1. To Create One or more tables with following Constraints, in addition to the first two Constraint (PK & FK).

a. Check Constraint

b. Unique Constraint

c. Not Null Constraint

```
postgres=# CREATE DATABASE COLLAGE;
postgres=# CEATE TABLE StudInfo (
  id INT PRIMARY KEY NOT NULL,
  name VARCHAR(30) NOT NULL,
  age INT CHECK (age>18),
  address VARCHAR(30),
  fees INT NOT NULL
);

postgres=# INSERT INTO StudInfo(id,name,age,address,fees)
postgres-# VALUES(101,'Sumit',19,Borad,10000),
          (102,'Diptesh',19,Borad,10000), (103,'Aakash',20,'Shahada',12000),
         (104,'Narendra',21,'Shahada',12500), (105,'Ritesh',22,'Dhule',13500);
postgres=# SELECT * FROM StudInfo;

id |   name   | age | address | fees
-----+----------+-----+---------+-------
 101 | Sumit    |  19 | Borad   | 10000
 102 | Diptesh  |  19 | Borad   | 10000
 103 | Aakash   |  20 | Shahada | 12000
 104 | Narendra |  21 | Shahada | 12500
 105 | Ritesh   |  22 | Dhule   | 13500

postgres=# CREATE TABLE Department(Dept_id INT, Dept_name VARCHAR(30) NOT NULL,
Stud_Enroll INTREFERENCE studinfo(id));

INSERT INTO Department(
  Dept_id INT.
  Dept_name NOT NULL,
  Stud_Enroll INTREFERENCE StudInfo(id)
);

postgres=# INSERT INTO Department(Dept_id, Dept_name, Stud_Enroll)
  VALUES(1, ‘Computer’, 200),  (2,’Chemistry, 150), (3, ‘Biotechnology’,100);

postgres=# SELECT * FROM Department;
 dept_id |   dept_name   | stud_enroll
---------+---------------+-------------
       1 | Computer      |         200
       2 | Chemistry     |         150
       3 | Biotechnology |         100
```

2. To Drop a table, alter Schema of a table, Insert / Update / Delete Records using Tables Created in Previous Assignments.
(Use Simple Forms of Insert / Update / Delete Statements)…

```
postgres=# CREATE DATABASE College;
postgres=# CREATE TABLE Student (
  Roll_no INT,
  Name VARCHAR(30), NOT NULL,
  Marks FLOAT(5)
);

postgres=# DROP TABLE Student;
postgres=# CREATE TABLE Student (
  Roll_no INT,
  Name VARCHAR(30), NOT NULL,
  Marks FLOAT(5)
);

postgres=# INSERT INTO Student(Roll_No,Name,Marks)
postgres-# VALUES(101,'Sumit',98.3),(102,'Diptesh',87.6),
        (103,'Aakash',83.2), (104,'Narendra',98.5), (105,'Ritesh',79.2);

postgres=# SELECT * FROM Student;

Roll_No |   Name   | Marks
----------+----------+----------
     101 | Sumit     |  98.3
     102 | Diptesh   |  87.6
     103 | Aakash    |  83.2
     104 | Narendra |  98.5
     105 | Ritesh    |  79.2

postgres=# ALTER TABLE Student ADD COLUMN Address VARCHAR(30);

postgres=# SELECT * FROM Student;
Roll_No |   Name   | Marks  | Addres
----------+----------+---------+----------------
     101 | Sumit     |  98.3   |
     102 | Diptesh   |  87.6   |
     103 | Aakash    |  83.2   |
     101 | Narendra |  98.5   |
     102 | Ritesh    |  79.2   |

postgres=# UPDATE Student SET Address='Shahada' WHERE Roll_No=104;

postgres=# SELECT * FROM Student;
```

```
Roll_No |   Name   | Marks  | Addres
-----------+-----------+---------+----------------
      101 | Sumit      |  98.3    |
      102 | Diptesh   |  87.6    |
      103 | Aakash    |  83.2   |
      101 | Narendra |  98.5   | Shahada
      102 | Ritesh     |  79.2    |
```

```
postgres=# UPDATE Student SET Address=’Borad’ WHERE Roll_No=101;

postgres=# UPDATE Student SET Address=’Borad’ WHERE Roll_No=102;

postgres=# UPDATE Student SET Address=’Shahada’ WHERE Roll_No=103;

postgres=# UPDATE Student SET Address=’Dhule’ WHERE Roll_No=105;

postgres=# SELECT * FROM Student;
```

```
Roll_No |   Name   | Marks  | Addres
-----------+-----------+---------+----------------
      101 | Sumit      |  98.3    | Borad
      102 | Diptesh   |  87.6    | Borad
      103 | Aakash    |  83.2   | Shahada
      101 | Narendra |  98.5   | Shahada
      102 | Ritesh     |  79.2    | Dhule
```

