```
================================================================================================
Simple Queries:
--------------
Set A:
------

Q1 find name and id of student with second least percentage
SELECT S.FirstName,S.LastName,S.StudentID
from Students S,Integreted_result I
where I.StudentID=S.StudentID and
Final_per=(SELECT min(Final_per) from Integreted_result where Final_per>(SELECT min(Final_per) from Integreted_result));
firstname | lastname | studentid
-----------+----------+-----------
 Kaushal   | Pawar    |   2371002
(1 row)

------------------------------------------------------------------------------------------------------
Q2 find count of Professors holding Ph.D or double Ph.D
SELECT count(P.ProfessorID) from Professors P where P.Qualifications = 'Ph.D' or P.Qualifications='Double Ph.D';
 count
-------
     6
(1 row)

------------------------------------------------------------------------------------------------------
Q3 find Students names and subject whom Professor 'Vikram Joshi' is teaching
select CONCAT(S.FirstName,S.LastName) as Student,CONCAT(C.SubjectName) as Subject
from Students S,Professors P,Course_Details C,Student_course T
where T.StudentID=S.StudentID and P.ProfessorID=C.ProfessorID and C.CourseName=T.Course and P.ProfessorID=6
GROUP BY Student,Subject
HAVING COUNT(S.StudentID)>1;
      student      |    subject
-------------------+---------------
 AkankshaPimparkar | Calculus I
 AkankshaPimparkar | Linear Algebra
 JatinJoshi        | Calculus I
 JatinJoshi        | Linear Algebra
 MahekPatel        | Calculus I
 MahekPatel        | Linear Algebra
(6 rows)

------------------------------------------------------------------------------------------------------
Q4 find subjects taught by each Professors
select CONCAT(P.FirstName) as firstname,CONCAT(P.LastName) as lastname,CONCAT(C.SubjectName) as subject
from Course_Details C,Professors P
where C.ProfessorID=P.ProfessorID group by subject,firstname,lastname having count(P.ProfessorID)>1;

 firstname | lastname |          subject
-----------+----------+---------------------------
 Preeti    | Dave     | World History
```

```
 Amit      | Sharma   | Electronics
 Neelam    | Verma    | rDNA Technology
 Rajiv     | Gupta    | Computer Science
 Neha      | Mishra   | Ecology
 Vikram    | Joshi    | Calculus I
 Vikram    | Joshi    | Linear Algebra
 Smita     | Patil    | Mathematics
 Anjali    | Rajput   | Ancient Civilizations
 Suresh    | Shukla   | Molecular Biology
 Neelam    | Verma    | Statistics
 Neelam    | Verma    | Probability and Statistics
 Rahul     | Kumar    | Cancer Biology
(13 rows)

-----------------------------------------------------------------------------------------------------------
Q5 Find the details of all teachers teaching the subjects in the course "Computer Science".
SELECT P.*
from Professors P,Course_Details C
where P.ProfessorID=C.ProfessorID and SemID='Sem_1' and C.CourseName='Computer Science';
 professorid | firstname | lastname | qualifications  |           email           |  p_number  |   address    | gender
-------------+-----------+----------+-----------------+---------------------------+------------+--------------+--------
           1 | Amit      | Sharma   | Ph.D            | amit.sharma@example.com   | 1234567890 | Shivajinagar | M
           2 | Smita     | Patil    | Double Ph.D     | smita.patil@example.com   | 2345678901 | Aundh        | F
           3 | Rajiv     | Gupta    | Post Graduation | rajiv.gupta@example.com   | 3456789012 | Aundh        | M
           4 | Neelam    | Verma    | Double Ph.D     | neelam.verma@example.com  | 4567890123 | Shivajinagar | F
           5 | Anjali    | Rajput   | Post Graduation | anjali.rajput@example.com | 5678901234 | Shivajinagar | F
(5 rows)

-----------------------------------------------------------------------------------------------------------
Q6 find details of all students whose professor has qualification 'Double Ph.D'
SELECT S.*
from Students S,Course_Details C,Student_course T,Professors P
where P.ProfessorID=C.ProfessorID and C.CourseName=T.Course and T.StudentID=S.StudentID
and P.Qualifications='Double Ph.D'
group by S.StudentID having count(S.StudentID)>1;
 studentid | firstname | lastname  | dateofbirth |           email            |  s_number  | parent_no  |    address    | gender
-----------+-----------+-----------+-------------+----------------------------+------------+------------+---------------+--------
   2371001 | Advait    | Pradhan   | 2000-05-15  | advaitpradhan@example.com | 1234567890 | 9876543210 | ShivajiNagar  | M
   2371002 | Kaushal   | Pawar     | 2002-04-20  | kaushalp@example.com      | 5678901234 | 5432109876 | Akurdi        | M
   2371003 | Isha      | Vaidya    | 2000-09-18  | isha.v@example.com        | 8901234567 | 2109876543 | Sahakarnagar  | F
   2371004 | Shambhavi | Marne     | 2002-12-04  | sham.m@example.com        | 9012345678 | 1230987654 | Kothrud       | F
   2371005 | Riya      | Kedari    | 2001-04-03  | riya.k@example.com        | 5678901234 | 6543210987 | Shanivar Peth | F
   2371008 | Mahek     | Patel     | 2001-02-28  | Mahek.p@example.com       | 2345678901 | 8765432109 | Shanivar Peth | F
   2371009 | Jatin     | Joshi     | 2000-06-30  | jatin.j@example.com       | 3456789012 | 9876543210 | Sahakarnagar  | M
   2371010 | Akanksha  | Pimparkar | 2003-03-29  | akanksha.p@example.com    | 7890123456 | 3210987654 | Sahakarnagar  | F
(8 rows)

-----------------------------------------------------------------------------------------------------------
Q7 find Names of Professors who teaches student 'Advait'
SELECT CONCAT(P.FirstName,P.LastName) as Name
```

```
from Professors P,Students S,Course_Details C,Student_course T
where S.FirstName='Advait' and P.ProfessorID=C.ProfessorID and C.CourseName=T.Course
and T.StudentID=S.StudentID
group by Name having count(P.ProfessorID)>0;
    name
--------------
 AmitSharma
 AnjaliRajput
 NeelamVerma
 RajivGupta
 SmitaPatil
(5 rows)

-----------------------------------------------------------------------------------------------------------
Q8 List the names of all teachers along with the total number of students they are teaching.
SELECT CONCAT(P.FirstName,P.LastName) as name,count(S.StudentID)
from Professors P,Students S,Course_Details C,Student_course T
where P.ProfessorID=C.ProfessorID and C.CourseName=T.Course and T.StudentID=S.StudentID
and SemID = 'Sem_1' group by name having count(S.StudentID)>1;
    name     | count
--------------+-------
 AmitSharma   |     2
 AnjaliRajput |     4
 NeelamVerma  |     8
 NehaMishra   |     6
 PreetiDave   |     2
 RahulKumar   |     3
 RajivGupta   |     2
 SmitaPatil   |     2
 SureshShukla |     3
 VikramJoshi  |     6
(10 rows)

=============================================================================================================================================
Set B:
------

1. Find the names of the students from “MicroBiology” branch.
==>SELECT CONCAT(FirstName,' ',LastName) AS StudentName
FROM Students S
JOIN Student_course SC
ON S.StudentID = SC.StudentID

   studentname
-----------------
 Isha Vaidya
 Shambhavi Marne
 Riya Kedari
(3 rows)
```

