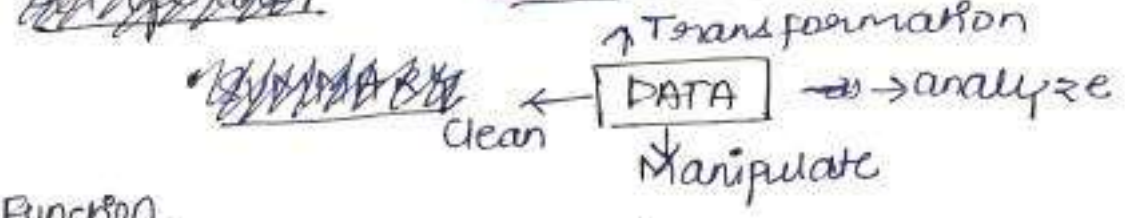
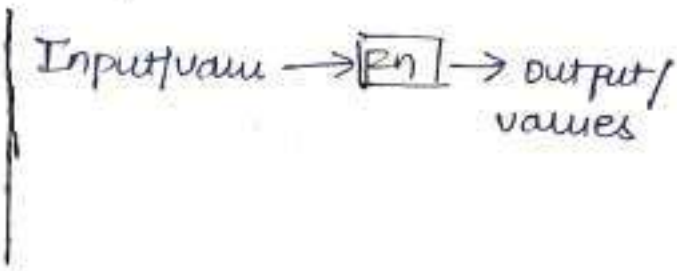
SQL Functions.



Function.

* A built in SQL code
  - accepts an input value.
  - process it.
  - Returns an output value.



1) Single Row Function — 'MARIA' → LOWER() → 'maria'

2) Multi-Row Function — 30, 10, 20 → SUM() → 60

* NESTED Function. : Function used inside another fn.

'Maria' (input) → LEFT(2) ① → 'Ma' (output) → LOWER() ② (input) → 'ma' (output)

③ LEN(LOWER(LEFT('Maria',2)))

① ②

SQL-Function

Row-Level-Calculation. → Single-Row Function.

- String Functions
- Numeric Functions
- Date & Time Function
- NULL Functions

Aggregation. → Multi-Row Function.

- Aggregate Functions (basic)
- window function (advanced)

• SQL STRING FUNCTIONS

Manipulation : CONCAT, UPPER, LOWER, TRIM, REPLACE

Calculation : LEN

String Extraction : LEFT, RIGHT, SUBSTRING

• CONCAT : Combine Multiple string into one.

• Combine firstname and country.

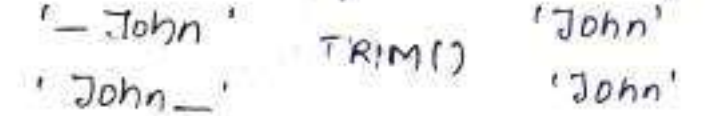
```
SELECT
    Firstname,
    country,
CONCAT(firstname, ' ', country) AS country_name
FROM customers.
```

• UPPER & LOWER : Converts all character to lower case.
↳ : convert all character to upper case.

• Convert firstname to lower case / upper case.

```
SELECT
    LOWER(first_name) AS low_name
    UPPER(first_name) AS upper_name
    FROM customers.
```

• TRIM : Remove leading and Trailing spaces.



Ex:- customer whose first contain leading or trailing spaces.

```
SELECT
    First_name,
    LEN(First_name)
    LEN(TRIM(First_name))
FROM customers
WHERE LEN(First_name) != LEN(TRIM(First_name))
```

- **Replace**: Replace specific character with new character.

123-456-7890 → Replace → 123/456/7890
old value = '-'
new value '/'

Ex: Remove dashes (-) from number.

```
SELECT
'123-456-7890' AS phone
REPLACE('123-456-7890', '-', '/') AS clean-phone.
```

* CALCULATION

- LEN: Counts how many character.

Maria → LEN() → 5

Ex: calculate len of first_name

```
SELECT
  first_name,
  LEN(first_name)
FROM customers.
```

* STRING EXTRACTION

- LEFT: Extract specific No. of character from start
- RIGHT: Extract specifies No. of character from End.

LEFT(value, character)

LEFT(Maria, 2)

Ma

Maria

RIGHT(value, chara.)

