Branch: MCA (Data Science) Kargil	**Semester:** 2
Student Name: Sameer Shahi	**UID:** 25MCD10025
Subject Name: Technical Training - I Lab	**Subject Code**: 25CAP-652
Section/Group: 25MCD KAR-1	**Date of Performance:** 27-02-2026

Experiment 6

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

(Company Tags: Amazon, Zoho, ServiceNow)

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

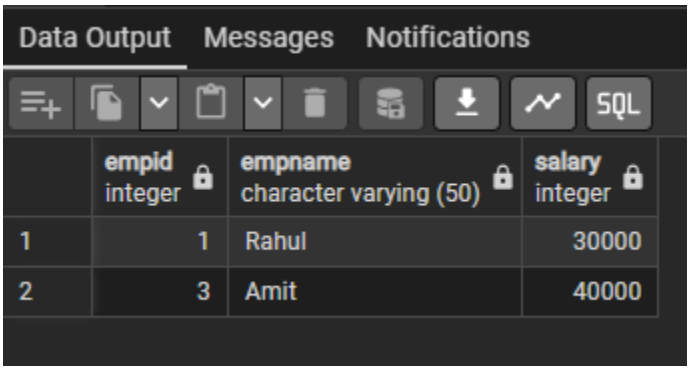
Implementing a view to provide a quick list of active employees without exposing the entire table structure.

Query:

```
-- Query 1

-- Create Employee table
CREATE TABLE IF NOT EXISTS Employee (
    EmpID INT PRIMARY KEY,
    EmpName VARCHAR(50),
    Salary INT,
    Status VARCHAR(20)
);

-- Insert sample data
INSERT INTO Employee VALUES (1, 'Rahul', 30000, 'Active')
ON CONFLICT (EmpID) DO NOTHING;

INSERT INTO Employee VALUES (2, 'Priya', 35000, 'Inactive')
ON CONFLICT (EmpID) DO NOTHING;

INSERT INTO Employee VALUES (3, 'Amit', 40000, 'Active')
ON CONFLICT (EmpID) DO NOTHING;

-- Create View to show only Active Employees
```

```
CREATE OR REPLACE VIEW ActiveEmployees AS

SELECT EmpID, EmpName, Salary

FROM Employee

WHERE Status = 'Active';

-- Query the View

SELECT * FROM ActiveEmployees;
```

Output:



Step 2: Creating a View for Joining Multiple Tables

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

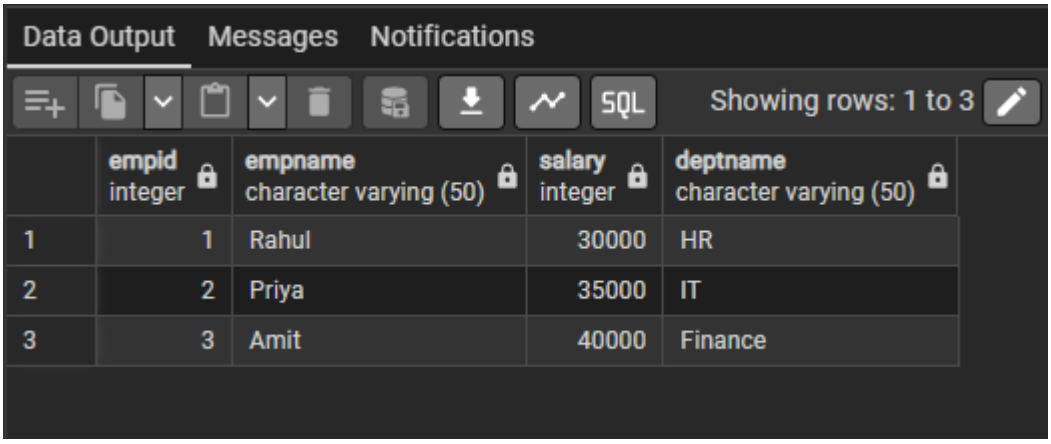
Query:

```
-- Create Department table

CREATE TABLE IF NOT EXISTS Department (

    DeptID INT PRIMARY KEY,

    DeptName VARCHAR(50)
```

```
);

-- Insert sample departments

INSERT INTO Department VALUES (101, 'HR')

ON CONFLICT (DeptID) DO NOTHING;

INSERT INTO Department VALUES (102, 'IT')

ON CONFLICT (DeptID) DO NOTHING;

INSERT INTO Department VALUES (103, 'Finance')

ON CONFLICT (DeptID) DO NOTHING;

-- Add DeptID column to Employee table

ALTER TABLE Employee

ADD COLUMN IF NOT EXISTS DeptID INT;

-- Assign departments to employees

UPDATE Employee SET DeptID = 101 WHERE EmpID = 1;

UPDATE Employee SET DeptID = 102 WHERE EmpID = 2;

UPDATE Employee SET DeptID = 103 WHERE EmpID = 3;

-- Create View joining Employee and Department

CREATE OR REPLACE VIEW EmployeeDepartmentView AS

SELECT
```

```
    e.EmpID,
    e.EmpName,
    e.Salary,
    d.DeptName
FROM Employee e
JOIN Department d
ON e.DeptID = d.DeptID;
-- Query the View
SELECT * FROM EmployeeDepartmentView;
```

Output:



	empid integer	empname character varying (50)	salary integer	deptname character varying (50)
1	1	Rahul	30000	HR
2	2	Priya	35000	IT
3	3	Amit	40000	Finance

Step 3: Advanced Summarization View

Creating a view to provide department-level statistics automatically

Query:

```
-- Create summarization view showing department statistics

CREATE OR REPLACE VIEW DepartmentSummaryView AS

SELECT

    d.DeptName,

    COUNT(e.EmpID) AS TotalEmployees,

    AVG(e.Salary) AS AverageSalary,

    SUM(e.Salary) AS TotalSalary

FROM Employee e

JOIN Department d

ON e.DeptID = d.DeptID

GROUP BY d.DeptName;



-- Query the View

SELECT * FROM DepartmentSummaryView;
```

Output:

Data Output | Messages | Notifications

Showing rows: 1 to 3

	deptname character varying (50)	totalemployees bigint	averagesalary numeric	totalsalary bigint
1	Finance	1	40000.000000000000	40000
2	IT	1	35000.000000000000	35000
3	HR	1	30000.000000000000	30000

Outcomes:

- **Abstraction Proficiency:** Students will be able to create and query views to simplify efficient data access and abstraction.
- **Security Implementation:** Students will understand how to use views for data masking and providing restricted access to sensitive information.
- **Syntactic Accuracy:** Students will demonstrate the correct syntax for creating and querying views, ensuring logical clarity in naming conventions.
- **Real-world Application:** Students will be able to design views for practical domains like Library Management Systems or Payroll Systems to demonstrate functionality.