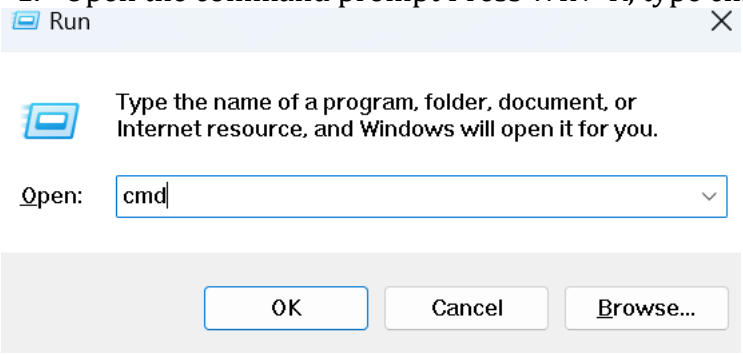
1. Open the command prompt Press WIN+R, type cmd.



2. Once cmd prompt open sqlplus

```
Microsoft Windows [Version 10.0.22631.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\S Varsha>sqlplus

SQL*Plus: Release 21.0.0.0.0 - Production on Sun Jan 7 10:52:37 2024
Version 21.3.0.0.0

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

3. Login with System

```
Enter user-name: system
Enter password:
Last Successful login time: Sun Jan 07 2024 10:51:12 +05:30

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

4. Now create a User with your Id number and grant all privileges to the user

```
SQL> CREATE USER C##CSE_5B7 IDENTIFIED BY 2001 ;

User created.

SQL> GRANT ALL PRIVILEGES TO C##CSE_5B7;

Grant succeeded.
```

5. Now login to sqlplus with the user you have created.

```
C:\Users\S Varsha>sqlplus

SQL*Plus: Release 21.0.0.0.0 - Production on Sun Jan 7 11:08:51 2024
Version 21.3.0.0.0

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

Enter user-name: C##CSE_5B7
Enter password:

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

6.Now, we have successfully logged in into sqlplus.

DDL COMMANDS

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

Create Table:

Syntax:

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

Example:

```
224G1A05B7>CREATE TABLE Persons (
  2      PersonID int,
  3      LastName varchar(255),
  4      FirstName varchar(255),
  5      Address varchar(255),
  6      City varchar(255)
  7  );

Table created.
```

```
224G1A05B7>desc persons
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PERSONID                                           NUMBER(38)
 LASTNAME                                           VARCHAR2(255)
 FIRSTNAME                                          VARCHAR2(255)
 ADDRESS                                            VARCHAR2(255)
 CITY                                               VARCHAR2(255)
```

ALTER

Syntax:

1. ALTER TABLE tablename
   {ADD | MODIFY} (column_name data_type [{ADD|MODIFY}
   Column_name data_type]);

2. ALTER TABLE tablename
   DROP {PRIMARY KEY | COLUMN column_name | CONSTRAINT constraint_name);

Example:

```
224G1A05B7>ALTER TABLE Persons
  2  ADD (Phn NUMBER(10,0));

Table altered.

224G1A05B7>desc persons
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PERSONID                                           NUMBER(38)
 LASTNAME                                           VARCHAR2(255)
 FIRSTNAME                                          VARCHAR2(255)
 ADDRESS                                            VARCHAR2(255)
 CITY                                               VARCHAR2(255)
 PHN                                                NUMBER(10)
```

```
224G1A05B7>ALTER TABLE Persons
  2  DROP COLUMN CITY;

Table altered.

