NULL FUNCTIONS:

ISNULL ():

Replaces NULL with specified values.

COALESCE ():

Returns the first non-null value from the list.

ISNULL	COALESCE
Limited to two values	Unlimited
Fast	Slow
SQL Server → IsNULL Oracle → NVL MySQL → IFNULL	Available in All Databases

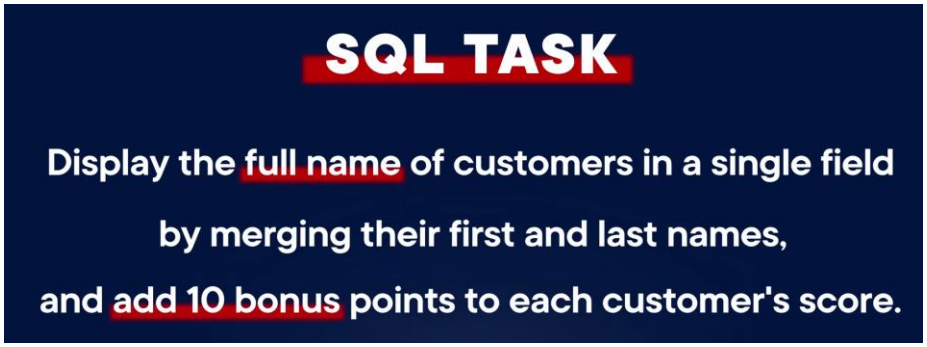
USE CASE – HANDLING NULS

DATA AGGREGATION:

```sql
SELECT
CustomerID,
Score,
AVG(Score) OVER() Wrong_Avg_Score,
--AVG function in SQL ignores NULL value, The sum will be divided by 4 instaed of 5
AVG(COALESCE(Score,0)) OVER() Correct_Avg_Score
FROM Sales.Customers
```

	CustomerID	Score	Wrong_Avg_Score	Correct_Avg_Score
1	1	350	625	500
2	2	900	625	500
3	3	750	625	500
4	4	500	625	500
5	5	NULL	625	500

MATHEMATICAL OPERATIONS:

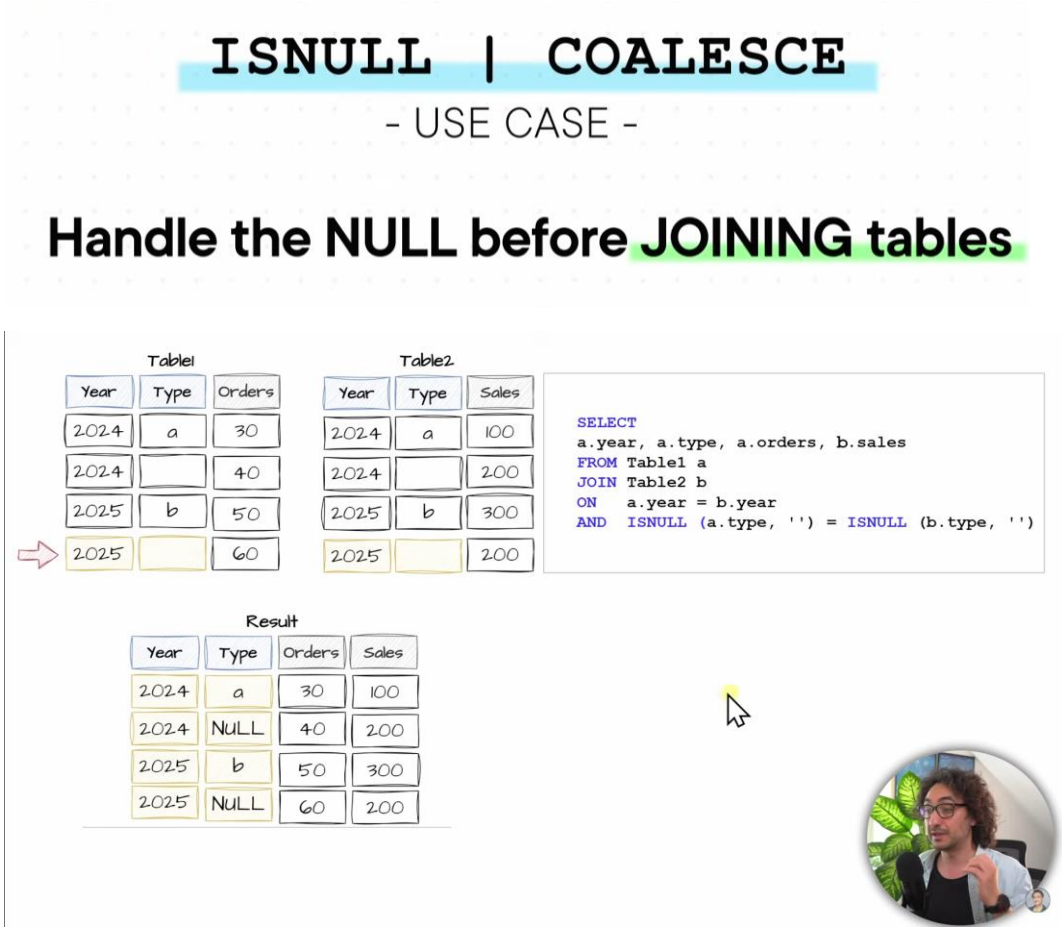



```sql
SELECT
CustomerID,
FirstName,
LastName,
FirstName + ' ' + COALESCE(LastName,'') AS Full_Name,
COALESCE(Score,0) + 10 AS Bonus_Score
FROM Sales.Customers
```

	CustomerID	FirstName	LastName	Full_Name	Bonus_Score
1	1	Jossef	Goldberg	Jossef Goldberg	360
2	2	Kevin	Brown	Kevin Brown	910
3	3	Mary	NULL	Mary	760
4	4	Mark	Schwarz	Mark Schwarz	510
5	5	Anna	Adams	Anna Adams	10

