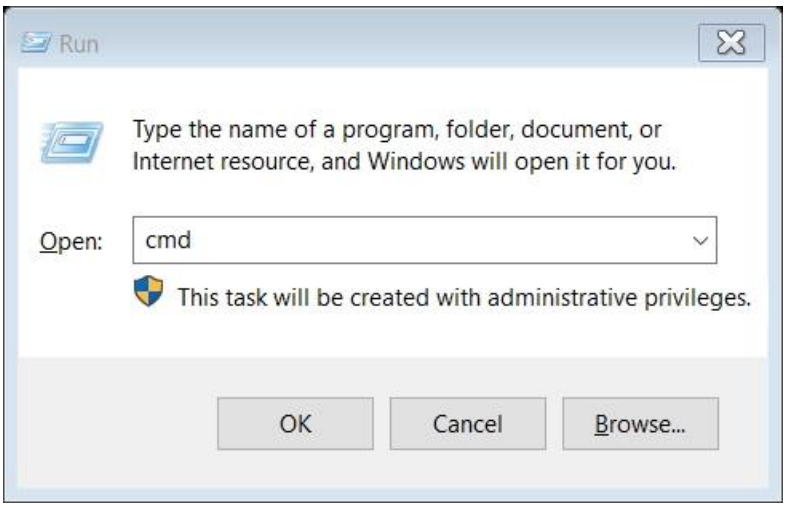
1. Open the command prompt Press WIN+R , type cmd



2. Create user with your id number and grant all privileges.

```
sanu594>CREATE USER SRIT594 IDENTIFIED BY SRIT;

User created.

sanu594>GRANT ALL PRIVILEGES TO SRIT594;

Grant succeeded.
```

1.DDL COMMANDS

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

Example:

```
SRIT594>CREATE TABLE Orders(
  2  OrderID int NOT NULL,
  3  OrderNumber int NOT NULL,
  4  PersonID int,
  5  PRIMARY KEY (OrderID)
  6  );

Table created.
```

ALTER TABLE

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

Syntax 2;
ALTER TABLE tablename
ADD constraint [ADD constraint];

Syntax 3:
ALTER TABLE tablename
DROP {PRIMARY KEY | COLUMN column_name | CONSTRAINT constraint_name);

Syntax 4:
ALTER TABLE tablename
ENABLE CONSTRAINT constraint_name;

Example:

```
SRIT594>ALTER TABLE Orders
  2  ADD(Mail varchar(32));

Table altered.
```

**DESC Orders:

```
SRIT594>DESC Orders
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDERID                                   NOT NULL NUMBER(38)
 ORDERNUMBER                               NOT NULL NUMBER(38)
 PERSONID                                           NUMBER(38)
 MAIL                                               VARCHAR2(32)
```

DROP TABLE

Syntax:

DROP TABLE table_name;

Example:

```
SRIT594>drop table Orders;

Table dropped.
```

2.DML COMMANDS

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

```
SRIT594>CREATE TABLE Student(
  2  Roll_no INT NOT NULL PRIMARY KEY,
  3  Name VARCHAR(50) NOT NULl,
  4  Age INT NOT NULL,
  5  Address VARCHAR(255),
  6  Date_Of_Birth DATE
  7  );

Table created.
```

INSERT

Syntax:

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

Syntax 2:

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

Example:

```
Table created.

SRIT594>INSERT INTO Student(Roll_no, Name, Age)
  2  VALUES(2, 'sweety', 24);

1 row created.
```

SELECT

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

Example:

```
SRIT594>SELECT * FROM Student;

   ROLL_NO NAME                                               AGE
---------- -------------------------------------------------- ----------
ADDRESS
--------------------------------------------------------------------------------
DATE_OF_B
---------
         2 sweety                                              24
```

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

Example:

```
SRIT594>UPDATE Student
  2  SET Age = Age+1;

1 row updated.
```

DELETE
Syntax:
DELETE FROM table_Name WHERE condition;
Example:

```
SRIT594>DELETE FROM Student
  2  WHERE Age < 21;

0 rows deleted.
```

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>

```
SRIT594>CREATE VIEW FACULTY AS
  2  SELECT ID,NAME,DEPT_NAME
  3  FROM INSTRUCTOR;

View created.
```

