Experiment 6

Student Name: Yash Karde **UID: 25MCI10090**
Branch: MCA (AI & ML) **Section/Group: 25MAM-1**
Semester: 2nd **Date of Performance:27/02/26**
Subject Name: Technical Training **Subject Code:25CAP-652**

AIM:

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

OBJECTIVE:

- **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.**

Implementation:

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.

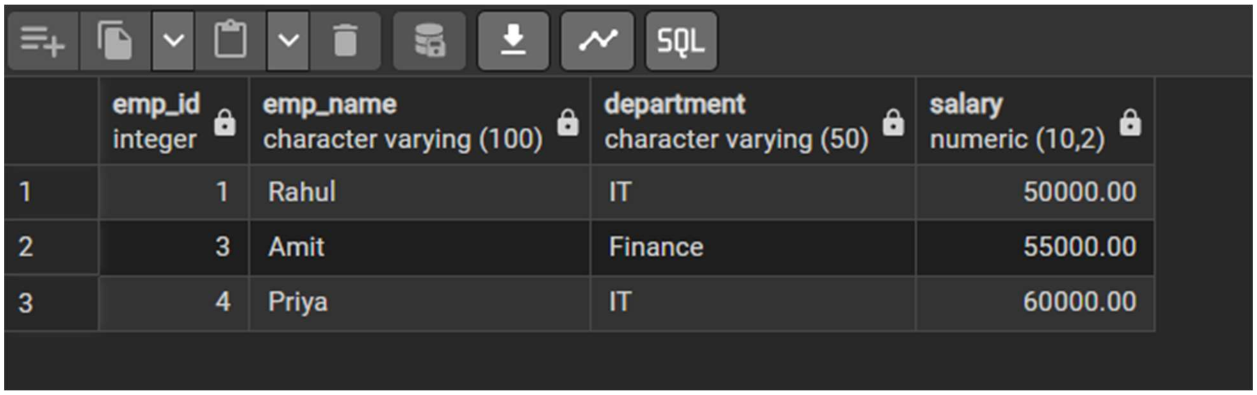
```
CREATE TABLE Employee (
    emp_id SERIAL PRIMARY KEY,
    emp_name VARCHAR(100),
    department VARCHAR(50),
    salary NUMERIC(10,2),
    is_active BOOLEAN
);
```

```
-- Data Insertion

INSERT INTO Employee (emp_name, department, salary, is_active) VALUES
('Rahul', 'IT', 50000, TRUE),
('Sneha', 'HR', 40000, FALSE),
('Amit', 'Finance', 55000, TRUE),
('Priya', 'IT', 60000, TRUE);-- Cursor Commands
```

Query : -

```
CREATE OR REPLACE VIEW active_employees AS
SELECT emp_id, emp_name, department, salary
FROM Employee
WHERE is_active = TRUE;

SELECT * FROM active_employees;
```

Output :-



	emp_id integer	emp_name character varying (100)	department character varying (50)	salary numeric (10,2)
1	1	Rahul	IT	50000.00
2	3	Amit	Finance	55000.00
3	4	Priya	IT	60000.00

