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]);

```
  1  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  total_blocks NUMBER,
  8  PRIMARY KEY(clg_id)
  9* )
SQL-CSE553>/

Table created.
```

```
SQL-CSE553>DESC college;
 Name
                   Null?    Type
 ----------------------------------------------------
---------- -------- ------------------------------
 COLLEGE_NAME
                            VARCHAR2(5)
 CLG_ID
                   NOT NULL VARCHAR2(5)
 PLACE
                            VARCHAR2(5)
 STD_STRENGTH
                            NUMBER
 TOTAL_BRANCHES
                            NUMBER
 TOTAL_BLOCKS
                            NUMBER
```

ALTER TABLE :

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

```
SQL-CSE553>ALTER TABLE college
  2  ADD clg_fee NUMBER NOT NULL;

Table altered.

SQL-CSE553>DESC college;
 Name
                   Null?    Type
 ----------------------------------------------------
---------- -------- ------------------------------
 COLLEGE_NAME
                            VARCHAR2(5)
 CLG_ID
                   NOT NULL VARCHAR2(5)
 PLACE
                            VARCHAR2(5)
 STD_STRENGTH
                            NUMBER
 TOTAL_BRANCHES
                            NUMBER
 TOTAL_BLOCKS
                            NUMBER
 CLG_FEE
```

```
SQL-CSE553>ALTER TABLE college
  2  DROP COLUMN total_blocks;

Table altered.

SQL-CSE553>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
```

DROP TABLE:

Deletes the specified table.

SYNTAX:

DROP TABLE table_name;

```
SQL-CSE553>CREATE TABLE products(
  2  p_name VARCHAR(10) NOT NULL,
  3  p_id NUMBER NOT NULL,
  4  PRIMARY KEY(p_id)
  5  );

Table created.

SQL-CSE553>DROP TABLE products;

Table dropped.

SQL-CSE553>DESC products;
ERROR:
ORA-04043: object products does not exist
```

```
SQL-CSE553>ALTER TABLE college
  2  ADD clg_fee NUMBER NOT NULL;

Table altered.
```

```
SQL-CSE553>DESC college;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COLLEGE_NAME                                       VARCHAR2(5)
 CLG_ID                                    NOT NULL VARCHAR2(5)
 PLACE                                              VARCHAR2(5)
 STD_STRENGTH                                       NUMBER
 TOTAL_BRANCHES                                     NUMBER
 TOTAL_BLOCKS                                       NUMBER
 CLG_FEE                                   NOT NULL NUMBER
```

RENAME TABLE:

To rename table_name, column_name SYNTAXES:

RENAME new_table_name TO old_table_name;

```
SQL-CSE553>RENAME college to data;

Table renamed.

SQL-CSE553>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;

```
SQL-CSE553>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 :

```
SQL-CSE553>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);

```
SQL-CSE553>INSERT INTO address(place,pincode,village,district)
  2  VALUES('dmm',515671,'colony','satya sai');

1 row created.

SQL-CSE553>INSERT INTO address(place,pincode,village,district)
  2  VALUES('atp',515672,'nagar','atp');

1 row created.

SQL-CSE553>INSERT INTO address(place,pincode,village,district)
  2  VALUES('nandyal',615898,'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;

```
SQL-CSE553>SELECT * FROM address;

PLACE          PINCODE VILLAGE    DISTRICT
---------- ---------- ---------- ----------
dmm            515671 colony     satya sai
atp            515672 nagar      atp
nandyal        615898 area       kurnool
```

```
SQL-CSE553>SELECT district FROM address;

DISTRICT
----------
satya sai
atp
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];

```
SQL-CSE553>UPDATE address SET village='nijampet' WHERE pincode=615898;

1 row updated.

SQL-CSE553>SELECT * FROM address;

PLACE         PINCODE VILLAGE    DISTRICT
---------- ---------- ---------- ----------
dmm            515671 colony     satya sai
atp            515672 nagar      atp
nandyal        615898 nijampet   kurnool
```

DELETE COMMAND:

To delete all rows or specified rows in a table.

SYNTAX:

DELETE FROM tablename [WHERE condition_list];

```
SQL-CSE553>DELETE from address WHERE place='atp';

1 row deleted.