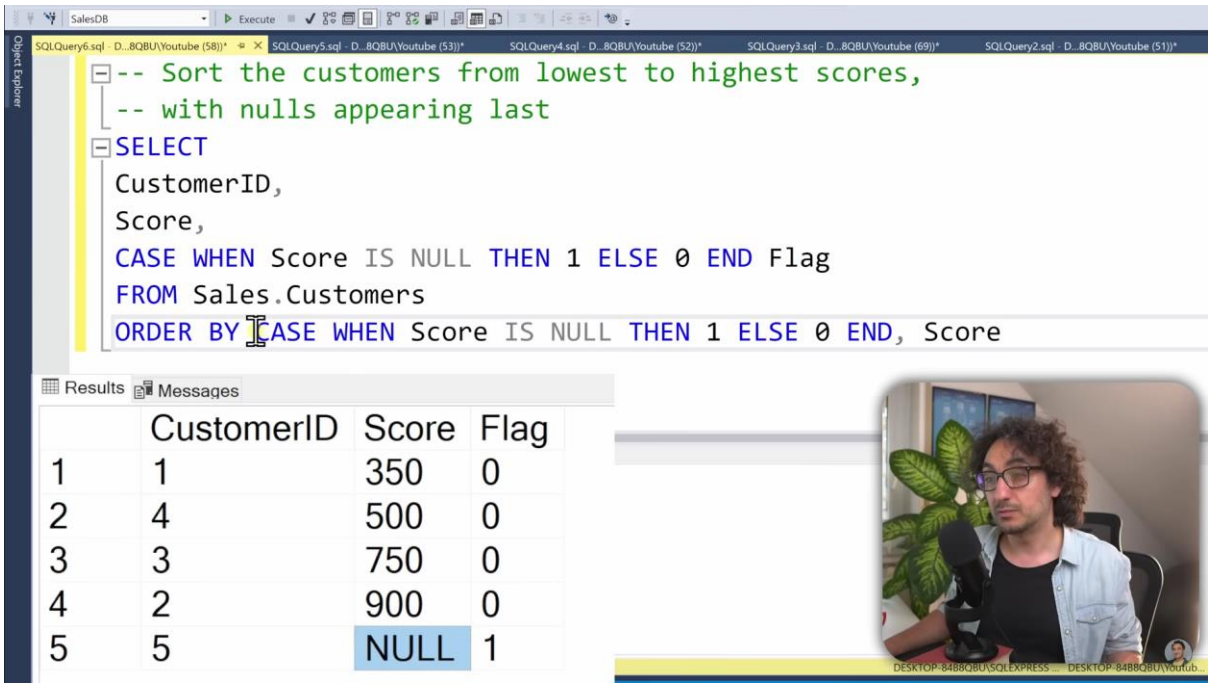
HANDLING NULLS

JOINS:



HANDLING NULLS:

SORTING DATA

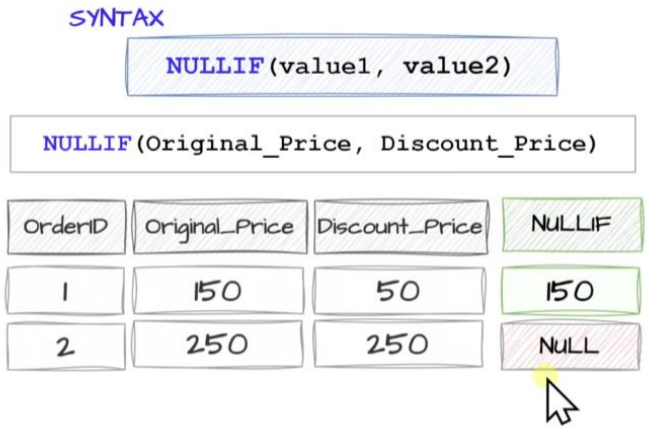


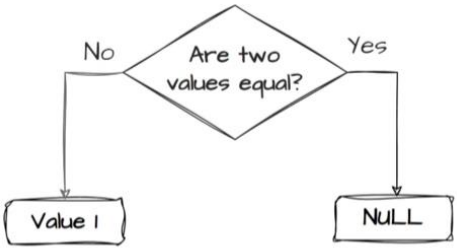
NULLIF ():

Compares two expressions returns:

-NULL if they are equal.

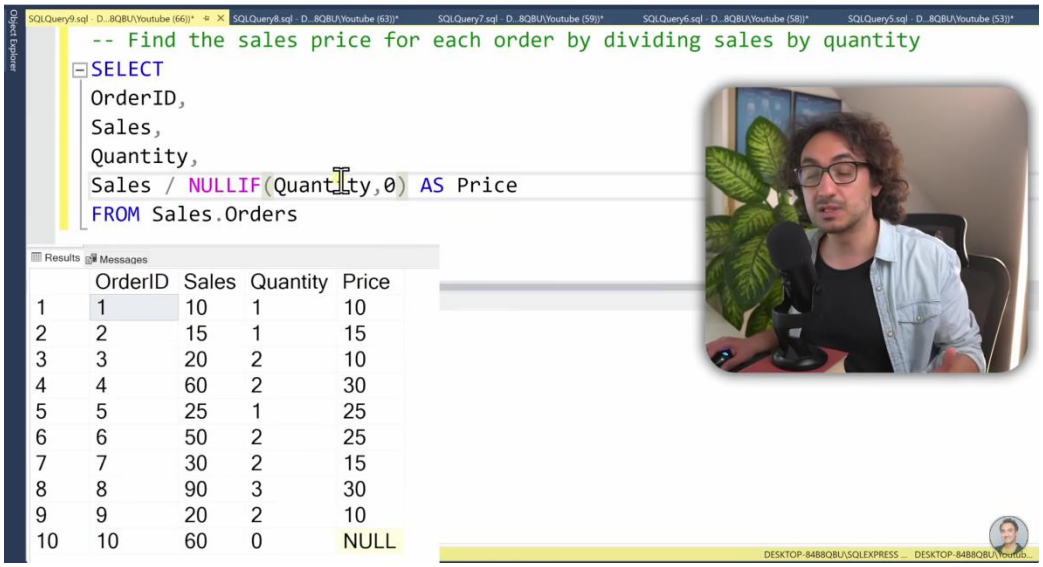
-First Value, if they are not equal.





