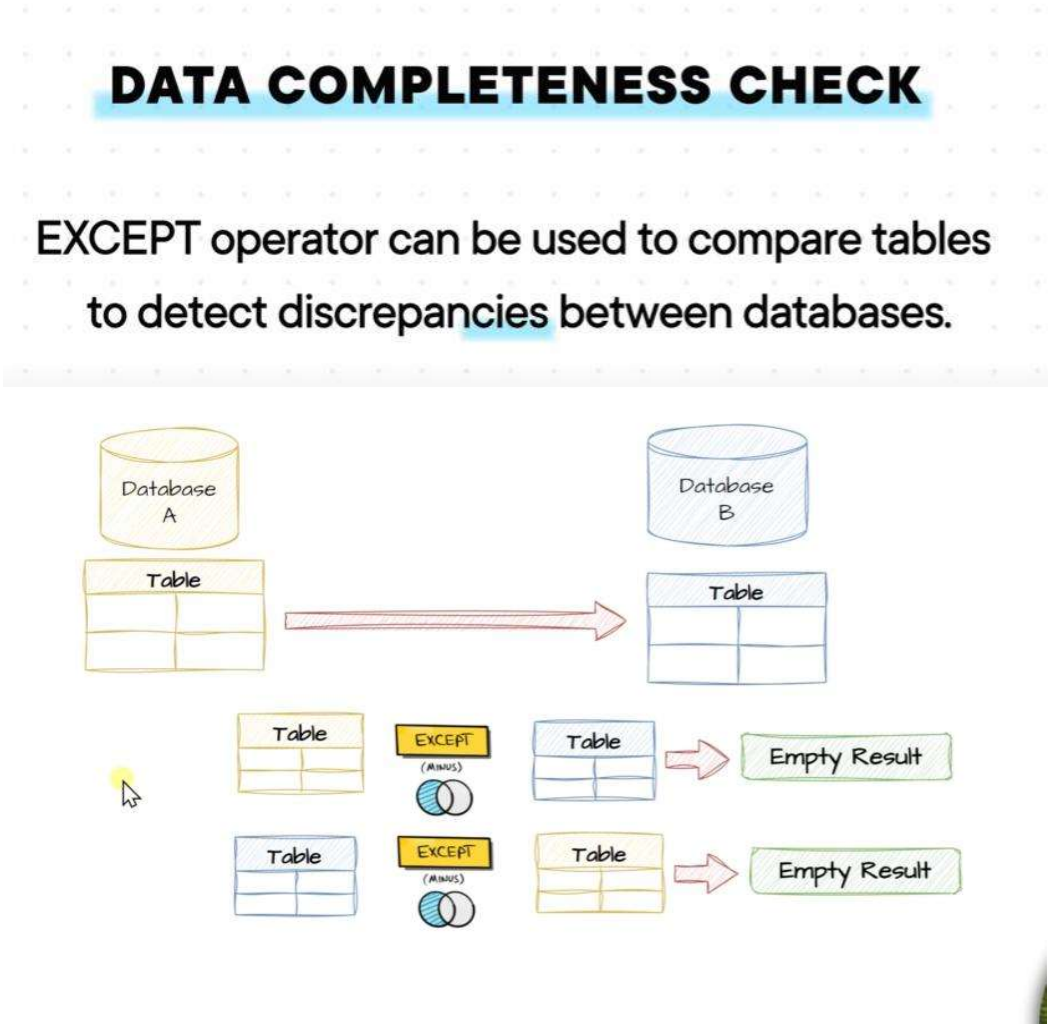
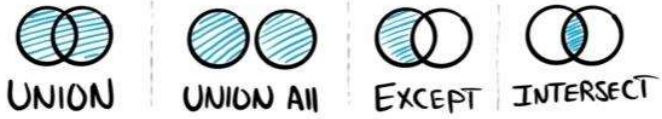