SQL-CSE553>select * from address;

PLACE         PINCODE VILLAGE    DISTRICT
---------- ---------- ---------- ----------
dmm            515671 colony     satya sai
nandyal        615898 nijampet   kurnool
```

Experiment-3

DBMS

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

Creating a table:

```
SQL-CSE553>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

```
SQL-CSE553>INSERT INTO students VALUES('Jagadeesh',530,'A','CSE',1);

1 row created.

SQL-CSE553>INSERT INTO students VALUES('Sharath',599,'B','CSE',2);

1 row created.

SQL-CSE553>INSERT INTO students VALUES('Dinesh',433,'A','ECE',3);

1 row created.

SQL-CSE553>INSERT INTO students VALUES('VIJAY',389,'B','EEE',4);

1 row created.

SQL-CSE553>INSERT INTO students VALUES('Anees',553,'A','CSE',5);

1 row created.
```

Creating view councellor:

```
SQL-CSE553>CREATE VIEW councellor AS SELECT name,roll_no,id_no FROM students;

View created.
```

Inserting values into councellor:

```
SQL-CSE553>INSERT INTO councellor VALUES('sasi',543,6);

1 row created.

SQL-CSE553>INSERT INTO councellor VALUES('tauheed',547,7);

1 row created.

SQL-CSE553>INSERT INTO councellor VALUES('tarun',550,8);

1 row created.
```

```
SQL-CSE553>SELECT * FROM councellor;

NAME           ROLL_NO      ID_NO
---------- ---------- ----------
Jagadeesh         530          1
Sharath           599          2
Dinesh            433          3
VIJAY             389          4
Anees             553          5
sasi              543          6
tauheed           547          7
tarun             550          8

8 rows selected.
```

Selecting specific row :

```
SQL-CSE553>SELECT * FROM councellor WHERE id_no =4;

NAME           ROLL_NO      ID_NO
---------- ---------- ----------
VIJAY             389          4
```

Update :

```
SQL-CSE553>UPDATE councellor SET name = 'Jagan' WHERE id_no = 1;

1 row updated.

SQL-CSE553>SELECT * FROM councellor;

NAME           ROLL_NO      ID_NO
---------- ---------- ----------
Jagan             530          1
Sharath           599          2
Dinesh            433          3
VIJAY             389          4
Anees             553          5
sasi              543          6
tauheed           547          7
tarun             550          8

8 rows selected.
```

truncate or drop view:

```
SQL-CSE553>DROP VIEW councellor;

View dropped.
```

EXPERIMENT-4

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

Creating tables :

```
SQL-CSE553>CREATE TABLE personal_data(
  2  name VARCHAR(10),
  3  age NUMBER,
  4  gender VARCHAR(10),
  5  job VARCHAR(10),
  6  salary NUMBER,
  7  PRIMARY KEY(name)
  8  );

Table created.

SQL-CSE553>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 :

```
SQL-CSE553>INSERT INTO personal_data VALUES('Jagadeesh',19,'male','student',250000);

1 row created.

SQL-CSE553>INSERT INTO personal_data VALUES('venkat',20,'male','DENTIST',350000);

1 row created.

SQL-CSE553>INSERT INTO personal_data VALUES('basha',18,'male','driver',150000);

1 row created.

SQL-CSE553>INSERT INTO personal_data VALUES('baba',17,'male','owner',350000);

1 row created.
```

Inserting values into **information** table :

```
SQL-CSE553>INSERT INTO information VALUES('baba',509,'CSE',4,'A');

1 row created.

SQL-CSE553>INSERT INTO information VALUES('Jagadeesh',530,'CSE',1,'A');

1 row created.

SQL-CSE553>INSERT INTO information VALUES('arun',507,'CSE',1,'B');

1 row created.

SQL-CSE553>INSERT INTO information VALUES('balaji',510,'CSE',2,'main');

1 row created.

