EXPERIMENT-1

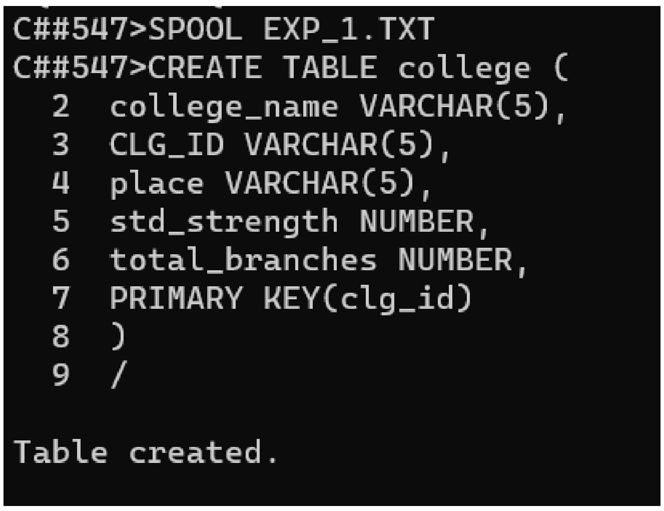
AIM : Write SQL queries to CREATE TABLES for various databases using DDL commands (i.e.CREATE, ALTER, DROP, TRUNCATE).

CREATE TABLE:

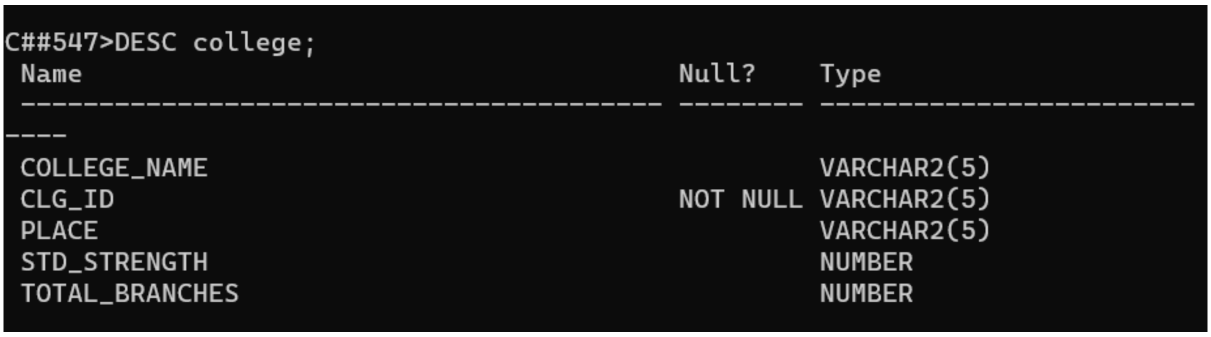
Creates a table with specified constraints

SYNTAX:

CREATE TABLE tablename (
column1 data_ type [constraint] [,
column2 data_ type [constraint]] [,
PRIMARY KEY (column1 [, column2])] [,
FOREIGN KEY (column1 [, column2]) REFERENCES tablename] [,CONSTRAINT constraint]);



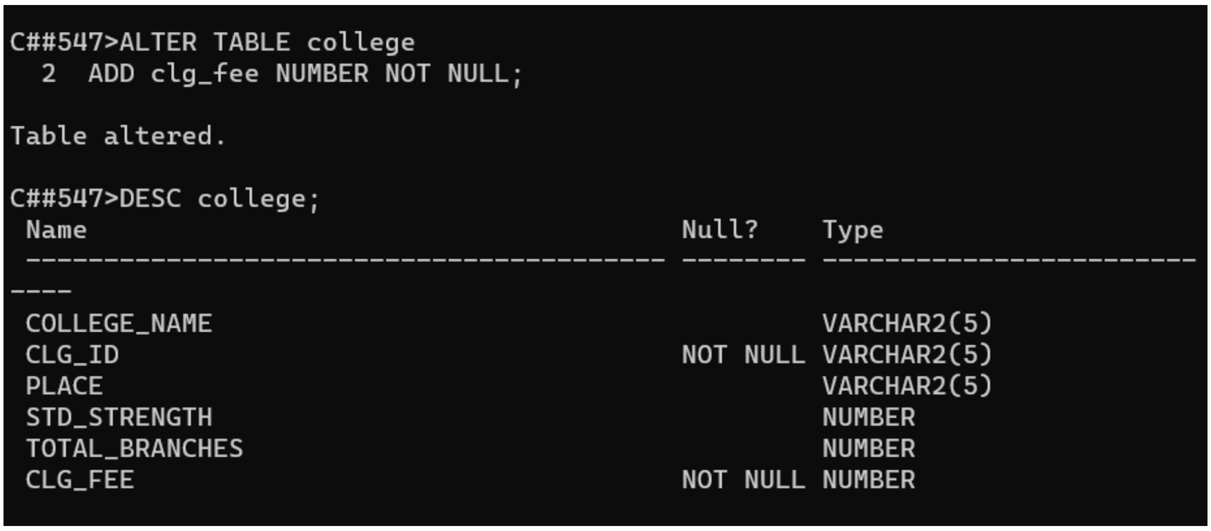
```
C##547>SPOOL EXP_1.TXT
C##547>CREATE TABLE college (
  2  college_name VARCHAR(5),
  3  CLG_ID VARCHAR(5),
  4  place VARCHAR(5),
  5  std_strength NUMBER,
  6  total_branches NUMBER,
  7  PRIMARY KEY(clg_id)
  8  )
  9  /

Table created.
```



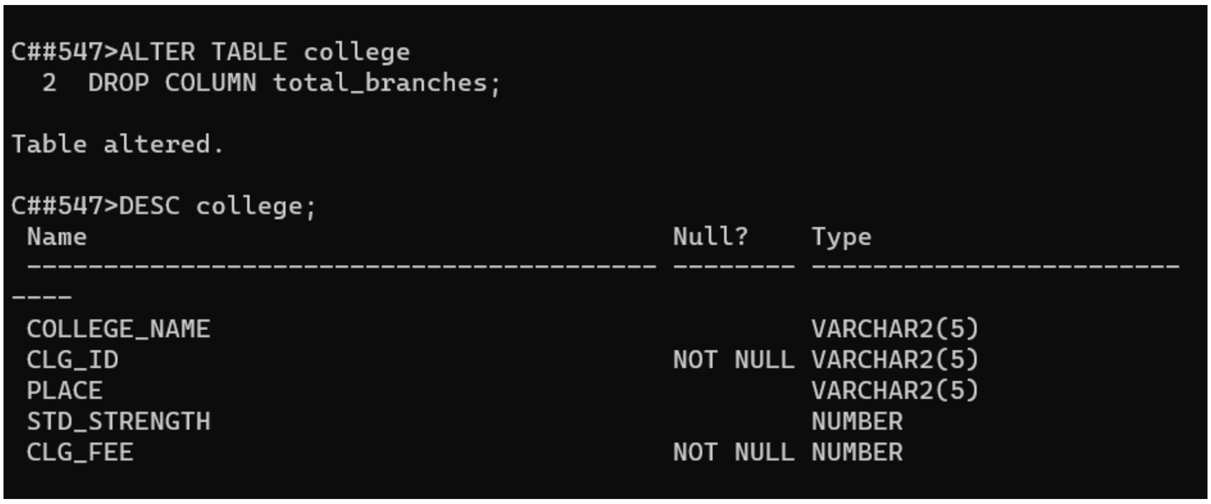
```
C##547>DESC college;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
----
 COLLEGE_NAME                                       VARCHAR2(5)
 CLG_ID                                    NOT NULL VARCHAR2(5)
 PLACE                                              VARCHAR2(5)
 STD_STRENGTH                                       NUMBER
 TOTAL_BRANCHES                                     NUMBER
```

ALTER TABLE :

Used to add or modify table details like column names and data types, column constraints.



```
C##547>ALTER TABLE college
  2  ADD clg_fee NUMBER NOT NULL;

Table altered.

C##547>DESC college;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 COLLEGE_NAME                                       VARCHAR2(5)
 CLG_ID                                    NOT NULL VARCHAR2(5)
 PLACE                                              VARCHAR2(5)
 STD_STRENGTH                                       NUMBER
 TOTAL_BRANCHES                                     NUMBER
 CLG_FEE                                   NOT NULL NUMBER
```



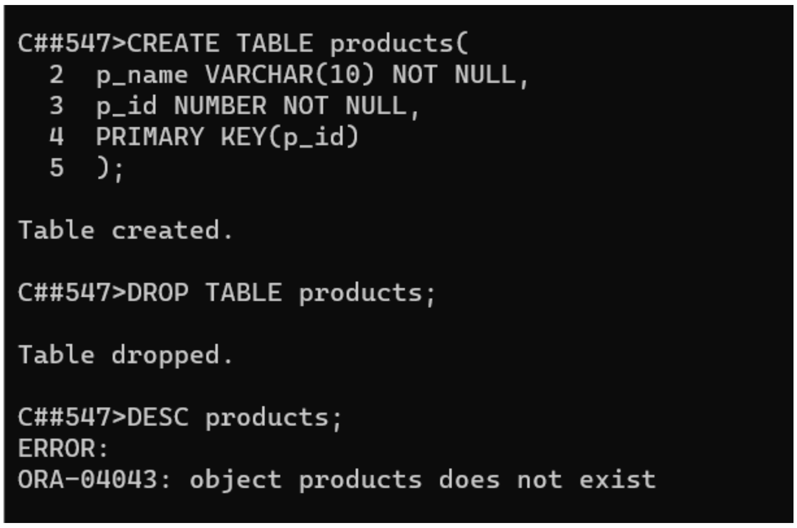
```
C##547>ALTER TABLE college
  2  DROP COLUMN total_branches;

Table altered.

C##547>DESC college;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 COLLEGE_NAME                                       VARCHAR2(5)
 CLG_ID                                    NOT NULL VARCHAR2(5)
 PLACE                                              VARCHAR2(5)
 STD_STRENGTH                                       NUMBER
 CLG_FEE                                   NOT NULL NUMBER
```

DROP TABLE:

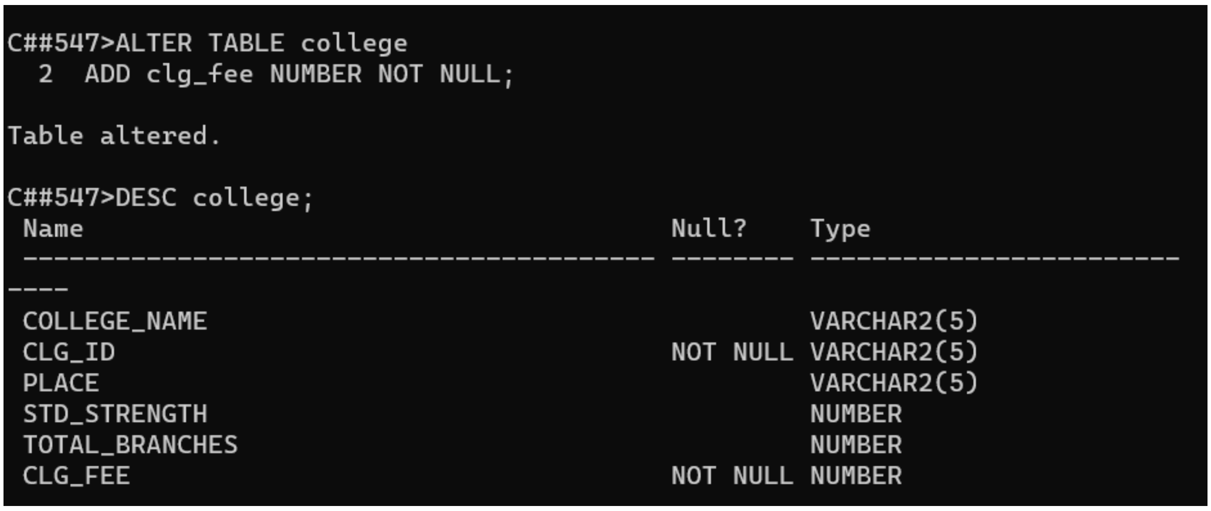
Deletes the specified table.

SYNTAX:

DROP TABLE table_name;



```
C##547>CREATE TABLE products(
  2  p_name VARCHAR(10) NOT NULL,
  3  p_id NUMBER NOT NULL,
  4  PRIMARY KEY(p_id)
  5  );

Table created.

C##547>DROP TABLE products;

Table dropped.

C##547>DESC products;
ERROR:
ORA-04043: object products does not exist
```



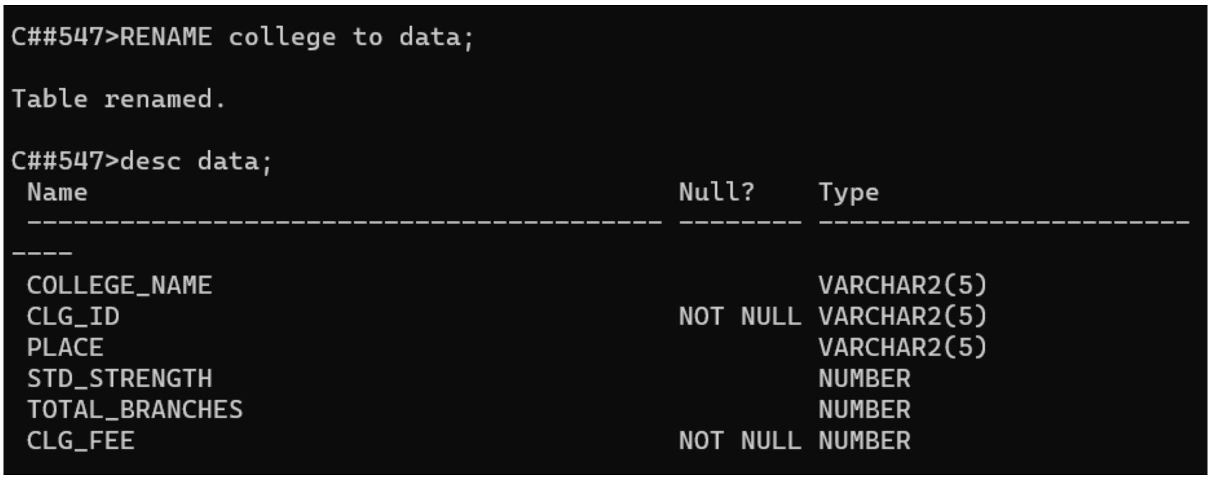
```
C##547>ALTER TABLE college
  2  ADD clg_fee NUMBER NOT NULL;

Table altered.

C##547>DESC college;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 COLLEGE_NAME                                       VARCHAR2(5)
 CLG_ID                                    NOT NULL VARCHAR2(5)
 PLACE                                              VARCHAR2(5)
 STD_STRENGTH                                       NUMBER
 TOTAL_BRANCHES                                     NUMBER
 CLG_FEE                                   NOT NULL NUMBER
```

RENAME TABLE:

To rename table_name, column_name

SYNTAXES:

RENAME new_table_name TO old_table_name;



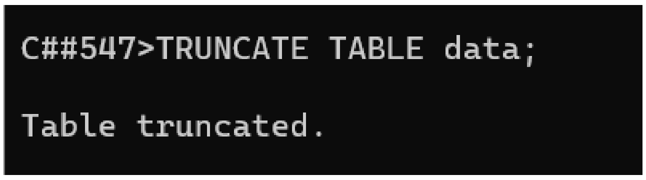
```
C##547>RENAME college to data;

Table renamed.

C##547>desc data;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 COLLEGE_NAME                                       VARCHAR2(5)
 CLG_ID                                    NOT NULL VARCHAR2(5)
 PLACE                                              VARCHAR2(5)
 STD_STRENGTH                                       NUMBER
 TOTAL_BRANCHES                                     NUMBER
 CLG_FEE                                   NOT NULL NUMBER
```

TRUNCATE TABLE:

To remove all rows in a specified table.

SYNTAX:

TRUNCATE TABLE table_name;

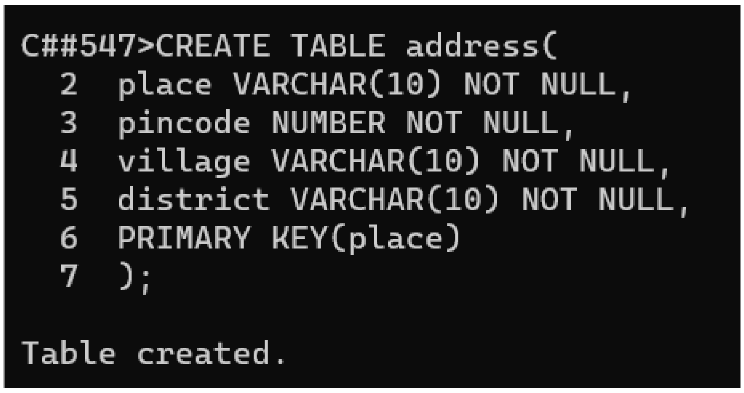


```
C##547>TRUNCATE TABLE data;

Table truncated.
```

EXPERIMENT-2

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

Creating table :



```
C##547>CREATE TABLE address(
  2  place VARCHAR(10) NOT NULL,
  3  pincode NUMBER NOT NULL,
  4  village VARCHAR(10) NOT NULL,
  5  district VARCHAR(10) NOT NULL,
  6  PRIMARY KEY(place)
  7  );

Table created.
```

INSERT COMMAND:

It is used to add values to a table.

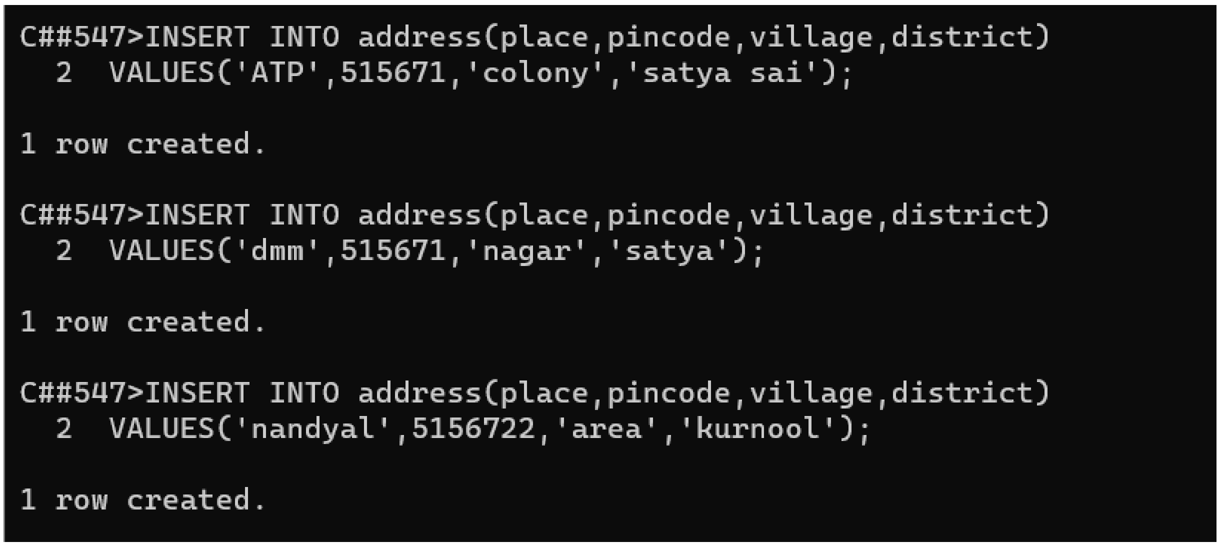
SYNTAX:

INSERT INTO tablename

VALUES (value1,value2,...,valuen);

INSERT INTO tablename (column1, column2,...,column)

VALUES (value1, value2,...,valuen);



```
C##547>INSERT INTO address(place,pincode,village,district)
  2  VALUES('ATP',515671,'colony','satya sai');

1 row created.

C##547>INSERT INTO address(place,pincode,village,district)
  2  VALUES('dmm',515671,'nagar','satya');

1 row created.

C##547>INSERT INTO address(place,pincode,village,district)
  2  VALUES('nandyal',5156722,'area','kurnool');

1 row created.
```

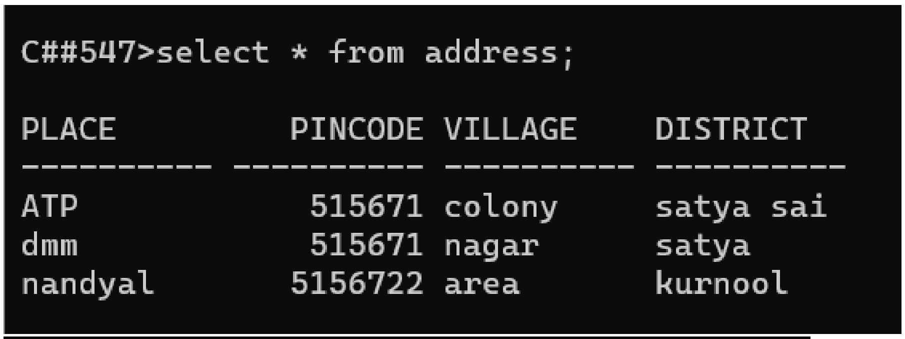
SELECT COMMAND:

The SELECT command used to list the contents of a table.

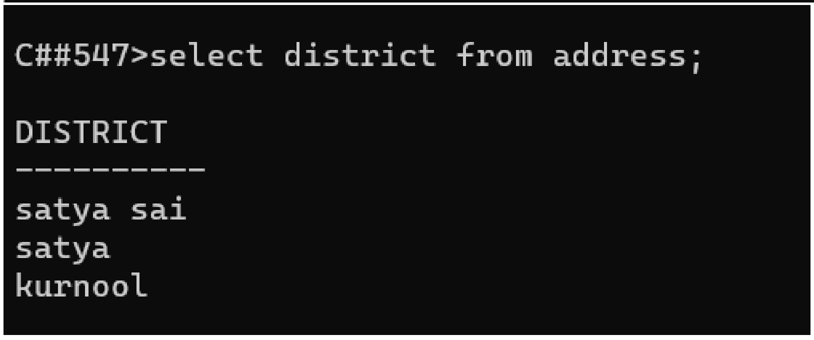
SYNTAX:

Select * from table_name;

Select col_name from table_name;



```
C##547>select * from address;

PLACE         PINCODE VILLAGE    DISTRICT
---------- ---------- ---------- ----------
ATP            515671 colony     satya sai
dmm            515671 nagar      satya
nandyal       5156722 area       kurnool
```



```
C##547>select district from address;

DISTRICT
----------
satya sai
satya
kurnool
```

UPDATE COMMAND:

The update command used to modify the contents of specified table.

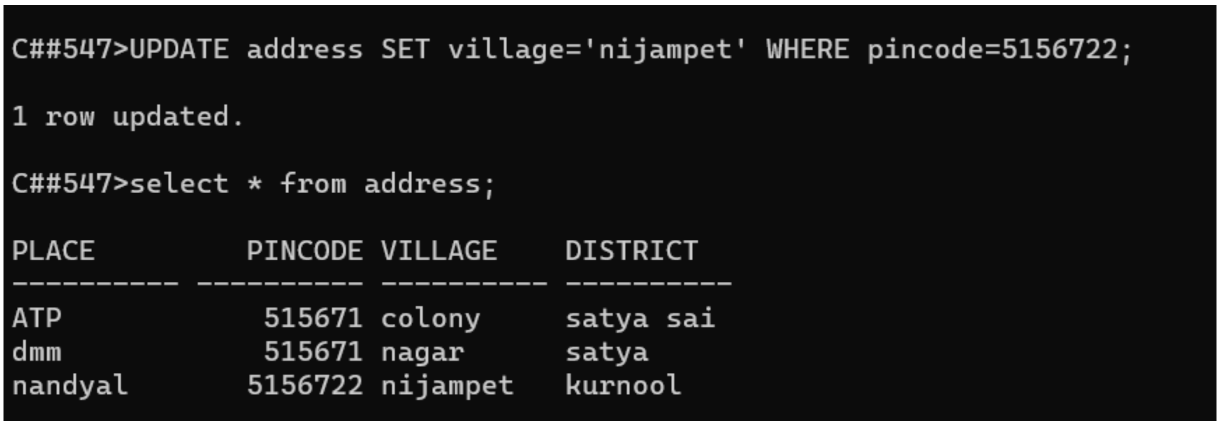
SYNTAX:

UPDATE tablename

SET column_name = value[,

Column_name = value]

[WHERE condition_lsit];



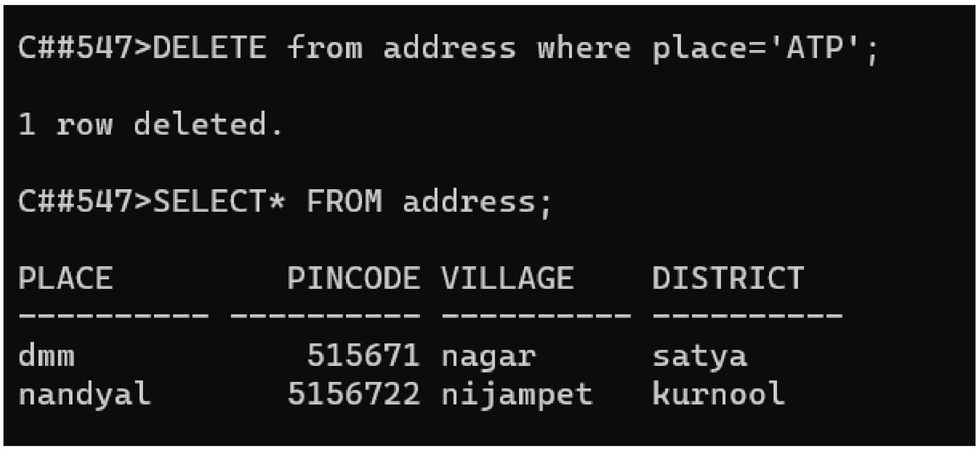
```
C##547>UPDATE address SET village='nijampet' WHERE pincode=5156722;

1 row updated.

C##547>select * from address;

PLACE         PINCODE VILLAGE    DISTRICT
---------- ---------- ---------- ----------
ATP            515671 colony     satya sai
dmm            515671 nagar      satya
nandyal       5156722 nijampet   kurnool
```

DELETE COMMAND:

To delete all rows or specified rows in a table.

