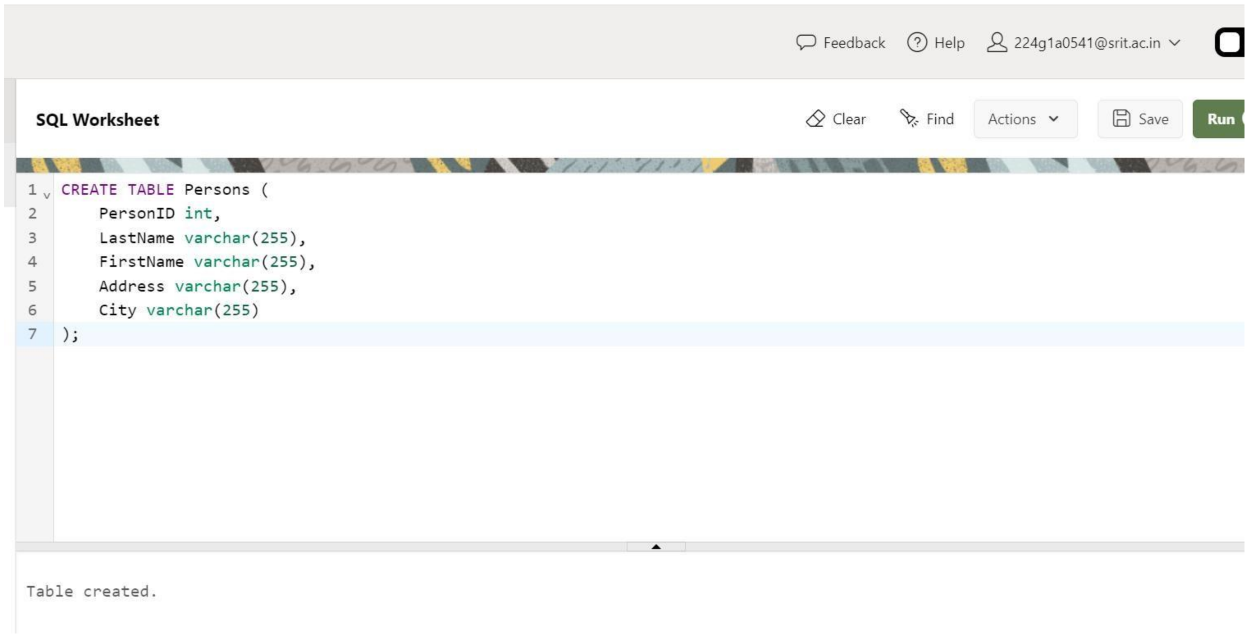
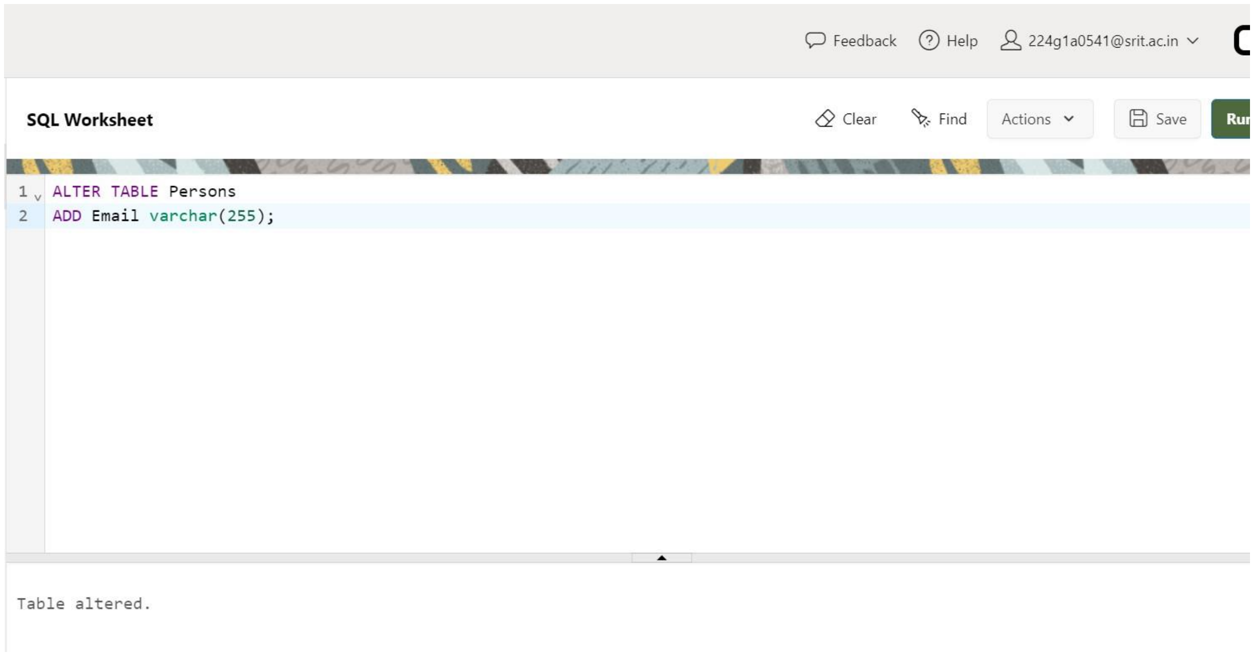
## Experiment 1

1. Write SQL queries to CREATE TABLES for various databases using DDL commands (i.e. CREATE, ALTER, DROP, TRUNCATE).



```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

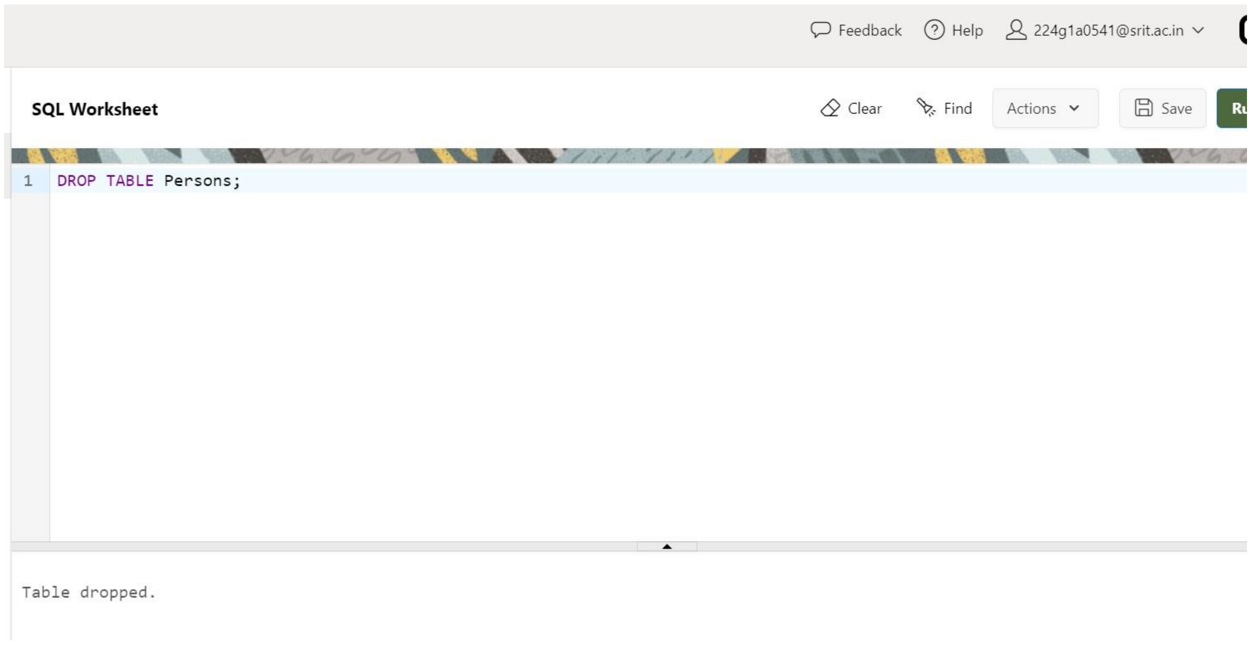
```
Table created.
```



```
ALTER TABLE Persons
ADD Email varchar(255);
```

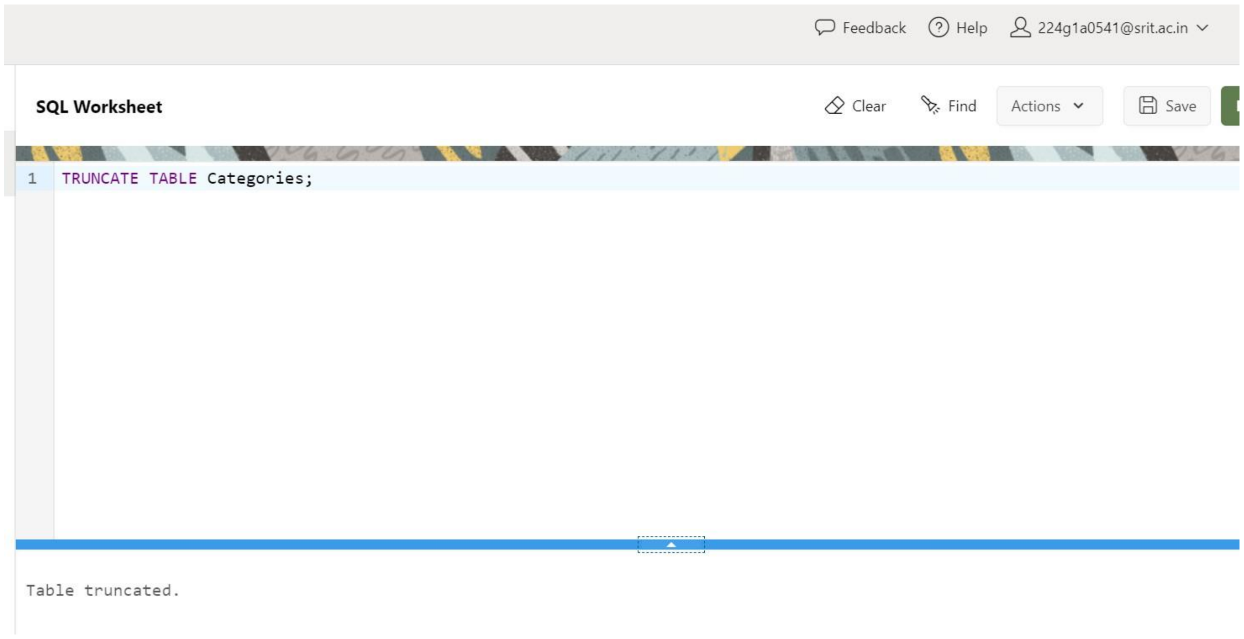
```
Table altered.
```

224G1A0541



```
DROP TABLE Persons;
```

```
Table dropped.
```

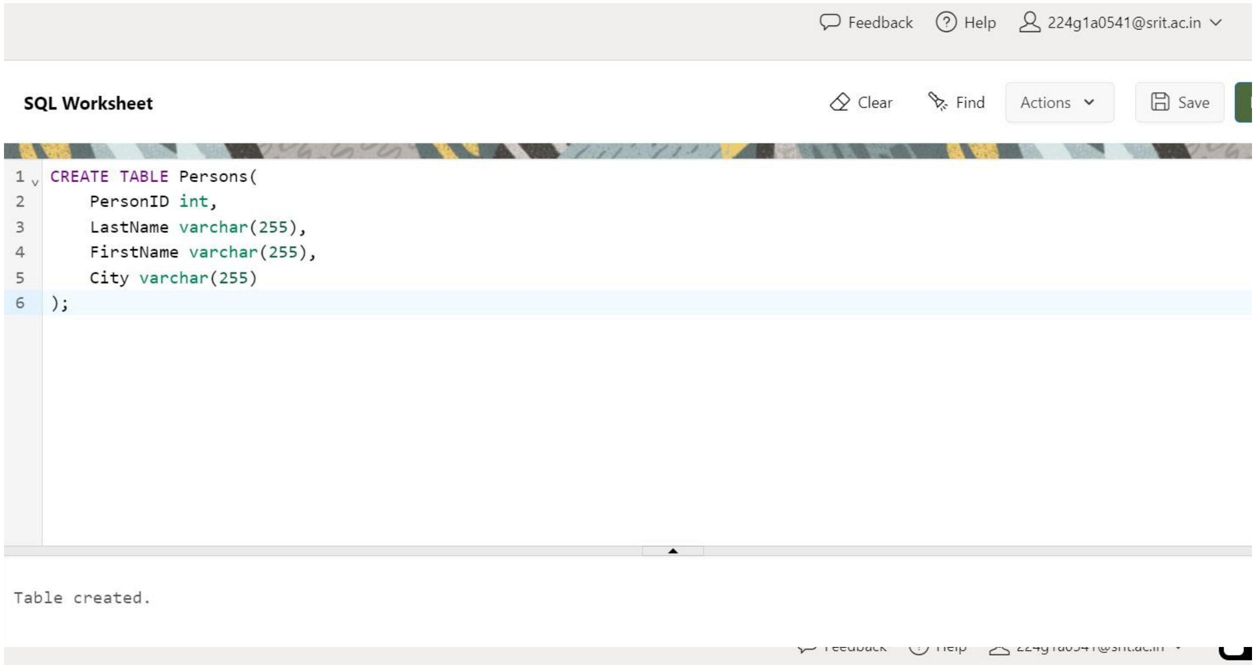


```
TRUNCATE TABLE Categories;
```

```
Table truncated.
```

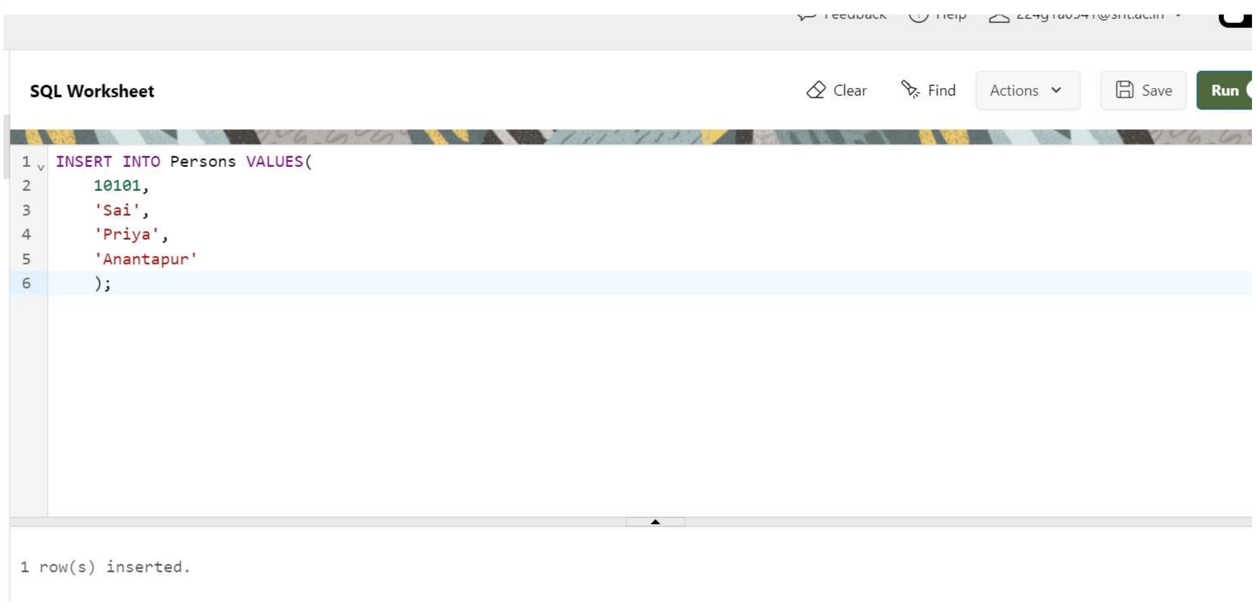
## Experiment 2

2. Write SQL queries to MANIPULATE TABLES for various databases using DML commands (i.e. INSERT, SELECT, UPDATE, DELETE).



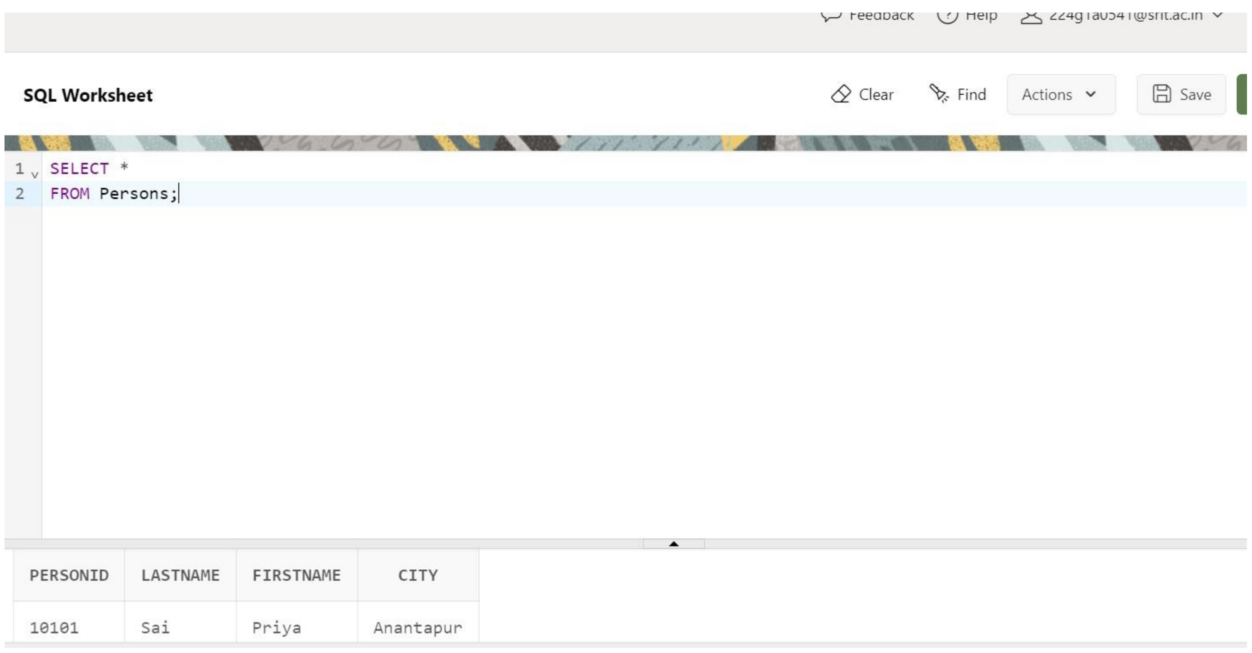
```sql
CREATE TABLE Persons(
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    City varchar(255)
);
```