RIGHT(Maria, 2)

ia

Ex: Retrieve first and last two character of first_name.

```
SELECT
  first_name
LEFT(first_name, 2) AS Start
RIGHT(first_name, 2) AS End
FROM customers
```

- **Substring**: Extract a part of string at a specified position.

SUBSTRING(value, start, length)

	Start	Length
Maria	3	len() → ria
Martin	3	len() → rtin

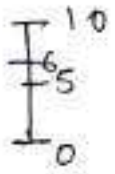
Ex:- First names after removing first character.

```
SELECT
    first_name,
SUBSTRING(TRIM(first_name),2, LENGTH(first_name))
    AS sub_name
FROM customers
```

* Number functions

Roundup ↑

3.516 → ROUND2 → 3.52

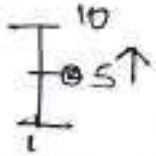


↳ ROUND1 → 3.500

Nothing

↳ ROUND0 → 4.000

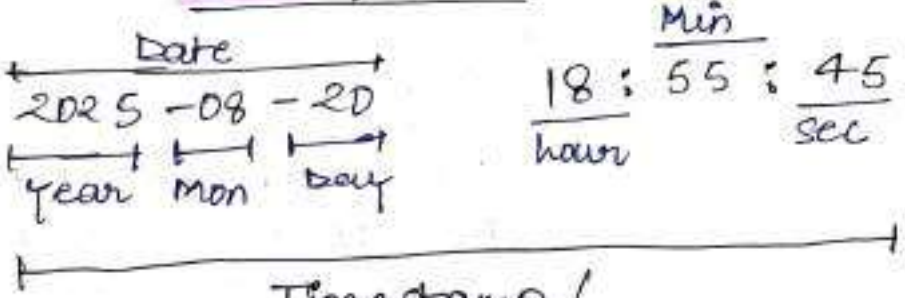
Roundup ↑



• ABS : Return absolute(positive) value of a number, remove any negative sign.

```
SELECT
    -19,
    ABS(-19).
```

DATE & TIME

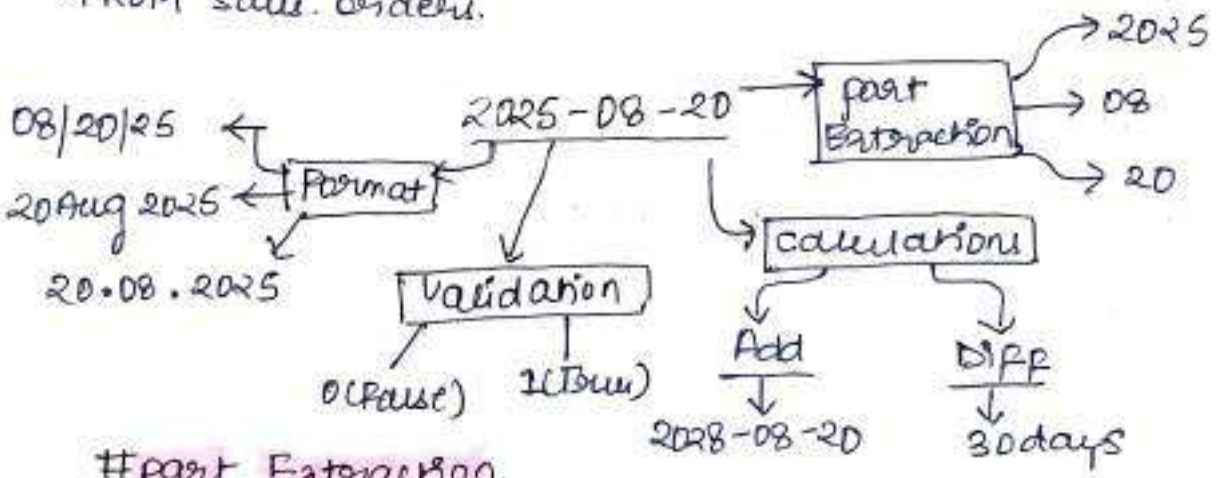


*DATE & TIME

- VALUES

1) Date Column From a Table.
2) Hardcoded constant string value.
3) GetDATE() Function.
   ↳ Return the current date and time at the moment when the Query is executed.

```
SELECT
  CreatedOnTime,  ——→ column
  '2025-08-20' Hardcoded,  ——→ Hardcoded String.
  GETDATE() Today  ——→ Function.
FROM Sales.orders.
```

