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]);

```
SQL> CREATE TABLE employees (
  2      employee_id INT PRIMARY KEY,
  3      first_name VARCHAR(50),
  4      last_name VARCHAR(50),
  5      job_title VARCHAR(100),
  6      hire_date DATE,
  7      salary DECIMAL(10, 2)
  8  );

Table created.
```

```
SQL> DESC employees;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 EMPLOYEE_ID                               NOT NULL NUMBER(38)
 FIRST_NAME                                         VARCHAR2(50)
 LAST_NAME                                          VARCHAR2(50)
 JOB_TITLE                                          VARCHAR2(100)
 HIRE_DATE                                          DATE
 SALARY                                             NUMBER(10,2)
```

ALTER TABLE :

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

```
SQL> ALTER TABLE employees
  2  ADD email VARCHAR(100);

Table altered.

SQL> DESC employees;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 EMPLOYEE_ID                               NOT NULL NUMBER(38)
 FIRST_NAME                                         VARCHAR2(50)
 LAST_NAME                                          VARCHAR2(50)
 JOB_TITLE                                          VARCHAR2(100)
 HIRE_DATE                                          DATE
 SALARY                                             NUMBER(10,2)
 EMAIL                                              VARCHAR2(100)
```

```
SQL> ALTER TABLE employees
  2  DROP COLUMN hire_date;

Table altered.

SQL> DESC employees;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 EMPLOYEE_ID                               NOT NULL NUMBER(38)
 FIRST_NAME                                         VARCHAR2(50)
 LAST_NAME                                          VARCHAR2(50)
 JOB_TITLE                                          VARCHAR2(100)
 SALARY                                             NUMBER(10,2)
 EMAIL                                              VARCHAR2(100)
```

DROP TABLE:

Deletes the specified table.

SYNTAX:

DROP TABLE table_name;

```
SQL> CREATE TABLE example_table (
  2       id INT PRIMARY KEY,
  3       name VARCHAR(50),
  4       date_of_birth DATE
  5  );

Table created.

SQL> DROP TABLE example_table;

Table dropped.

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

RENAME TABLE:

To rename table_name, column_name SYNTAXES:

RENAME new_table_name TO old_table_name;

```
SQL> RENAME employees to employee;

Table renamed.

SQL> desc employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 EMPLOYEE_ID                               NOT NULL NUMBER(38)
 FIRST_NAME                                         VARCHAR2(50)
 LAST_NAME                                          VARCHAR2(50)
 JOB_TITLE                                          VARCHAR2(100)
 SALARY                                             NUMBER(10,2)
 EMAIL                                              VARCHAR2(100)
```

TRUNCATE TABLE:

To remove all rows in a specified table.

SYNTAX:

TRUNCATE TABLE table_name;

```
SQL> TRUNCATE TABLE employee;

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> CREATE TABLE student1 (
  2    id INT PRIMARY KEY,
  3       name VARCHAR(50),
  4       age INT,
  5       email VARCHAR(100),
  6    registration_date NUMBER
  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> INSERT INTO student1 VALUES(2,'parvathi',19,'parvathi@gmail.com',2023-2-2);

1 row created.

SQL> INSERT INTO student1 VALUES(3,'naveen',17,'naveen@gmail.com',2023-3-3);

1 row created.

SQL> INSERT INTO student1 VALUES(4,'kavya',17,'kavya@gmail.com',2023-4-4);

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> select * FROM student1;

        ID NAME                                               AGE
---------- -------------------------------------------------- ----------
EMAIL
--------------------------------------------------------------------------------
REGISTRATION_DATE
-----------------
         1 bindu                                              18
bindu@gmail.com
             2021

         2 parvathi                                           19
parvathi@gmail.com
             2019
```

```
        ID NAME                                                AGE
---------- -------------------------------------------------- ----------
EMAIL
--------------------------------------------------------------------------------
REGISTRATION_DATE
-----------------

         3 naveen                                              17
naveen@gmail.com
              2017

         4 kavya                                               17
kavya@gmail.com
```

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> UPDATE student1 SET name='shiva' WHERE age=19;

1 row updated.

SQL> SELECT * FROM student1;

        ID NAME                                            AGE
---------- -------------------------------------------------- ----------
EMAIL
--------------------------------------------------------------------------------
REGISTRATION_DATE
-----------------
         1 bindu                                            18
bindu@gmail.com
              2021

         2 shiva                                            19
parvathi@gmail.com
              2019
```

DELETE COMMAND:

To delete all rows or specified rows in a table.