Improve data Migration quality

SET OPERATORS

Combine the results of multiple queries into a single result set

Types

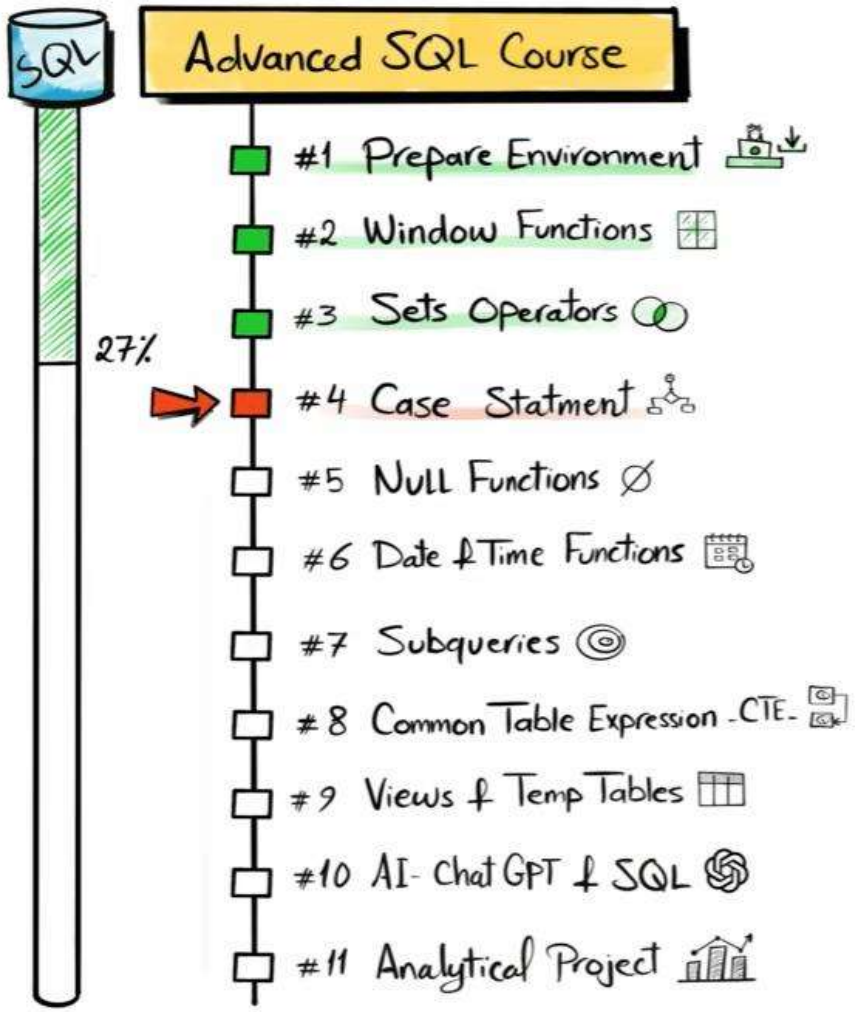


RULES

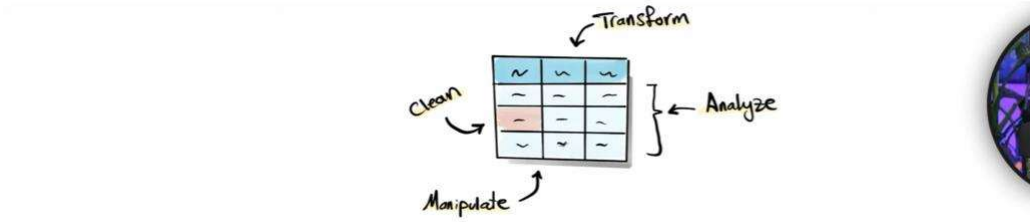
- Same Nr. of Columns, Data Types, order of Columns.
- 1st Query controls Column names.

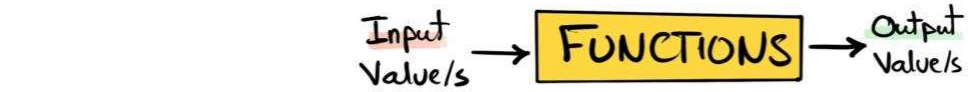
USE CASES

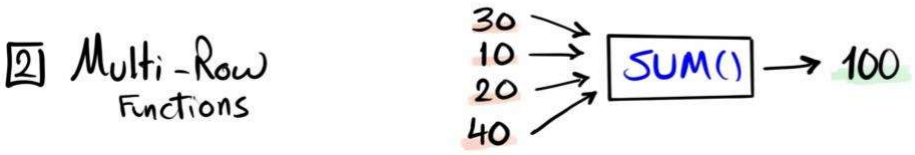
- Combine Information (UNION & UNION All)
- Delta Detection (EXCEPT)
- Data Completeness Check (EXCEPT)



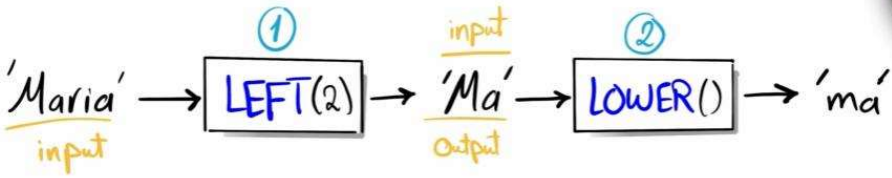
SQL Functions:

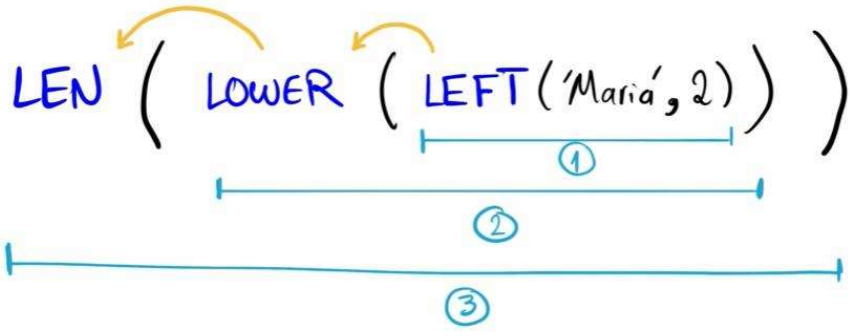




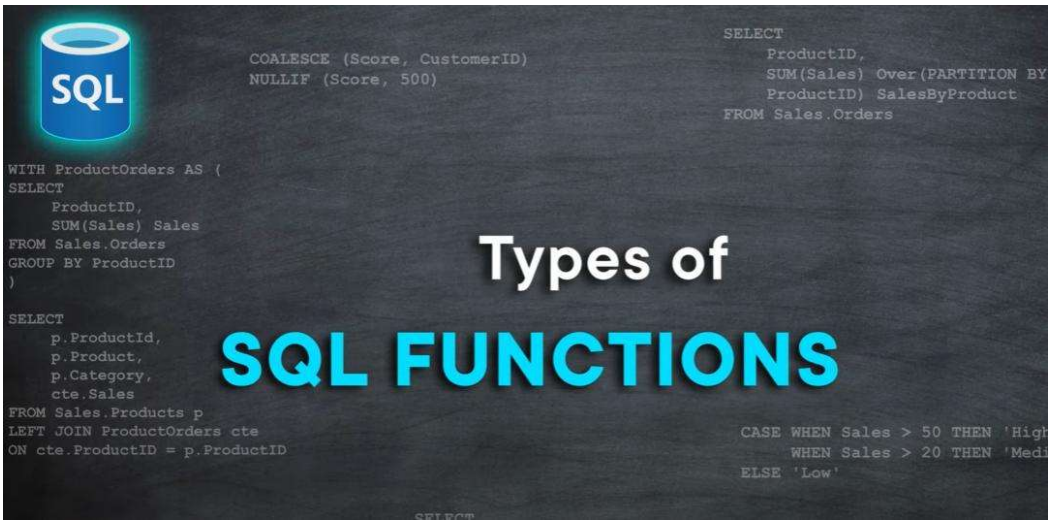


Function used inside another function



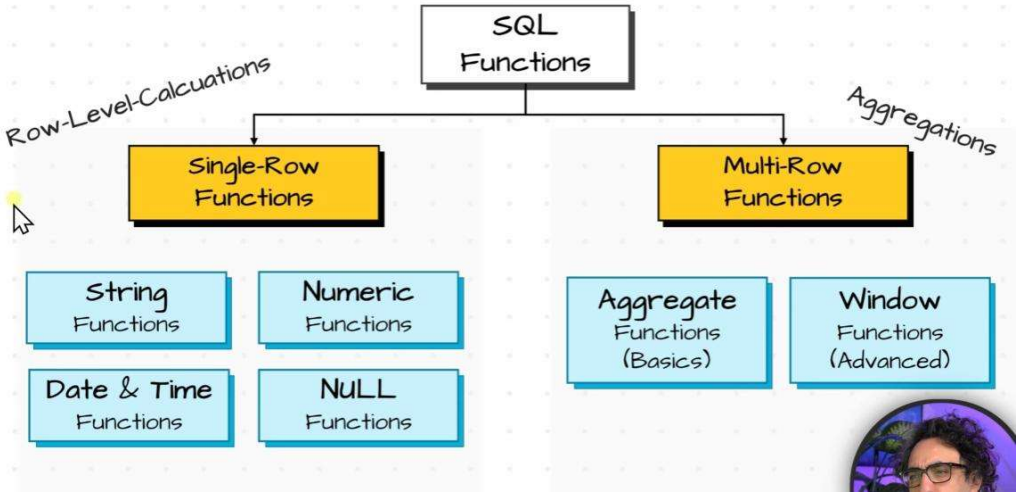


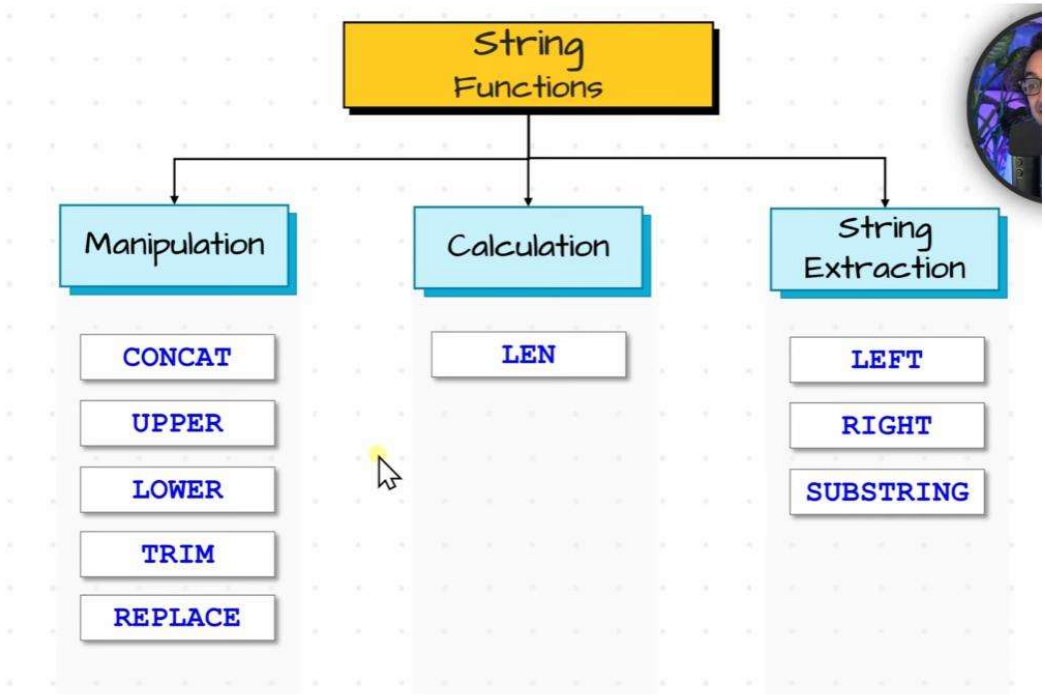
Exectuion is same as order



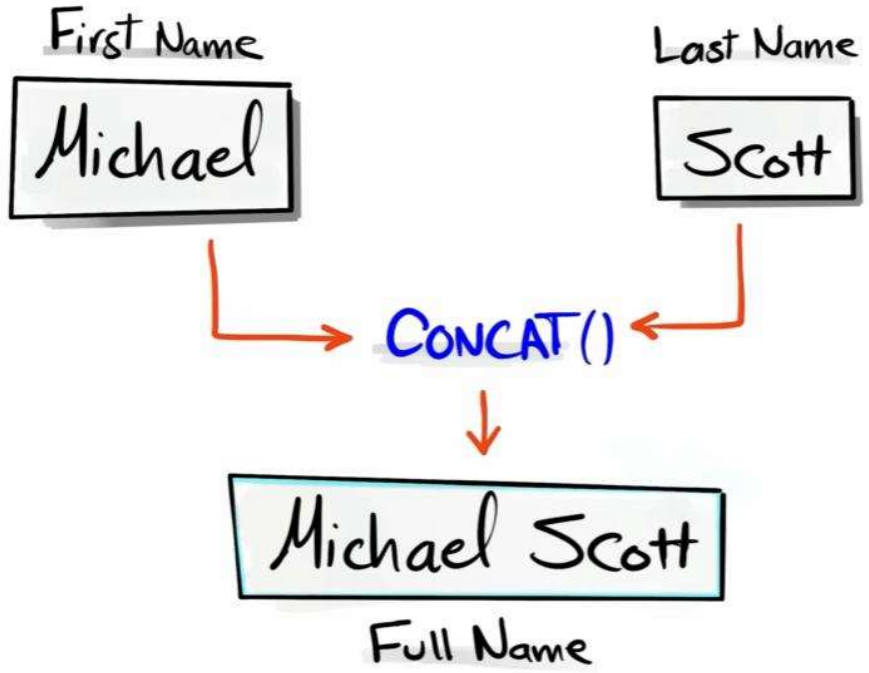
Single row function used to manipulate the data and prepare the data for the second group means for multi-row functions.
So data engineers used single row functions Inorder to clean up, manipulate, and transform the data inorder to prepare it for Analyzes

Multi-row functions used genrally by data analyst





String Concatination:




```sql
-- Show a list of customers' first names together with their country in one column.
SELECT
first_name,
country,
CONCAT(first_name, ' ', country) AS name_country
FROM customers
```