NULLIF USE CASES:

DIVISION BY ZERO



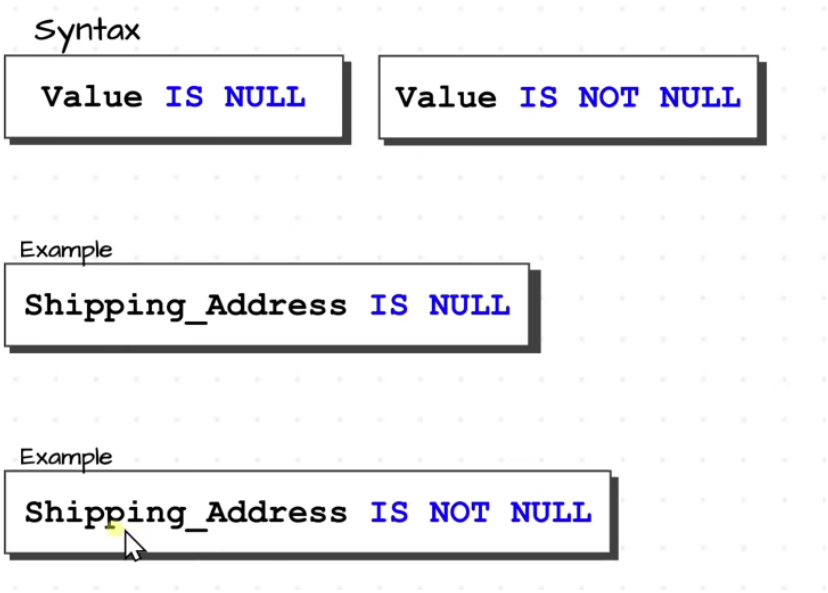
IS NULL AND IS NOT NULL:

IS NULL:

Return true if the value is null. If not null returns false.

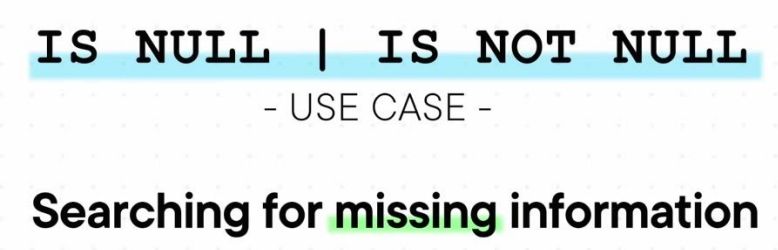
IS NOT NULL:

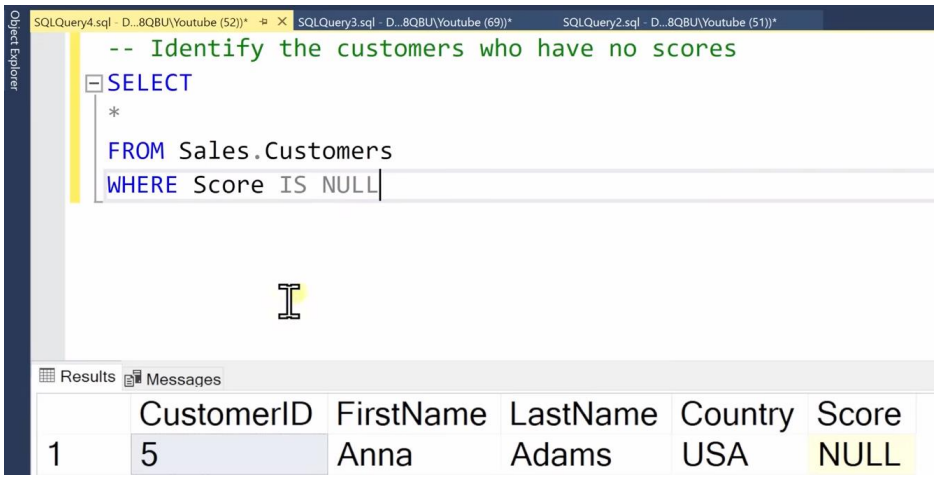
Return true if the value is not null. If null returns false.



IS NULL USECASE:

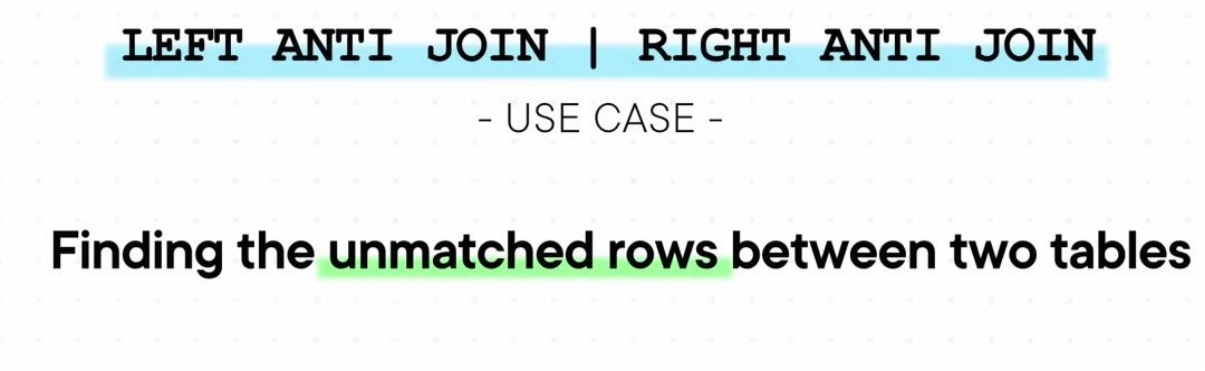
FILTERING DATA.