224G1A05B7>desc Persons
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PERSONID                                           NUMBER(38)
 LASTNAME                                           VARCHAR2(255)
 FIRSTNAME                                          VARCHAR2(255)
 ADDRESS                                            VARCHAR2(255)
 PHN                                                NUMBER(10)
```

TRUNCATE

Syntax:

TRUNCATE TABLE table_name;

Example:

```
224G1A05B7>Truncate table stu;

Table truncated.
```

DROP

Syntax:

DROP TABLE table_name;

Example:

```
224G1A05B7>drop table stu;

Table dropped.
```

2. DML COMMANDS

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

```
224G1A05B7>CREATE TABLE student (
  2    roll_no INT NOT NULL PRIMARY KEY ,
  3    name VARCHAR(50) NOT NULL,
  4    age INT,
  5    address VARCHAR(255)
  6  );

Table created.
```

INSERT
Syntax:

INSERT INTO tablename
VALUES (value1,value2,...,valuen);

INSERT INTO tablename
(column1, column2,…,column) VALUES (value1, value2,...,valuen);

Example:

```
224G1A05B7>INSERT INTO student(roll_no, name,age,address)
  2  VALUES (2,'zyan',17,'uk');

1 row created.

224G1A05B7>INSERT INTO student(roll_no, name,age,address)
  2  VALUES (3,'dita',17,'uk');

1 row created.

224G1A05B7>INSERT INTO student(roll_no, name,age,address)
  2  VALUES (4,'niota',17,'nyc');

1 row created.
```

SELECT

Syntax:

```
SELECT *
DROM <table_name>;
```

Example:

```
224G1A05B7>SELECT* FROM student;

   ROLL_NO NAME
    AGE
---------- -------------------------------------------------- ----------
ADDRESS
--------------------------------------------------------------------------------
         1 hira
     17
nyc

         2 zyan
     17
uk

         3 dita
     17
uk


   ROLL_NO NAME
    AGE
---------- -------------------------------------------------- ----------
ADDRESS
-----------+--------------------------------------------------------------------
         4 niota
     17
nyc
```

UPDATE

Syntax:

UPDATE table_name SET [column_name1= value_1, column_name2= value_2,...]
WHERE CONDITION;

Example:

```
224G1A05B7>UPDATE student
  2  SET age = age-1;

4 rows updated.
```

```
224G1A05B7>select * from student;

   ROLL_NO NAME
    AGE
---------- -------------------------------------------------- ----------
ADDRESS
--------------------------------------------------------------------------------
         1 hira
     16
nyc

         2 zyan
     16
uk

         3 dita
     16
uk


   ROLL_NO NAME
    AGE
---------- -------------------------------------------------- ----------
ADDRESS
----------  ------------------------------------------------------------
         4 niota
     16
nyc
```

DELETE

Syntax:

DELETE FROM table_Name WHERE condition;

Example:

```
224G1A05B7>DELETE FROM student
  2  WHERE address = 'nyc';

2 rows deleted.

224G1A05B7>select * from student;

   ROLL_NO NAME
    AGE
---------- ---------------------------------------------------
ADDRESS
----------------------------------------------------------------
         2 zyan
     16
uk

         3 dita
     16
uk
```

3. VIEWS

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

View syntax:

CREATE VIEW VIEW_NAME AS <QUERY EXPRESSION>

CREATE VIEW

```
224G1A05B7> CREATE VIEW FACULTY AS
  2   SELECT ID,NAME,DEPT_NAME
  3   FROM INSTRUCTOR;

View created.
```

```
224G1A05B7> SELECT * FROM faculty;

ID    NAME                 DEPT_NAME
----- -------------------- ----------
10101 Srinivasan           Comp. Sci.
12121 Wu                   Finance
15151 Mozart               Music
22222 Einstein             Physics
32343 El Said              History
33456 Gold                 Physics
45565 Katz                 Comp. Sci.
58583 Califieri            History
76543 Singh                Finance
76766 Crick                Biology
83821 Brandt               Comp. Sci.

