

1 Run the setup script from the [companion blog post](#) first. Every block then runs as written, in order.

2 Each block says what it does: READ ONLY CHANGES DATA CHANGES OBJECTS CHANGES SERVER

1. Query it Reading data: joins, windows, CTEs, grouping and pivots

Logical query processing order

1 FROM / JOIN / APPLY 2 WHERE 3 GROUP BY 4 HAVING 5 SELECT 6 DISTINCT
7 ORDER BY 8 OFFSET / FETCH

That is why a column alias works in ORDER BY but not in WHERE. The plan can run steps in another physical order.

Paging and TOP

```
SELECT OrderID, OrderDate FROM dbo.Orders
ORDER BY OrderDate DESC, OrderID
OFFSET 10 ROWS FETCH NEXT 10 ROWS ONLY; -- page 2
```

READ ONLY

```
SELECT TOP (6) WITH TIES OrderID, Amount
FROM dbo.Orders ORDER BY Amount DESC; -- 6 plus ties at 300
```

READ ONLY

TOP without ORDER BY returns any rows, not the first ones.

Joins, APPLY and EXISTS

INNER JOIN	only matching rows
LEFT JOIN	all left rows, NULLs where no match
FULL JOIN	all rows from both sides
CROSS JOIN	every row times every row
CROSS APPLY	run a subquery per row (inner)
OUTER APPLY	same, keeps rows with no result

Latest 2 orders per customer

```
SELECT c.Name, x.OrderID, x.OrderDate
FROM dbo.Customers AS c
CROSS APPLY (SELECT TOP (2) o.OrderID, o.OrderDate
             FROM dbo.Orders AS o
             WHERE o.CustomerID = c.CustomerID
             ORDER BY o.OrderDate DESC, o.OrderID DESC) AS x;
```

READ ONLY

Joins, APPLY and EXISTS (continued)

Customers with no orders (NULL safe, unlike NOT IN)

```
SELECT c.CustomerID, c.Name FROM dbo.Customers AS c
WHERE NOT EXISTS (SELECT 1 FROM dbo.Orders AS o
                  WHERE o.CustomerID = c.CustomerID);
```

READ ONLY

Window functions

```
SELECT OrderID, CustomerID, Amount,
       ROW_NUMBER() OVER (PARTITION BY CustomerID
                          ORDER BY OrderDate DESC, OrderID DESC) AS RowNum,
       SUM(Amount) OVER (PARTITION BY CustomerID
                         ORDER BY OrderDate, OrderID
                         ROWS UNBOUNDED PRECEDING) AS RunningTotal,
       LAG(Amount) OVER (PARTITION BY CustomerID
                        ORDER BY OrderDate, OrderID) AS PrevAmount,
       Amount * 100.0 / SUM(Amount) OVER () AS PctOfAll
FROM dbo.Orders;
```

READ ONLY

Add a unique tie-breaker such as OrderID whenever row order matters. Ranking keeps ties on purpose:

Amount	ROW_NUMBER	RANK	DENSE_RANK	NTILE(2)
300	1	1	1	1
200	2	2	2	1
200	3	2	2	2
100	4	4	3	2

Delete duplicates, keep one

```
WITH d AS (
  SELECT *, ROW_NUMBER() OVER (PARTITION BY Email
                              ORDER BY CustomerID) AS rn
  FROM dbo.Customers)
DELETE FROM d WHERE rn > 1;
```

CHANGES DATA

CASE, NULL and safe math

READ ONLY

```
SELECT OrderID,
CASE WHEN Amount >= 500 THEN 'A'
      WHEN Amount >= 100 THEN 'B' ELSE 'C' END AS Tier,
IIF(Status = 'Open', 1, 0) AS IsOpen,
Amount / NULLIF(Qty, 0) AS UnitPrice,      -- no divide by 0
COALESCE(Amount, 0) AS AmountOrZero
FROM dbo.Orders;
```

Compare values that can be NULL **2022**

READ ONLY

```
SELECT c.CustomerID FROM dbo.Customers AS c
WHERE c.Phone IS DISTINCT FROM '555-0100'; -- NULL counts
```

Group and pivot

READ ONLY

```
SELECT CustomerID,
SUM(CASE WHEN YEAR(OrderDate) = 2024 THEN Amount END) AS Y24,
SUM(CASE WHEN YEAR(OrderDate) = 2025 THEN Amount END) AS Y25
FROM dbo.Orders GROUP BY CustomerID;
```

READ ONLY

```
SELECT * FROM (SELECT CustomerID, Status, Amount
                FROM dbo.Orders) AS s
PIVOT (SUM(Amount) FOR Status
      IN ([Open], [Shipped], [Closed])) AS p;
```

CTE and recursive CTE

READ ONLY

```
WITH Org AS (
SELECT EmployeeID, ManagerID, Name, 0 AS Lvl
FROM dbo.Employees WHERE ManagerID IS NULL -- anchor
UNION ALL
SELECT e.EmployeeID, e.ManagerID, e.Name, o.Lvl + 1
FROM dbo.Employees AS e
JOIN Org AS o ON e.ManagerID = o.EmployeeID -- recursion
SELECT * FROM Org
OPTION (MAXRECURSION 100); -- default 100, 0 = no limit
```

Habits that save you

- Avoid functions on a filtered column. A range lets an index help.
- NOT EXISTS is safer than NOT IN when NULLs are possible.
- SELECT only the columns you need. Skip SELECT *.
- BEGIN TRANSACTION before a risky change, check the row count, then COMMIT.
- SET STATISTICS IO, TIME ON, then compare reads and time before and after a change.
- Test the restore, not only the backup.