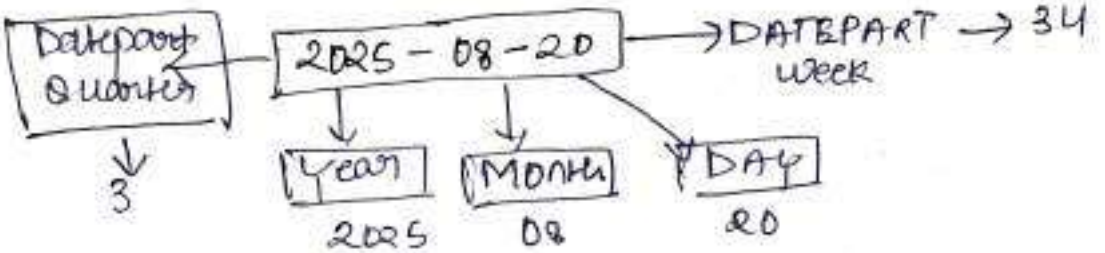


Part Extraction.

- Day | Month | Year.
  - Day() Return the day from date.
  - Return Year()
  - Return the Month()

```
SELECT
  CreationTime, ABC ABC,
  YEAR(CreationTime),
  DAY(CreationTime),
  MONTH(CreationTime)
  FROM Sales.orders.
```

• **Datepart()** : Return a specific part of date as a Number.



Syntax.

DATEPART(part, date)

Examples.

• DATEPART(MONTH, OrderDate)
(mm)

• DATEPART(WEEKDAY, Date)

• DATEPART(Year, CreationTime)

• **DATENAME()** : Return a string name of a specific part of a date.

DATENAME(WEEKDAY) = wednesday.

2025-08-20 → DATENAME → August
MONTH

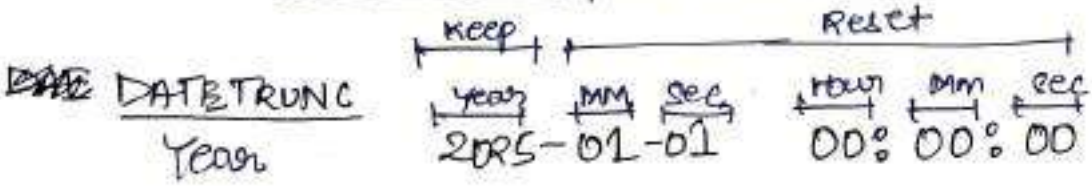
Syntax

DATENAME(part, DATE)

DATENAME Type is string.

• **DATETRUNC()** : Truncates the date to specific part.

Syntax: DATETRUNC(%part, date)

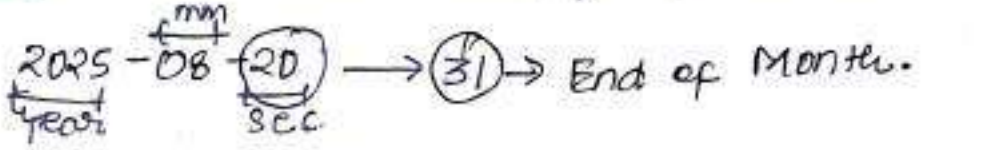


Datepart resets to 1 | Timepart starts to 00

```
SELECT
DATETRUNC(month, CreationTime) as ,
COUNT(*)
GROUP BY DATETRUNC(month, Creation Time).
```

TO analyse the order

- EOMONTH() : Returns the last day of Month.



Syntax.

```
EOMONTH(date)
```

```
SELECT
EOMONTH(CreationTime)
FROM Sales.Orders
```

- Part Extraction use case.

Data Aggregations

- How many order were placed each year?