SYNTAX:

DELETE FROM tablename [WHERE condition_list];

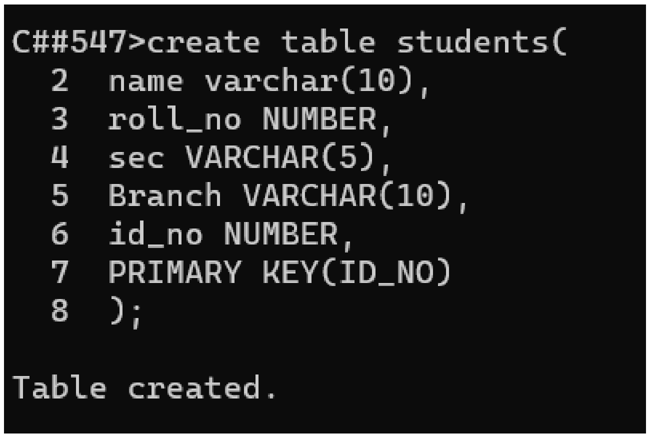


```
C##547>DELETE from address where place='ATP';

1 row deleted.

C##547>SELECT* FROM address;

PLACE         PINCODE VILLAGE    DISTRICT
---------- ---------- ---------- ----------
dmm            515671 nagar      satya
nandyal       5156722 nijampet   kurnool
```

Experiment-3

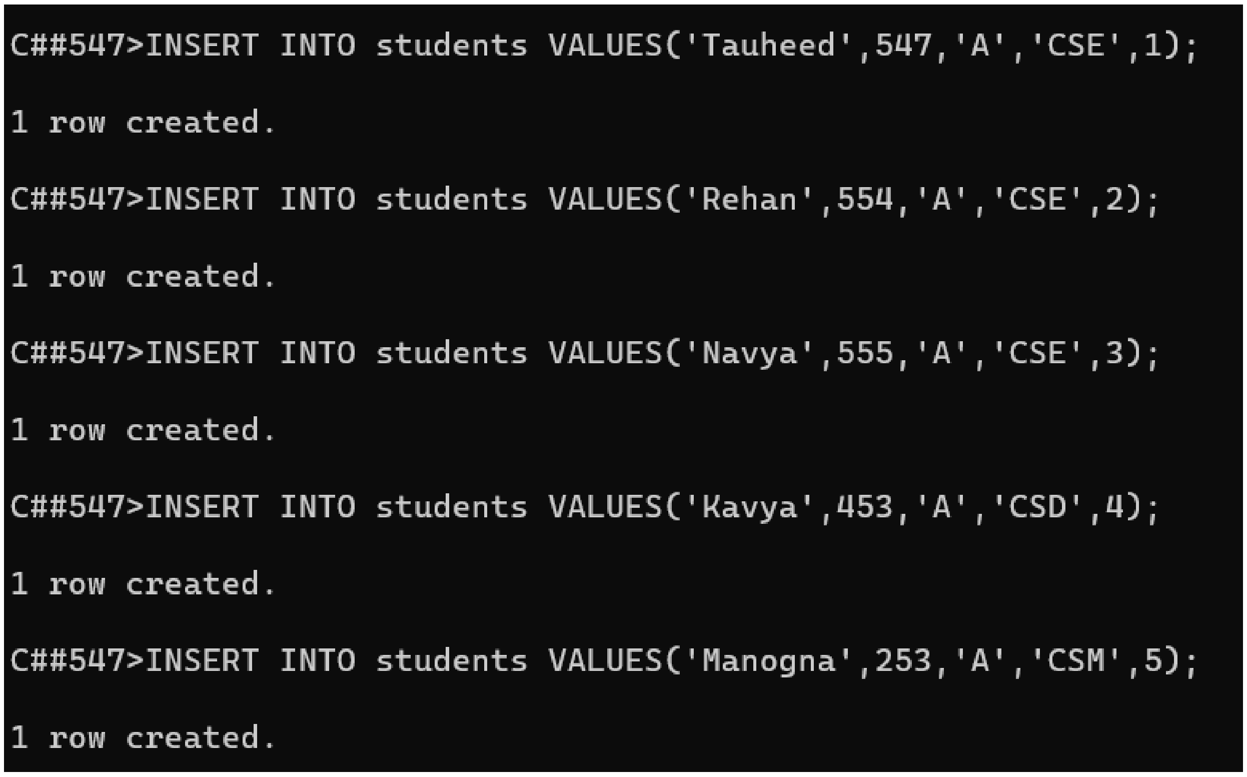
Aim: To implement a view level design using CREATE VIEW,ALTER VIEW and DELETE VIEW ddl commands.

Creating a table:



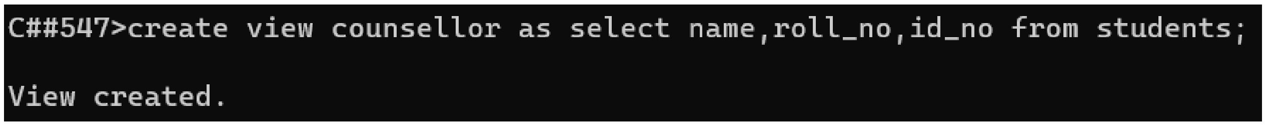
```
C##547>create table students(
  2  name varchar(10),
  3  roll_no NUMBER,
  4  sec VARCHAR(5),
  5  Branch VARCHAR(10),
  6  id_no NUMBER,
  7  PRIMARY KEY(ID_NO)
  8  );

Table created.
```

By using insert command we can insert values in a tables



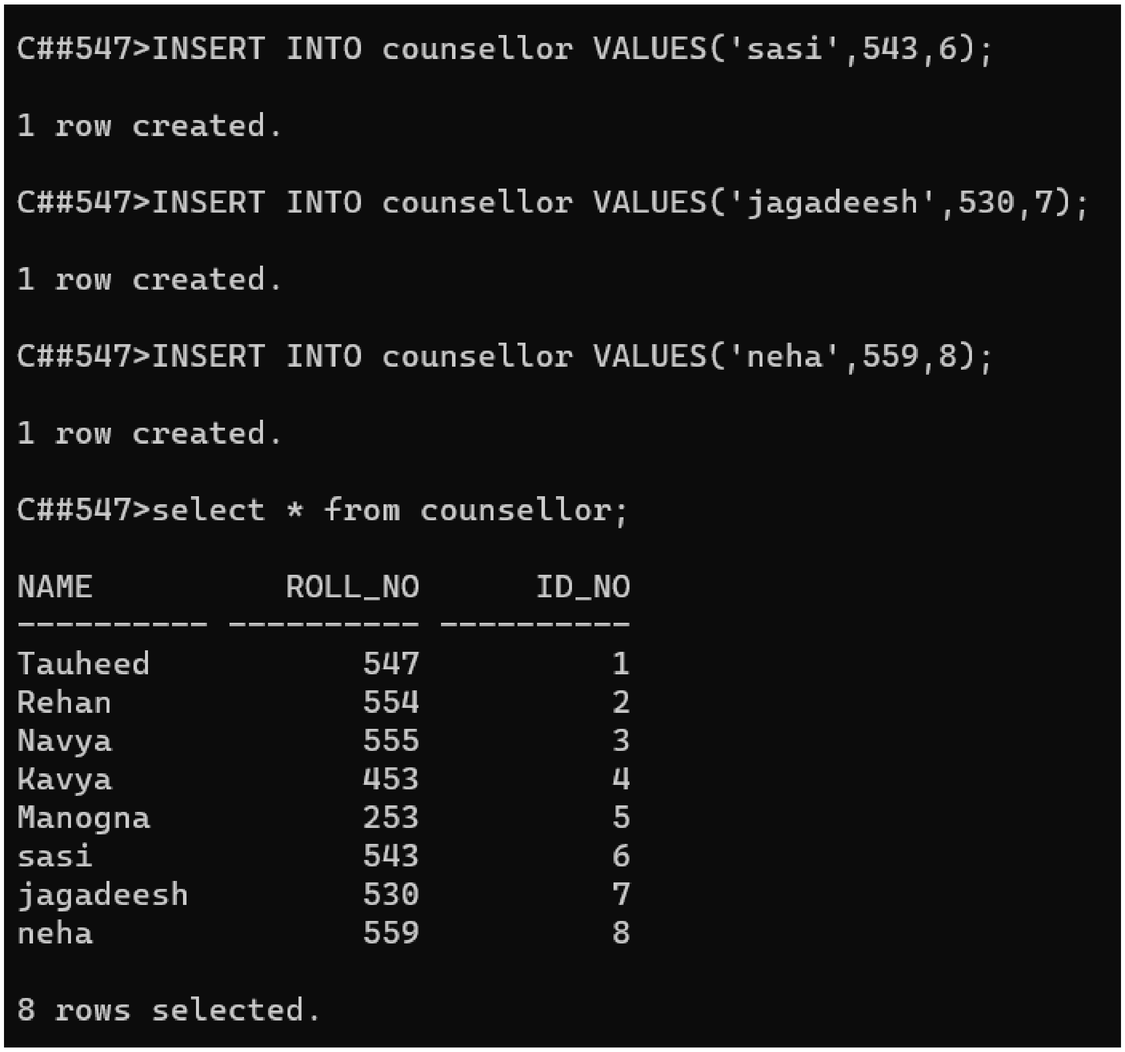
```
C##547>INSERT INTO students VALUES('Tauheed',547,'A','CSE',1);

1 row created.

C##547>INSERT INTO students VALUES('Rehan',554,'A','CSE',2);

1 row created.

C##547>INSERT INTO students VALUES('Navya',555,'A','CSE',3);

1 row created.

C##547>INSERT INTO students VALUES('Kavya',453,'A','CSD',4);

1 row created.

C##547>INSERT INTO students VALUES('Manogna',253,'A','CSM',5);

1 row created.
```

Creating view councellor:



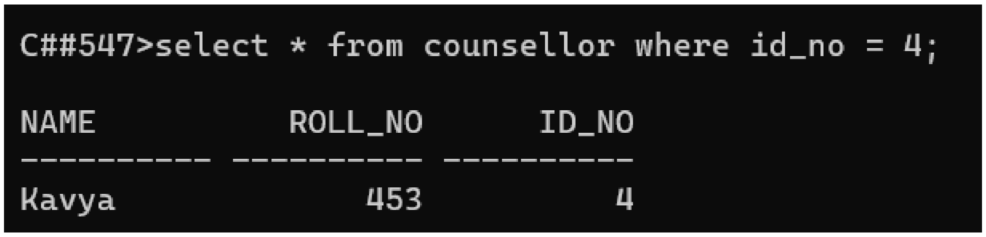
```
C##547>create view counsellor as select name,roll_no,id_no from students;

View created.
```

Inserting values into councellor:



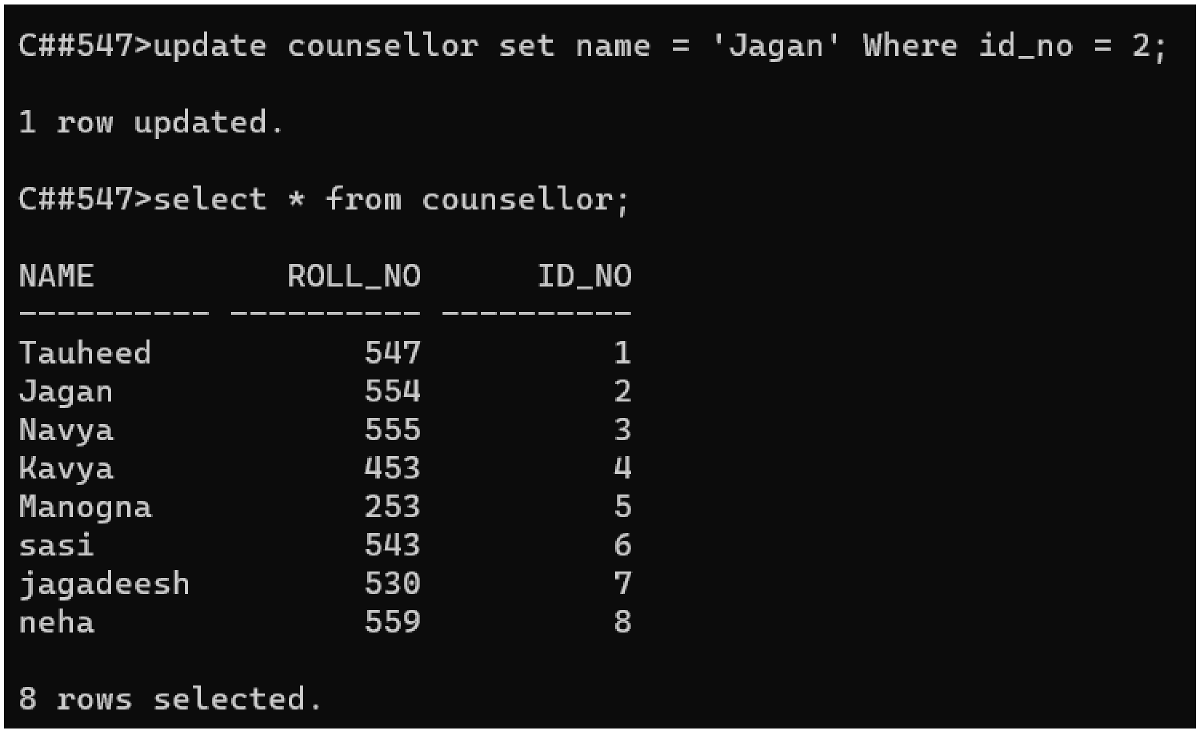
```
C##547>INSERT INTO counsellor VALUES('sasi',543,6);

1 row created.

C##547>INSERT INTO counsellor VALUES('jagadeesh',530,7);

1 row created.

C##547>INSERT INTO counsellor VALUES('neha',559,8);

1 row created.

C##547>select * from counsellor;

NAME          ROLL_NO      ID_NO
---------- ---------- ----------
Tauheed           547          1
Rehan             554          2
Navya             555          3
Kavya             453          4
Manogna           253          5
sasi              543          6
jagadeesh         530          7
neha              559          8

8 rows selected.
```

Selecting specific row :



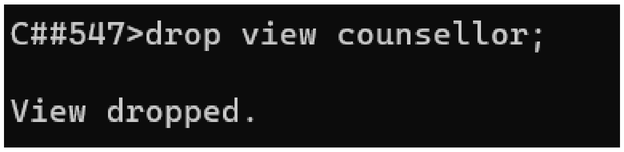
```
C##547>select * from counsellor where id_no = 4;

NAME          ROLL_NO      ID_NO
---------- ---------- ----------
Kavya             453          4
```

Update :



```
C##547>update counsellor set name = 'Jagan' Where id_no = 2;

1 row updated.

C##547>select * from counsellor;

NAME          ROLL_NO      ID_NO
---------- ---------- ----------
Tauheed           547          1
Jagan             554          2
Navya             555          3
Kavya             453          4
Manogna           253          5
sasi              543          6
jagadeesh         530          7
neha              559          8

8 rows selected.
```

truncate or drop view:

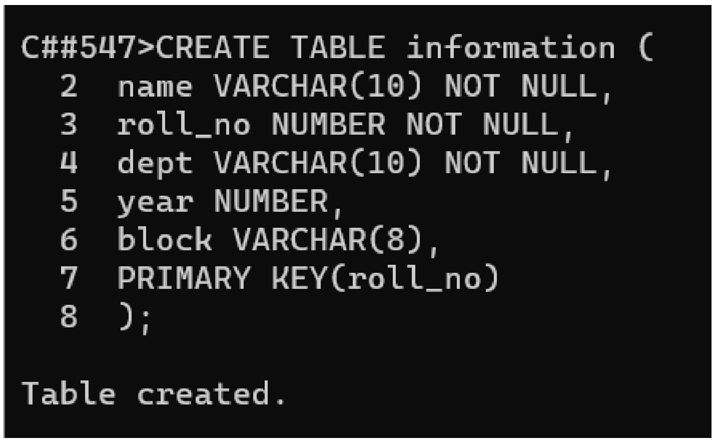


```
C##547>drop view counsellor;

View dropped.
```

EXPERIMENT-4

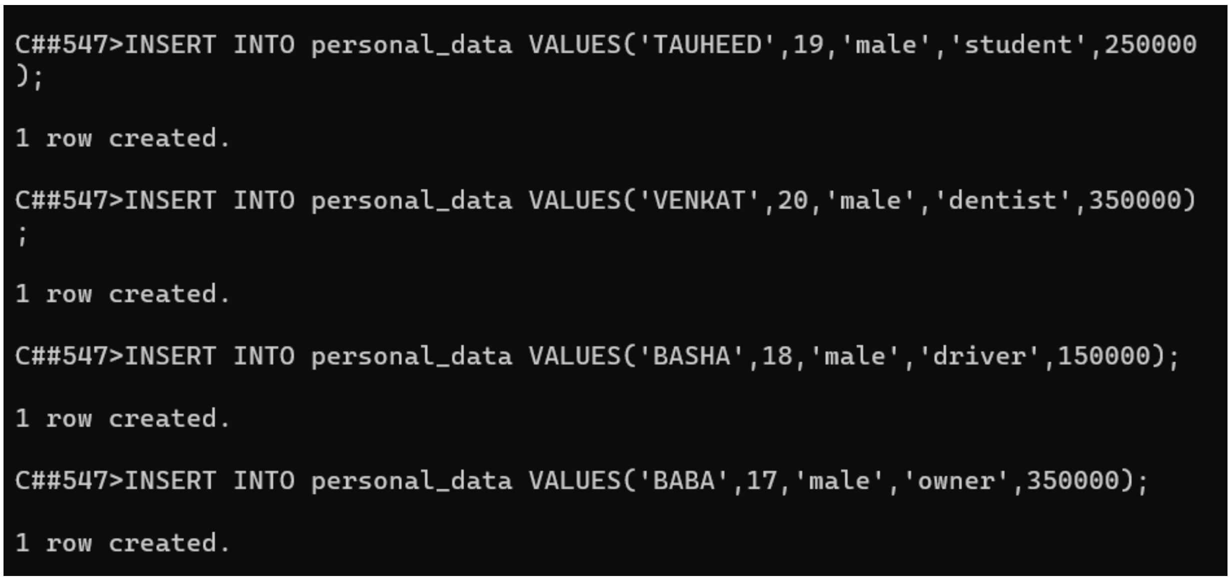
AIM : To create/perform relational set operations(i.e UNION UNIONALL, INTERSECT, MINUS, CROSS JOIN, NATURAL JOIN.)

Creating tables:



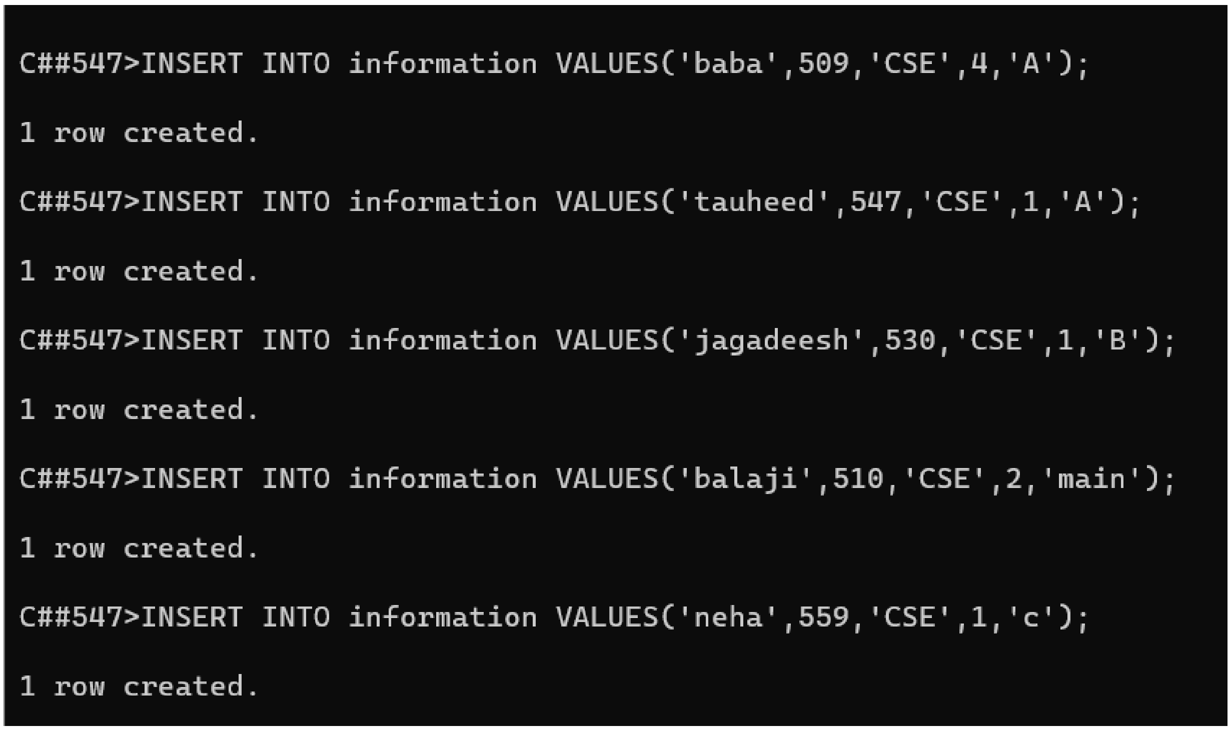
```
C##547>CREATE TABLE information (
  2  name VARCHAR(10) NOT NULL,
  3  roll_no NUMBER NOT NULL,
  4  dept VARCHAR(10) NOT NULL,
  5  year NUMBER,
  6  block VARCHAR(8),
  7  PRIMARY KEY(roll_no)
  8  );

Table created.
```

Inserting values into **personal_data** table :



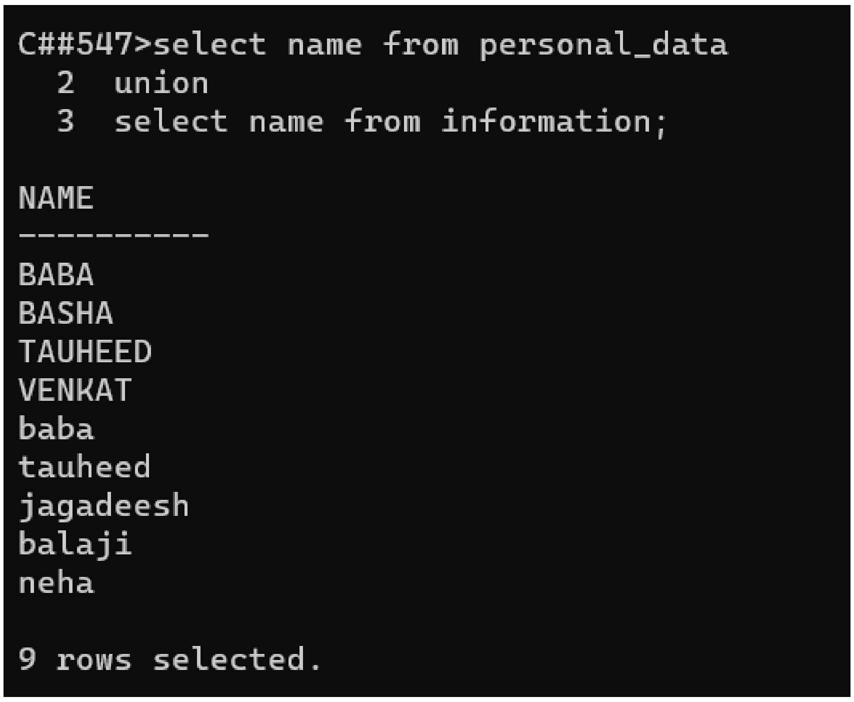
```
C##547>INSERT INTO personal_data VALUES('TAUHEED',19,'male','student',250000
);

1 row created.

C##547>INSERT INTO personal_data VALUES('VENKAT',20,'male','dentist',350000)
;

1 row created.

C##547>INSERT INTO personal_data VALUES('BASHA',18,'male','driver',150000);

1 row created.

C##547>INSERT INTO personal_data VALUES('BABA',17,'male','owner',350000);

1 row created.
```