2. Shape it Strings, regex, numbers, dates and JSON

Strings

READ ONLY

```
SELECT CONCAT_WS(' ', Name, City, Phone) AS Line, -- no NULLs
Name || ' <' || Email || '>' AS NameEmail, 2025
SUBSTRING(Email, CHARINDEX('@', Email) + 1)
AS Domain 2025
FROM dbo.Customers;
```

READ ONLY

```
SELECT CustomerID,
STRING_AGG(CAST(OrderID AS varchar(10)), ',')
WITHIN GROUP (ORDER BY OrderID) AS OrderList
FROM dbo.Orders GROUP BY CustomerID;
```

READ ONLY

```
SELECT value, ordinal
FROM STRING_SPLIT('red,green,blue', ',', 1) 2022
ORDER BY ordinal;
```

Regular expressions **2025**

READ ONLY

```
SELECT Email FROM dbo.Customers
WHERE REGEXP_LIKE(Email, '^[@\s]+@[@\s]+\.[a-z]{2,}$', 'i');
```

READ ONLY

```
SELECT REGEXP_REPLACE(Phone, '[^0-9]', '') AS DigitsOnly,
REGEXP_SUBSTR(Email, '@(.+)$', 1, 1, '', 1) AS Domain
FROM dbo.Customers;
```

Also REGEXP_COUNT, REGEXP_INSTR and REGEXP_SPLIT_TO_TABLE. REGEXP_LIKE needs compatibility level 170.

Numbers and safe conversion

```
SELECT GREATEST(10, 42, 7) AS Biggest,
       LEAST(10, 42, 7) AS Smallest,
       TRY_CAST('12x' AS int) AS BadToNull,
       TRY_CONVERT(date, '2025-02-30') AS NoSuchDay;
```

READ ONLY

```
SELECT PRODUCT(1 + Rate) - 1 AS Compounded
FROM (VALUES (0.05), (0.10), (0.02)) AS r (Rate);
```

READ ONLY

Dates and times

```
SELECT SYSDATETIME() AS Now2,
       CURRENT_DATE AS Today,
       DATEADD(day, -7, CURRENT_DATE) AS WeekAgo,
       DATEDIFF(day, '2025-01-01', '2025-12-31') AS Days,
       EOMONTH(CURRENT_DATE) AS MonthEnd,
       DATETRUNC(month, SYSDATETIME()) AS MonthStart,
       DATE_BUCKET(week, 1, CURRENT_DATE) AS Wk,
       SYSDATETIMEOFFSET()
       AT TIME ZONE 'Eastern Standard Time' AS Eastern;
```

READ ONLY

Index friendly date range (no function on the column)

```
SELECT OrderID FROM dbo.Orders
WHERE OrderDate >= '2025-01-01' AND OrderDate < '2026-01-01';
```

READ ONLY

CONVERT(varchar, @d, style)

Style	Result
23	2025-03-07
101	03/07/2025
103	07/03/2025
112	20250307
120	2025-03-07 14:05:09

JSON

```
CREATE TABLE dbo.Events (
    EventID int IDENTITY PRIMARY KEY,
    Payload json NOT NULL);
INSERT dbo.Events (Payload)
VALUES ('{"user":"pinal","tags":["sql","ai"],"score":9}');
```

CHANGES OBJECTS

JSON (continued)

```
SELECT JSON_VALUE(Payload, '$.user') AS UserName,
       JSON_QUERY(Payload, '$.tags') AS Tags
FROM dbo.Events;
```

READ ONLY

```
SELECT e.EventID, t.value AS Tag
FROM dbo.Events AS e
CROSS APPLY OPENJSON(e.Payload, '$.tags') AS t;
```

READ ONLY

```
SELECT CustomerID, Name FROM dbo.Customers FOR JSON PATH;
```

READ ONLY

3. Change it Safe data changes, tables, procedures, transactions and indexes

Tables and constraints

```
CREATE TABLE dbo.Products (
    ProductID int IDENTITY(1,1)
    CONSTRAINT PK_Products PRIMARY KEY,
    Sku varchar(20) NOT NULL
    CONSTRAINT UQ_Products_Sku UNIQUE,
    Price decimal(10,2) NOT NULL
    CONSTRAINT CK_Products_Price CHECK (Price >= 0),
    CreatedAt datetime2(0) NOT NULL
    CONSTRAINT DF_Products_CreatedAt DEFAULT SYSDATETIME(),
    CategoryID int NULL
    CONSTRAINT FK_Products_Categories
    REFERENCES dbo.Categories (CategoryID));
ALTER TABLE dbo.Products ADD Notes nvarchar(400) NULL;
```