12 rows selected.
```

DELETE VIEW:

```
224G1A05B7> DROP view faculty;

View dropped.
```

4. RELATIONAL SET OPERTATIONS

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

```
224G1A05B7> CREATE TABLE CLASSROOM
  2   (BUILDING  VARCHAR2(15),
  3    ROOM_NUMBER  VARCHAR2(7),
  4    CAPACITY  NUMERIC(4,0),
  5    PRIMARY KEY (BUILDING, ROOM_NUMBER)
  6   );

Table created.
```

```
224G1A05B7> CREATE TABLE SECTION
  2   (COURSE_ID VARCHAR2(8),        SEC_ID   VARCHAR2(8),
  3    SEMESTER  VARCHAR2(6)  CHECK (SEMESTER IN ('FALL', 'WINTE
R',
  4  'SPRING', 'SUMMER')),
  5    YEAR   NUMERIC(4,0) CHECK (YEAR > 1701 AND YEAR < 2100),

  6    BUILDING  VARCHAR2(15),
  7    ROOM_NUMBER  VARCHAR2(7),
  8    TIME_SLOT_ID  VARCHAR2(4),
  9  PRIMARY KEY (COURSE_ID, SEC_ID, SEMESTER, YEAR),
 10  FOREIGN KEY (BUILDING, ROOM_NUMBER) REFERENCES CLASSROOM(BU
ILDING,
 11  ROOM_NUMBER)
 12    ON DELETE SET NULL
 13   );

Table created.
```

UNION

```
224G1A05B7> (SELECT course_id
  2   FROM section
  3   where semester = 'Fall' AND year= 2009)
  4   UNION
  5   (SELECT course_id
  6   FROM section
  7   WHERE semester = 'Spring' AND year= 2010);

no rows selected
```

UNION ALL

INTERSECT
MINUS,
CROSS JOIN
NATURAL JOIN

5.SPECIAL OPERATIONS

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

. ISNULL
BETWEEN
LIKE
IN
EXISTS

6.JOIN OPERATIONS

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

CONDITIONAL JOIN
EQUI JOIN
LEFT OUTER JOIN
RIGHT OUTER JOIN
FULL OUTER JOIN

7. AGGREGATE OPERATIONS

Write SQL queries to perform AGGREGATE OPERATIONS (i.e. SUM, COUNT, AVG, MIN, MAX).

SUM
COUNT
AVG
MIN,
MAX

8.ORACLE BUILT-IN FUNCTIONS

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

case-conversion functions:

```
LOWER('SQL
----------
sql course

224G1A05B7> SELECT UPPER('SQL Course')
  2   FROM DUAL;

UPPER('SQL
----------
SQL COURSE

224G1A05B7> SELECT INITCAP('SQL course')
  2   FROM DUAL;

INITCAP('S
----------
Sql Course
```

character manipulation functions:

```
224G1A05B7> SELECT CONCAT('HELLO', 'WORLD')
  2   FROM DUAL;

CONCAT('HE
----------
HELLOWORLD

224G1A05B7>
224G1A05B7>  SELECT SUBSTR('HELLO WORLD',1,5)
  2   FROM DUAL;

SUBST
-----
HELLO
224G1A05B7> SELECT CONCAT('HELLO', 'WORLD')
  2   FROM DUAL;

CONCAT('HE
----------
HELLOWORLD

224G1A05B7>
224G1A05B7>  SELECT INSTR('HELLO WORLD', 'WORLD')
  2   FROM DUAL;

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

```
224G1A05B7> SELECT LPAD(SALARY, 10, '*')
  2   FROM INSTRUCTOR;

LPAD(SALARY,10,'*')
------------------------------------
*****70000
*****95000
*****45000
****100000
*****65000
*****92000
*****80000
*****67000
*****85000
*****77000
*****97000

LPAD(SALARY,10,'*')
------------------------------------
*****85000

12 rows selected.
```

```
224G1A05B7> SELECT REPLACE('JACK and JUE','J','BL')
  2   FROM DUAL;

REPLACE('JACKA
--------------
BLACK and BLUE
```

```
224G1A05B7> SELECT RPAD(SALARY, 10, '*')
  2   FROM INSTRUCTOR;

RPAD(SALARY,10,'*')
---------------------------------------
70000*****
95000*****
45000*****
100000****
65000*****
92000*****
80000*****
67000*****
85000*****
77000*****
97000*****

RPAD(SALARY,10,'*')
---------------------------------------
85000*****

12 rows selected.
```

```
224G1A05B7> SELECT TRIM('H' FROM 'HelloWorld' )
  2   FROM DUAL;

TRIM('H'F
---------
elloWorld
```