Inserting values into **information** table :



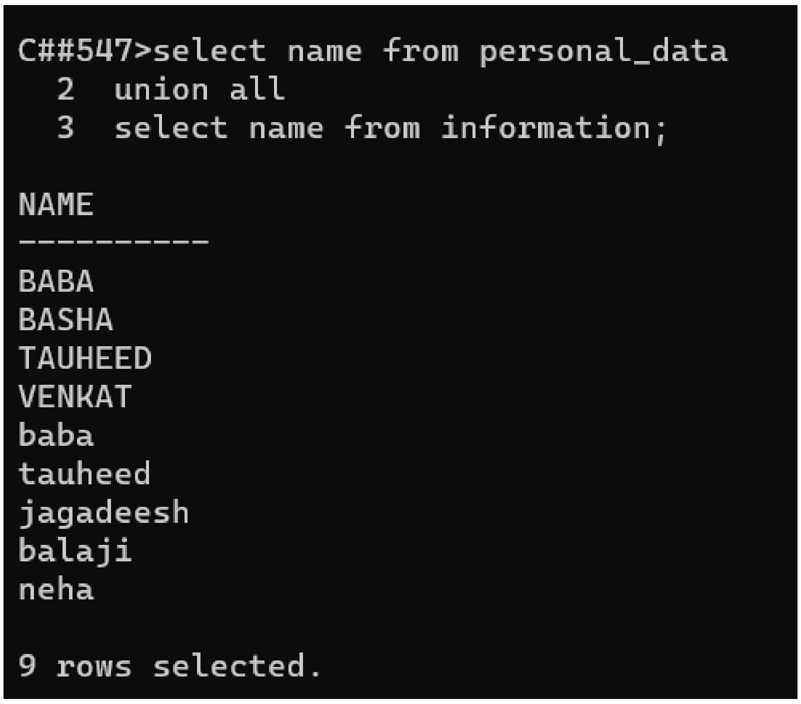
```
C##547>INSERT INTO information VALUES('baba',509,'CSE',4,'A');

1 row created.

C##547>INSERT INTO information VALUES('tauheed',547,'CSE',1,'A');

1 row created.

C##547>INSERT INTO information VALUES('jagadeesh',530,'CSE',1,'B');

1 row created.

C##547>INSERT INTO information VALUES('balaji',510,'CSE',2,'main');

1 row created.

C##547>INSERT INTO information VALUES('neha',559,'CSE',1,'c');

1 row created.
```

Union operation :



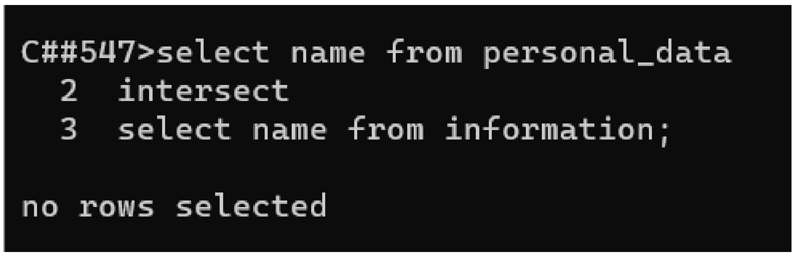
```
C##547>select name from personal_data
  2  union
  3  select name from information;

NAME
----------
BABA
BASHA
TAUHEED
VENKAT
baba
tauheed
jagadeesh
balaji
neha

9 rows selected.
```

Union all operation :



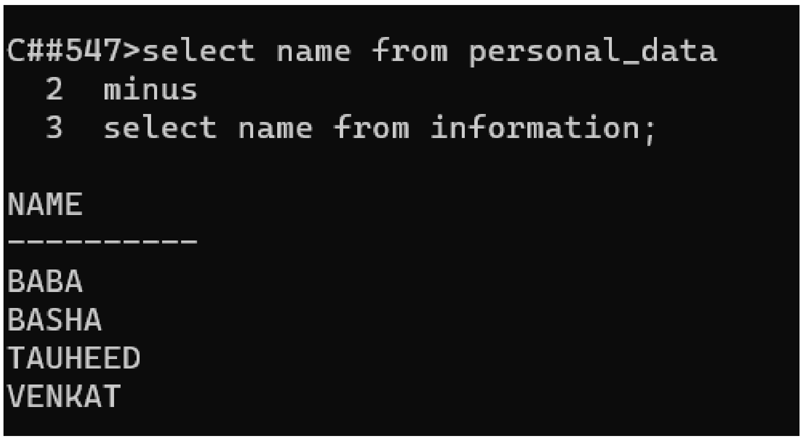
```
C##547>select name from personal_data
  2  union all
  3  select name from information;

NAME
----------
BABA
BASHA
TAUHEED
VENKAT
baba
tauheed
jagadeesh
balaji
neha

9 rows selected.
```

Intersect operation :



```
C##547>select name from personal_data
  2  intersect
  3  select name from information;

no rows selected
```

Minus operation :

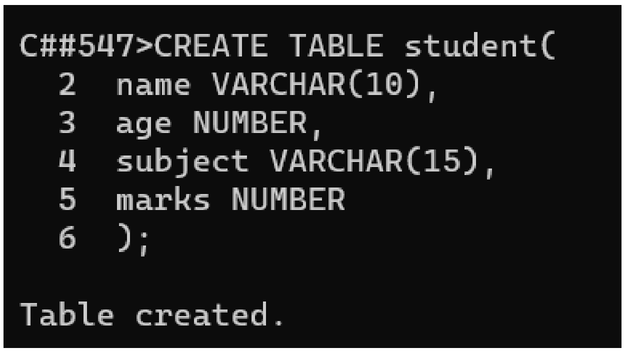


```
C##547>select name from personal_data
  2  minus
  3  select name from information;

NAME
----------
BABA
BASHA
TAUHEED
VENKAT
```

EXPERIMENT-5

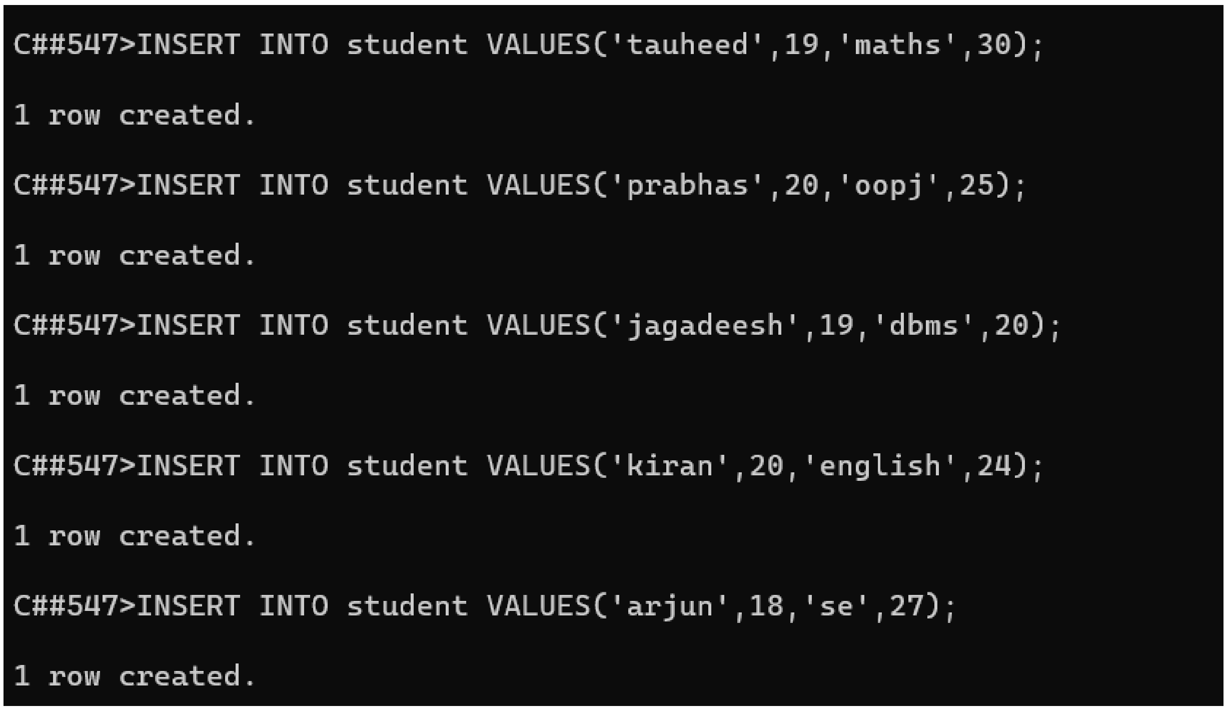
Aim: write SQL queries for the aggregate functions(sum,count,min,max,avg)

Creating a table:



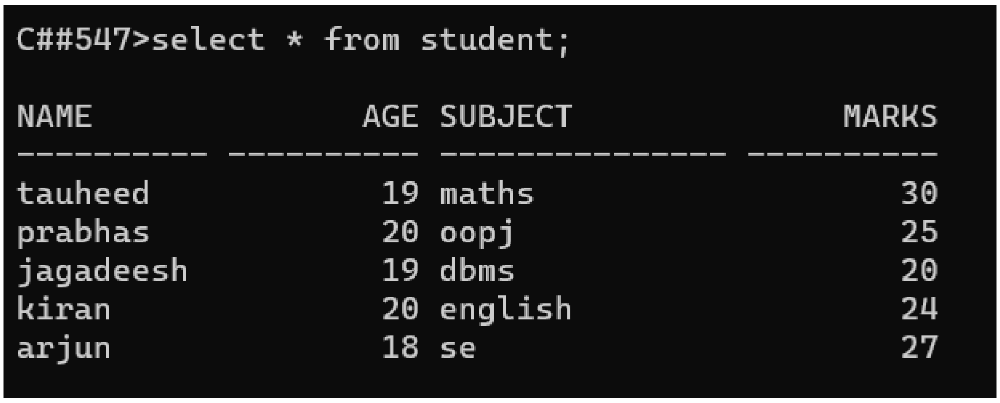
```
C##547>CREATE TABLE student(
name VARCHAR(10),
age NUMBER,
subject VARCHAR(15),
marks NUMBER
);

Table created.
```

Inserting values into table :



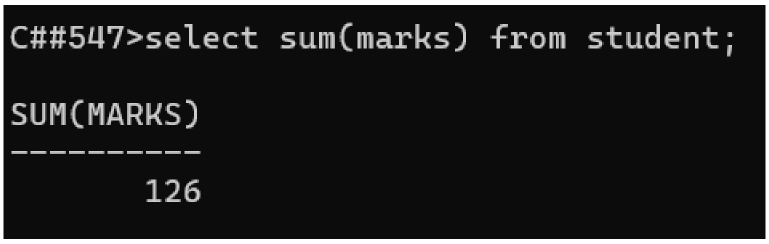
```
C##547>INSERT INTO student VALUES('tauheed',19,'maths',30);

1 row created.

C##547>INSERT INTO student VALUES('prabhas',20,'oopj',25);

1 row created.

C##547>INSERT INTO student VALUES('jagadeesh',19,'dbms',20);

1 row created.

C##547>INSERT INTO student VALUES('kiran',20,'english',24);

1 row created.

C##547>INSERT INTO student VALUES('arjun',18,'se',27);

1 row created.
```

Selecting table :



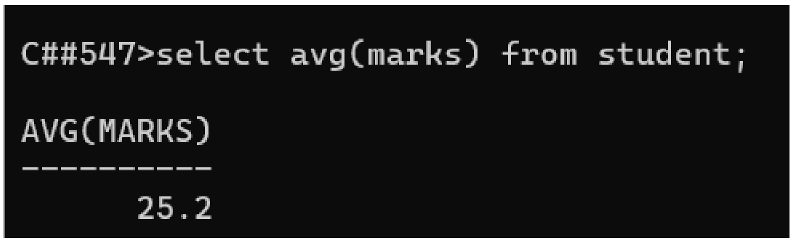
```
C##547>select * from student;

NAME                  AGE SUBJECT               MARKS
---------- ---------- --------------- ----------
tauheed                19 maths                    30
prabhas                20 oopj                     25
jagadeesh              19 dbms                     20
kiran                  20 english                  24
arjun                  18 se                       27
```

Sum();



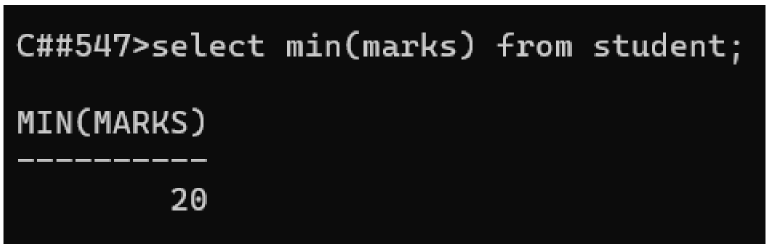
```
C##547>select sum(marks) from student;

SUM(MARKS)
----------
       126
```

Avg();



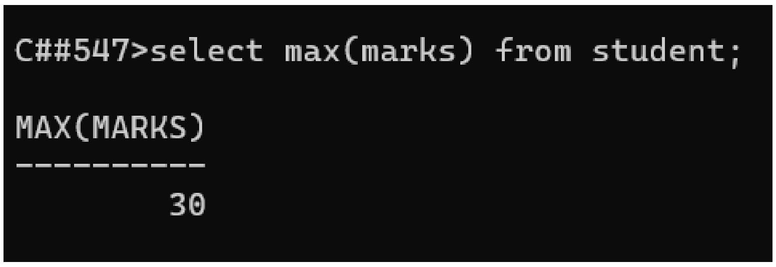
```
C##547>select avg(marks) from student;

AVG(MARKS)
----------
      25.2
```

Min();



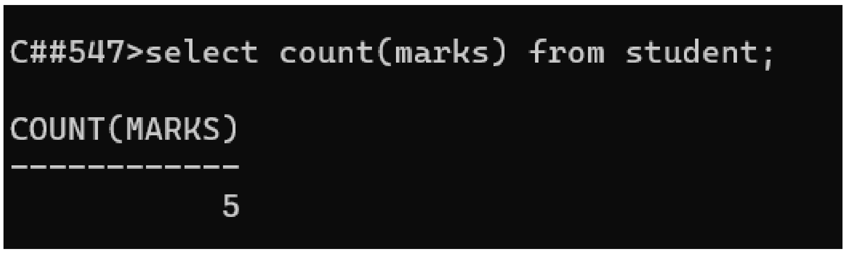
```
C##547>select min(marks) from student;

MIN(MARKS)
----------
        20
```

Max();



```
C##547>select max(marks) from student;

MAX(MARKS)
----------
        30
```

Count();

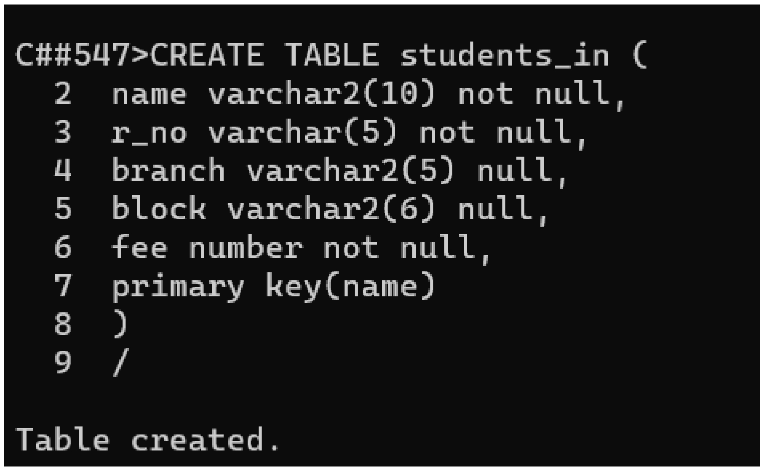


```
C##547>select count(marks) from student;

COUNT(MARKS)
------------
           5
```

EXPERIMENT-6

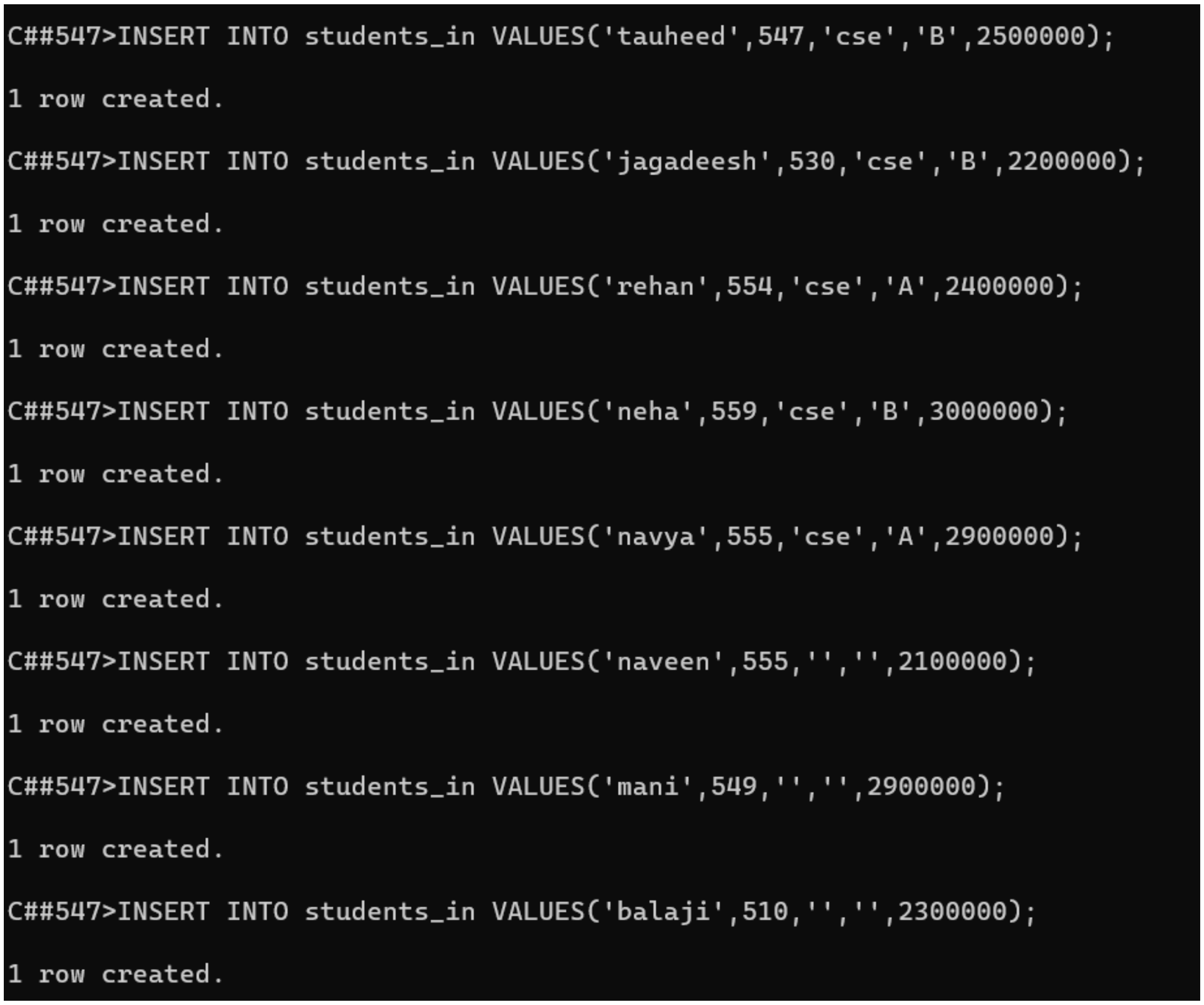
AIM: TO WRITE SQL QUERIES TO PERFORM SPECIAL OPERATIONS (i.e LIKE, BETWEEN, ISNULL, ISNOTNULL)

Creating a table



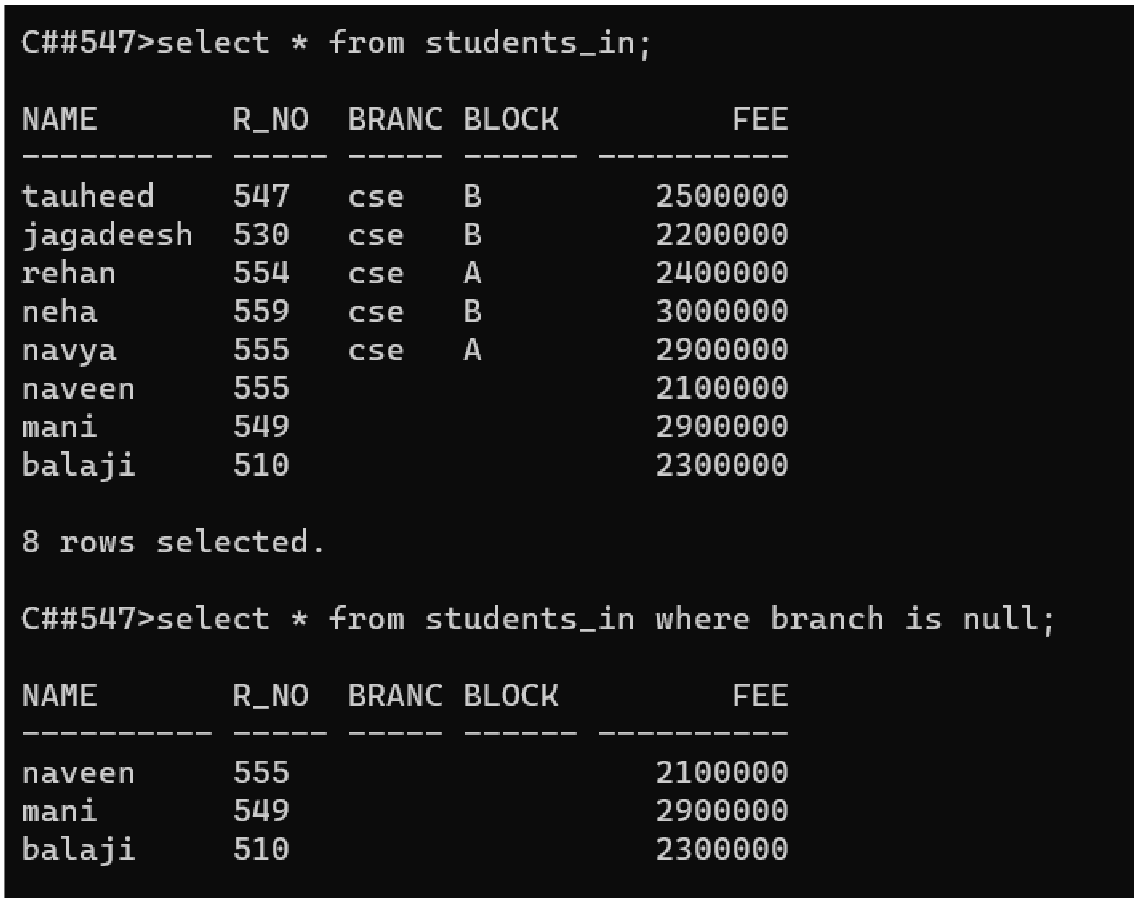
```
C##547>CREATE TABLE students_in (
     name varchar2(10) not null,
     r_no varchar(5) not null,
     branch varchar2(5) null,
     block varchar2(6) null,
     fee number not null,
     primary key(name)
     )
     /

Table created.
```

Inserting values :