Results Messages

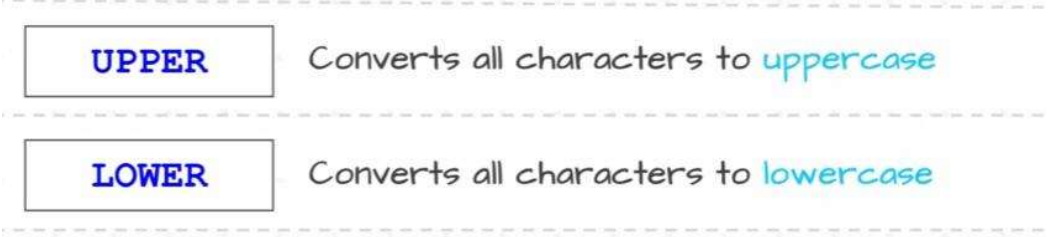
	first_name	country	name_country
1	Maria	Germany	Maria Germany
2	John	USA	John USA
3	Georg	UK	Georg UK
4	Martin	Germany	Martin Germany
5	Peter	USA	Peter USA

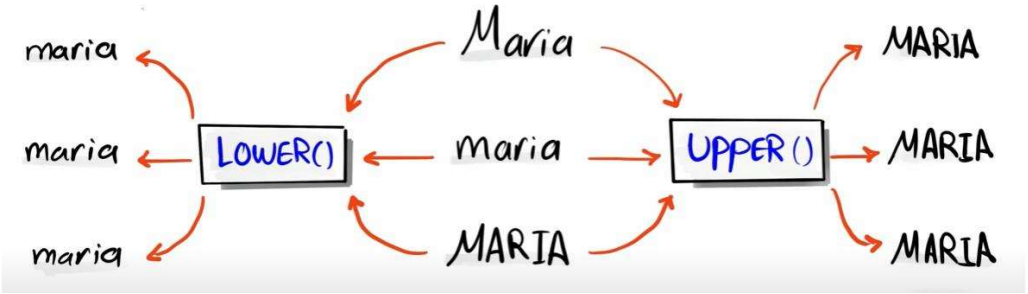
```sql
-- Show a list of customers' first names together with their country in one column.
SELECT
first_name,
country,
CONCAT(first_name, '-', country) AS name_country
FROM customers
```

Results | Messages

	first_name	country	name_country
1	Maria	Germany	Maria-Germany
2	John	USA	John-USA
3	Georg	UK	Georg-UK
4	Martin	Germany	Martin-Germany
5	Peter	USA	Peter-USA

Lower and Upper Function





```sql
-- Transform the customer's first name to lowercase
SELECT
first_name,
country,
CONCAT(first_name, '-', country) AS name_country,
LOWER(first_name) AS low_name
FROM customers
```

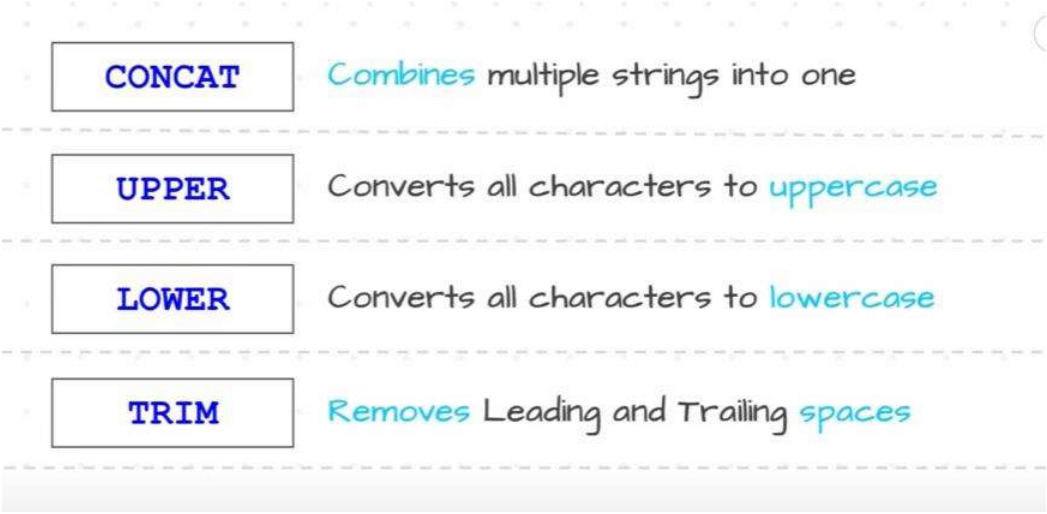
Results | Messages

	first_name	country	name_country	low_name
1	Maria	Germany	Maria-Germany	maria
2	John	USA	John-USA	john
3	Georg	UK	Georg-UK	georg
4	Martin	Germany	Martin-Germany	martin
5	Peter	USA	Peter-USA	peter

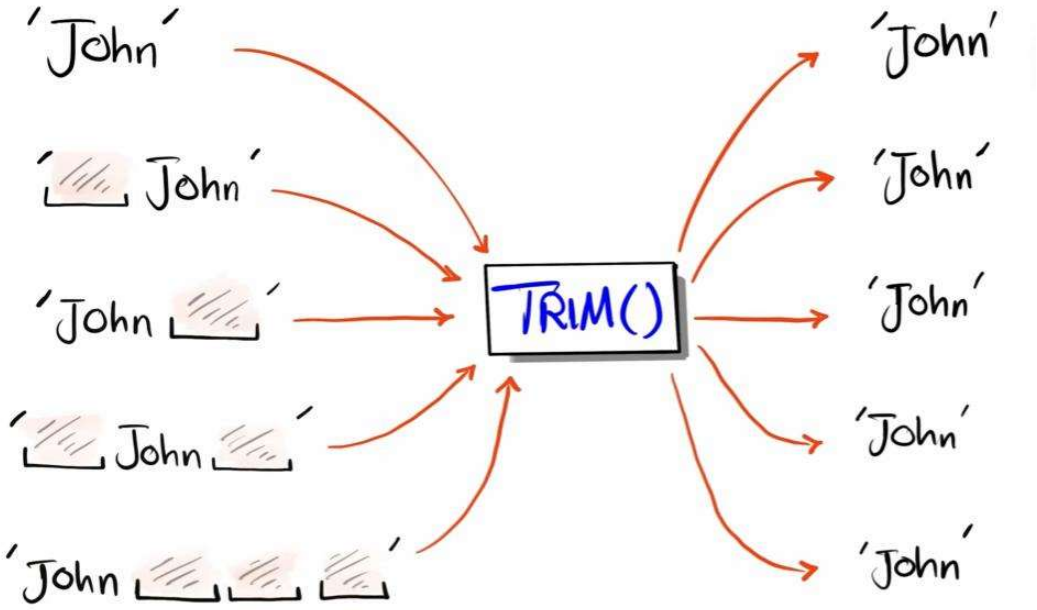
```sql
-- Transform the customer's first name to uppercase.
SELECT
first_name,
country,
CONCAT(first_name, '-', country) AS name_country,
LOWER(first_name) AS low_name,
UPPER(first_name) AS up_name
FROM customers
```

