📘 Module 25: Views and Indexes

🔷 1. VIEW

✅ What is a View?

A **view** is a **virtual table** that is based on the result of a SQL query.

It **does not store data** itself but presents data from one or more tables.

Views are used to:

- Simplify complex queries
- Improve data security (show limited data)
- Help standardise reporting

✅ Syntax:

1. CREATE VIEW view_name AS
   SELECT column1, column2 FROM table
   WHERE condition;
2. DROP VIEW view_name;
3. ALTER VIEW view_name AS
   SELECT ...

📌 Example:

CREATE VIEW CustomerOrders AS
SELECT c.CustomerName, o.OrderDate, o.Amount FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID;

✔️ Now you can run:

SELECT * FROM CustomerOrders;

```
CREATE VIEW CustomerOrders AS
SELECT c.CustomerName, o.OrderDate, o.Amount
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID;

SELECT * FROM CustomerOrders;
```

100 %

Results | Messages

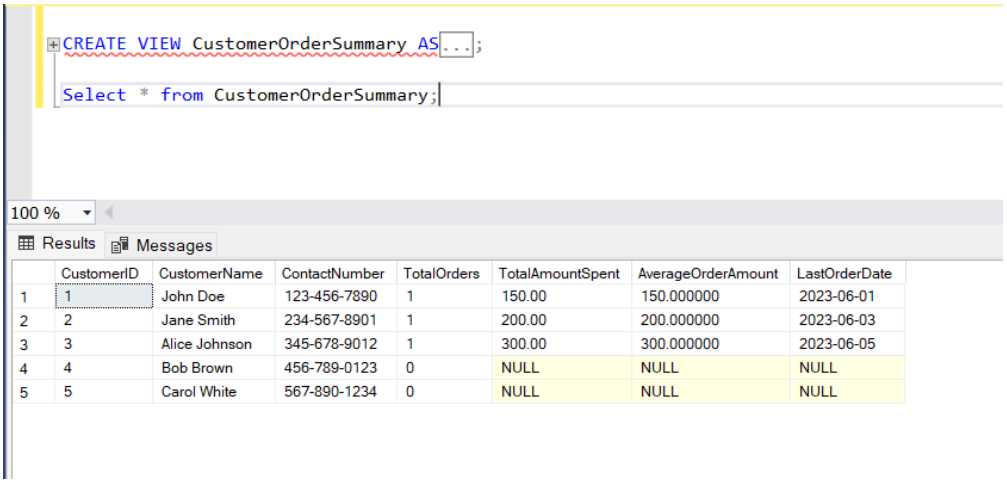
	CustomerName	OrderDate	Amount
1	John Doe	2023-06-01	150.00
2	Jane Smith	2023-06-03	200.00
3	Alice Johnson	2023-06-05	300.00

💡 Example:

```
CREATE VIEW CustomerOrderSummary AS
SELECT c.CustomerID, c.CustomerName, c.ContactNumber,

-- Total number of orders per customer
(SELECT COUNT(*) FROM Orders o
 WHERE o.CustomerID = c.CustomerID) AS TotalOrders,

-- Total amount spent by the customer
(SELECT SUM(o.Amount) FROM Orders o
 WHERE o.CustomerID = c.CustomerID) AS TotalAmountSpent,

-- Average amount per order by the customer
(SELECT AVG(o.Amount) FROM Orders o
 WHERE o.CustomerID = c.CustomerID) AS AverageOrderAmount,

-- Date of the latest order by the customer
(SELECT MAX(o.OrderDate) FROM Orders o
 WHERE o.CustomerID = c.CustomerID) AS LastOrderDate

FROM Customers c;
```

✔️ Now you can run:

```
SELECT * FROM CustomerOrderSummary;
```



	CustomerID	CustomerName	ContactNumber	TotalOrders	TotalAmountSpent	AverageOrderAmount	LastOrderDate
1	1	John Doe	123-456-7890	1	150.00	150.000000	2023-06-01
2	2	Jane Smith	234-567-8901	1	200.00	200.000000	2023-06-03
3	3	Alice Johnson	345-678-9012	1	300.00	300.000000	2023-06-05
4	4	Bob Brown	456-789-0123	0	NULL	NULL	NULL
5	5	Carol White	567-890-1234	0	NULL	NULL	NULL

🔁 Can a View Be Updated?