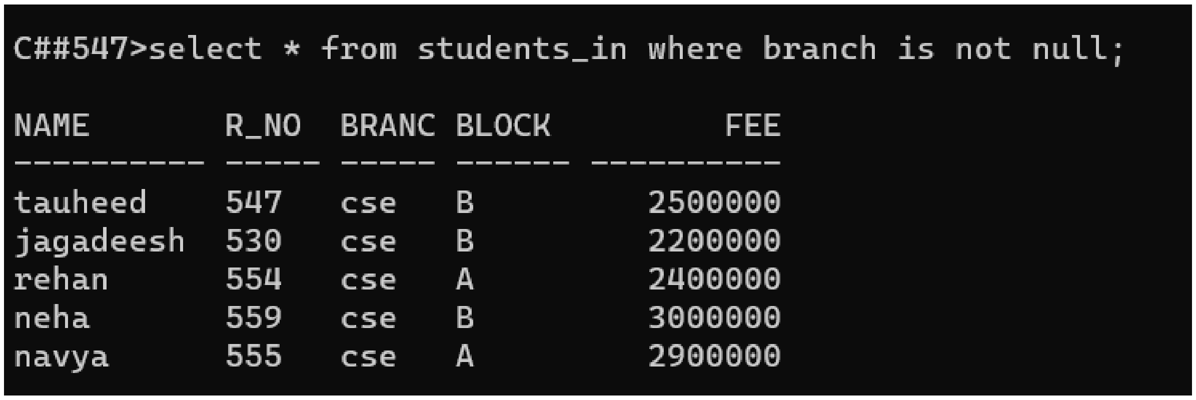
```
C##547>INSERT INTO students_in VALUES('tauheed',547,'cse','B',2500000);

1 row created.

C##547>INSERT INTO students_in VALUES('jagadeesh',530,'cse','B',2200000);

1 row created.

C##547>INSERT INTO students_in VALUES('rehan',554,'cse','A',2400000);

1 row created.

C##547>INSERT INTO students_in VALUES('neha',559,'cse','B',3000000);

1 row created.

C##547>INSERT INTO students_in VALUES('navya',555,'cse','A',2900000);

1 row created.

C##547>INSERT INTO students_in VALUES('naveen',555,'','',2100000);

1 row created.

C##547>INSERT INTO students_in VALUES('mani',549,'','',2900000);

1 row created.

C##547>INSERT INTO students_in VALUES('balaji',510,'','',2300000);

1 row created.
```

Is Null operation :



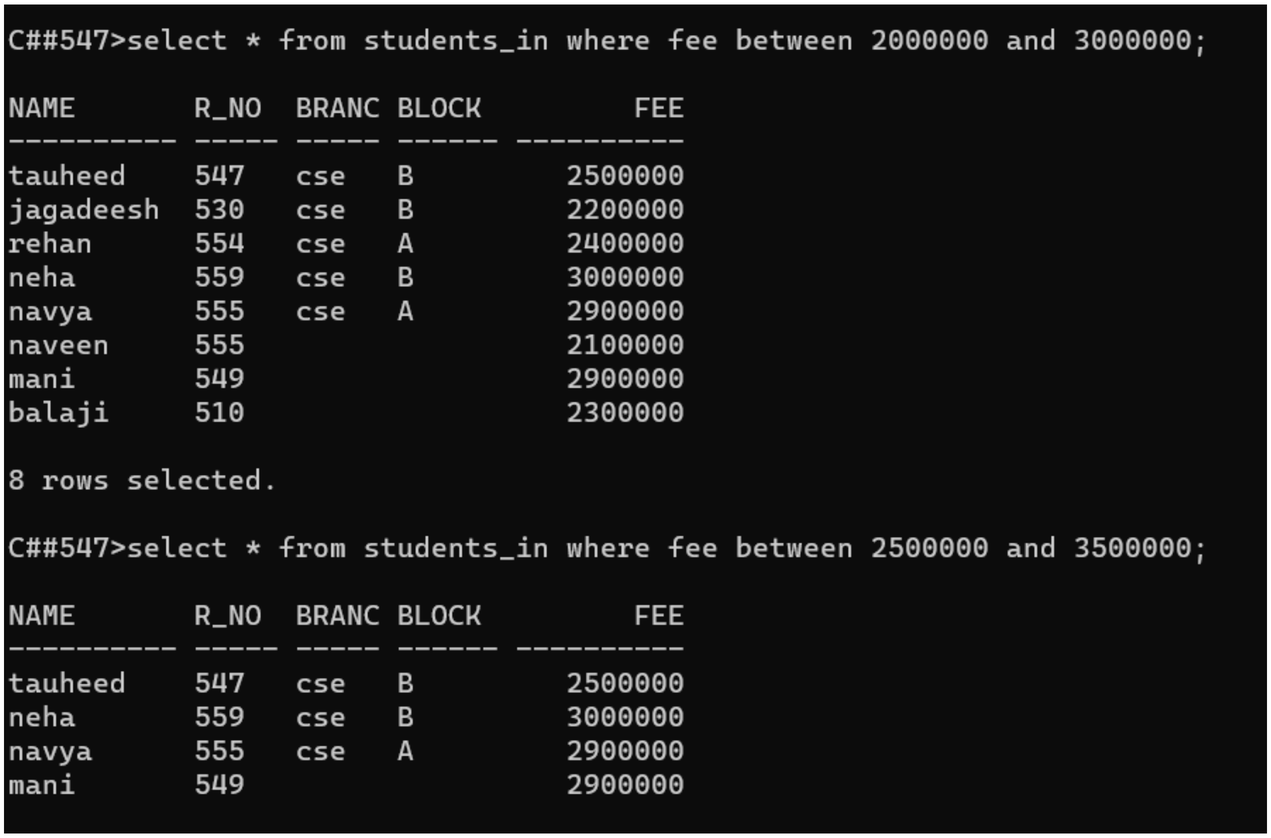
```
C##547>select * from students_in;

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
tauheed    547   cse   B         2500000
jagadeesh  530   cse   B         2200000
rehan      554   cse   A         2400000
neha       559   cse   B         3000000
navya      555   cse   A         2900000
naveen     555                   2100000
mani       549                   2900000
balaji     510                   2300000

8 rows selected.

C##547>select * from students_in where branch is null;

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
naveen     555                   2100000
mani       549                   2900000
balaji     510                   2300000
```

Is not null operation :



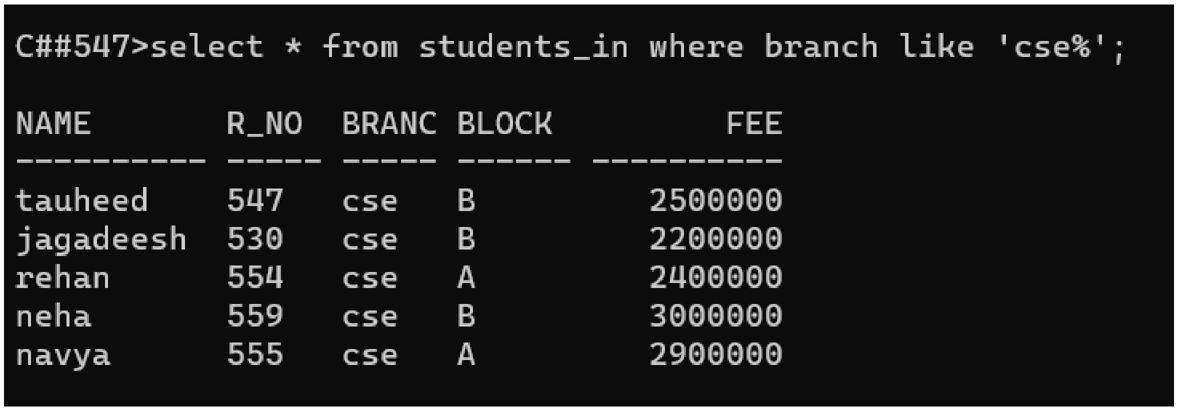
```
C##547>select * from students_in where branch is not null;

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
tauheed    547   cse   B         2500000
jagadeesh  530   cse   B         2200000
rehan      554   cse   A         2400000
neha       559   cse   B         3000000
navya      555   cse   A         2900000
```

Between operation :

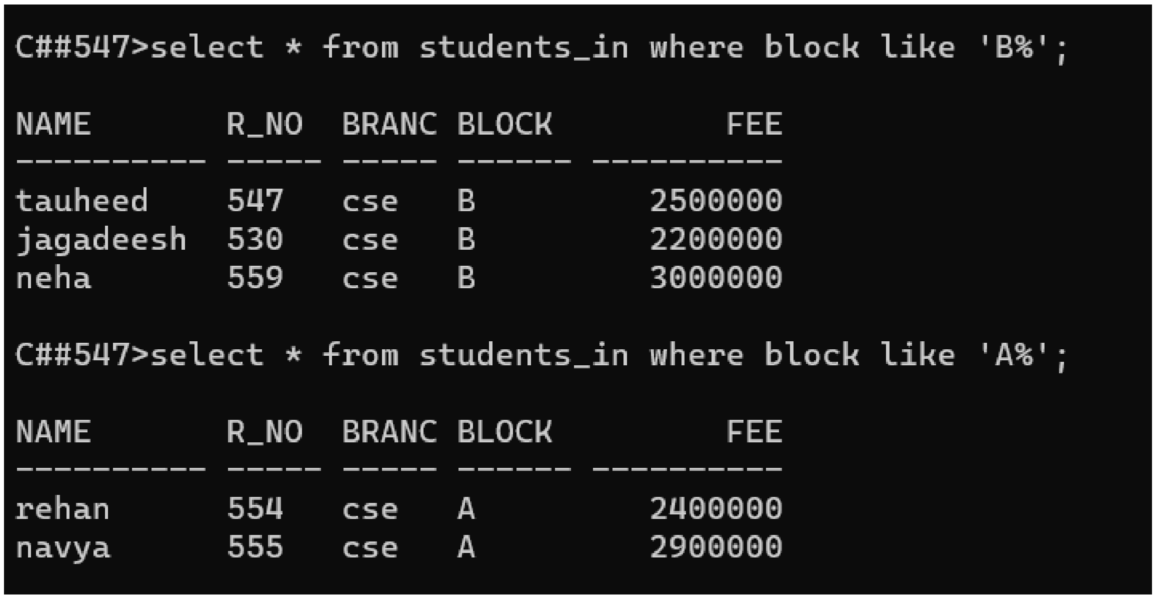


```
C##547>select * from students_in where fee between 2000000 and 3000000;

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
tauheed    547   cse   B         2500000
jagadeesh  530   cse   B         2200000
rehan      554   cse   A         2400000
neha       559   cse   B         3000000
navya      555   cse   A         2900000
naveen     555                   2100000
mani       549                   2900000
balaji     510                   2300000

8 rows selected.

C##547>select * from students_in where fee between 2500000 and 3500000;

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
tauheed    547   cse   B         2500000
neha       559   cse   B         3000000
navya      555   cse   A         2900000
mani       549                   2900000
```

Like operation:

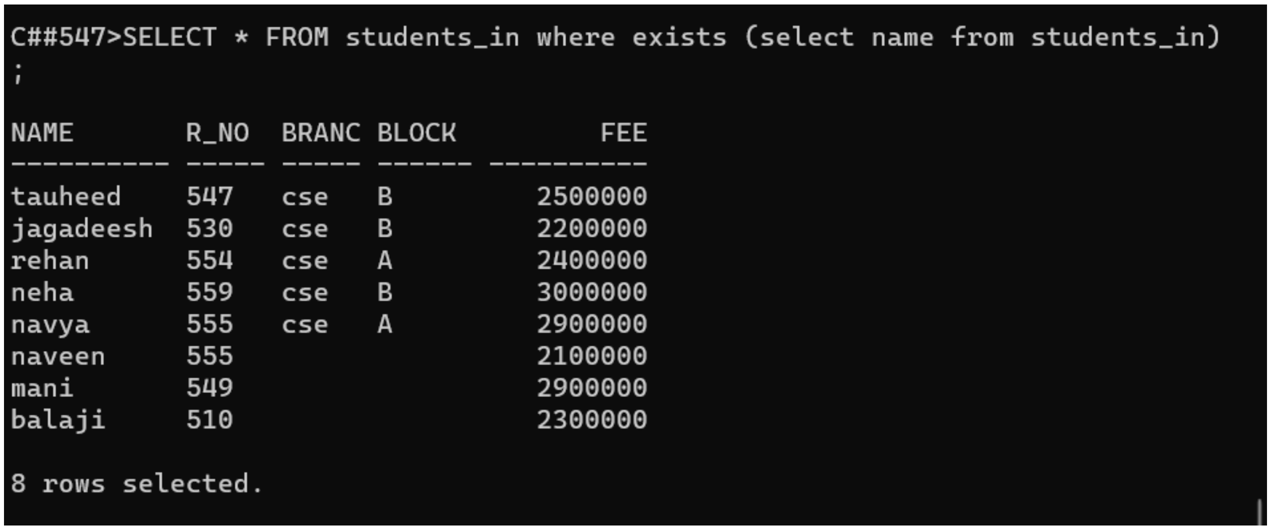


```
C##547>select * from students_in where branch like 'cse%';

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
tauheed    547   cse   B         2500000
jagadeesh  530   cse   B         2200000
rehan      554   cse   A         2400000
neha       559   cse   B         3000000
navya      555   cse   A         2900000
```



```
C##547>select * from students_in where block like 'B%';

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
tauheed    547   cse   B         2500000
jagadeesh  530   cse   B         2200000
neha       559   cse   B         3000000

C##547>select * from students_in where block like 'A%';

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
rehan      554   cse   A         2400000
navya      555   cse   A         2900000
```

Exists operation :

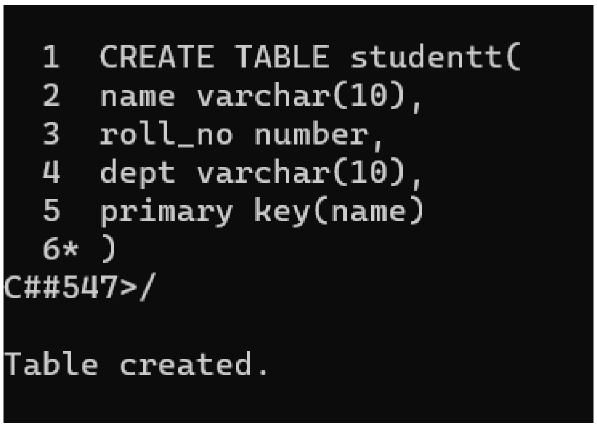


```
C##547>SELECT * FROM students_in where exists (select name from students_in)
;

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
tauheed    547   cse   B         2500000
jagadeesh  530   cse   B         2200000
rehan      554   cse   A         2400000
neha       559   cse   B         3000000
navya      555   cse   A         2900000
naveen     555                   2100000
mani       549                   2900000
balaji     510                   2300000

8 rows selected.
```

EXPERIMENT-7

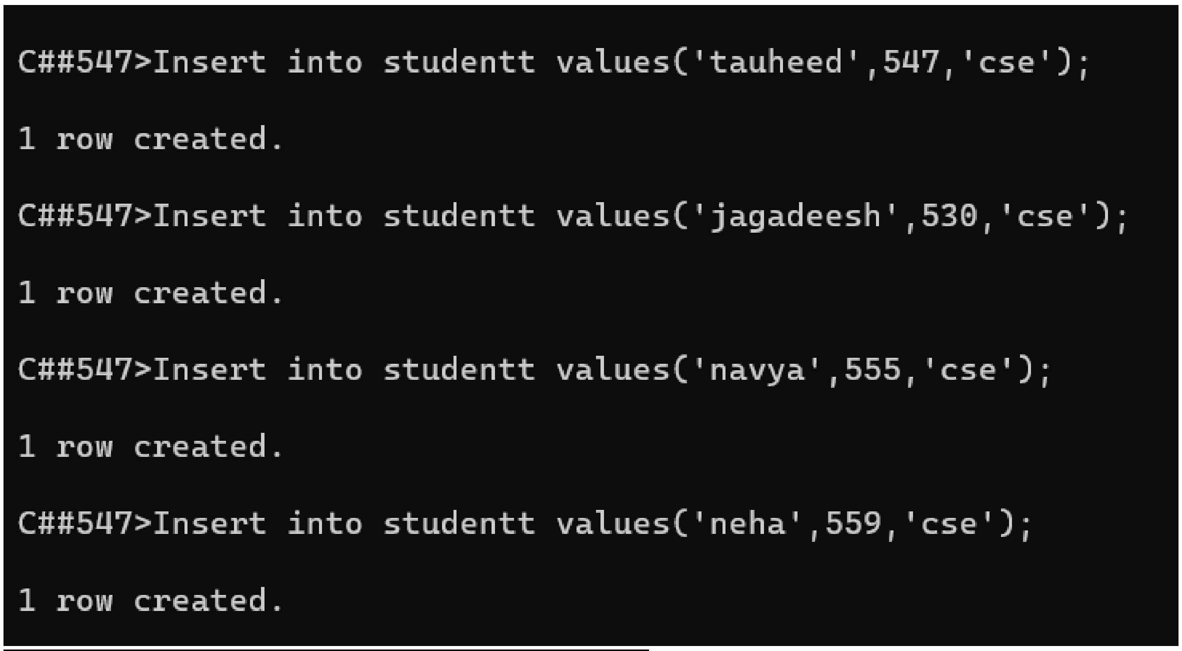
AIM: Write SQL queries to perform JOIN OPERATIONS (i.e. CONDITIONAL JOIN, EQUI JOIN,LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTfER JOIN)

CREATING TABLE student :

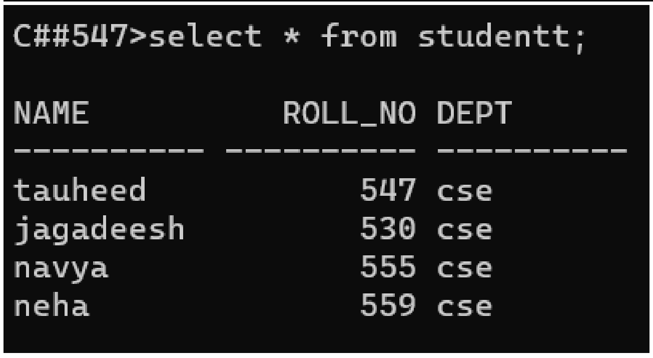


```
CREATE TABLE studentt(
name varchar(10),
roll_no number,
dept varchar(10),
primary key(name)
)
C##547>/

Table created.
```

Inserting tables into student table :

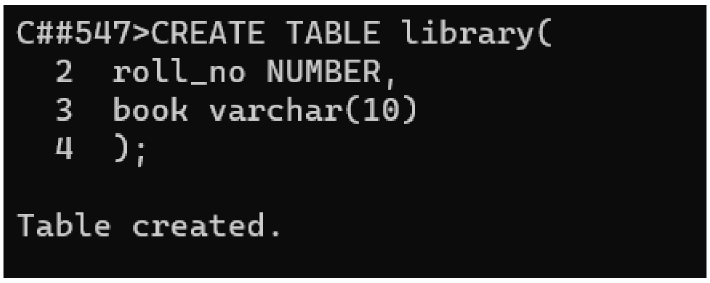


```
C##547>Insert into studentt values('tauheed',547,'cse');

1 row created.

C##547>Insert into studentt values('jagadeesh',530,'cse');

1 row created.

C##547>Insert into studentt values('navya',555,'cse');

1 row created.

C##547>Insert into studentt values('neha',559,'cse');

1 row created.
```



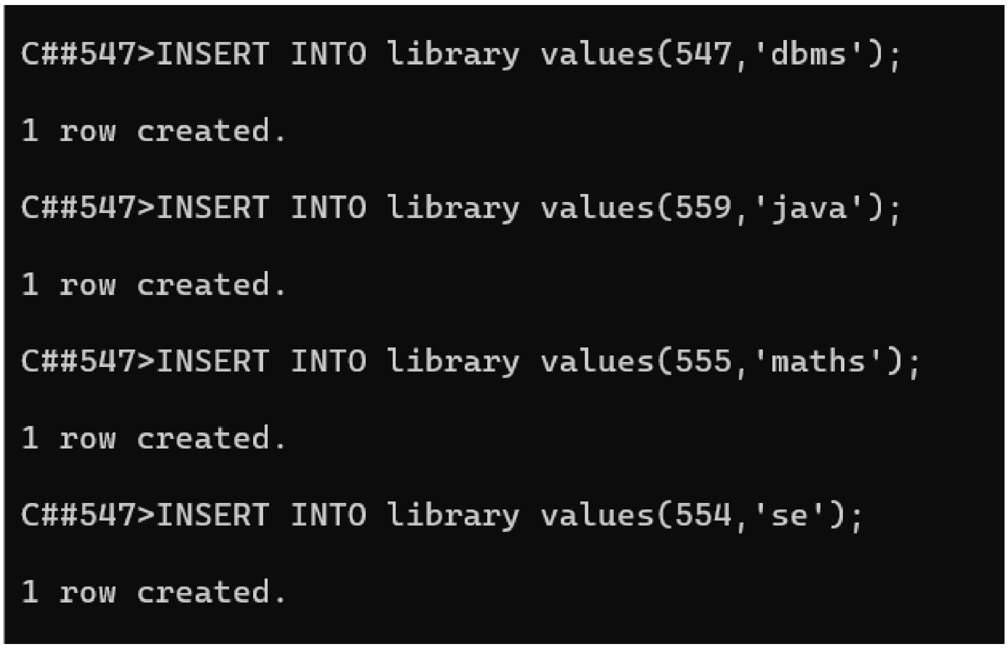
```
C##547>select * from studentt;

NAME          ROLL_NO DEPT
---------- ---------- ----------
tauheed           547 cse
jagadeesh         530 cse
navya             555 cse
neha              559 cse
```

Creating table Library :

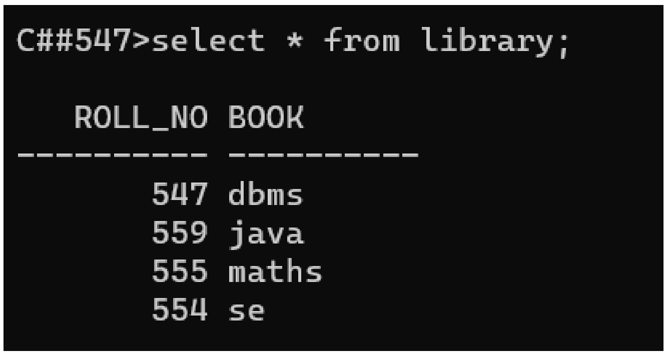


```
C##547>CREATE TABLE library(
roll_no NUMBER,
book varchar(10)
);

Table created.
```

Inserting values into library table :

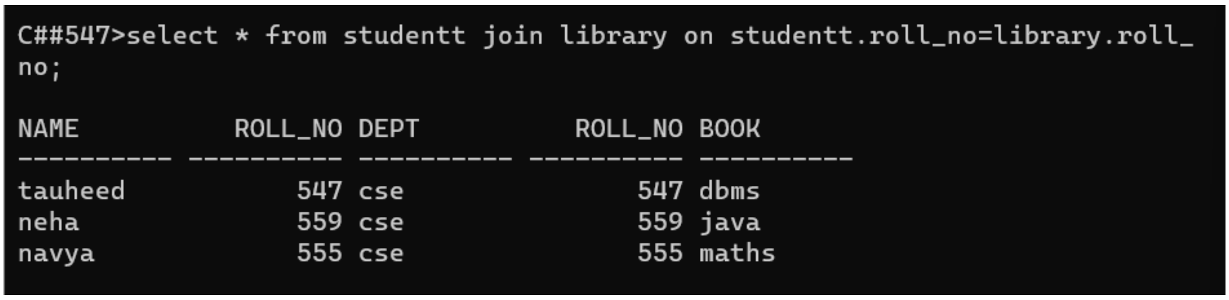


```
C##547>INSERT INTO library values(547,'dbms');

1 row created.

C##547>INSERT INTO library values(559,'java');

1 row created.

C##547>INSERT INTO library values(555,'maths');

1 row created.

C##547>INSERT INTO library values(554,'se');

1 row created.
```



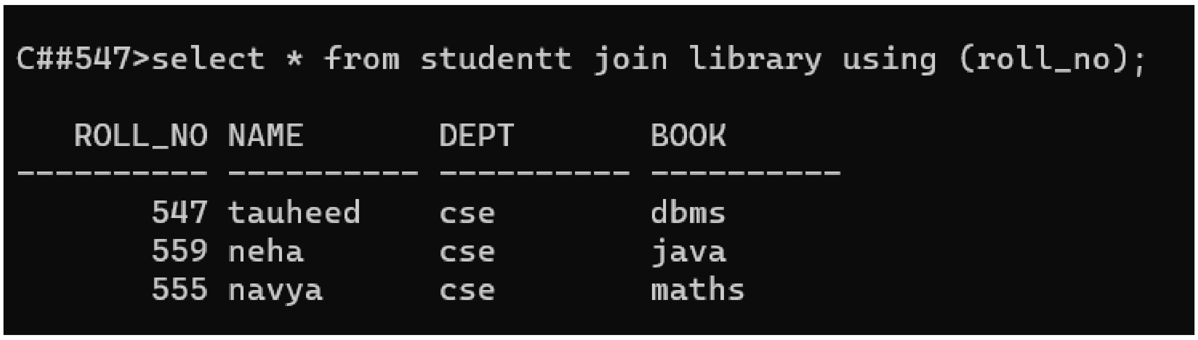
```
C##547>select * from library;

   ROLL_NO BOOK
---------- ----------
       547 dbms
       559 java
       555 maths
       554 se
```