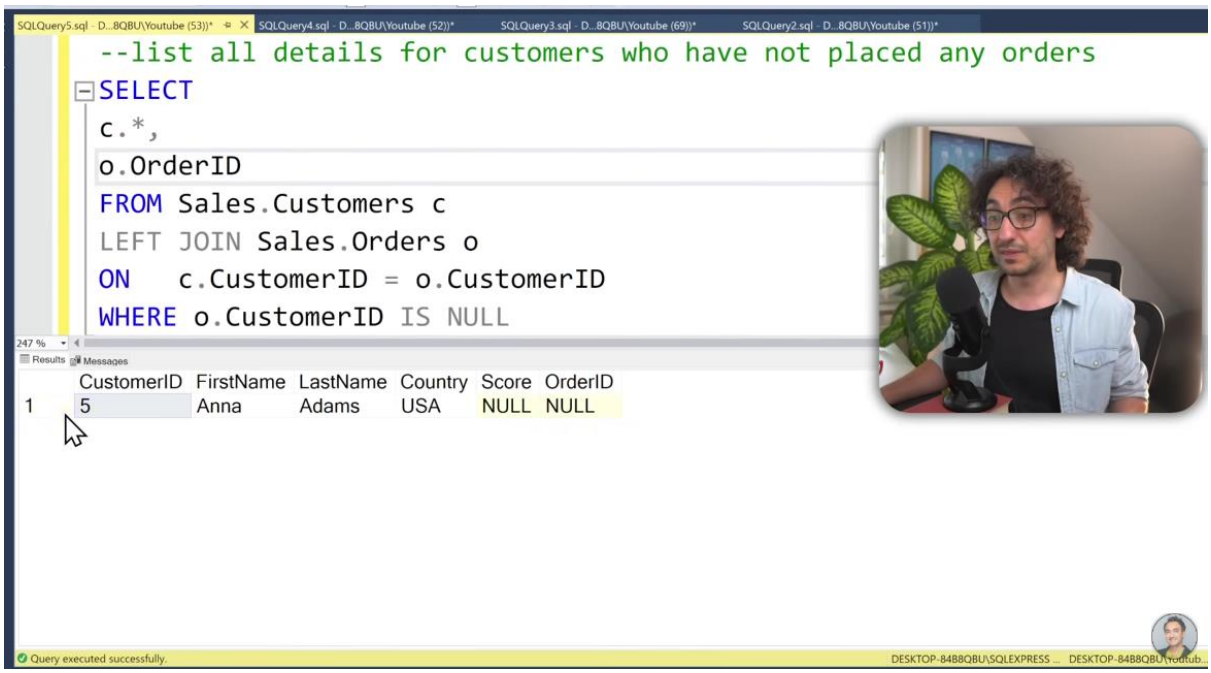




IS NULL USE CASES:

ANTI-JOINS.



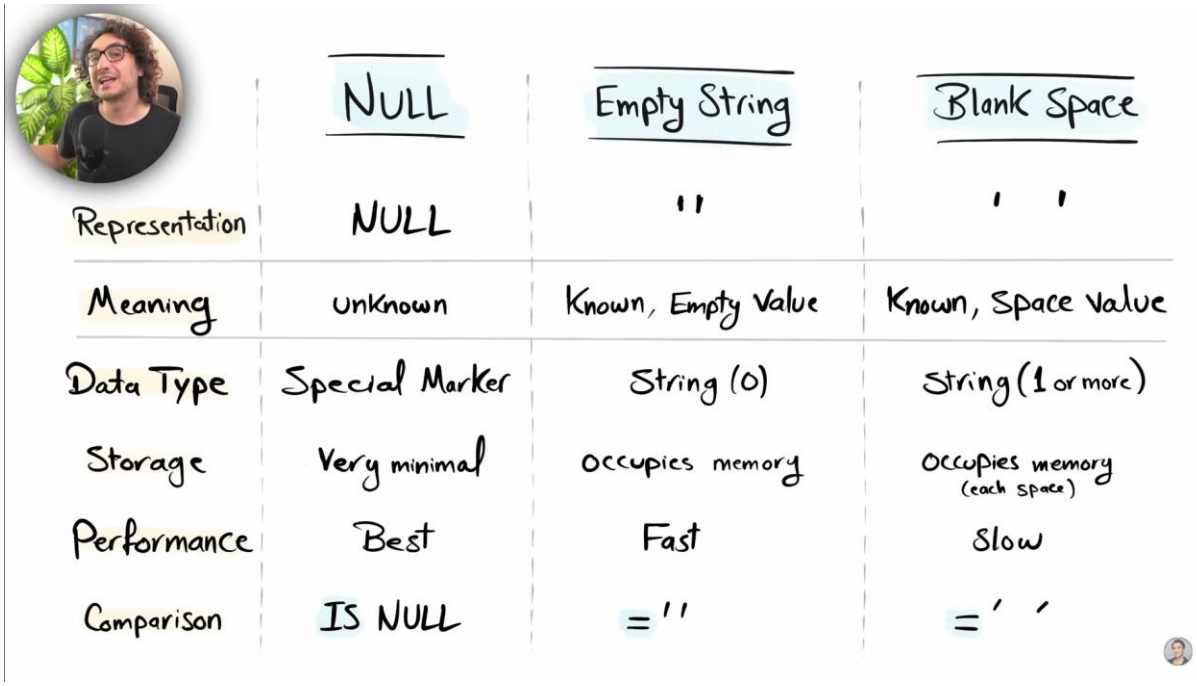


NULL vs EMPTY vs SPACE.

```sql
WITH Orders AS (
SELECT 1 Id, 'A' Category UNION
SELECT 2,  NULL  UNION
SELECT 3,  ''  UNION
SELECT 4,  ' '
)
SELECT
*,
DATALENGTH (Category) CategoryLen
FROM Orders
```

Results | Messages

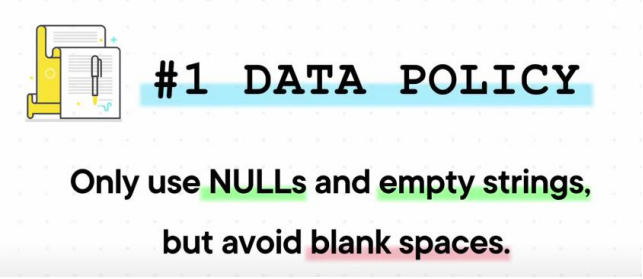
	Id	Category	CategoryLen
1	1	A	1
2	2	NULL	NULL
3	3		0
4	4		1