---

2. List the names of the professor who teaches less than two subjects.
==>SELECT P.ProfessorID , CONCAT(P.FirstName,' ',P.LastName) AS ProfessorName FROM Professors P JOIN Course_Details CD
ON P.ProfessorID = CD.ProfessorID
WHERE CD.SemID = 'Sem_1'
GROUP BY P.ProfessorID, P.FirstName, P.LastName
HAVING COUNT(CD.ProfessorID) < 2;

```
 professorid | professorname
-------------+---------------
           1 | Amit Sharma
           2 | Smita Patil
           3 | Rajiv Gupta
           7 | Preeti Dave
           8 | Rahul Kumar
          10 | Suresh Shukla
(6 rows)
```

---

3. Find the maximum percentage in integrated result.
==>SELECT MAX(Final_per) FROM Integreted_result;

```
  max
-------
 93.92
(1 row)
```

---

4. Find out the total number of student who have passed in 1st year without backlog in
“History“ course.
==>SELECT COUNT(RW.StudentID) FROM Results_SemWise RW
JOIN Student_course SC ON SC.StudentID = RW.StudentID
WHERE RW.Back1_per ISNULL AND RW.Back2_per ISNULL AND SC.Course = 'History';

```
 count
-------
     2
(1 row)
```

---

5. Count the number of studentS received backlog in “rDNA Technology” subject in sem 2.
==>SELECT COUNT(S.StudentID) FROM Students S
JOIN Backlog B
ON B.StudentID = S.StudentID
WHERE B.Sem2_Back = 'MB203';

```
 count
-------
     1
```

(1 row)

---

6. List all the names of Professors along with the subject names and course which are thought
in sem 4.
```
==>SELECT CONCAT(P.FirstName,' ',P.LastName) AS ProfessorName, CD.CourseName , CD.SubjectName
FROM Professors P , Course_Details CD
WHERE SemID = 'Sem_4' AND CD.ProfessorID = P.ProfessorID;
```
```
 professorname |    coursename    |      subjectname
---------------+------------------+-----------------------
 Amit Sharma   | Computer Science | Electronics
 Smita Patil   | Computer Science | Mathematics
 Suresh Shukla | Microbiology     | Molecular Biology
 Rahul Kumar   | Microbiology     | Cancer Biology
 Anjali Rajput | History          | Ancient Civilizations
 Preeti Dave   | History          | World History
 Vikram Joshi  | Mathematics      | Calculus I
 Vikram Joshi  | Mathematics      | Linear Algebra
(8 rows)
```

(8 rows)

---

7. List the details of the students whose name starts with letter A.
```
==>SELECT *
FROM  Students
WHERE FirstName LIKE 'A%';
```
```
 studentid | firstname | lastname  | dateofbirth |          email           |  s_number  | parent_no  |   address    | gender
-----------+-----------+-----------+-------------+--------------------------+------------+------------+--------------+--------
   2371001 | Advait    | Pradhan   | 2000-05-15  | advaitpradhan@example.com | 1234567890 | 9876543210 | ShivajiNagar | M
   2371010 | Akanksha  | Pimparkar | 2003-03-29  | akanksha.p@example.com    | 7890123456 | 3210987654 | Sahakarnagar | F
(2 rows)
```