Commands to insert, Delete and update view

```
SRIT594>CREATE VIEW HISTORY_INSTRUCTOR AS
  2  SELECT * FROM INSTRUCTOR
  3  WHERE DEPT_NAME='HISTORY';

View created.
```

DELETE VIEW:

DROP VIEW view_name;

4.RELATIONAL SET OPERATIONS

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

```
SRIT594>CREATE TABLE CLASSROOM(
  2  BUILDING VARCHAR2(15),
  3  ROOM_NUMBER VARCHAR(65),
  4  CAPACITY NUMERIC(3,0),
  5  PRIMARY KEY (BUILDING, ROOM_NUMBER)
  6  );

Table created.
```

```
SRIT594>CREATE TABLE SEC(
  2  COURSE_ID VARCHAR2(81),
  3  SEC_ID VARCHAR2(25),
  4  SEMESTER VARCHAR2(36) CHECK (SEMESTER IN('FALL', 'WINTER','SPRING','SUMMER')),
  5  YEAR NUMERIC(4,0) CHECK (YEAR > 1701 AND YEAR < 2100),
  6  BUILDING VARCHAR2(15),
  7  ROOM_NUMBER VARCHAR(44),
  8  TIME_SLOT_ID VARCHAR(23),
  9  PRIMARY KEY(COURSE_ID, SEC_ID, SEMESTER, YEAR),
 10  FOREIGN KEY(BUILDING, ROOM_NUMBER) REFERENCES CLASSROOM(BUILDING,ROOM_NUMBER)
 11  ON DELETE SET NULL
 12  );

Table created.
```

```
SRIT594>INSERT INTO CLASSROOM VALUES('watt', '123','45');

1 row created.

SRIT594>INSERT INTO CLASSROOM VALUES('wat', '124','55');

1 row created.

SRIT594>INSERT INTO CLASSROOM VALUES('was', '125','65');

1 row created.

SRIT594>INSERT INTO CLASSROOM VALUES('das', '126','75');

1 row created.

SRIT594>INSERT INTO CLASSROOM VALUES('happy', '127','85');

1 row created.
```

Union operation:

```
SRIT594>SELECT course_id
  2  FROM sec
  3  WHERE SEMESTER = 'FALL' AND YEAR = 2009
  4  UNION SELECT COURSE_ID
  5  FROM SEC
  6  WHERE SEMESTER = 'SPRING' AND YEAR = 2010;

no rows selected
```

```
SRIT594>SELECT COURSE_ID
  2  FROM SEC
  3  WHERE SEMESTER = 'FALL' AND YEAR = 2009
  4  UNION ALL
  5  SELECT COURSE_ID
  6  FROM SECTION
  7  WHERE SEMESTER = 'SPRING' AND YEAR = 2010;

no rows selected
```

Intersect Operation:

```
SRIT594>SELECT COURSE_ID
  2  FROM SEC
  3  WHERE SEMESTER = 'FALL' AND YEAR = 2009
  4  INTERSECT
  5  SELECT COURSE_ID
  6  FROM SECTION
  7  WHERE SEMESTER = 'SPRING' AND YEAR = 2010;

no rows selected
```

Intersect all operation:

```
SRIT594>SELECT COURSE_ID
  2  FROM SEC
  3  WHERE SEMESTER = 'FALL' AND YEAR = 2009
  4  INTERSECT ALL
  5  SELECT COURSE_ID
  6  FROM SECTION
  7  WHERE SEMESTER = 'SPRING' AND YEAR = 2010;

no rows selected
```

except or minus operation:

```
SRIT594>SELECT course_id
  2  FROM sec
  3  WHERE SEMESTER = 'FALL' AND YEAR = 2009
  4  EXCEPT
  5  SELECT COURSE_ID
  6  FROM SECTION
  7  WHERE SEMESTER = 'SPRING' AND YEAR = 2010;

no rows selected
```

except all or minus all operations:

```
SRIT594>SELECT course_id
  2  FROM sec
  3  WHERE SEMESTER = 'FALL' AND YEAR = 2009
  4  EXCEPT ALL
  5  SELECT COURSE_ID
  6  FROM SECTION
  7  WHERE SEMESTER = 'SPRING' AND YEAR = 2010;

no rows selected
```