```
Table created.
```



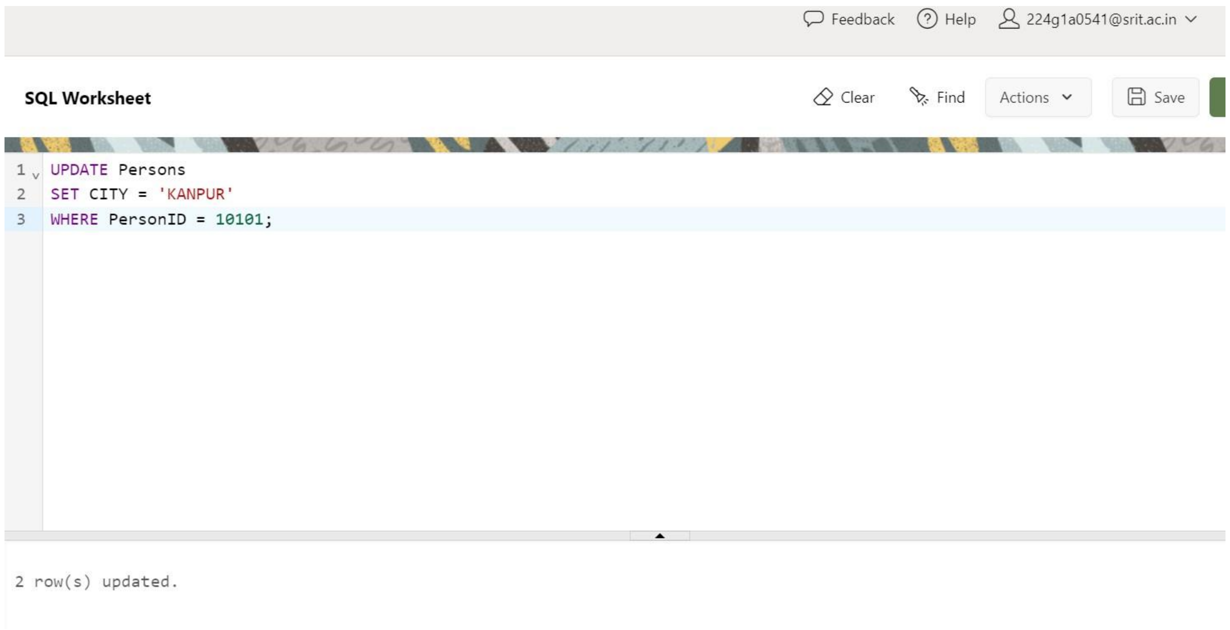
```sql
INSERT INTO Persons VALUES(
    10101,
    'Sai',
    'Priya',
    'Anantapur'
    );
```

```
1 row(s) inserted.
```



```
SELECT *
FROM Persons;
```

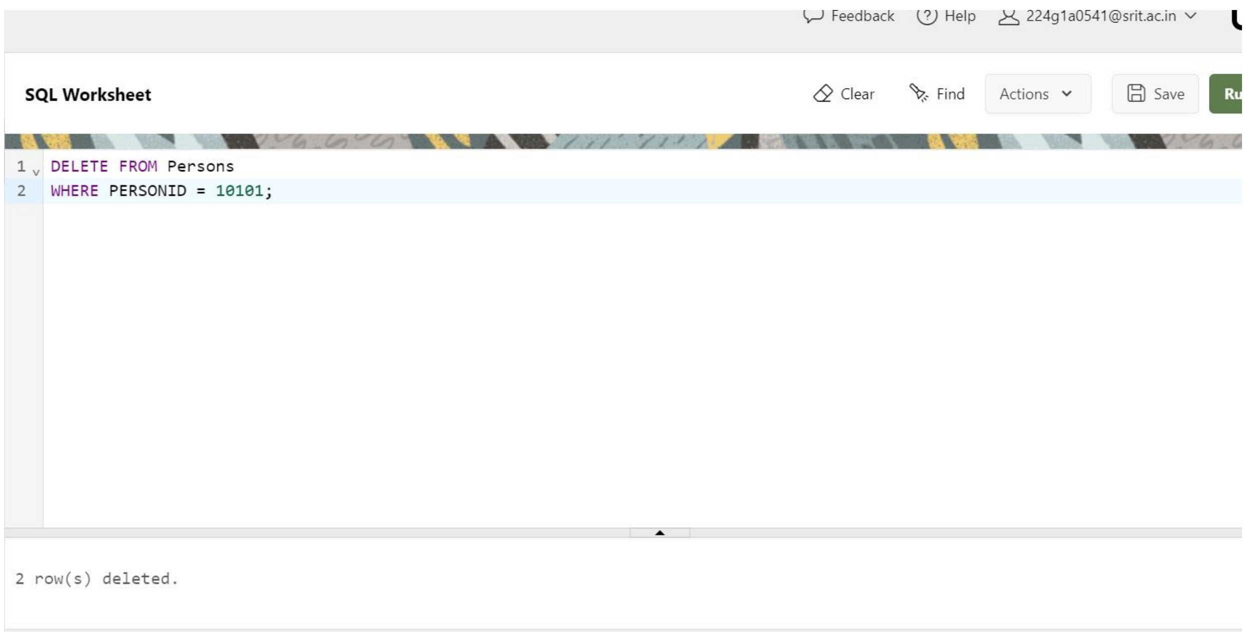
| PERSONID | LASTNAME | FIRSTNAME | CITY |
|---|---|---|---|
| 10101 | Sai | Priya | Anantapur |



```
UPDATE Persons
SET CITY = 'KANPUR'
WHERE PersonID = 10101;
```

```
2 row(s) updated.
```

224G1A0541

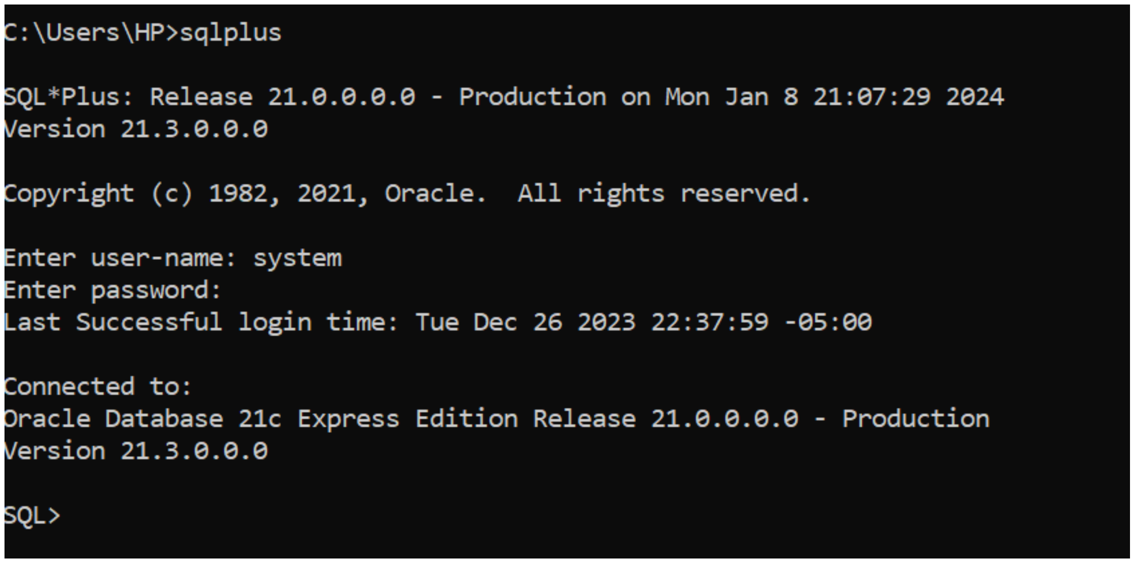


```
DELETE FROM Persons
WHERE PERSONID = 10101;
```

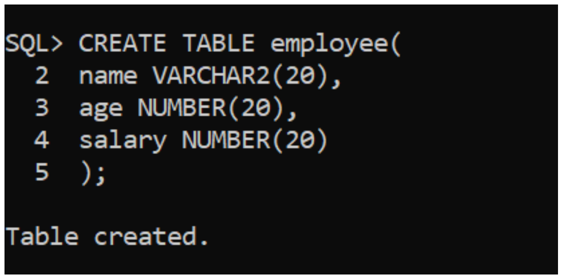
```
2 row(s) deleted.
```

## Experiment 3

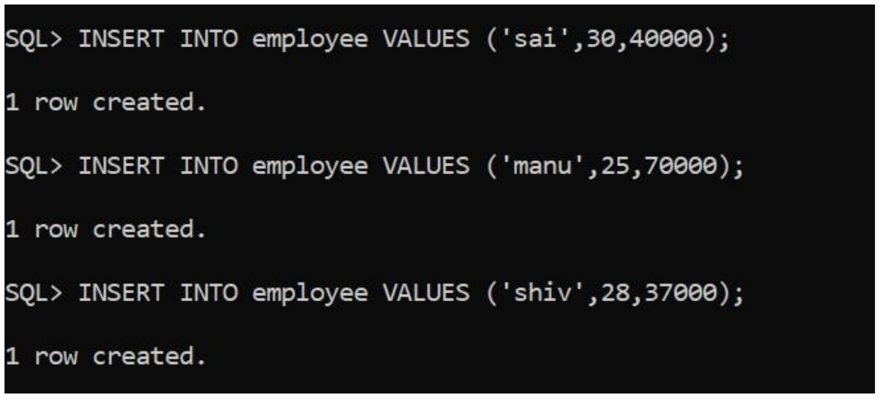
3. Write SQL queries to create VIEWS for various databases (i.e. CREATE VIEW, UPDATE VIEW, ALTER VIEW, and DELETE VIEW).



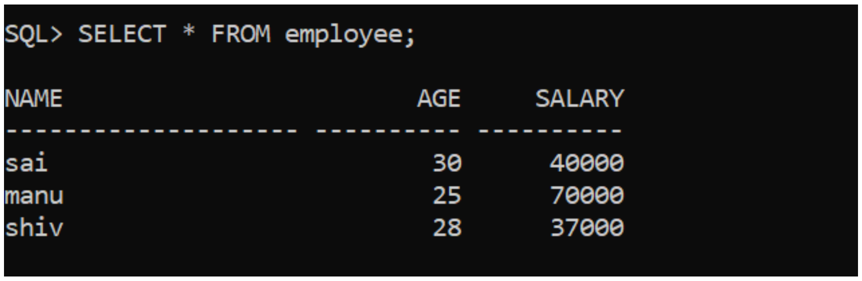
```
C:\Users\HP>sqlplus

SQL*Plus: Release 21.0.0.0.0 - Production on Mon Jan 8 21:07:29 2024
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.

Enter user-name: system
Enter password:
Last Successful login time: Tue Dec 26 2023 22:37:59 -05:00

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL>
```

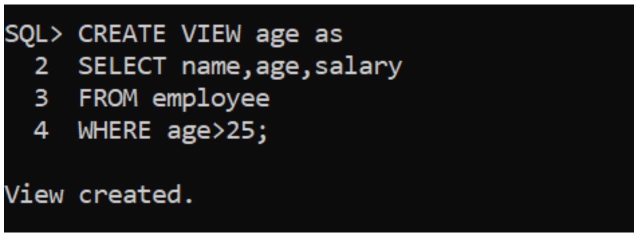


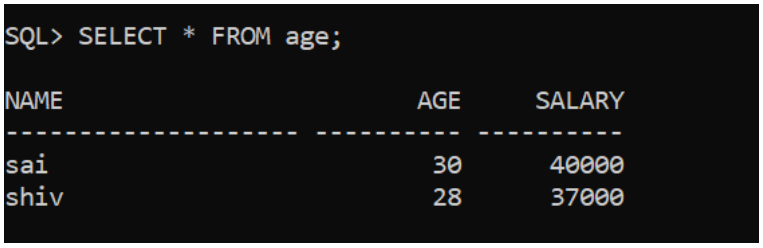
```
SQL> CREATE TABLE employee(
  2  name VARCHAR2(20),
  3  age NUMBER(20),
  4  salary NUMBER(20)
  5  );

Table created.
```

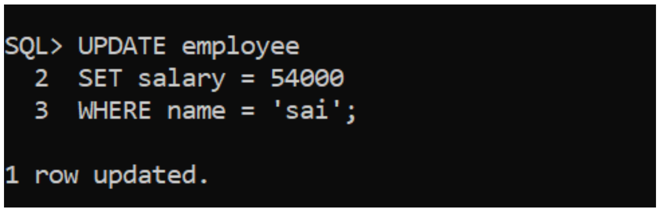


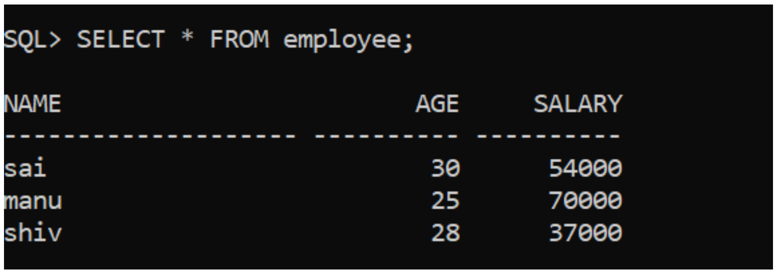
```
SQL> INSERT INTO employee VALUES ('sai',30,40000);

1 row created.

SQL> INSERT INTO employee VALUES ('manu',25,70000);

1 row created.

SQL> INSERT INTO employee VALUES ('shiv',28,37000);

1 row created.
```

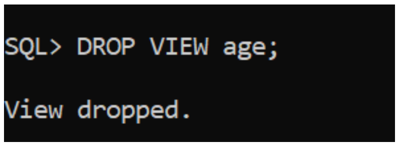

```
SQL> SELECT * FROM employee;

NAME                        AGE     SALARY
-------------------- ---------- ----------
sai                          30      40000
manu                         25      70000
shiv                         28      37000
```


```
SQL> CREATE VIEW age as
  2  SELECT name,age,salary
  3  FROM employee
  4  WHERE age>25;

View created.
```


```
SQL> SELECT * FROM age;

NAME                        AGE     SALARY
-------------------- ---------- ----------
sai                          30      40000
shiv                         28      37000
```

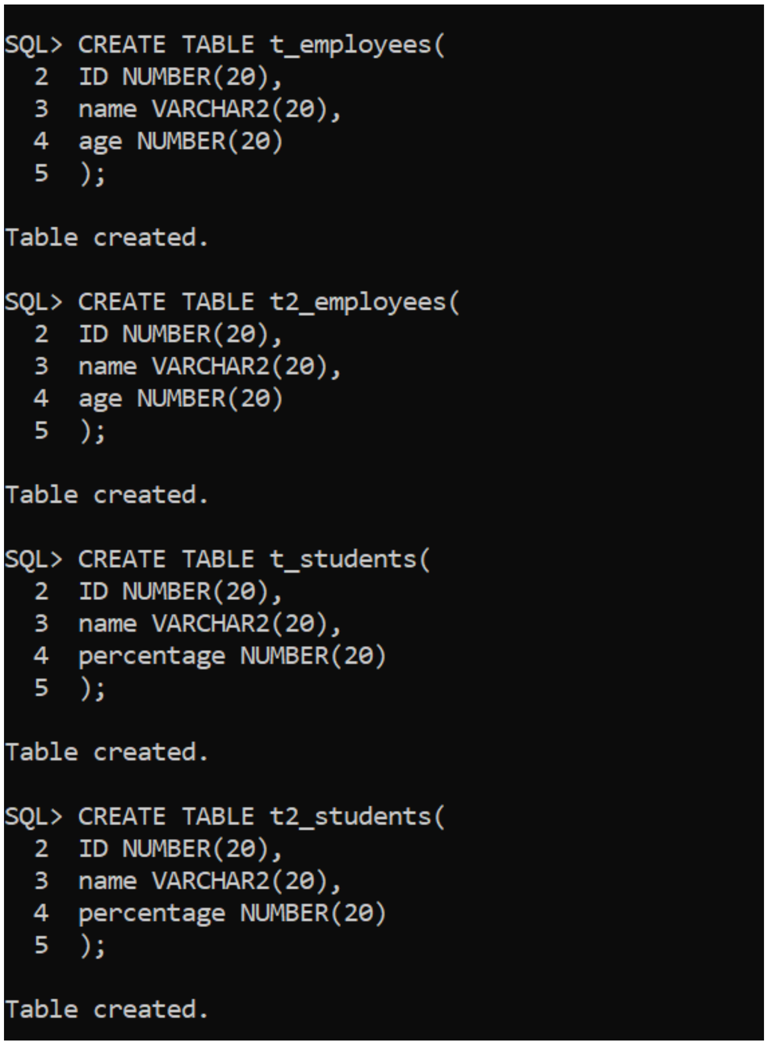

```
SQL> UPDATE employee
  2  SET salary = 54000
  3  WHERE name = 'sai';

1 row updated.
```


```
SQL> SELECT * FROM employee;

NAME                        AGE     SALARY
-------------------- ---------- ----------
sai                          30      54000
manu                         25      70000
shiv                         28      37000
```

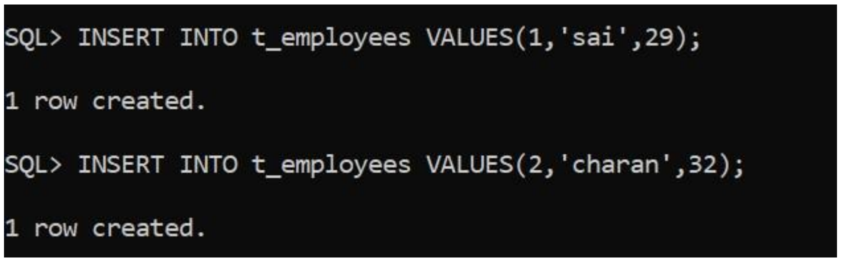

```
SQL> DROP VIEW age;

View dropped.
```

## Experiment 4

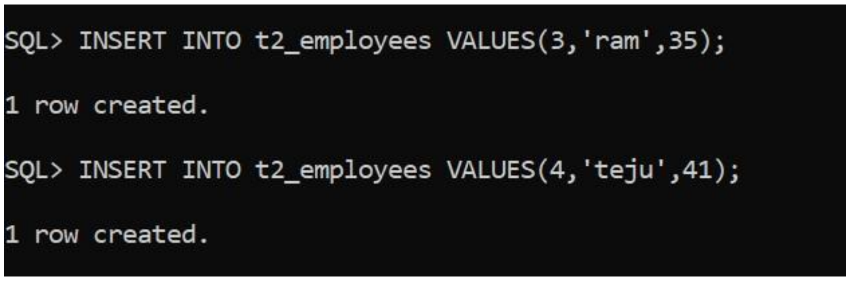
4. Write SQL queries to perform RELATIONAL SET OPERATIONS (i.e. UNION, UNION ALL, INTERSECT, MINUS, CROSS JOIN, NATURAL JOIN).



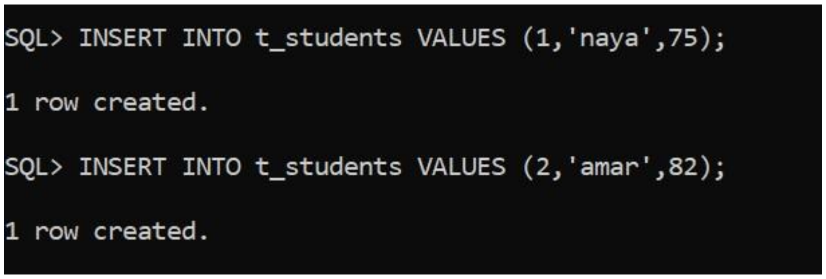
```
SQL> CREATE TABLE t_employees(
  2  ID NUMBER(20),
  3  name VARCHAR2(20),
  4  age NUMBER(20)
  5  );

Table created.

SQL> CREATE TABLE t2_employees(
  2  ID NUMBER(20),
  3  name VARCHAR2(20),
  4  age NUMBER(20)
  5  );

Table created.

SQL> CREATE TABLE t_students(
  2  ID NUMBER(20),
  3  name VARCHAR2(20),
  4  percentage NUMBER(20)
  5  );

Table created.

SQL> CREATE TABLE t2_students(
  2  ID NUMBER(20),
  3  name VARCHAR2(20),
  4  percentage NUMBER(20)
  5  );

Table created.
```



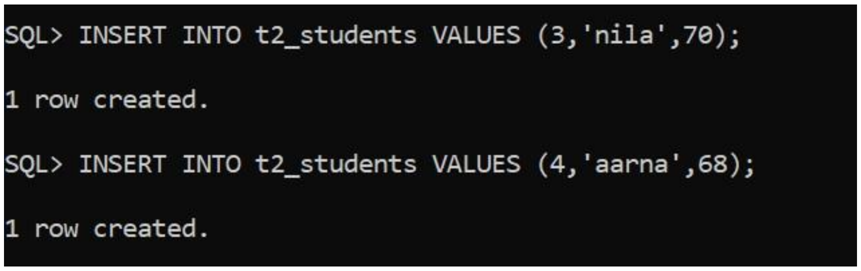
```
SQL> INSERT INTO t_employees VALUES(1,'sai',29);

1 row created.

