SQL JOINS

Query:

Write an SQL query to retrieve a list of employees with their department names from the "Employees" and "Departments" tables using an INNER JOIN.

Answer:

```
SELECT Employees.EmployeeID, Employees.FirstName, Employees.LastName,
Departments.DepartmentName
FROM Employees
INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;
```

Explain the concept of a self-join and provide an example.

Answer:

```
-- Self-Join Example

SELECT e1.EmployeeID, e1.FirstName, e1.ManagerID, e2.FirstName AS ManagerName

FROM Employees e1

INNER JOIN Employees e2 ON e1.ManagerID = e2.EmployeeID;
```

what is a CROSS JOIN, and when might you use it?

Answer:

A CROSS JOIN returns the Cartesian product of two tables, meaning it combines every row from the first table with every row from the second table. It is used when you want to combine all rows from one table with all rows from another table.

```
-- Cross Join Example

SELECT *

FROM Table1

CROSS JOIN Table2;
```

Pattern matching

Query:

Find all employees whose last name starts with 'A' and ends with 'n'.

Answer:

```sql
SELECT *
FROM Employees
WHERE LastName LIKE 'A%n';
```

Using the "Products" table, find products with names that contain the word "phone" regardless of case.

Answer:

```sql
SELECT *

FROM Products

WHERE LOWER(ProductName) LIKE '%phone%';
```

Explain the usage of the underscore (_) wildcard in SQL pattern matching.

Answer:

The underscore (_) wildcard in SQL is used to match any single character. For example, `LIKE 'a_c'` would match 'abc', 'adc', etc.

views

Query:

Create a view named "EmployeeDetails" that combines information from the "Employees" and "Departments" tables.

Answer:

```sql
CREATE VIEW EmployeeDetails AS
```

```
SELECT Employees.EmployeeID, Employees.FirstName, Employees.LastName,
Departments.DepartmentName
FROM Employees
INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;
```

CTE(common table expression)

Query:

Use a Common Table Expression (CTE) to find the average salary for each department.

Answer:

```
WITH AvgSalaryCTE AS (
 SELECT DepartmentID, AVG(Salary) AS AvgSalary
 FROM Employees
 GROUP BY DepartmentID
)
SELECT Departments.DepartmentName, AvgSalary
FROM AvgSalaryCTE
JOIN Departments ON AvgSalaryCTE.DepartmentID = Departments.DepartmentID;
```

Case expression

Query:

Use a CASE expression to categorize employees into salary ranges.

Answer:

```
SELECT FirstName, LastName, Salary,
 CASE
 WHEN Salary < 50000 THEN 'Low'
 WHEN Salary BETWEEN 50000 AND 80000 THEN 'Medium'
 ELSE 'High'
 END AS SalaryCategory
FROM Employees;
```

Function

Query:

Retrieve the total number of employees and the highest salary using advanced functions.

Answer:

```
SELECT COUNT(*) AS TotalEmployees, MAX(Salary) AS HighestSalary
FROM Employees;
```

Stored procedure

Query:

Create a stored procedure named "GetEmployeeDetails" that takes an EmployeeID as a parameter and returns the details of that employee.

Answer:

```
CREATE PROCEDURE GetEmployeeDetails(@EmployeeID INT)
AS
BEGIN
 SELECT * FROM Employees WHERE EmployeeID = @EmployeeID;
END;
```

Trigger in SQL

Query:

Create a trigger that automatically updates the "LastModified" column in the "Employees" table when a record is updated.

Answer:

```
CREATE TRIGGER UpdateLastModified
ON Employees
AFTER UPDATE
AS
BEGIN
 UPDATE Employees
 SET LastModified = GETDATE()
 FROM Employees
 INNER JOIN INSERTED ON Employees.EmployeeID = INSERTED.EmployeeID;
END;
```