SYNTAX:

DELETE FROM tablename [WHERE condition_list];

```
SQL> DELETE from student1 WHERE age=18;

1 row deleted.

SQL> select * from student1;

        ID NAME                                                AGE
---------- -------------------------------------------------- ----------
EMAIL
--------------------------------------------------------------------------------
REGISTRATION_DATE
-----------------
         2 shiva                                              19
parvathi@gmail.com
             2019

         3 naveen                                             17
naveen@gmail.com
             2017
```

Experiment-3

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

Creating a table:

```
SQL> CREATE TABLE student2 (
  2    id INT PRIMARY KEY,
  3    name VARCHAR(50),
  4    age INT,
  5    email VARCHAR(100),
  6    registration_date NUMBER,
  7    grade VARCHAR(10)
  8  );

Table created.
```

By using insert command we can insert values in a tables

```
SQL> INSERT INTO student2 VALUES(1,'bindu',12,'bindu@gmail.com',2023-01-01,'A');

1 row created.

SQL> INSERT INTO student2 VALUES(2,'preethi',13,'preethi@gmail.com',2023-02-02,'B');

1 row created.

SQL> INSERT INTO student2 VALUES(3,'pooji',13,'pooji@gmail.com',2023-02-03,'C');

1 row created.

SQL> INSERT INTO student2 VALUES(4,'suppi',14,'suppi@gmail.com',2023-02-04,'D');

1 row created.

SQL> INSERT INTO student2 VALUES(5,'usha',15,'usha@gmail.com',2023-02-05,'E');

1 row created.
```

Creating view councellor:

```
SQL> CREATE VIEW counsellor AS SELECT name,age,grade FROM student2;

View created.
```

Inserting values into councellor:

```
SQL> INSERT INTO student2 VALUES(1,'bindu',12,'bindu@gmail.com',2023-01-01,'A');

1 row created.

SQL> INSERT INTO student2 VALUES(2,'preethi',13,'preethi@gmail.com',2023-02-02,'B');

1 row created.

SQL> INSERT INTO student2 VALUES(3,'pooji',13,'pooji@gmail.com',2023-02-03,'C');

1 row created.

SQL> INSERT INTO student2 VALUES(4,'suppi',14,'suppi@gmail.com',2023-02-04,'D');

1 row created.

SQL> INSERT INTO student2 VALUES(5,'usha',15,'usha@gmail.com',2023-02-05,'E');

1 row created.
```

Selecting specific row :

```
SQL> SELECT * FROM counsellor;

NAME                                               AGE GRADE
-------------------------------------------------- ---------- -------
bindu                                               12 A
preethi                                             13 B
pooji                                               13 C
suppi                                               14 D
usha                                                15 E
```

Update :

```
SQL> UPDATE counsellor SET name = 'samyu' WHERE age=12;

1 row updated.

SQL> SELECT * FROM counsellor;

NAME                                               AGE GRADE
-------------------------------------------------- ---------- ------
samyu                                               12 A
preethi                                             13 B
pooji                                               13 C
suppi                                               14 D
usha                                                15 E
```

truncate or drop view:

```
SQL> DROP VIEW counsellor;

View dropped.
```

EXPERIMENT-4

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

Creating tables :

```
SQL> CREATE TABLE students3 (
  2    student_id INT PRIMARY KEY,
  3    name VARCHAR(50),
  4    age INT,
  5  email VARCHAR(100),
  6    registration_date NUMBER
  7  );

Table created.
```

```
SQL> CREATE TABLE courses1 (
  2    course_id INT PRIMARY KEY,
  3    course_name VARCHAR(50),
  4    instructor VARCHAR(50),
  5    start_date NUMBER,
  6    end_date NUMBER
  7  );

Table created.
```

Inserting values into **personal_data** table :

```
SQL> INSERT INTO students3 VALUES(1,'jaggu',10,'jaggu@gmail.com',2023-10-10);

1 row created.

SQL> INSERT INTO students3 VALUES(2,'govardhan',10,'govardhan@gmail.com',2023-10-11);

1 row created.

SQL> INSERT INTO students3 VALUES(3,'kutty',11,'kutty@gmail.com',2023-10-12);

1 row created.

SQL> INSERT INTO students3 VALUES(4,'sonu',12,'sonu@gmail.com',2023-10-13);

1 row created.
```