CHANGES OBJECTS

Stored procedure

```
CREATE OR ALTER PROCEDURE dbo.GetCustomerOrders
    @CustomerID int
AS
BEGIN
    SET NOCOUNT ON;
    SELECT OrderID, OrderDate, Amount
    FROM dbo.Orders WHERE CustomerID = @CustomerID;
END;
GO
EXEC dbo.GetCustomerOrders @CustomerID = 1;
```

CHANGES OBJECTS

Insert, update, delete

```
INSERT dbo.Categories (CategoryID, Name)
VALUES (10, N'Books'), (11, N'Courses'); -- many rows
```

CHANGES DATA

```
UPDATE o SET o.Status = 'Shipped' -- update with join
FROM dbo.Orders AS o
JOIN dbo.Customers AS c ON c.CustomerID = o.CustomerID
WHERE c.City = N'Austin' AND o.Status = 'Open';
```

CHANGES DATA

```
UPDATE dbo.Orders SET Status = 'Closed' -- see what changed
OUTPUT deleted.Status AS OldStatus,
        inserted.Status AS NewStatus, inserted.OrderID
WHERE OrderDate < '2024-01-01' AND Status = 'Shipped';
```

CHANGES DATA

Transactions, errors, dynamic SQL

```
SET XACT_ABORT ON; -- runtime errors roll back
BEGIN TRY
    BEGIN TRANSACTION;
    UPDATE dbo.Accounts SET Balance -= 100 WHERE AccountID = 1;
    UPDATE dbo.Accounts SET Balance += 100 WHERE AccountID = 2;
    COMMIT;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0 ROLLBACK;
    THROW; -- re-raise to the caller
END CATCH;
```

CHANGES DATA

Raise your own: THROW 50001, 'Order not found.', 1;

```
DECLARE @sql nvarchar(max) = N'SELECT OrderID FROM dbo.Orders
WHERE CustomerID = @cid';
EXEC sys.sp_executesql @sql, N'@cid int', @cid = 1;
```

READ ONLY

Indexes

```
CREATE NONCLUSTERED INDEX IX_Orders_Customer_Date
ON dbo.Orders (CustomerID, OrderDate DESC) -- equality first
INCLUDE (Amount) -- covers the SELECT
WITH (ONLINE = ON, DATA_COMPRESSION = PAGE);
```

CHANGES OBJECTS

Indexes (continued)

```
CREATE INDEX IX_Orders_Open ON dbo.Orders (OrderDate)
WHERE Status = 'Open'; -- filtered
```

CHANGES OBJECTS

```
ALTER INDEX IX_Orders_Customer_Date ON dbo.Orders
REBUILD WITH (ONLINE = ON, RESUMABLE = ON);
ALTER INDEX ALL ON dbo.Orders REORGANIZE;
DROP INDEX IF EXISTS IX_Orders_Open ON dbo.Orders;
```

CHANGES OBJECTS

ONLINE and RESUMABLE need Enterprise, Enterprise Developer or Evaluation edition. On Standard, leave them out.

4. Run it Backups, CHECKDB and SSMS shortcuts

Backup and CHECKDB

```
ALTER DATABASE CheatDemo SET RECOVERY FULL; -- for Log backups
BACKUP DATABASE CheatDemo TO DISK = 'CheatDemo_Full.bak'
WITH COMPRESSION (ALGORITHM = ZSTD), CHECKSUM;
BACKUP LOG CheatDemo TO DISK = 'CheatDemo_Log.trn'
WITH COMPRESSION, CHECKSUM;
```

CHANGES SERVER

```
RESTORE VERIFYONLY FROM DISK = 'CheatDemo_Full.bak'
WITH CHECKSUM;
DBCC CHECKDB (CheatDemo) WITH NO_INFOMSGS, ALL_ERRORMSG;
```

READ ONLY

A file name without a folder goes to the server's default backup folder.

SSMS shortcuts

F5	Run (selection or all)	Alt+Break	Cancel the query
Ctrl+L	Estimated plan	Ctrl+M	Include actual plan
Ctrl+R	Hide or show results	F6	Editor and results
Ctrl+K, Ctrl+C	Comment	Ctrl+K, Ctrl+U	Uncomment
Ctrl+Shift+U / L	Upper / lower case	Ctrl+Shift+R	Refresh IntelliSense
Alt+F1	sp_help on a name	Shift+Alt+Arrows	Column select

Default SQL Server keyboard scheme. Source: Microsoft Learn, SSMS keyboard shortcuts (30 Oct 2025).