**Lab 2 - Working with Joins & Aggregations**

**1. Introduction**

This lab builds upon the fundamentals of SQL by introducing JOIN operations and aggregate functions. Students will learn how to retrieve related data from multiple tables and perform aggregations using SQL functions.

**2. Learning Outcomes**

By completing this lab, students will:

* Understand different types of SQL JOINs (INNER, LEFT, RIGHT, FULL OUTER).
* Perform data aggregation using COUNT, SUM, AVG, MIN, and MAX.
* Use GROUP BY and HAVING clauses for grouped data analysis.
* Apply complex queries involving multiple tables.

**3. Lab Setup and Configuration**

**3.1 Required Software**

* MySQL Workbench / PostgreSQL pgAdmin
* MySQL Server / PostgreSQL
* Any Code Editor (VS Code, Sublime, etc.)

**3.2 Database Setup**

1. Create a new database company:

CREATE DATABASE company;

*This command creates a new database named company where all tables will be stored.*

1. Switch to the newly created database:

USE company; -- For MySQL

\c company -- For PostgreSQL

*This selects the company database so that all subsequent commands apply to it.*

1. Create employees and departments tables:

CREATE TABLE departments (

dept\_id INT PRIMARY KEY,

dept\_name VARCHAR(100)

);

CREATE TABLE employees (

emp\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

age INT,

dept\_id INT,

salary DECIMAL(10,2),

FOREIGN KEY (dept\_id) REFERENCES departments(dept\_id)

);

*The departments table stores department details, while employees holds employee records linked to departments using a foreign key.*

1. Insert sample data:

INSERT INTO departments (dept\_id, dept\_name) VALUES

(1, 'IT'),

(2, 'HR'),

(3, 'Finance');

INSERT INTO employees (emp\_id, first\_name, last\_name, age, dept\_id, salary) VALUES

(101, 'Alice', 'Johnson', 25, 1, 60000.00),

(102, 'Bob', 'Smith', 30, 2, 50000.00),

(103, 'Charlie', 'Davis', 28, 1, 65000.00),

(104, 'David', 'Brown', 35, 3, 70000.00);

*These commands add sample department and employee records to the tables.*

**4. Lab Tasks**

**Task 1: Retrieving Data using INNER JOIN**

SELECT employees.first\_name, employees.last\_name, departments.dept\_name

FROM employees

INNER JOIN departments ON employees.dept\_id = departments.dept\_id;

*INNER JOIN retrieves records where there is a match between employees.dept\_id and departments.dept\_id.*

**Task 2: Using LEFT JOIN**

SELECT departments.dept\_name, employees.first\_name, employees.last\_name

FROM departments

LEFT JOIN employees ON departments.dept\_id = employees.dept\_id;

*A LEFT JOIN returns all records from departments, with matching records from employees if available.*

**Task 3: Using RIGHT JOIN**

SELECT employees.first\_name, employees.last\_name, departments.dept\_name

FROM employees

RIGHT JOIN departments ON employees.dept\_id = departments.dept\_id;

*A RIGHT JOIN returns all records from employees, with matching departments records if available.*

**Task 4: Using FULL OUTER JOIN (PostgreSQL only)**

SELECT employees.first\_name, employees.last\_name, departments.dept\_name

FROM employees

FULL OUTER JOIN departments ON employees.dept\_id = departments.dept\_id;

*A FULL OUTER JOIN returns all records from both tables, filling in NULL where no match is found.*

**Task 5: Using CROSS JOIN**

SELECT employees.first\_name, employees.last\_name, departments.dept\_name

FROM employees

CROSS JOIN departments;

*A CROSS JOIN returns the Cartesian product of both tables, pairing each employee with each department.*

**Task 6: Counting Employees per Department**

SELECT departments.dept\_name, COUNT(employees.emp\_id) AS employee\_count

FROM departments

LEFT JOIN employees ON departments.dept\_id = employees.dept\_id

GROUP BY departments.dept\_name;

*COUNT() returns the number of employees per department.*

**Task 7: Calculating Average Salary by Department**

SELECT departments.dept\_name, AVG(employees.salary) AS avg\_salary

FROM employees

INNER JOIN departments ON employees.dept\_id = departments.dept\_id

GROUP BY departments.dept\_name;

*AVG() calculates the average salary per department.*

**Task 8: Finding Maximum Salary per Department**

SELECT departments.dept\_name, MAX(employees.salary) AS max\_salary

FROM employees

INNER JOIN departments ON employees.dept\_id = departments.dept\_id

GROUP BY departments.dept\_name;

*MAX() finds the highest salary in each department.*

**Task 9: Finding Minimum Salary per Department**

SELECT departments.dept\_name, MIN(employees.salary) AS min\_salary

FROM employees

INNER JOIN departments ON employees.dept\_id = departments.dept\_id

GROUP BY departments.dept\_name;

*MIN() finds the lowest salary in each department.*

**Task 10: Using HAVING Clause**

SELECT departments.dept\_name, COUNT(employees.emp\_id) AS employee\_count

FROM employees

INNER JOIN departments ON employees.dept\_id = departments.dept\_id

GROUP BY departments.dept\_name

HAVING COUNT(employees.emp\_id) > 1;

*HAVING filters groups based on aggregate values, here selecting departments with more than one employee.*

**5. Deliverable**

Students must create a new database school and design teachers and subjects tables. They must:

1. Insert at least 5 records in each table.
2. Retrieve teacher names with their respective subjects.
3. Count the number of teachers per subject.
4. Retrieve subjects with more than one teacher.
5. Drop both tables at the end.

**Submission:** Students must submit a text file containing all executed queries and their respective outputs.

This completes the Week 2 lab session.