CONDITIONAL JOIN :

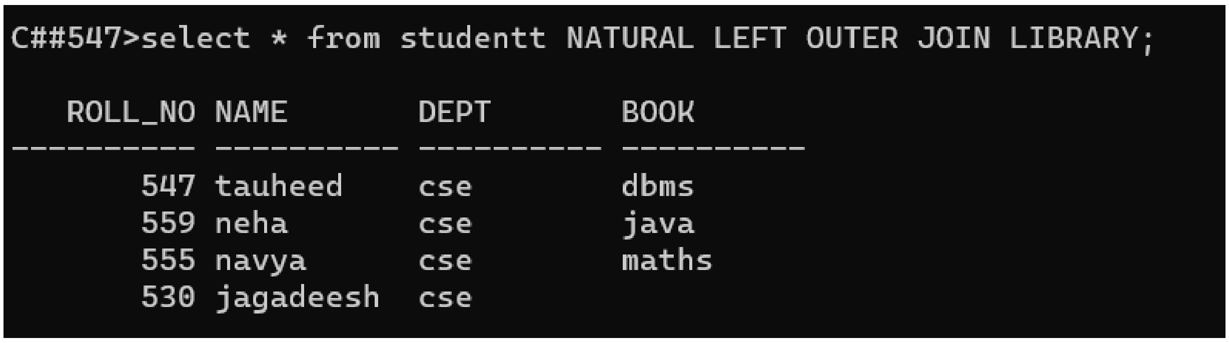


```
C##547>select * from studentt join library on studentt.roll_no=library.roll_
no;

NAME          ROLL_NO DEPT          ROLL_NO BOOK
---------- ---------- ---------- ---------- ----------
tauheed           547 cse               547 dbms
neha              559 cse               559 java
navya             555 cse               555 maths
```

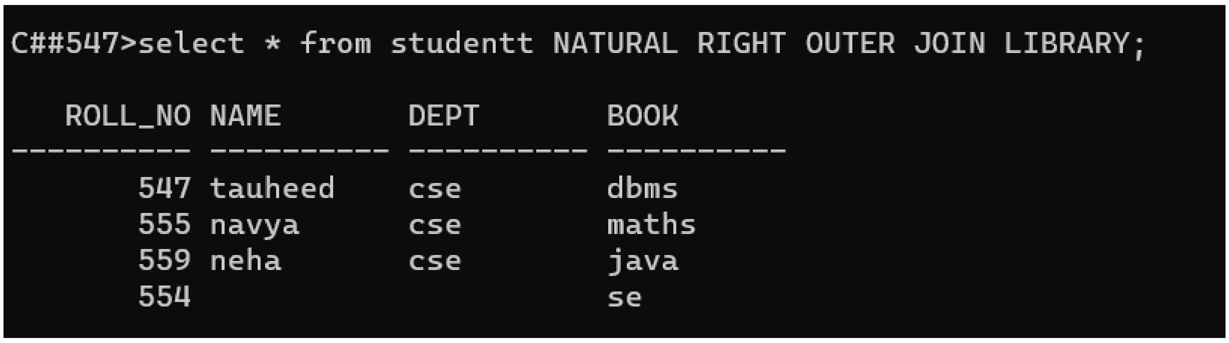
EQUI JOIN :


```
C##547>select * from studentt join library using (roll_no);

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
       547 tauheed    cse        dbms
       559 neha       cse        java
       555 navya      cse        maths
```

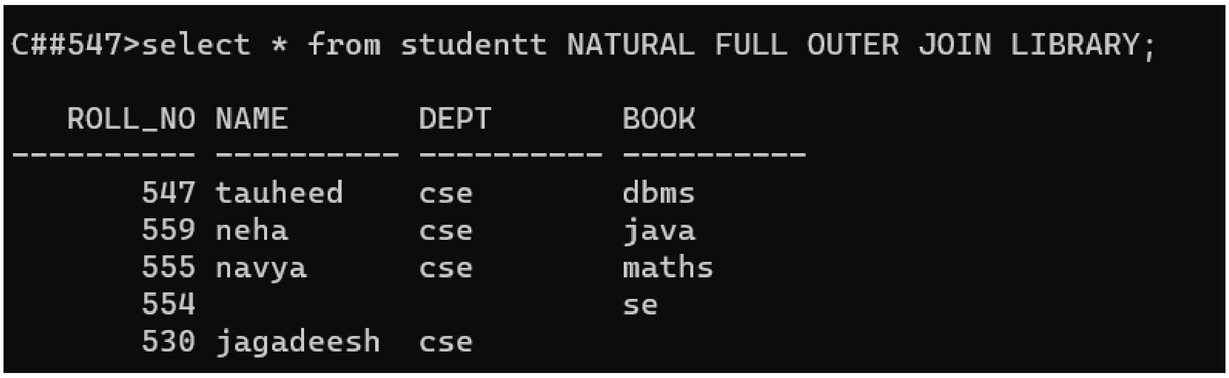
NATURAL LEFT OUTER JOIN :


```
C##547>select * from studentt NATURAL LEFT OUTER JOIN LIBRARY;

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
       547 tauheed    cse        dbms
       559 neha       cse        java
       555 navya      cse        maths
       530 jagadeesh  cse
```

NATURAL RIGHT OUTER JOIN :


```
C##547>select * from studentt NATURAL RIGHT OUTER JOIN LIBRARY;

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
       547 tauheed    cse        dbms
       555 navya      cse        maths
       559 neha       cse        java
       554                       se
```

NATURAL FULL OUTER JOIN :

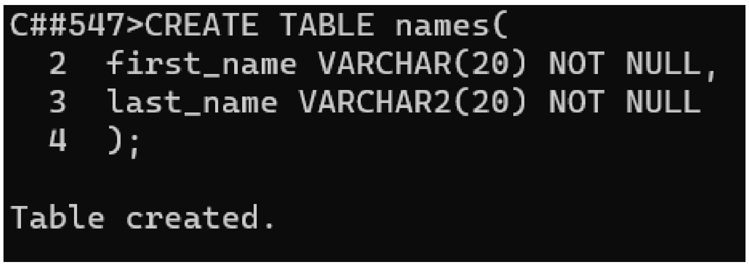

```
C##547>select * from studentt NATURAL FULL OUTER JOIN LIBRARY;

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
       547 tauheed    cse        dbms
       559 neha       cse        java
       555 navya      cse        maths
       554                       se
       530 jagadeesh  cse
```

EXPERIMENT-8

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

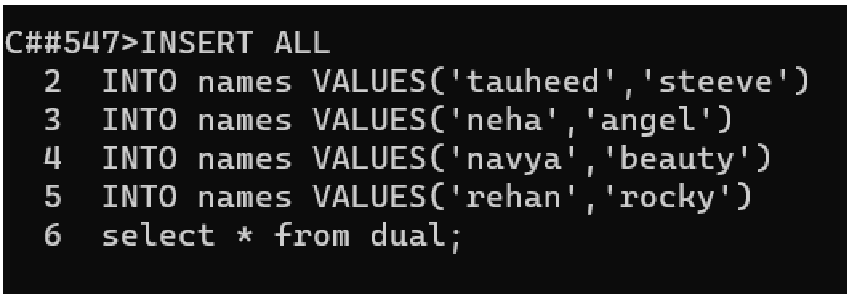
Built-in Functions

1. **Character Functions**
   I. **Case-conversion functions**
   II. **Character manipulation functions**
2. **Number Functions**
3. **DATE functions** CREATING TABLE :



```
C##547>CREATE TABLE names(
  2  first_name VARCHAR(20) NOT NULL,
  3  last_name VARCHAR2(20) NOT NULL
  4  );

Table created.
```

INSERTING VALUES :

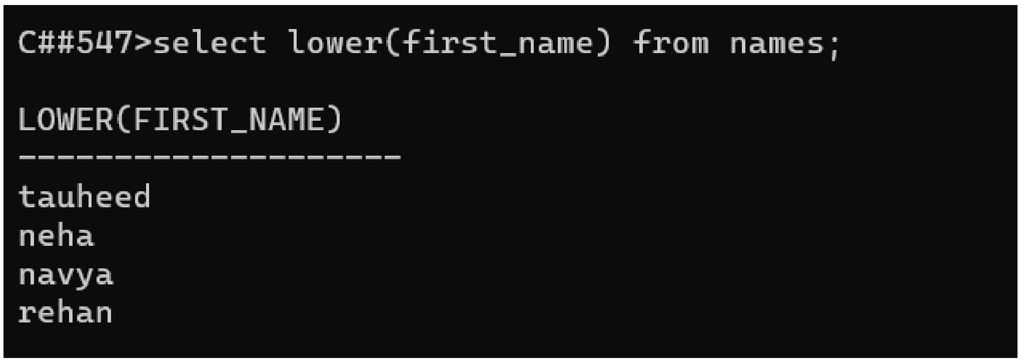


```
C##547>INSERT ALL
  2  INTO names VALUES('tauheed','steeve')
  3  INTO names VALUES('neha','angel')
  4  INTO names VALUES('navya','beauty')
  5  INTO names VALUES('rehan','rocky')
  6  select * from dual;
```

1. Character Functions

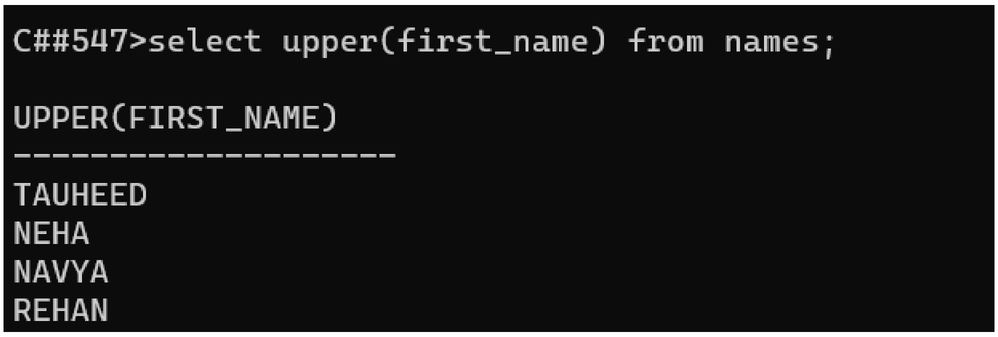
I. Case-conversion functions :

LOWER ();



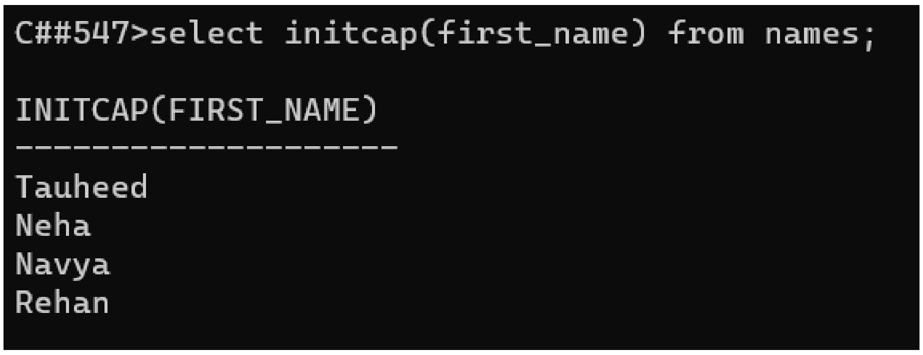
```
C##547>select lower(first_name) from names;

LOWER(FIRST_NAME)
--------------------
tauheed
neha
navya
rehan
```

UPPER();



```
C##547>select upper(first_name) from names;

UPPER(FIRST_NAME)
--------------------
TAUHEED
NEHA
NAVYA
REHAN
```

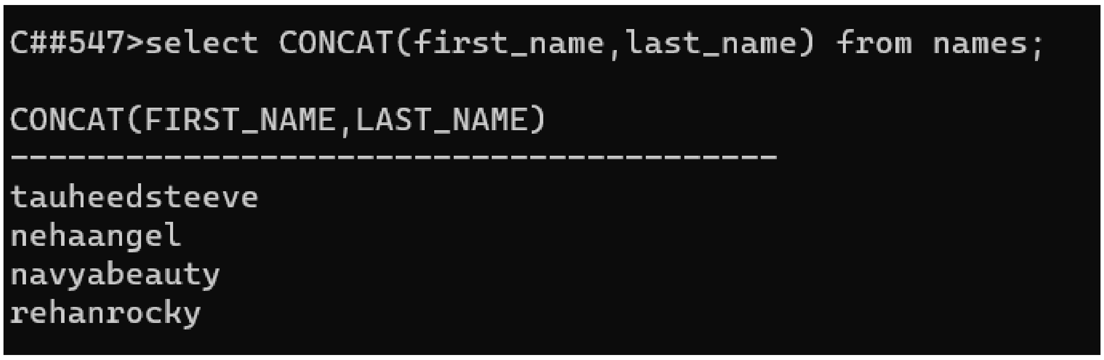
INITCAP();



```
C##547>select initcap(first_name) from names;

INITCAP(FIRST_NAME)
--------------------
Tauheed
Neha
Navya
Rehan
```

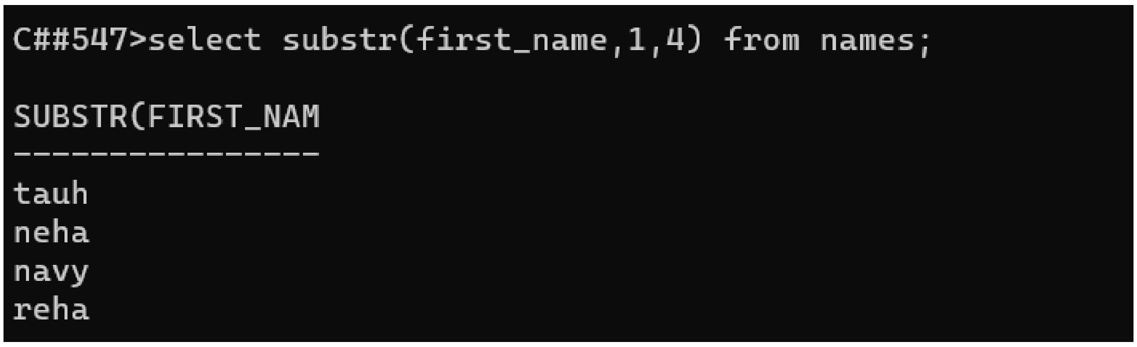
Character manipulation functions:

CONCAT():



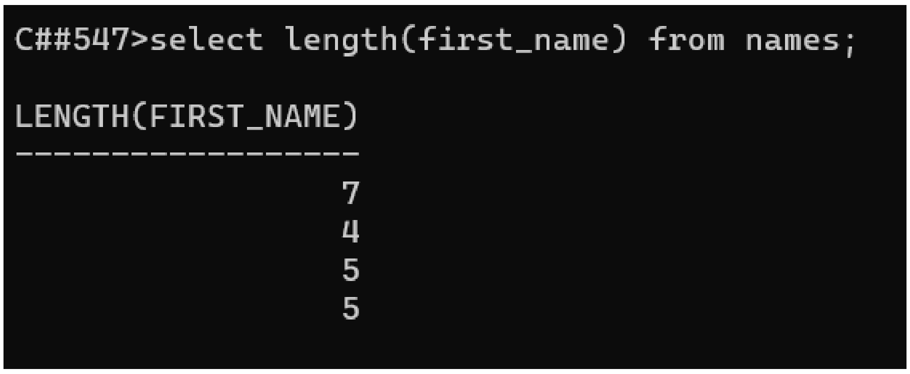
```
C##547>select CONCAT(first_name,last_name) from names;

CONCAT(FIRST_NAME,LAST_NAME)
----------------------------------------
tauheedsteeve
nehaangel
navyabeauty
rehanrocky
```

SUBSTR():



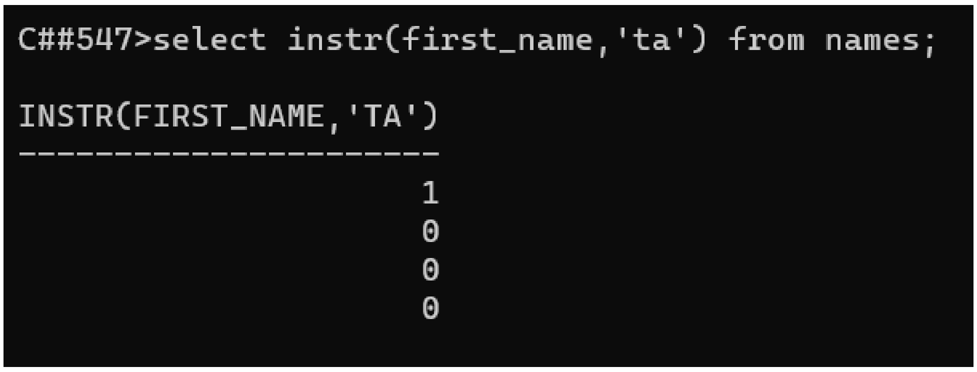
```
C##547>select substr(first_name,1,4) from names;

SUBSTR(FIRST_NAM
----------------
tauh
neha
navy
reha
```

LENGTH() :



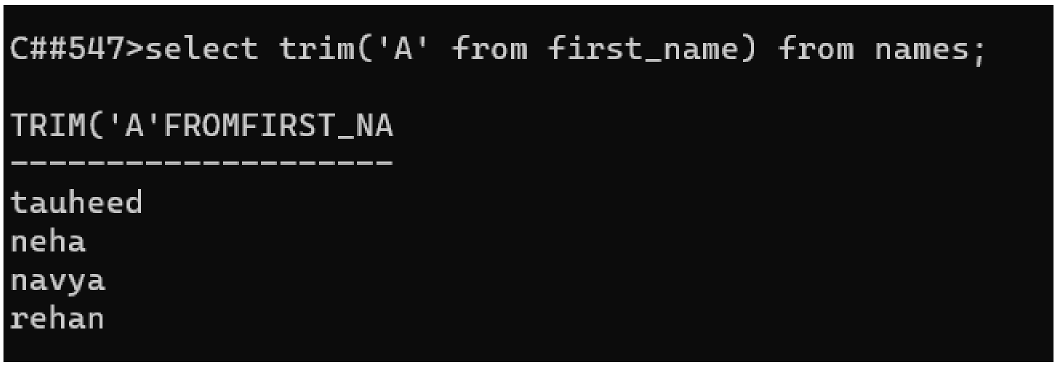
```
C##547>select length(first_name) from names;

LENGTH(FIRST_NAME)
------------------
                 7
                 4
                 5
                 5
```

INSTR() :



```
C##547>select instr(first_name,'ta') from names;

INSTR(FIRST_NAME,'TA')
----------------------
                     1
                     0
                     0
                     0
```

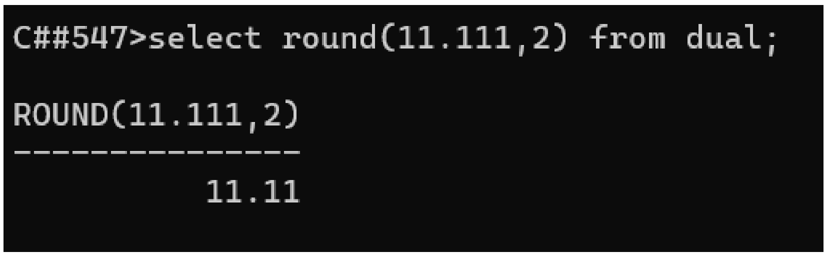
TRIM() :



```
C##547>select trim('A' from first_name) from names;

TRIM('A'FROMFIRST_NA
--------------------
tauheed
neha
navya
rehan
```

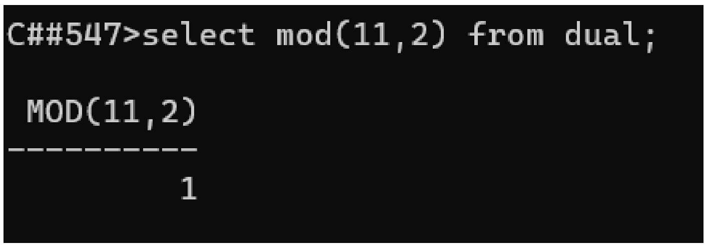
2. Number Functions :

ROUND() :



```
C##547>select round(11.111,2) from dual;

ROUND(11.111,2)
---------------
          11.11
```

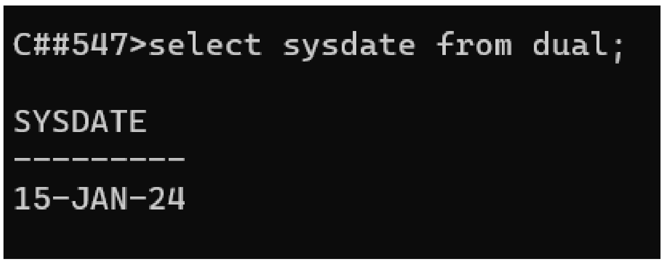
MOD() :



```
C##547>select mod(11,2) from dual;

 MOD(11,2)
----------
         1
```

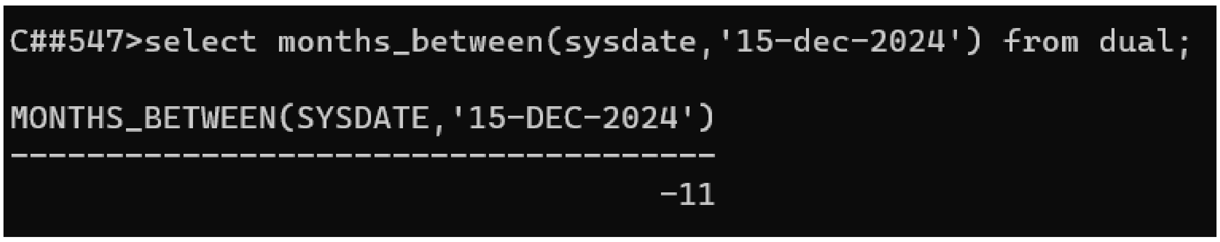
2.DATE functions :

SYSDATE()



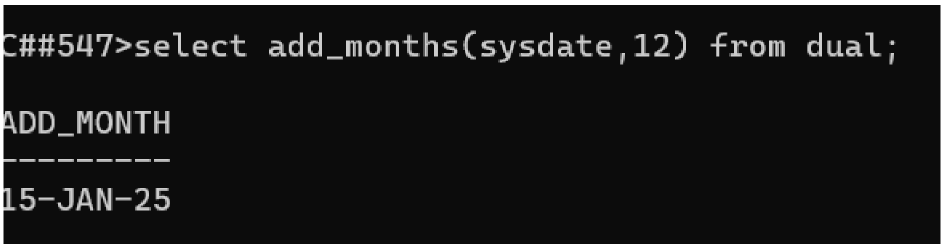
```
C##547>select sysdate from dual;

SYSDATE
---------
15-JAN-24
```

MONTHS-BETWEEN() :



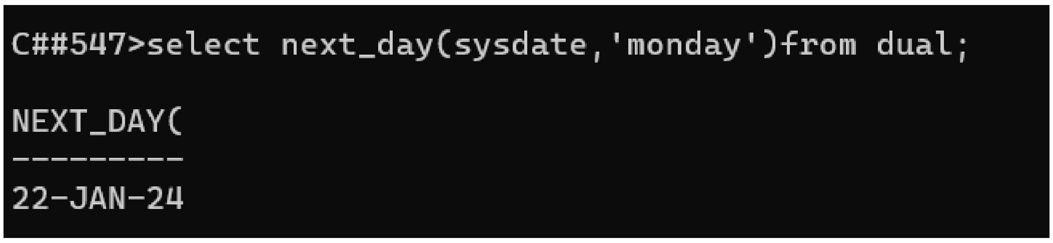
```
C##547>select months_between(sysdate,'15-dec-2024') from dual;

MONTHS_BETWEEN(SYSDATE,'15-DEC-2024')
-------------------------------------
                                  -11
```

ADD_MONTHS() :



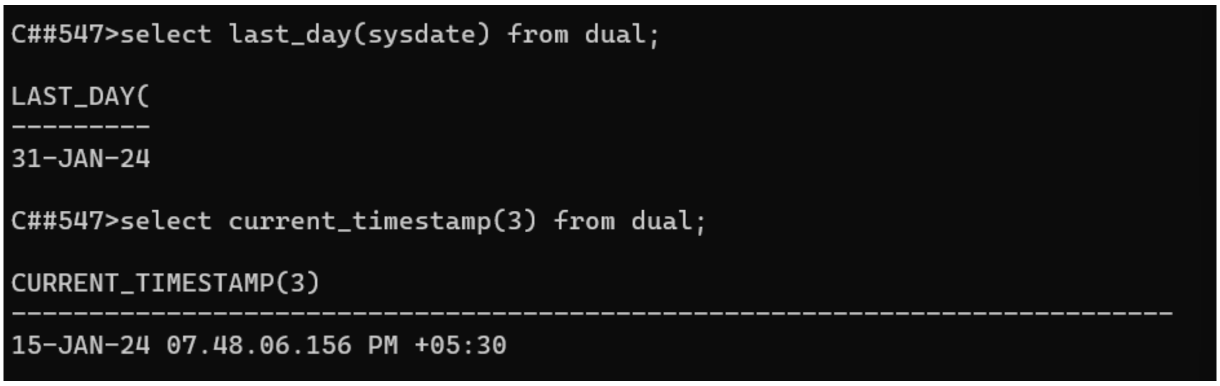
```
C##547>select add_months(sysdate,12) from dual;

ADD_MONTH
---------
15-JAN-25
```

NEXT_DAY():



```
C##547>select next_day(sysdate,'monday')from dual;

NEXT_DAY(
---------
22-JAN-24
```

LAST_DAY() :



```
C##547>select last_day(sysdate) from dual;

LAST_DAY(
---------
31-JAN-24

C##547>select current_timestamp(3) from dual;

CURRENT_TIMESTAMP(3)
---------------------------------------------------------------------------
15-JAN-24 07.48.06.156 PM +05:30
```

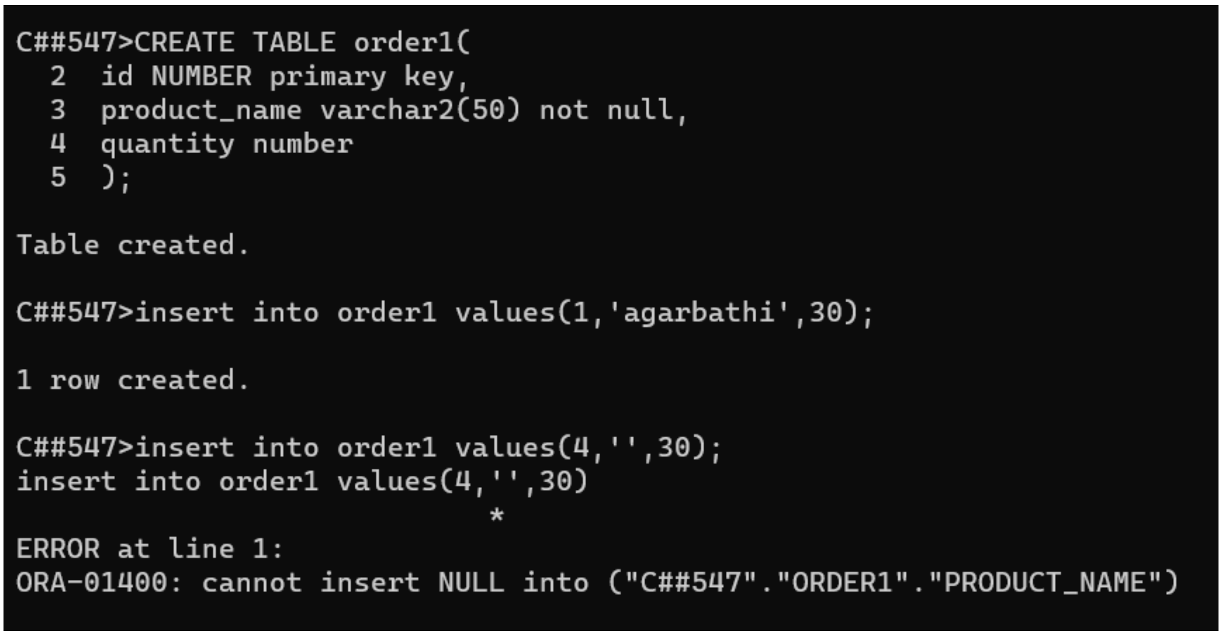
EXPERIMENT-9

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

Types of SQL Constraints.

