

Experiment No. 6

Student Name: Roshan Kumar Singh

Branch: MCA(GEN)

Semester: 2nd

Subject Name: Technical Training-1

UID: 25MCA20067

Section/Group: 25MCA-1_A

Date of Performance: 24/02/26

Subject Code: 25CAP-652

Aim:

To learn how to create, query, and manage views in PostgreSQL in order to simplify database queries and provide a layer of abstraction and security for end-users.

Objective:

- To understand data abstraction using views
- To learn how views enhance database security
- To simplify complex queries using views
- To understand creation, modification, and deletion of views
- To apply views in real-world scenarios like payroll and reporting.

Tools Used:

PostgreSQL

Procedure:

Step 1: Creating a Simple View for Data Filtering

- Create a view to show only active employees
- Hide unnecessary columns

Step 2: Creating a View for Joining Multiple Tables

- Combine employee and department data
- Simplify multi-table queries

Step 3: Advanced Summarization View

- Generate department-level statistics automatically

Code:

```
CREATE TABLE department (  
    dept_id INT PRIMARY KEY,  
    dept_name VARCHAR(50)  
);
```

```
CREATE TABLE employee (  
    emp_id INT PRIMARY KEY,  
    emp_name VARCHAR(50),  
    salary NUMERIC,  
    status VARCHAR(10),  
    dept_id INT,  
    FOREIGN KEY (dept_id) REFERENCES department(dept_id)  
);
```

```
INSERT INTO department VALUES  
(1, 'HR'),  
(2, 'IT'),  
(3, 'Finance');
```

```
INSERT INTO employee VALUES  
(1, 'Roshan', 30000, 'Active', 2),  
(2, 'Swayam', 40000, 'Active', 2),  
(3, 'Riya', 25000, 'Inactive', 1),  
(4, 'Ankush', 35000, 'Active', 3),  
(5, 'Sanchit', 28000, 'Active', 1);
```

--1

```
CREATE VIEW active_employees AS
SELECT emp_id, emp_name, dept_id
FROM employee
WHERE status = 'Active';
```

```
SELECT * FROM active_employees;
```

--2

```
CREATE VIEW employee_department_view AS
SELECT e.emp_id, e.emp_name, d.dept_name
FROM employee e
JOIN department d ON e.dept_id = d.dept_id;
```

```
SELECT * FROM employee_department_view;
```

--3

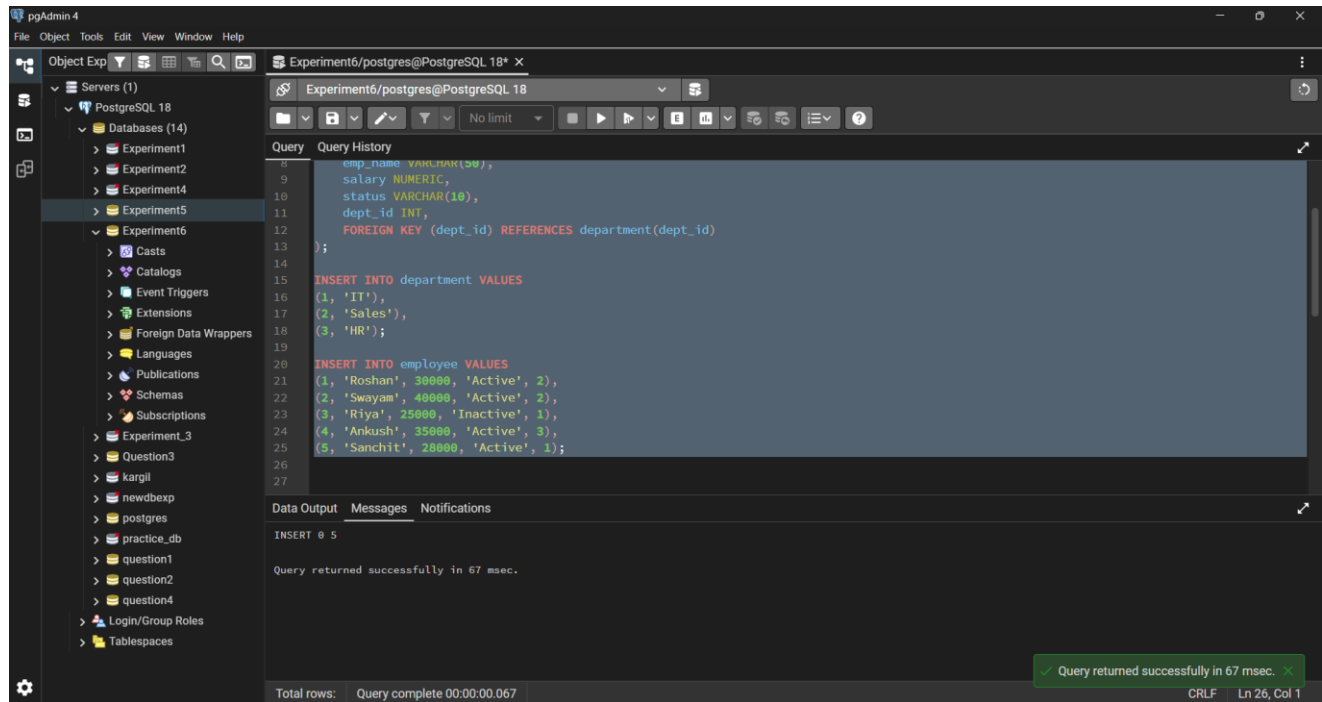
```
CREATE VIEW department_summary AS
SELECT d.dept_name,
       COUNT(e.emp_id) AS total_employees,
       AVG(e.salary) AS average_salary
FROM department d
JOIN employee e ON d.dept_id = e.dept_id
GROUP BY d.dept_name;
```

```
SELECT * FROM department_summary;
```

```
DROP VIEW department_summary;
```