3. To Query the Tables Using Simple Form of Select Statement Select from table [where order by] Select from table [where group by <> having <> order by <>].

```
postgres=# CREATE DATABASE College;

postgres=# CREATE TABLE Student(Roll_No INT PRIMARY KEY, Name VARCHAR(30) NOT
NULL, Marks FLOAT(3), Department VARCHAR(20));

postgres=# INSERT INTO Student(Roll_No, Name, Marks, Department)
VALUES(1,'Sumit',99.9,'Computer'), (2,'Aakash',78.9,'Chemistry'),
 (3,'Narendra',98.9,'Biotechnology'), (4,'Diptesh',88.9,'BCA'), (5,'Ritesh',68.9,'BA');

postgres=# SELECT * FROM Student;
 roll_no |   name   | marks |  department
---------+----------+-------+--------------
       1 | Sumit    |  99.9 | Computer
       2 | Aakash   |  78.9 | Chemistry
       3 | Narendra |  98.9 | Biotechnology
       4 | Diptesh  |  88.9 | BCA
       5 | Ritesh   |  68.9 | BA

postgres=# SELECT * FROM Student WHERE Marks >=70;

 roll_no |   name   | marks |  department
---------+----------+-------+--------------
       1 | Sumit    |  99.9 | Computer
       2 | Aakash   |  78.9 | Chemistry
       3 | Narendra |  98.9 | Biotechnology
       4 | Diptesh  |  88.9 | BCA

postgres=# SELECT * FROM Student ORDER BY Marks ASC;

 roll_no |   name   | marks |  department
---------+----------+-------+--------------
       5 | Ritesh   |  68.9 | BA
       2 | Aakash   |  78.9 | Chemistry
       4 | Diptesh  |  88.9 | BCA
       3 | Narendra |  98.9 | Biotechnology
       1 | Sumit    |  99.9 | Computer

postgres=# SELECT * FROM Student Where Marks>=60 AND Department='BA';
 roll_no |  name  | marks | department
---------+--------+-------+------------
       5 | Ritesh |  68.9 | BA

postgres=# SELECT * FROM Student Where Marks>=60 AND Department='Physics';
 roll_no | name | marks | department
---------+------+-------+------------
```

```
postgres=# SELECT * FROM Student Where Marks>=70 OR Department='Physics';

 roll_no |   name   | marks |  department
---------+----------+-------+--------------
       1 | Sumit    |  99.9 | Computer
       2 | Aakash   |  78.9 | Chemistry
       3 | Narendra |  98.9 | Biotechnology
       4 | Diptesh  |  88.9 | BCA

postgres=# SELECT * FROM Student Where Marks>=70 OR Department='Computer';

 roll_no |   name   | marks |  department
---------+----------+-------+--------------
       1 | Sumit    |  99.9 | Computer
       2 | Aakash   |  78.9 | Chemistry
       3 | Narendra |  98.9 | Biotechnology
       4 | Diptesh  |  88.9 | BCA

postgres=# SELECT COUNT(Department), Department FROM Student WHERE Marks>=70
GROUP BY Department;

 count |  department
-------+--------------
     1 | BCA
     1 | Chemistry
     1 | Biotechnology
     1 | Computer

postgres=# SELECT * FROM Student ORDER BY Marks DESC;

 roll_no |   name   | marks |  department
---------+----------+-------+--------------
       1 | Sumit    |  99.9 | Computer
       3 | Narendra |  98.9 | Biotechnology
       4 | Diptesh  |  88.9 | BCA
       2 | Aakash   |  78.9 | Chemistry
       5 | Ritesh   |  68.9 | BA
```