5.SPECIAL OPERATIONS

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

```
SRIT594>CREATE TABLE DEPT(
  2  DEPT_ID INT NOT NULL PRIMARY KEY,
  3  DEPT_NAME VARCHAR(254) NOT NULL
  4  );

Table created.
```

```
SRIT594>CREATE TABLE EMPLOYEE(
  2  EMP_ID INT NOT NULL PRIMARY KEY,
  3  EMP_NAME VARCHAR2(244) NOT NULL,
  4  EMP_SALARY DECIMAL(10,2) NOT NULL,
  5  EMP_DEPTID INT NOT NULL,
  6  EMP_DEPTNAME VARCHAR2(23) NOT NULL,
  7  CONSTRAINT FK_EMP_DEPTID FOREIGN KEY (EMP_DEPTID) REFERENCES DEPT(DEPT_ID)
  8  );

Table created.
```

```
SRIT594>INSERT INTO DEPT VALUES('1','ENGINEERING');

1 row created.

SRIT594>INSERT INTO DEPT VALUES('3','MEDICAL');

1 row created.

SRIT594>INSERT INTO DEPT VALUES('5','MARKETING');

1 row created.

SRIT594>INSERT INTO DEPT VALUES('7','BUSINESS');

1 row created.
```

```
SRIT594>INSERT INTO EMPLOYEE VALUES('121','DFRG','24467889','1','ENGINEERING');

1 row created.

SRIT594>INSERT INTO EMPLOYEE VALUES('141','RYTHGFG','244547889','3','MEDICAL');
INSERT INTO EMPLOYEE VALUES('141','RYTHGFG','244547889','3','MEDICAL')
                                            *
ERROR at line 1:
ORA-01438: value larger than specified precision allowed for this column


SRIT594>INSERT INTO EMPLOYEE VALUES('141','RYHGFG','2449','3','MEDICAL');

1 row created.

SRIT594>INSERT INTO EMPLOYEE VALUES('241','RYH','244549','5','MARKETING');

1 row created.

SRIT594>INSERT INTO EMPLOYEE VALUES('2414','RYTGH','244439','7','BUSINESS');

1 row created.
```

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)

Natural JOIN

```
SRIT594>SELECT NAME, COURSE_ID
  2  FROM INSTRUCTOR NATURAL JOIN SEC;

no rows selected
```

CONDITIONAL JOIN

OUTER JOINS
left outer join operation.

RIGHT OUTER JOIN

7.AGGREGATEOPERATIONS

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

```
SRIT594>CREATE TABLE DEPARTMENT(
  2  DEPT_NAME VARCHAR2(20),
  3  BUILDING VARCHAR(255),
  4  BUDGET NUMERIC(12,6) CHECK (BUDGET > 21),
  5  PRIMARY KEY (DEPT_NAME)
  6  );

Table created.
```

```
SRIT594>CREATE TABLE INSTRUCT(
  2  ID VARCHAR2(5),
  3  NAME VARCHAR2(343) NOT NULL,
  4  DEPT_NAME VARCHAR2(20),
  5  SALARY NUMERIC(23,3) CHECK (SALARY >23434),
  6  PRIMARY KEY(ID),
  7  FOREIGN KEY (DEPT_NAME) REFERENCES DEPARTMENT(DEPT_NAME)
  8  ON DELETE SET NULL
  9  );

Table created.
```

```
SRIT594>INSERT INTO DEPARTMENT VALUES('CSE','FGFD','144567');

1 row created.

SRIT594>INSERT INTO DEPARTMENT VALUES('CSD','FGFFD','154567');

1 row created.

SRIT594>INSERT INTO DEPARTMENT VALUES('ECE','GFFD','454567');

1 row created.

SRIT594>INSERT INTO DEPARTMENT VALUES('EEE','GFFJD','464567');

1 row created.

SRIT594>INSERT INTO DEPARTMENT VALUES('MEC','GFYJD','466567');

1 row created.
```

```
SRIT594>INSERT INTO INSTRUCT VALUES('1234','DRERG','CSE','65000');

1 row created.

SRIT594>INSERT INTO INSTRUCT VALUES('1334''RERG','CSD','85000');
INSERT INTO INSTRUCT VALUES('1334''RERG','CSD','85000')
           *
ERROR at line 1:
ORA-00947: not enough values


SRIT594>INSERT INTO INSTRUCT VALUES('1334','RERG','CSD','85000');

1 row created.

SRIT594>INSERT INTO INSTRUCT VALUES('13434','REGG','ECE','35000');

1 row created.

SRIT594>INSERT INTO INSTRUCT VALUES('1T434','REBGG','EEE','55000');

1 row created.
```