Sol.

```
SELECT
YEAR(OrderDate),
COUNT(*)
FROM Sales.Orders
GROUP BY YEAR(OrderDate)
```

```
SELECT
DATENAME(Orderdate),
COUNT(*)
FROM Sales.Orders
GROUP BY DATENAME(Orde
-Date).
```

- Data Filtering

- Show all order were placed during Feb Month.

```
SELECT *
  FROM Sales.Orders
WHERE MONTH(OrderDate) = 2
```

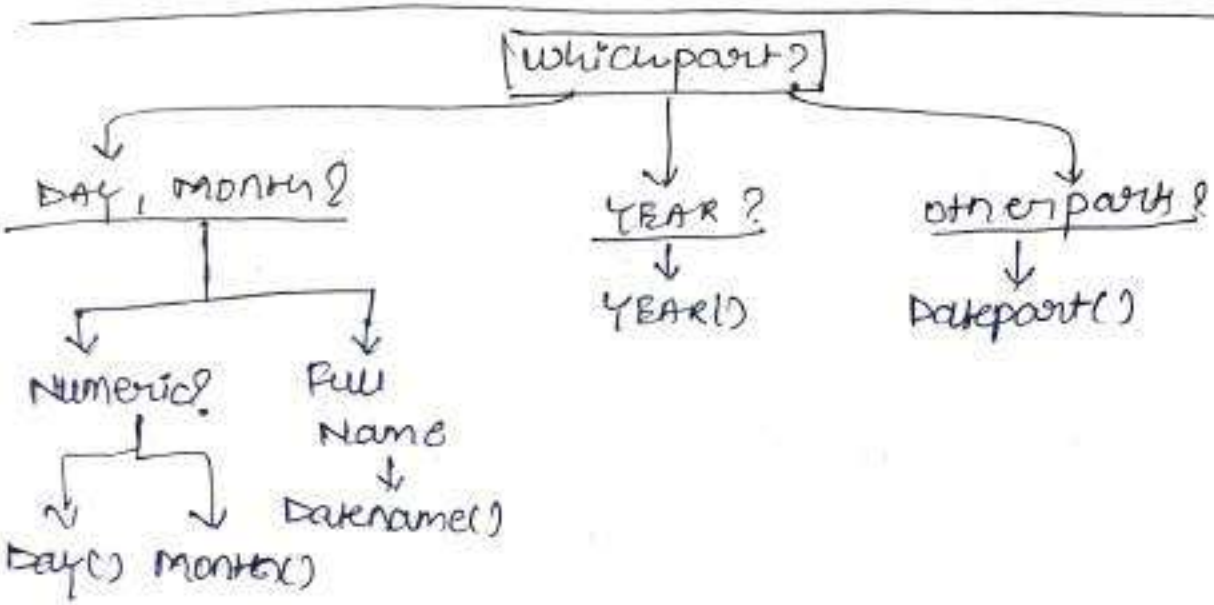
- Functions comparison

Day Month Year Datepart → INT

DateName → String

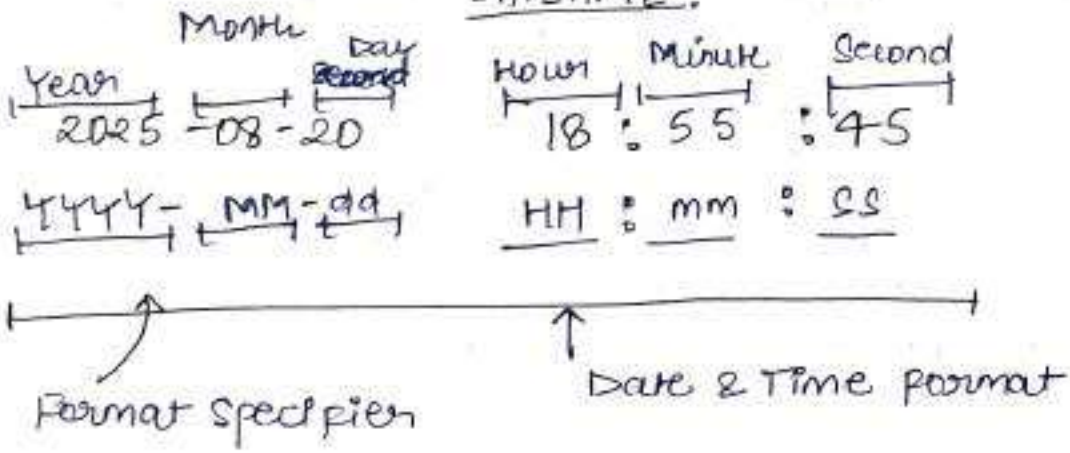
Date Trunc → DateTime

EOMONTH → Date



FORMART, CONVERT, CAST

DATETIME.