4. To Query Table, Using Set Operations (Union, Intersection)..

```
postgres=# CREATE DATABASE College;

CREATE TABLE Girls(sr INT, Roll_No INT, Name VARCHAR(30));

postgres=# INSERT INTO Girls(sr,Roll_No, Name) VALUES
(1,101,'Puja'), (2,102,'Jayshri'), (3,103,'Kirti'), (3,104,'Yugal');


postgres=# SELECT * FROM Girls;
 sr | roll_no |  name
----+---------+---------
  1 |     101 | Puja
  2 |     102 | Jayshri
  3 |     103 | Kirti
  3 |     104 | Yugal

postgres=# CREATE TABLE Boys(sr INT, Roll_No INT, Name VARCHAR(30));


postgres=# INSERT INTO Boys(sr,Roll_No, Name) VALUES
          (1,201,'Lalit'), (2,202,'Narendra'), (3,203,'Sumit'), (4,204,'Diptesh');

postgres=# SELECT * FROM Boys;
 sr | roll_no |   name
----+---------+----------
  1 |     201 | Lalit
  2 |     202 | Narendra
  3 |     203 | Sumit
  4 |     204 | Diptesh


postgres=# SELECT Roll_No, Name FROM Girls UNION SELECT Roll_No, Name FROM Boys;
 roll_no |   name
---------+----------
     101 | Puja
     104 | Yugal
     103 | Kirti
     204 | Diptesh
     203 | Sumit
     201 | Lalit
     202 | Narendra
     102 | Jayshri
```

```
postgres=# SELECT Roll_No, Name FROM Girls UNION ALL SELECT Roll_No, Name FROM
Boys;
 roll_no |   name
---------+----------
     101 | Puja
     102 | Jayshri
     103 | Kirti
     104 | Yugal
     201 | Lalit
     202 | Narendra
     203 | Sumit
     204 | Diptesh
```

```
postgres=# SELECT Roll_No,Name FROM Girls INTERSECT SELECT Roll_No,Name FROM
Boys;
 roll_no | name
---------+------
(0 rows)
```

5. To Query Tables Using Nested Queries (Use of ‘Except’, Exists, Not Exists, all Clauses.

```
postgres=# CREATE DATABASE College;

postgres=# CREATE TABLE Table1(Roll_No INT, Name VARCHAR(30), Marks FLOAT(3));

postgres=# INSERT INTO Table1(Roll_No, Name, Marks) VALUES
        (1,'Sumit',98.5), (2,'Diptesh',94.5), (3,'Aakash',74.5), (4,'Narendra',64.5);

postgres=# SELECT * FROM Table1;
 roll_no |   name   | marks
---------+----------+-------
      1 | Sumit    |  98.5
      2 | Diptesh  |  94.5
      3 | Aakash   |  74.5
      4 | Narendra |  64.5

postgres=# CREATE TABLE Table2(Roll_No INT, Name VARCHAR(30), Marks FLOAT(3));

postgres=# INSERT INTO Table2(Roll_No, Name, Marks) VALUES
        (1,'Sumit',98.5), (4,'Narendra',64.5), (5,'Tushar',78.5), (6,'Roshan',58.5);

postgres=# SELECT * FROM Table2;
 roll_no |   name   | marks
---------+----------+-------
      1 | Sumit    |  98.5
      4 | Narendra |  64.5
      5 | Tushar   |  78.5
      6 | Roshan   |  58.5

postgres=# SELECT Roll_No,Name FROM TABLE1 EXCEPT SELECT Roll_No,Name FROM
Table2;
 roll_no |  name
---------+---------
      3 | Aakash
      2 | Diptesh

postgres=# SELECT * FROM Table1 WHERE EXISTS (SELECT * FROM Table2 WHERE
Table1.Roll_No = Table2.Roll_No);
 roll_no |   name   | marks
---------+----------+-------
      1 | Sumit    |  98.5
      4 | Narendra |  64.5
```

```
postgres=# SELECT * FROM Table1 WHERE NOT EXISTS (SELECT * FROM Table2 WHERE
Table1.Roll_No = Table2.Roll_No);
 roll_no |  name   | marks
---------+---------+-------
       2 | Diptesh |  94.5
       3 | Aakash  |  74.5

postgres=# SELECT * FROM Table1 WHERE NOT EXISTS (SELECT * FROM Table2 WHERE
Table2.Roll_No > 2);
 roll_no | name | marks
---------+------+-------

postgres=# SELECT * FROM Table1 WHERE EXISTS (SELECT * FROM Table2 WHERE
Table2.Roll_No > 2);
 roll_no |   name   | marks
---------+----------+-------
       1 | Sumit    |  98.5
       2 | Diptesh  |  94.5
       3 | Aakash   |  74.5
       4 | Narendra |  64.5

postgres=# SELECT Name,Marks FROM Table1 WHERE Marks > ALL (SELECT Marks From
Table2);
 name | marks
------+-------
(0 rows)

postgres=# SELECT Name,Marks FROM Table1 WHERE Marks > ANY (SELECT Marks FROM
Table2);
   name   | marks
----------+-------
 Sumit    |  98.5
 Diptesh  |  94.5
 Aakash   |  74.5
 Narendra |  64.5
```

