Functions in SQL

Aggregate Functions

Aggregate functions perform calculations on a group of values and return a single result.

1. `SUM()`
    - Calculates the total sum of a numeric column.
    - Example:

      ```
      SELECT SUM(Salary) AS TotalSalary FROM Employees;
      ```

2. `AVG()`
    - Returns the average value of a numeric column.
    - Example:

      ```
      SELECT AVG(Age) AS AverageAge FROM Employees;
      ```

3. `COUNT()`
    - Returns the number of rows that match a condition.
    - Example:

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

4. `MAX()`
    - Returns the highest value in a column.
    - Example:

      ```
      SELECT MAX(Salary) AS HighestSalary FROM Employees;
      ```

5. `MIN()`
    - Returns the lowest value in a column.
    - Example:

      ```
      SELECT MIN(Salary) AS LowestSalary FROM Employees;
      ```

Scalar Functions

Scalar functions operate on a single value and return a single value.

String Functions

1. `LEN()`
    - Returns the length of a string.
    - Example:

      ```
      SELECT LEN(Name) AS NameLength FROM Employees;
      ```

2. `CONCAT()`

- Combines two or more strings into one.
- Example:

```
SELECT CONCAT(FirstName, ' ', LastName) AS FullName FROM Employees;
```

3. SUBSTRING()

- Extracts a portion of a string.
- Example:

```
SELECT SUBSTRING(Name, 1, 3) AS FirstThreeLetters FROM Employees;
```

Date Functions

1. NOW()

- Returns the current date and time.
- Example:

```
SELECT NOW() AS CurrentDateTime;
```

2. CURDATE()

- Returns the current date.
- Example:

```
SELECT CURDATE() AS CurrentDate;
```

3. DATE_FORMAT()

- Formats a date based on a specific pattern.
- Example:

```
SELECT DATE_FORMAT(JoiningDate, '%d-%m-%Y') AS FormattedDate FROM
Employees;
```

Mathematical Functions

1. ROUND()

- Rounds a number to a specified number of decimal places.
- Example:

```
SELECT ROUND(Salary, 2) AS RoundedSalary FROM Employees;
```

2. ABS()

- Returns the absolute value of a number.
- Example:

```
SELECT ABS(-100) AS AbsoluteValue;
```

3. POWER()

- Returns the value of a number raised to the power of another number.
- Example:

```
SELECT POWER(2, 3) AS PowerResult;
```

Joins and Relationships

Understanding Relationships in Databases

- Relationships link data between two or more tables using keys.
- **Primary Key**: Uniquely identifies a row in a table.
- **Foreign Key**: Refers to the primary key in another table to establish a relationship.

Types of Joins

Joins combine rows from two or more tables based on related columns.

1. **Inner Join**
    - Returns rows with matching values in both tables.
    - Example:

      ```
      SELECT Employees.Name, Departments.DepartmentName
      FROM Employees
      INNER JOIN Departments
      ON Employees.DepartmentID = Departments.DepartmentID;
      ```

2. **Left Join**
    - Returns all rows from the left table and matching rows from the right table. Non-matching rows in the right table are returned as NULL.
    - Example:

      ```
      SELECT Employees.Name, Departments.DepartmentName
      FROM Employees
      LEFT JOIN Departments
      ON Employees.DepartmentID = Departments.DepartmentID;
      ```

3. **Right Join**
    - Returns all rows from the right table and matching rows from the left table. Non-matching rows in the left table are returned as NULL.
    - Example:

      ```
      SELECT Employees.Name, Departments.DepartmentName
      FROM Employees
      RIGHT JOIN Departments
      ON Employees.DepartmentID = Departments.DepartmentID;
      ```

4. **Full Outer Join**
    - Returns all rows when there is a match in either table. Non-matching rows are filled with NULLs.
    - Example:

      ```
      SELECT Employees.Name, Departments.DepartmentName
      FROM Employees
      FULL OUTER JOIN Departments
      ON Employees.DepartmentID = Departments.DepartmentID;
      ```

5. **Cross Join**

- Combines every row of the first table with every row of the second table.
- Example:

```
SELECT Employees.Name, Departments.DepartmentName
FROM Employees
CROSS JOIN Departments;
```

6. **Self Join**

- Joins a table with itself to compare rows within the same table.
- Example:

```
SELECT A.Name AS Employee1, B.Name AS Employee2
FROM Employees A, Employees B
WHERE A.ManagerID = B.EmployeeID;
```