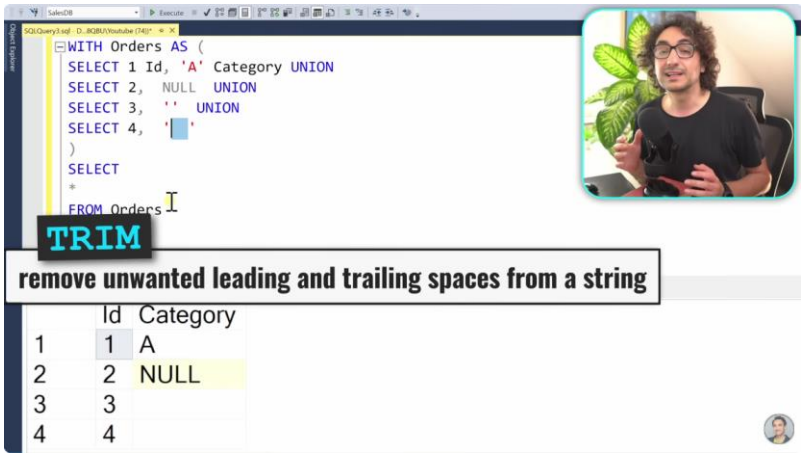


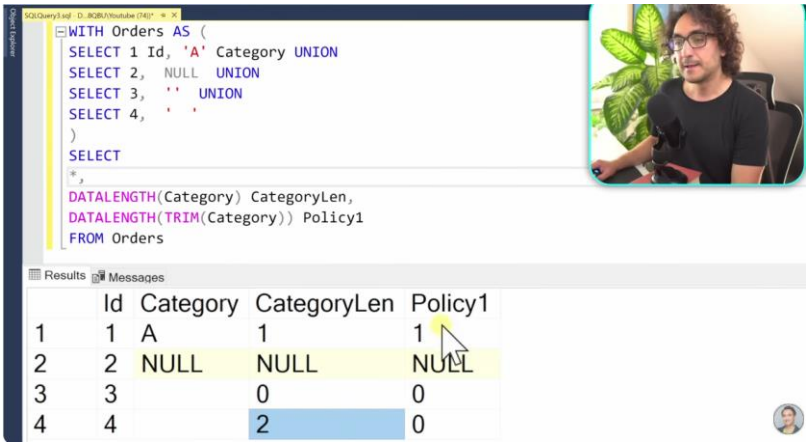
	NULL	Empty String	Blank Space
Representation	NULL	''	' '
Meaning	unknown	Known, Empty Value	Known, Space Value
Data Type	Special Marker	String (0)	String (1 or more)
Storage	Very minimal	occupies memory	Occupies memory (each space)
Performance	Best	Fast	Slow
Comparison	IS NULL	= ''	= ' '

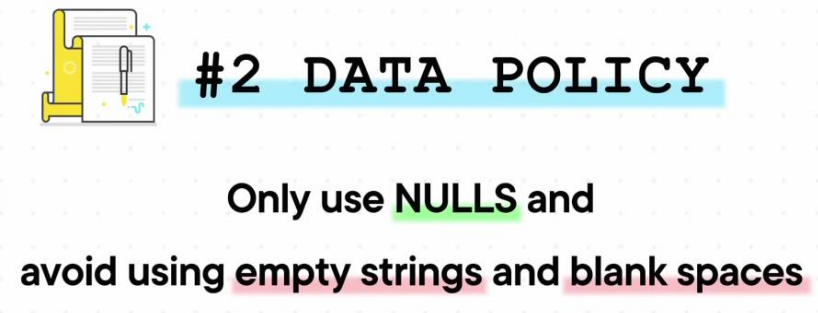
HANDLING NULLS:

DATA POLICIES.









```sql
WITH Orders AS (
SELECT 1 Id, 'A' Category UNION
SELECT 2,  NULL  UNION
SELECT 3,  ''  UNION
SELECT 4,  '  '
)
SELECT
*,
TRIM(Category) Policy1,
NULLIF(TRIM(Category), '') Policy2
FROM Orders
```

Results | Messages

	Id	Category	Policy1	Policy2
1	1	A	A	A
2	2	NULL	NULL	NULL
3	3			NULL
4	4			NULL

#3 DATA POLICY

Use the default value 'unknown' and avoid using nulls, empty strings, and blank spaces.

```sql
WITH Orders AS (
SELECT 1 Id, 'A' Category UNION
SELECT 2,  NULL  UNION
SELECT 3,  ''  UNION
SELECT 4,  '  '
)
SELECT
*,
TRIM(Category) Policy1,
NULLIF(TRIM(Category), '') Policy2,
COALESCE(NULLIF(TRIM(Category), '') , 'unknown') Policy3
FROM Orders
```

Results | Messages

	Id	Category	Policy1	Policy2	Policy3
1	1	A	A	A	A
2	2	NULL	NULL	NULL	unknown
3	3			NULL	unknown
4	4			NULL	unknown

#2 DATA POLICY

- USE CASE -

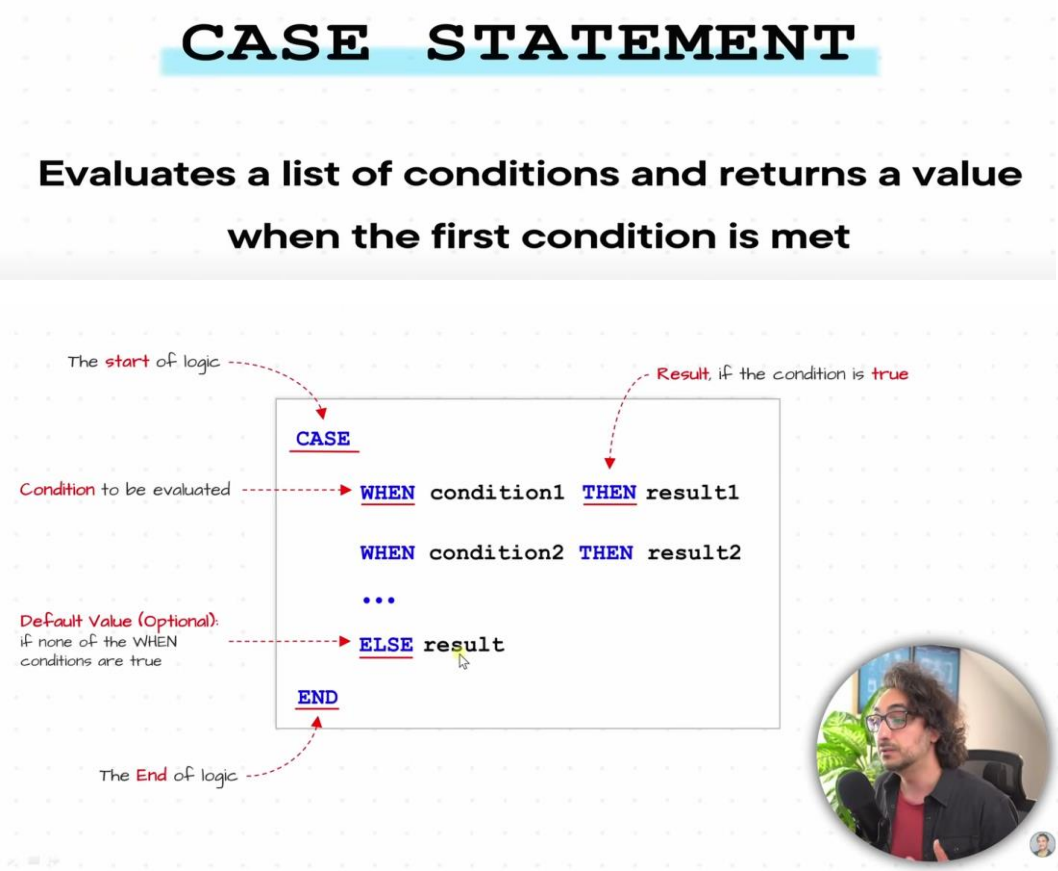
Replacing empty strings and blanks with NULL during data preparation before inserting into a database to optimize storage and performance.