Results | Messages

	first_name	country	name_country	low_name	up_name
1	Maria	Germany	Maria-Germany	maria	MARIA
2	John	USA	John-USA	john	JOHN
3	Georg	UK	Georg-UK	georg	GEORG
4	Martin	Germany	Martin-Germany	martin	MARTIN
5	Peter	USA	Peter-USA	peter	PETER



Means trim remove space from start and at end



Detect Leading and Trailing space

```sql
-- Find customers whose first name contains leading or trailing spaces

SELECT
    first_name
FROM customers
WHERE first_name != TRIM(first_name)
```

```sql
-- Find customers whose first name contains leading or trailing spaces

SELECT
    first_name,
    LEN(first_name) len_name
FROM customers
--WHERE first_name != TRIM(first_name)
```

Results | Messages

	first_name	len_name
1	Maria	5
2	John	5
3	Georg	5
4	Martin	6
5	Peter	5

```sql
-- Find customers whose first name contains leading or trailing spaces
SELECT
    first_name,
    LEN(first_name) len_name,
    LEN(TRIM(first_name)) len_trim_name
FROM customers
--WHERE first_name != TRIM(first_name)
```

Results | Messages

	first_name	len_name	len_trim_name
1	Maria	5	5
2	John	5	4
3	Georg	5	5
4	Martin	6	6
5	Peter	5	5

```sql
-- Find customers whose first name contains leading or trailing spaces
SELECT
    first_name,
    LEN(first_name) len_name,
    LEN(TRIM(first_name)) len_trim_name,
    LEN(first_name) - LEN(TRIM(first_name)) flag
FROM customers
--WHERE first_name != TRIM(first_name)
```

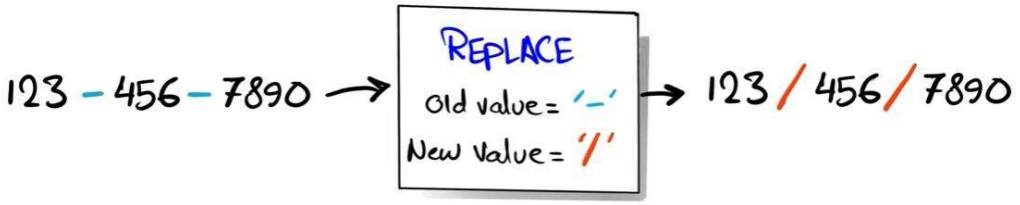
Results Messages

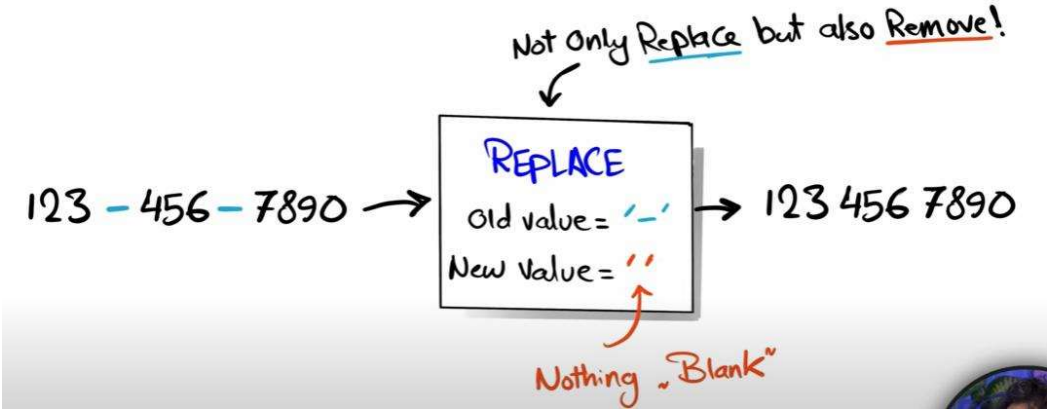
	first_name	len_name	len_trim_name	flag
1	Maria	5	5	0
2	John	5	4	1
3	Georg	5	5	0
4	Martin	6	6	0
5	Peter	5	5	0

```sql
-- Find customers whose first name contains leading or trailing spaces
SELECT
    first_name,
    LEN(first_name) len_name,
    LEN(TRIM(first_name)) len_trim_name,
    LEN(first_name) - LEN(TRIM(first_name)) flag
FROM customers
WHERE LEN(first_name)  != LEN(TRIM(first_name))
-- WHERE first_name != TRIM(first_name)
```

Results Messages

	first_name	len_name	len_trim_name	flag
1	John	5	4	1





```sql
-- Remove dashes (-) from a phone number

SELECT
'123-456-7890' AS phone,
REPLACE('123-456-7890', '-', '') AS clean_phone
```

Results Messages

	phone	clean_phone
1	123-456-7890	1234567890

Usecase of Replace Functions:

```sql
-- Remove dashes (-) from a phone number

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

Results Messages

	phone	clean_phone
1	123-456-7890	123/456/7890

Another usecase

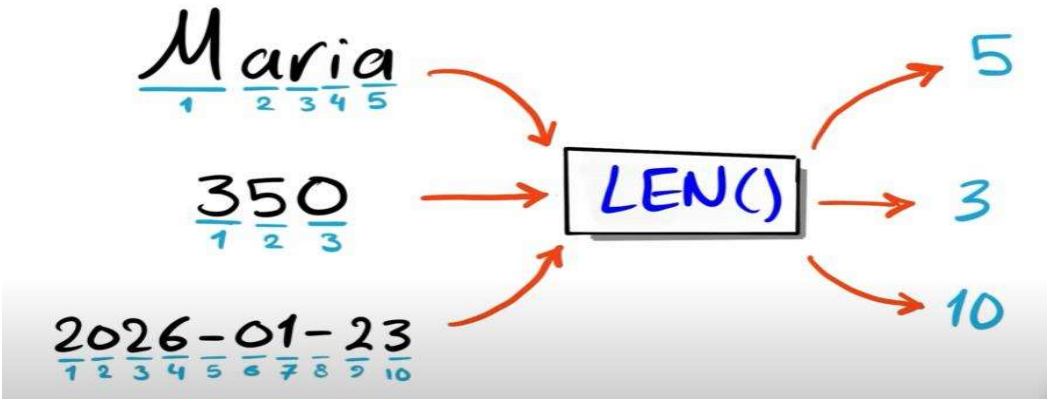
```sql
-- Replace File Extence from txt to csv