SQL-CSE553>INSERT INTO information VALUES('tauheed',547,'CSE',1,'C');

1 row created.
```

Union operation :

```
SQL-CSE553>SELECT name from personal_data
  2  UNION
  3  SELECT name FROM information;

NAME
----------
Jagadeesh
arun
baba
balaji
basha
tauheed
venkat

7 rows selected.
```

Union all operation :

```
SQL-CSE553>SELECT name from personal_data
  2  UNION ALL
  3  SELECT name FROM information;

NAME
----------
Jagadeesh
venkat
basha
baba
baba
Jagadeesh
arun
balaji
tauheed

9 rows selected.
```

Intersect operation :

```
SQL-CSE553>SELECT name from personal_data
  2  INTERSECT
  3  SELECT name FROM information;

NAME
----------
Jagadeesh
baba
```

Minus operation :

```
SQL-CSE553>SELECT name from personal_data
  2  MINUS
  3  SELECT name FROM information;

NAME
----------
basha
venkat
```

EXPERIMENT-5

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

Creating a table:

```
  1  CREATE TABLE student(
  2  name  VARCHAR(10),
  3  age NUMBER,
  4  subject VARCHAR(15),
  5  marks NUMBER
  6* )
SQL-CSE553>/

Table created.
```

Inserting values into table :

```
SQL-CSE553>INSERT INTO student VALUES('Jagadeesh',19,'maths',30);

1 row created.

SQL-CSE553>INSERT INTO student VALUES('prabhas',20,'oopj',25);

1 row created.

SQL-CSE553>INSERT INTO student VALUES('Jagan',19,'DBMS',20);

1 row created.

SQL-CSE553>INSERT INTO student VALUES('KIRAN',20,'ENGLISH',24);

1 row created.

SQL-CSE553>INSERT INTO student VALUES('Arjun',18,'SE',27);

1 row created.
```

Selecting table :

```
SQL-CSE553>SELECT * FROM student;

NAME                AGE SUBJECT                MARKS
---------- ---------- --------------- ----------
Jagadeesh           19 maths                    30
prabhas             20 oopj                     25
Jagan               19 DBMS                     20
KIRAN               20 ENGLISH                  24
Arjun               18 SE                       27
```

Sum();

```
SQL-CSE553>SELECT SUM(marks) FROM student;

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

Avg();

```
SQL-CSE553>SELECT AVG(marks) FROM student;

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

Min();

```
SQL-CSE553>SELECT MIN(marks) FROM student;

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

Max();

```
SQL-CSE553>SELECT MAX(marks) FROM student;

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

Count();

```
SQL-CSE553>SELECT COUNT(marks) FROM student;

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

EXPERIMENT-5

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

Creating a table

```
  1  CREATE TABLE students_in(
  2  name VARCHAR2(10) NOT NULL,
  3  r_no VARCHAR2(5) NOT NULL,
  4  branch VARCHAR2(5) NULL,
  5  block VARCHAR2(6) NULL,
  6  fee NUMBER NOT NULL,
  7* PRIMARY KEY(name))
SQL-CSE553>/

Table created.
```

Inserting values :

```
SQL-CSE553>INSERT INTO students_in VALUES('Jagadeesh',530,'CSE','B',2500000);

1 row created.

SQL-CSE553>INSERT INTO students_in VALUES('Anees',553,'CSE','B',2200000);

1 row created.

SQL-CSE553>INSERT INTO students_in VALUES('Balaji',510,'CSE','A',2200000);

1 row created.

SQL-CSE553>INSERT INTO students_in VALUES('Baba',509,'CSE','A',2900000);

1 row created.

SQL-CSE553>INSERT INTO students_in VALUES('Tauheed',547,'CSE','A',3500000);

1 row created.
```

```
SQL-CSE553>INSERT INTO students_in VALUES('Tarun',547,'','',3500000);

1 row created.

SQL-CSE553>INSERT INTO students_in VALUES('Mani',549,'','',3900000);

1 row created.

SQL-CSE553>INSERT INTO students_in VALUES('Rehan',554,'','',3900000);

1 row created.
```

