POSTGRESQL ASSIGNMENT 3

Explanation of Queries

1. Aggregated Data with GROUP BY

Query:

```
SELECT
    E.DepartmentID,
    COUNT(E.EmployeeID) AS TotalEmployee,
    AVG(E.Salary) AS AvgSalary
FROM
    Employees E
    INNER JOIN
    Departments D
    ON E.DepartmentID = D.DepartmentID
GROUP BY
    E.DepartmentID
HAVING
    COUNT(E.EmployeeID) > 5
```

```
ORDER BY
    TotalEmployee DESC;
```

- **Requirements Addressed**:
  - **GROUP BY**: Groups employees by department.
  - **HAVING**: Filters departments where the total number of employees is greater than 5.
  - **Sorting**: Orders by the total number of employees in descending order.
- **Execution Plan**:
  - **Join**: Inner join matches rows from `Employees` and `Departments` using `DepartmentID`.
  - **Aggregate Functions**: `COUNT()` calculates total employees, and `AVG()` calculates average salary.
  - **Filter**: The `HAVING` clause eliminates groups with 5 or fewer employees.
  - **Sort**: The query sorts the results by `TotalEmployee`.

---

2. Filtered Aggregation Using an Inner Query

Query:

```
SELECT
    E.DepartmentID,
    D.DepartmentName,
    AVG(E.Salary) AS AvgSalary
FROM
    Employees E
    INNER JOIN
    Departments D
    ON E.DepartmentID = D.DepartmentID
GROUP BY
    E.DepartmentID, D.DepartmentName
```

```
HAVING
    AVG(E.Salary) > (SELECT AVG(Salary) FROM Employees);
```

- **Requirements Addressed**:
  - **Inner Query**: Calculates the overall average salary across the company.
  - **GROUP BY**: Groups employees by `DepartmentID` and `DepartmentName`.
  - **HAVING**: Filters departments where the average salary exceeds the overall average salary.
- **Execution Plan**:
  - **Subquery Execution**: Calculates overall average salary before the main query executes.
  - **Join**: Matches rows from `Employees` and `Departments`.
  - **Filter**: The `HAVING` clause applies after the grouping.

---

3. Nested Query with HAVING

Query:

```
SELECT
    P.ProjectID,
    P.ProjectName,
    D.DepartmentName
FROM
    Projects P
    INNER JOIN Departments D
    ON P.DepartmentID = D.DepartmentID
WHERE D.DepartmentID IN (
    SELECT
        E.DepartmentID
    FROM
        Employees E
    GROUP BY
        E.DepartmentID
    HAVING
        AVG(E.Salary) > 75000
);
```

- **Requirements Addressed**:
  - **Nested Query**: Identifies departments with an average salary > 75,000.
  - **HAVING**: Filters departments based on the salary condition.
  - **INNER JOIN**: Matches the filtered departments with their projects.
- **Execution Plan**:
  - **Subquery Execution**: Groups `Employees` by `DepartmentID` and calculates average salary.
  - **Filter**: Retains only departments satisfying the salary condition.
  - **Join**: Links projects to departments for the final result.

---

4. Advanced Grouping and Filtering

Query:

```
SELECT
    D.DepartmentID,
    D.DepartmentName,
    COUNT(E.EmployeeID) AS EmployeeEarnOver90k,
    AVG(E.Salary) AS AvgSalary
FROM
    Departments D
    INNER JOIN Employees E
    ON D.DepartmentID = E.DepartmentID
WHERE
    E.Salary > 90000
GROUP BY
    D.DepartmentID, D.DepartmentName
HAVING COUNT(E.EmployeeID) >= 2;
```

- **Requirements Addressed**:
    - **WHERE Clause**: Filters employees earning more than 90,000.
    - **GROUP BY**: Groups data by department.
    - **HAVING**: Retains departments with at least 2 employees meeting the salary condition.
    - **Aggregate Functions**: Counts eligible employees and calculates their average salary.
- **Execution Plan**:
    - **Filter**: Excludes rows with salaries ≤ 90,000 early in the process.
    - **Join**: Matches departments with eligible employees.
    - **Group and Aggregate**: Applies grouping and calculates aggregate values.
    - **Filter Groups**: Retains groups with at least 2 employees.

---

5. Combining HAVING with Multiple Conditions

Query:

```
SELECT
    D.DepartmentID,
    SUM(E.Salary) AS TotalSalary,
    D.DepartmentName
FROM
    Employees E
    INNER JOIN Departments D
    ON E.DepartmentID = D.DepartmentID
GROUP BY
    D.DepartmentID, D.DepartmentName
HAVING
    COUNT(E.EmployeeID) > 10
    AND D.DepartmentName LIKE '%Tech%'
ORDER BY
```

```
    TotalSalary DESC
LIMIT 3;
```

- **Requirements Addressed**:
  - **GROUP BY**: Groups employees by department.
  - **HAVING**: Combines conditions:
    - Employee count > 10.
    - Department name contains "Tech."
  - **Sorting and Limiting**: Sorts by total salary in descending order and limits results to the top 3.
- **Execution Plan**:
  - **Filter**: Applies `HAVING` after grouping to refine results.
  - **Sort**: Orders by `TotalSalary` in descending order.
  - **Limit**: Returns only the top 3 rows.