#3 DATA POLICY

- USE CASE -

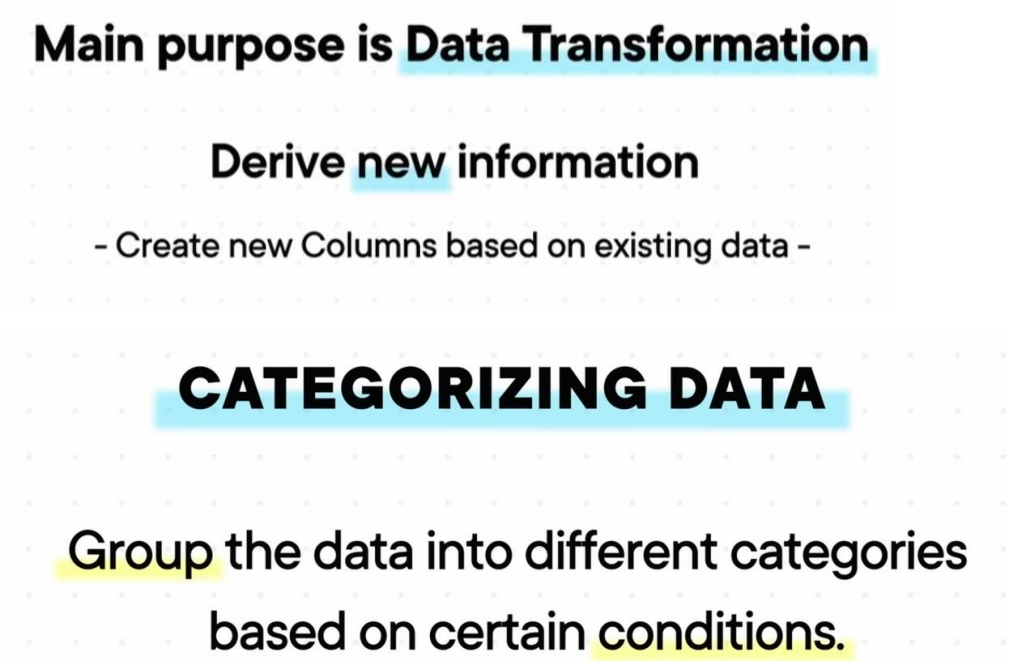
Replacing empty strings, blanks,NULL with default value during data preparation before using it in reporting to improve readiblity and reduce confusion

CASE STATEMENT:



USE CASE:

CATEGORIZING DATA.



SQL TASK

Generate a report showing the total sales for each category:

- High: If the sales higher than 50
- Medium: If the sales between 20 and 50
- Low: If the sales equal or lower than 20

Sort the result from lowest to highest.

```sql
/* Create report showing total sales for each of the following categories:
   High (sales over 50), Medium (sales 21-50), and Low (sales 20 or less)
   Sort the categories from highest sales to lowest */

SELECT
OrderID,
Sales,
CASE
    WHEN Sales > 50 THEN 'High'
    WHEN Sales > 20 THEN 'Medium'
    ELSE 'Low'
END Category
FROM Sales.Orders
```

	OrderID	Sales	Category
1	1	10	Low
2	2	15	Low
3	3	20	Low
4	4	60	High
5	5	25	Medium
6	6	50	Medium
7	7	30	Medium
8	8	90	High
9	9	20	Low
10	10	60	High

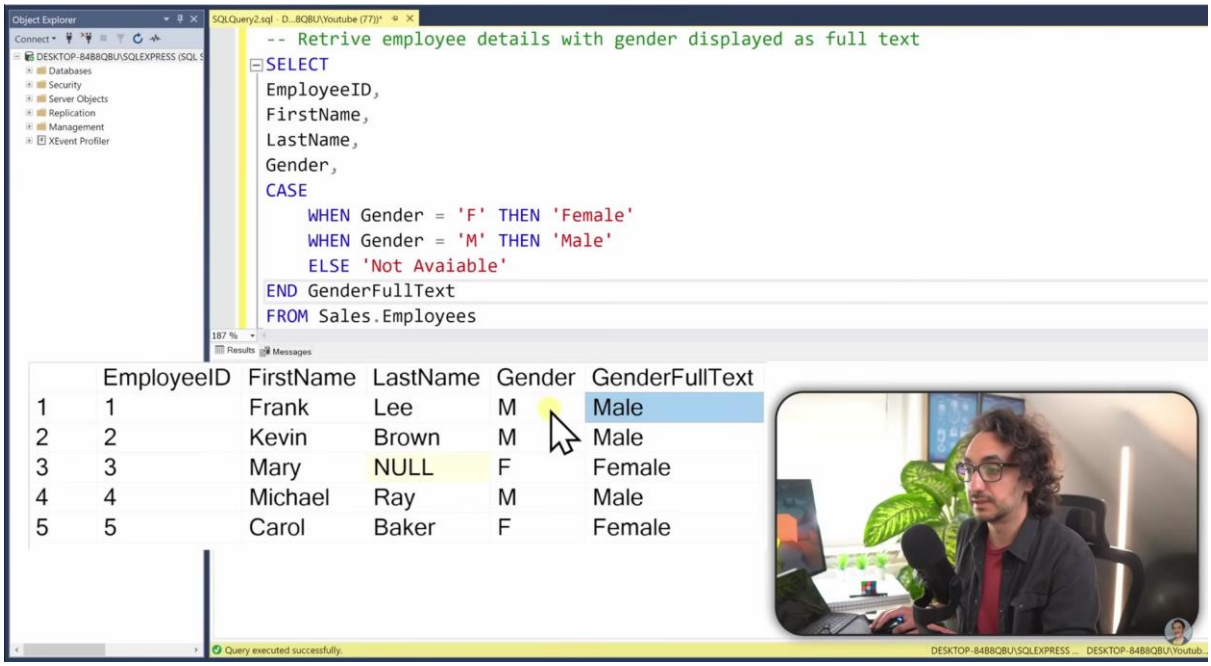
```sql
SELECT
Category,
SUM(Sales) AS TotalSales
FROM(
    SELECT
    OrderID,
    Sales,
    CASE
        WHEN Sales > 50 THEN 'High'
        WHEN Sales > 20 THEN 'Medium'
        ELSE 'Low'
    END Category
    FROM Sales.Orders
)t
GROUP BY Category
```