SELECT
'report.txt' AS old_filename,
REPLACE('report.txt', '.txt', '.csv') AS new_filename
```

Results Messages

	old_filename	new_filename
1	report.txt	report.csv

CONCAT	Combines multiple strings into one
UPPER	Converts all characters to uppercase
LOWER	Converts all characters to lowercase
TRIM	Removes Leading and Trailing spaces
REPLACE	Replaces specific character with a new character
LEN	Counts how many characters



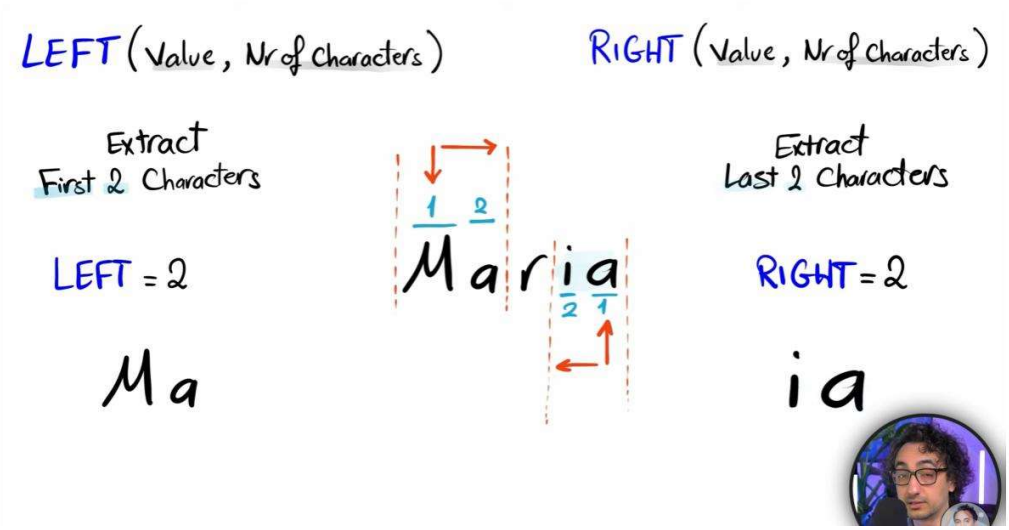

```sql
-- Calculate the length of each customer's first name.

SELECT
first_name,
LEN(first_name) AS len_name
FROM customers
```

Results Messages

	first_name	len_name
1	Maria	5
2	John	5
3	Georg	5
4	Martin	6
5	Peter	5

CONCAT	Combines multiple strings into one
UPPER	Converts all characters to uppercase
LOWER	Converts all characters to lowercase
TRIM	Removes Leading and Trailing spaces
REPLACE	Replaces specific character with a new character
LEN	Counts how many characters
LEFT	Extracts specific Number of Characters from the start
RIGHT	Extracts specific Number of Characters from the End



John has only J because it has leading space

```sql
-- Retrieve the first two characters of each first name.

SELECT
    first_name,
    LEFT(first_name, 2) first_2_char
FROM customers
```

Results | Messages

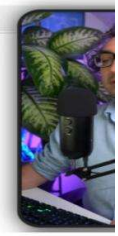
	first_name	first_2_char
1	Maria	Ma
2	John	J
3	Georg	Ge
4	Martin	Ma
5	Peter	Pe

```sql
-- Retrieve the first two characters of each first name.

SELECT
    first_name,
    LEFT(TRIM(first_name), 2) first_2_char
FROM customers
```

Results | Messages

	first_name	first_2_char
1	Maria	Ma
2	John	Jo
3	Georg	Ge
4	Martin	Ma
5	Peter	Pe