SQL> INSERT INTO t_employees VALUES(2,'charan',32);

1 row created.
```



```
SQL> INSERT INTO t2_employees VALUES(3,'ram',35);

1 row created.

SQL> INSERT INTO t2_employees VALUES(4,'teju',41);

1 row created.
```

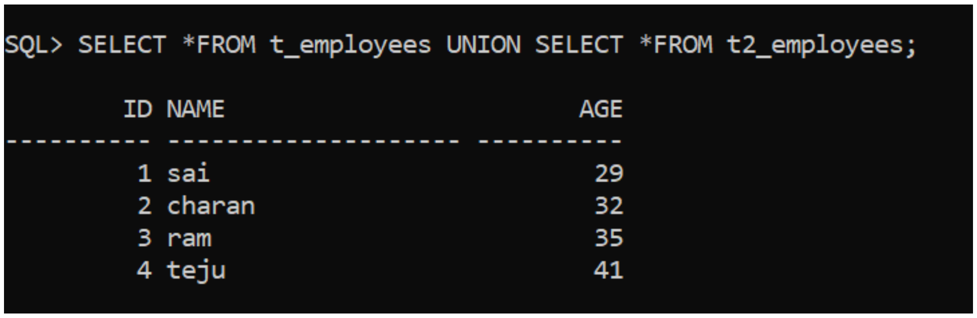


```
SQL> INSERT INTO t_students VALUES (1,'naya',75);

1 row created.

SQL> INSERT INTO t_students VALUES (2,'amar',82);

1 row created.
```

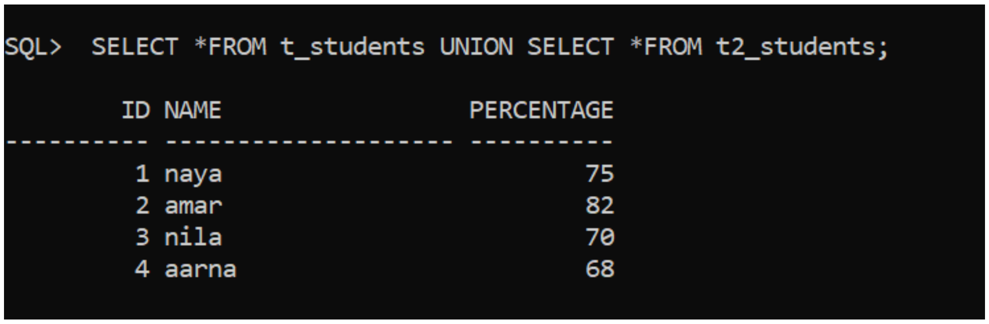


```
SQL> INSERT INTO t2_students VALUES (3,'nila',70);

1 row created.

SQL> INSERT INTO t2_students VALUES (4,'aarna',68);

1 row created.
```

UNION:

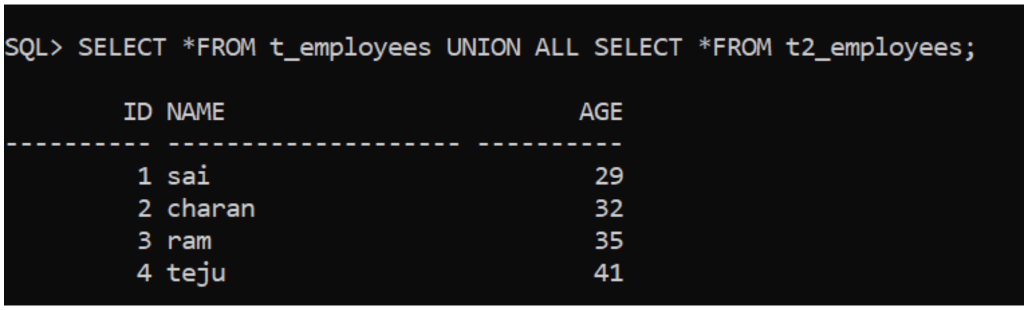


```
SQL> SELECT *FROM t_employees UNION SELECT *FROM t2_employees;

        ID NAME                        AGE
---------- -------------------- ----------
         1 sai                          29
         2 charan                       32
         3 ram                          35
         4 teju                         41
```

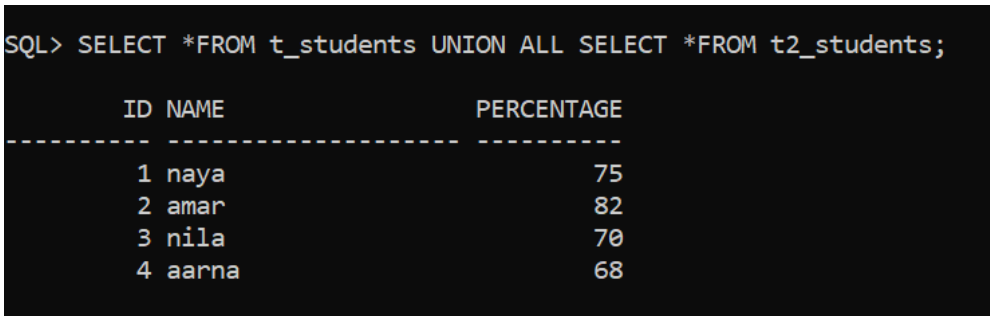


```
SQL>  SELECT *FROM t_students UNION SELECT *FROM t2_students;

        ID NAME                 PERCENTAGE
---------- -------------------- ----------
         1 naya                         75
         2 amar                         82
         3 nila                         70
         4 aarna                        68
```

UNION ALL:

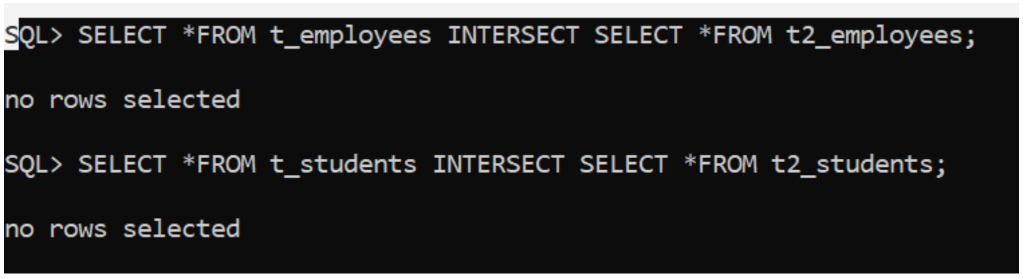


```
SQL> SELECT *FROM t_employees UNION ALL SELECT *FROM t2_employees;

        ID NAME                        AGE
---------- -------------------- ----------
         1 sai                          29
         2 charan                       32
         3 ram                          35
         4 teju                         41
```



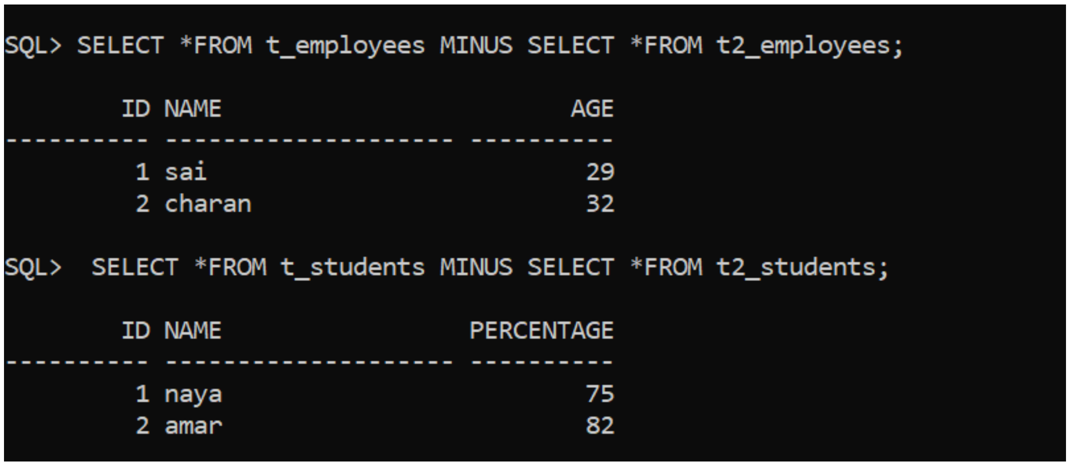
```
SQL> SELECT *FROM t_students UNION ALL SELECT *FROM t2_students;

        ID NAME                 PERCENTAGE
---------- -------------------- ----------
         1 naya                         75
         2 amar                         82
         3 nila                         70
         4 aarna                        68
```

INTERSECT:



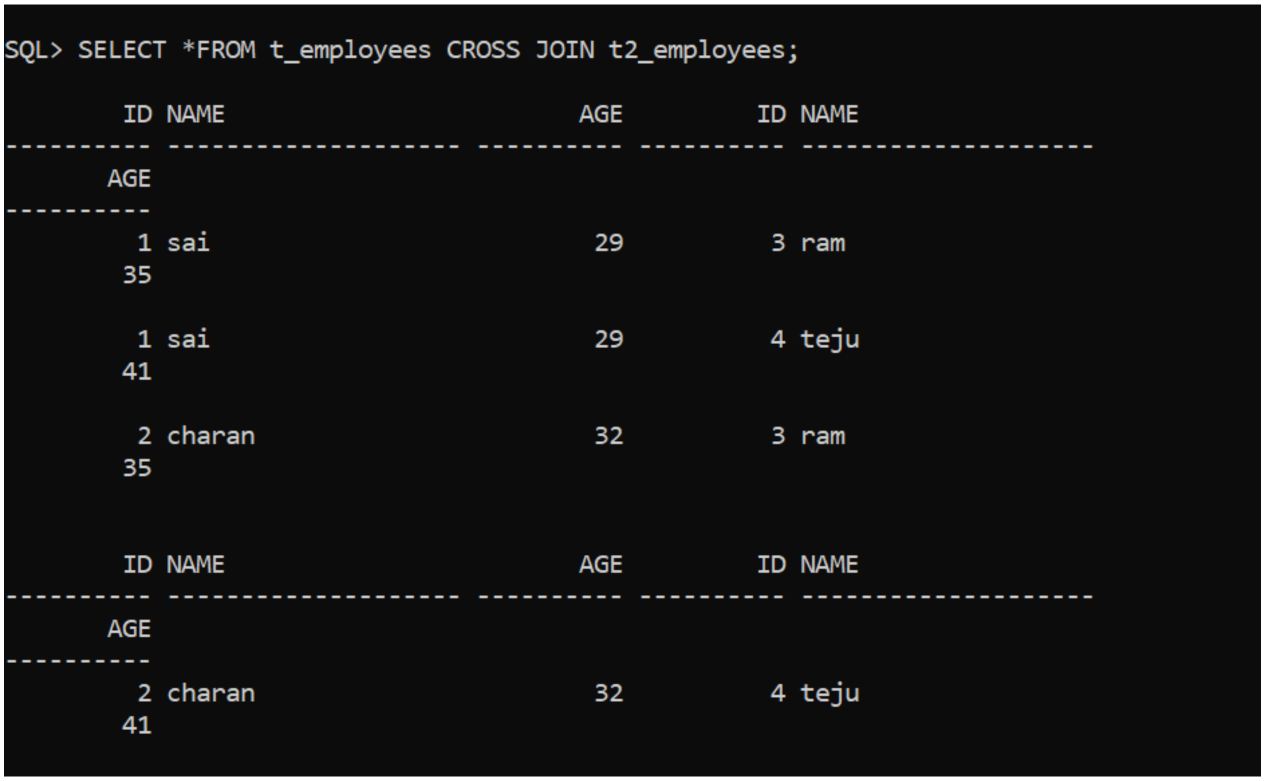
```
SQL> SELECT *FROM t_employees INTERSECT SELECT *FROM t2_employees;

no rows selected

SQL> SELECT *FROM t_students INTERSECT SELECT *FROM t2_students;

no rows selected
```

MINUS:



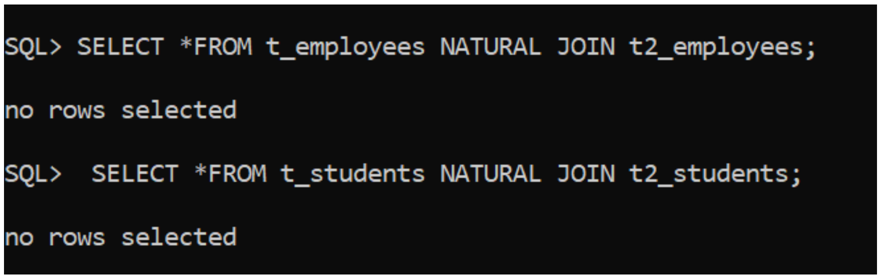
```
SQL> SELECT *FROM t_employees MINUS SELECT *FROM t2_employees;

        ID NAME                        AGE
---------- -------------------- ----------
         1 sai                          29
         2 charan                       32

SQL>  SELECT *FROM t_students MINUS SELECT *FROM t2_students;

        ID NAME                 PERCENTAGE
---------- -------------------- ----------
         1 naya                         75
         2 amar                         82
```

CROSS JOIN:



```
SQL> SELECT *FROM t_employees CROSS JOIN t2_employees;

        ID NAME                        AGE         ID NAME
---------- -------------------- ---------- ---------- --------------------
       AGE
----------
         1 sai                          29          3 ram
        35

         1 sai                          29          4 teju
        41

         2 charan                       32          3 ram
        35


        ID NAME                        AGE         ID NAME
---------- -------------------- ---------- ---------- --------------------
       AGE
----------
         2 charan                       32          4 teju
        41
```

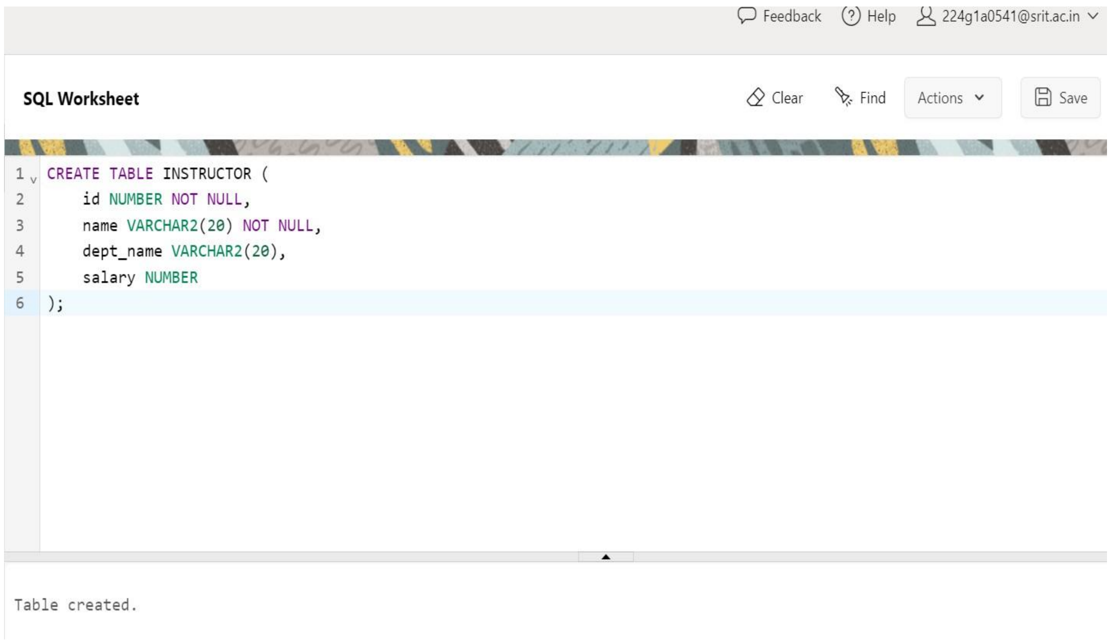
NATURAL JOIN:



```
SQL> SELECT *FROM t_employees NATURAL JOIN t2_employees;

no rows selected

SQL>  SELECT *FROM t_students NATURAL JOIN t2_students;

no rows selected
```

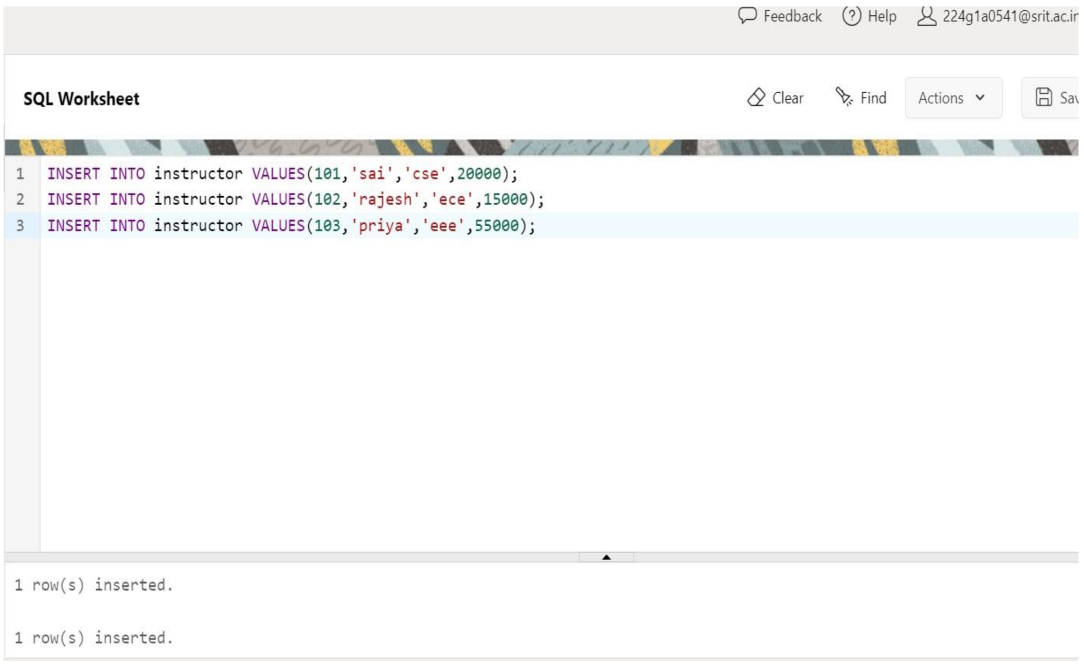
## Experiment 5

5. Write SQL queries to perform SPECIAL OPERATIONS (i.e. ISNULL, BETWEEN, LIKE, IN, EXISTS).



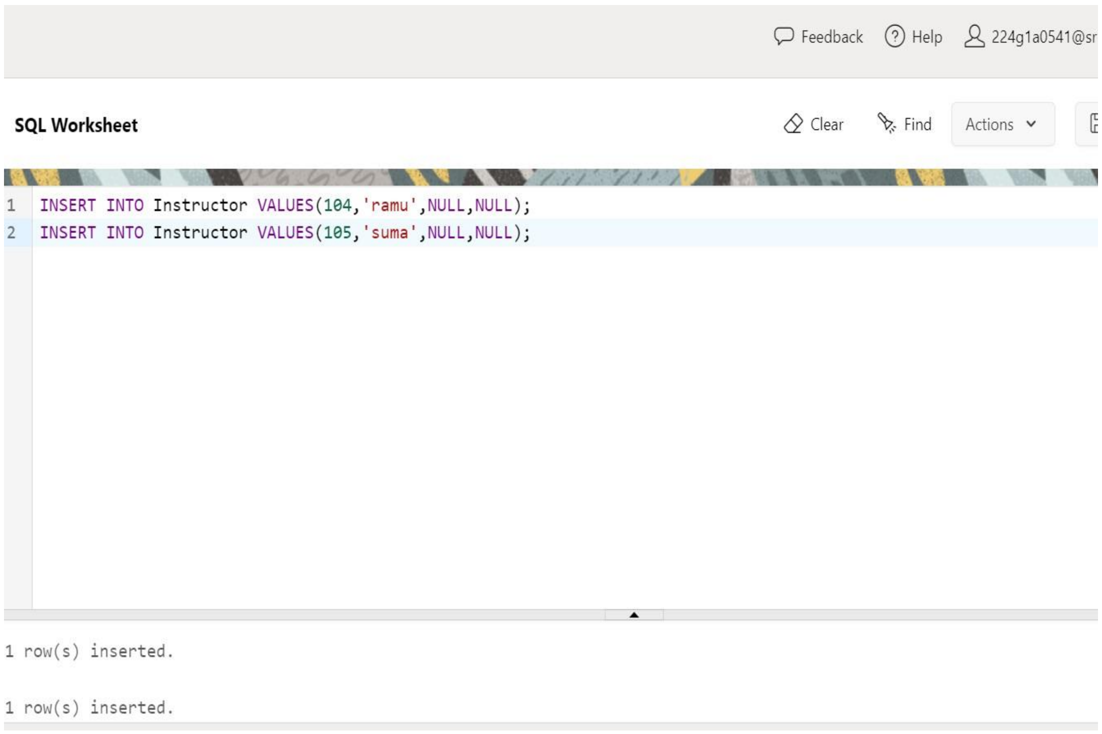
```
CREATE TABLE INSTRUCTOR (
    id NUMBER NOT NULL,
    name VARCHAR2(20) NOT NULL,
    dept_name VARCHAR2(20),
    salary NUMBER
);
```