	Category	TotalSales
1	High	210
2	Low	65
3	Medium	105

Query executed successfully.

```sql
/* Create report showing total sales for each of the following categories:
   High (sales over 50), Medium (sales 20-50), and Low (sales 20 or less)
   Sort the categories from highest sales to lowest */
SELECT
Category,
SUM(Sales) AS TotalSales
FROM(
    SELECT
    OrderID,
    Sales,
    CASE
        WHEN Sales > 50 THEN 'High'
        WHEN Sales > 20 THEN 'Medium'
        ELSE 'Low'
    END Category
    FROM Sales.Orders
)t
GROUP BY Category
ORDER BY TotalSales DESC
```

MAPPING:



SQL TASK

Retrieve customer details with abbreviated country code

```sql
-- Retrive customers details with abbreviated country code
SELECT
    CustomerID,
    FirstName,
    LastName,
    Country,
    CASE
        WHEN Country = 'Germany' THEN 'DE'
        WHEN Country = 'USA'     THEN 'US'
        ELSE 'n/a'
    END CountryAbbr
FROM Sales.Customers
```

	CustomerID	FirstName	LastName	Country	CountryAbb
1	1	Jossef	Goldberg	Germany	DE
2	2	Kevin	Brown	USA	US
3	3	Mary	NULL	USA	US
4	4	Mark	Schwarz	Germany	DE
5	5	Anna	Adams	USA	US

QUICK FORM:

Column Name
to be evaluated (Only One)

```sql
CASE
    WHEN Country = 'Germany' THEN 'DE'
    WHEN Country = 'India'   THEN 'IN'
    WHEN Country = 'United States' THEN 'US'
    WHEN Country = 'France'  THEN 'FR'
    WHEN Country = 'Italy'   THEN 'IT'
    ELSE 'n/a'
END
```

Full Form

```sql
CASE Country
    WHEN 'Germany' THEN 'DE'
    WHEN 'India'   THEN 'IN'
    WHEN 'United States' THEN 'US'
    WHEN 'France'  THEN 'FR'
    WHEN 'Italy'   THEN 'IT'
    ELSE 'n/a'
END
```

Quick Form

HANDLING NULLS:

HANDLING NULLS

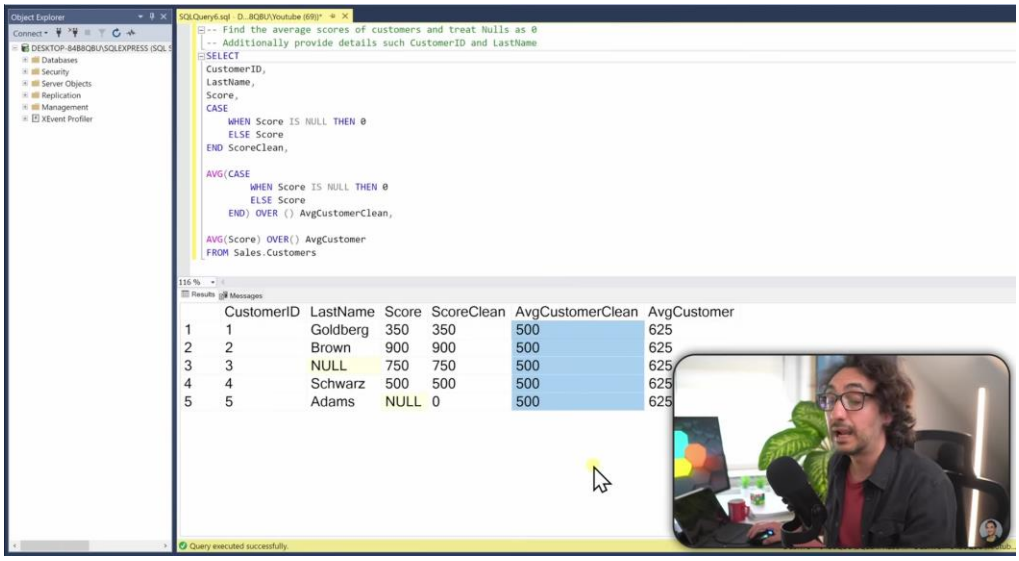
Replace NULLs with a specific value.

NULLs can lead to inaccurate results, which can lead to wrong decision-making.

