

Experiment 2

AIM: To Demonstrate Department-Course Subquery and Access Control

Theory: Subqueries in SQL are queries nested inside another query to filter, count, or manipulate data dynamically. They allow efficient data retrieval across related tables. PostgreSQL access control is managed using roles and privileges. The GRANT and REVOKE commands define what actions different users can perform on tables, ensuring database security and proper authorization.

CODE (with output):

1. CREATING THE TBALES:

```
CREATE TABLE Department (dept_id SERIAL PRIMARY KEY, dept_name  
VARCHAR(100) NOT NULL);
```

```
CREATE TABLE Course (course_id SERIAL PRIMARY KEY, course_name  
VARCHAR(100) NOT NULL, dept_id INT REFERENCES Department(dept_id));
```

```
INSERT INTO Department (dept_name) VALUES ('Computer Science'),  
('Electronics'), ('Mechanical');
```

```
INSERT INTO Course (course_name, dept_id) VALUES('DBMS', 1),('Computer  
Networks', 1),('Digital Logic', 2),('Thermodynamics', 3);
```

Data Output [Messages](#) Notifications

INSERT 0 4

Query returned successfully in 54 msec.

✓ Query returned successfully in 54 msec. ✕

2. Subquery examples:

2.1 Finding all courses offered by Computer Science Department

```
SELECT course_name FROM Course WHERE dept_id = ( SELECT  
dept_id FROM Department WHERE dept_name = 'Computer Science');
```

OUTPUT:

Data Output Messages Notifications	
Showing rows: 1 to 2 Page No: 1 of 1	
course_name character varying (100)	
1 DBMS	
2 Computer Networks	

✓ Successfully run. Total query runtime: 83 msec. 2 rows affected. ✕

2.2 Find departments that offer more than 1 course

```
SELECT dept_name FROM Department d WHERE (SELECT COUNT(*)  
FROM Course c WHERE c.dept_id = d.dept_id) > 1;
```

OUTPUT:

Data Output Messages Notifications	
Showing rows: 1 to 1 Page No: 1 of 1	
dept_name character varying (100)	
1 Computer Science	

2.3 Find courses not assigned to any department

```
SELECT course_name FROM Course WHERE dept_id NOT IN (SELECT  
dept_id FROM Department);
```

OUTPUT:

Data Output Messages Notifications	
course_name character varying (100)	

✓ Successfully run. Total query runtime: 126 msec. 0 rows affected. ✕

3. ACCESS CONTROL:

3.1 Create Roles

```
CREATE ROLE student LOGIN PASSWORD 'stud123';  
CREATE ROLE professor LOGIN PASSWORD 'prof123';
```

OUTPUT:

Data Output [Messages](#) Notifications

ERROR: role "professor" already exists

SQL state: 42710



Data Output [Messages](#) Notifications

ERROR: role "student" already exists

SQL state: 42710



3.2 Grant privileges

GRANT SELECT ON Course TO student; -- students can only read courses

GRANT INSERT, UPDATE ON Course TO professor; -- professors can add/update courses

OUTPUT:

Data Output [Messages](#) Notifications

GRANT

Query returned successfully in 84 msec.



✓ Query returned successfully in 84 msec. ✕

3.3 Revoke a privilege

REVOKE UPDATE on Course FROM student;

OUTPUT:

Data Output Messages Notifications



REVOKE

Query returned successfully in 62 msec.

3.4 Grant schema usage

GRANT USAGE on SCHEMA public TO Student;

OUTPUT:

Data Output Messages Notifications



GRANT

Query returned successfully in 63 msec.

✓ Query returned successfully in 63 msec. ✕

4.OPTIONALS(only to be used I case the above commands are not running because of pre existence):

DROP Table Department;

DROP Table Course;

Learning Outcomes

- Understood how **subqueries** can be applied to filter and analyze relational data.
- Learned how to **create roles** and manage **user privileges** using GRANT and REVOKE in PostgreSQL.
- Gained practical knowledge of combining **data retrieval** with **database security** in a single workflow.