✔️ **Yes**, if:

- It refers to only **one base table**
- It has **no aggregates**, **joins**, `DISTINCT`, `GROUP BY`, etc.

❌ **No**, if:

- It includes multiple tables, joins, calculations, etc.

📝 Important Points to Remember About Views:

- Views are not physical tables – they store SQL logic.
- Data shown in a view is always the **latest** from the base tables.
- You can use views in `SELECT`, `JOIN`, and even in other views.
- Use `WITH SCHEMABINDING` to **lock the structure** if needed.

🔷 2. INDEX

✅ What is an Index?

An **index** in SQL Server is like an **index in a book** — it helps **speed up searches** and queries on large tables.

Indexes are created on columns to make data retrieval **faster**, especially for:

- `WHERE` filters
- `ORDER BY` clauses
- `JOIN` conditions

✅ Syntax:

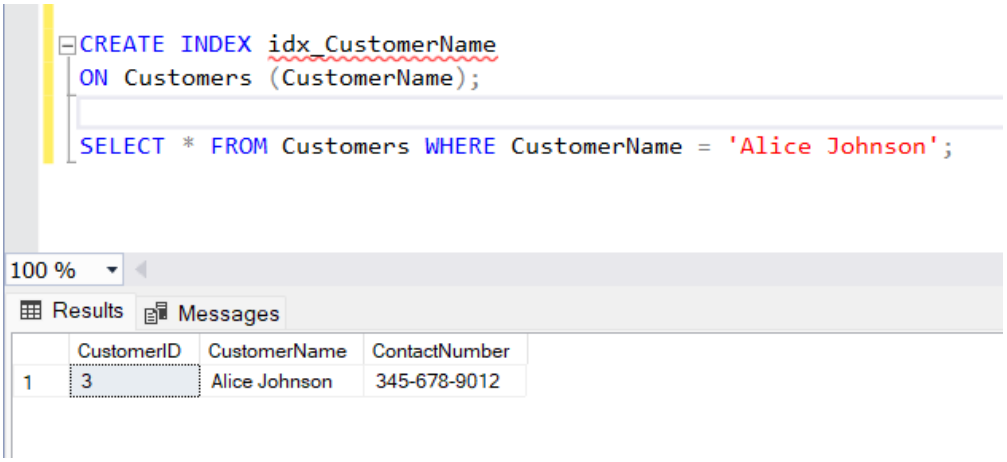
1. CREATE INDEX index_name
   ON table_name (column1, column2);
2. DROP INDEX index_name ON table_name;
3. Alter
   - - Rebuild (fix fragmentation)
     ALTER INDEX index_name ON table_name REBUILD;
   - - Reorganise (light maintenance)
     ALTER INDEX index_name ON table_name REORGANIZE;

📌 Example:

CREATE INDEX idx_CustomerName
ON Customers (CustomerName);

✔️ This will speed up queries like:

SELECT * FROM Customers WHERE CustomerName = 'Alice Johnson';



🔍 Types of Indexes

Type	Description
Clustered	Reorders the **actual data rows** — only one per table
Non-Clustered	A **pointer** index — stores sorted pointers to the actual data rows
Unique Index	Ensures that all values in the indexed column are unique
Composite Index	Indexes **multiple columns together**
Full-text Index	Used for fast searching in large text columns (e.g., articles, descriptions)

✅ Good Practices for Indexing

✔️ Create indexes on columns used in:

- `WHERE`
- `JOIN`
- `ORDER BY`
- `GROUP BY`

✔️ Don't over-index — too many indexes slow down **INSERT**, **UPDATE**, **DELETE**

✔️ Always **name your indexes** clearly (`idx_tablename_columnname`)

✔️ Use the **Composite Index** if you frequently filter by multiple columns

✔️ Analyse performance using SQL Server's **Execution Plan**

✅ Summary

Concept	Purpose	Key Commands
VIEW	Virtual table for simplifying data	`CREATE VIEW`, `SELECT FROM view`
INDEX	Speed up search performance	`CREATE INDEX`, `DROP INDEX`