**Case Study 1: Retrieve Duplicate Records**

* **Problem**: Find duplicate rows in the Sales table based on ProductID and SaleDate.
* **Solution**:
* SELECT ProductID, SaleDate, COUNT(\*) AS DuplicateCount
* FROM Sales
* GROUP BY ProductID, SaleDate
* HAVING COUNT(\*) > 1;

**Case Study 2: Update Data Using a Subquery**

* **Problem**: Update the TotalSales column in the Products table based on data from the Sales table.
* **Solution**:
* UPDATE Products
* SET TotalSales = (
* SELECT SUM(Amount)
* FROM Sales
* WHERE Sales.ProductID = Products.ProductID
* );

**Case Study 3: Calculate Running Total**

* **Problem**: Calculate a running total of sales in the Sales table.
* **Solution**:
* SELECT ProductID, SaleDate, Amount,
* SUM(Amount) OVER (PARTITION BY ProductID ORDER BY SaleDate) AS RunningTotal
* FROM Sales;

**Case Study 4: Find the Top 3 Sales per Product**

* **Problem**: Retrieve the top 3 sales transactions for each product based on amount.
* **Solution**:
* SELECT ProductID, SaleID, Amount,
* RANK() OVER (PARTITION BY ProductID ORDER BY Amount DESC) AS Rank
* FROM Sales
* WHERE Rank <= 3;

**Case Study 5: Insert Data from Another Table**

* **Problem**: Insert records into the ArchivedSales table for transactions before 2023.
* **Solution**:
* INSERT INTO ArchivedSales (SaleID, ProductID, Amount, SaleDate)
* SELECT SaleID, ProductID, Amount, SaleDate
* FROM Sales
* WHERE SaleDate < '2023-01-01';

**Case Study 6: Delete Duplicate Rows**

* **Problem**: Remove duplicate rows in the Employees table based on EmployeeName and HireDate.
* **Solution**:
* WITH CTE AS (
* SELECT EmployeeID, ROW\_NUMBER() OVER (PARTITION BY EmployeeName, HireDate ORDER BY EmployeeID) AS RowNum
* FROM Employees
* )
* DELETE FROM Employees
* WHERE EmployeeID IN (SELECT EmployeeID FROM CTE WHERE RowNum > 1);

**Case Study 7: Generate a Pivot Table**

* **Problem**: Create a pivot table for total sales by Region and ProductCategory.
* **Solution**:
* SELECT ProductCategory, [North], [South], [East], [West]
* FROM (
* SELECT Region, ProductCategory, Amount
* FROM Sales
* ) AS SourceTable
* PIVOT (
* SUM(Amount) FOR Region IN ([North], [South], [East], [West])
* ) AS PivotTable;

**Case Study 8: Identify Gaps in Sequences**

* **Problem**: Identify missing InvoiceNumbers in the Invoices table.
* **Solution**:
* SELECT InvoiceNumber + 1 AS MissingInvoice
* FROM Invoices i
* WHERE NOT EXISTS (
* SELECT 1
* FROM Invoices
* WHERE InvoiceNumber = i.InvoiceNumber + 1
* );

**Case Study 9: Rank Employees by Department**

* **Problem**: Rank employees within each department based on their salary.
* **Solution**:
* SELECT EmployeeID, DepartmentID, Salary,
* RANK() OVER (PARTITION BY DepartmentID ORDER BY Salary DESC) AS Rank
* FROM Employees;

**Case Study 10: Merge Data**

* **Problem**: Merge records from the NewEmployees table into the Employees table.
* **Solution**:
* MERGE INTO Employees AS Target
* USING NewEmployees AS Source
* ON Target.EmployeeID = Source.EmployeeID
* WHEN MATCHED THEN
* UPDATE SET Name = Source.Name, DepartmentID = Source.DepartmentID
* WHEN NOT MATCHED THEN
* INSERT (EmployeeID, Name, DepartmentID) VALUES (Source.EmployeeID, Source.Name, Source.DepartmentID);

**Case Study 11: Find Customers with No Orders**

* **Problem**: Identify customers who have not placed any orders.
* **Solution**:
* SELECT CustomerID
* FROM Customers
* WHERE CustomerID NOT IN (SELECT DISTINCT CustomerID FROM Orders);

**Case Study 12: Compute Percentile**

* **Problem**: Calculate the percentile rank of employees' salaries in the company.
* **Solution**:
* SELECT EmployeeID, Salary,
* PERCENT\_RANK() OVER (ORDER BY Salary) AS PercentileRank
* FROM Employees;

**Case Study 13: Calculate Age**

* **Problem**: Calculate the age of employees from their DateOfBirth.
* **Solution**:
* SELECT EmployeeID, Name,
* DATEDIFF(YEAR, DateOfBirth, GETDATE()) AS Age
* FROM Employees;

**Case Study 14: Filter with Window Functions**

* **Problem**: Find employees who earn more than the department average salary.
* **Solution**:
* SELECT EmployeeID, Name, Salary, DepartmentID
* FROM (
* SELECT EmployeeID, Name, Salary, DepartmentID,
* AVG(Salary) OVER (PARTITION BY DepartmentID) AS AvgSalary
* FROM Employees
* ) AS SubQuery
* WHERE Salary > AvgSalary;

**Case Study 15: Recursive Query**

* **Problem**: Find the hierarchy of managers and employees.
* **Solution**:
* WITH EmployeeHierarchy AS (
* SELECT EmployeeID, ManagerID, Name, 0 AS Level
* FROM Employees
* WHERE ManagerID IS NULL
* UNION ALL
* SELECT e.EmployeeID, e.ManagerID, e.Name, eh.Level + 1
* FROM Employees e
* INNER JOIN EmployeeHierarchy eh ON e.ManagerID = eh.EmployeeID
* )
* SELECT \* FROM EmployeeHierarchy;

**Case Study 16: Generate a Monthly Report**

* **Problem**: Calculate monthly sales totals for the past year.
* **Solution**:
* SELECT YEAR(SaleDate) AS Year, MONTH(SaleDate) AS Month, SUM(Amount) AS TotalSales
* FROM Sales
* WHERE SaleDate >= DATEADD(YEAR, -1, GETDATE())
* GROUP BY YEAR(SaleDate), MONTH(SaleDate)
* ORDER BY Year, Month;

**Case Study 17: Dynamic SQL Query**

* **Problem**: Create a dynamic query to filter data based on user input.
* **Solution**:
* DECLARE @Region NVARCHAR(50) = 'North';
* DECLARE @Query NVARCHAR(MAX);
* SET @Query = N'SELECT \* FROM Sales WHERE Region = @Region';
* EXEC sp\_executesql @Query, N'@Region NVARCHAR(50)', @Region;

**Case Study 18: Validate Data Integrity**

* **Problem**: Check if all records in the Orders table have a valid CustomerID in the Customers table.
* **Solution**:
* SELECT o.OrderID
* FROM Orders o
* LEFT JOIN Customers c ON o.CustomerID = c.CustomerID
* WHERE c.CustomerID IS NULL;

**Case Study 19: Perform a Full Join**

* **Problem**: Perform a full outer join between TableA and TableB.
* **Solution**:
* SELECT \*
* FROM TableA
* FULL OUTER JOIN TableB ON TableA.ID = TableB.ID;

**Case Study 20: Partitioned Table Maintenance**

* **Problem**: Archive old data from a partitioned table.
* **Solution**:
* ALTER PARTITION FUNCTION pf\_MyPartitionFunction()
* MERGE RANGE ('2023-01-01');