Is Null operation :

```
SQL-CSE553>SELECT * from students_in;

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
Jagadeesh  530   CSE   B         2500000
Anees      553   CSE   B         2200000
Balaji     510   CSE   A         2200000
Baba       509   CSE   A         2900000
Tauheed    547   CSE   A         3500000
Tarun      547                   3500000
Mani       549                   3900000
Rehan      554                   3900000

8 rows selected.

SQL-CSE553>SELECT *FROM students_in WHERE branch IS NULL;

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
Tarun      547                   3500000
Mani       549                   3900000
Rehan      554                   3900000
```

Is not null operation :

```
SQL-CSE553>SELECT *FROM students_in WHERE branch IS NOT NULL;

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
Jagadeesh  530   CSE   B         2500000
Anees      553   CSE   B         2200000
Balaji     510   CSE   A         2200000
Baba       509   CSE   A         2900000
Tauheed    547   CSE   A         3500000
```

Between operation :

```
SQL-CSE553>SELECT *FROM students_in WHERE fee BETWEEN 2000000 and 3000000;

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
Jagadeesh  530   CSE   B         2500000
Anees      553   CSE   B         2200000
Balaji     510   CSE   A         2200000
Baba       509   CSE   A         2900000

SQL-CSE553>SELECT *FROM students_in WHERE fee BETWEEN 2500000 and 3500000;

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
Jagadeesh  530   CSE   B         2500000
Baba       509   CSE   A         2900000
Tauheed    547   CSE   A         3500000
Tarun      547                   3500000
```

Like operation:

```
SQL-CSE553>SELECT *FROM students_in WHERE branch LIKE 'CSE%';

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
Jagadeesh  530   CSE   B         2500000
Anees      553   CSE   B         2200000
Balaji     510   CSE   A         2200000
Baba       509   CSE   A         2900000
Tauheed    547   CSE   A         3500000
```

```
SQL-CSE553>SELECT *FROM students_in WHERE bLOCK LIKE 'B%';

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
Jagadeesh  530   CSE   B         2500000
Anees      553   CSE   B         2200000

SQL-CSE553>SELECT *FROM students_in WHERE bLOCK LIKE 'A%';

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
Balaji     510   CSE   A         2200000
Baba       509   CSE   A         2900000
Tauheed    547   CSE   A         3500000
```

Exists operation :

```
SQL-CSE553>SELECT *FROM students_in WHERE EXISTS (SELECT name FROM students_in);

NAME       R_NO  BRANC BLOCK         FEE
---------- ----- ----- ------ ----------
Jagadeesh  530   CSE   B         2500000
Anees      553   CSE   B         2200000
Balaji     510   CSE   A         2200000
Baba       509   CSE   A         2900000
Tauheed    547   CSE   A         3500000
Tarun      547                   3500000
Mani       549                   3900000
Rehan      554                   3900000

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 OUTER JOIN)

CREATING TABLE student :

```
  1  CREATE TABLE student(
  2  name VARCHAR(10),
  3  roll_no NUMBER,
  4  dept VARCHAR(10),
  5  PRIMARY KEY(name)
  6*  )
SQL-CSE553>/

Table created.
```

Inserting tables into student table :

```
SQL-CSE553>INSERT INTO student VALUES ('Jagadeesh',530,'CSE');

1 row created.

SQL-CSE553>INSERT INTO student VALUES ('Bindu',529,'CSE');

1 row created.

SQL-CSE553>INSERT INTO student VALUES ('Jagan',531,'CSE');

1 row created.

SQL-CSE553>INSERT INTO student VALUES ('Arif',506,'CSM');

1 row created.
```

```
SQL-CSE553>SELECT * FROM student;

NAME           ROLL_NO DEPT
---------- ---------- ----------
Jagadeesh         530 CSE
Bindu             529 CSD
Jagan             531 ECE
Arif              506 CSM
```

Creating table Library :

```
SQL-CSE553>CREATE TABLE library(
  2  roll_no NUMBER,
  3  book VARCHAR(10)
  4  );

Table created.
```

Inserting values into library table :

```
SQL-CSE553>INSERT INTO library VALUES (530,'DBMS');

1 row created.