6. To Create Views.

```
postgres=# CREATE DATABASE College;

postgres=# CREATE TABLE Student(Roll_No INT PRIMARY KEY, Name VARCHAR(30) NOT
NULL, Marks FLOAT(3), Department VARCHAR(20));

postgres=# INSERT INTO Student(Roll_No, Name, Marks, Department) VALUES
          (1,'Sumit',99.9,'Computer'), (2,'Aakash',78.9,'Chemistry'),
  (3,'Narendra',98.9,'Biotechnology'), (4,'Diptesh',88.9,'BCA'), (5,'Ritesh',68.9,'BA');

postgres=# SELECT * FROM Student;
 roll_no |   name   | marks |  department
---------+----------+-------+--------------
       1 | Sumit    |  99.9 | Computer
       2 | Aakash   |  78.9 | Chemistry
       3 | Narendra |  98.9 | Biotechnology
       4 | Diptesh  |  88.9 | BCA
       5 | Ritesh   |  68.9 | BA

postgres=# CREATE VIEW Stud_View AS SELECT Roll_No,Name,Department FROM Student;

postgres=# SELECT * FROM Stud_View;
 roll_no |   name   |  department
---------+----------+--------------
       1 | Sumit    | Computer
       2 | Aakash   | Chemistry
       3 | Narendra | Biotechnology
       4 | Diptesh  | BCA
       5 | Ritesh   | BA

postgres=# ALTER VIEW Stud_View RENAME COLUMN Roll_No TO Id;

postgres=# SELECT * FROM Stud_View;
 id |   name   |  department
----+----------+--------------
  1 | Sumit    | Computer
  2 | Aakash   | Chemistry
  3 | Narendra | Biotechnology
  4 | Diptesh  | BCA
  5 | Ritesh   | BA
```

```
postgres=# ALTER VIEW Stud_View RENAME COLUMN Department To Dept;

postgres=# SELECT * FROM Stud_View;
 id |   name   |     dept
----+----------+--------------
  1 | Sumit    | Computer
  2 | Aakash   | Chemistry
  3 | Narendra | Biotechnology
  4 | Diptesh  | BCA
  5 | Ritesh   | BA

postgres=# DROP VIEW Stud_View;

postgres=# SELECT * FROM Stud_View;

ERROR:  relation "stud_view" does not exist
LINE 1: SELECT * FROM Stud_View;
```

7. To Create Stored Procedure.

- **A Simple Stored Procedure.**
- **A Stored Procedure with IN, OUT and IN/OUT Parameter.**

- **Simple Stored Procedure :**

```
postgres=# CREATE DATABASE MyDataBase;

postgres=# CREATE OR REPLACE PROCEDURE Greet (Name VARCHAR(20)) AS $$
postgres=# BEGIN
postgres=# RAISE NOICE 'Good Morning %', Name;
postgres=# END;
postgres=# $$ LANGUAGE plpgsql;

CREATE PROCESDURE

postgres=# CALL Greet('Lalit Patil');
NOTICE : Good Morning Lalit Patil
CALL

postgres=# CALL Greet('Sumit Patil');
NOTICE : Good Morning Sumit Patil
CALL
```