Inserting values into **information** table :

```
SQL> INSERT INTO courses1 VALUES(11,'cse','shiva',2023-10-13,2023-10-30);

1 row created.

SQL> INSERT INTO courses1 VALUES(12,'csd','shamu',2023-10-14,2023-11-30);

1 row created.

SQL> INSERT INTO courses1 VALUES(13,'csm','sharun',2023-10-15,2023-11-28);

1 row created.

SQL> INSERT INTO courses1 VALUES(14,'eee','shonn',2023-10-16,2023-11-27);

1 row created.

SQL> INSERT INTO courses1 VALUES(15,'ece','shony',2023-10-18,2023-11-23);

1 row created.
```

Union operation :

```
SQL> SELECT name from students3
  2  UNION
  3  SELECT course_name from courses1;

NAME
------------------------------------
csd
cse
csm
ece
eee
govardhan
jaggu
kutty
sonu

9 rows selected.
```

Union all operation :

```
SQL> SELECT name from students3
  2  UNION ALL
  3  SELECT course_name from courses1;

NAME
--------------------------------------------------
jaggu
govardhan
kutty
sonu
cse
csd
csm
eee
ece

9 rows selected.
```

Intersect operation :

```
SQL> SELECT name from students3
  2  INTERSECT
  3  SELECT course_name from courses1;

no rows selected
```

Minus operation :

```
SQL> SELECT name from students3
  2  MINUS
  3  SELECT course_name from courses1;

NAME
--------------------------------------------------
govardhan
jaggu
kutty
sonu
```

EXPERIMENT-5

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

Creating a table:

```
SQL> CREATE TABLE students4 (
  2      student_id INT PRIMARY KEY,
  3      first_name VARCHAR(50),
  4      last_name VARCHAR(50),
  5    phone_number VARCHAR(15),
  6      address VARCHAR(255)
  7  );

Table created.
```

Inserting values into table :

```
SQL> INSERT INTO students4 VALUES(1,'Y','bindu',123456,'atp');

1 row created.

SQL> INSERT INTO students4 VALUES(2,'k','jyothi',123478,'ktc');

1 row created.

SQL> INSERT INTO students4 VALUES(3,'A','usha',123409,'tdp');

1 row created.

SQL> INSERT INTO students4 VALUES(4,'u','suppi',123402,'amp');

1 row created.
```

Selecting table :

```
STUDENT_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                          PHONE_NUMBER
-------------------------------------------------- ---------------
ADDRESS
--------------------------------------------------------------------------------
         1 Y
bindu                                              123456
atp

         2 k
jyothi                                             123478
ktc
```

Sum();

```
SQL> SELECT SUM(student_id) FROM students4;

SUM(STUDENT_ID)
---------------
             15
```

Avg();

```
SQL> SELECT AVG(student_id) FROM students4;

AVG(STUDENT_ID)
---------------
              3
```

Min();

```
SQL> SELECT MIN(student_id) FROM students4;

MIN(STUDENT_ID)
---------------
              1
```

Max();

```
SQL> SELECT MAX(student_id) FROM students4;

MAX(STUDENT_ID)
---------------
              5
```

Count();

```
SQL> SELECT COUNT(student_id) FROM students4;

COUNT(STUDENT_ID)
-----------------
                5
```

EXPERIMENT-6

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 :

```
SQL> CREATE TABLE student10(
  2  name varchar(10),
  3  roll_no number,
  4  dept varchar(10),
  5  primary key(name)
  6  );

Table created.
```

Inserting tables into student table :

```
SQL> INSERT INTO student10 VALUES('SHIVA',531,'CSE');

1 row created.

SQL> INSERT INTO student10 VALUES('MOUNIKA',532,'CSE');

1 row created.
```

```
SQL> SELECT * FROM student10;

NAME          ROLL_NO DEPT
---------- ---------- ----------
SHIVA             531 CSE
MOUNIKA           532 CSE
JAGAN             530 CSE
ARJUN             505 CSE
```

Creating table Library :

```
SQL> CREATE TABLE library(
  2  roll_no number,
  3  book varchar(10)
  4  );

Table created.
```