8. ORACLE BUILT IN FUNCTIONS

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

case-conversion functions:

```
SRIT594>SELECT LOWER('SQL COURSE')
  2  FROM DUAL;

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

```
SRIT594>SELECT UPPER('SQL COURSE')
  2  FROM DUAL;

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

```
SRIT594>SELECT INITCAP('SQL COURSE')
  2  FROM DUAL;

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

character manipulation functions

```
SRIT594>SELECT CONCAT('HELLO','WORLD')
  2  FROM DUAL;

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

```
SRIT594>SELECT SUBSTR('HELLO WORLD',1,5)
  2  FROM DUAL;

SUBST
-----
HELLO
```

```
SRIT594>SELECT INSTR('HELLO WORLD','WORLD')
  2  FROM DUAL;

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

SRIT594>SELECT LPAD(SALARY,10,'*')
  2  FROM INSTRUCT;

LPAD(SALARY,10,'*')
---------------------------------------
*****65000
*****85000
*****35000
*****55000
```

```
SRIT594>SELECT RPAD(SALARY,10,'*')
  2  FROM INSTRUCT;

RPAD(SALARY,10,'*')
---------------------------------------
65000*****
85000*****
35000*****
55000*****

SRIT594>SELECT REPLACE('JAY AND KIE','W','EW')
  2  FROM DUAL;

REPLACE('JA
-----------
JAY AND KIE
```

Number Functions:

```
SRIT594>SELECT ROUND(23.344,3)
  2  FROM DUAL;

ROUND(23.344,3)
---------------
         23.344

SRIT594>SELECT ROUND(2334.33,2)
  2  FROM DUAL;

ROUND(2334.33,2)
----------------
         2334.33

SRIT594>SELECT ROUND(24.33,2)
  2  FROM DUAL;

ROUND(24.33,2)
--------------
         24.33

SRIT594>SELECT ROUND(24.33,-2)
  2  FROM DUAL;

ROUND(24.33,-2)
---------------
              0

SRIT594>SELECT ROUND(24.33,0)
  2  FROM DUAL;

ROUND(24.33,0)
--------------
            24
```

```
SRIT594>SELECT TRUNC(34.34,2)
  2  FROM DUAL;

TRUNC(34.34,2)
--------------
         34.34

SRIT594>SELECT TRUNC(34.34,0)
  2  FROM DUAL;

TRUNC(34.34,0)
--------------
            34

SRIT594>SELECT TRUNC(34.34,-1)
  2  FROM DUAL;

TRUNC(34.34,-1)
---------------
             30

SRIT594>SELECT TRUNC(34.34,-2)
  2  FROM DUAL;

TRUNC(34.34,-2)
---------------
              0
```

Date functions:

```
SRIT594>SELECT SYSDATE
  2  FROM DUAL;

SYSDATE
---------
31-JAN-24

SRIT594>SELECT MONTHS_BETWEEN(SYSDATE,'11-MAR-2005')
  2  FROM DUAL;

MONTHS_BETWEEN(SYSDATE,'11-MAR-2005')
-------------------------------------
                            226.668617

SRIT594>SELECT ADD_MONTHS(SYSDATE,2)
  2  FROM DUAL;

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

SRIT594>SELECT NEXT_DAY(SYSDATE,'THURSDAY')
  2  FROM DUAL;

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

```
SRIT594>SELECT LAST_DAY(SYSDATE)
  2  FROM DUAL;

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

9.KEY CONSTRAINTS

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

NOT NULL COnstraint Example

```
SRIT594>CREATE TABLE STU(
  2  ID INT NOT NULL,
  3  LASTNAME VARCHAR2(24) NOT NULL,
  4  FIRSTNAME VARCHAR2(24) NOT NULL,
  5  AGE INT
  6  );

Table created.
```

```
SRIT594>ALTER TABLE STU
  2  MODIFY AGE INT NOT NULL;

Table altered.
```