SQL-CSE553>INSERT INTO library VALUES (531,'JAVA');

1 row created.

SQL-CSE553>INSERT INTO library VALUES (537,'MATHS');

1 row created.

SQL-CSE553>INSERT INTO library VALUES (528,'SE');

1 row created.
```

```
SQL-CSE553>SELECT * FROM library;

   ROLL_NO BOOK
---------- ----------
       530 DBMS
       531 JAVA
       537 MATHS
       528 SE
```

CONDITIONAL JOIN :

```
SQL-CSE553>SELECT * FROM student JOIN library on student.roll_no =library.roll_no;

NAME          ROLL_NO DEPT          ROLL_NO BOOK
--------- --------- --------- --------- ----------
Jagadeesh         530 CSE               530 DBMS
Jagan             531 ECE               531 JAVA
```

EQUI JOIN :

```
SQL-CSE553>SELECT * FROM student JOIN library USING (roll_no);

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
       530 Jagadeesh  CSE        DBMS
       531 Jagan      ECE        JAVA
```

NATURAL LEFT OUTER JOIN :

```
SQL-CSE553>SELECT * FROM student NATURAL LEFT OUTER JOIN LIBRARY;

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
       530 Jagadeesh  CSE        DBMS
       531 Jagan      ECE        JAVA
       506 Arif       CSM
       529 Bindu      CSD
```

NATURAL RIGHT OUTER JOIN :

```
SQL-CSE553>SELECT * FROM student NATURAL RIGHT OUTER JOIN LIBRARY;

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
       530 Jagadeesh  CSE        DBMS
       531 Jagan      ECE        JAVA
       528                       SE
       537                       MATHS
```

NATURAL FULL OUTER JOIN :

```
SQL-CSE553>SELECT * FROM student NATURAL FULL OUTER JOIN LIBRARY;

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
       530 Jagadeesh  CSE        DBMS
       531 Jagan      ECE        JAVA
       537                       MATHS
       528                       SE
       506 Arif       CSM
       529 Bindu      CSD
```

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 :

```
SQL-CSE553>CREATE TABLE  names(
  2  first_name VARCHAR(20) NOT NULL,
  3  last_name VARCHAR(20) NOT NULL
  4  );

Table created.
```

INSERTING VALUES :

```
SQL-CSE553>INSERT ALL
  2  INTO names VALUES('Jagadeesh','steeve')
  3  INTO names VALUES('Balaji','guduru')
  4  INTO names VALUES('Akhil','akkineni')
  5  INTO names VALUES('nani','nandamuri')
  6  SELECT *FROM dual;

4 rows created.
```

1. Character Functions

I. Case-conversion functions :

LOWER ();

```
SQL-CSE553>SELECT LOWER(first_name) FROM names;

LOWER(FIRST_NAME)
--------------------
jagadeesh
balaji
akhil
nani
```

UPPER();

```
SQL-CSE553>SELECT UPPER(first_name) FROM names;

UPPER(FIRST_NAME)
--------------------
JAGADEESH
BALAJI
AKHIL
NANI
```

INITCAP();

```
SQL-CSE553>SELECT INITCAP(first_name) FROM names;

INITCAP(FIRST_NAME)
--------------------
Jagadeesh
Balaji
Akhil
Nani
```

Character manipulation functions:

CONCAT():

```
SQL-CSE553>SELECT CONCAT(first_name,last_name) FROM names;

CONCAT(FIRST_NAME,LAST_NAME)
------------------------------------
Jagadeeshsteeve
Balajiguduru
Akhilakkineni
naninandamuri
```

SUBSTR():

```
SQL-CSE553>SELECT SUBSTR(first_name,1,4) FROM names;

SUBSTR(FIRST_NAM
----------------
Jaga
Bala
Akhi
nani
```

LENGTH() :

```
SQL-CSE553>SELECT LENGTH(first_name) FROM names;

LENGTH(FIRST_NAME)
------------------
                 9
                 6
                 5
                 4
```

INSTR() :

```
SQL-CSE553>SELECT INSTR(first_name,'Ja') FROM names;