```
Table created.
```



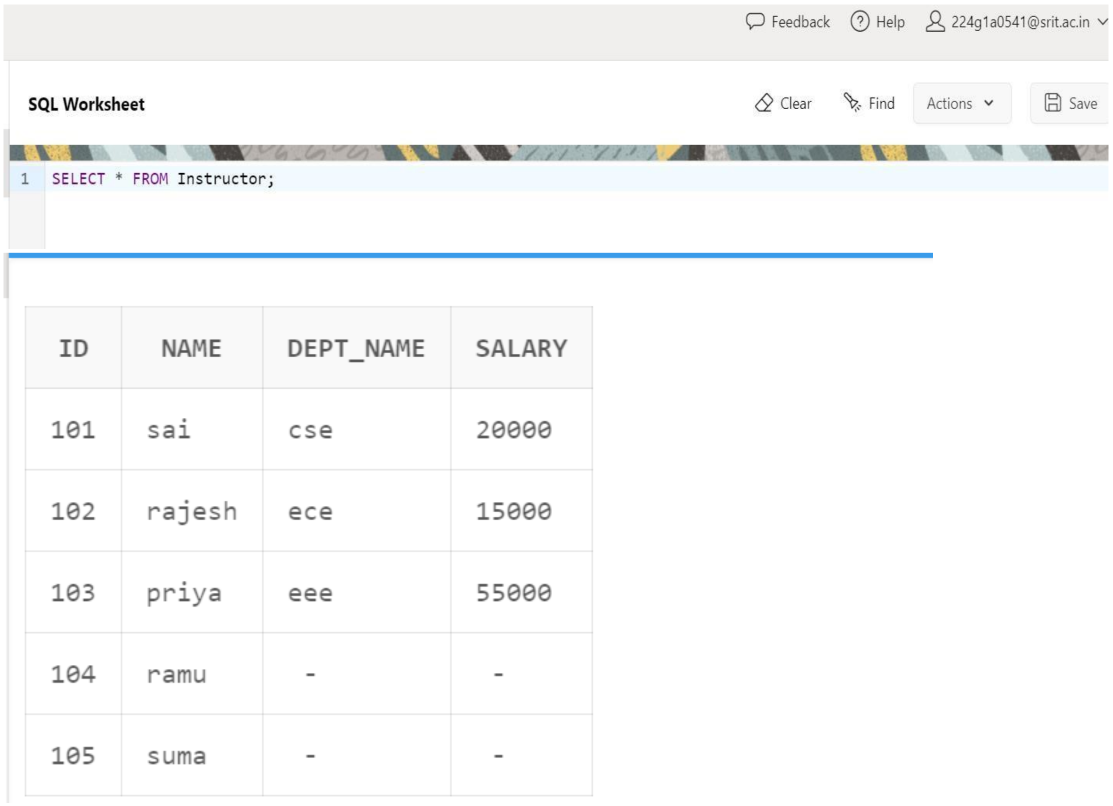
```
INSERT INTO instructor VALUES(101,'sai','cse',20000);
INSERT INTO instructor VALUES(102,'rajesh','ece',15000);
INSERT INTO instructor VALUES(103,'priya','eee',55000);
```

```
1 row(s) inserted.

1 row(s) inserted.
```



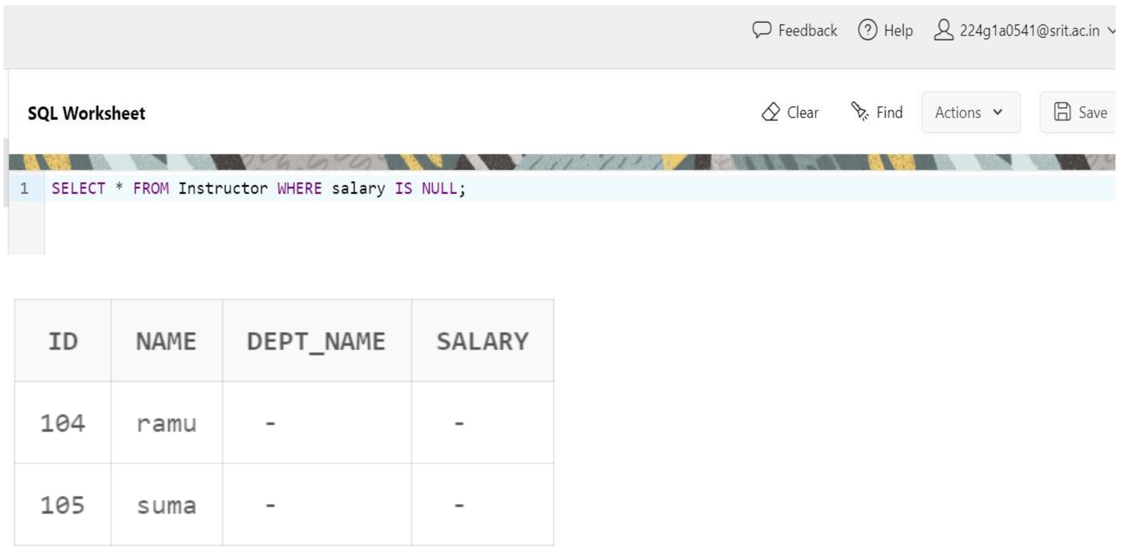
```
INSERT INTO Instructor VALUES(104,'ramu',NULL,NULL);
INSERT INTO Instructor VALUES(105,'suma',NULL,NULL);
```

```
1 row(s) inserted.

1 row(s) inserted.
```



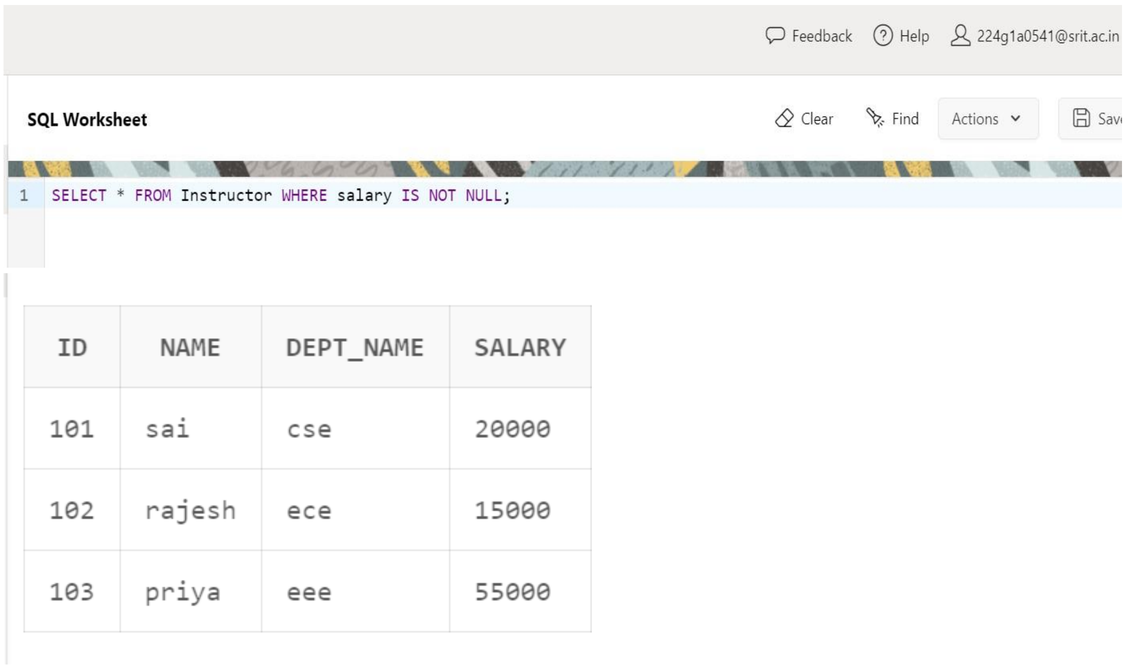
```
SELECT * FROM Instructor;
```

| ID | NAME | DEPT_NAME | SALARY |
|---|---|---|---|
| 101 | sai | cse | 20000 |
| 102 | rajesh | ece | 15000 |
| 103 | priya | eee | 55000 |
| 104 | ramu | - | - |
| 105 | suma | - | - |



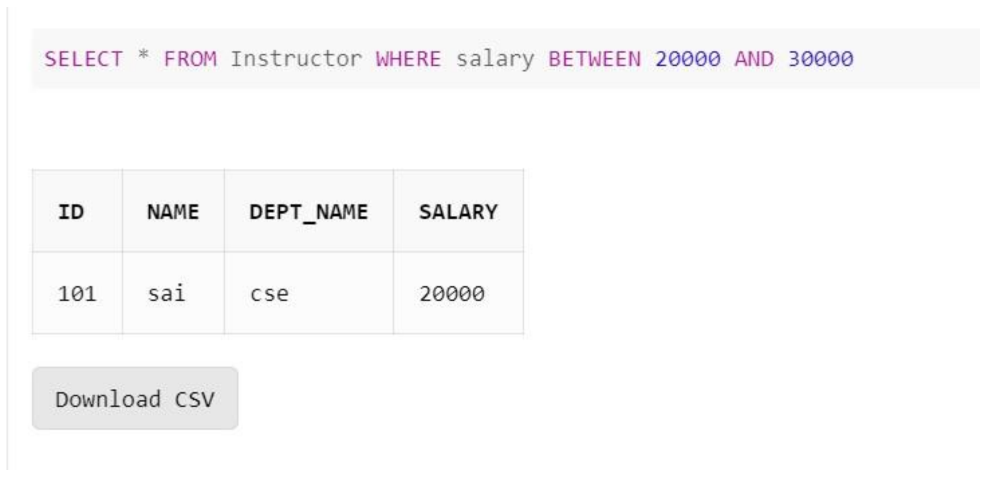
```
SELECT * FROM Instructor WHERE salary IS NULL;
```

| ID | NAME | DEPT_NAME | SALARY |
|---|---|---|---|
| 104 | ramu | - | - |
| 105 | suma | - | - |



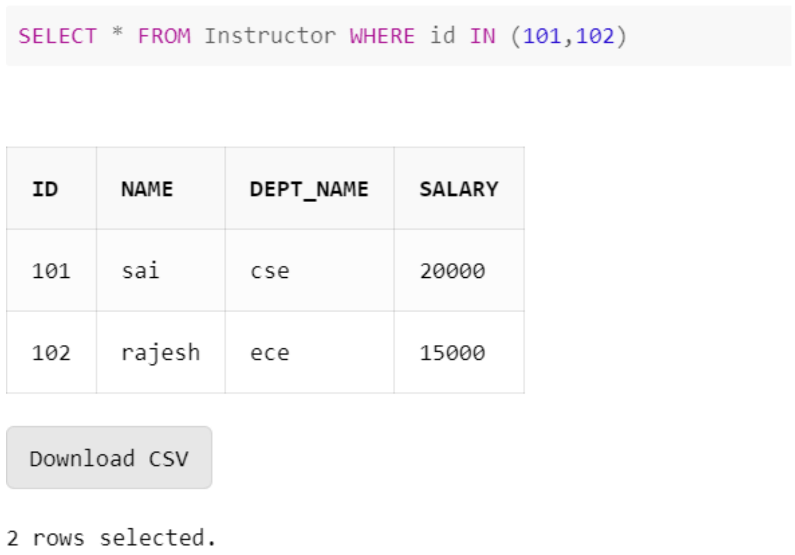
```
SELECT * FROM Instructor WHERE salary IS NOT NULL;
```

| ID | NAME | DEPT_NAME | SALARY |
|---|---|---|---|
| 101 | sai | cse | 20000 |
| 102 | rajesh | ece | 15000 |
| 103 | priya | eee | 55000 |



```
SELECT * FROM Instructor WHERE salary BETWEEN 20000 AND 30000
```

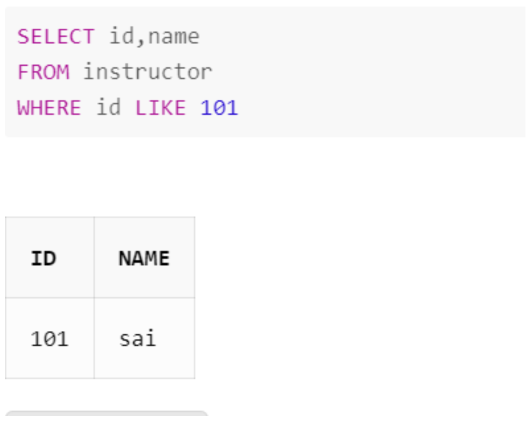
| ID | NAME | DEPT_NAME | SALARY |
|---|---|---|---|
| 101 | sai | cse | 20000 |



```
SELECT * FROM Instructor WHERE id IN (101,102)
```

| ID | NAME | DEPT_NAME | SALARY |
|---|---|---|---|
| 101 | sai | cse | 20000 |
| 102 | rajesh | ece | 15000 |

```
2 rows selected.
```

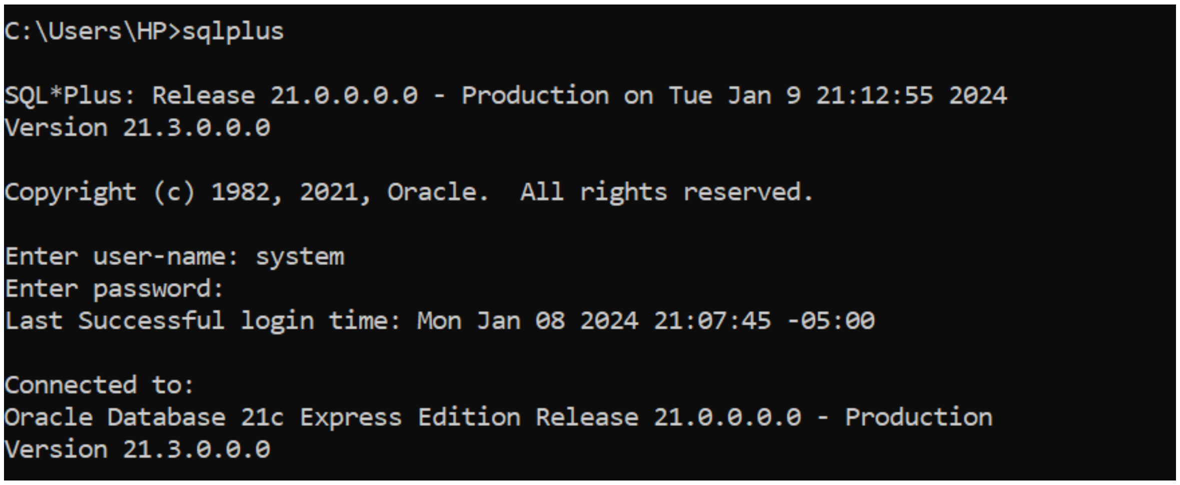


```
SELECT id,name
FROM instructor
WHERE id LIKE 101
```

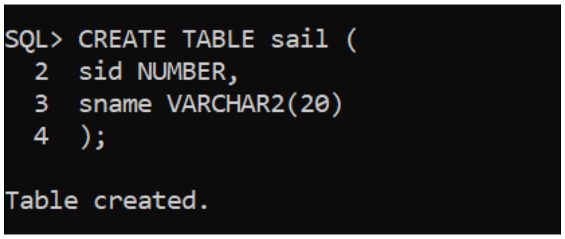
| ID | NAME |
|---|---|
| 101 | sai |

## Experiment 6

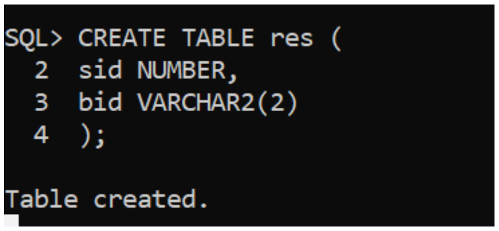
Write SQL queries to perform JOIN OPERATIONS (i.e. CONDITIONAL JOIN, EQUI JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN)



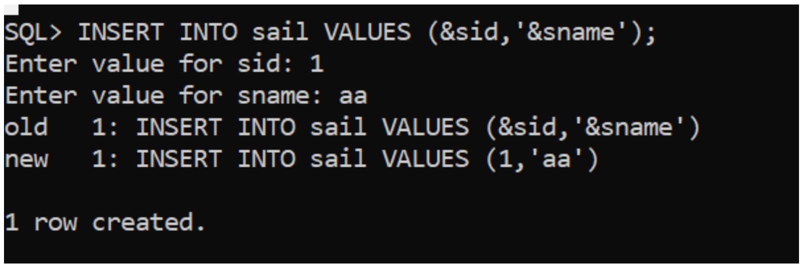
```
C:\Users\HP>sqlplus

SQL*Plus: Release 21.0.0.0.0 - Production on Tue Jan 9 21:12:55 2024
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.