Inserting values into library table :

```
SQL-CSE530>INSERT INTO library VALUES (530,'DBMS');

1 row created.

SQL-CSE530>INSERT INTO library VALUES (531,'JAVA');

1 row created.

SQL-CSE530>INSERT INTO library VALUES (537,'MATHS');

1 row created.

SQL-CSE530>INSERT INTO library VALUES (528,'SE');

1 row created.
```

```
SQL-CSE530>SELECT * FROM library;

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

CONDITIONAL JOIN :

```
SQL> SELECT * FROM student10 JOIN library on student10.roll_no = library.roll_no;

NAME          ROLL_NO DEPT          ROLL_NO BOOK
---------- ---------- ---------- ---------- ----------
SHIVA            531 CSE              531 DBMS
MOUNIKA          532 CSE              532 JAVA
JAGAN            530 CSE              530 MATHS
ARJUN            505 CSE              505 SE
```

EQUI JOIN :

```
SQL> SELECT * FROM student10 JOIN library USING (roll_no);

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
       531 SHIVA      CSE        DBMS
       532 MOUNIKA    CSE        JAVA
       530 JAGAN      CSE        MATHS
       505 ARJUN      CSE        SE
```

NATURAL LEFT OUTER JOIN :

```
SQL> SELECT * FROM student10 NATURAL LEFT OUTER JOIN library ;

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
       531 SHIVA      CSE        DBMS
       532 MOUNIKA    CSE        JAVA
       530 JAGAN      CSE        MATHS
       505 ARJUN      CSE        SE
```

NATURAL RIGHT OUTER JOIN :

```
SQL> SELECT * FROM student10 NATURAL RIGHT OUTER JOIN library ;

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
       531 SHIVA      CSE        DBMS
       532 MOUNIKA    CSE        JAVA
       530 JAGAN      CSE        MATHS
       505 ARJUN      CSE        SE
```

NATURAL FULL OUTER JOIN :

```
SQL> SELECT * FROM student10 NATURAL FULL OUTER JOIN library ;

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
       531 SHIVA      CSE        DBMS
       532 MOUNIKA    CSE        JAVA
       530 JAGAN      CSE        MATHS
       505 ARJUN      CSE        SE
```

EXPERIMENT-7

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

Creating a table

```
SQL> CREATE TABLE students6 (
  2   student_id INT PRIMARY KEY,
  3       first_name VARCHAR(50),
  4       last_name VARCHAR(50),
  5       date_of_birth NUMBER,
  6       gender CHAR(1),
  7       email VARCHAR(100)
  8  );

Table created.
```

Inserting values :

```
SQL> INSERT INTO students6 VALUES(1,'A','bindu',2023-02-01,'f','bindu@gmail.com');

1 row created.

SQL> INSERT INTO students6 VALUES(2,'B','kutty',2023-02-02,'f','kutty@gmail.com');

1 row created.

SQL> INSERT INTO students6 VALUES(3,'c','sonu',2023-03-02,'f','sonu@gmail.com');

1 row created.

SQL> INSERT INTO students6 VALUES(4,'d','sunny',2023-03-03,'m','sunny@gmail.com');

1 row created.

SQL> INSERT INTO students6 VALUES(5,'e','sandeep',2023-03-07,'m','sandeep@gmail.com');

1 row created.

SQL> INSERT INTO students6 VALUES(6,'f','netra',2023-03-08,'f','netra@gmail.com');

1 row created.

SQL> INSERT INTO students6 VALUES(7,'g','abhi',2023-06-08,'m','abhi@gmail.com');

1 row created.
```

Is Null operation :

```
SQL> SELECT * from students6;

STUDENT_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                          DATE_OF_BIRTH G
-------------------------------------------------- ------------ -
EMAIL
--------------------------------------------------------------------------------
         1 A
bindu                                                    2020 f
bindu@gmail.com

         2 B
kutty                                                    2019 f
kutty@gmail.com

STUDENT_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                          DATE_OF_BIRTH G
-------------------------------------------------- ------------ -
EMAIL
--------------------------------------------------------------------------------

         3 c
sonu                                                     2018 f
sonu@gmail.com

         4 d
sunny                                                    2017 m
```

```
SQL> SELECT * FROM students6 WHERE gender IS NULL;

no rows selected
```

Is not null operation :

```
SQL> SELECT * FROM students6 WHERE gender IS NOT NULL;