```sql
-- Retrieve the last two characters of each first name.
SELECT
    first_name,
    LEFT(TRIM(first_name), 2) first_2_char,
    RIGHT(first_name, 2) last_2_char
FROM customers
```

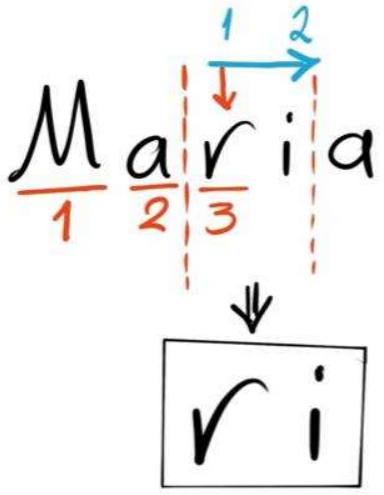
Results | Messages

	first_name	first_2_char	last_2_char
1	Maria	Ma	ia
2	John	Jo	hn
3	Georg	Ge	rg
4	Martin	Ma	in
5	Peter	Pe	er

Substring Extracts a part of string at a specified position

SUBSTRING (Value, Start, Length)

After the 2nd Character extract 2 Characters

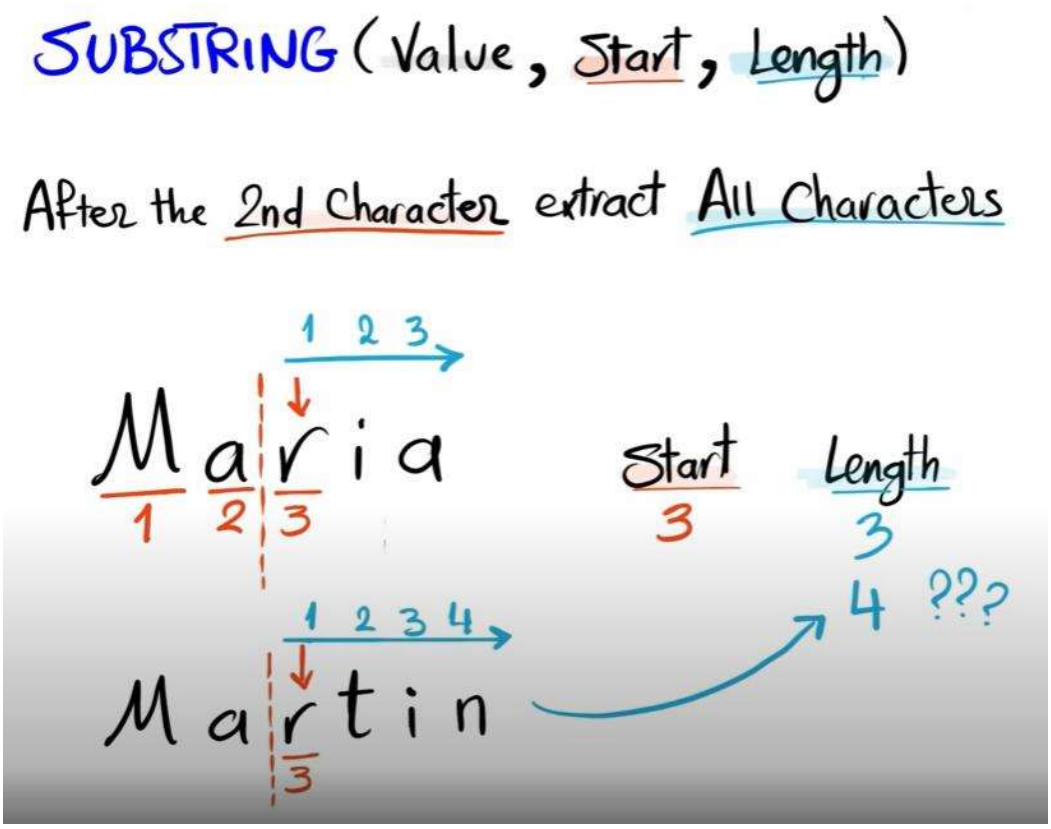


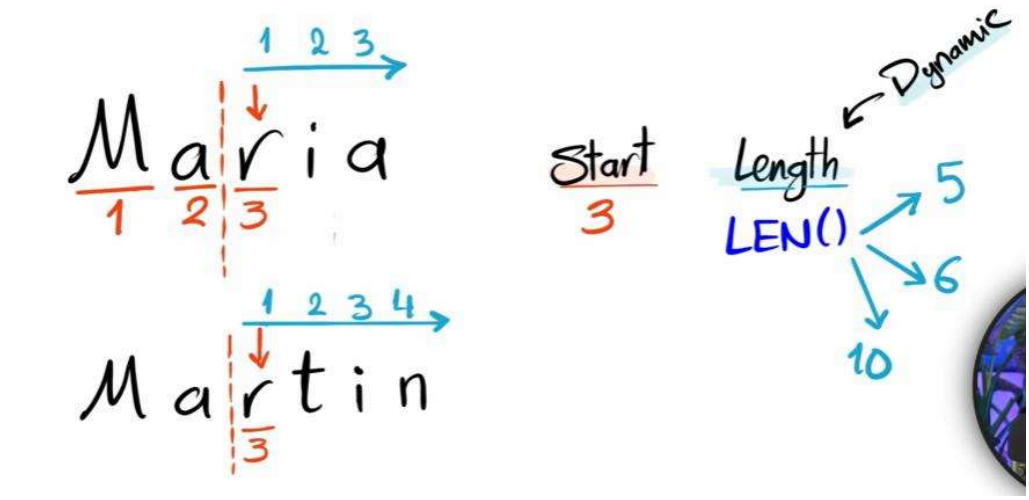
Start 3

Length 2

r i

We used len function to get exact number of len of value we did not used static length number
We used dynamic length.





Issue of using static length value we want

```sql
-- Retrieve a list of customers' first names after removing the first character.
SELECT
    first_name,
    SUBSTRING(first_name, 2, 4) AS sub_name
FROM customers
```

Results Messages

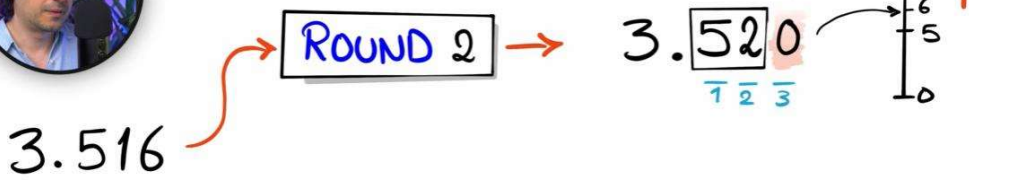
	first_name	sub_name
1	Maria	aria
2	John	John
3	Georg	eorg
4	Martin	arti
5	Peter	eter

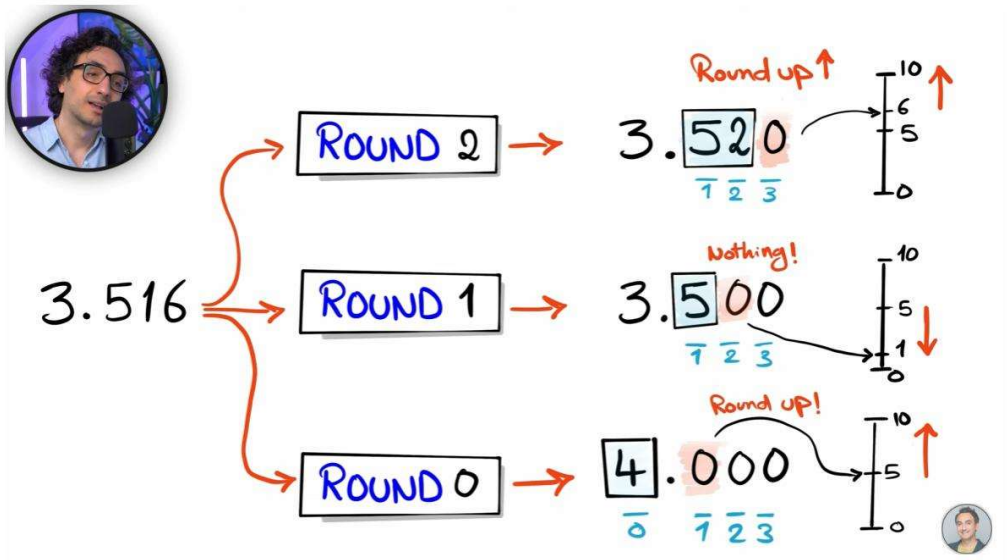
```sql
-- Retrieve a list of customers' first names after removing the first character.