2025-08-20 } International Standard (ISO 8601)
YYYY-MM-dd } → SQL Server

08-20-2025 } USA standard
MM-dd-YYYY }

20-08-2025 } European Standard
dd-MM-YYYY }

* What are formatting & casting

↙ changing the format of a value from one to another.

• changing how the data looks.

* FORMATTING
2025-08-20 → FORMAT → MM/dd/YYYY → 08/20/2025
→ MMM/YYYY → Aug 2025
→ CONVERT → style (6) → 20 aug 25
→ (112) → 20250820

Date.

Number
1234567.89 → FORMAT → (N) → 1,234,567.89
→ (C) → $1,234,567.89
→ (P) → 123,456,789.00%.

* CASTING : change the data type from one to another.

CAST()
CONVERT()

String '123' → 123 Number

Date 2025-08-20 → '2025-08-20' String.

* FORMAT()

Format the Date and Time value.

Syntax : FORMAT(Value, Format [, culture])
optional

Ex. FORMAT(orderdate, 'dd/MM/yyyy') — use

FORMAT(orderdate, 'dd/MM/yyyy', 'ja-JP')

FORMAT(1234.56, 'D', 'fr-FR') → optional

Ex SELECT
orderID,
creationTime,
FORMAT(creationTime, 'dd',) dd
FROM Sales.orders.

'dddd' → Full Name , 'MMMM' → Full Name.

• Show = DAY Wed Jan Q1 2025 12:34:56 PM

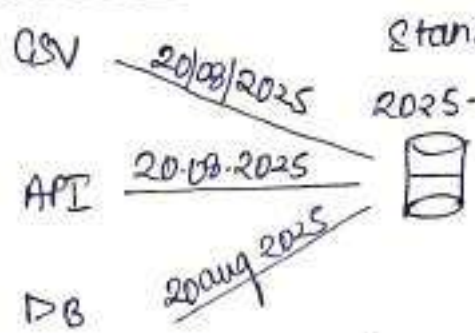
SELECT
orderID,
creationTime,
'DAY ' + FORMAT(creationTime, 'ddd MMM') +
'Q ' + DATENAME(QUARTER, creationTime) + ' ' +
FORMAT(creationTime, 'yyyy hh:mm:ss tt') AS
customFormat
FROM Sales.orders.

• Formatting use case

Data aggregations.

SELECT
FORMAT(creationTime, 'MMM yy') orderdate
COUNT(*)
FROM Sales.orders
GROUP BY FORMAT(orderdate, 'MMM yy')

- Data Standardization



- All formats.

- CONVERT() converts a date or time value to a different data type. & it helps FORMAT the value.

Syntax CONVERT(data-type, value [,style])
optional.

Eg CONVERT(INT, '123')
CONVERT(VARCHAR, orderDate, 34')

Default style = 0.

Eg
```
SELECT
CONVERT(INT, '123')
CONVERT(DATE, '2025-08-20')
CONVERT(DATE, creationTime)
FROM sales.orders.
```

- CAST():

Converts a value to specified data-type.

Syntax CAST(value AS data_type)

Ex: CAST('123' AS INT)
CAST('2025-08-20' AS DATE).

Not format specified.

SELECT

CAST('123' AS INT)

CAST('2025-08-20' AS DATE)

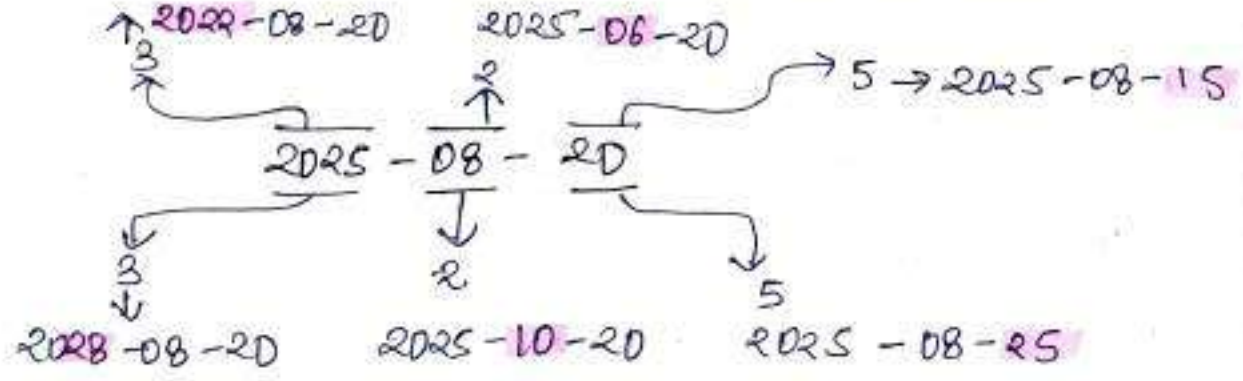
CAST('2025-08-20' AS DATETIME)