- **Stored Procedure Using IN and OUT Parameters :**

```
postgres=# CREATE OR REPLACE PROCEDURE Sum_and_Mul (IN a INT, IN b INT, OUT Sum
INT, OUT Mul INT) AS $$

postgres=# BEGIN
postgres=# sum = a + b;
postgres=# mul = a * b;
postgres=# END;
postgres=# LANGUAGE plpgsql;

CREATE PROCEDURE
postgres=# CALL Sum_and_Mul(4,5,4,5);

sum | mul
-----+-------
9    | 20

postgres=# CALL Sum_and_Mul(10,2,10,2);
sum | mul
-----+-------
12    | 20
```

- **Stored Procedure With IN / OUT Parameter.**

```
postgres=# CREATE OR REPLACE PROCEDURE NewMethod(INOUT a INT, INOUT b INT) AS
$$

postgres=# BEGIN
postgres=# a := a + b;
postgres=# b := a * b;
postgres=# END;

postgres=# $$ LANGUAGE plpgsql;

CREATE PROCEDURE

postgres=# CALL NewMethod(8,8);
 a  |  b
----+-------
 16 | 128


postgres=# CALL NewMethod(5,2);
 a  |  b
----+-------
 7  | 14
```

8. Stored Function

- **A Simple Stored Function**
- **A Stored Function that Returns**
- **A Stored Function Recursive**

- **A Simple Stored Function**

```
CREATE OR REPLACE FUNCTION Voting (age INT) RETURNS VOID AS $$
BEGIN
IF (age >= 18) THEN
RAISE NOICE 'You are Eligible to Vote..!';
ELSE
RAISE NOICE 'You are Not Eligible to Vote..!';
END IF;
END;
$$ LANGUAGE plpgsql;

postgres=# SELECT Voting(20);
NOTICE : You are Eligible to Vote..!

postgres=# SELECT Voting(16);
NOTICE : You are Not Eligible to Vote..!
```

- **A Stored Function that Returns**.

```
CREATE OR REPLACE FUNCTION Sum(a INT, b INT) RETURNS INT AS $$
DECLARE
Sum INT;
BEGIN;
Sum := a + b;
RETURN Sum;
END;
$$ LANGUAGE plpgsql;

postgres=# SELECT Sum(10,20);
sum
-------
30

postgres=# SELECT Sum(456,876);
sum
-------
1332
```

- **A Stored Function Recursive.**

```
CREATE OR REPLACE FUNCTION Fact(n INT) RETURNS INT AS $$
BEGIN
IF (n <= 1) THEN
RETURN 1;
ELSE
RETURN n * Fact(n - 1);
END IF;
END;
$$ LANGUAGE plpgsql;

postgres=# SELECT Fact(5);
Fact
-------
120


postgres=# SELECT Fact(8);
Fact
-------
40320
```

9. Cursors

- **A Simple Cursor**
- **A Parameterize Cursor**

```
postgres=# CREATE DATABASE Compony;

postgres=# CREATE TABLE Product(id INT PRIMARY KEY, name VARCHAR(20) NOT NULL,
Category_Id INT NOT NULL, Price INT NOT NULL);

postgres=# INSERT INTO Product(id, name, Category_Id, Price) VALUES
postgres-# (1,'Mouse',1, 100), (2,'KeyBoard',2,200), (3,'Monitor',2,500), (4,'Printer',1,700),
(5,'CPU',1,900);

postgres=# CREATE OR REPLACE FUNCTION Fun_Cursor() RETURNS TEXT LANGUAGE
plpgsql AS $$

DECLARE
Test_Cursor CURSOR FOF
SELECT id, name, price FROM Product;
CurrentId INT;
CurrentProductName VARCHAR(100);
CurrentPrice INT;
BEGIN
OPEN test_Cursor;
LOOP
FETCH Test_Cursor INTO CurrentId, CurrentProductName, CurrentPrice;
EXIT WHEN NOT FOUND;
RAISE NOTICE '% % (ID:%)', CurrentProductName, CurrentPrice, CurrentID;
END LOOP;
CLOSE Test_Cursor;
RETURN 'DONE';
END $$;

SELECT * FROM Fun_Cursor();

NOTICE : Mouse 100 (ID:1)
NOTICE : KeyBoard 200 (ID:2)
NOTICE : Monitor 500 (ID:3)
NOTICE : Printer 700 (ID:4)
NOTICE : CPU 900 (ID:5)
```