A Comprehensive Guide to SQL

Your Name

December 15, 2024

1 Introduction

SQL (Structured Query Language) is a standard language for accessing and manipulating databases. It allows you to create, read, update, and delete data in a relational database management system (RDBMS).

2 Installation

To use SQL, you need a database management system (DBMS). Popular options include MySQL, PostgreSQL, and SQLite. You can install SQLite for quick setup.

3 Basic SQL Commands

3.1 Creating a Table

You can create a table using the `CREATE TABLE` statement.

```
CREATE TABLE Employees (
    ID INT PRIMARY KEY,
    Name VARCHAR(100),
    Age INT,
    City VARCHAR(100),
    Salary DECIMAL(10, 2)
);
```

3.1.1 Output

The output of this command does not produce a result set but creates a new table named ‘Employees‘.

3.2 Inserting Data

You can insert data into a table using the `INSERT INTO` statement.

```
INSERT INTO Employees (ID, Name, Age, City, Salary)
VALUES (1, 'Alice', 25, 'New York', 70000),
       (2, 'Bob', 30, 'Los Angeles', 80000),
       (3, 'Charlie', 35, 'Chicago', 90000),
       (4, 'David', 28, 'Miami', 100000),
       (5, 'Eva', 22, 'Seattle', 95000);
```

3.2.1 Output

The output of this command does not produce a result set but inserts the specified records into the 'Employees' table.

3.3 Querying Data

You can retrieve data using the `SELECT` statement.

```
SELECT * FROM Employees;
```

3.3.1 Output

The output of the above command will be:

ID	Name	Age	City	Salary
1	Alice	25	New York	70000.00
2	Bob	30	Los Angeles	80000.00
3	Charlie	35	Chicago	90000.00
4	David	28	Miami	100000.00
5	Eva	22	Seattle	95000.00

4 Joins

Joins are used to combine rows from two or more tables based on a related column.

4.1 Creating a Second Table

Let's create a second table named 'Departments'.

```
CREATE TABLE Departments (
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(100)
);
```

4.1.1 Inserting Data into Departments

You can insert data into the ‘Departments‘ table.

```
INSERT INTO Departments (DeptID, DeptName)
VALUES (1, 'HR'),
       (2, 'Engineering'),
       (3, 'Marketing');
```

4.2 Performing a Join

You can perform an inner join to combine data from both tables.

```
SELECT e.Name, e.Age, e.City, e.Salary, d.DeptName
FROM Employees e
JOIN Departments d ON e.ID = d.DeptID;
```

4.2.1 Output

The output of the above command will be:

Name	Age	City	Salary	DeptName
Alice	25	New York	70000.00	HR
Bob	30	Los Angeles	80000.00	Engineering
Charlie	35	Chicago	90000.00	Marketing

5 Calculations and OVER() Clause

You can perform calculations and use the ‘OVER()‘ clause for window functions.

5.1 Calculating Average Salary

Let’s calculate the average salary of employees and use the ‘OVER()‘ clause to add it as a new column.

```
SELECT *,
       AVG(Salary) OVER () AS AvgSalary
FROM Employees;
```

5.1.1 Output

The output of the above command will be:

ID	Name	Age	City	Salary	AvgSalary
1	Alice	25	New York	70000.00	82500.00
2	Bob	30	Los Angeles	80000.00	82500.00
3	Charlie	35	Chicago	90000.00	82500.00
4	David	28	Miami	100000.00	82500.00
5	Eva	22	Seattle	95000.00	82500.00

6 Frequently Used Procedures and Workflows

Here are some common SQL workflows that can be useful for data analysis:

- **Data Ingestion:** Use `INSERT` statements to load data into your tables.
- **Data Cleaning:** Use `UPDATE` and `DELETE` statements to clean your data.
- **Data Transformation:** Use `SELECT` with calculations to create derived columns.
- **Data Aggregation:** Use `GROUP BY` and aggregate functions (e.g., `SUM`, `AVG`) to summarize your data.
- **Data Analysis:** Use joins to combine data from multiple tables and gain insights.
- **Window Functions:** Use the `OVER()` clause for running totals, moving averages, etc.

7 Comprehensive Example: Employee Analysis Workflow

In this section, we will apply everything we have learned using a sample dataset. We will perform the following steps:

1. Create the 'Employees' and 'Departments' tables. 2. Insert data into both tables. 3. Perform joins and calculations. 4. Analyze the data with aggregate functions.

7.1 Step 1: Create Tables

Let's create the 'Employees' and 'Departments' tables.

```
CREATE TABLE Employees (
    ID INT PRIMARY KEY,
    Name VARCHAR(100),
    Age INT,
    City VARCHAR(100),
    Salary DECIMAL(10, 2)
);

CREATE TABLE Departments (
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(100)
);
```

7.2 Step 2: Insert Data

We will insert data into both tables.

```
INSERT INTO Employees (ID, Name, Age, City, Salary)
VALUES (1, 'Alice', 25, 'New York', 70000),
       (2, 'Bob', 30, 'Los Angeles', 80000),
       (3, 'Charlie', 35, 'Chicago', 90000),
       (4, 'David', 28, 'Miami', 100000),
       (5, 'Eva', 22, 'Seattle', 95000);

INSERT INTO Departments (DeptID, DeptName)
VALUES (1, 'HR'),
       (2, 'Engineering'),
       (3, 'Marketing');
```

7.3 Step 3: Perform Joins and Calculations

We will perform an inner join to combine data from both tables and calculate average salaries.

```
SELECT e.Name, e.Age, e.City, e.Salary, d.DeptName,
       AVG(e.Salary) OVER () AS AvgSalary
FROM Employees e
JOIN Departments d ON e.ID = d.DeptID;
```

7.3.1 Output

The output of the above command will be:

Name	Age	City	Salary	DeptName	AvgSal
Alice	25	New York	70000.00	HR	82500.
Bob	30	Los Angeles	80000.00	Engineering	82500.
Charlie	35	Chicago	90000.00	Marketing	82500.

8 Conclusion

This comprehensive guide provided an overview of SQL, covering basic commands for data manipulation, the use of joins, calculations, and analytical functions with the ‘OVER()‘ clause. By practicing these concepts, you will strengthen your understanding of SQL and gain confidence in using it for data analysis tasks.