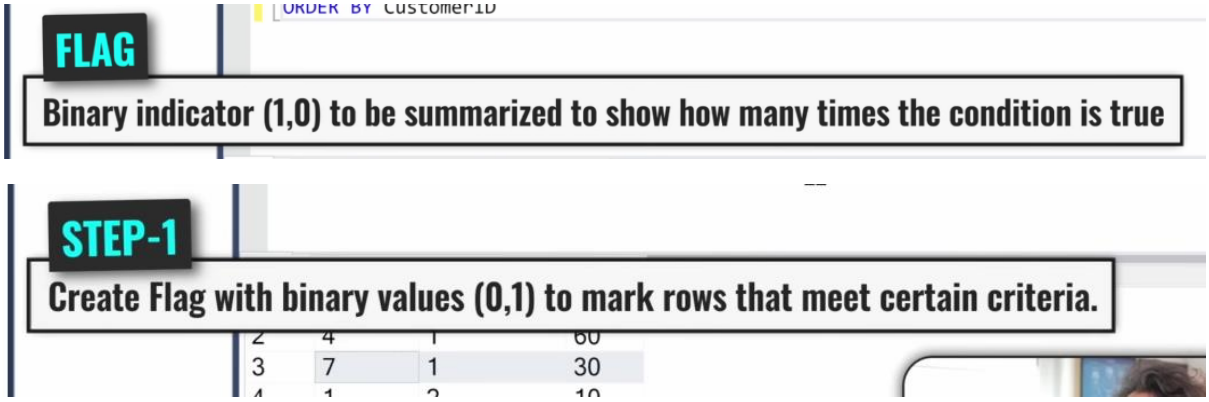
SQL TASK

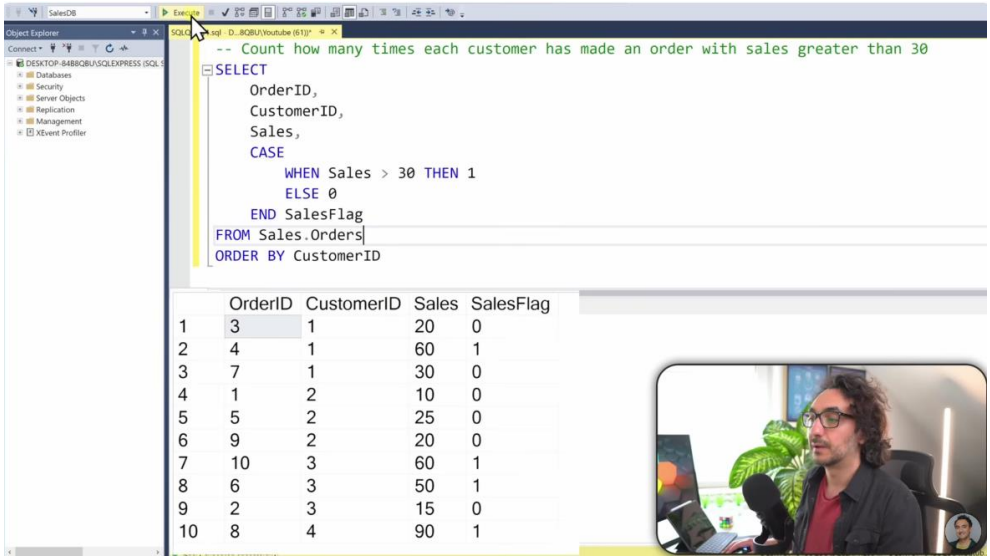
Find the average scores of customers and treat Nulls as 0

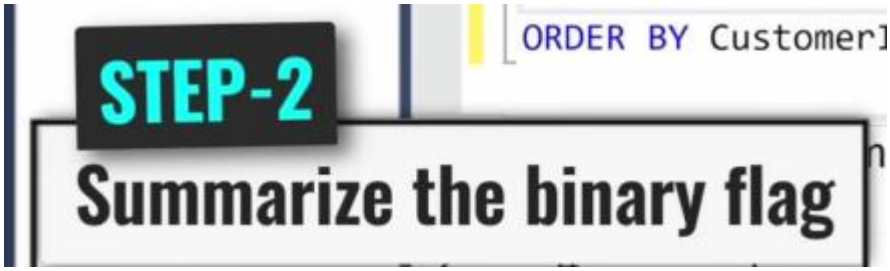
And additional provide details such CustomerID & LastName

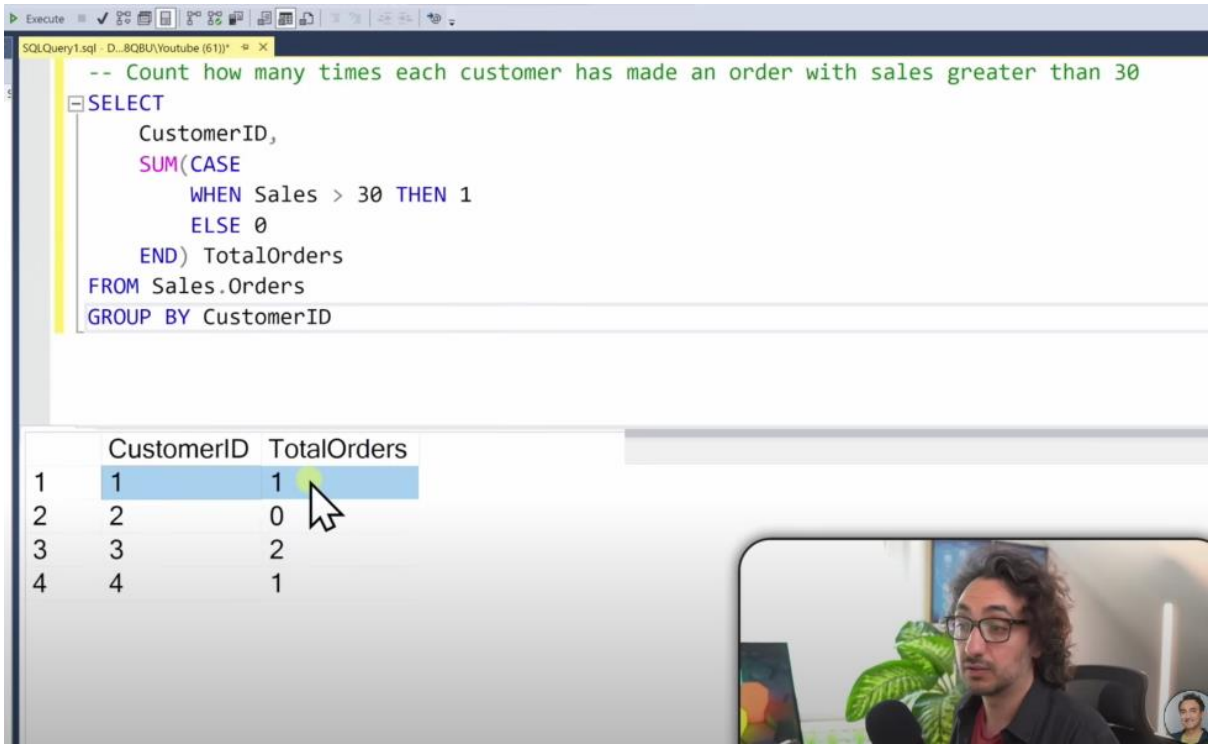


CONDITIONAL AGGRGATION:






ORDER BY Customer
STEP-2
Summarize the binary flag


Execute
SQLQuery1.sql - D...8QBU\Youtube (61))*
-- Count how many times each customer has made an order with sales greater than 30
SELECT
CustomerID,
SUM(CASE
WHEN Sales > 30 THEN 1
ELSE 0
END) TotalOrders
FROM Sales.Orders
GROUP BY CustomerID
CustomerID TotalOrders
1 1 1
2 2 0
3 3 2
4 4 1