Step 2: Creating a View for Joining Multiple Tables

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

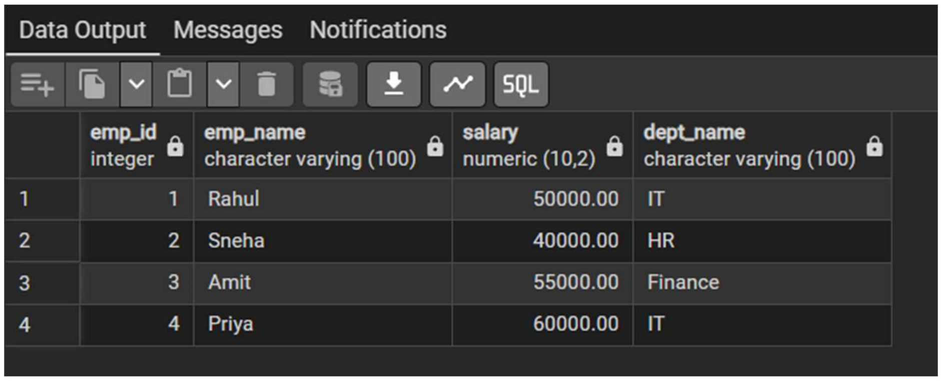
```
CREATE TABLE Departments (
   dept_id SERIAL PRIMARY KEY,
   dept_name VARCHAR(100)
);
```

```
INSERT INTO Departments (dept_name) VALUES
('IT'),
('HR'),
('Finance');

CREATE TABLE Employees (
   emp_id SERIAL PRIMARY KEY,
   emp_name VARCHAR(100),
   salary NUMERIC(10,2),
   dept_id INT,
   FOREIGN KEY (dept_id) REFERENCES Departments(dept_id)
);

INSERT INTO Employees (emp_name, salary, dept_id) VALUES
('Rahul', 50000, 1),
('Sneha', 40000, 2),
('Amit', 55000, 3),
('Priya', 60000, 1);
```

Query:-

```
CREATE OR REPLACE VIEW employee_department_view AS
SELECT
   e.emp_id,
   e.emp_name,
   e.salary,
   d.dept_name
FROM Employees e
JOIN Departments d
ON e.dept_id = d.dept_id;

SELECT * FROM employee_department_view;
```

Output :-



Data Output Messages Notifications

	emp_id integer	emp_name character varying (100)	salary numeric (10,2)	dept_name character varying (100)
1	1	Rahul	50000.00	IT
2	2	Sneha	40000.00	HR
3	3	Amit	55000.00	Finance
4	4	Priya	60000.00	IT

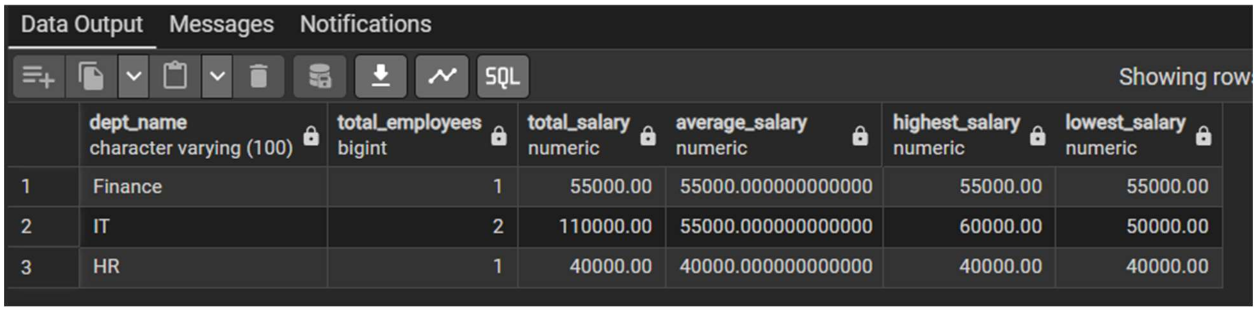
Step 3: Advanced Summarization View

Creating a view to provide department-level statistics automatically

Query:-

```
CREATE OR REPLACE VIEW department_statistics AS
SELECT
    d.dept_name,
    COUNT(e.emp_id) AS total_employees,
    SUM(e.salary) AS total_salary,
    AVG(e.salary) AS average_salary,
    MAX(e.salary) AS highest_salary,
    MIN(e.salary) AS lowest_salary
FROM Departments d
LEFT JOIN Employees e
ON d.dept_id = e.dept_id
GROUP BY d.dept_name;
```

Output : -



Data Output Messages Notifications

Showing row

	dept_name character varying (100)	total_employees bigint	total_salary numeric	average_salary numeric	highest_salary numeric	lowest_salary numeric
1	Finance	1	55000.00	55000.000000000000	55000.00	55000.00
2	IT	2	110000.00	55000.000000000000	60000.00	50000.00
3	HR	1	40000.00	40000.000000000000	40000.00	40000.00

<u>LEARNING OUTCOMES:</u>

- **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.**