Number Functions:

```
224G1A05B7> SELECT ROUND(45.626,2)
  2   FROM DUAL;

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

224G1A05B7>  SELECT ROUND(45.626,0)
  2   FROM DUAL;

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

224G1A05B7>  SELECT ROUND(45.626,-1)
  2   FROM DUAL;

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

224G1A05B7>  SELECT ROUND(45.626,-2)
  2   FROM DUAL;

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

```
224G1A05B7> SELECT TRUNC(45.626, 2)
  2   FROM DUAL;

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

224G1A05B7>  SELECT TRUNC(45.626, 0)
  2   FROM DUAL;

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

224G1A05B7>  SELECT TRUNC(45.626, -1)
  2   FROM DUAL;

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

224G1A05B7>  SELECT TRUNC(45.626, -2)
  2   FROM DUAL;

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

224G1A05B7>
224G1A05B7>
224G1A05B7>  SELECT MOD(1600,300)
  2   FROM DUAL;

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

Date Functions:

```
224G1A05B7>  SELECT MONTHS_BETWEEN(SYSDATE,'15-FEB-20')
  2   FROM DUAL;

MONTHS_BETWEEN(SYSDATE,'15-FEB-20')
-----------------------------------
                          47.5393302

224G1A05B7>
224G1A05B7>  SELECT ADD_MONTHS(SYSDATE, 2)
  2   FROM DUAL;

ADD_MONTH
---------
31-MAR-24
```

```
224G1A05B7>  SELECT LAST_DAY(SYSDATE)
  2   FROM DUAL;

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

224G1A05B7>  SELECT NEXT_DAY(SYSDATE,'THURSDAY')
  2   FROM DUAL;

NEXT_DAY(
---------
01-FEB-24
```

9.KEY CONSTRAINTS

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

PRIMARY KEY:

```
224G1A05B7>  CREATE TABLE Persons (
  2   ID int NOT NULL,
  3   LastName varchar(255) NOT NULL,
  4   FirstName varchar(255),
  5   Age int,
  6   CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
  7   );

Table created.
```

```
224G1A05B7> ALTER TABLE Persons
  2   DROP CONSTRAINT PK_Person;

Table altered.
```

FOREIGN KEY:

```
224G1A05B7> CREATE TABLE Custom (
  2    customer_id INT PRIMARY KEY,
  3    name VARCHAR(255) NOT NULL,
  4    email VARCHAR(255) UNIQUE NOT NULL
  5  );

Table created.
```

```
224G1A05B7> CREATE TABLE Orders (
  2    order_id INT PRIMARY KEY,
  3  customer_id INT NOT NULL,
  4    product_name VARCHAR(255) NOT NULL,
  5    order_date DATE NOT NULL,
  6  FOREIGN KEY (customer_id) REFERENCES Custom(customer_id)
  7  );

Table created.

224G1A05B7> DESC ORDERS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 ORDER_ID                                  NOT NULL NUMBER(38)
 CUSTOMER_ID                               NOT NULL NUMBER(38)
 PRODUCT_NAME                              NOT NULL VARCHAR2(255)
 ORDER_DATE                                NOT NULL DATE
```

UNIQUE:

```
224G1A05B7>  CREATE TABLE Students(
  2   ID int NOT NULL,
  3   LastName varchar(255) NOT NULL,
  4   FirstName varchar(255),
  5   Age int,
  6   CONSTRAINT UC_Person UNIQUE (ID,LastName)
  7   );

Table created.
```

```
224G1A05B7> ALTER TABLE students
  2   DROP CONSTRAINT UC_Person;

Table altered.
```