Output:

Table create and data insert



The screenshot shows the pgAdmin 4 interface with a PostgreSQL 18 database. The query editor contains the following SQL code:

```

emp_name VARCHAR(50),
salary NUMERIC,
status VARCHAR(10),
dept_id INT,
FOREIGN KEY (dept_id) REFERENCES department(dept_id)
);

INSERT INTO department VALUES
(1, 'IT'),
(2, 'Sales'),
(3, 'HR');

INSERT INTO employee VALUES
(1, 'Roshan', 30000, 'Active', 2),
(2, 'Swayam', 40000, 'Active', 2),
(3, 'Riya', 25000, 'Inactive', 1),
(4, 'Ankush', 35000, 'Active', 3),
(5, 'Sanchit', 28000, 'Active', 1);

```

The Data Output tab shows the result of the INSERT operation:

```

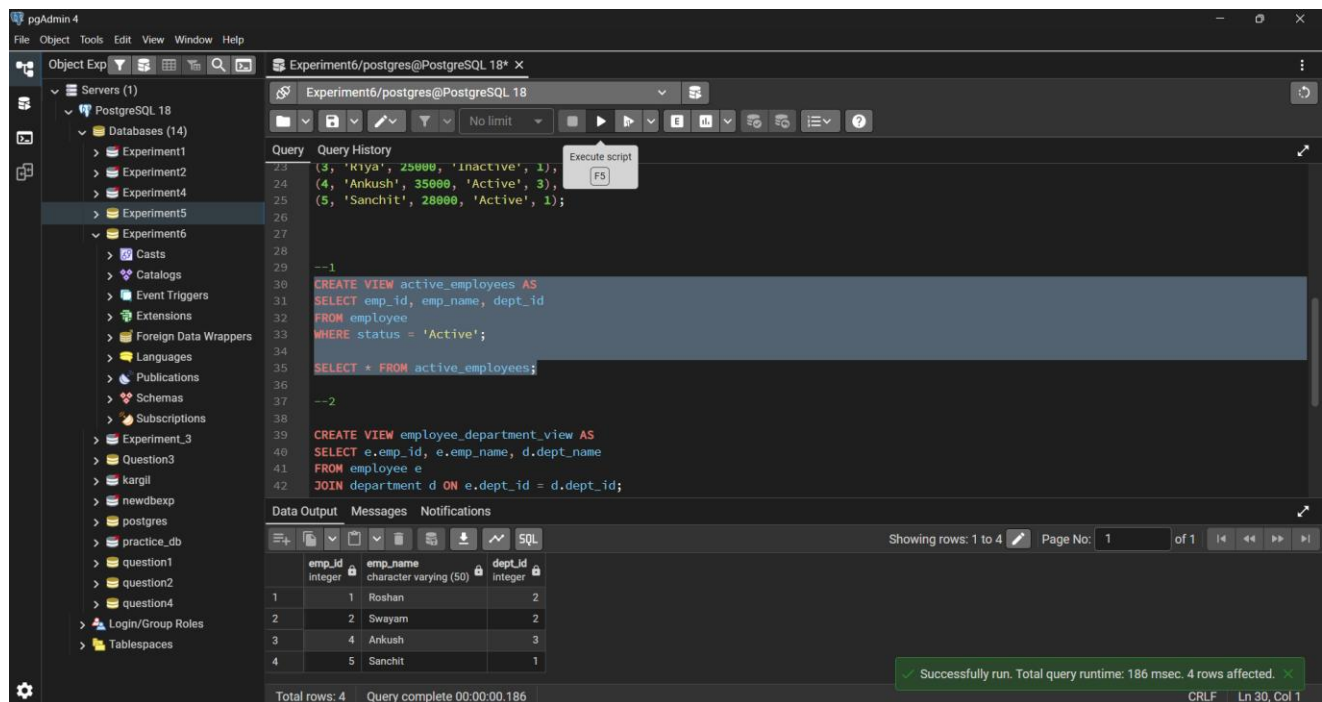
INSERT 0 5

Query returned successfully in 67 msec.