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

TRIM() :

```
SQL-CSE553>SELECT TRIM('A' FROM first_name) FROM names;

TRIM('A'FROMFIRST_NA
--------------------
Jagadeesh
Balaji
khil
nani
```

2. Number Functions :

ROUND() :

```
SQL-CSE553>SELECT ROUND(11.111,2) FROM dual;

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

MOD() :

```
SQL-CSE553>SELECT MOD(11,2) FROM dual;

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

2.DATE functions :

SYSDATE()

```
SQL-CSE553>SELECT SYSDATE FROM dual;

SYSDATE
---------
13-DEC-23
```

MONTHS-BETWEEN() :

```
SQL-CSE553>SELECT MONTHS_BETWEEN(SYSDATE,'13-DEC-2024') FROM dual;

MONTHS_BETWEEN(SYSDATE,'13-DEC-2024')
-------------------------------------
                                  -12
```

ADD_MONTHS() :

```
SQL-CSE553>SELECT ADD_MONTHS(SYSDATE,12) FROM dual;

ADD_MONTH
---------
13-DEC-24
```

NEXT_DAY():

```
SQL-CSE553>SELECT NEXT_DAY(SYSDATE,'MONDAY') FROM dual;

NEXT_DAY(
---------
18-DEC-23
```

LAST_DAY() :

```
SQL-CSE553>SELECT LAST_DAY(SYSDATE) FROM dual;

LAST_DAY(
---------
31-DEC-23
```

```
SQL-CSE553>SELECT CURRENT_TIMESTAMP(3) FROM dual;

CURRENT_TIMESTAMP(3)
---------------------------------------------------------------------------
13-DEC-23 01.50.32.927 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:

```
SQL-CSE553>CREATE TABLE  order1(
  2  id NUMBER PRIMARY KEY,
  3  product_name VARCHAR2(50) NOT NULL,
  4  quantity NUMBER
  5  );

Table created.

SQL-CSE553>INSERT INTO order1 VALUES(1,'AGARBATHI',30);

1 row created.

SQL-CSE553>INSERT INTO order1 VALUES(4,'',30);
INSERT INTO order1 VALUES(4,'',30)
                          *
ERROR at line 1:
ORA-01400: cannot insert NULL into ("CSE530"."ORDER1"."PRODUCT_NAME")
```

2.UNIQUE CONSTRAINT Example:

```
SQL-CSE553>CREATE TABLE employees (
  2  id NUMBER PRIMARY KEY,
  3  name VARCHAR(50) NOT NULL,
  4  e_mail VARCHAR2(50) UNIQUE
  5  );

Table created.

SQL-CSE553>INSERT INTO employees VALUES(530,'Jagadeesh','jagadeesh8897@gmail.com');

1 row created.
```

3.PRIMARY KEY CONSTRAINT Example:

```
SQL-CSE553>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.
```

```
SQL-CSE553>INSERT INTO stud VALUES(530,'HARRY','POTTER');

1 row created.
```

4.FORIEGN KEY CONSTRAINTS Example:

```
SQL-CSE553>CREATE TABLE orders(
  2  id NUMBER PRIMARY KEY,
  3  order_num NUMBER NOT NULL,
  4  stud_id NUMBER REFERENCES stud(id)
  5  );

Table created.

SQL-CSE553>INSERT INTO orders VALUES(11,2,111);
INSERT INTO orders VALUES(11,2,111)
*
ERROR at line 1:
ORA-02291: integrity constraint (CSE530.SYS_C007076) violated - parent key not
found
```

5.CHECK CONSTRAINTS Example:

```
SQL-CSE553>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.

SQL-CSE553>INSERT INTO parts1 VALUES(1,'AGARBATHI',897);

1 row created.