STUDENT_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                          DATE_OF_BIRTH G
-------------------------------------------------- ------------ -
EMAIL
--------------------------------------------------------------------------------
         1 A
bindu                                                    2020 f
bindu@gmail.com

         2 B
kutty                                                    2019 f
kutty@gmail.com
```

Between operation :

```
SQL> SELECT * FROM students6 WHERE student_id BETWEEN 1 and 5;

STUDENT_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                          DATE_OF_BIRTH G
-------------------------------------------------- ------------ -
EMAIL
--------------------------------------------------------------------------------
         1 A
bindu                                                  2020 f
bindu@gmail.com

         2 B
kutty                                                  2019 f
kutty@gmail.com

STUDENT_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                          DATE_OF_BIRTH G
-------------------------------------------------- ------------ -
EMAIL
--------------------------------------------------------------------------------

         3 c
sonu                                                   2018 f
sonu@gmail.com

         4 d
sunny                                                  2017 m
```

Like operation:

```
SQL-CSE530>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> SELECT * FROM students6 WHERE last_name LIKE 'sunny%';

STUDENT_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                          DATE_OF_BIRTH G
-------------------------------------------------- ------------ -
EMAIL
--------------------------------------------------------------------------------
         4 d
sunny                                                  2017 m
sunny@gmail.com
```

Exists operation :

```
SQL> SELECT * FROM students6 WHERE EXISTS (SELECT last_name FROM students6);

STUDENT_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                          DATE_OF_BIRTH G
-------------------------------------------------- ------------- -
EMAIL
--------------------------------------------------------------------------------
         1 A
bindu                                                    2020 f
bindu@gmail.com

         2 B
kutty                                                    2019 f
kutty@gmail.com
```

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**
4. **CREATING TABLE :**

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

Table created.
```

INSERTING VALUES :

```
SQL> INSERT ALL
  2  INTO names VALUES('bindu','chitran')
  3  INTO names VALUES('preethi','reddy')
  4  INTO names VALUES('pooji','gattamaneni')
  5  INTO names VALUES('kavya','battini')
  6  SELECT * FROM dual;

4 rows created.
```

Character Functions

I. Case-conversion functions :

LOWER ();

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

LOWER(FIRST_NAME)
--------------------
bindu
preethi
pooji
kavya
```

UPPER();

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

UPPER(FIRST_NAME)
--------------------
BINDU
PREETHI
POOJI
KAVYA
```

INITCAP();

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

INITCAP(FIRST_NAME)
--------------------
Bindu
Preethi
Pooji
Kavya
```

Character manipulation functions:

CONCAT():

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

CONCAT(FIRST_NAME,LAST_NAME)
----------------------------------------
binduchitran
preethireddy
poojigattamaneni
kavyabattini
```

SUBSTR():

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

SUBSTR(FIRST_NAM
----------------
bind
pree
pooj
kavy
```

LENGTH() :

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

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

INSTR() :

```
SQL> SELECT INSTR(first_name,'KA') FROM names;

INSTR(FIRST_NAME,'KA')
----------------------
                     0
                     0
                     0
                     0
```

TRIM() :

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

TRIM('A'FROMFIRST_NA
--------------------
bindu
preethi
pooji
kavya
```

2. Number Functions :

ROUND() :

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

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

MOD() :

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

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

2.DATE functions :

SYSDATE()

```
SQL> SELECT SYSDATE FROM dual;

SYSDATE
---------
19-DEC-23
```

MONTHS-BETWEEN() :

```
SQL> SELECT MONTHS_BETWEEN(SYSDATE,'19-DEC-23') FROM dual;

MONTHS_BETWEEN(SYSDATE,'19-DEC-23')
-----------------------------------
                                  0
```

ADD_MONTHS() :

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

ADD_MONTH
---------
19-DEC-24
```

NEXT_DAY():

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

NEXT_DAY(
---------
25-DEC-23
```

LAST_DAY() :

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

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

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

CURRENT_TIMESTAMP(3)
---------------------------------------------------------------------------
19-DEC-23 10.41.44.884 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> CREATE TABLE order2(
  2  id NUMBER PRIMARY KEY,
  3  product_name VARCHAR2(50) NOT NULL,
  4  quantity NUMBER
  5  );

Table created.

SQL> INSERT INTO order2 VALUES(1,'AGRABATHI',29);

1 row created.