---

8. List the students details who have passed with less then 80%.
```
==>SELECT S.*
FROM  Students S
JOIN Integreted_result IR
ON IR.StudentID = S.StudentID
WHERE Final_per > 80;
```
```
 studentid | firstname | lastname | dateofbirth |        email        |  s_number  | parent_no  |    address    | gender
-----------+-----------+----------+-------------+---------------------+------------+------------+---------------+--------
   2371005 | Riya      | Kedari   | 2001-04-03  | riya.k@example.com  | 5678901234 | 6543210987 | Shanivar Peth | F
   2371008 | Mahek     | Patel    | 2001-02-28  | Mahek.p@example.com | 2345678901 | 8765432109 | Shanivar Peth | F
   2371009 | Jatin     | Joshi    | 2000-06-30  | jatin.j@example.com | 3456789012 | 9876543210 | Sahakarnagar  | M
```

```
(3 rows)


=============================================================================================================
View:
----
Set A:
------

Q1 Create a view which contains the details with Integrated result of all student who have applied for a "History" course.
==>CREATE view v1 as SELECT I.Final_per
from Integreted_result I,Student_course T
where T.Course='History' and T.StudentID=I.StudentID;

CREATE VIEW

-------------------------------------------------------------------------------------------------------

Q2 Create a view which contains details with Integrated result of student of all professor.
==>CREATE view v2 as SELECT P.* , CONCAT(I.Final_per) as result
from Professors P,Integreted_result I,Student_course T,Course_Details C
where I.StudentID=T.StudentID and T.Course=C.CourseName and C.ProfessorID=P.ProfessorID
group by P.ProfessorID,result
having count(C.ProfessorID)>1;
CREATE VIEW

-------------------------------------------------------------------------------------------------------

Q3. Write the following Queries, on the above created views :
-------------------------------------------------------------
A.] List the details of professors who stay in "Sahakarnagar".
==>SELECT v2.* from v2 where Address = 'Sahakarnagar'  ;
 professorid | firstname | lastname | qualifications  |          email           |  p_number  |   address    | gender | result
-------------+-----------+----------+-----------------+--------------------------+------------+--------------+--------+--------
           8 | Rahul     | Kumar    | Graduation      | rahul.kumar@example.com  | 8901234567 | Sahakarnagar | M      | 76.80
           8 | Rahul     | Kumar    | Graduation      | rahul.kumar@example.com  | 8901234567 | Sahakarnagar | M      | 76.93
           8 | Rahul     | Kumar    | Graduation      | rahul.kumar@example.com  | 8901234567 | Sahakarnagar | M      | 80.55
           7 | Preeti    | Dave     | Post Graduation | kavita.joshi@example.com | 7890123456 | Sahakarnagar | F      | 64.12
           7 | Preeti    | Dave     | Post Graduation | kavita.joshi@example.com | 7890123456 | Sahakarnagar | F      | 79.25
(5 rows)

-------------------------------------------------------------------------------------------------------

B List the details of professor from "Shanivar Peth", where student result is above 75
==>SELECT v2.* from v2 where Address = 'Shanivar Peth' and cast(result as numeric) > 75.00 ;
 professorid | firstname | lastname | qualifications |          email           |  p_number  |    address    | gender | result
-------------+-----------+----------+----------------+--------------------------+------------+---------------+--------+--------
           9 | Neha      | Mishra   | Double Ph.D    | neha.mishra@example.com  | 9012345678 | Shanivar Peth | F      | 76.80
           9 | Neha      | Mishra   | Double Ph.D    | neha.mishra@example.com  | 9012345678 | Shanivar Peth | F      | 76.93
```

```
          9 | Neha      | Mishra   | Double Ph.D    | neha.mishra@example.com | 9012345678 | Shanivar Peth | F      | 80.55
(3 rows)

=============================================================================================================
Set B:
------

1. Create a view to list the details of all subjects from ‘ Microbiology ‘ course.
==>CREATE VIEW V1 AS SELECT CD.*
FROM Course_Details CD
WHERE CD.CourseName = 'Microbiology';

CREATE VIEW


-------------------------------------------------------------------------------------------------------------

2. Create a view to list all students full name and his course details.
==>CREATE VIEW V2 AS SELECT Students.StudentID,
    CONCAT(Students.FirstName, ' ', Students.LastName) AS StudentName,
    Course_Details.CourseID,
    Course_Details.CourseName,
    Course_Details.StartDate,
    Course_Details.ExamDate
FROM Students
JOIN Student_course ON Students.StudentID = Student_course.StudentID
JOIN Course_Details ON Student_course.Course = Course_Details.CourseName
GROUP BY
    Students.StudentID,
    Course_Details.CourseID
HAVING COUNT(Course_Details.CourseID)<2
ORDER BY
    Students.StudentID ASC,
    Course_Details.CourseID;

CREATE VIEW

-------------------------------------------------------------------------------------------------------------

3. Write the following Queries, on the above created views :
-----------------------------------------------------------
a. List all available details of students from ‘Mathemathics’ course, where student name start with
   "M".
==>SELECT V2.* FROM V2
WHERE CourseName = 'Mathematics' AND StudentName LIKE 'M%';

 studentid | studentname | courseid | coursename  | startdate  |  examdate
-----------+-------------+----------+-------------+------------+------------
   2371008 | Mahek Patel | MAT101   | Mathematics | 2023-08-15 | 2023-12-10
   2371008 | Mahek Patel | MAT102   | Mathematics | 2023-08-15 | 2023-12-11
```

```
   2371008 | Mahek Patel | MAT103   | Mathematics | 2023-08-15 | 2023-12-12
   2371008 | Mahek Patel | MAT201   | Mathematics | 2024-01-15 | 2024-05-19
   2371008 | Mahek Patel | MAT202   | Mathematics | 2024-01-15 | 2024-05-20
   2371008 | Mahek Patel | MAT203   | Mathematics | 2024-01-15 | 2024-05-21