SQL-CSE553>INSERT INTO parts1 VALUES(1,'AGARBATHI',-897);
INSERT INTO parts1 VALUES(1,'AGARBATHI',-897)
*
ERROR at line 1:
ORA-02290: check constraint (CSE530.SYS_C007083) violated
```

6.DEFAULT CONSTRAINTS Example:

```
SQL-CSE553>CREATE TABLE customers1 (
  2  name VARCHAR2(50) NOT NULL,
  3  id NUMBER PRIMARY KEY,
  4  country VARCHAR2(20) DEFAULT 'IND'
  5  );

Table created.
```

```
SQL-CSE553>INSERT INTO customers1 VALUES('ARJUN',1,'AUS');

1 row created.

SQL-CSE553>INSERT INTO customers1(name,id) VALUES('allu',2);

1 row created.

SQL-CSE553>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.

```
SQL-CSE553>set serverout on
SQL-CSE553>set verify off
SQL-CSE553>ed
Wrote file afiedt.buf

  1  DECLARE
  2  fac NUMBER :=1;
  3  n NUMBER ;
  4  BEGIN
  5  n := &n;
  6  WHILE n > 0 LOOP
  7  fac:=n*fac;
  8  n:=n-1;
  9  END LOOP;
 10  DBMS_OUTPUT.PUT_LINE(FAC);
 11* END;
SQL-CSE553>/
Enter value for n: 6
720

PL/SQL procedure successfully completed.
```

EXPERIMENT-11

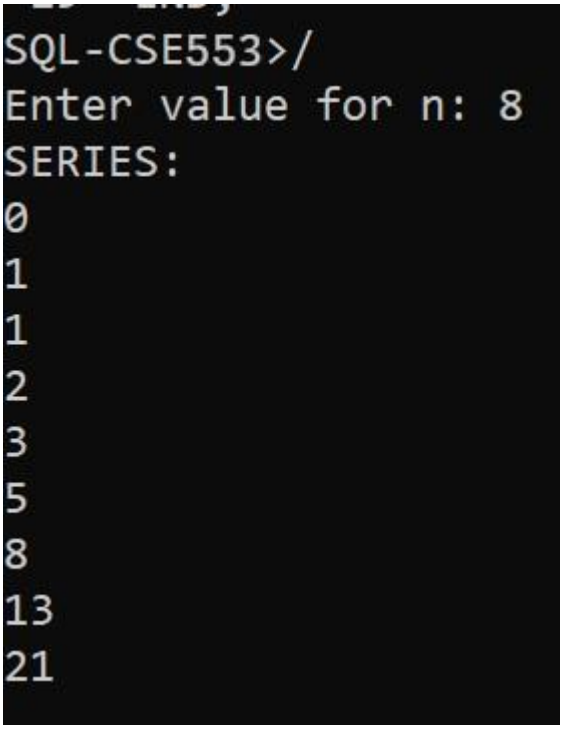
11. Write a PL/SQL program for finding the given number is prime number or not.

```
SQL-CSE553>DECLARE
  2  n NUMBER;
  3  i NUMBER;
  4  temp NUMBER;
  5  BEGIN
  6  n :=&n;
  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  /
Enter value for n: 78
78 is not a prime number

PL/SQL procedure successfully completed.

SQL-CSE553>
```

EXPERIMENT-12

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

```
Wrote file afiedt.buf

  1  DECLARE
  2  FIRST NUMBER := 0;
  3  SECOND NUMBER := 1;
  4  TEMP NUMBER;
  5  N NUMBER ;
  6  I NUMBER;
  7  BEGIN
  8  N := &N;
  9  DBMS_OUTPUT.PUT_LINE('SERIES:');
 10  DBMS_OUTPUT.PUT_LINE(FIRST);
 11  DBMS_OUTPUT.PUT_LINE(SECOND);
 12  FOR I IN 2..N
 13  LOOP
 14  TEMP:=FIRST+SECOND;
 15  FIRST := SECOND;
 16  SECOND := TEMP;
 17  DBMS_OUTPUT.PUT_LINE(TEMP);
 18  END LOOP;
 19* END;
SQL-CSE530>/
Enter value for n: 8
SERIES:
0
```



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-CSE553>CREATE TABLE SAILOR(ID NUMBER(10) PRIMARY KEY,NAME VARCHAR2(100));

Table created.

