**Part 1: Research & Documentation**

**1. Types of Views in SQL Server**

**1.1 Standard View**

* **Definition**: A virtual table based on the result of a SELECT query. It does not store data physically.
* **Differences**: Cannot be indexed unless certain conditions are met. Mostly read-only.
* **Use Case**: In a banking system, a standard view can show combined data from Customer and Account for teller screens.
* **Limitations**: Performance can degrade for complex joins; cannot have ORDER BY without TOP.

**1.2 Indexed View (Materialized View)**

* **Definition**: A view that has a unique clustered index. Data is stored physically.
* **Differences**: Unlike standard views, indexed views improve performance but consume storage.
* **Use Case**: In e-commerce, tracking total sales per product efficiently.
* **Limitations**: Strict rules (e.g., deterministic functions, schema binding); more costly to maintain.

**1.3 Partitioned View (Union View)**

* **Definition**: A view that combines multiple tables using UNION ALL to simulate table partitioning.
* **Differences**: Unlike others, it spans across tables (even across databases in distributed views).
* **Use Case**: A university storing students per campus in separate tables but querying them as one.
* **Limitations**: INSERT/UPDATE requires CHECK constraints and identical schema.

**2. DML Operations on Views**

* **Allowed DML**: INSERT, UPDATE, DELETE are allowed on standard and indexed views (with restrictions).
* **Restrictions**:
  + The view must refer to one base table (for INSERT).
  + No aggregate, group by, distinct, joins (in many cases).
* **Useful Scenario**: HR system with a view to update employee contact info, hiding salary.

**Example 1: View for Customer + Account (Used by Call Center Agents)**

sql

CopyEdit

CREATE VIEW CustomerAccountSummary AS

SELECT

c.CustomerID,

c.FullName,

c.Phone,

a.AccountID,

a.AccountType,

a.Status,

a.Balance

FROM Customer c

JOIN Account a ON c.CustomerID = a.CustomerID;

**Usage Without View:**

sql

CopyEdit

SELECT

c.FullName, c.Phone, a.AccountType, a.Balance

FROM Customer c

JOIN Account a ON c.CustomerID = a.CustomerID

WHERE a.Status = 'Active';

**Usage With View:**

sql

CopyEdit

SELECT

FullName, Phone, AccountType, Balance

FROM CustomerAccountSummary

WHERE Status = 'Active';

**📌 Example 2: View for Account + Transaction (Used by Finance Auditors)**

sql

CopyEdit

CREATE VIEW AccountTransactionView AS

SELECT

a.AccountID,

a.AccountType,

t.TransactionID,

t.Amount,

t.Type AS TransactionType,

t.TransactionDate

FROM Account a

JOIN [Transaction] t ON a.AccountID = t.AccountID;

**Usage Without View:**

sql

CopyEdit

SELECT

a.AccountID, a.AccountType, t.Amount, t.TransactionDate

FROM Account a

JOIN [Transaction] t ON a.AccountID = t.AccountID

WHERE t.TransactionDate >= DATEADD(DAY, -7, GETDATE());

**Usage With View:**

sql

CopyEdit

SELECT

AccountID, AccountType, Amount, TransactionDate

FROM AccountTransactionView

WHERE TransactionDate >= DATEADD(DAY, -7, GETDATE());

**- Scenario Example – Call Center Agent:**

Call center agents regularly need to check:

* Customer's name and phone number
* Account type and status
* Current balance

Instead of writing complex JOIN queries, they simply query the view:

sql

CopyEdit

SELECT FullName, Phone, AccountType, Status, Balance

FROM CustomerAccountSummary

WHERE Status = 'Active' AND Phone LIKE '%9722';