```

A green status bar at the bottom indicates: "Query returned successfully in 67 msec." Total rows: Query complete 00:00:00.067.

Step1: Creating a Simple View for Data Filtering



The screenshot shows the pgAdmin 4 interface with a PostgreSQL 18 database. The query editor contains the following SQL code:

```

--1
CREATE VIEW active_employees AS
SELECT emp_id, emp_name, dept_id
FROM employee
WHERE status = 'Active';

SELECT * FROM active_employees;

--2
CREATE VIEW employee_department_view AS
SELECT e.emp_id, e.emp_name, d.dept_name
FROM employee e
JOIN department d ON e.dept_id = d.dept_id;

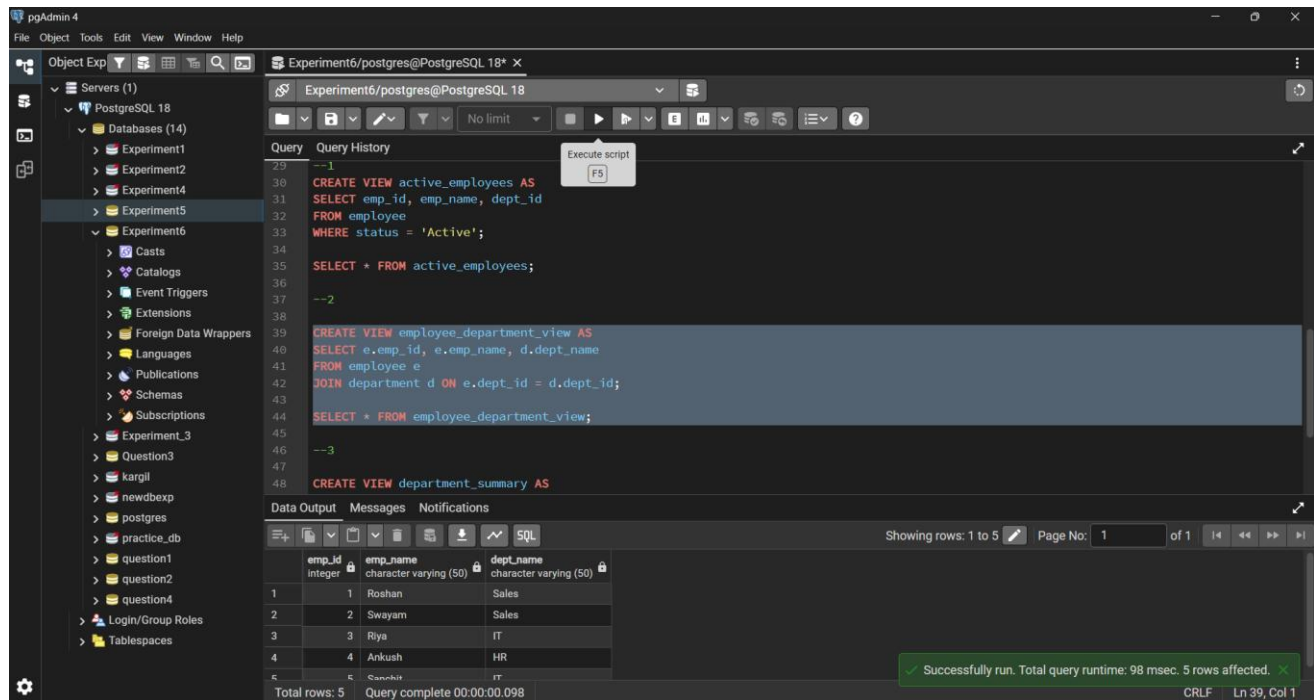
```

The Data Output tab shows the results of the two queries:

emp_id	emp_name	dept_id
1	Roshan	2
2	Swayam	2
3	Ankush	3
4	Sanchit	1

A green status bar at the bottom indicates: "Successfully run. Total query runtime: 186 msec. 4 rows affected." Total rows: 4 Query complete 00:00:00.186.