   2371008 | Mahek Patel | MAT301   | Mathematics | 2024-08-15 | 2024-12-10
   2371008 | Mahek Patel | MAT302   | Mathematics | 2024-08-15 | 2024-12-11
   2371008 | Mahek Patel | MAT401   | Mathematics | 2025-01-15 | 2025-05-21
   2371008 | Mahek Patel | MAT402   | Mathematics | 2025-01-15 | 2025-05-22
(10 rows)


-----------------------------------------------------------------------------------------------

b. List the names of subjects having exam date as "2023-11-17".
==>SELECT SubjectName FROM V1
WHERE ExamDate = '2023-11-17';

   subjectname
-----------------
 rDNA Technology
(1 row)


-----------------------------------------------------------------------------------------------

c. List the names of student whose course start "2023-07-15" having course "Computer Science".
==>SELECT StudentName FROM V2
WHERE StartDate = '2023-07-15' AND CourseName = 'Computer Science'
GROUP BY
    StudentName
HAVING COUNT(StudentName)>1;

  studentname
----------------
 Advait Pradhan
 Kaushal Pawar
(2 rows)


===============================================================================================
Stored Functions:
----------------
Set A:
------

Q1 a)Write a PL/pgsql function to find a Student Id having maximum percentage.
CREATE or replace function StudentID()
returns integer as $$
declare
S integer;
```

```
begin
     SELECT StudentID into S
     from Integreted_result
     where Final_per=(SELECT max(Final_per) from Integreted_result);
     return S;
end;$$
LANGUAGE 'plpgsql';
CREATE FUNCTION
SELECT StudentID();
studentid
-----------
   2371009
(1 row)

-----------------------------------------------------------------------------------------------------

Q2 b)Write a PL/pgsql function to count the total number of students from ‘Akurdi’ Area.
CREATE or replace  function S_count()
returns integer as $$
declare
cnt integer;
begin
     select count(Address) into cnt
     from Students
     where Address = 'Akurdi';
     return cnt;
end;$$
language 'plpgsql';
CREATE FUNCTION
select S_count();
s_count
---------
       2
(1 row)

=====================================================================================================================================
Set B:
------

a)Write a function to count total number of student having percentage higher than 75% in sem3.
==>CREATE OR REPLACE FUNCTION STUDENT_COUNT()
RETURNS INTEGER AS $$
DECLARE
CNT INTEGER;
BEGIN
      SELECT COUNT(StudentID) INTO CNT
      FROM Results_SemWise
      WHERE ((Sem3_per > 75 AND Back3_per IS NULL)
      OR (Sem3_per < 75 AND Back3_per IS NOT NULL AND Back3_per > 75));
      RETURN CNT;
```

```
END;$$
LANGUAGE 'plpgsql';

CREATE FUNCTION

SELECT STUDENT_COUNT();
 student_count
---------------
             3
(1 row)

-----------------------------------------------------------------------------------------------------

b)Write a function to find student name having highest score in sem 4.
==>CREATE OR REPLACE FUNCTION Highest_percentage()
RETURNS INTEGER AS $$
DECLARE
ID INT;
PER INT;
BEGIN
      SELECT MAX(Sem4_per) INTO PER
      FROM Results_SemWise ;
      SELECT RW.StudentID INTO ID
      FROM Results_SemWise RW
      WHERE Sem4_per IN (SELECT MAX(Sem4_per) FROM Results_SemWise);
      RAISE NOTICE 'Student Name : %',(SELECT CONCAT(FirstName,' ',LastName) AS StudentName FROM Students WHERE StudentID=ID);
      RETURN PER ;
END;$$
LANGUAGE 'plpgsql';
CREATE FUNCTION

SELECT Highest_percentage();
 highest_percentage
--------------------
                 99
(1 row)
psql:commands.sql:414: NOTICE:  Student Name : Jatin Joshi

=====================================================================================================

Exeption Handling:
-----------------
Set A:
------

1. Write a stored function to accept the student ID and display the detail of student.
Raise an exception in case of invalid enrollment number.
==>CREATE OR REPLACE FUNCTION Student_Details(ID INTEGER)
RETURNS VOID AS $$
DECLARE
```

```
    R1 RECORD;
BEGIN
    IF(ID IN (SELECT StudentID FROM Students))
    THEN
    SELECT * INTO R1
    FROM Students
    WHERE StudentID = ID;
    RAISE NOTICE 'Students Details :';
    RAISE NOTICE 'Student ID : %',R1.StudentID;
    RAISE NOTICE 'Student Name : %',(SELECT CONCAT(FirstName,' ',LastName) FROM Students WHERE StudentID = ID);
    RAISE NOTICE 'Date of birth : %',R1.DateOfBirth;
    RAISE NOTICE 'Students Email : %',R1.Email;
    RAISE NOTICE 'Students Number : %',R1.S_Number;
    RAISE NOTICE 'Parents Number : %',R1.Parent_no;
    RAISE NOTICE 'Address : %',R1.Address;
    RAISE NOTICE 'Gender : %',R1.Gender;
    ELSE
    RAISE EXCEPTION 'INVALID Student ID..';
    END IF;
END;$$
LANGUAGE 'plpgsql';

CREATE FUNCTION

SELECT Student_Details(2371005);
psql:commands.sql:396: NOTICE:  Students Details :
psql:commands.sql:396: NOTICE:  Student ID : 2371005
psql:commands.sql:396: NOTICE:  Student Name : Riya Kedari
psql:commands.sql:396: NOTICE:  Date of birth : 2001-04-03
psql:commands.sql:396: NOTICE:  Students Email : riya.k@example.com
psql:commands.sql:396: NOTICE:  Students Number : 5678901234
psql:commands.sql:396: NOTICE:  Parents Number : 6543210987
psql:commands.sql:396: NOTICE:  Address : Shanivar Peth
psql:commands.sql:396: NOTICE:  Gender : F


SELECT Student_Details(2371008);
psql:commands.sql:396: NOTICE:  Students Details :
psql:commands.sql:396: NOTICE:  Student ID : 2371008
psql:commands.sql:396: NOTICE:  Student Name : Mahek Patel
psql:commands.sql:396: NOTICE:  Date of birth : 2001-02-28
psql:commands.sql:396: NOTICE:  Students Email : Mahek.p@example.com
psql:commands.sql:396: NOTICE:  Students Number : 2345678901
psql:commands.sql:396: NOTICE:  Parents Number : 8765432109
psql:commands.sql:396: NOTICE:  Address : Shanivar Peth
psql:commands.sql:396: NOTICE:  Gender : F

-----------------------------------------------------------------------------------------------

2. Write a stored function to accept subject name as input and display Course Name and
```