	CASTING	FORMATING
CAST	any type to any type	XNO formating
CONVERT	any type to any type	Format only Date & time
FORMAT	any type to only String	Formats

Formats → Date & time, Number

DATE CALCULATIONS

- **DATEADD()** : Add or substracts a specific time interval to/from a date.



syntax.

DATEADD(part, interval, date)

Ex:

DATEADD(year, 2, orderDate)

DATEADD(month, -4, orderDate)

• Add 3 Month and 2 year

```
SELECT
    OrderID,
    OrderDate,
DATEADD(Month, 3, orderdate),
DATEADD(YEAR, 2, orderdate)
FROM Sales.orders.
```

LAG(): access a value from previous ROW.

• DATEDIFF(): Find differen b/w two dates.
(Start Date − End date)

Syntax: DATEDIFF(part, start_date, End_date)

Eg: DATEDIFF(YEAR, orderDate, shipDate)

DAY
Month

• Calculate age from current

```
SELECT
    EmployeeID,
    BirthDate,
DATEDIFF(YEAR, BirthDate,
    GETDATE())
FROM Sales.Employees.
```

• average shipping duration in days for each month.

```
SELECT
    Month(orderDate) AS orderdate,
    AVG(DATEDIFF(DAY, orderdate, shipDate)) avg
FROM Sales.orders
GROUPBY MONTH(OrderDate).
```

• Find no. of day b/w each order date and previous.

```
SELECT
    OrderID,
    orderDate current,
    LAG(orderDate) OVER(ORDER BY Orderdate) prev.
DATEDIFF(DAY, LAG(orderDate) OVER(ORDER BY orderdate),
        orderDate) No of days.
FROM Sales.orders
```

- **ISDATE()**: Check If a value is a date.
  Returns 1 If the string value is a valid date.

Syntax ISDATE(value)

```
ISDATE('2025-08-20')
ISDATE(2025)

SELECT
 ISDATE('1:23') check1
 ISDATE('2025-20-08') check2.
```

```
SELECT
    --CAST(orderdate AS DATE) orderDate,
    OrderDate,
    ISDATE(OrderDate),
    CASE WHEN ISDATE(orderdate) = 1 THEN
    CAST(OrderDate AS DATE)
        ELSE '9999-01-01'
    END NEWorderdate
FROM
  (
    SELECT '2025-08-20' AS orderDate UNION
    SELECT '2025-08-21' UNION
    SELECT '2025-08-23' UNION
    SELECT '2025-08'
  ) t
  --WHERE ISDATE(Order Date) = 0
```

NULL

- NULL means nothing, unknown!
- Null is not equal to anything.
  - Null is not zero - Null is not empty string
  - Null is not blank space.
- I don't know what this value is this.

Q How handle null inside our data?

Replace value.

NULL → (IS NULL / COALESCE) → 40 ; 40 → (NULLIF) → NULL

check whether we have Null or not.

check for value.

NULL → IS NULL → TRUE (space)
NULL → IS NOT NULL → FALSE

#NULL Function.

* ISNULL : Replaces NULL with a specified value.

Syntax. ISNULL(value, replacement_value)

Eg: ISNULL(shipping_add, 'unknown') ← always not unknown we also another column for help of value

ISNULL(shipping_add, Billing_add)

* COALESCE : Return the first non-null value from list.