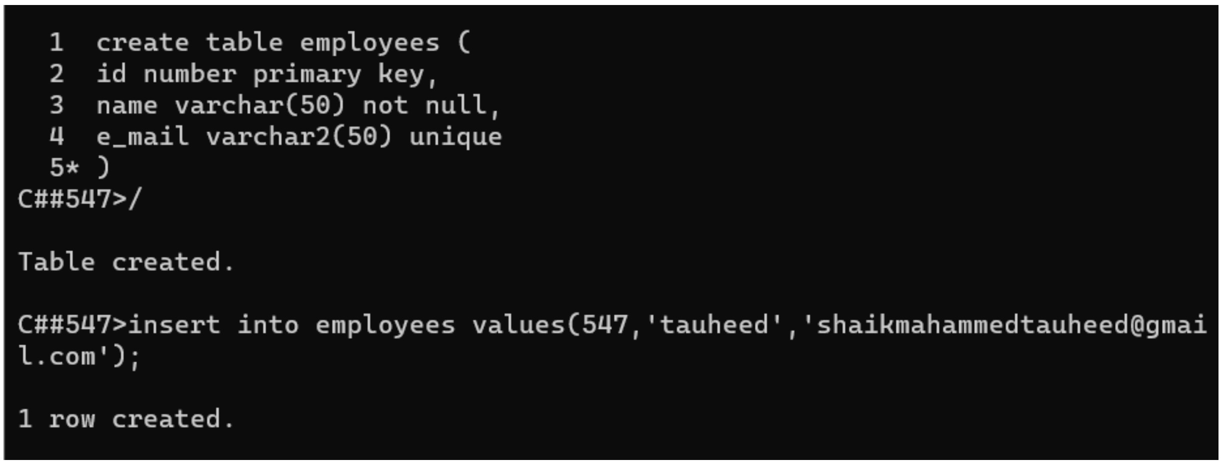
1. NOT NULL - Ensures that a column cannot have a NULL value
2. UNIQUE - Ensures that all values in a column are different
3. PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely I identifies each row in a table
4. FOREIGN KEY - Uniquely identifies a row/record in another table
5. CHECK - Ensures that all values in a column satisfies a specific condition
6. DEFAULT - Sets a default value for a column when no value is specified

1.NOT NULL Constraint Example:



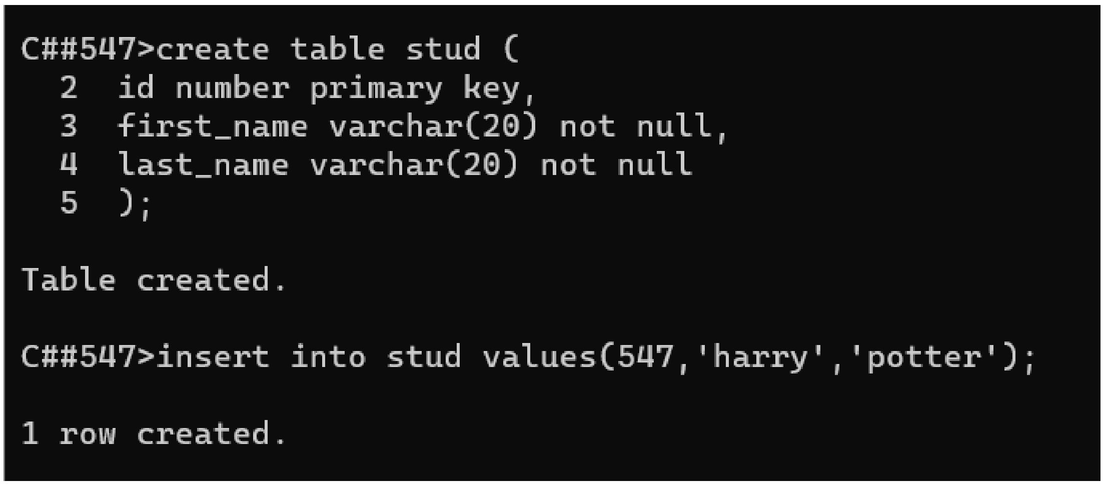
```
C##547>CREATE TABLE order1(
  2  id NUMBER primary key,
  3  product_name varchar2(50) not null,
  4  quantity number
  5  );

Table created.

C##547>insert into order1 values(1,'agarbathi',30);

1 row created.

C##547>insert into order1 values(4,'',30);
insert into order1 values(4,'',30)
                            *
ERROR at line 1:
ORA-01400: cannot insert NULL into ("C##547"."ORDER1"."PRODUCT_NAME")
```

2.UNIQUE CONSTRAINT Example:



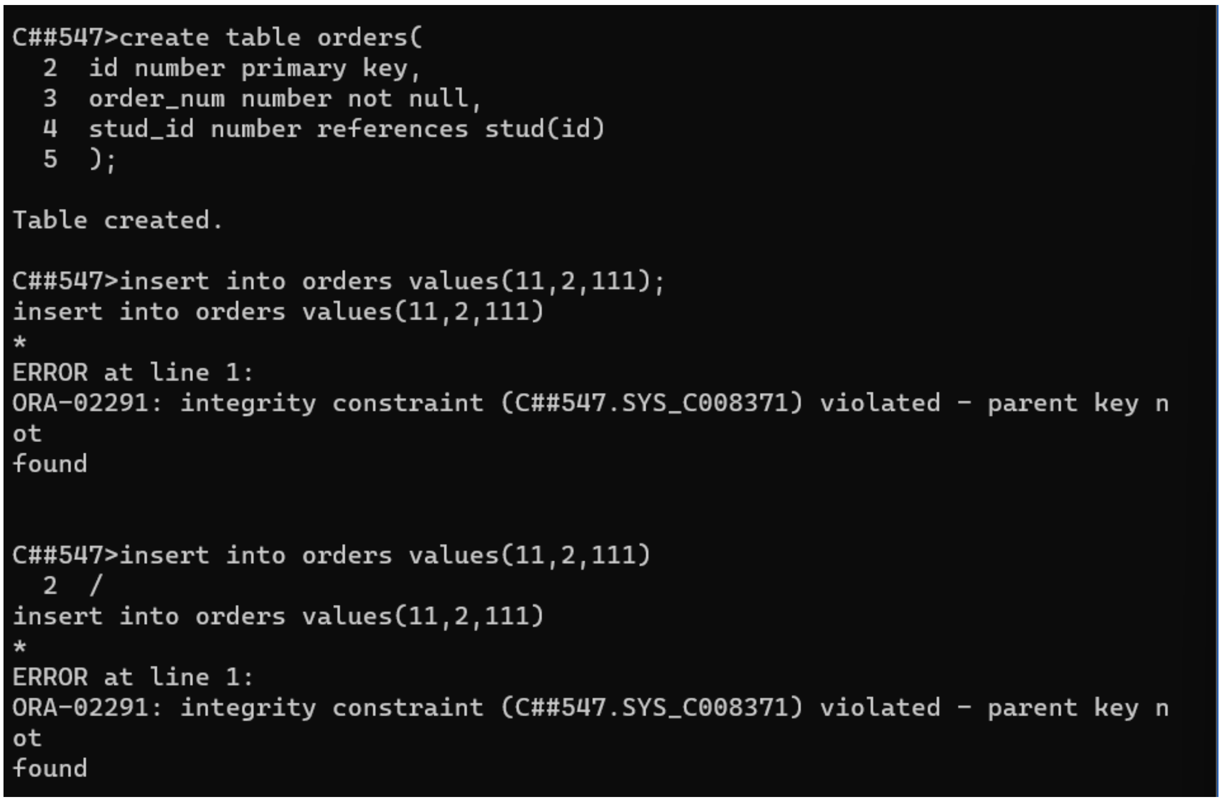
```
  1  create table employees (
  2  id number primary key,
  3  name varchar(50) not null,
  4  e_mail varchar2(50) unique
  5* )
C##547>/

Table created.

C##547>insert into employees values(547,'tauheed','shaikmahammedtauheed@gmai
l.com');

1 row created.
```

3.PRIMARY KEY CONSTRAINT Example:



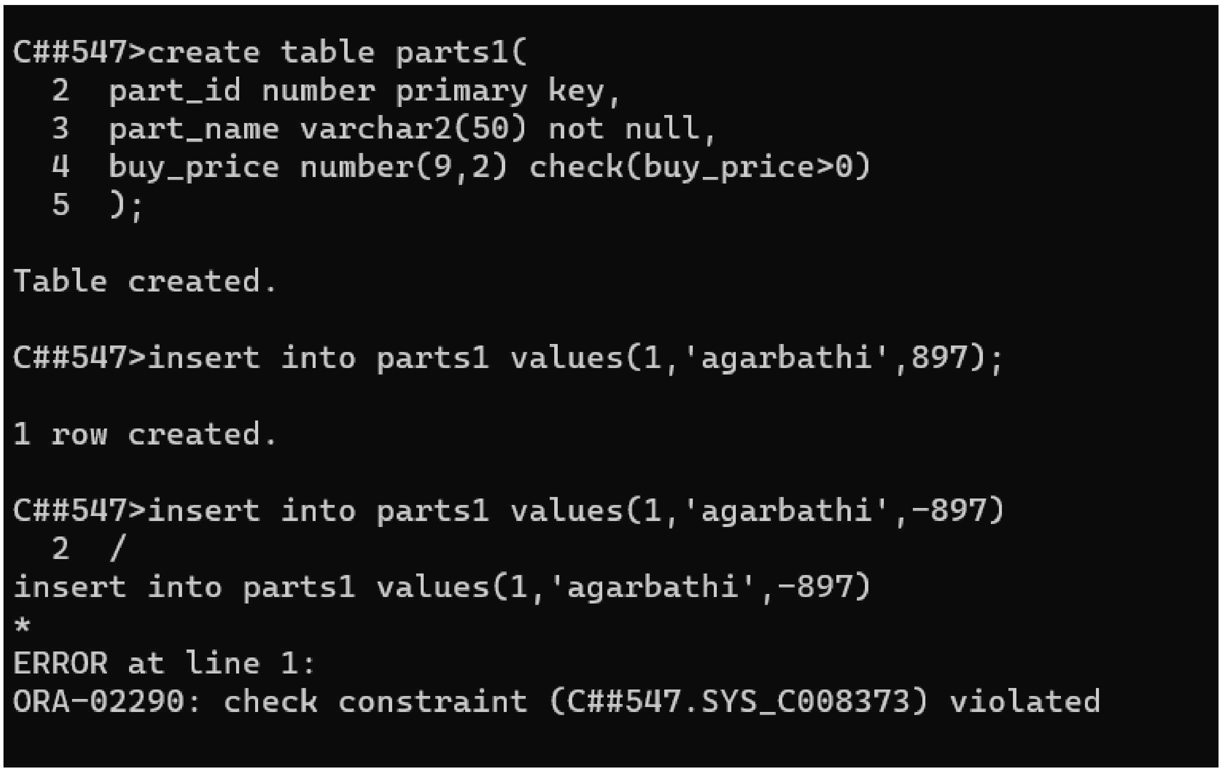
```
C##547>create table stud (
  2  id number primary key,
  3  first_name varchar(20) not null,
  4  last_name varchar(20) not null
  5  );

Table created.

C##547>insert into stud values(547,'harry','potter');

1 row created.
```

4.FORIEGN KEY CONSTRAINTS Example:



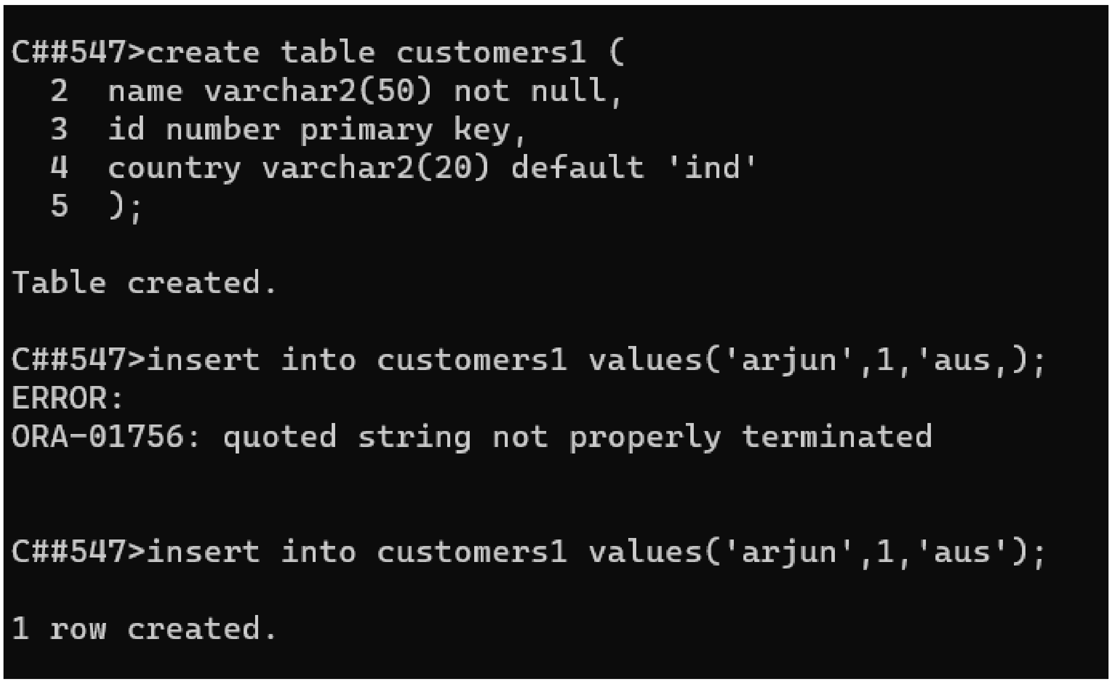
```
C##547>create table orders(
  2  id number primary key,
  3  order_num number not null,
  4  stud_id number references stud(id)
  5  );

Table created.

C##547>insert into orders values(11,2,111);
insert into orders values(11,2,111)
*
ERROR at line 1:
ORA-02291: integrity constraint (C##547.SYS_C008371) violated - parent key n
ot
found


C##547>insert into orders values(11,2,111)
  2  /
insert into orders values(11,2,111)
*
ERROR at line 1:
ORA-02291: integrity constraint (C##547.SYS_C008371) violated - parent key n
ot
found
```

5.CHECK CONSTRAINTS Example:

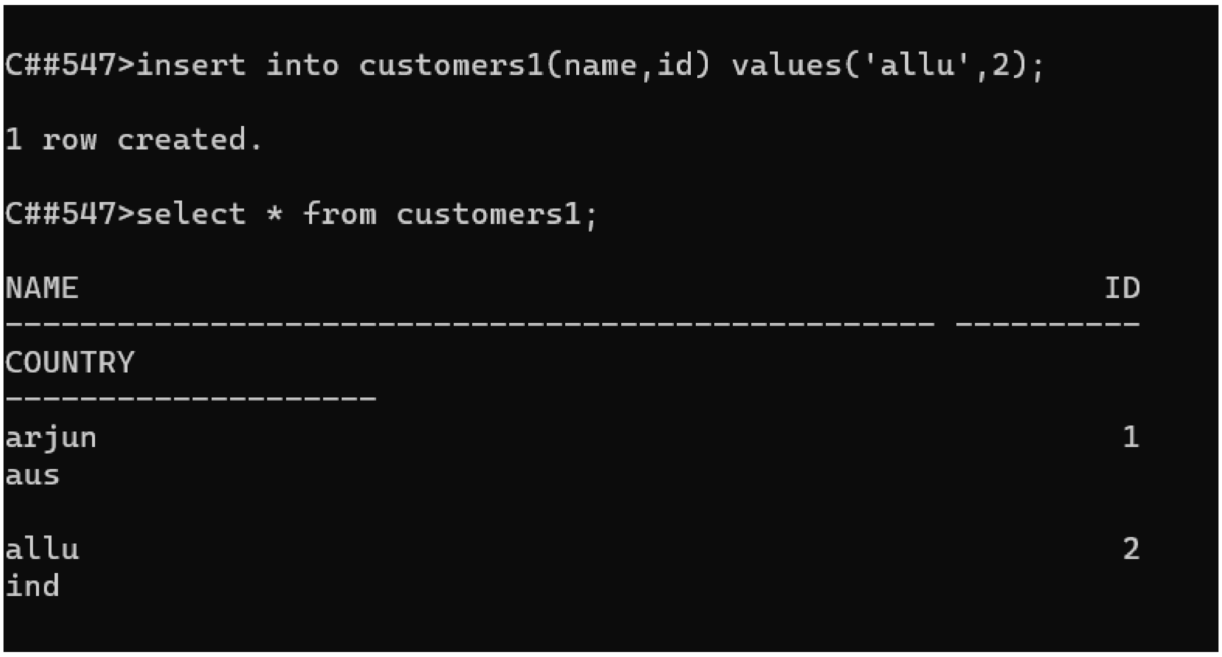


```
C##547>create table parts1(
  2  part_id number primary key,
  3  part_name varchar2(50) not null,
  4  buy_price number(9,2) check(buy_price>0)
  5  );

Table created.

C##547>insert into parts1 values(1,'agarbathi',897);

1 row created.

C##547>insert into parts1 values(1,'agarbathi',-897)
  2  /
insert into parts1 values(1,'agarbathi',-897)
*
ERROR at line 1:
ORA-02290: check constraint (C##547.SYS_C008373) violated
```

6.DEFAULT CONSTRAINTS Example:



```
C##547>create table customers1 (
  2  name varchar2(50) not null,
  3  id number primary key,
  4  country varchar2(20) default 'ind'
  5  );

Table created.

C##547>insert into customers1 values('arjun',1,'aus,);
ERROR:
ORA-01756: quoted string not properly terminated


C##547>insert into customers1 values('arjun',1,'aus');

1 row created.
```

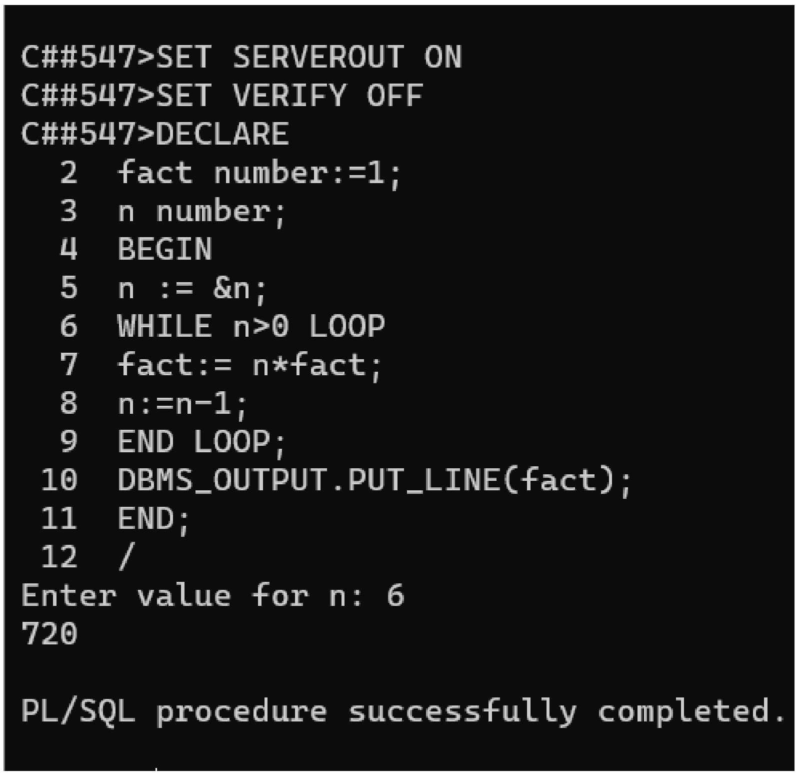


```
C##547>insert into customers1(name,id) values('allu',2);

1 row created.

C##547>select * from customers1;

NAME                                                       ID
-------------------------------------------------- ----------
COUNTRY
--------------------
arjun                                                       1
aus

allu                                                        2
ind
```

EXPERIMENT-10

AIM: To write a PL/SQL program for calculating the factorial of a given number.