professor name for the respected subject . Raise an exception in case of invalid subject name.

```
==>CREATE OR REPLACE FUNCTION Details(Name VARCHAR(50))
RETURNS VOID AS $$
DECLARE
    R1 RECORD;
BEGIN
  IF(NAME IN (SELECT SubjectName FROM Course_Details))
  THEN
    SELECT CONCAT(P.FirstName,' ',P.LastName) AS ProfessorName , CD.CourseName , CD.SubjectName INTO R1
    FROM Professors P , Course_Details CD
    WHERE NAME = CD.SubjectName
    AND CD.ProfessorID = P.ProfessorID;

      RAISE NOTICE 'Professor Name : %',R1.ProfessorName;
      RAISE NOTICE 'Course Name : %',R1.CourseName;
      RAISE NOTICE 'Subject Name : %',R1.SubjectName;

    ELSE
      RAISE EXCEPTION 'INVALID STUDENT NAME..';
  END IF;
END;$$
LANGUAGE 'plpgsql';

CREATE FUNCTION
SELECT Details('rDNA Technology');
CREATE FUNCTION
psql:commands.sql:422: NOTICE:  Professor Name : Neelam Verma
psql:commands.sql:422: NOTICE:  Course Name : Microbiology
psql:commands.sql:422: NOTICE:  Subject Name : rDNA Technology
```