Enter user-name: system
Enter password:
Last Successful login time: Mon Jan 08 2024 21:07:45 -05:00

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
```



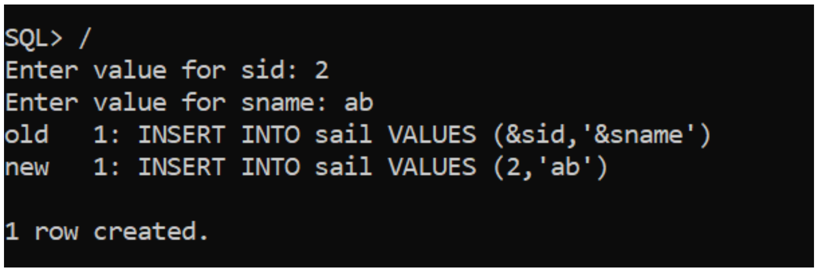
```
SQL> CREATE TABLE sail (
  2  sid NUMBER,
  3  sname VARCHAR2(20)
  4  );

Table created.
```



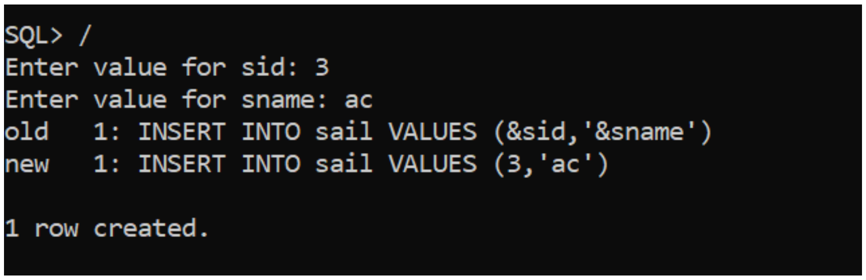
```
SQL> CREATE TABLE res (
  2  sid NUMBER,
  3  bid VARCHAR2(2)
  4  );

Table created.
```



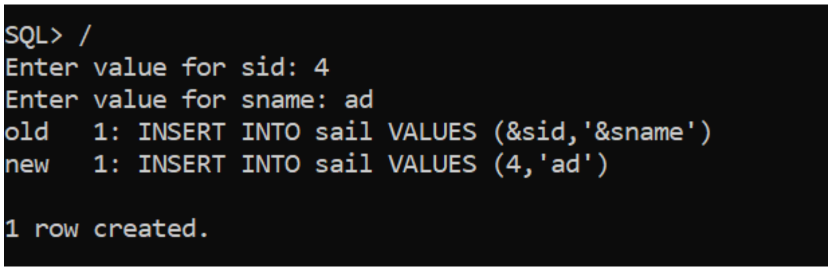
```
SQL> INSERT INTO sail VALUES (&sid,'&sname');
Enter value for sid: 1
Enter value for sname: aa
old   1: INSERT INTO sail VALUES (&sid,'&sname')
new   1: INSERT INTO sail VALUES (1,'aa')

1 row created.
```



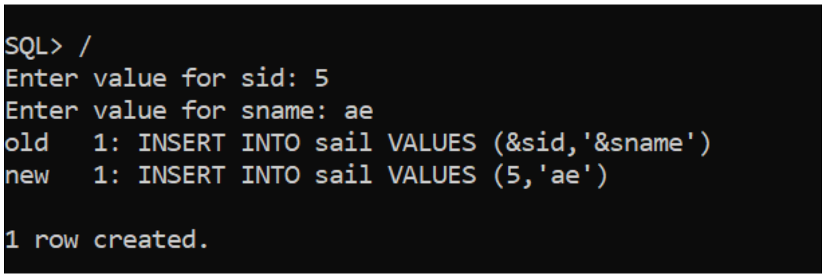
```
SQL> /
Enter value for sid: 2
Enter value for sname: ab
old   1: INSERT INTO sail VALUES (&sid,'&sname')
new   1: INSERT INTO sail VALUES (2,'ab')

1 row created.
```



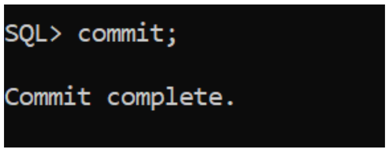
```
SQL> /
Enter value for sid: 3
Enter value for sname: ac
old   1: INSERT INTO sail VALUES (&sid,'&sname')
new   1: INSERT INTO sail VALUES (3,'ac')

1 row created.
```



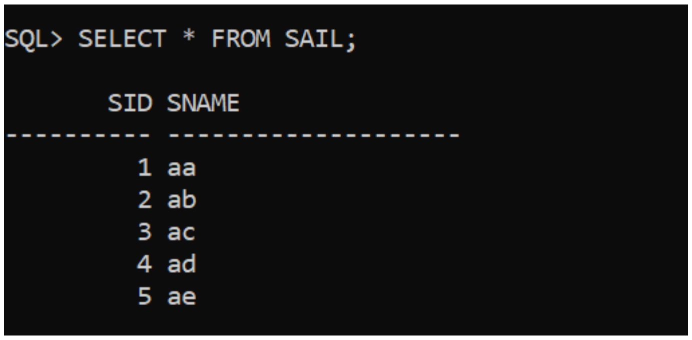
```
SQL> /
Enter value for sid: 4
Enter value for sname: ad
old   1: INSERT INTO sail VALUES (&sid,'&sname')
new   1: INSERT INTO sail VALUES (4,'ad')

1 row created.
```



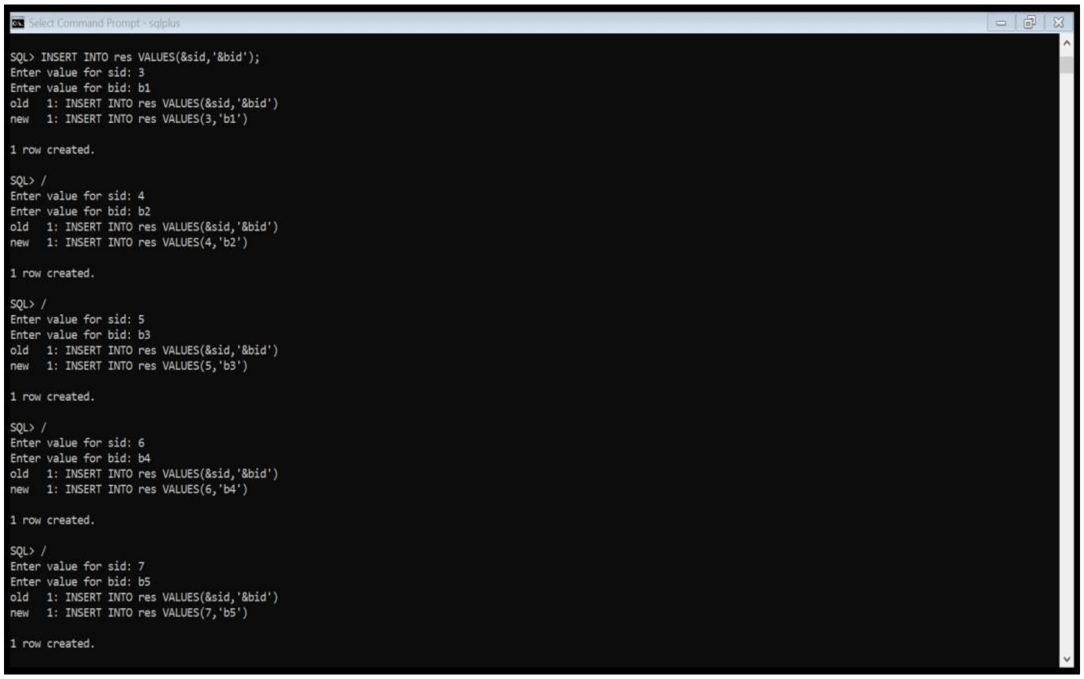
```
SQL> /
Enter value for sid: 5
Enter value for sname: ae
old   1: INSERT INTO sail VALUES (&sid,'&sname')
new   1: INSERT INTO sail VALUES (5,'ae')

1 row created.
```



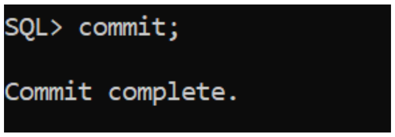
```
SQL> commit;

Commit complete.
```



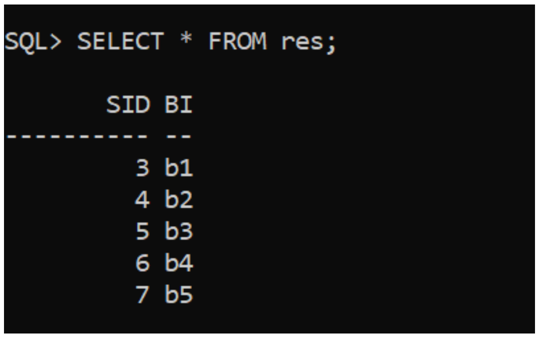
```
SQL> SELECT * FROM SAIL;

       SID SNAME
---------- --------------------
         1 aa
         2 ab
         3 ac
         4 ad
         5 ae
```



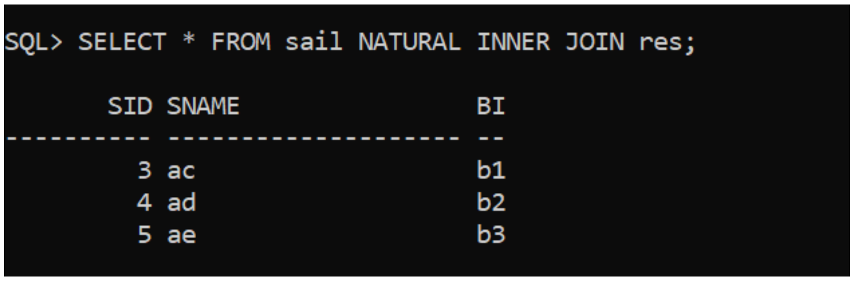
```
SQL> INSERT INTO res VALUES(&sid,'&bid');
Enter value for sid: 3
Enter value for bid: b1
old   1: INSERT INTO res VALUES(&sid,'&bid')
new   1: INSERT INTO res VALUES(3,'b1')

1 row created.

SQL> /
Enter value for sid: 4
Enter value for bid: b2
old   1: INSERT INTO res VALUES(&sid,'&bid')
new   1: INSERT INTO res VALUES(4,'b2')

1 row created.

SQL> /
Enter value for sid: 5
Enter value for bid: b3
old   1: INSERT INTO res VALUES(&sid,'&bid')
new   1: INSERT INTO res VALUES(5,'b3')

1 row created.

SQL> /
Enter value for sid: 6
Enter value for bid: b4
old   1: INSERT INTO res VALUES(&sid,'&bid')
new   1: INSERT INTO res VALUES(6,'b4')

1 row created.

SQL> /
Enter value for sid: 7
Enter value for bid: b5
old   1: INSERT INTO res VALUES(&sid,'&bid')
new   1: INSERT INTO res VALUES(7,'b5')

1 row created.
```



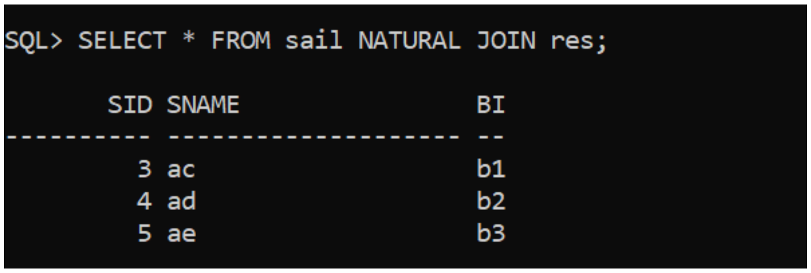
```
SQL> commit;

Commit complete.
```



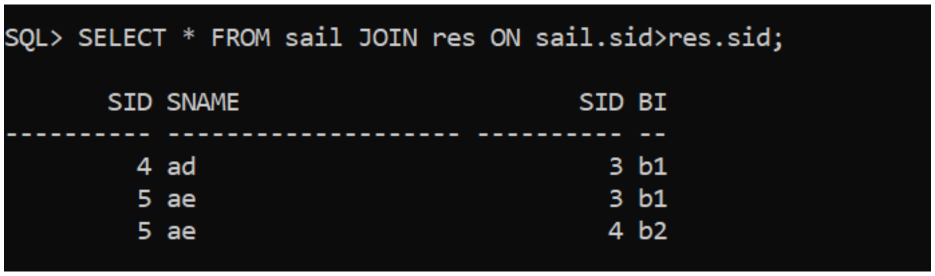
```
SQL> SELECT * FROM res;

       SID BI
---------- --
         3 b1
         4 b2
         5 b3
         6 b4
         7 b5
```



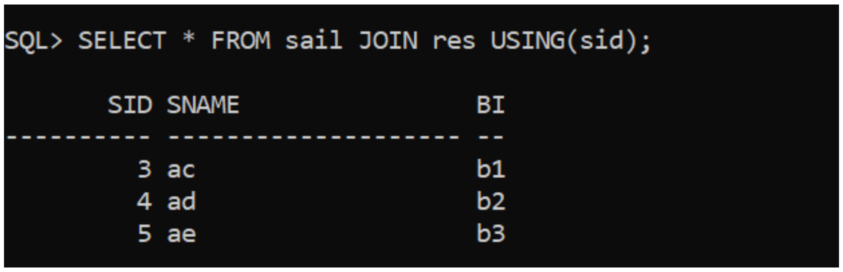
```
SQL> SELECT * FROM sail NATURAL INNER JOIN res;

       SID SNAME                BI
---------- -------------------- --
         3 ac                   b1
         4 ad                   b2
         5 ae                   b3
```



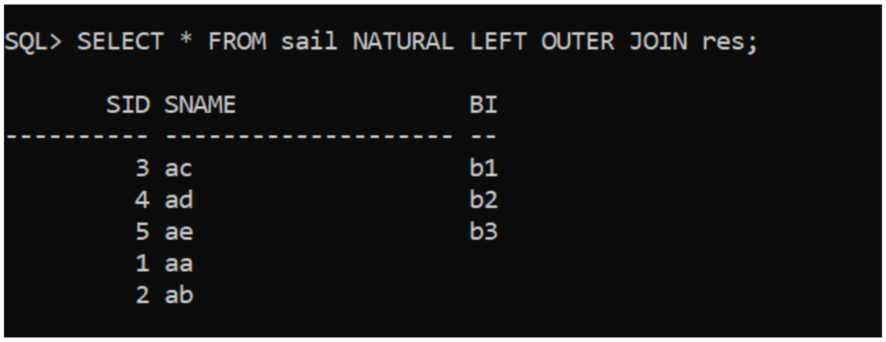
```
SQL> SELECT * FROM sail NATURAL JOIN res;

       SID SNAME                BI
---------- -------------------- --
         3 ac                   b1
         4 ad                   b2
         5 ae                   b3
```



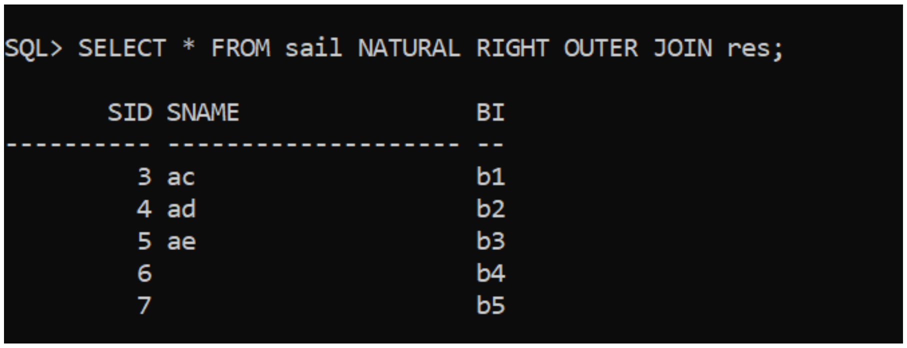
```
SQL> SELECT * FROM sail JOIN res ON sail.sid>res.sid;

       SID SNAME                       SID BI
---------- -------------------- ---------- --
         4 ad                            3 b1
         5 ae                            3 b1
         5 ae                            4 b2
```



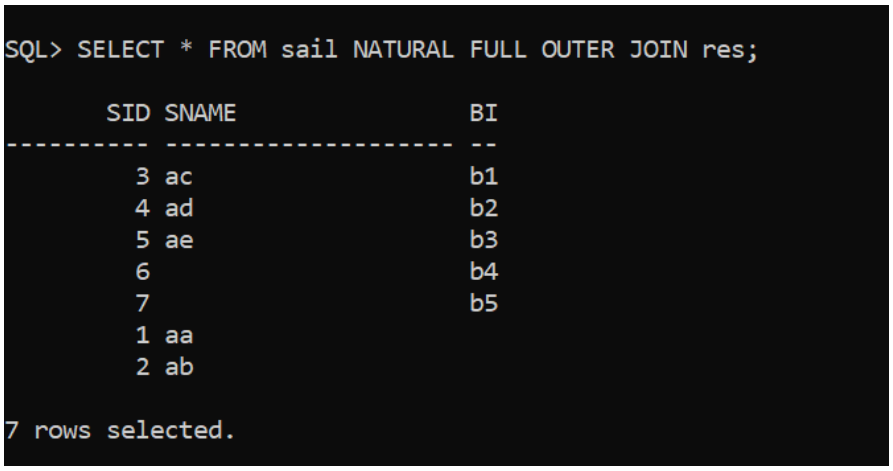
```
SQL> SELECT * FROM sail JOIN res USING(sid);

       SID SNAME                BI
---------- -------------------- --
         3 ac                   b1
         4 ad                   b2
         5 ae                   b3
```



```
SQL> SELECT * FROM sail NATURAL LEFT OUTER JOIN res;

       SID SNAME                BI
---------- -------------------- --
         3 ac                   b1
         4 ad                   b2
         5 ae                   b3
         1 aa
         2 ab
```



```
SQL> SELECT * FROM sail NATURAL RIGHT OUTER JOIN res;

       SID SNAME                BI
---------- -------------------- --
         3 ac                   b1
         4 ad                   b2
         5 ae                   b3
         6                      b4
         7                      b5
```



```
SQL> SELECT * FROM sail NATURAL FULL OUTER JOIN res;

       SID SNAME                BI
---------- -------------------- --
         3 ac                   b1
         4 ad                   b2
         5 ae                   b3
         6                      b4
         7                      b5
         1 aa
         2 ab

7 rows selected.
```

# EXPERIMENT-7

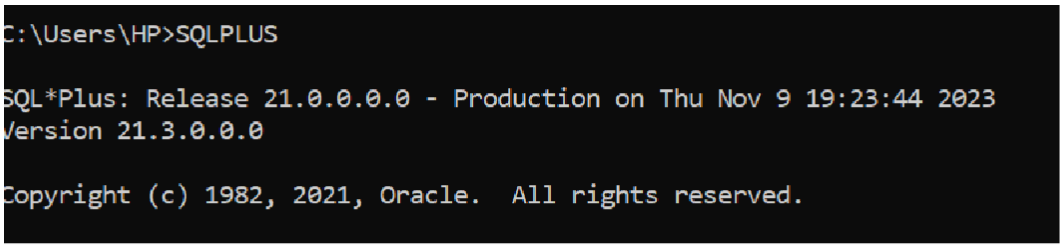
WRITE SQL QUERIES TO PERFORM AGREGATE FUNCTIONS (count, sum, average, Min, max)

## Aim:

To implement SQL QUERIES to perform Aggregate Functions (count, sum, avg, min, and max)

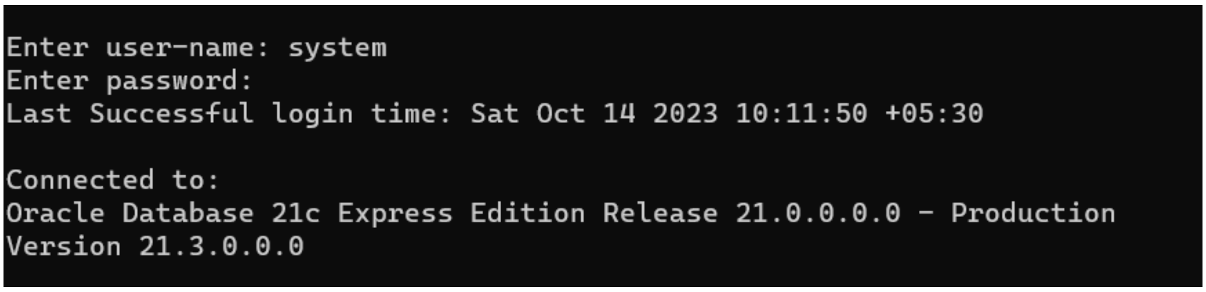
## Procedure:

Open the command prompt and type SQLPLUS



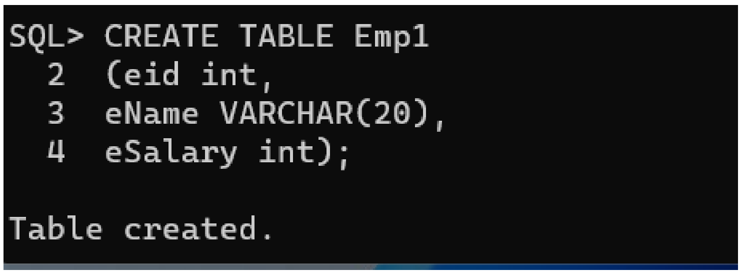
```
:\Users\HP>SQLPLUS