NOT NULL:

```
224G1A05B7> CREATE TABLE student (
  2   ID int NOT NULL,
  3   LastName varchar(255) NOT NULL,
  4   FirstName varchar(255) NOT NULL,
  5   Age int
  6   );

Table created.
```

```
224G1A05B7> ALTER TABLE student
  2   MODIFY Age int NOT NULL;

Table altered.
```

DEFAULT:

```
224G1A05B7> CREATE TABLE Persons(
  2   ID int NOT NULL,
  3   LastName varchar(255) NOT NULL,
  4   FirstName varchar(255),
  5   Age int,
  6   City varchar(255) DEFAULT 'Sandnes'
  7   );

Table created.
```

```
224G1A05B7> ALTER TABLE Persons
  2   MODIFY City DEFAULT 'Sandnes';

Table altered.
```

10. FACTORIAL

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

```
224G1A05B7> DECLARE
  2  fac NUMBER :=1;
  3  n NUMBER := 10;
  4  BEGIN
  5  WHILE n > 0 LOOP
  6  fac:=n*fac;
  7  n:=n-1;
  8  END LOOP;
  9  DBMS_OUTPUT.PUT_LINE(FAC);
 10  END;
 11  /

PL/SQL procedure successfully completed.
```

11.PRIME OR NUMBER

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

```
224G1A05B7> DECLARE
  2  n NUMBER;
  3  i NUMBER;
  4  temp NUMBER;
  5  BEGIN
  6  n := 13;
  7  i := 2;
  8  temp := 1;
  9  FOR i IN 2..n/2
 10  LOOP
 11  IF MOD(n, i) = 0
 12  THEN
 13  temp := 0;
 14  EXIT;
 15  END IF;
 16  END LOOP;
 17  IF temp = 1
 18  THEN
 19  DBMS_OUTPUT.PUT_LINE(n||' is a prime number');
 20  ELSE
 21  DBMS_OUTPUT.PUT_LINE(n||' is not a prime number');
 22  END IF;
 23  END;
 24  /

PL/SQL procedure successfully completed.
```

12. FIBONACCI

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

```
224G1A05B7> DECLARE
  2  FIRST NUMBER := 0;
  3  SECOND NUMBER := 1;
  4  TEMP NUMBER;
  5  N NUMBER := 5;
  6  I NUMBER;
  7  BEGIN
  8  DBMS_OUTPUT.PUT_LINE('SERIES:');
  9  DBMS_OUTPUT.PUT_LINE(FIRST);
 10  DBMS_OUTPUT.PUT_LINE(SECOND);
 11  FOR I IN 2..N
 12  LOOP
 13  TEMP:=FIRST+SECOND;
 14  FIRST := SECOND;
 15  SECOND := TEMP;
 16  DBMS_OUTPUT.PUT_LINE(TEMP);
 17  END LOOP;
 18  END;
 19  /

PL/SQL procedure successfully completed.
```

13.STORED PROCEDURE

Write PL/SQL program to implement Stored Procedure on table

SYNTAX:

CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter [,parameter])]
(IS | AS)
[declaration_section]
BEGIN
executable_section
[EXCEPTION exception_section]
END
[procedure_name];

```
224G1A05B7>           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.
```

```
224G1A05B7> DECLARE
  2   CNT NUMBER;
  3   BEGIN
  4  INSERTUSER(101,'NARASIMHA');
  5   SELECT COUNT(*) INTO CNT FROM SAILOR;
  6   DBMS_OUTPUT.PUT_LINE(CNT||' RECORD IS INSERTED SUCCESSFULLY');
  7   END;
  8  /

PL/SQL procedure successfully completed.
```

```
224G1A05B7> DROP PROCEDURE insertuser;

Procedure dropped.
```

