EXCEPTION HANDLING

SET A:Project-Employee database

Consider the following database:

Project (pno int, pname char (30), ptype char (20), duration int)

Employee (empno int, ename char (20), joining_date date)

The relationship between Project and Employee is many to many with descriptive attribute

start_date.

Create the above database in PostGreSQL and insert sufficient records.

a. Write a stored function to accept project name as input and print the names ofemployees

working on the project. Also print the total number of employees working onthat project.

Raise an exception for an invalid project name.

b. Write a stored function to accept empno as an input parameter from the user and count

the number of projects of a given employee. Raise an exception if the employee number

is invalid.

```
CREATE TABLE Project (
   pno SERIAL PRIMARY KEY,
   pname VARCHAR(30) NOT NULL,
   ptype VARCHAR(20),
   duration INT
);

CREATE TABLE Employee (
   empno SERIAL PRIMARY KEY,
   ename VARCHAR(20) NOT NULL,
   joining_date DATE
);

CREATE TABLE Project_Employee (
   pno INT REFERENCES Project(pno),
   empno INT REFERENCES Employee(empno),
   start_date DATE);

-- Inserting data into the Project table
INSERT INTO Project VALUES (1,'Project A', 'Type 1', 12);
INSERT INTO Project VALUES(2,'Project B', 'Type 2', 24);
INSERT INTO Project VALUES(3,'Project C', 'Type 3', 6);

-- Inserting data into the Employee table
INSERT INTO Employee VALUES (1,'Alice', '2020-01-01');
INSERT INTO Employee VALUES(2,'Bob', '2021-05-15');
INSERT INTO Employee VALUES(3,'Charlie', '2022-08-10');
INSERT INTO Employee VALUES(4,'David', '2021-03-22');

-- Inserting data into the Project_Employee table
INSERT INTO Project_Employee VALUES(1, 1, '2020-01-01');
INSERT INTO Project_Employee VALUES(1, 2, '2021-05-15');
INSERT INTO Project_Employee VALUES(2, 3, '2022-08-10');
INSERT INTO Project_Employee VALUES(3, 4, '2021-03-22');
```

Q1. **Write a stored function to accept project name as input and print the names ofemployees working on the project. Also print the total number of employees working onthat project. Raise an exception for an invalid project name.**

```
CREATE OR REPLACE FUNCTION get_employees_by_project(pname_input
VARCHAR) RETURNS VOID AS '
DECLARE
    project_id INT;
    emp_name VARCHAR(20);
    emp_count INT := 0;
BEGIN
    SELECT pno INTO project_id
    FROM Project
    WHERE pname = pname_input;
    IF NOT FOUND THEN
        RAISE EXCEPTION ''Invalid project name: %'', pname_input;
    END IF;
       FOR emp_name IN
        SELECT e.ename
        FROM Employee e,Project_Employee pe  WHERE e.empno = pe.empno
AND pe.pno = project_id;
    LOOP
        RAISE NOTICE ''%'', emp_name;
        emp_count := emp_count + 1;
    END LOOP;
     RAISE NOTICE ''Total employees working on %: %'', pname_input,
emp_count;
END;
' LANGUAGE plpgsql;
```

--TO Call Function

```
SELECT get_employees_by_project('Project A');
```

Q2. **Write a stored function to accept empno as an input parameter from the user and count the number of projects of a given employee. Raise an exception if the employee number is invalid.**

```
CREATE OR REPLACE FUNCTION
count_projects_by_employee(empno_input INT) RETURNS VOID AS '
DECLARE
   project_count INT := 0;
BEGIN
   IF NOT EXISTS (SELECT 1 FROM Employee WHERE empno =
empno_input) THEN
      RAISE EXCEPTION "Invalid employee number: %", empno_input;
   END IF;
   SELECT COUNT(*) INTO project_count
   FROM Project_Employee
   WHERE empno = empno_input;
   RAISE NOTICE "Employee % is working on % projects", empno_input,
project_count;
END;
' LANGUAGE plpgsql;
```

--TO Call Function

```
SELECT count_projects_by_employee(1);
```