SQL> INSERT INTO order2 VALUES(4,'',29);
INSERT INTO order2 VALUES(4,'',29)
                          *
ERROR at line 1:
ORA-01400: cannot insert NULL into ("C##513"."ORDER2"."PRODUCT_NAME")
```

2.UNIQUE CONSTRAINT Example:

```
SQL> CREATE TABLE employee1(
  2  id NUMBER PRIMARY KEY,
  3  name VARCHAR(50) NOT NULL,
  4  e_mail VARCHAR2(50) UNIQUE
  5  );

Table created.

SQL> INSERT INTO employee1 VALUES(529,'HimaBindu','himabindu567@gmail.com');


1 row created.
```

3.PRIMARY KEY CONSTRAINT Example:

```
SQL> CREATE TABLE stud1(
  2  ID NUMBER PRIMARY KEY,
  3  first_name VARCHAR(20) NOT NULL,
  4  last_name VARCHAR(20) NOT NULL
  5  );

Table created.

SQL> INSERT INTO stud VALUES(529,'HARRY','POTTER');

1 row created.
```

4.FORIEGN KEY CONSTRAINTS Example:

```
SQL> CREATE TABLE orders3(
  2  id NUMBER PRIMARY KEY,
  3  order_num NUMBER NOT NULL,
  4  stud_id NUMBER REFERENCES stud(id)
  5  );

Table created.

SQL> INSERT INTO orders3 VALUES(11,2,111);
INSERT INTO orders3 VALUES(11,2,111)
*
ERROR at line 1:
ORA-02291: integrity constraint (C##513.SYS_C008386) violated - parent key n
ot
found
```

5.CHECK CONSTRAINTS Example:

```
SQL> CREATE TABLE parts2(
  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> INSERT INTO parts2 VALUES(1,'AGRABATHI',876);

1 row created.

SQL> INSERT INTO parts2 VALUES(1,'AGRABATHI',-876);
INSERT INTO parts2 VALUES(1,'AGRABATHI',-876)
*
ERROR at line 1:
ORA-02290: check constraint (C##513.SYS_C008388) violated
```

6.DEFAULT CONSTRAINTS Example:

```
SQL> CREATE TABLE customers2(
  2  name VARCHAR2(50) NOT NULL,
  3  id NUMBER PRIMARY KEY,
  4  country VARCHAR2(20) DEFAULT 'IND'
  5  );

Table created.
```

```
SQL> INSERT INTO customers2 VALUES('ARYA',1,'USA');

1 row created.

SQL> INSERT INTO customers2(name,id) VALUES('ALLU',2);

1 row created.

SQL> SELECT * FROM customers2;

NAME                                                        ID
---------------------------------------------------- ----------
COUNTRY
--------------------
ARYA                                                         1
USA

ALLU                                                         2
IND
```

Experiment -10

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

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

C:\Users\HP>sqlplus

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

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

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

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

SQL>
```

1.

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

2.

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

3.

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

4.

```
SQL> @ex10
3628800

PL/SQL procedure successfully completed.

SQL>
```

Experiment -11

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

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

C:\Users\HP>sqlplus

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

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

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

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

1.

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

2.

experiment11 - Notepad

File Edit Format View Help

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

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

PL/SQL procedure successfully completed.
```

Experiment -12

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

1.

```
C:\Users\HP>sqlplus

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

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

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

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

```
SQL> SET SERVEROUT ON
SQL> edit experiment12
```

experiment12 - Notepad

File Edit Format View Help

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

2.

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

PL/SQL procedure successfully completed.

SQL>
```

Experiment -13

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

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

Table created.

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

Procedure created.

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

Statement processed.
RECORD INSERTED SUCCESSFULLY
1 RECORD IS INSERTED SUCCESSFULLY

Experiment – 14

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

1.

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

Function created.

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

Statement processed.
ADDITION IS: 33

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

Function created.

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

Statement processed.
Factorial 6 is 720

DROP FUNCTION fact;

Experiment – 15

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

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

Table created.

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

Table created.

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

1 row(s) inserted.

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

Trigger created.

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

Statement processed.
no instructors updated

Experiment – 16

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

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

Table created.

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

1 row(s) inserted.

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

1 row(s) inserted.

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

1 row(s) inserted.

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

Statement processed.
3 customers updated

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

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