**SQL Examples with Code and Explanation**

**1. Creating a Table**

To start working with a database, you need to create tables to store data. The CREATE TABLE statement is used for this purpose.

**Example: Creating a Table**

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Age INT,

Department VARCHAR(50),

Salary DECIMAL(10, 2)

);

**Explanation:**

* EmployeeID: Integer column to store unique employee IDs. It's the primary key, meaning each value must be unique.
* FirstName and LastName: Text columns for the first and last names of employees.
* Age: Integer column to store the age of the employee.
* Department: Text column to store the department name.
* Salary: Decimal column to store the employee's salary with up to 10 digits, including 2 digits after the decimal point.

**2. Inserting Data into a Table**

The INSERT INTO statement adds new records to a table.

**Example: Inserting Data**

INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department, Salary)

VALUES

(1, 'John', 'Doe', 30, 'IT', 50000.00),

(2, 'Jane', 'Smith', 25, 'HR', 45000.00),

(3, 'Alice', 'Johnson', 40, 'Finance', 60000.00);

**Explanation:**

* This statement inserts three records into the Employees table. Each record corresponds to one employee with values for EmployeeID, FirstName, LastName, Age, Department, and Salary.

**3. Querying Data from a Table**

The SELECT statement retrieves data from a table. You can specify which columns to retrieve and apply conditions.

**Example: Basic SELECT Statement**

SELECT FirstName, LastName, Age, Department FROM Employees;

**Explanation:**

* This retrieves the FirstName, LastName, Age, and Department columns for all employees in the Employees table.

**Example: SELECT with WHERE Clause**

SELECT \* FROM Employees WHERE Age > 30;

**Explanation:**

* The WHERE clause filters the records, returning only employees older than 30 years.

**4. Updating Data in a Table**

The UPDATE statement is used to modify existing records in a table.

**Example: Updating a Record**

UPDATE Employees

SET Salary = 55000.00

WHERE EmployeeID = 1;

**Explanation:**

* This updates the Salary column for the employee with EmployeeID = 1 (John Doe) to 55,000.00.

**5. Deleting Data from a Table**

The DELETE statement removes records from a table.

**Example: Deleting a Record**

DELETE FROM Employees WHERE EmployeeID = 2;

**Explanation:**

* This deletes the record for the employee with EmployeeID = 2 (Jane Smith).

**6. Using JOINs to Combine Data from Multiple Tables**

SQL joins are used to combine rows from two or more tables based on a related column between them.

**Example: Inner Join**

First, let’s create another table for Departments.

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(50)

);

INSERT INTO Departments (DepartmentID, DepartmentName)

VALUES

(1, 'IT'),

(2, 'HR'),

(3, 'Finance');

Now, use an INNER JOIN to combine data from both tables:

SELECT Employees.FirstName, Employees.LastName, Departments.DepartmentName

FROM Employees

INNER JOIN Departments ON Employees.Department = Departments.DepartmentName;

**Explanation:**

* This query retrieves the FirstName, LastName, and DepartmentName by joining the Employees table with the Departments table.
* It matches records based on the department name, linking Employees.Department with Departments.DepartmentName.

**Example: LEFT JOIN**

SELECT Employees.FirstName, Employees.LastName, Departments.DepartmentName

FROM Employees

LEFT JOIN Departments ON Employees.Department = Departments.DepartmentName;

**Explanation:**

* This LEFT JOIN retrieves all employees, even those who don’t have a matching department in the Departments table.
* If an employee doesn't belong to any department, the DepartmentName will be NULL.

**7. Using Aggregation Functions**

SQL provides functions like COUNT, SUM, AVG, MAX, and MIN to perform calculations on data.

**Example: COUNT and GROUP BY**

SELECT Department, COUNT(\*) AS EmployeeCount

FROM Employees

GROUP BY Department;

**Explanation:**

* This query counts the number of employees in each department by grouping records by Department and using COUNT(\*) to get the total number of employees in each group.

**Example: AVG Salary by Department**

SELECT Department, AVG(Salary) AS AverageSalary

FROM Employees

GROUP BY Department;

**Explanation:**

* This query calculates the average salary for each department by grouping employees by Department and using AVG(Salary).

**8. Using WHERE with Aggregate Functions**

SELECT Department, MAX(Salary) AS HighestSalary

FROM Employees

WHERE Age > 30

GROUP BY Department;

**Explanation:**

* This query finds the highest salary for each department, but only considers employees older than 30 years (WHERE Age > 30).

**9. Using SQL Constraints**

Constraints enforce rules on the data in a table. Here’s an example where we use constraints like PRIMARY KEY, NOT NULL, and UNIQUE.

**Example: Creating a Table with Constraints**

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Email VARCHAR(100) UNIQUE,

PhoneNumber VARCHAR(15)

);

**Explanation:**

* CustomerID is the primary key, ensuring that each customer has a unique identifier.
* Name cannot be NULL.
* Email must be unique across the table, preventing duplicate email addresses.

**10. Using Subqueries**

A subquery is a query within another query. It is used to retrieve data that will be used in the main query.

**Example: Subquery in SELECT**

SELECT FirstName, LastName, Salary

FROM Employees

WHERE Salary > (SELECT AVG(Salary) FROM Employees);

**Explanation:**

* The subquery (SELECT AVG(Salary) FROM Employees) calculates the average salary of all employees. The main query then retrieves the FirstName, LastName, and Salary for employees whose salary is greater than this average.