QL*Plus: Release 21.0.0.0.0 - Production on Thu Nov 9 19:23:44 2023
ersion 21.3.0.0.0

opyright (c) 1982, 2021, Oracle.  All rights reserved.
```

Enter user name and password. Login to Oracle database



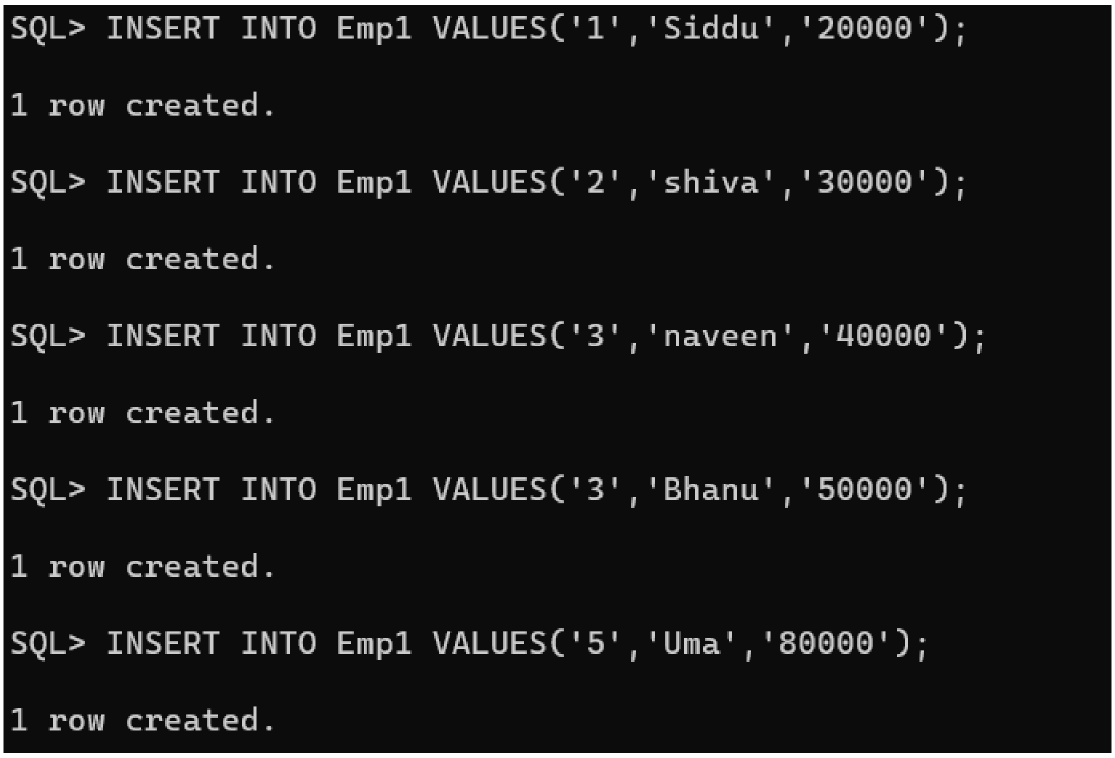
```
Enter user-name: system
Enter password:
Last Successful login time: Sat Oct 14 2023 10:11:50 +05:30

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
```

Create Emp1 table

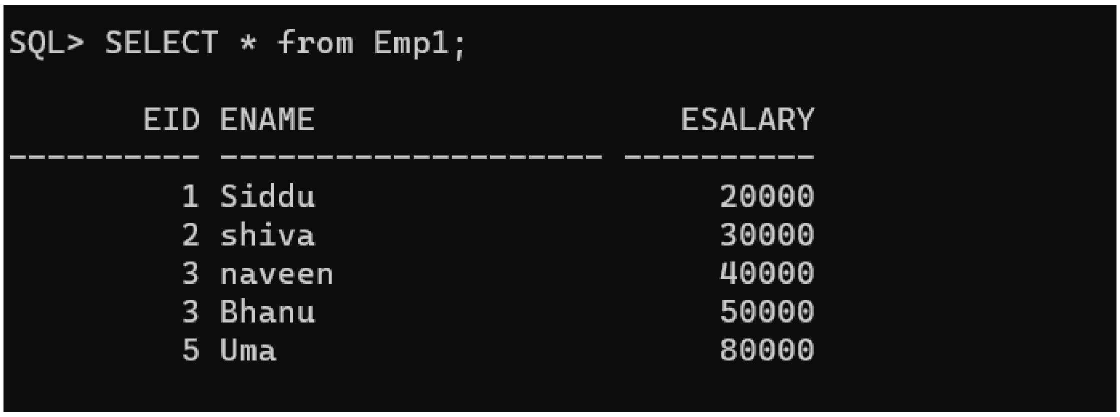


```
SQL> CREATE TABLE Emp1
  2  (eid int,
  3  eName VARCHAR(20),
  4  eSalary int);

Table created.
```

Insert values into Emp1 table

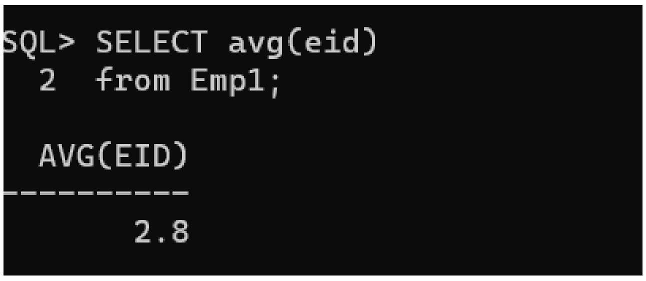


```
SQL> INSERT INTO Emp1 VALUES('1','Siddu','20000');

1 row created.

SQL> INSERT INTO Emp1 VALUES('2','shiva','30000');

1 row created.

SQL> INSERT INTO Emp1 VALUES('3','naveen','40000');

1 row created.

SQL> INSERT INTO Emp1 VALUES('3','Bhanu','50000');

1 row created.

SQL> INSERT INTO Emp1 VALUES('5','Uma','80000');

1 row created.
```

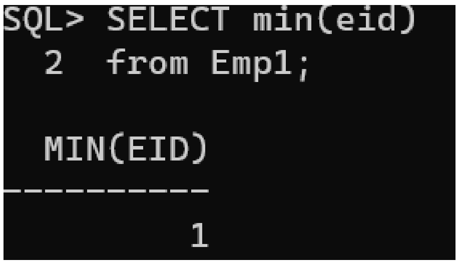


```
SQL> SELECT * from Emp1;

       EID ENAME                   ESALARY
---------- -------------------- ----------
         1 Siddu                     20000
         2 shiva                     30000
         3 naveen                    40000
         3 Bhanu                     50000
         5 Uma                       80000
```

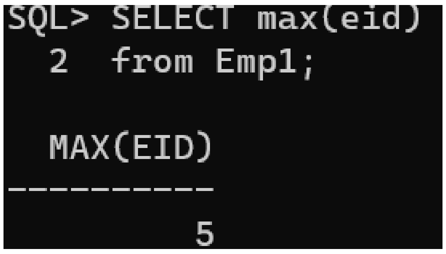
Perform avg, max, min, total, count operations for the table



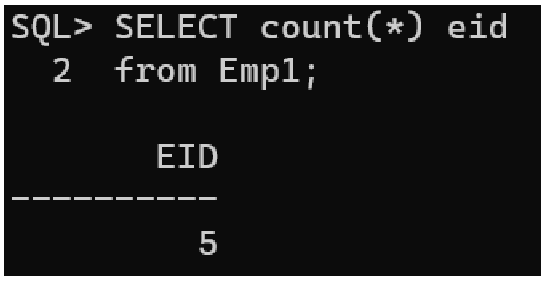
```
SQL> SELECT avg(eid)
  2  from Emp1;

  AVG(EID)
----------
       2.8
```



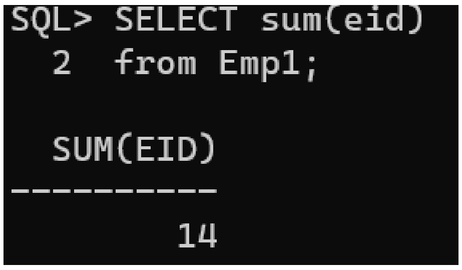
```
SQL> SELECT min(eid)
  2  from Emp1;

  MIN(EID)
----------
         1
```



```
SQL> SELECT max(eid)
  2  from Emp1;

  MAX(EID)
----------
         5
```



```
SQL> SELECT count(*) eid
  2  from Emp1;

       EID
----------
         5
```



```
SQL> SELECT sum(eid)
  2  from Emp1;

  SUM(EID)
----------
        14
```

## Conclusion:

In this lab, we implemented aggregate functions successfully.

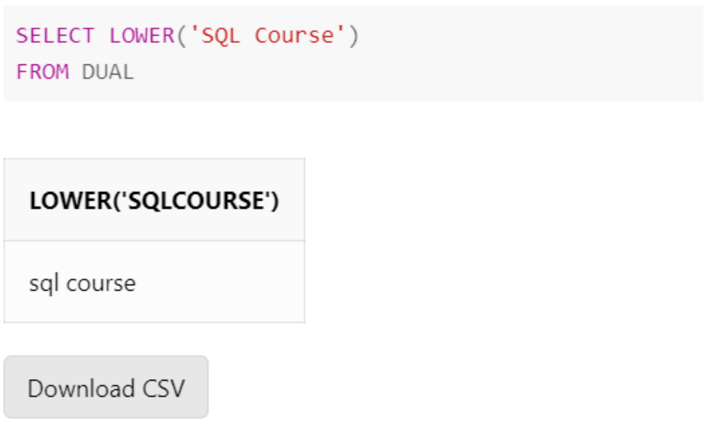
Submitted by

R.LIKHITHA

224G1A0541

## Experiment 8

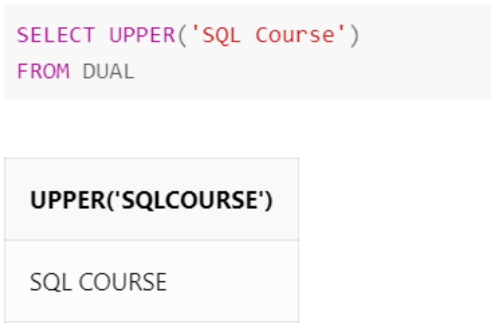
8. Write SQL queries to perform ORACLE BUILT-IN FUNCTIONS (i.e. DATE, TIME).



```
SELECT LOWER('SQL Course')
FROM DUAL
```

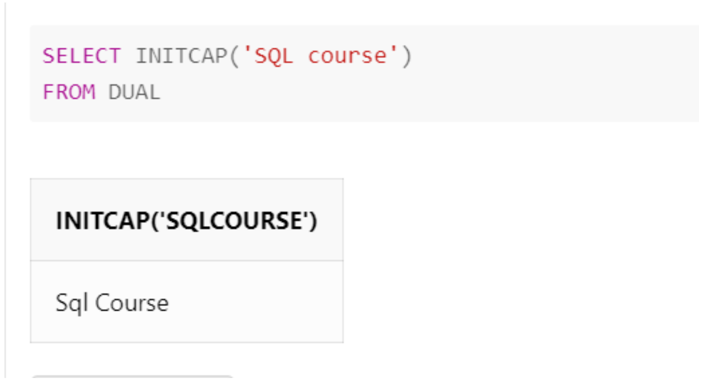
| LOWER('SQLCOURSE') |
|---|
| sql course |

Download CSV



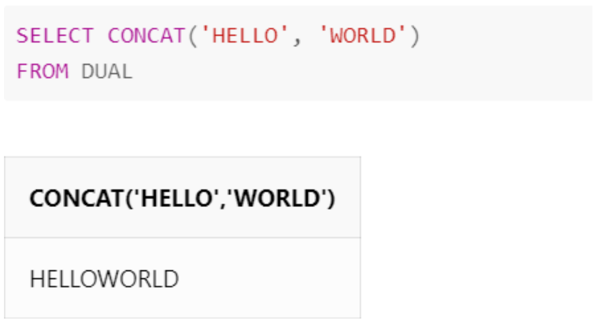
```
SELECT UPPER('SQL Course')
FROM DUAL
```

| UPPER('SQLCOURSE') |
|---|
| SQL COURSE |



```
SELECT INITCAP('SQL course')
FROM DUAL
```

| INITCAP('SQLCOURSE') |
|---|
| Sql Course |



```
SELECT CONCAT('HELLO', 'WORLD')
FROM DUAL
```

| CONCAT('HELLO','WORLD') |
|---|
| HELLOWORLD |

```
SELECT SUBSTR('HELLO WORLD',1,5)
FROM DUAL
```

| SUBSTR('HELLOWORLD',1,5) |
|---|
| HELLO |

```
SELECT LENGTH('HELLO WORLD')
FROM DUAL
```

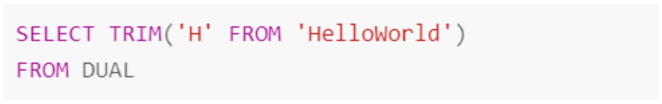
| LENGTH('HELLOWORLD') |
|---|
| 11 |

```
SELECT INSTR('HELLO WORLD', 'WORLD')
FROM DUAL
```

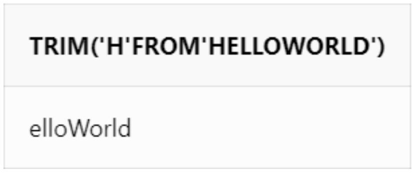
| INSTR('HELLOWORLD','WORLD') |
|---|
| 7 |

```
SELECT REPLACE('JACK and JUE','J','BL')
FROM DUAL
```

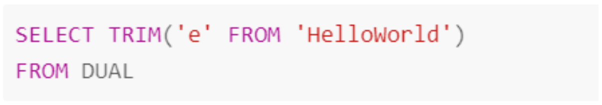
| REPLACE('JACKANDJUE','J','BL') |
|---|
| BLACK and BLUE |



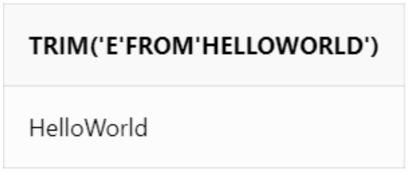
```sql
SELECT TRIM('H' FROM 'HelloWorld')
FROM DUAL
```



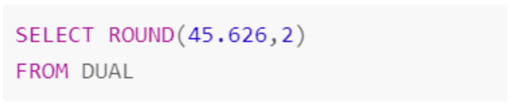
| TRIM('H'FROM'HELLOWORLD') |
|---|
| elloWorld |



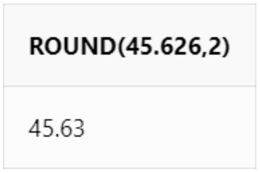
```sql
SELECT TRIM('e' FROM 'HelloWorld')
FROM DUAL
```



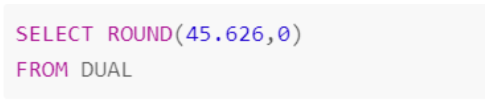
| TRIM('E'FROM'HELLOWORLD') |
|---|
| HelloWorld |



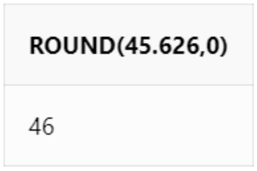
```sql
SELECT ROUND(45.626,2)
FROM DUAL
```



| ROUND(45.626,2) |
|---|
| 45.63 |



```sql
SELECT ROUND(45.626,0)
FROM DUAL
```



| ROUND(45.626,0) |
|---|
| 46 |

```
SELECT ROUND(45.626,-1)
FROM DUAL
```

| ROUND(45.626,-1) |
|---|
| 50 |

```
SELECT ROUND(45.626,-2)
FROM DUAL
```

| ROUND(45.626,-2) |
|---|
| 0 |

```
SELECT TRUNC(45.626, 2)
FROM DUAL
```

| TRUNC(45.626,2) |
|---|
| 45.62 |

```
SELECT TRUNC(45.626, 0)
FROM DUAL
```

| TRUNC(45.626,0) |
|---|
| 45 |

```
SELECT TRUNC(45.626, -1)
FROM DUAL
```

| TRUNC(45.626,-1) |
|---|
| 40 |

```
SELECT TRUNC(45.626, -2)
FROM DUAL
```

| TRUNC(45.626,-2) |
|---|
| 0 |

```
SELECT MOD(1600,300)
FROM DUAL
```

| MOD(1600,300) |
|---|
| 100 |

```
SELECT SYSDATE
FROM DUAL
```

| SYSDATE |
|---|
| 07-DEC-23 |

```
SELECT MONTHS_BETWEEN(SYSDATE,'15-FEB-20')
FROM DUAL
```

| MONTHS_BETWEEN(SYSDATE,'15-FEB-20') |
|---|
| 45.75167189366786140979689366786140979689 |

```
SELECT ADD_MONTHS(SYSDATE, 2)
FROM DUAL
```

| ADD_MONTHS(SYSDATE,2) |
|---|
| 07-FEB-24 |

```
SELECT NEXT_DAY(SYSDATE,'MONDAY')
FROM DUAL
```

| NEXT_DAY(SYSDATE,'MONDAY') |
|---|
| 11-DEC-23 |

```
SELECT LAST_DAY(SYSDATE)
FROM DUAL
```

| LAST_DAY(SYSDATE) |
|---|
| 31-DEC-23 |

## Experiment 9

Write SQL queries to perform KEY CONSTRAINTS (i.e. PRIMARY KEY, FOREIGN KEY, UNIQUE NOT NULL, CHECK, and DEFAULT).

```
CREATE TABLE student (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255) NOT NULL,
Age int
)
```

```
Table created.
```

```
ALTER TABLE student
MODIFY Age int NOT NULL
```

```
Table altered.
```

```
CREATE TABLE Students(
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
CONSTRAINT UC_Person UNIQUE (ID,LastName)
)
```

```
Table created.
```

```
ALTER TABLE students
DROP CONSTRAINT UC_Person
```

```
Table altered.
```

```
ALTER TABLE students
ADD CONSTRAINT UC_Person UNIQUE (ID,LastName)
```

Table altered.

```
CREATE TABLE Persons (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
)
```

Table created.

```
ALTER TABLE Persons
DROP CONSTRAINT PK_Person
```

Table altered.

```
ALTER TABLE Persons
ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
```

Table altered.

```
CREATE TABLE Orders (
OrderID int NOT NULL,
OrderNumber int NOT NULL,
PersonID int,
PRIMARY KEY (OrderID),
CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
REFERENCES Persons(PersonID)
)
```

```
CREATE TABLE Persons (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
City varchar(255),
CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
)
```

Table created.

```
ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18 AND City='Sandnes')
```

Table altered.

```
ALTER TABLE persons
DROP CONSTRAINT chk_personAge
```

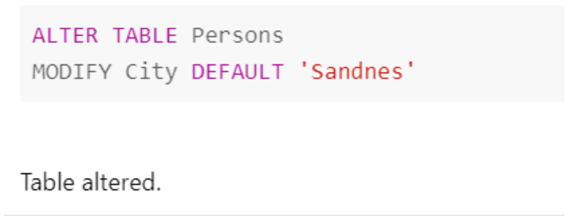
Table altered.

```
DROP TABLE PERSONS
```

Table dropped.

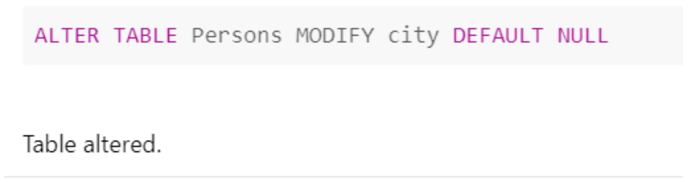
```
CREATE TABLE Persons(
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
City varchar(255) DEFAULT 'Sandnes'
)
```

Table created.