==========

Set B:

---

1. Write a stored function to accept course name as input and display the names of students studing in that course.
(Accept course name as input parameter).

```
==>
CREATE or REPLACE FUNCTION Course_Name(Name VARCHAR(50))
RETURNS TABLE (student_name VARCHAR(50)) as $$
DECLARE
    R1 RECORD;
BEGIN
    if(Name in (SELECT Course from Student_course))
      Then
      SELECT * into R1
      from Student_course T ,Students S
      where Course = Name and S.StudentID = T.StudentID;
```

```
      Raise NOTICE 'Students Details :';
      Raise NOTICE 'StudentID : %',R1.StudentID;
      Raise NOTICE 'Student Name :%',(SELECT CONCAT(FirstName,' ',LastName) from Students Where StudentID = R1.StudentID);

    Else
     Raise EXCEPTION 'INVALID Course Name..';
    END if;
End;$$
LANGUAGE 'plpgsql';


SELECT * from Course_Name('History');
psql:commands.sql:388: NOTICE:  Students Details :
psql:commands.sql:388: NOTICE:  StudentID : 2371006
psql:commands.sql:388: NOTICE:  Student Name :Prathamesh Shinde

-----------------------------------------------------------------------------------------------

2. Write a stored function to accept Student name as input and display the details of students.(Accept student name as input parameter). Raise an exception for an
invalid student name.
CREATE or REPLACE FUNCTION Student_Name(Name VARCHAR(50))
RETURNS VOID as $$
DECLARE
    R1 RECORD;
BEGIN
    if(Name in (SELECT CONCAT(FirstName,' ',LastName) as s_name from Students))
      Then
      SELECT * into R1
      from Students S
      where CONCAT(FirstName,' ',LastName) = Name  ;
     RAISE NOTICE 'Students Details :';
    RAISE NOTICE 'Student ID : %',R1.StudentID;
    RAISE NOTICE 'Student Name : %',(SELECT CONCAT(FirstName,' ',LastName) FROM Students WHERE  CONCAT(FirstName,' ',LastName) = Name);
    RAISE NOTICE 'Date of birth : %',R1.DateOfBirth;
    RAISE NOTICE 'Students Email : %',R1.Email;
    RAISE NOTICE 'Students Number : %',R1.S_Number;
    RAISE NOTICE 'Parents Number : %',R1.Parent_no;
    RAISE NOTICE 'Address : %',R1.Address;
    RAISE NOTICE 'Gender : %',R1.Gender;
    ELSE
    RAISE EXCEPTION 'INVALID Student Name..';
    END IF;
END;$$
LANGUAGE 'plpgsql';


SELECT * from Student_Name('Mahek Patel');

psql:commands.sql:419: NOTICE:  Students Details :
psql:commands.sql:419: NOTICE:  Student ID : 2371008
```

```
psql:commands.sql:419: NOTICE:  Student Name : Mahek Patel
psql:commands.sql:419: NOTICE:  Date of birth : 2001-02-28
psql:commands.sql:419: NOTICE:  Students Email : Mahek.p@example.com
psql:commands.sql:419: NOTICE:  Students Number : 2345678901
psql:commands.sql:419: NOTICE:  Parents Number : 8765432109
psql:commands.sql:419: NOTICE:  Address : Shanivar Peth
psql:commands.sql:419: NOTICE:  Gender : F

================================================================================================
================================================================================================
Cursor:
------
Set A:
------

1. Write a function using cursor to list all courses and their professors names.
CREATE or REPLACE FUNCTION Course_Professor()
RETURNS VOID as $$
DECLARE

    c1 cursor for SELECT P.*,C.CourseName from Professors P,Course_Details C  WHERE P.ProfessorID=C.ProfessorID;
    R1 RECORD;

BEGIN
    open c1;
    RAISE NOTICE 'Names of Professors::';
loop
    fetch c1 into R1;
    exit when not found;
    RAISE NOTICE '% % % %',R1.ProfessorID,R1.FirstName,R1.LastName,R1.CourseName;
end loop;
close c1;

END;$$
LANGUAGE 'plpgsql';
CREATE FUNCTION
SELECT Course_Professor();
course_professor
------------------

(1 row)
psql:commands.sql:414: NOTICE:  Names of Professors::
psql:commands.sql:414: NOTICE:  1 Amit Sharma Computer Science
psql:commands.sql:414: NOTICE:  2 Smita Patil Computer Science
psql:commands.sql:414: NOTICE:  3 Rajiv Gupta Computer Science
psql:commands.sql:414: NOTICE:  4 Neelam Verma Computer Science
psql:commands.sql:414: NOTICE:  5 Anjali Rajput Computer Science
psql:commands.sql:414: NOTICE:  10 Suresh Shukla Microbiology
psql:commands.sql:414: NOTICE:  8 Rahul Kumar Microbiology
```

```
psql:commands.sql:414: NOTICE:  4 Neelam Verma Microbiology
psql:commands.sql:414: NOTICE:  9 Neha Mishra Microbiology
psql:commands.sql:414: NOTICE:  9 Neha Mishra Microbiology
psql:commands.sql:414: NOTICE:  5 Anjali Rajput History
psql:commands.sql:414: NOTICE:  7 Preeti Dave History
psql:commands.sql:414: NOTICE:  6 Vikram Joshi Mathematics
psql:commands.sql:414: NOTICE:  6 Vikram Joshi Mathematics
psql:commands.sql:414: NOTICE:  4 Neelam Verma Mathematics
psql:commands.sql:414: NOTICE:  1 Amit Sharma Computer Science
psql:commands.sql:414: NOTICE:  2 Smita Patil Computer Science
psql:commands.sql:414: NOTICE:  3 Rajiv Gupta Computer Science
psql:commands.sql:414: NOTICE:  4 Neelam Verma Computer Science
psql:commands.sql:414: NOTICE:  10 Suresh Shukla Microbiology
psql:commands.sql:414: NOTICE:  8 Rahul Kumar Microbiology
psql:commands.sql:414: NOTICE:  4 Neelam Verma Microbiology
psql:commands.sql:414: NOTICE:  9 Neha Mishra Microbiology
psql:commands.sql:414: NOTICE:  5 Anjali Rajput History
psql:commands.sql:414: NOTICE:  7 Preeti Dave History
psql:commands.sql:414: NOTICE:  6 Vikram Joshi Mathematics
psql:commands.sql:414: NOTICE:  6 Vikram Joshi Mathematics
psql:commands.sql:414: NOTICE:  4 Neelam Verma Mathematics
psql:commands.sql:414: NOTICE:  1 Amit Sharma Computer Science
psql:commands.sql:414: NOTICE:  2 Smita Patil Computer Science
psql:commands.sql:414: NOTICE:  3 Rajiv Gupta Computer Science
psql:commands.sql:414: NOTICE:  10 Suresh Shukla Microbiology
psql:commands.sql:414: NOTICE:  8 Rahul Kumar Microbiology
psql:commands.sql:414: NOTICE:  4 Neelam Verma Microbiology