Source code:



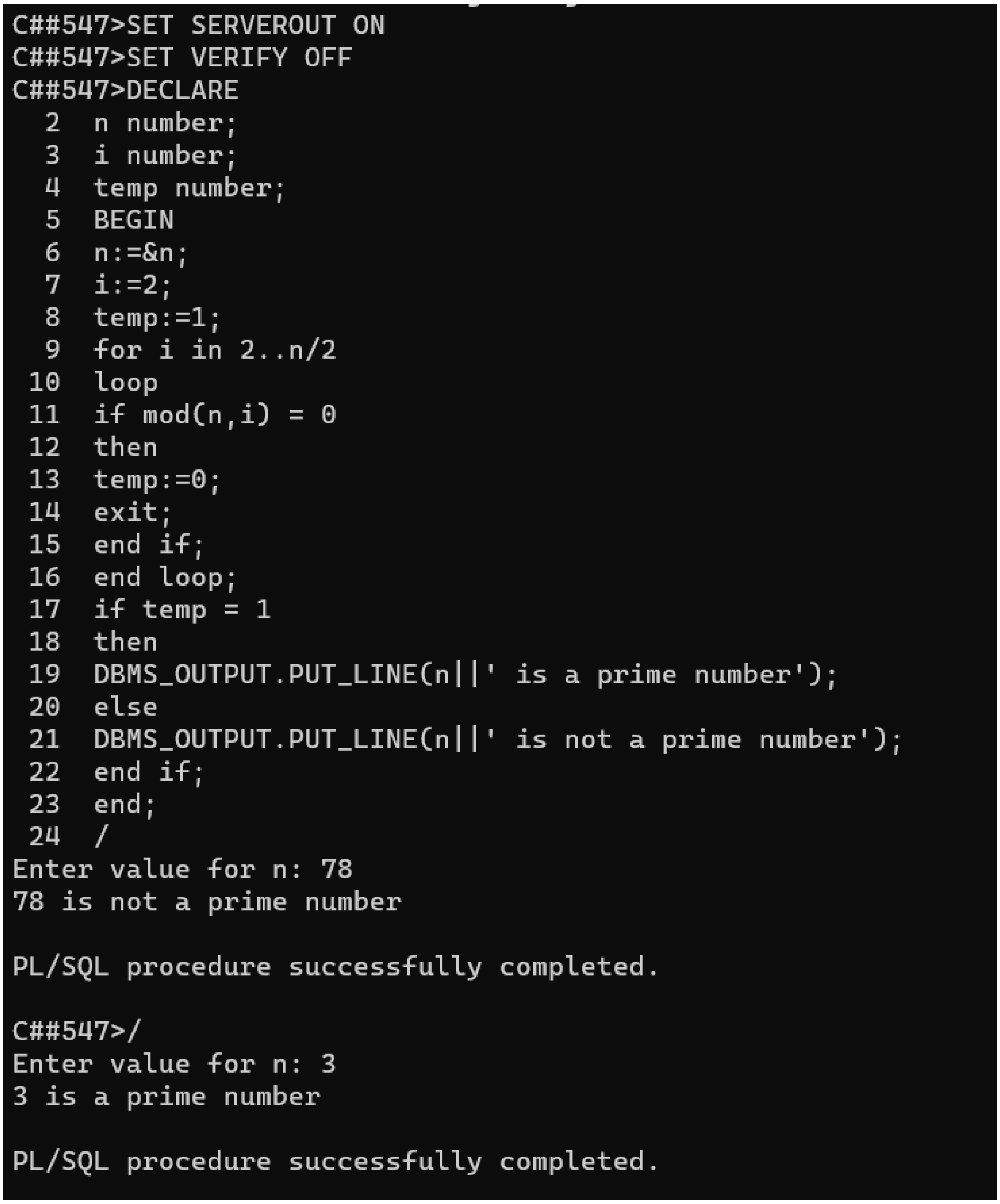
```
C##547>SET SERVEROUT ON
C##547>SET VERIFY OFF
C##547>DECLARE
fact number:=1;
n number;
BEGIN
n := &n;
WHILE n>0 LOOP
fact:= n*fact;
n:=n-1;
END LOOP;
DBMS_OUTPUT.PUT_LINE(fact);
END;
/
Enter value for n: 6
720

PL/SQL procedure successfully completed.
```

Conclusion : The pl/sql program is successfully executed.

EXPERIMENT-11

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

SOURCE CODE:



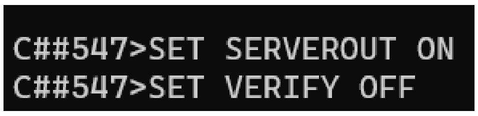
```
C##547>SET SERVEROUT ON
C##547>SET VERIFY OFF
C##547>DECLARE
n number;
i number;
temp number;
BEGIN
n:=&n;
i:=2;
temp:=1;
for i in 2..n/2
loop
if mod(n,i) = 0
then
temp:=0;
exit;
end if;
end loop;
if temp = 1
then
DBMS_OUTPUT.PUT_LINE(n||' is a prime number');
else
DBMS_OUTPUT.PUT_LINE(n||' is not a prime number');
end if;
end;
/
Enter value for n: 78
78 is not a prime number

PL/SQL procedure successfully completed.

C##547>/
Enter value for n: 3
3 is a prime number

PL/SQL procedure successfully completed.
```

CONCLUSION: The pl/sql program is successfully executed.

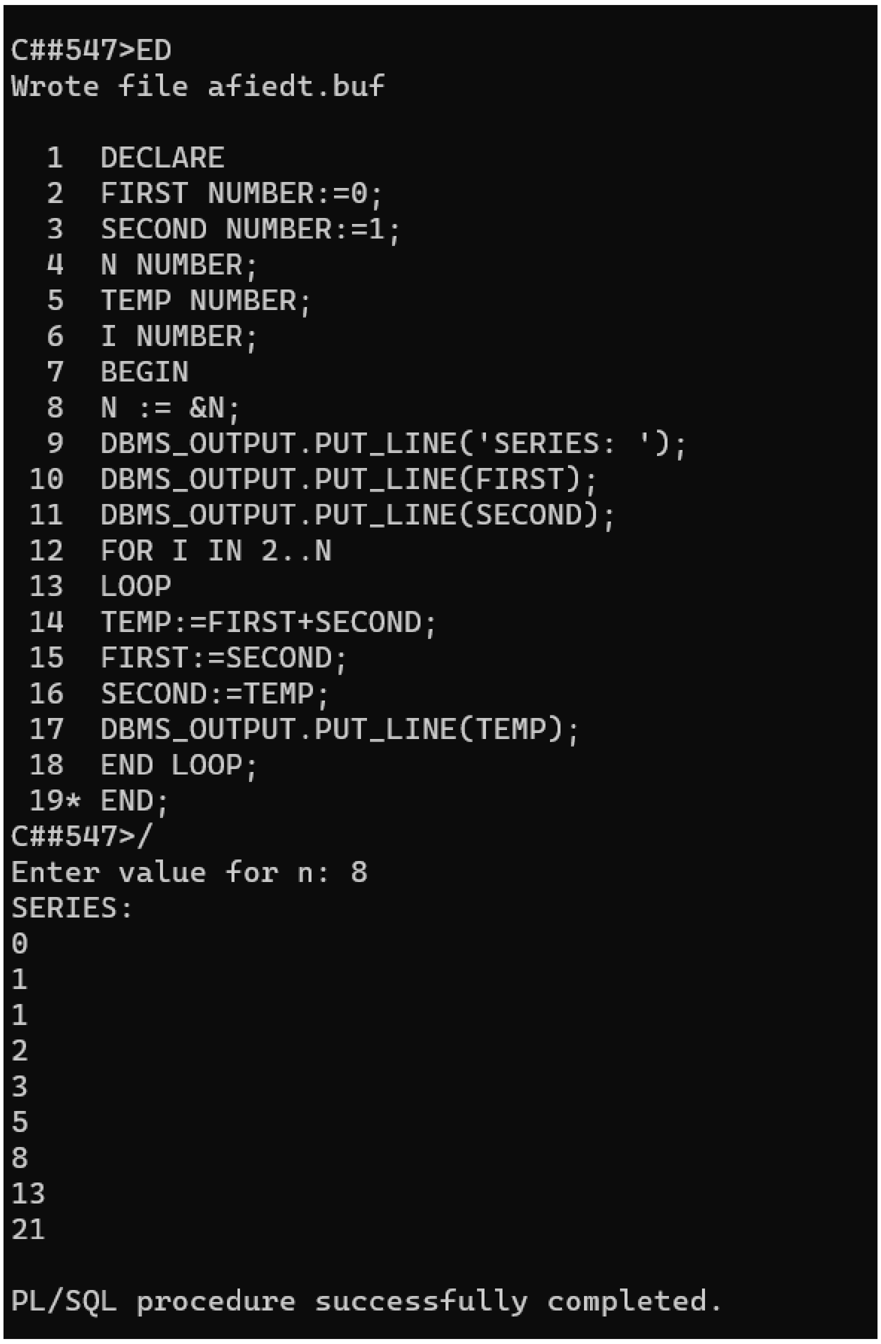
EXPERIMENT-12

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

SOURCE CODE:



```
C##547>SET SERVEROUT ON
C##547>SET VERIFY OFF
```



```
C##547>ED
Wrote file afiedt.buf

DECLARE
FIRST NUMBER:=0;
SECOND NUMBER:=1;
N NUMBER;
TEMP NUMBER;
I NUMBER;
BEGIN
N := &N;
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;
C##547>/
Enter value for n: 8
SERIES:
0
1
1
2
3
5
8
13
21

PL/SQL procedure successfully completed.
```

CONCLUSION: The pl/sql program is successfully executed.

EXPERIMENT-13

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

AIM:

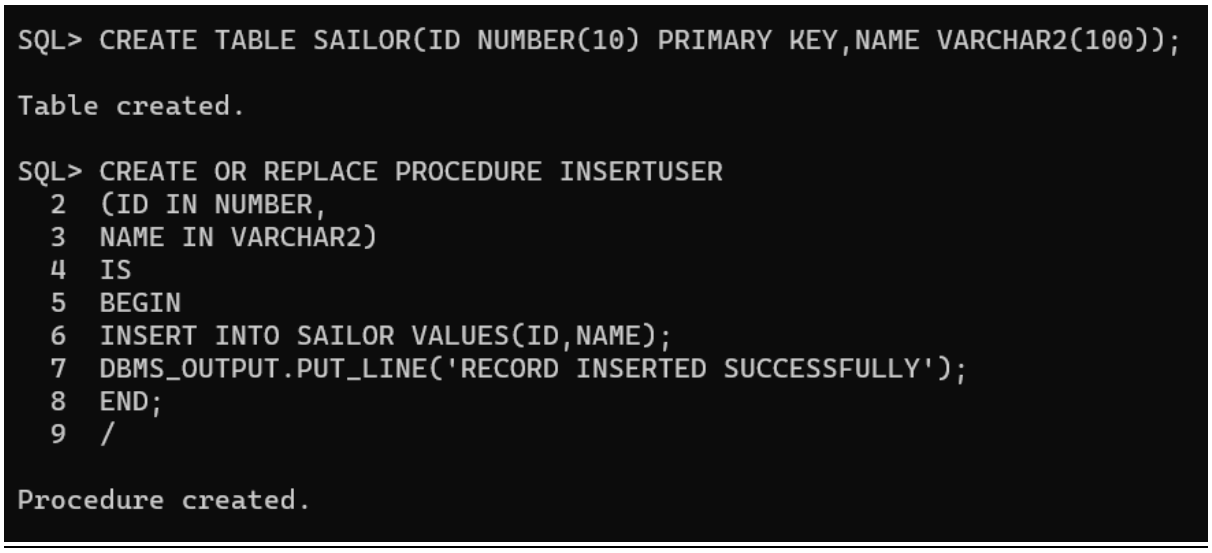
Write PL/SQL program to implement Stored Procedure on table.

PL/SQL Procedure:

The PL/SQL stored procedure or simply a procedure is a PL/SQL block which performs one or more specific tasks. It is just like procedures in other programming languages.

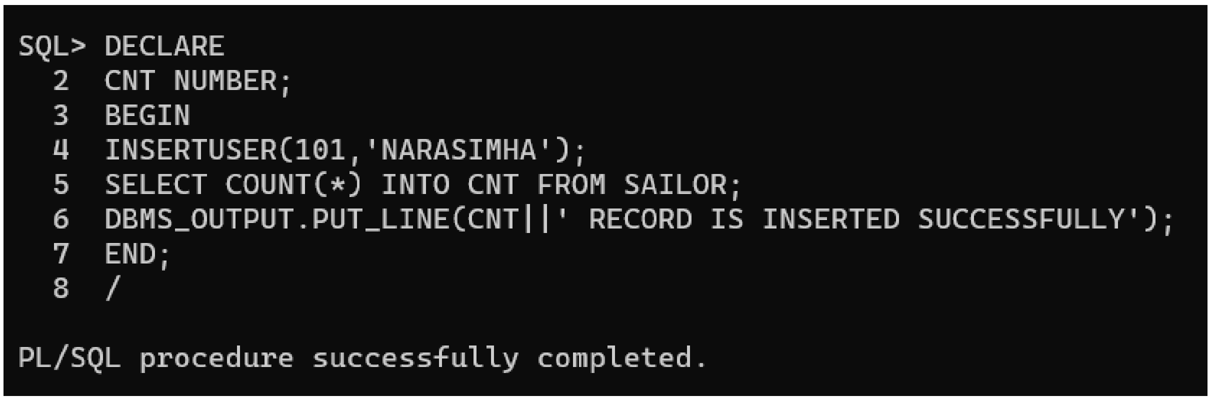
The procedure contains a header and a body.

EXAMPLE :1



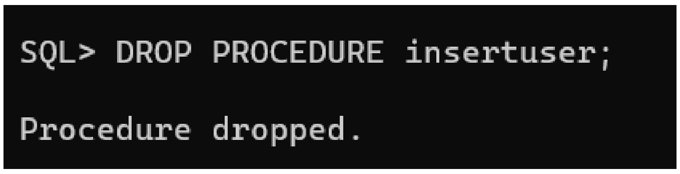
```
SQL> CREATE TABLE SAILOR(ID NUMBER(10) PRIMARY KEY,NAME VARCHAR2(100));

Table created.

SQL> CREATE OR REPLACE PROCEDURE INSERTUSER
  2  (ID IN NUMBER,
  3  NAME IN VARCHAR2)
  4  IS
  5  BEGIN
  6  INSERT INTO SAILOR VALUES(ID,NAME);
  7  DBMS_OUTPUT.PUT_LINE('RECORD INSERTED SUCCESSFULLY');
  8  END;
  9  /

Procedure created.
```

Execution Procedure:



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

PL/SQL procedure successfully completed.
```

DROP PROCEDURE:



```
SQL> DROP PROCEDURE insertuser;

Procedure dropped.
```

CONCLUSION :

The pl/sql programs is successfully executed.

EXPERIMENT-14

AIM:

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

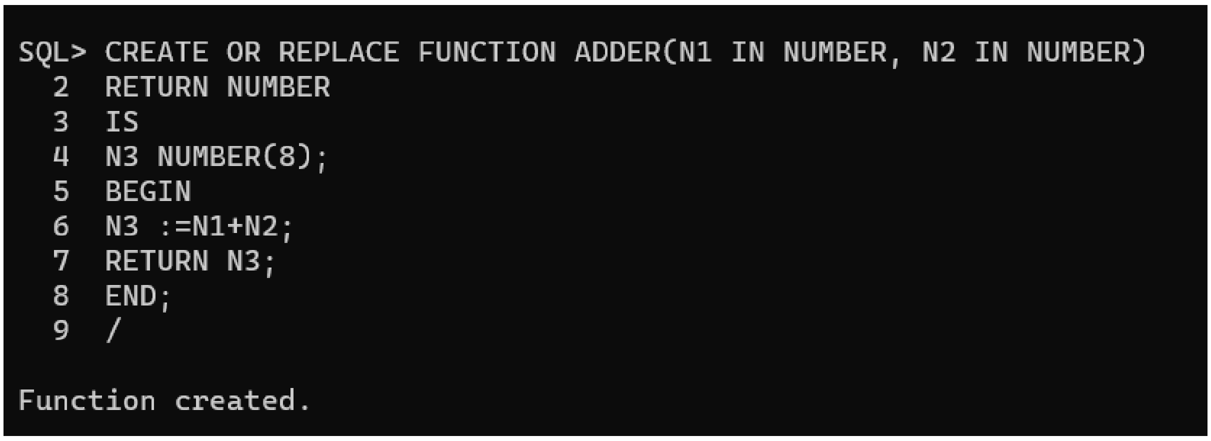
PL/SQL Function:

The PL/SQL Function is very similar to PL/SQL Procedure. The main difference between

procedure and a function is, a function must always return a value, and on the other hand a

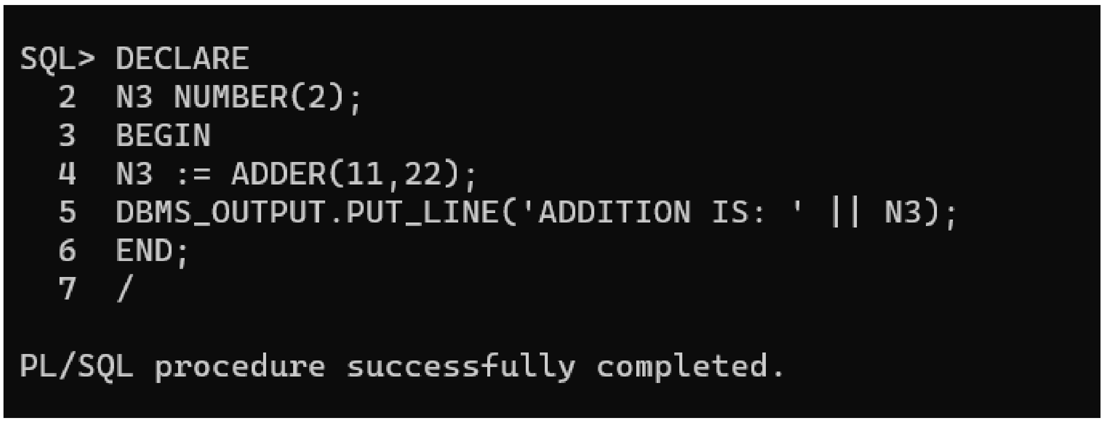
procedure may or may not return a value. Except this, all the other things of PL/SQL procedure

are true for PL/SQL function too.



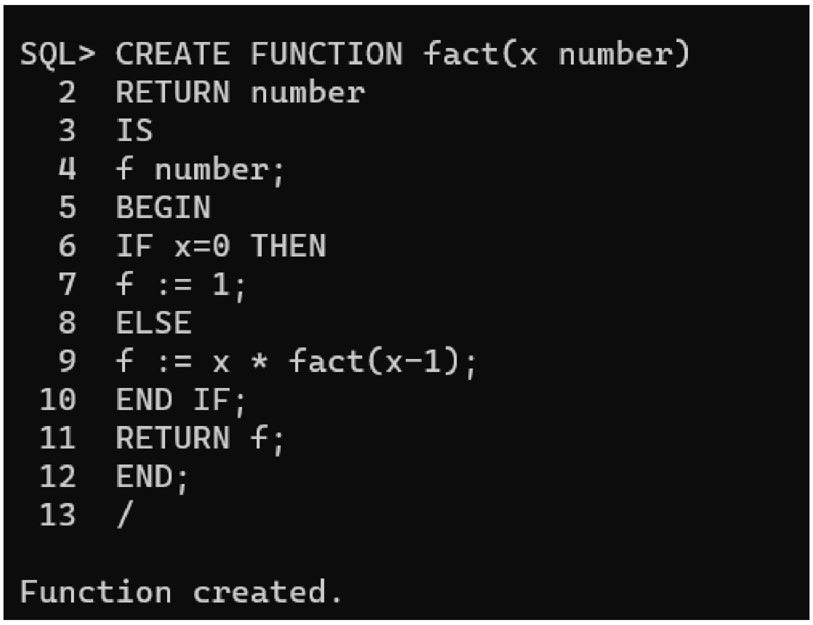
```
SQL> CREATE OR REPLACE FUNCTION ADDER(N1 IN NUMBER, N2 IN NUMBER)
     RETURN NUMBER
     IS
     N3 NUMBER(8);
     BEGIN
     N3 :=N1+N2;
     RETURN N3;
     END;
     /

Function created.
```

Execution Procedure:



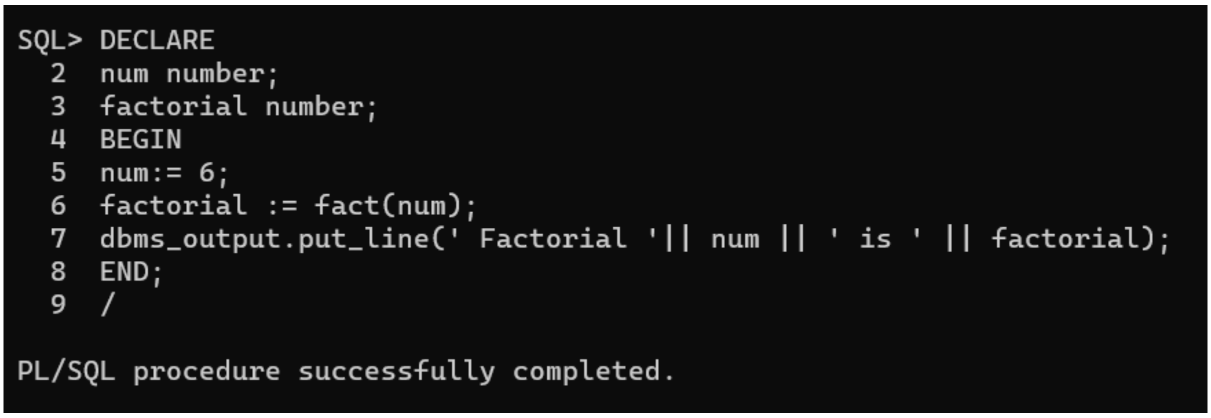
```
SQL> DECLARE
     N3 NUMBER(2);
     BEGIN
     N3 := ADDER(11,22);
     DBMS_OUTPUT.PUT_LINE('ADDITION IS: ' || N3);
     END;
     /

PL/SQL procedure successfully completed.
```

EXAMPLE : 2



```
SQL> 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.
```

Execution Procedure:



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

PL/SQL procedure successfully completed.
```

Conclusion:

The pl/sql program is successfully executed.

EXPERIMENT-15

AIM : TO Write PL/SQL program to implement Trigger on table.

Trigger is invoked by Oracle engine automatically whenever a specified event occurs. Trigger is

stored into database and invoked repeatedly, when specific condition match. Triggers are

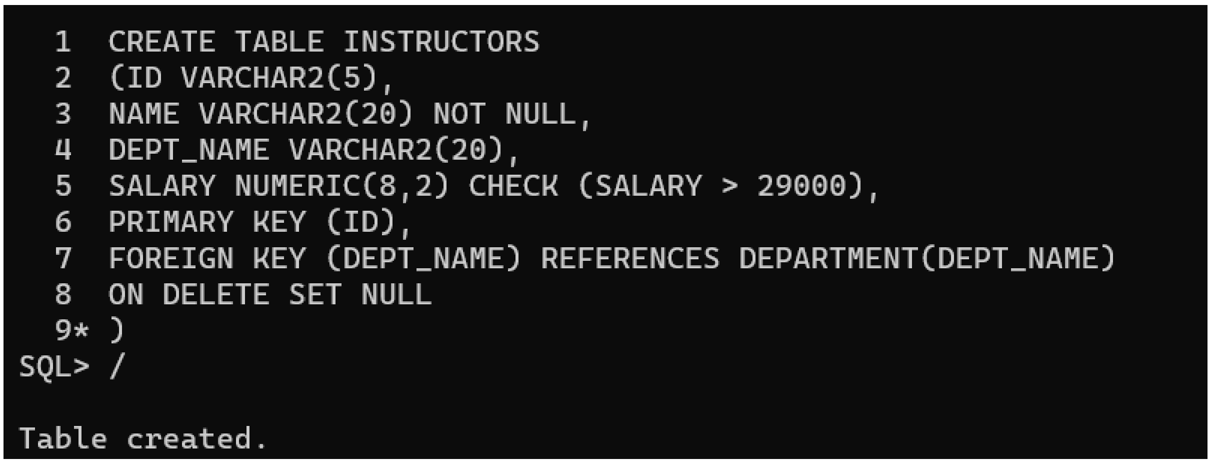
stored programs, which are automatically executed or fired when some event occurs. Triggers

are written to be executed in response to any of the following events.

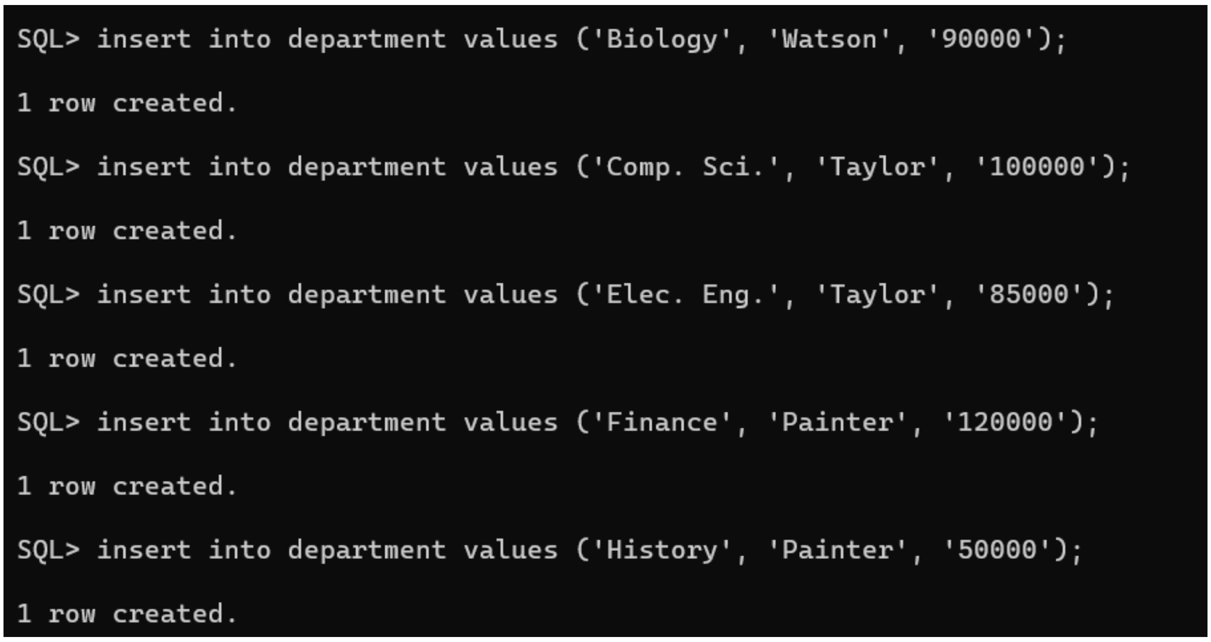
A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).