```
ALTER TABLE Persons
MODIFY City DEFAULT 'Sandnes'
```

```
Table altered.
```

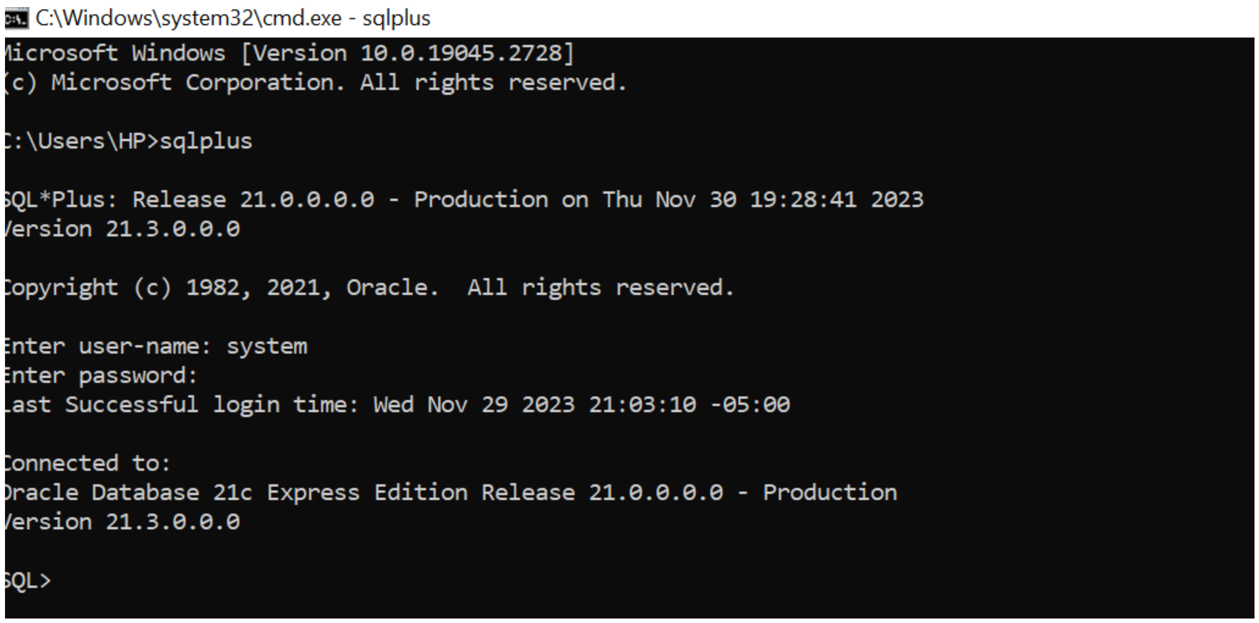


```
ALTER TABLE Persons MODIFY city DEFAULT NULL
```

```
Table altered.
```

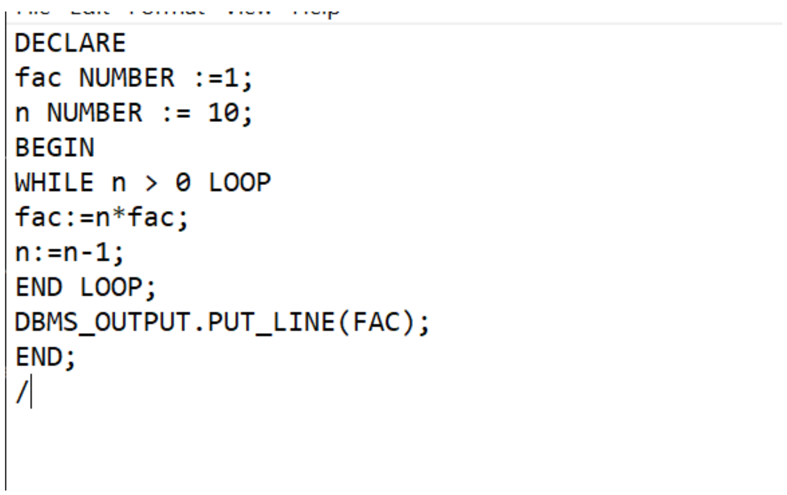
## Experiment -10

Write a PL/ SQL program for calculating the factorial of a given number.

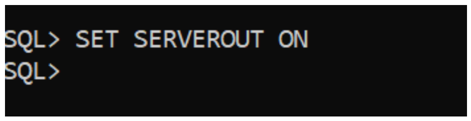


C:\Windows\system32\cmd.exe - sqlplus

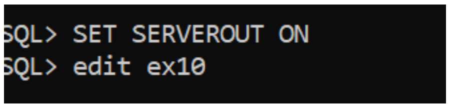
```
icrosoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

:\Users\HP>sqlplus

QL*Plus: Release 21.0.0.0.0 - Production on Thu Nov 30 19:28:41 2023
ersion 21.3.0.0.0

opyright (c) 1982, 2021, Oracle.  All rights reserved.

nter user-name: system
nter password:
ast Successful login time: Wed Nov 29 2023 21:03:10 -05:00

onnected to:
racle Database 21c Express Edition Release 21.0.0.0.0 - Production
ersion 21.3.0.0.0

QL>
```



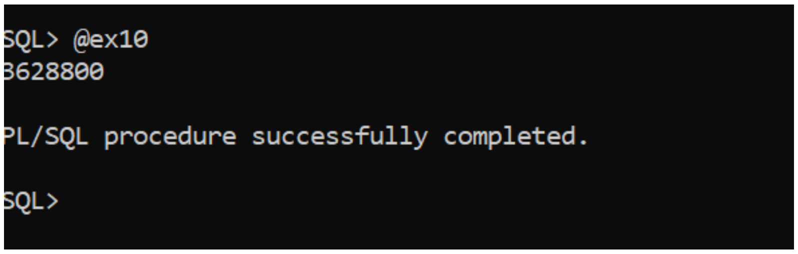
```
DECLARE
fac NUMBER :=1;
n NUMBER := 10;
BEGIN
WHILE n > 0 LOOP
fac:=n*fac;
n:=n-1;
END LOOP;
DBMS_OUTPUT.PUT_LINE(FAC);
END;
/
```



```
SQL> SET SERVEROUT ON
SQL>
```



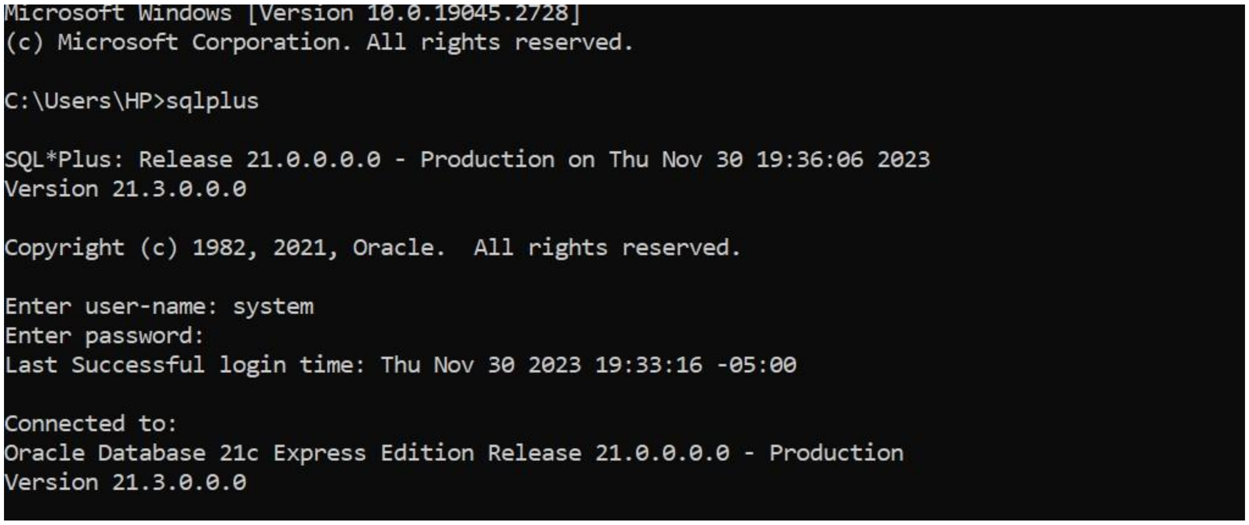
```
SQL> SET SERVEROUT ON
SQL> edit ex10
```



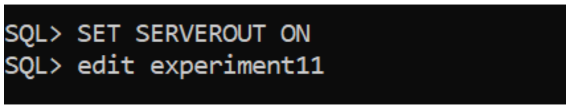
```
SQL> @ex10
3628800

PL/SQL procedure successfully completed.

SQL>
```

## Experiment -11

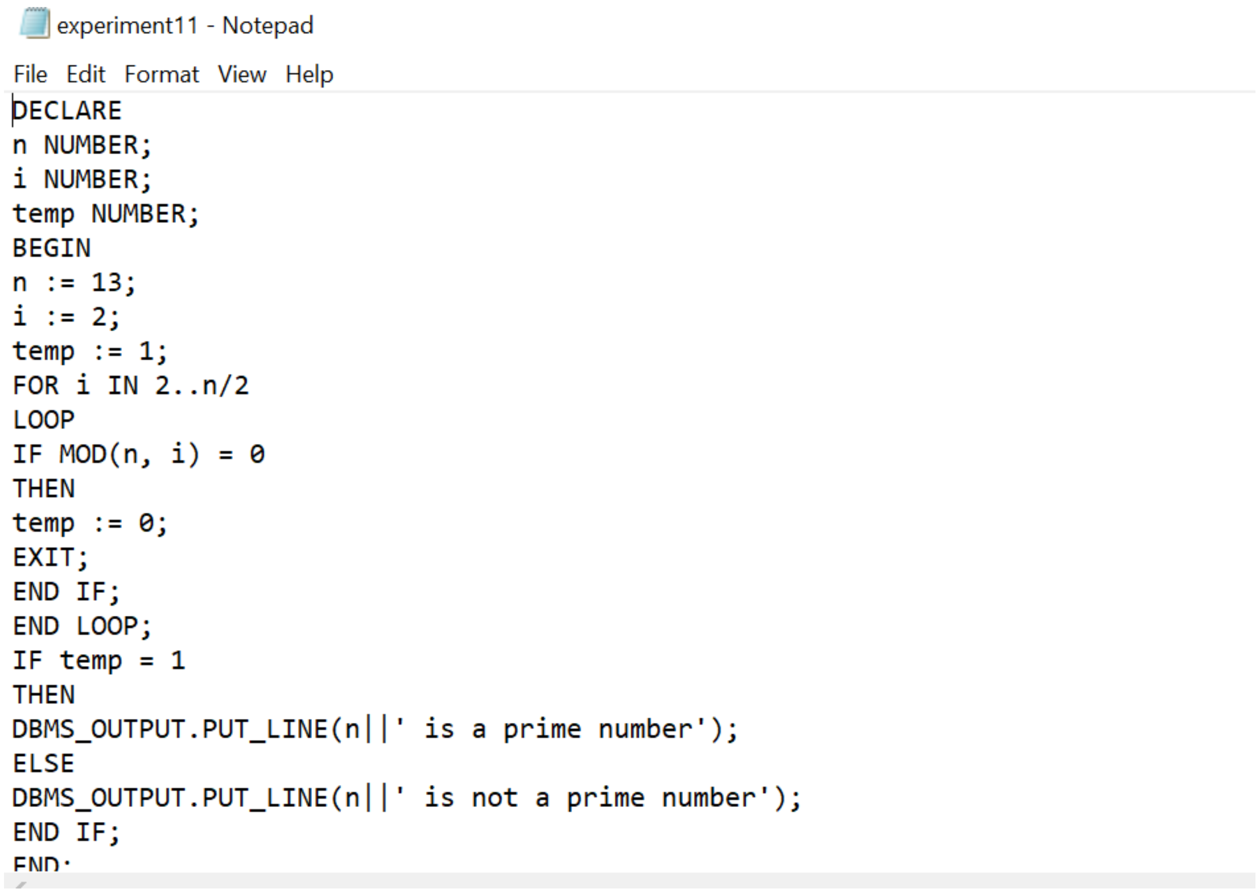
Write a PL/SQL program for finding the given number is prime number or not.



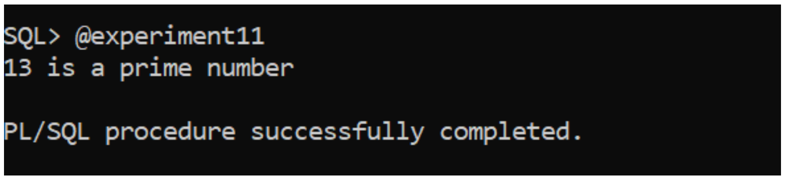
```
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>sqlplus

SQL*Plus: Release 21.0.0.0.0 - Production on Thu Nov 30 19:36:06 2023
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.

Enter user-name: system
Enter password:
Last Successful login time: Thu Nov 30 2023 19:33:16 -05:00

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
```



```
SQL> SET SERVEROUT ON
SQL> edit experiment11
```



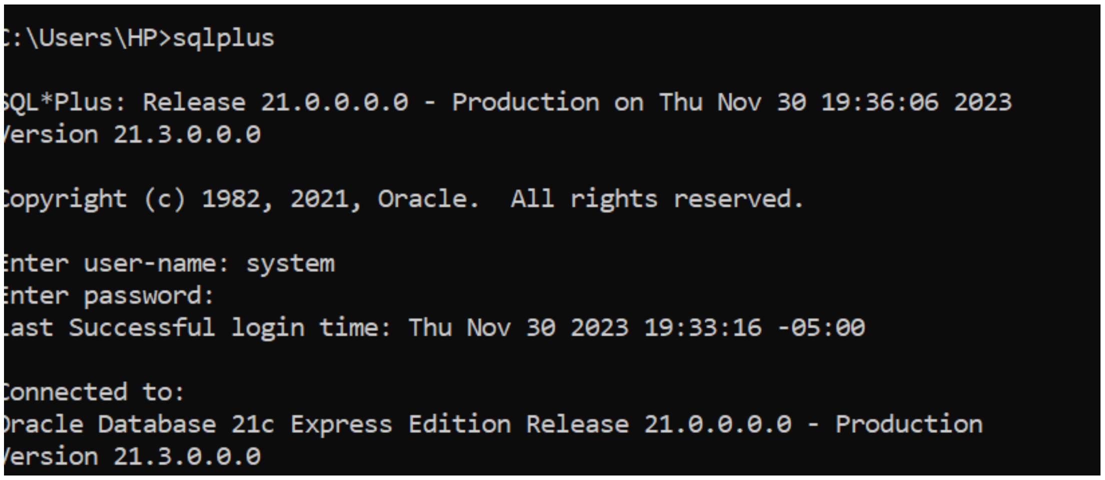
```
DECLARE
n NUMBER;
i NUMBER;
temp NUMBER;
BEGIN
n := 13;
i := 2;
temp := 1;
FOR i IN 2..n/2
LOOP
IF MOD(n, i) = 0
THEN
temp := 0;
EXIT;
END IF;
END LOOP;
IF temp = 1
THEN
DBMS_OUTPUT.PUT_LINE(n||' is a prime number');
ELSE
DBMS_OUTPUT.PUT_LINE(n||' is not a prime number');
END IF;
END;
```



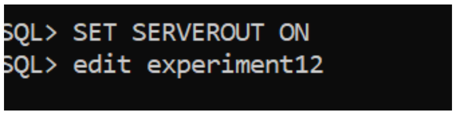
```
SQL> @experiment11
13 is a prime number

PL/SQL procedure successfully completed.
```

## Experiment -12

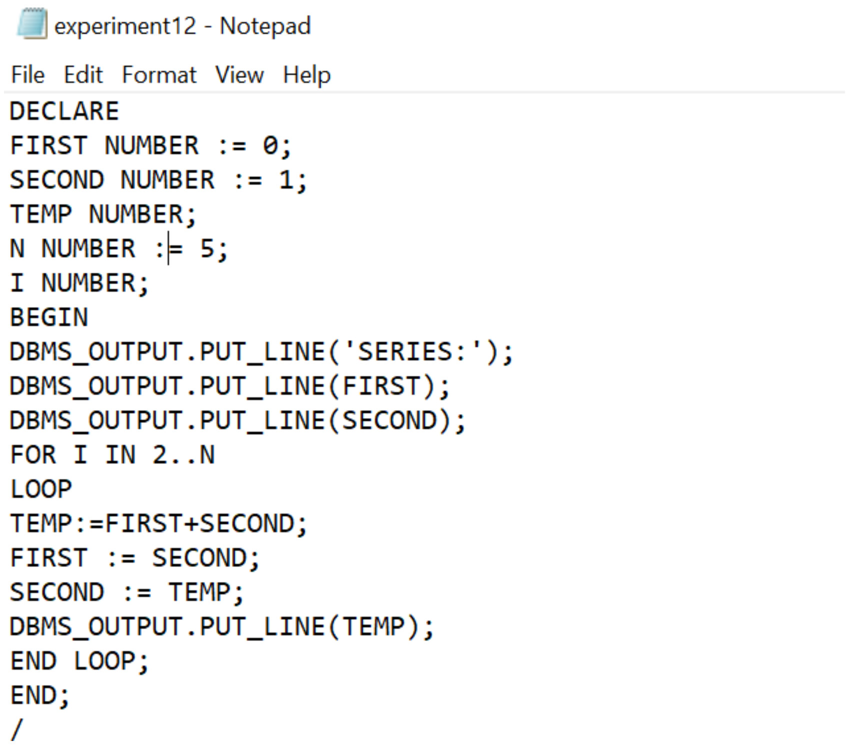
Write a PL/SQL program for displaying the Fibonacci series up to an integer.



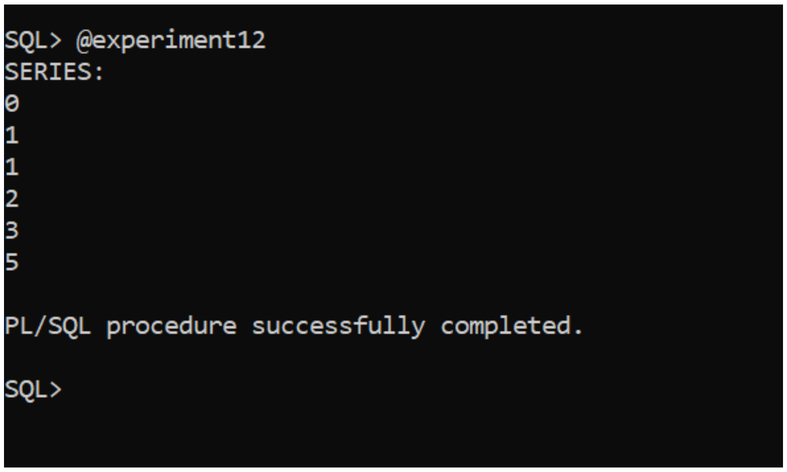
```
:\Users\HP>sqlplus

QL*Plus: Release 21.0.0.0.0 - Production on Thu Nov 30 19:36:06 2023
ersion 21.3.0.0.0

opyright (c) 1982, 2021, Oracle.  All rights reserved.

nter user-name: system
nter password:
ast Successful login time: Thu Nov 30 2023 19:33:16 -05:00

onnected to:
racle Database 21c Express Edition Release 21.0.0.0.0 - Production
ersion 21.3.0.0.0
```



```
QL> SET SERVEROUT ON
QL> edit experiment12
```



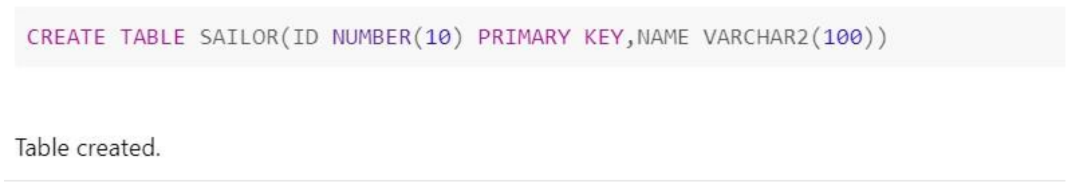
```
DECLARE
FIRST NUMBER := 0;
SECOND NUMBER := 1;
TEMP NUMBER;
N NUMBER := 5;
I NUMBER;
BEGIN
DBMS_OUTPUT.PUT_LINE('SERIES:');
DBMS_OUTPUT.PUT_LINE(FIRST);
DBMS_OUTPUT.PUT_LINE(SECOND);
FOR I IN 2..N
LOOP
TEMP:=FIRST+SECOND;
FIRST := SECOND;
SECOND := TEMP;
DBMS_OUTPUT.PUT_LINE(TEMP);
END LOOP;
END;
/
```



```
SQL> @experiment12
SERIES:
0
1
1
2
3
5

PL/SQL procedure successfully completed.