psql:commands.sql:414: NOTICE:  5 Anjali Rajput History
psql:commands.sql:414: NOTICE:  7 Preeti Dave History
psql:commands.sql:414: NOTICE:  6 Vikram Joshi Mathematics
psql:commands.sql:414: NOTICE:  6 Vikram Joshi Mathematics
psql:commands.sql:414: NOTICE:  1 Amit Sharma Computer Science
psql:commands.sql:414: NOTICE:  2 Smita Patil Computer Science
psql:commands.sql:414: NOTICE:  10 Suresh Shukla Microbiology
psql:commands.sql:414: NOTICE:  8 Rahul Kumar Microbiology
psql:commands.sql:414: NOTICE:  5 Anjali Rajput History
psql:commands.sql:414: NOTICE:  7 Preeti Dave History
psql:commands.sql:414: NOTICE:  6 Vikram Joshi Mathematics
psql:commands.sql:414: NOTICE:  6 Vikram Joshi Mathematics

-----------------------------------------------------------------------------------------------

Q2. Write a function to accept course name and display details of students studing it.
CREATE or REPLACE FUNCTION Course_Student(VARCHAR(50))
RETURNS VOID as $$
DECLARE
    K alias for $1;
    c1 cursor for SELECT S.* from Student_course T ,Students S WHERE T.StudentID=S.StudentID and T.Course=K;

    R1 RECORD;
```

```
BEGIN
    open c1;
    if(K in (SELECT CourseName from Course_Details))then
    RAISE NOTICE 'Name of Course is::';
loop
    fetch c1 into R1;
    exit when not found;
    RAISE NOTICE '% % % % % % % % ',R1.StudentID,R1.FirstName,R1.LastName,R1.DateOfBirth,R1.Email,R1.S_Number,R1.Parent_no,R1.Address,R1.Gender;
end loop;


end if;
close c1;

END;$$
LANGUAGE 'plpgsql';
CREATE FUNCTION
SELECT Course_Student('Mathematics');
course_student
----------------

(1 row)

psql:commands.sql:419: NOTICE:  Name of Course is::
psql:commands.sql:419: NOTICE:  2371008 Mahek Patel 2001-02-28 Mahek.p@example.com 2345678901 8765432109 Shanivar Peth F
psql:commands.sql:419: NOTICE:  2371009 Jatin Joshi 2000-06-30 jatin.j@example.com 3456789012 9876543210 Sahakarnagar M
psql:commands.sql:419: NOTICE:  2371010 Akanksha Pimparkar 2003-03-29 akanksha.p@example.com 7890123456 3210987654 Sahakarnagar F

=============================================================================================

Set B:
------

1.]Write a function using cursor which accept course name as input and display the
details of professor teaching on that course.
==>
CREATE OR REPLACE FUNCTION Prof_Details(Course VARCHAR(20))
RETURNS VOID AS $$
DECLARE
    C1 CURSOR FOR SELECT P.* FROM Professors P
    JOIN Course_Details CD
    on P.ProfessorID = CD.ProfessorID
    WHERE CD.CourseName = Course
    AND SemID = 'Sem_1';
    R1 RECORD;

BEGIN
  OPEN C1;
```

```
    LOOP
      FETCH C1 INTO R1;
      EXIT WHEN NOT FOUND;
        RAISE NOTICE 'Professors ID:: %',R1.ProfessorID;
        RAISE NOTICE 'Professors FirstName :: %',R1.FirstName;
        RAISE NOTICE 'Professors LastName :: %',R1.LastName;
        RAISE NOTICE 'Professors Qualifications :: %',R1.Qualifications;
        RAISE NOTICE 'Professors Email :: %',R1.Email;
        RAISE NOTICE 'Professors Number :: %',R1.P_number;
        RAISE NOTICE 'Professors Address :: %',R1.Address;
        RAISE NOTICE 'Professors Gender :: %',R1.Gender;
    END LOOP;
  CLOSE C1;
END;$$
LANGUAGE 'plpgsql';


SELECT Prof_Details('History');

psql:commands.sql:399: NOTICE:  Professors ID:: 5
psql:commands.sql:399: NOTICE:  Professors FirstName :: Anjali
psql:commands.sql:399: NOTICE:  Professors LastName :: Rajput
psql:commands.sql:399: NOTICE:  Professors Qualifications :: Post Graduation
psql:commands.sql:399: NOTICE:  Professors Email :: anjali.rajput@example.com
psql:commands.sql:399: NOTICE:  Professors Number :: 5678901234
psql:commands.sql:399: NOTICE:  Professors Address :: Shivajinagar
psql:commands.sql:399: NOTICE:  Professors Gender :: F
psql:commands.sql:399: NOTICE:  Professors ID:: 7
psql:commands.sql:399: NOTICE:  Professors FirstName :: Preeti
psql:commands.sql:399: NOTICE:  Professors LastName :: Dave
psql:commands.sql:399: NOTICE:  Professors Qualifications :: Post Graduation
psql:commands.sql:399: NOTICE:  Professors Email :: kavita.joshi@example.com
psql:commands.sql:399: NOTICE:  Professors Number :: 7890123456
psql:commands.sql:399: NOTICE:  Professors Address :: Sahakarnagar
psql:commands.sql:399: NOTICE:  Professors Gender :: F
-----------------------------------------------------------------------------------------------

2.]Write a function using cursor to accept year and display students which have birth in given
year.
==>
CREATE OR REPLACE FUNCTION Stud_Details(Year INTEGER)
RETURNS VOID AS $$
DECLARE
    C1 CURSOR FOR SELECT CONCAT(FirstName,' ',LastName) AS StudentName FROM Students
    WHERE EXTRACT(YEAR FROM DateOfBirth) = Year;
    R1 RECORD;

BEGIN
  OPEN C1;
    LOOP
```

```
      FETCH C1 INTO R1;
      EXIT WHEN NOT FOUND;
        RAISE NOTICE 'Student Name :: %',R1.StudentName;
    END LOOP;
  CLOSE C1;
END;$$
LANGUAGE 'plpgsql';


SELECT Stud_Details(2005);
psql:commands.sql:389: NOTICE:  Student Name :: Pushkar Oli
psql:commands.sql:389: NOTICE:  Student Name :: Akanksha Pimparkar

SELECT Stud_Details(2001);
psql:commands.sql:389: NOTICE:  Student Name :: Riya Kedari
psql:commands.sql:389: NOTICE:  Student Name :: Mahek Patel

================================================================================================
Trigger:
-------
Set A:
------

1. Write a trigger after insert on Student to display message "STUDENT IS ELIGIBLE FOR GIVING
EXAM" If the date year less than 2003.
==>CREATE OR REPLACE FUNCTION Check_Student_Date()
RETURNS TRIGGER AS $$
BEGIN
IF NEW.DateOfBirth > '2004-12-31'  THEN
RAISE EXCEPTION 'STUDENT IS NOT ELIGIBLE TO GIVE EXAM';
END IF;
RETURN NEW;
END; $$
LANGUAGE 'plpgsql';

CREATE FUNCTION

CREATE TRIGGER StudentID_TRIGGER AFTER INSERT OR UPDATE ON Students FOR EACH ROW EXECUTE PROCEDURE Check_Student_Date();
