SQL Implementation for Table & View Operations

Step 1: Create Necessary Base Tables

1. Create Tables

Customer Table

```
CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY,
    CustomerFirstName VARCHAR(50),
    CustomerLastName VARCHAR(50),
    CustomerPhoneNumber VARCHAR(15)
);
```

Inventory Table

```
CREATE TABLE Inventory (
    InventoryID INT PRIMARY KEY,
    InventoryName VARCHAR(50),
    UnitPrice DECIMAL(10, 2)
);
```

Sale Table

```
CREATE TABLE Sale (
    SaleID INT PRIMARY KEY,
    CustomerID INT,
    InventoryID INT,
    SaleDate DATE,
    SaleQuantity INT,
    SaleUnitPrice DECIMAL(10, 2),
    FOREIGN KEY (CustomerID) REFERENCES
Customer(CustomerID),
    FOREIGN KEY (InventoryID) REFERENCES
Inventory(InventoryID)
);
```

---

2. Insert Data

Insert Data into Customer

```
INSERT INTO Customer VALUES
(1, 'John', 'Doe', '1234567890'),
```

```
(2, 'Jane', 'Smith', '9876543210'),
(3, 'Alice', 'Brown', '4561237890');
```

Insert Data into Inventory

```
INSERT INTO Inventory VALUES
(1, 'Laptop', 1000.00),
(2, 'Phone', 700.00),
(3, 'Tablet', 500.00);
```

Insert Data into Sale

```
INSERT INTO Sale VALUES
(1, 1, 1, '2024-01-15', 2, 1000.00),
(2, 2, 2, '2024-01-20', 1, 700.00),
(3, 3, 3, '2024-02-01', 3, 500.00);
```

Step 2: Perform Queries

---

1. View Table

A **View** is a virtual table that dynamically reflects changes made to the base tables.

```
CREATE VIEW CustomerView AS
SELECT
    CustomerFirstName + ' ' + CustomerLastName AS
[Customer Name],
    CustomerPhoneNumber,
    InventoryName,
    SaleDate,
    SaleQuantity,
    SaleUnitPrice,
    SaleQuantity * SaleUnitPrice AS [Total Amount]
FROM Customer
INNER JOIN Sale ON Customer.CustomerID =
Sale.CustomerID
INNER JOIN Inventory ON Sale.InventoryID =
Inventory.InventoryID;
```

Query the View:

```
SELECT * FROM CustomerView;
```

Output:

Customer Name	CustomerPhone Number	Inventory Name	Sale Date	SaleQuantity	SaleUnit Price	Total Amount
John Doe	1234567890	Laptop	2024-01-15	2	1000.00	2000.00
Jane Smith	9876543210	Phone	2024-01-20	1	700.00	700.00
Alice Brown	4561237890	Tablet	2024-02-01	3	500.00	1500.00

---

2. Temporary Table

A **Temp Table** is used to store data temporarily during the session.

```
DROP TABLE IF EXISTS #Temp_Employee;
```

```
CREATE TABLE #Temp_Employee (
    JobTitle VARCHAR(100),
    EmployeesPerJob INT,
    AvgAge INT,
    AvgSalary DECIMAL(10, 2)
);

INSERT INTO #Temp_Employee
SELECT
    JobTitle,
    COUNT(JobTitle),
    AVG(Age),
    AVG(Salary)
FROM EmployeeDemographics ED
JOIN EmployeeSalary ES ON ED.EmployeeID = ES.EmployeeID
GROUP BY JobTitle;

SELECT * FROM #Temp_Employee;
```

3. Common Table Expression (CTE)

A **CTE** is used for creating temporary result sets in memory for complex queries.

```
WITH CTE_Employee AS
(
    SELECT
        FirstName,
        LastName,
        Gender,
        Salary,
        COUNT(Gender) OVER (PARTITION BY Gender) AS
TotalGender
    FROM EmployeeDemographics ED
    JOIN EmployeeSalary ES ON ED.EmployeeID =
ES.EmployeeID
    WHERE Salary > 45000
)
SELECT FirstName, LastName, Gender, TotalGender
FROM CTE_Employee
```

```
WHERE TotalGender = (SELECT MIN(TotalGender) FROM
CTE_Employee);
```

---

4. Duplicate Table

The `SELECT INTO` statement creates a new table and copies the result of a query into it.

```
SELECT
    CustomerFirstName + ' ' + CustomerLastName AS
[Customer Name],
    CustomerPhoneNumber,
    InventoryName,
    SaleDate,
    SaleQuantity,
    SaleUnitPrice,
    SaleQuantity * SaleUnitPrice AS [Total Amount]
INTO CustomerRec
FROM Customer
INNER JOIN Sale ON Customer.CustomerID =
Sale.CustomerID
```

```
INNER JOIN Inventory ON Sale.InventoryID =
Inventory.InventoryID

ORDER BY CustomerFirstName + ' ' + CustomerLastName,
InventoryName;
```

Query the Duplicate Table:

```
SELECT * FROM CustomerRec;
```

Output:

Customer Name	CustomerPhone Number	Inventory Name	Sale Date	SaleQuantity	SaleUnit Price	Total Amount
John Doe	1234567890	Laptop	2024-01-15	2	1000.00	2000.00
Jane Smith	9876543210	Phone	2024-01-20	1	700.00	700.00
Alice Brown	4561237890	Tablet	2024-02-01	3	500.00	1500.00