SQL Lab on Aggregate Functions and Data Constraints

Introduction

This SQL lab will cover the following topics:

1. Common Aggregate Functions: `COUNT()`, `SUM()`, `AVG()`, `MIN()`, `MAX()`.
2. SQL Data Constraints: Primary Key, Foreign Key, Composite Key, Unique, Not Null, Check, Default.

Aggregate Functions

Aggregate functions perform calculations on a set of values and return a single value. The most common aggregate functions in SQL are:

1. `COUNT()`: Returns the number of rows.
2. `SUM()`: Returns the sum of a numeric column.
3. `AVG()`: Returns the average value of a numeric column.
4. `MIN()`: Returns the minimum value in a set of values.
5. `MAX()`: Returns the maximum value in a set of values.

Data Constraints

Data constraints enforce rules on the data in a database. Constraints are used to ensure data integrity. Common constraints include:

1. Primary Key: Uniquely identifies each record in a table.
2. Foreign Key: Ensures referential integrity by linking one table to another.
3. Composite Key: A primary key composed of multiple columns.
4. Unique: Ensures all values in a column are unique.
5. Not Null: Ensures a column cannot have NULL values.
6. Check: Ensures all values in a column satisfy a specific condition.
7. Default: Sets a default value for a column when no value is specified.

SQL Lab Exercises

1. Setup

Create a new database and necessary tables.

```
-- Create database
CREATE DATABASE CompanyDB;

USE CompanyDB;

-- Create Departments table
CREATE TABLE Departments (
        department_id INT AUTO_INCREMENT,
        department_name VARCHAR(50),
        PRIMARY KEY (department_id)
);

-- Insert data into Departments Table
INSERT INTO Departments (department_name) VALUES ('HR'), ('IT'), ('Finance');

-- Create Employees table
CREATE TABLE Employees (
        employee_id INT AUTO_INCREMENT,
        first_name VARCHAR(50),
        last_name VARCHAR(50),
        department_id INT,
        salary DECIMAL(10, 2),
        hire_date DATE,
        PRIMARY KEY (employee_id),
        FOREIGN KEY (department_id) REFERENCES Departments(department_id)
);

-- Insert data into Employees table
INSERT INTO Employees (first_name, last_name, department_id, salary, hire_date) VALUES
('John', 'Doe', 1, 50000, '2020-01-15'),
('Jane', 'Smith', 2, 60000, '2019-03-22'),
('Emily', 'Jones', 3, 75000, '2018-05-30'),
('Michael', 'Brown', 2, 80000, '2021-07-14'),
('Sarah', 'Davis', 1, 55000, '2020-11-11'),
('David', 'Wilson', 3, 72000, '2017-08-19'),
('Laura', 'Martinez', 1, 58000, '2019-10-05');
```

2. Aggregate Functions

– Write a query that Returns a Count of the number of employees name the column ***total_employees***

```
SELECT COUNT(*) AS total_employees FROM Employees;
```

– Write a query that Calculates the total salary of all employees name the column ***total_salary***

```
SELECT SUM(salary) AS total_salary FROM Employees;
```

-- Write a query that Calculates the average salary of all employees name the column ***average_salary***

```
SELECT AVG(salary) AS average_salary FROM Employees;
```

-- Write a query that Calculates the minimum salary name the column ***minimum_salary***

```
SELECT MIN(salary) AS minimum_salary FROM Employees;
```

--Write a query that Calculates the the maximum salary name the column ***maximum_salary***

```
SELECT MAX(salary) AS maximum_salary FROM Employees;
```

3. Data Constraints

Create a new table `Projects` to demonstrate various data constraints.

-- Create Projects table

```
CREATE TABLE Projects (
        project_id INT AUTO_INCREMENT,
        project_name VARCHAR(100) NOT NULL,
        start_date DATE DEFAULT CURDATE(),
        end_date DATE,
        budget DECIMAL(10, 2) CHECK (budget > 0),
        PRIMARY KEY (project_id),
        UNIQUE (project_name)
);
```

-- Insert data into Projects

```
INSERT INTO Projects (project_name, end_date, budget) VALUES
('Project Alpha', '2022-12-31', 100000),
('Project Beta', '2023-06-30', 200000),
('Project Gamma', '2024-03-15', 150000),
('Project Delta', '2023-09-20', 250000),
('Project Epsilon', '2024-11-25', 175000),
('Project Zeta', '2025-02-10', 300000),
('Project Eta', '2023-05-05', 120000);
```

Composite Key

A composite key is a primary key that consists of multiple columns. Let's create an `EmployeeProjects` table to track which employees are assigned to which projects.

Each employee can be assigned to multiple projects, and each project can have multiple employees.

```
-- Create EmployeeProjects table
CREATE TABLE EmployeeProjects (
        employee_id INT,
        project_id INT,
        assignment_date DATE,
        PRIMARY KEY (employee_id, project_id),
        FOREIGN KEY (employee_id) REFERENCES Employees(employee_id),
        FOREIGN KEY (project_id) REFERENCES Projects(project_id)
);

-- Assign employees to projects
INSERT INTO EmployeeProjects (employee_id, project_id, assignment_date) VALUES
(1, 1, '2021-01-01'),
(2, 1, '2021-02-01'),
(3, 2, '2021-03-01'),
(4, 2, '2021-04-01'),
(5, 3, '2021-05-01'),
(6, 4, '2021-06-01'),
(7, 5, '2021-07-01'),
(1, 6, '2021-08-01'),
(2, 6, '2021-09-01'),
(3, 7, '2021-10-01');
```

4. Queries Involving Constraints

Perform the following queries to understand the impact of constraints:

Primary Key and Foreign Key

```
-- Retrieve all employees and their department names

SELECT e.first_name, e.last_name, d.department_name
FROM Employees e
JOIN Departments d ON e.department_id = d.department_id;
```

Unique and Check Constraints

-- Attempt to insert a project with a negative budget.

INSERT INTO Projects (project_name, end_date, budget) VALUES ('Project Gamma', '2024-12-31', -50000);

This query fails because of the `CHECK (budget > 0)` constraint. The negative budget value `-50000` violates this constraint, preventing the insertion.

-- Attempt to insert a duplicate project name.

INSERT INTO Projects (project_name, end_date, budget) VALUES ('Project Alpha', '2024-12-31', 150000);

This query fails because of the `UNIQUE` constraint on the `project_name` column. Attempting to insert another project named 'Project Alpha' violates this constraint, preventing the insertion.

5. Composite Key Use Case

Retrieve all projects and the employees assigned to them.

```
SELECT p.project_name, e.first_name, e.last_name, ep.assignment_date
FROM EmployeeProjects ep
JOIN Projects p ON ep.project_id = p.project_id
JOIN Employees e ON ep.employee_id = e.employee_id;
```

We will take a look into Joins in the next lab.

Happy Coding!!