CREATE TRIGGER

INSERT INTO Students VALUES (2371011,'Komal','Yadav','2001-08-13','komal.y@example.com',7237832983,5475205830,'Taljai','F');
INSERT 0 1

INSERT INTO Students VALUES (2371012,'Preet','Nartekar','2005-03-20','preet.n@example.com',4857385839,4576920572,'BalagiNagar','M');
psql:commands.sql:430: ERROR:  STUDENT IS NOT ELIGIBLE TO GIVE EXAM

------------------------------------------------------------------------------------------------

2. Create a trigger that automatically updates the "Integreted_result" table when new results are
inserted into the "Result_Semwise" table.
```

```
==>CREATE OR REPLACE FUNCTION UpdateIntegratedResult()
RETURNS TRIGGER AS $$
DECLARE
    final_percentage DECIMAL(5, 2);
BEGIN
    -- Calculate final percentage (GPA) for the student
    final_percentage := ((NEW.Sem1_per + NEW.Sem2_per + NEW.Sem3_per + NEW.Sem4_per) / 4 );

    -- Update the Integrated_result table
    INSERT INTO Integreted_result (ResultID,StudentID,Cerdits,Final_per)
    VALUES
        (NEW.ResultID,
        NEW.StudentID,
        96,
        final_percentage);

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE FUNCTION

CREATE TRIGGER UpdateIntegratedResultTrigger
AFTER INSERT ON Results_SemWise
FOR EACH ROW
EXECUTE FUNCTION UpdateIntegratedResult();

CREATE TRIGGER


INSERT INTO Semwise_marks( StudentID , Sem1_Regu , Marks_1 ,
Sem2_Regu , Marks_2 ,
Sem3_Regu , Marks_3 ,
Sem4_Regu , Marks_4 )
VALUES
      (2371011,'H101',65,'H201',87,'H301',76,'H401',93),
      (2371011,'H102',76,'H202',56,'H302',86,'H402',96);
INSERT 0 2

INSERT INTO Results_SemWise (ResultID ,StudentID ,
Sem1_per , Back1_per , Sem2_per , Back2_per ,
Sem3_per ,Back3_per ,Sem4_per ,Back4_per)
VALUES
      (2200511,2371011,70.5,NULL,71.5,NULL,81.0,NULL,94.5,NULL);
INSERT 0 1

SELECT * FROM Results_SemWise;
 resultid | studentid | sem1_per | back1_per | sem2_per | back2_per | sem3_per | back3_per | sem4_per | back4_per
----------+-----------+----------+-----------+----------+-----------+----------+-----------+----------+-----------
  2200501 |   2371001 |    75.40 |           |    72.25 |           |    42.34 |     74.67 |    88.50 |
```

```
  2200502 |   2371002 |    41.80 |     68.80 |    74.00 |           |    68.67 |           |    86.50 |
  2200503 |   2371003 |    72.60 |           |    53.75 |     76.25 |    67.34 |           |    91.50 |
  2200504 |   2371004 |    76.20 |           |    63.75 |     76.00 |    51.34 |     72.00 |    47.00 |     83.00
  2200505 |   2371005 |    64.60 |     77.20 |    75.50 |           |    82.00 |           |    87.50 |
  2200506 |   2371006 |    63.50 |           |    57.00 |           |    40.00 |     67.00 |    69.00 |
  2200507 |   2371007 |    79.00 |           |    73.50 |           |    71.00 |           |    93.50 |
  2200508 |   2371008 |    83.00 |           |    63.00 |     89.34 |    79.50 |           |    92.00 |
  2200509 |   2371009 |    95.34 |           |    87.34 |           |    94.00 |           |    99.00 |
  2200510 |   2371010 |    46.67 |     70.00 |    74.67 |           |    67.50 |           |    89.50 |
  2200511 |   2371011 |    70.50 |           |    71.50 |           |    81.00 |           |    94.50 |
(11 rows)

SELECT * FROM Integreted_result;
 resultid | studentid | cerdits | final_per
----------+-----------+---------+-----------
  2200501 |   2371001 |      96 |     77.70
  2200502 |   2371002 |      96 |     74.49
  2200503 |   2371003 |      96 |     76.93
  2200504 |   2371004 |      96 |     76.80
  2200505 |   2371005 |      96 |     80.55
  2200506 |   2371006 |      96 |     64.12
  2200507 |   2371007 |      96 |     79.25
  2200508 |   2371008 |      96 |     85.96
  2200509 |   2371009 |      96 |     93.92
  2200510 |   2371010 |      96 |     75.42
  2200511 |   2371011 |      96 |     79.38
(11 rows)

=============================================================================================================
Set B:
------

1. Write a trigger before insert the record of Student . If the StudentID is less than or equal to zero give message " Invalid Number ".
CREATE or REPLACE FUNCTION Student_ID()
RETURNS TRIGGER as $$

BEGIN
   if NEW.StudentID <= 0 then
   RAISE EXCEPTION 'INVALID Number:';
END if;
RETURN NEW;
END;$$
LANGUAGE 'plpgsql';

CREATE TRIGGER StudentID_TRIGGER AFTER INSERT OR UPDATE ON Students FOR EACH ROW EXECUTE PROCEDURE Student_ID();
CREATE FUNCTION
CREATE TRIGGER
INSERT INTO Students VALUES
(23,'Om','Karnik','2005-09-07','OK@example.com',6237871683,0981388091,'Mumbai','M');
INSERT 0 1
```

```
INSERT INTO Students VALUES
(0,'Omkar','Kapoor','2003-07-09','OmkarK@example.com',6239871683,0981986091,'Mumbai','M');
psql:commands.sql:409: ERROR:  INVALID Number:

-----------------------------------------------------------------------------------------------

2. Write a trigger before update a student’s email from student table. Display appropriate message.
CREATE OR REPLACE FUNCTION validate_student_email()
RETURNS TRIGGER AS $$
BEGIN

    IF NEW.email ~ '^[a-zA-Z0-9._%+-]+@example.com' THEN

        RETURN NEW;
    ELSE

        RAISE EXCEPTION 'Invalid email format. Email must end with "@example.com".';
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_update_student_email
BEFORE UPDATE ON Students
FOR EACH ROW
EXECUTE FUNCTION validate_student_email();
CREATE FUNCTION
CREATE TRIGGER

UPDATE students
SET email = 'riya@example.com'
WHERE studentid = 2371005;
UPDATE 1

UPDATE students
SET email = 'advait@university.com'
WHERE studentid = 2371001;
psql:commands.sql:390: ERROR:  Invalid email format. Email must end with "@example.com".
=============================================================================================================================================
```