Syntax : COALESCE(value1, value2, value3, ---)

Eg: COALESCE(shipping_add, 'unknown')

Eg: COALESCE(shipping_add, Billing_add)

Eg: COALESCE(shipping_add, Billing_add, 'unknown')

- ISNULL VS COALESCE

ISNULL	COALESCE
• Limited to two values.	• unlimited.
• Fast	• Slow
SQL Server → ISNULL	• Available in all databases
Oracle → NVL	
MySQL → IFNULL	

- Use case - Handling values.

* Data aggregation.

- Handle NULL before doing data aggregations.

```
SELECT
  customerID,
  Score,
COALESCE(Score,0) Score2
AVG(Score) OVER() AvgScore,
AVG(COALESCE(Score,0)) OVER() AvgScore2
FROM Sales.customers.
```

This handle Null values.

it return avg via handle Null values.

* Mathematical operations

- Handle NULL before doing any mathematical operation.

```
Null + 5 → Null
5 + 1 → 6
"A" + "B" → "AB"
```

If we do maths operation with NULL the answer is always NULL.

```
SELECT
  customerID,
  Firstname,
  Secondname,
  Firstname + ' ' + COALESCE(Lastname,'') AS Fullname,
  Score,
  COALESCE(Score,0) + 10 AS BonusScore
FROM Sales.customers.
```

* Handle NULL Before joining Tables

```
SELECT
  a.year, a.type, a.orders, b.sales
FROM Table1 a
JOIN Table2 b
ON a.year = b.year
AND ISNULL(a.type,'') = ISNULL(b.type,'')
```

↓ Remove NULL to do join the Tables.

* Handle Null

• Sorting Data in

```
SELECT
  customer ID,
  Score,
  CASE WHEN Score IS NULL THEN 1 ELSE 0 END
FROM Sales.customers
```

• Sort customer from lowest to highest with Nulls appearing last.

```
order By 1, Score
```

• First non-null score in ascending order, then Null is last.

• If non-null same then use Score and sort them.

* NULL FUNCTION - NULLIF

Compare two expressions returns:

- NULL, If they are equal.
- First value, If they are not equal.

Syntax NULLIF(Value1, value2)

Eg NULLIF(Ship-add, 'unknown')

Eg NULLIF(Ship-add, Bill-add)

- NULLIF use case
  - Division by zero : preventing error of dividing by zero.

```
SELECT
  orderID,
  Sales,
  Quantity,
  Sales/NULLIF(Quantity,0) AS price
FROM Sales.orders
```

NULLIF(Quantity,0) → prevent error if any zero in column.

- IsNULL / IsNOTNULL

Return True if value is NULL otherwise False.

Return True if value is not null otherwise False.

Syntax: `value IS NULL` `value IS NOT NULL`

Eg: Shipping_add IS NULL

Shipping_add IS NOT NULL

- IS NULL - use case
  - Filtering Data : Searching for missing information.

```
SELECT
  customerID,
  score
FROM Sales.customers
WHERE score IS NULL
WHERE score IS NOT NULL
```

- IS NULL - use case
  - ANTI JOIN : Finding ~~row~~ unmatched Rows between two columns.

```
SELECT
    C.*,
    O.OrderID
FROM Sales.Customers C
LEFT JOIN Sales.Orders O
ON C.CustomerID = O.CustomerID
WHERE O.CustomerID IS NULL
```

NULL VS Empty STRING VS BLANK Space

NULL → Means nothing, unknown.

'' → String value has zero characters.

BLANK SPACE (' ') : String value has one or more space characters.

	NULL	Empty String	Blank Space
Representation	NULL	''	' '
Meaning	unknown	known, Empty value	known, Space value
Datatype	special marker	String (0 or more)	String (1 or more)
Storage	very minimal	occupies Memory	occupies Memory
Performance	Best	Fast	Slow
Comparison	IS NULL	= ''	= ' '

- Handling NULL.

Data policies : Set of rules that defines how data should be handled.

1) Only use NULLs and empty spaces instead but avoid blank spaces.

2) Only use NULLs and avoid Empty strings and blank spaces

↑ Use NVL() to do NULL values.
↳ Use TRIM to remove spaces.

3) use default values (unknow) and avoid using nulls, empty strings, and blank spaces.

#2 Data policy Replace empty and blank spaces with null during data preparation before inserting into a database to optimize storage and performance.

#3 Dat policy : Replace empty and blank and Null with default value during data preparation before using it for Reporting. Improve Readability and reduce confusion.