14.STORED FUNCTION

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

SYNTAX:
CREATE [OR REPLACE] FUNCTION function_name
[(parameter [,parameter])]
RETURN return_datatype
(IS | AS)
[declaration_section]
BEGIN
executable_section
[EXCEPTION exception_section]
END
[procedure_name];

```
224G1A05B7> CREATE OR REPLACE FUNCTION ADDER(N1 IN NUMBER, N2 IN NUMBER)
  2   RETURN NUMBER
  3   IS
  4   N3 NUMBER(8);
  5   BEGIN
  6   N3 :=N1+N2;
  7   RETURN N3;
  8   END;
  9  /

Function created.
```

Execution Procedure:

```
224G1A05B7>   DECLARE
  2         N3 NUMBER(2);
  3    BEGIN
  4         N3 := ADDER(11,22);
  5         DBMS_OUTPUT.PUT_LINE('ADDITION IS: ' || N3);
  6      END;
  7      /

PL/SQL procedure successfully completed.
```

```
224G1A05B7> CREATE FUNCTION fact(x number)
  2  RETURN number
  3  IS
  4     f number;
  5  BEGIN
  6     IF x=0 THEN
  7        f := 1;
  8     ELSE
  9        f := x * fact(x-1);
 10     END IF;
 11  RETURN f;
 12  END;
 13  /

Function created.
```

```
224G1A05B7> DECLARE
  2   num number;
  3   factorial number;
  4   BEGIN
  5      num:= 6;
  6      factorial := fact(num);
  7      dbms_output.put_line(' Factorial '|| num || ' is ' || f
actorial);
  8   END;
  9  /

PL/SQL procedure successfully completed.
```

DROP FUNCTION SYNTAX:

DROP FUNCTION Func_name;
DROP FUNCTION Adder;

```
224G1A05B7> DROP FUNCTION ADDER;

Function dropped.
```

15.IMPLEMENT TRIGGER

Write PL/SQL program to implement Trigger on table

Syntax:

```
CREATE [OR REPLACE ] TRIGGER TRIGGER_NAME
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF COL_NAME]
ON TABLE_NAME
[REFERENCING OLD AS O NEW AS N]
[FOR EACH ROW]
WHEN (CONDITION)
DECLARE
DECLARATION-STATEMENTS
BEGIN
EXECUTABLE-STATEMENTS
EXCEPTION
EXCEPTION-HANDLING-STATEMENTS
END;
```

```
224G1A05B7> CREATE TABLE DEPARTMENT
  2   (DEPT_NAME  VARCHAR2(20),
  3    BUILDING  VARCHAR2(15),
  4    BUDGET   NUMERIC(12,2) CHECK (BUDGET > 0),
  5    PRIMARY KEY (DEPT_NAME)
  6   );

Table created.
224G1A05B7> CREATE TABLE INSTRUCTOR
  2   (ID   VARCHAR2(5),
  3    NAME   VARCHAR2(20) NOT NULL,
  4    DEPT_NAME  VARCHAR2(20),
  5    SALARY   NUMERIC(8,2) CHECK (SALARY > 29000),
  6    PRIMARY KEY (ID),
  7  FOREIGN KEY (DEPT_NAME) REFERENCES DEPARTMENT(DEPT_NAME)
  8    ON DELETE SET NULL
  9   );

Table created.
```

```
224G1A05B7> insert into department values ('Biology', 'Watson', '90000');

1 row created.

224G1A05B7>  insert into department values ('Comp. Sci.', 'Taylor', '100000');

1 row created.

224G1A05B7>  insert into department values ('Elec. Eng.', 'Taylor', '85000');

1 row created.

224G1A05B7>  insert into department values ('Finance', 'Painter', '120000');

1 row created.

224G1A05B7>  insert into department values ('History', 'Painter', '50000');

1 row created.
224G1A05B7>  insert into department values ('Music', 'Packard', '80000');

1 row created.

224G1A05B7>  insert into department values ('Physics', 'Watson', '70000');

1 row created.

224G1A05B7> insert into instructor values ('10101', 'Srinivasan', 'Comp. Sci.', '65000');

1 row created.

224G1A05B7>  insert into instructor values ('12121', 'Wu', 'Finance', '90000');

1 row created.

224G1A05B7>  insert into instructor values ('15151', 'Mozart', 'Music', '40000');

1 row created.
224G1A05B7>  insert into instructor values ('32343', 'El Said', 'History', '60000');

1 row created.

224G1A05B7>  insert into instructor values ('33456', 'Gold', 'Physics', '87000');

1 row created.

224G1A05B7>  insert into instructor values ('45565', 'Katz', 'Comp. Sci.', '75000');

1 row created.
```

```
224G1A05B7>  insert into instructor values ('58583', 'Califieri', 'History', '62000');

1 row created.

224G1A05B7>  insert into instructor values ('76543', 'Singh', 'Finance', '80000');

1 row created.

224G1A05B7>  insert into instructor values ('76766', 'Crick', 'Biology', '72000');

1 row created.

224G1A05B7>  insert into instructor values ('83821', 'Brandt', 'Comp. Sci.', '92000');

1 row created.

224G1A05B7>  insert into instructor values ('98345', 'Kim', 'Elec. Eng.', '80000');

1 row created.
```