SQL>
```

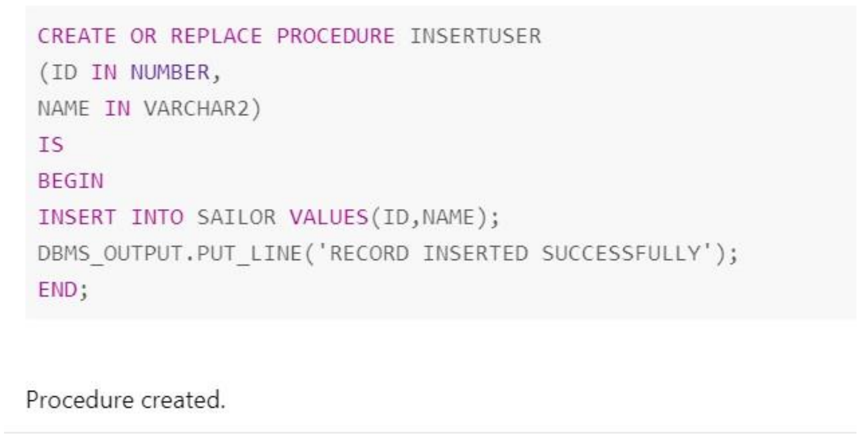
## Experiment -13

13. Write PL/SQL program to implement Stored Procedure on table.



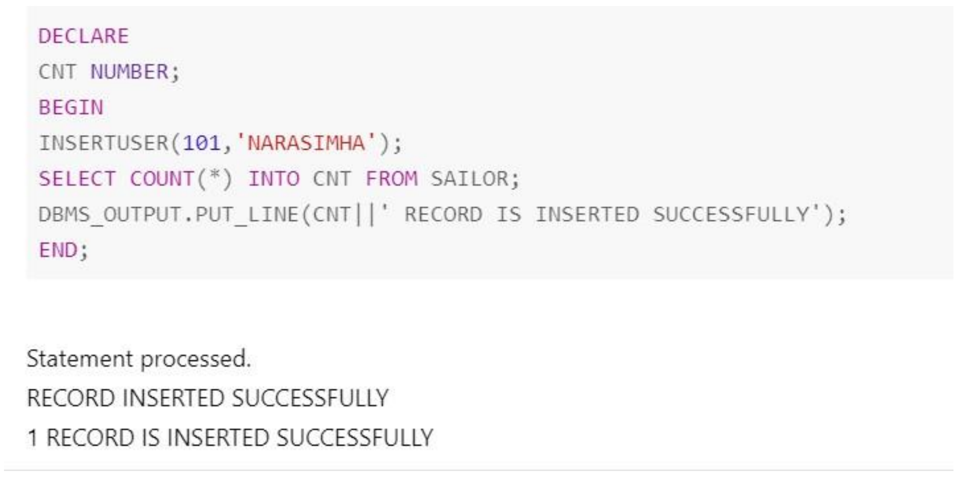
```
CREATE TABLE SAILOR(ID NUMBER(10) PRIMARY KEY,NAME VARCHAR2(100))
```

```
Table created.
```



```
CREATE OR REPLACE PROCEDURE INSERTUSER
(ID IN NUMBER,
NAME IN VARCHAR2)
IS
BEGIN
INSERT INTO SAILOR VALUES(ID,NAME);
DBMS_OUTPUT.PUT_LINE('RECORD INSERTED SUCCESSFULLY');
END;
```

```
Procedure created.
```

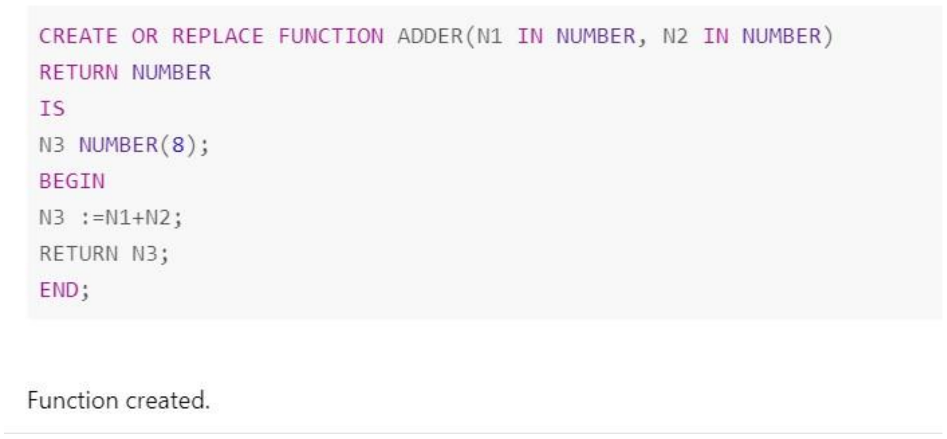


```
DECLARE
CNT NUMBER;
BEGIN
INSERTUSER(101,'NARASIMHA');
SELECT COUNT(*) INTO CNT FROM SAILOR;
DBMS_OUTPUT.PUT_LINE(CNT||' RECORD IS INSERTED SUCCESSFULLY');
END;
```

```
Statement processed.
RECORD INSERTED SUCCESSFULLY
1 RECORD IS INSERTED SUCCESSFULLY
```

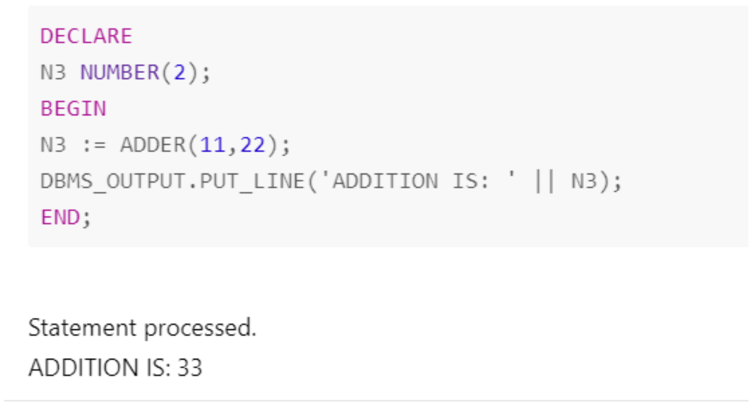
## Experiment – 14

14. Write PL/SQL program to implement Stored Function on table.



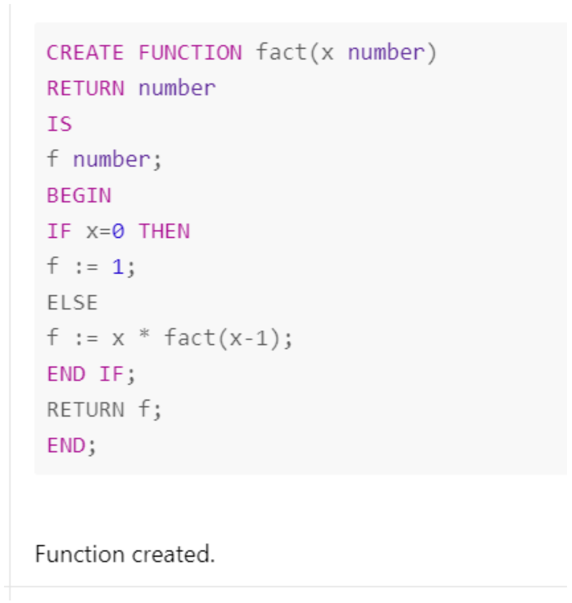
```
CREATE OR REPLACE FUNCTION ADDER(N1 IN NUMBER, N2 IN NUMBER)
RETURN NUMBER
IS
N3 NUMBER(8);
BEGIN
N3 :=N1+N2;
RETURN N3;
END;
```

```
Function created.
```



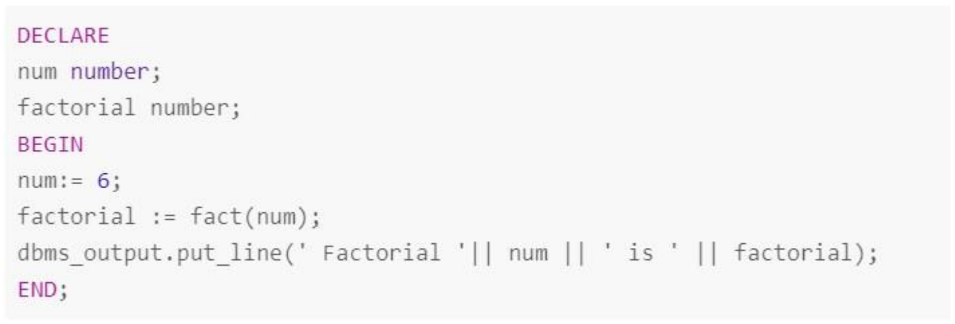
```
DECLARE
N3 NUMBER(2);
BEGIN
N3 := ADDER(11,22);
DBMS_OUTPUT.PUT_LINE('ADDITION IS: ' || N3);
END;
```

```
Statement processed.
ADDITION IS: 33
```

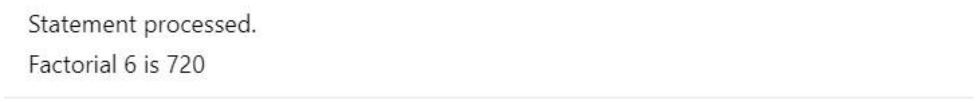


```
CREATE FUNCTION fact(x number)
RETURN number
IS
f number;
BEGIN
IF x=0 THEN
f := 1;
ELSE
f := x * fact(x-1);
END IF;
RETURN f;
END;
```

```
Function created.
```



```
DECLARE
num number;
factorial number;
BEGIN
num:= 6;
factorial := fact(num);
dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
END;
```

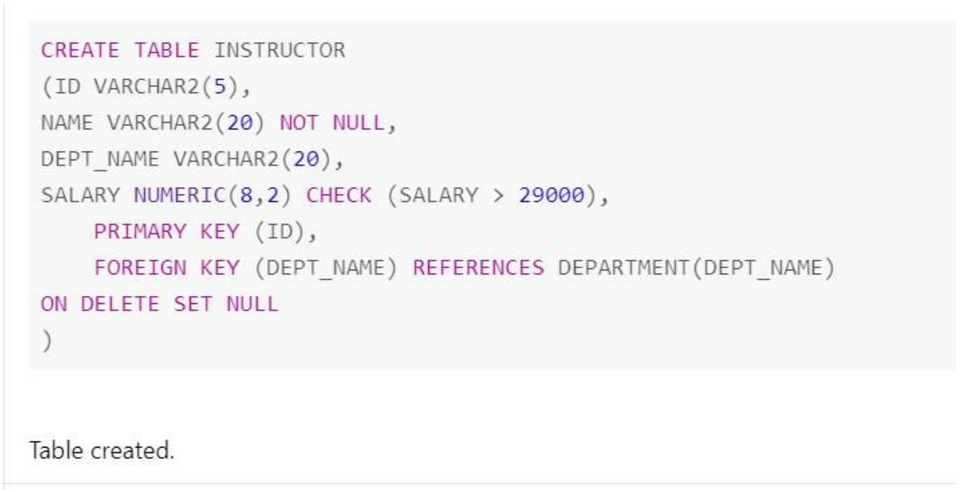


```
Statement processed.
Factorial 6 is 720
```

DROP FUNCTION fact;

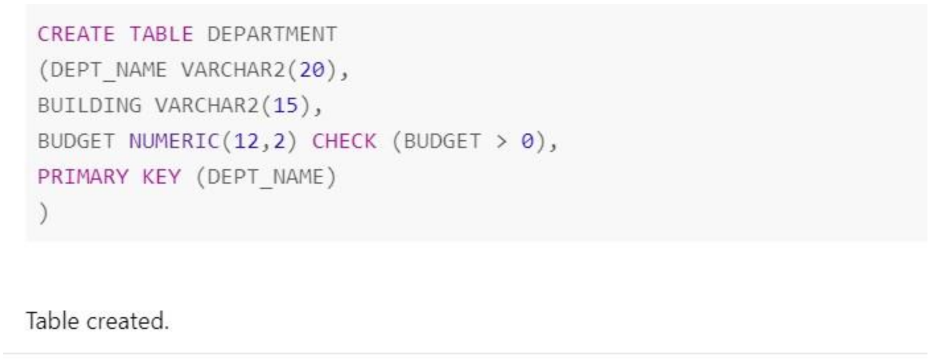
## Experiment – 15

Write PL/SQL program to implement Trigger on table.



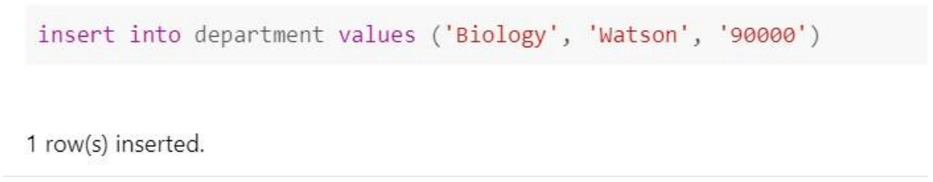
```
CREATE TABLE INSTRUCTOR
(ID VARCHAR2(5),
NAME VARCHAR2(20) NOT NULL,
DEPT_NAME VARCHAR2(20),
SALARY NUMERIC(8,2) CHECK (SALARY > 29000),
    PRIMARY KEY (ID),
    FOREIGN KEY (DEPT_NAME) REFERENCES DEPARTMENT(DEPT_NAME)
ON DELETE SET NULL
)
```

```
Table created.
```



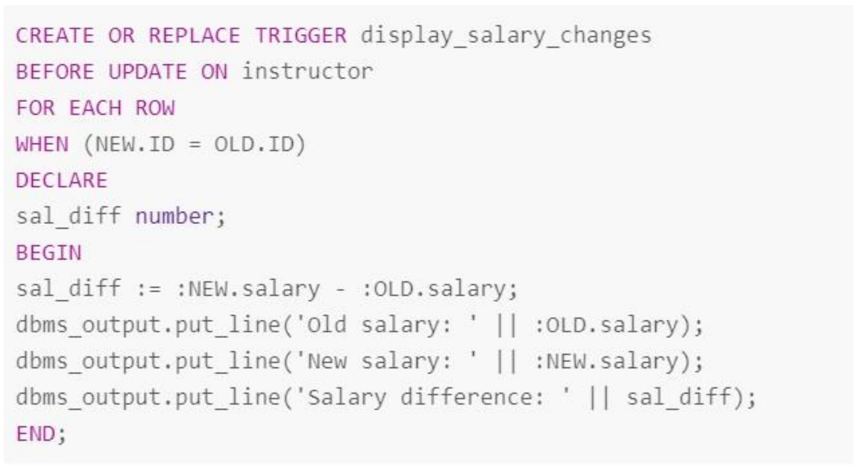
```
CREATE TABLE DEPARTMENT
(DEPT_NAME VARCHAR2(20),
BUILDING VARCHAR2(15),
BUDGET NUMERIC(12,2) CHECK (BUDGET > 0),
PRIMARY KEY (DEPT_NAME)
)
```

```
Table created.
```



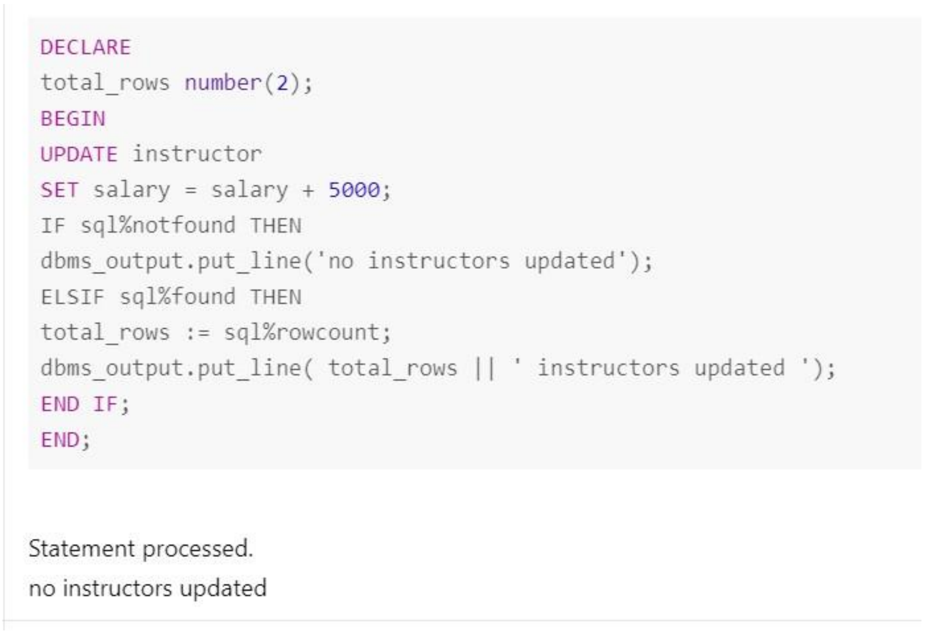
```
insert into department values ('Biology', 'Watson', '90000')
```

```
1 row(s) inserted.
```



```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE UPDATE ON instructor
FOR EACH ROW
WHEN (NEW.ID = OLD.ID)
DECLARE
sal_diff number;
BEGIN
sal_diff := :NEW.salary - :OLD.salary;
dbms_output.put_line('Old salary: ' || :OLD.salary);
dbms_output.put_line('New salary: ' || :NEW.salary);
dbms_output.put_line('Salary difference: ' || sal_diff);
END;
```

Trigger created.



```
DECLARE
total_rows number(2);
BEGIN
UPDATE instructor
SET salary = salary + 5000;
IF sql%notfound THEN
dbms_output.put_line('no instructors updated');
ELSIF sql%found THEN
total_rows := sql%rowcount;
dbms_output.put_line( total_rows || ' instructors updated ');
END IF;
END;
```

```
Statement processed.
no instructors updated
```

## Experiment – 16

Write PL/SQL program to implement Cursor on table.

```
CREATE TABLE customers(
ID NUMBER PRIMARY KEY,
NAME VARCHAR2(20) NOT NULL,
AGE NUMBER,
ADDRESS VARCHAR2(20),
SALARY NUMERIC(20,2))
```

```
Table created.
```

```
INSERT INTO customers VALUES(1,'Ramesh',23,'Allabad',25000)
```

```
1 row(s) inserted.
```

```
INSERT INTO customers VALUES(2, 'Suresh',22,'Kanpur',27000)
```

```
1 row(s) inserted.
```

```
INSERT INTO customers VALUES(3, 'Mahesh',24,'Ghaziabad',29000)
```

```
1 row(s) inserted.
```

```
DECLARE
total_rows number(2);
BEGIN
UPDATE customers
SET salary = salary + 5000;
IF sql%notfound THEN
dbms_output.put_line('no customers updated');
ELSIF sql%found THEN
total_rows := sql%rowcount;
dbms_output.put_line( total_rows || ' customers updated ');
END IF;
END;
```

```
Statement processed.
3 customers updated
```

```
DECLARE
c_id customers.id%type;
c_name customers.name%type;
c_addr customers.address%type;
CURSOR c_customers is
SELECT id, name, address FROM customers;
BEGIN
OPEN c_customers;
LOOP
FETCH c_customers into c_id, c_name, c_addr;
EXIT WHEN c_customers%notfound;
dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
END LOOP;
CLOSE c_customers;
END;
```

```
Statement processed.
2 Suresh Kanpur
1 Ramesh Allabad
3 Mahesh Ghaziabad
```