UNIQUE CONSTRAINT Example

```
SRIT594>CREATE TABLE STUDENTS(
  2  ID INT NOT NULL,
  3  LASTNAME VARCHAR2(25) NOT NULL,
  4  FIRSTNAME VARCHAR2(25) NOT NULL,
  5  AGE INT,
  6  CONSTRAINT UC_PERSON UNIQUE(ID,LASTNAME)
  7  );

Table created.
```

```
SRIT594>ALTER TABLE STUDENTS
  2  DROP CONSTRAINT UC_PERSON;

Table altered.
```

```
SRIT594>DESC STUDENTS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 LASTNAME                                  NOT NULL VARCHAR2(25)
 FIRSTNAME                                 NOT NULL VARCHAR2(25)
 AGE                                                NUMBER(38)
```

PRIMARY KEY CONSTRAINT Example:

```
SRIT594>CREATE TABLE PERSONS(
  2  ID INT NOT NULL,
  3  LASTNAME VARCHAR2(25) NOT NULL,
  4  FIRSTNAME VARCHAR2(23) NOT NULL,
  5  AGE INT,
  6  CONSTRAINT PK_PERSON PRIMARY KEY(ID,LASTNAME)
  7  );

Table created.
```

```
SRIT594>ALTER TABLE PERSONS
  2  DROP CONSTRAINT PK_PERSON;

Table altered.
```

```
SRIT594>DESC PERSONS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
 LASTNAME                                  NOT NULL VARCHAR2(25)
 FIRSTNAME                                 NOT NULL VARCHAR2(23)
 AGE                                                NUMBER(38)
```

CHECK CONSTRAINTS Example:

```
SRIT594>CREATE TABLE PERSONS(
  2  ID INT NOT NULL,
  3  LASTNAME VARCHAR2(25) NOT NULL,
  4  FIRSTNAME VARCHAR2(23) NOT NULL,
  5  AGE INT,
  6  CITY VARCHAR2(24),
  7  CONSTRAINT CHK_PERSON CHECK(AGE>=18 AND CITY='ATP')
  8  );

Table created.
```

```
SRIT594>ALTER TABLE PERSONS
  2  DROP CONSTRAINT CHK_PERSON;

Table altered.
```

```
SRIT594>ALTER TABLE PERSONS
  2  MODIFY CITY DEFAULT 'ATP';

Table altered.

SRIT594>ALTER TABLE PERSONS MODIFY CITY DEFAULT NULL;

Table altered.
```

10. FACTORIAL

```
SRIT594>SET SERVEROUTPUT ON
SRIT594>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  /
3628800
```

```
SRIT594>DECLARE
  2  N NUMBER;
  3  FACT NUMBER:=1;
  4  I NUMBER:=1;
  5  C NUMBER:=0;
  6  BEGIN
  7  N:=&N;
  8  IF(N<0)
  9  THEN
 10  DBMS_OUTPUT.PUT_LINE('FACTORIAL OF NEGATIVE NUMBER DOES NOT EXIST');
 11  END IF;
 12  WHILE(I<=N)
 13  LOOP
 14  FACT:=FACT*I;
 15  I:=I+1;
 16  END LOOP;
 17  DBMS_OUTPUT.PUT_LINE('FACTORIAL OF '||N||' IS '||FACT);
 18  END;
 19  /
Enter value for n: 4
old   7: N:=&N;
new   7: N:=4;
FACTORIAL OF 4 IS 24

PL/SQL procedure successfully completed.
```

11.PRIME NUMBER OR NOT

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

```
SRIT594>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  /
13 IS A PRIME NUMBER

PL/SQL procedure successfully completed.
```

```
SRIT594>DECLARE
  2  N NUMBER;
  3  I NUMBER;
  4  FLAG NUMBER;
  5  BEGIN
  6  I:=2;
  7  FLAG:=1;
  8  N:=&N;
  9  FOR I IN 2..N/2
 10  LOOP
 11  IF MOD(N,I)=0
 12  THEN
 13  FLAG:=0;EXIT;
 14  END IF;
 15  END LOOP;
 16  IF FLAG=1
 17  THEN
 18  DBMS_OUTPUT.PUT_LINE('PRIME');
 19  ELSE
 20  DBMS_OUTPUT.PUT_LINE('NOT PRIME');
 21  END IF;
 22  END;
 23  /
Enter value for n: 7
old   8: N:=&N;
new   8: N:=7;
PRIME

PL/SQL procedure successfully completed.
```

