Branch: MCA (Data Science)	**Semester:** 2
Student Name: Adarsh Mishra	**UID:** 25MCD10065
Subject Name: Technical Training - I Lab	**Subject Code**: 25CAP-652
Section/Group: 25MCD KAR-1	**Date of Performance:** 24 Februrary,2026

EXPERIMENT – 06

Aim

Learn how to create, query, and manage views in SQL to simplify database queries and provide a layer of abstraction for end-users.

Tools Used

- PostgreSQL

Objectives

- Data Abstraction: To understand how to hide complex table joins and calculations behind a simple virtual table interface.
- Enhanced Security: To learn how to restrict user access to sensitive columns by providing views instead of direct table access.
- Query Simplification: To master the creation of views that pre-join multiple tables, making reporting easier for non-technical users.
- View Management: To understand the syntax for creating, altering, and dropping views, as well as the naming conventions required for efficient data access.

Theory

A **View** is essentially a virtual table based on the result-set of an SQL statement. It does not contain data of its own but dynamically pulls data from the underlying "base tables".

1. **Simple Views:** Created from a single table without any aggregate functions or grouping. These are often updatable.
2. **Complex Views:** Created from multiple tables using JOINs, or including GROUP BY and aggregate functions. These provide a consolidated summary of the database.
3. **Security Layer:** In enterprise environments, views are used to grant permissions on specific subsets of data. For example, a "SalaryView" might exclude the "Employee_SSN" or "Home_Address" columns for privacy.
4. **Benefits:** They simplify the user experience, ensure data consistency across reports, and reduce the risk of accidental data modification by providing read-only abstractions.

.

Experiment Steps:

Step 1: Creating a Simple View for Data Filtering

Implementing a view to provide a quick list of active employees without exposing the entire table structure.

Step 2: Creating a View for Joining Multiple Tables

Simplifying the retrieval of data distributed across Employees and Departments tables.

Step 3: Advanced Summarization View

Creating a view to provide department-level statistics automatically

---

Query:

```
CREATE TABLE employee (
emp_id INT PRIMARY KEY,
```

```
    emp_name VARCHAR(50),
    department VARCHAR(50),
    salary INT
);

INSERT INTO employee VALUES
(1,'Aman','HR',30000),
(2,'Riya','IT',50000),
(3,'Karan','Finance',45000),
(4,'Neha','IT',60000),
(5,'Arjun','HR',35000);

-- Create a View
-- View showing only IT department employees
CREATE OR REPLACE VIEW it_employees AS
SELECT emp_id, emp_name, salary
FROM employee
WHERE department = 'IT';

-- Query the View
SELECT * FROM it_employees;
```

Output:

	emp_id integer	emp_name character varying (50)	salary integer
1	4	Neha	60000
2	2	Riya	50000

```
-- Create Another View (High Salary Employees)
CREATE OR REPLACE VIEW high_salary_employees AS
SELECT emp_id, emp_name, department, salary
FROM employee
WHERE salary > 40000;

-- Query Second View
SELECT * FROM high_salary_employees;
```

Output:

	emp_id integer	emp_name character varying (50)	department character varying (50)	salary integer
1	3	Karan	Finance	45000
2	4	Neha	IT	60000
3	2	Riya	IT	50000

-- Update Data Using View

```
UPDATE it_employees
SET salary = salary + 2000
WHERE emp_name = 'Riya';
```

-- Display Original Table to See Changes

```
SELECT * FROM employee;
```

Output:

	emp_id [PK] integer	emp_name character varying (50)	department character varying (50)	salary integer
1	1	Aman	HR	30000
2	3	Karan	Finance	45000
3	4	Neha	IT	60000
4	5	Arjun	HR	35000
5	2	Riya	IT	52000

-- Drop a View

```
DROP VIEW IF EXISTS high_salary_employees;
```

Output:

```
DROP VIEW

Query returned successfully in 153 msec.
```

Learning Outcomes:

- Understand the concept and purpose of **SQL Views**.
- Create views to simplify complex SQL queries.
- Retrieve data using views instead of direct table access.
- Modify and manage views using SQL commands.
- Understand how views improve **security and data abstraction**.
- Apply views in real-world database applications.