SET B

Student –Teacher database

Consider the following database

Teacher( t_no int, t_name varchar(20), age int, yr_experience int)

Subject (s_no int, s_namevarchar(15))

Teacher and Subject are related with many to many relationship

Create the above database in PostGreSQL and insert sufficient records.

a. Write a stored function using cursor which will accept the subject name and print the names of all teachers teaching that subject.

b. Write a cursor to accept the subject's name from the user as an input and display names of all teachers teaching that student.

**ANSWER**

```
CREATE TABLE Teacher (
    t_no INT PRIMARY KEY,
    t_name VARCHAR(20),
    age INT,
    yr_experience INT
);

CREATE TABLE Subject (
    s_no INT PRIMARY KEY,
    s_name VARCHAR(15)
);

CREATE TABLE Teacher_Subject (
    t_no INT,
    s_no INT,
    FOREIGN KEY (t_no) REFERENCES Teacher(t_no),
    FOREIGN KEY (s_no) REFERENCES Subject(s_no),
    PRIMARY KEY (t_no, s_no)
);
```

```
-- Insert data into Teacher table
INSERT INTO Teacher VALUES (1, 'Alice', 35, 10);
INSERT INTO Teacher VALUES ((2, 'Bob', 42, 15);
INSERT INTO Teacher VALUES  (3, 'Charlie', 30, 5);
INSERT INTO Teacher VALUES  (4, 'David', 38, 12);


-- Insert data into Subject table
INSERT INTO Subject VALUES (1, 'Math');
INSERT INTO Subject VALUES (2, 'Physics');
INSERT INTO Subject VALUES  (3, 'Chemistry');


-- Insert data into Teacher_Subject table
INSERT INTO Teacher_Subject VALUES (1, 1);
INSERT INTO Teacher_Subject VALUES (1, 2);
INSERT INTO Teacher_Subject VALUES (2, 1);
INSERT INTO Teacher_Subject VALUES (3, 3);
INSERT INTO Teacher_Subject VALUES (4, 2);
```

## Q1. Write a stored function using cursor which will accept the subject name and print the names of all teachers teaching that subject.

```
CREATE OR REPLACE FUNCTION get_teachers_by_subject(sub_name VARCHAR) RETURNS
Void  AS '

DECLARE

 teacher_name VARCHAR(20);

    teacher_cursor CURSOR FOR SELECT t.t_name FROM Teacher t,Teacher_Subject ts,Subject s

    WHERE s.s_name = sub_name AND t.t_no = ts.t_no AND ts.s_no = s.s_no;

BEGIN

    OPEN teacher_cursor;

    LOOP

        FETCH teacher_cursor INTO teacher_name;

        EXIT WHEN NOT FOUND;

        RAISE NOTICE "Teacher  %", teacher_name;

    END LOOP;

    CLOSE teacher_cursor;

END;

' LANGUAGE plpgsql;
```

### Call the Function

```
SELECT get_teachers_by_subject('Math');
```

## b. Write a cursor to accept the subject's name from the user as an input and display names of all teachers teaching that student.

```
CREATE OR REPLACE FUNCTION get_teachers_for_subject() RETURNS VOID AS '
DECLARE
   subject_name VARCHAR(15);
   teacher_name VARCHAR(20);
   teacher_cursor CURSOR FOR SELECT t.t_name  FROM Teacher t,Teacher_Subject ts,Subject s
   WHERE s.s_name = subject_name AND t.t_no = ts.t_no AND ts.s_no = s.s_no;

BEGIN
     RAISE NOTICE "Enter subject name";
   subject_name := "Math";

   OPEN teacher_cursor;
   LOOP
      FETCH teacher_cursor INTO teacher_name;
      EXIT WHEN NOT FOUND;
      RAISE NOTICE "Teacher: %", teacher_name;
   END LOOP;
   CLOSE teacher_cursor;
END;
' LANGUAGE plpgsql;
```

## Call the Function

```
SELECT get_teachers_for_subject();
```