Step2: Creating a View for Joining Multiple Tables



The screenshot shows the pgAdmin 4 interface with the following SQL query executed:

```
--1
CREATE VIEW active_employees AS
SELECT emp_id, emp_name, dept_id
FROM employee
WHERE status = 'Active';

SELECT * FROM active_employees;

--2
CREATE VIEW employee_department_view AS
SELECT e.emp_id, e.emp_name, d.dept_name
FROM employee e
JOIN department d ON e.dept_id = d.dept_id;

SELECT * FROM employee_department_view;

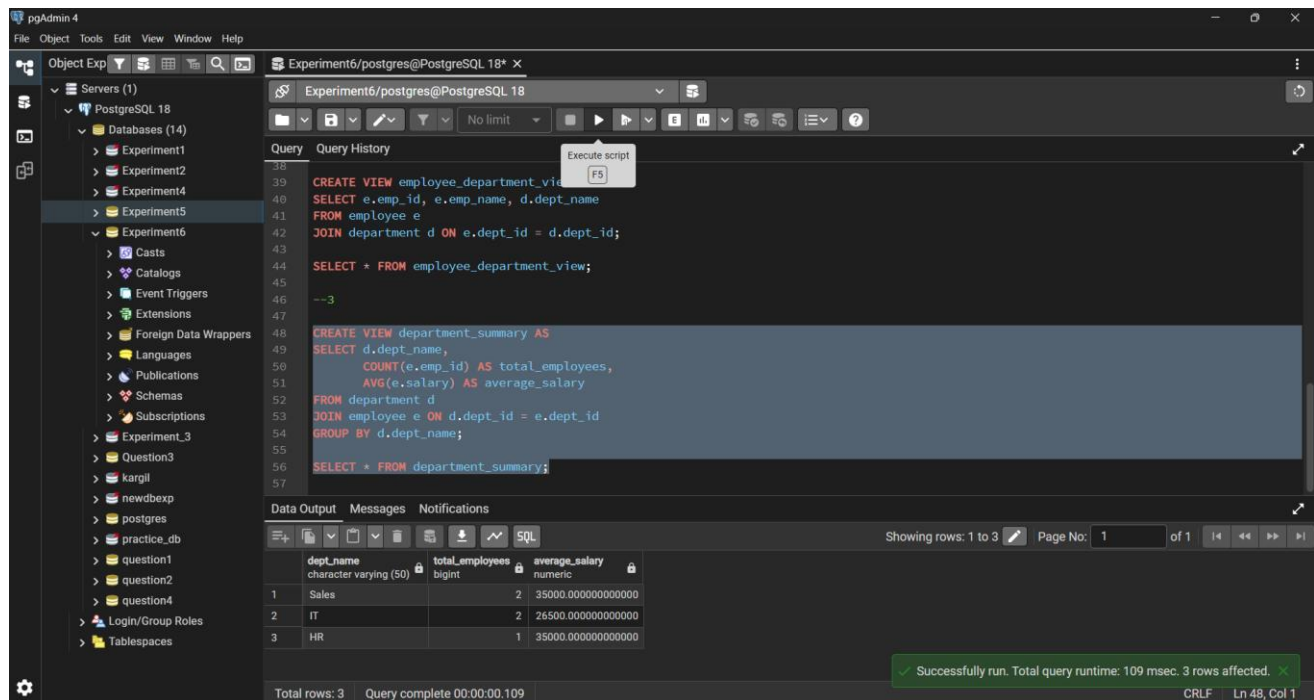
--3
CREATE VIEW department_summary AS
```

The Data Output tab shows the following table:

emp_id	emp_name	dept_name
1	Roshan	Sales
2	Swayam	Sales
3	Riya	IT
4	Ankush	HR

Successfully run. Total query runtime: 98 msec. 5 rows affected.

Step3: Advanced Summarization View



The screenshot shows the pgAdmin 4 interface with the following SQL query executed:

```
CREATE VIEW employee_department_view AS
SELECT e.emp_id, e.emp_name, d.dept_name
FROM employee e
JOIN department d ON e.dept_id = d.dept_id;

SELECT * FROM employee_department_view;

--3
CREATE VIEW department_summary AS
SELECT d.dept_name,
COUNT(e.emp_id) AS total_employees,
AVG(e.salary) AS average_salary
FROM department d
JOIN employee e ON d.dept_id = e.dept_id
GROUP BY d.dept_name;

SELECT * FROM department_summary;
```

The Data Output tab shows the following table:

dept_name	total_employees	average_salary
Sales	2	35000.000000000000
IT	2	26500.000000000000
HR	1	35000.000000000000

Successfully run. Total query runtime: 109 msec. 3 rows affected.

Learning Outcomes:

- Understood the concept of views in PostgreSQL
- Learnt how views provide data abstraction and security
- Created simple and complex views
- Used views for reporting and summarization
- Gained practical knowledge of real-world database design.