A database definition (DDL) statement (CREATE, ALTER, or DROP).

A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

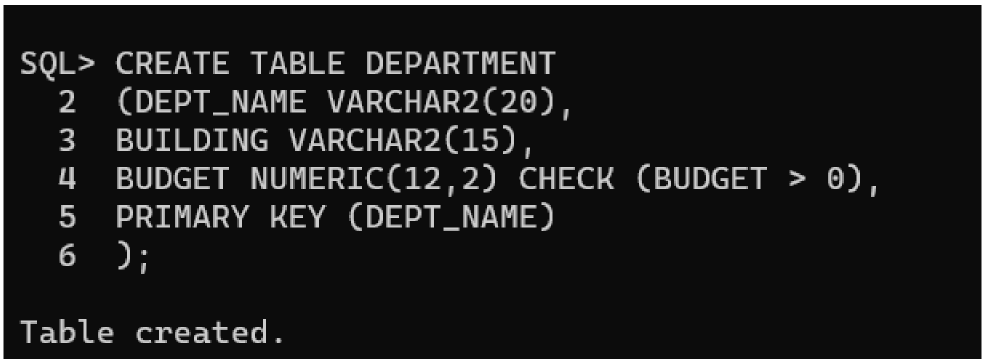


```
CREATE TABLE INSTRUCTORS
(ID VARCHAR2(5),
NAME VARCHAR2(20) NOT NULL,
DEPT_NAME VARCHAR2(20),
SALARY NUMERIC(8,2) CHECK (SALARY > 29000),
PRIMARY KEY (ID),
FOREIGN KEY (DEPT_NAME) REFERENCES DEPARTMENT(DEPT_NAME)
ON DELETE SET NULL
)
SQL> /

Table created.
```



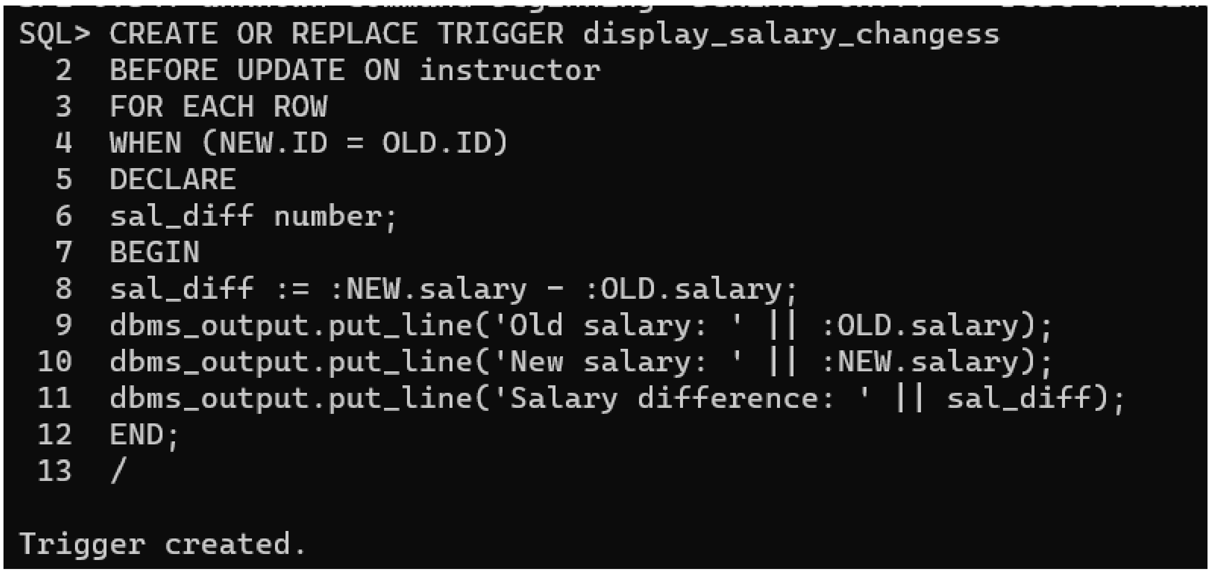
```
SQL> insert into department values ('Biology', 'Watson', '90000');

1 row created.

SQL> insert into department values ('Comp. Sci.', 'Taylor', '100000');

1 row created.

SQL> insert into department values ('Elec. Eng.', 'Taylor', '85000');

1 row created.

SQL> insert into department values ('Finance', 'Painter', '120000');

1 row created.

SQL> insert into department values ('History', 'Painter', '50000');

1 row created.
```

CREATING DEPARTMENT TABLE :



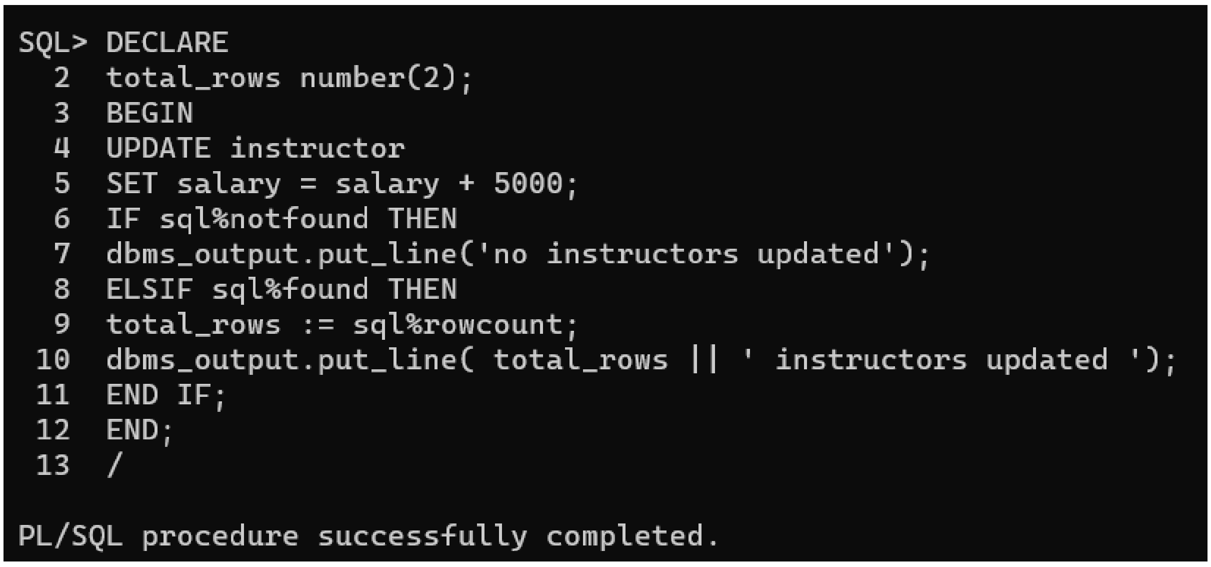
```
SQL> CREATE TABLE DEPARTMENT
     (DEPT_NAME VARCHAR2(20),
     BUILDING VARCHAR2(15),
     BUDGET NUMERIC(12,2) CHECK (BUDGET > 0),
     PRIMARY KEY (DEPT_NAME)
     );

Table created.
```

An example to create Trigger :



```
SQL> CREATE OR REPLACE TRIGGER display_salary_changess
     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.
```

A PL/SQL Procedure to execute a trigger:



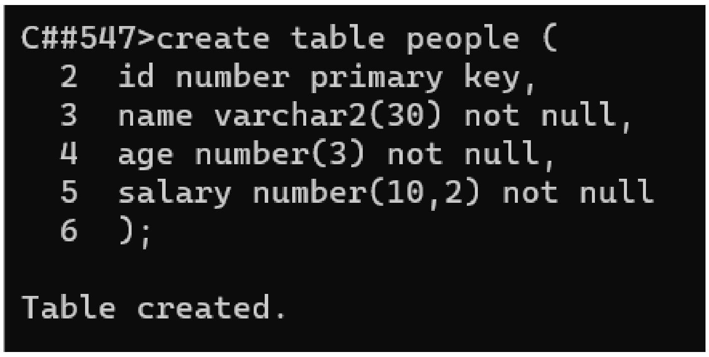
```
SQL> DECLARE
total_rows number(2);
BEGIN
UPDATE instructor
SET salary = salary + 5000;
IF sql%notfound THEN
dbms_output.put_line('no instructors updated');
ELSIF sql%found THEN
total_rows := sql%rowcount;
dbms_output.put_line( total_rows || ' instructors updated ');
END IF;
END;
/

PL/SQL procedure successfully completed.
```

Conclusion:

The pl/sql program is successfully executed.

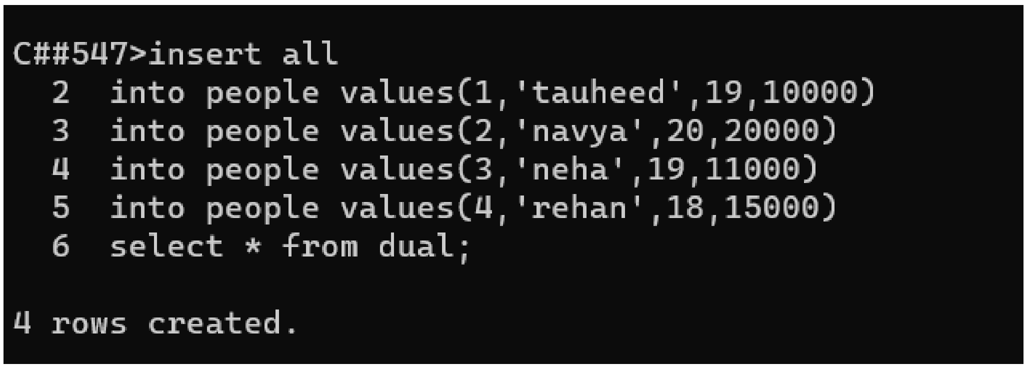
EXPERIMENT-16

AIM: To write PL/SQL program to implement Cursor on table.
Source code:



```
C##547>create table people (
id number primary key,
name varchar2(30) not null,
age number(3) not null,
salary number(10,2) not null
);

Table created.
```

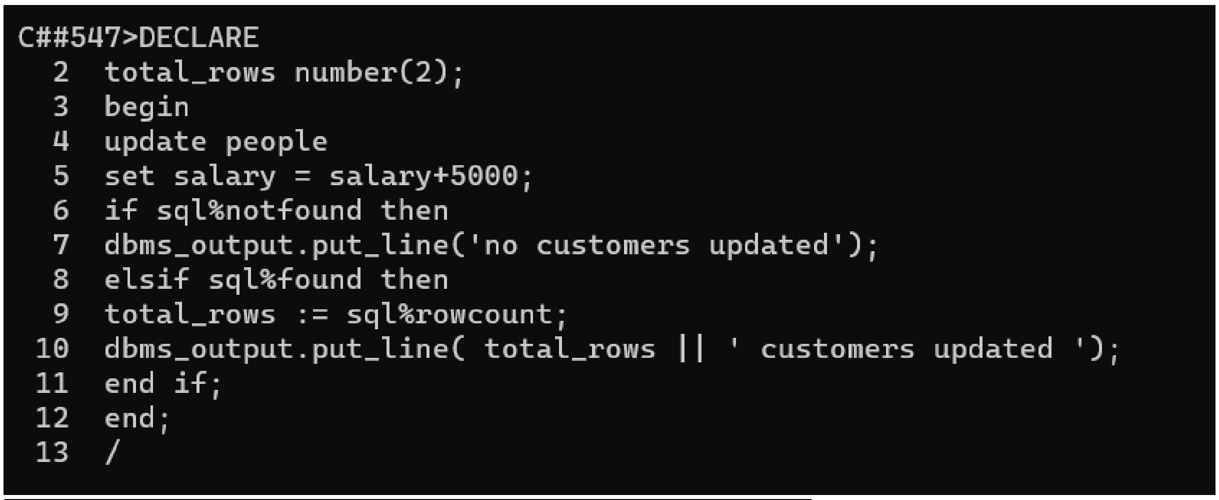
Instances of people :



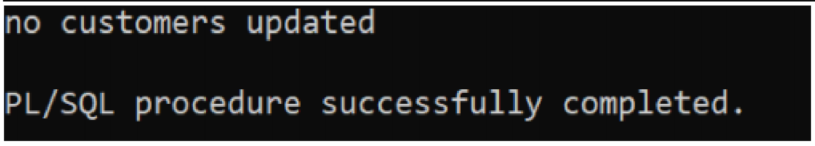
```
C##547>insert all
into people values(1,'tauheed',19,10000)
into people values(2,'navya',20,20000)
into people values(3,'neha',19,11000)
into people values(4,'rehan',18,15000)
select * from dual;

4 rows created.
```

Create update procedure

Create procedure:

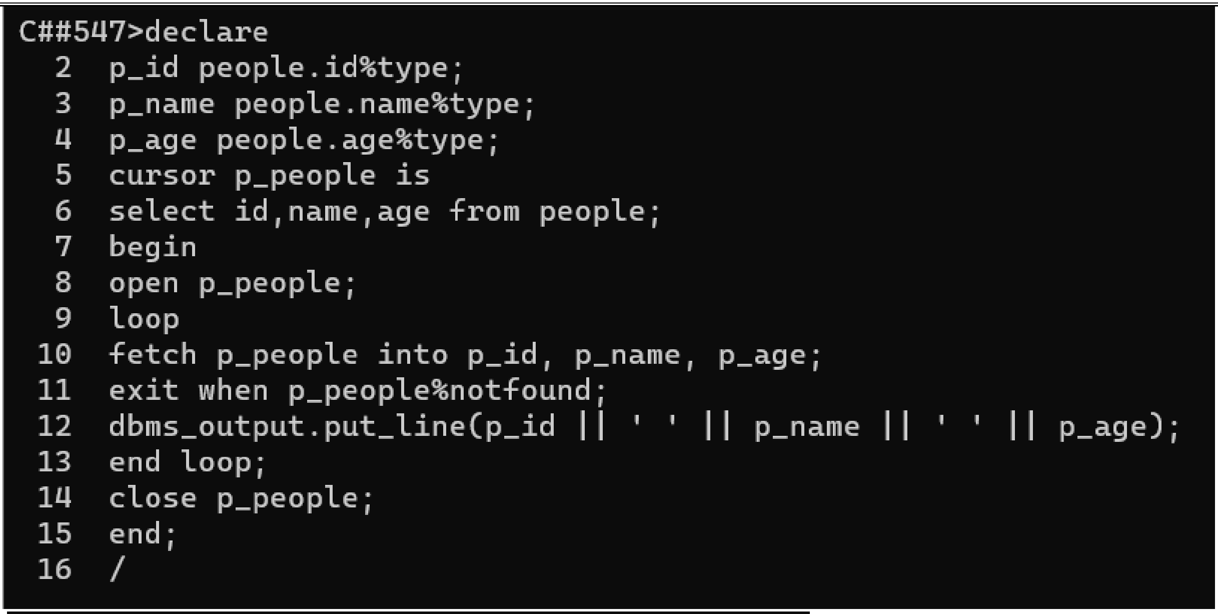


```
C##547>DECLARE
total_rows number(2);
begin
update people
set salary = salary+5000;
if sql%notfound then
dbms_output.put_line('no customers updated');
elsif sql%found then
total_rows := sql%rowcount;
dbms_output.put_line( total_rows || ' customers updated ');
end if;
end;
/
```

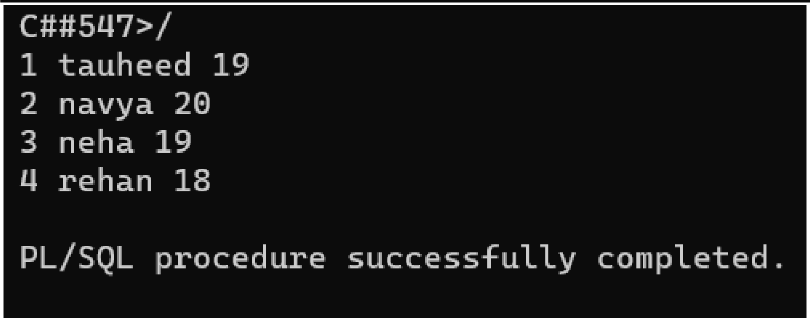


```
no customers updated

PL/SQL procedure successfully completed.
```

PL/SQL Program using Explicit Cursors :

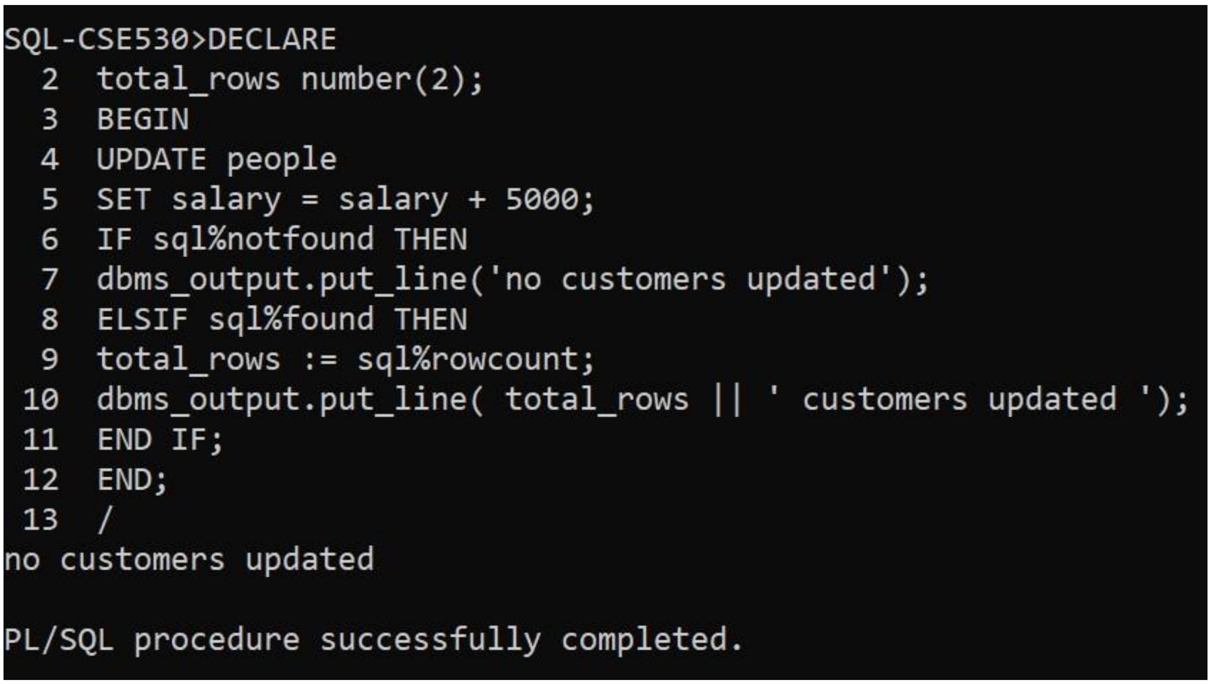


```
C##547>declare
p_id people.id%type;
p_name people.name%type;
p_age people.age%type;
cursor p_people is
select id,name,age from people;
begin
open p_people;
loop
fetch p_people into p_id, p_name, p_age;
exit when p_people%notfound;
dbms_output.put_line(p_id || ' ' || p_name || ' ' || p_age);
end loop;
close p_people;
end;
/
```



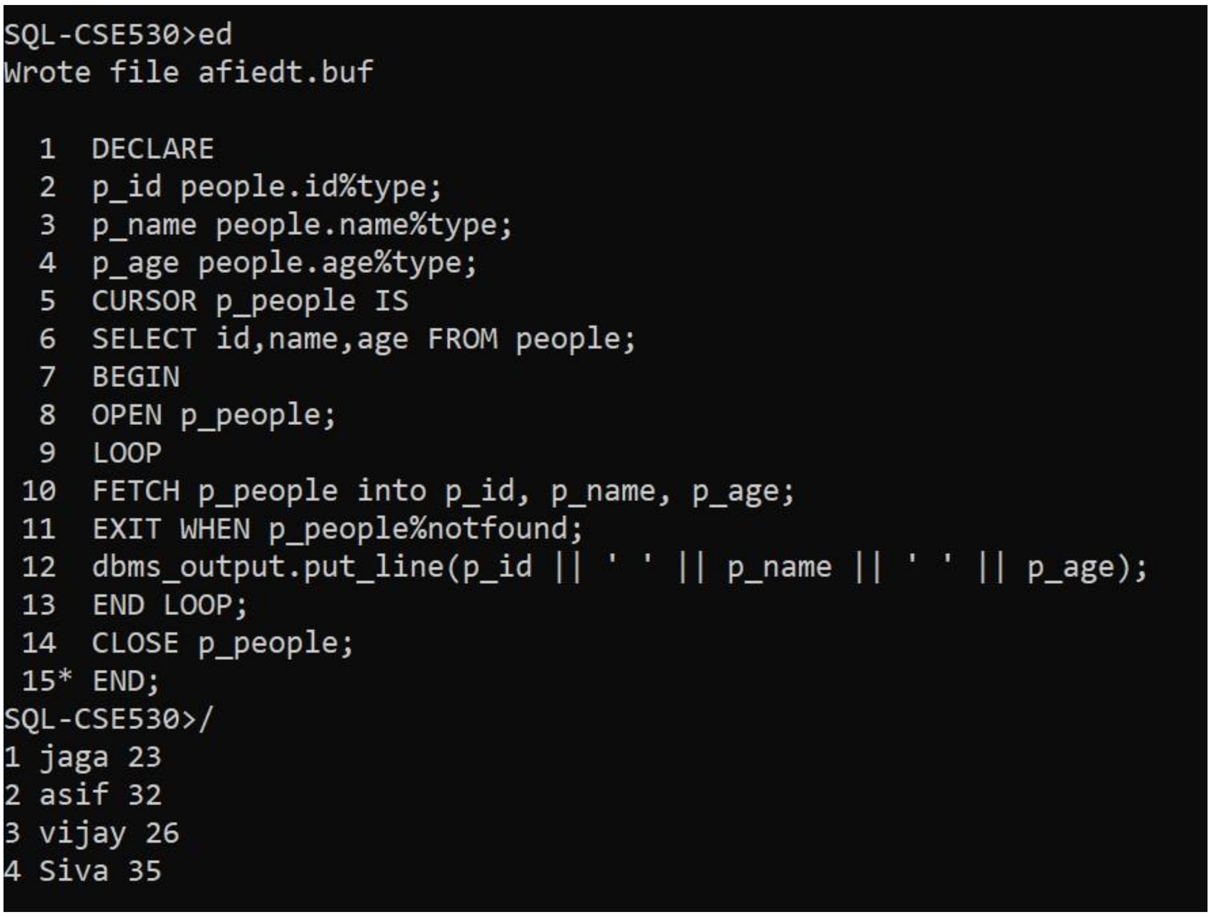
```
C##547>/
1 tauheed 19
2 navya 20
3 neha 19
4 rehan 18

PL/SQL procedure successfully completed.
```

CONCLUSION : The pl/sql program is successfully executed.



```
SQL-CSE530>DECLARE
total_rows number(2);
BEGIN
UPDATE people
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;
/
no customers updated

PL/SQL procedure successfully completed.
```

PL/SQL Program using Explicit Cursors :



```
SQL-CSE530>ed
Wrote file afiedt.buf

DECLARE
p_id people.id%type;
p_name people.name%type;
p_age people.age%type;
CURSOR p_people IS
SELECT id,name,age FROM people;
BEGIN
OPEN p_people;
LOOP
FETCH p_people into p_id, p_name, p_age;
EXIT WHEN p_people%notfound;
dbms_output.put_line(p_id || ' ' || p_name || ' ' || p_age);
END LOOP;
CLOSE p_people;
END;
SQL-CSE530>/
1 jaga 23
2 asif 32
3 vijay 26
4 Siva 35
```

CONCLUSION : The pl/sql program is successfully executed.