12.FIBONACCI

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

```
SRIT594>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  /
SERIES:
0
1
1
2
3
5

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];
```

Example:

```
SRIT594>DECLARE
  2  A NUMBER;
  3  B NUMBER;
  4  C NUMBER;
  5  PROCEDURE FINDMIN(X IN NUMBER, Y IN NUMBER, Z OUT NUMBER) IS
  6  BEGIN
  7  IF X<Y THEN
  8  Z:=X;
  9  ELSE
 10  Z:=Y;
 11  END IF;
 12  END;
 13  BEGIN
 14  A:=23;
 15  B:=45;
 16  FINDMIN(A,B,C);
 17  DBMS_OUTPUT.PUT_LINE('MINIMUM OF (23,45):'|| C);
 18  END;
 19  /
MINIMUM OF (23,45):23

PL/SQL procedure successfully completed.
```

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];

Example:

```
SRIT594>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.

SRIT594>DROP FUNCTION fact;

Function dropped.
```

```
SRIT594>CREATE FUNCTION FACT(X NUMBER)
  2  RETURN NUMBER
  3  IS
  4  F NUMBER;
  5  BEGIN
  6  IF X=0 THEN
  7  F:=X*FACT(X-1);
  8  END IF;
  9  RETURN F;
 10  END;
 11  /

Function created.
```

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;

```
SRIT594>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.

SRIT594>insert into department values ('Biology', 'Watson', '900
00');

1 row created.

SRIT594> insert into department values ('Comp. Sci.', 'Taylor',
'100000');

1 row created.

SRIT594> insert into department values ('Elec. Eng.', 'Taylor',
'85000');

1 row created.

SRIT594> insert into department values ('Finance', 'Painter', '1
20000');

1 row created.
```

```
SRIT594>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.

SRIT594>INSERT INTO customers VALUES(1,'Ramesh',23,'Allabad',250
00);

1 row created.

SRIT594> INSERT INTO customers VALUES(2, 'Suresh',22,'Kanpur',27
000);

1 row created.

SRIT594>INSERT INTO customers VALUES(3, 'Mahesh',24,'Ghaziabad',
29000);

1 row created.

SRIT594> INSERT INTO customers VALUES(4, 'chandhan',25,'Noida',3
1000);

1 row created.

SRIT594> INSERT INTO customers VALUES(5, 'Alex', 21, 'paris',330
00);

1 row created.
```

```
SRIT594> 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_di
ff);
 12      END;
 13  /

Trigger created.
```

```
SRIT594>  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.
```

```
SRIT594>DROP trigger  display_salary_changes;

Trigger dropped.
```

16. IMPLEMENT CURSOR

Write PL/SQL program to implement Cursor on table

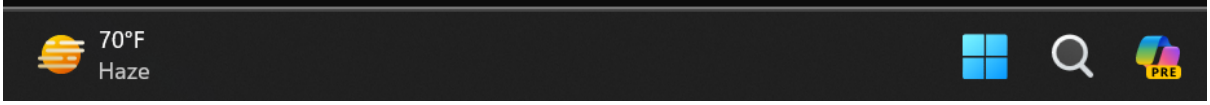
Declare the cursor:
SYNTAX:
CURSOR cursor_name IS select_statement;

Open the cursor
SYNTAX:
OPEN cursor_name;

Fetch the cursor
SYNTAX:
FETCH cursor_name INTO variable_list;

Close the cursor:
SYNTAX:
Close cursor_name;

```
SRIT594>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.

SRIT594>INSERT INTO customers VALUES(1,'Ramesh',23,'Allabad',250
00);

1 row created.

SRIT594> INSERT INTO customers VALUES(2, 'Suresh',22,'Kanpur',27
000);

1 row created.

SRIT594>INSERT INTO customers VALUES(3, 'Mahesh',24,'Ghaziabad',
29000);

1 row created.

SRIT594> INSERT INTO customers VALUES(4, 'chandhan',25,'Noida',3
1000);

1 row created.

SRIT594> INSERT INTO customers VALUES(5, 'Alex', 21, 'paris',330
00);

1 row created.
```



```
SRIT594>  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_addr);
 13         END LOOP;
 14         CLOSE c_customers;
 15      END;
 16      /

PL/SQL procedure successfully completed.
```