Create a trigger:

```
224G1A05B7> CREATE OR REPLACE TRIGGER display_salary_changes
  2   BEFORE UPDATE ON instructor
  3      FOR EACH ROW
  4      WHEN (NEW.ID = OLD.ID)
  5      DECLARE
  6         sal_diff number;
  7      BEGIN
  8         sal_diff := :NEW.salary  - :OLD.salary;
  9         dbms_output.put_line('Old salary: ' || :OLD.salary);
 10         dbms_output.put_line('New salary: ' || :NEW.salary);
 11         dbms_output.put_line('Salary difference: ' || sal_diff);
 12      END;
 13  /

Trigger created.
```

Execute a trigger:

```
224G1A05B7>    DECLARE
  2         total_rows number(2);
  3      BEGIN
  4         UPDATE  instructor
  5         SET salary = salary + 5000;
  6         IF sql%notfound THEN
  7            dbms_output.put_line('no instructors updated');

  8         ELSIF sql%found THEN
  9            total_rows := sql%rowcount;
 10            dbms_output.put_line( total_rows || ' instructors
 updated ');
 11         END IF;
 12      END;
 13  /

PL/SQL procedure successfully completed.
```

DROP the Trigger:

```
224G1A05B7> DROP TRIGGER display_salary_changes;

Trigger dropped.
```

16. IMPLEMENT CURSOR

Write PL/SQL program to implement Cursor on table

Declare the cursor:
SYNTAX:
CURSOR cursor_name IS select_statement;

Open the cursor
SYNTAX:
OPEN cursor_name;

Fetch the cursor
SYNTAX:
FETCH cursor_name INTO variable_list;

Close the cursor:
SYNTAX:
Close cursor_name;

```
224G1A05B7>CREATE TABLE customers(
  2  ID NUMBER PRIMARY KEY,
  3  NAME VARCHAR2(20) NOT NULL,
  4  AGE NUMBER,
  5  ADDRESS VARCHAR2(20),
  6  SALARY NUMERIC(20,2));

Table created.
```

```
224G1A05B7>INSERT INTO customers VALUES(3, 'Mahesh',24,'Ghaziabad',29000);

1 row created.

224G1A05B7>INSERT INTO customers VALUES(4, 'chandhan',25,'Noida',31000);

1 row created.
```

```
224G1A05B7>DECLARE
  2   c_id customers.id%type;
  3   c_name customers.name%type;
  4   c_addr customers.address%type;
  5   CURSOR c_customers is
  6   SELECT id, name, address FROM customers;
  7   BEGIN
  8   OPEN c_customers;
  9   LOOP
 10   FETCH c_customers into c_id, c_name, c_addr;
 11   EXIT WHEN c_customers%notfound;
 12   dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_add
r);
 13   END LOOP;
 14   CLOSE c_customers;
 15   END;
 16  /

PL/SQL procedure successfully completed.
```