SQL-CSE553>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;
```

Execution Procedure:

```
SQL> 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.
```

DROP PROCEDURE:

```
SQL> DROP PROCEDURE insertuser;

Procedure dropped.
```

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-CSE553>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:

```
SQL-CSE553>DECLARE
  2  N3 NUMBER(2);
  3  BEGIN
  4  N3 := ADDER(11,22);
  5  DBMS_OUTPUT.PUT_LINE('ADDITION IS: ' || N3);
  6  END;
  7  /
ADDITION IS: 33

PL/SQL procedure successfully completed.
```

```
SQL-CSE553>DROP FUNCTION Adder;

Function dropped.
```

EXAMPLE : 2

```
SQL-CSE553>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.
```

Execution Procedure:

```
SQL-CSE553>DECLARE
  2  num number;
  3  factorial number;
  4  BEGIN
  5  num:= 6;
  6  factorial := fact(num);
  7  dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
  8  END;
  9  /
Factorial 6 is 720

PL/SQL procedure successfully completed.
```

```
SQL-CSE553>DROP FUNCTION fact;

Function dropped.
```

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).

```
SQL-CSE553>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.
```

```
SQL-CSE553>insert into department values ('Biology', 'Watson', '90000');

1 row created.

SQL-CSE553>insert into department values ('Comp. Sci.', 'Taylor', '100000');

1 row created.

SQL-CSE553>insert into department values ('Elec. Eng.', 'Taylor', '85000');

1 row created.

SQL-CSE553>insert into department values ('Finance', 'Painter', '120000');

1 row created.

SQL-CSE553>insert into department values ('History', 'Painter', '50000');

1 row created.

SQL-CSE553>insert into department values ('Music', 'Packard', '80000');

1 row created.
```

CREATING DEPARTMENT TABLE :

```
SQL-CSE553>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.
```

An example to create Trigger :

```
SQL-CSE553>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.
```

A PL/SQL Procedure to execute a trigger:

```
SQL-CSE553>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  /
no instructors updated

PL/SQL procedure successfully completed.
```

Experiment-16

Aim : To write PL/SQL program to implement Cursor on table.

Table Creation :

```
SQL-CSE553>INSERT ALL
  2  INTO people VALUES(1,'jaga',23,800000)
  3  INTO people VALUES(2,'asif',32,700000)
  4  INTO people VALUES(3,'vijay',26,650000)
  5  INTO people VALUES(4,'Siva',35,4000000)
  6  SELECT * FROM dual;

4 rows created.
```

Instances of people :

```
SQL-CSE553>CREATE TABLE people(
  2  id number PRIMARY KEY,
  3  name VARCHAR2(30) NOT NULL,
  4  age NUMBER(3) NOT NULL,
  5  salary NUMBER(10,2) NOT NULL
  6  );

Table created.
```

Create update procedure Create

procedure:

```
SQL-CSE553>DECLARE
  2  total_rows number(2);
  3  BEGIN
  4  UPDATE people
  5  SET salary = salary + 5000;
  6  IF sql%notfound THEN
  7  dbms_output.put_line('no customers updated');
  8  ELSIF sql%found THEN
  9  total_rows := sql%rowcount;
 10  dbms_output.put_line( total_rows || ' customers updated ');
 11  END IF;
 12  END;
 13  /
no customers updated

PL/SQL procedure successfully completed.
```

PL/SQL Program using Explicit Cursors :

```
SQL-CSE553>ed
Wrote file afiedt.buf

  1  DECLARE
  2  p_id people.id%type;
  3  p_name people.name%type;
  4  p_age people.age%type;
  5  CURSOR p_people IS
  6  SELECT id,name,age FROM people;
  7  BEGIN
  8  OPEN p_people;
  9  LOOP
 10  FETCH p_people into p_id, p_name, p_age;
 11  EXIT WHEN p_people%notfound;
 12  dbms_output.put_line(p_id || ' ' || p_name || ' ' || p_age);
 13  END LOOP;
 14  CLOSE p_people;
 15* END;
SQL-CSE553>/
1 jaga 23
2 asif 32
3 vijay 26
4 Siva 35
```