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

Results | Messages

	first_name	sub_name
1	Maria	aria
2	John	ohn
3	Georg	eorg
4	Martin	artin
5	Pete	eter






```
SELECT
3.516,
ROUND(3.516, 2) AS round_2
```

Results | Messages

	(No column name)	round_2
	3.516	3.520

```
SELECT
3.516,
ROUND(3.516, 2) AS round_2,
ROUND(3.516, 1) AS round_1
```

Results | Messages

	(No column name)	round_2	round_1
1	3.516	3.520	3.500

```
SELECT
3.516,
ROUND(3.516, 2) AS round_2,
ROUND(3.516, 1) AS round_1,
ROUND(3.516, 0) AS round_0
```

Results | Messages

	(No column name)	round_2	round_1	round_0
1	3.516	3.520	3.500	4.000

ABS Absolute Function:
Make negative number to positive

ABS

returns the absolute (positive) value of a number, removing any negative sign

	(No column name)	round_2	round_1	round_0
1	3.516	3.520	3.500	4.000

```
SELECT
-10,
ABS(-10)
```

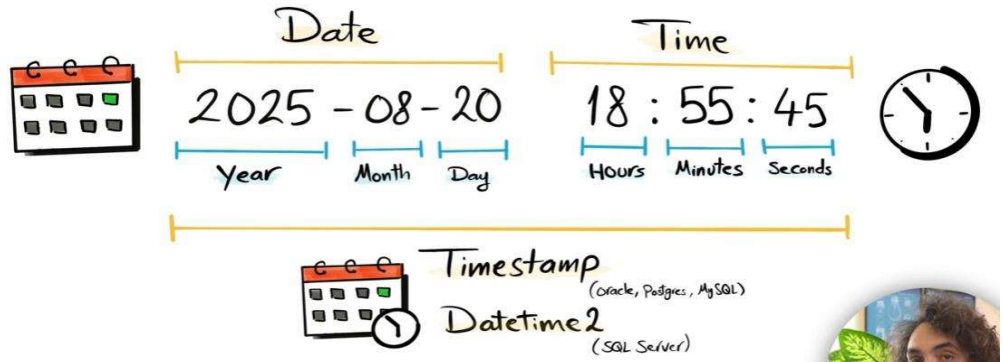
Results | Messages

	(No column name)	(No column name)
1	-10	10

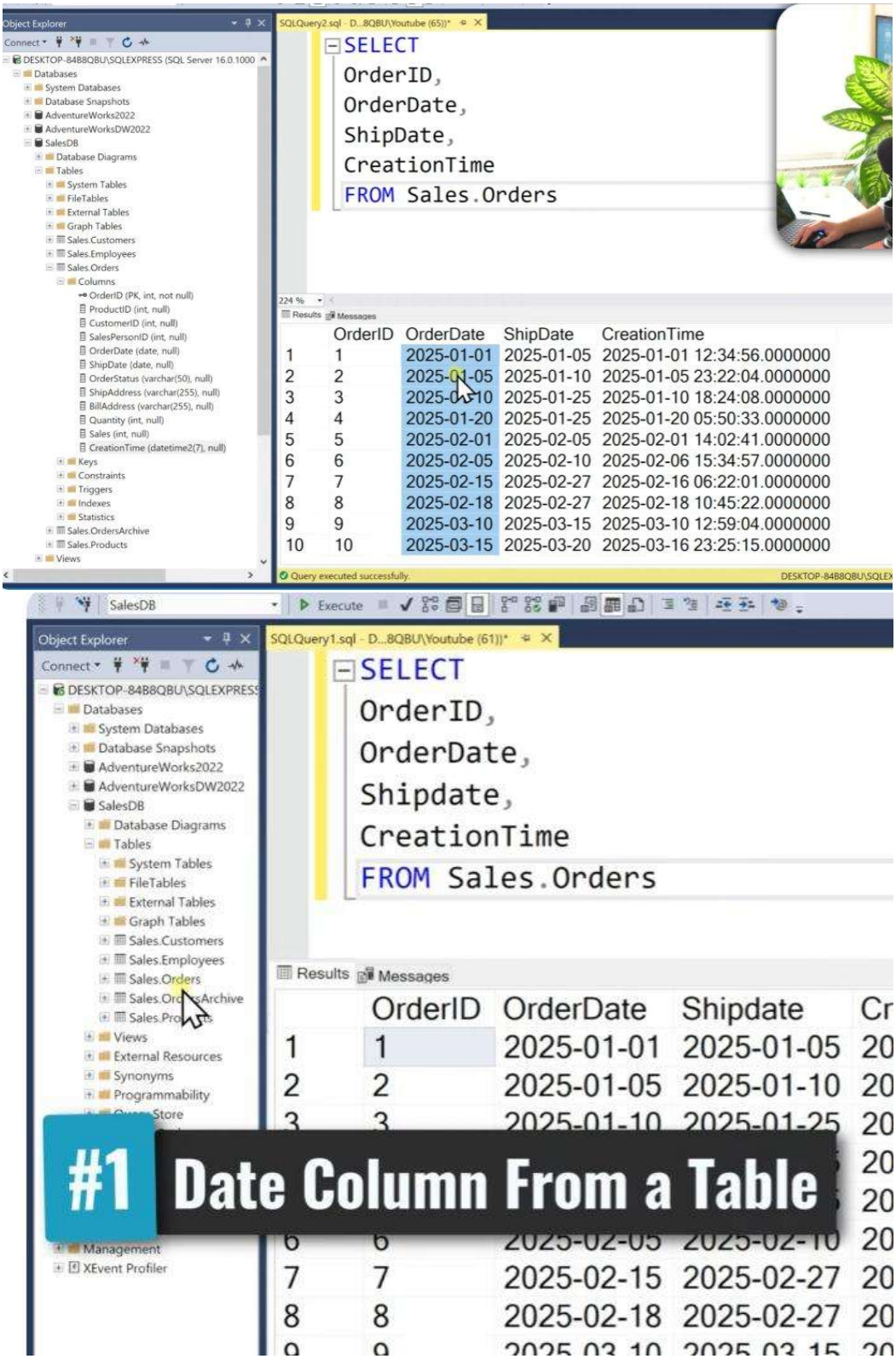
```
SELECT
-10,
ABS(-10),
ABS(10)
```

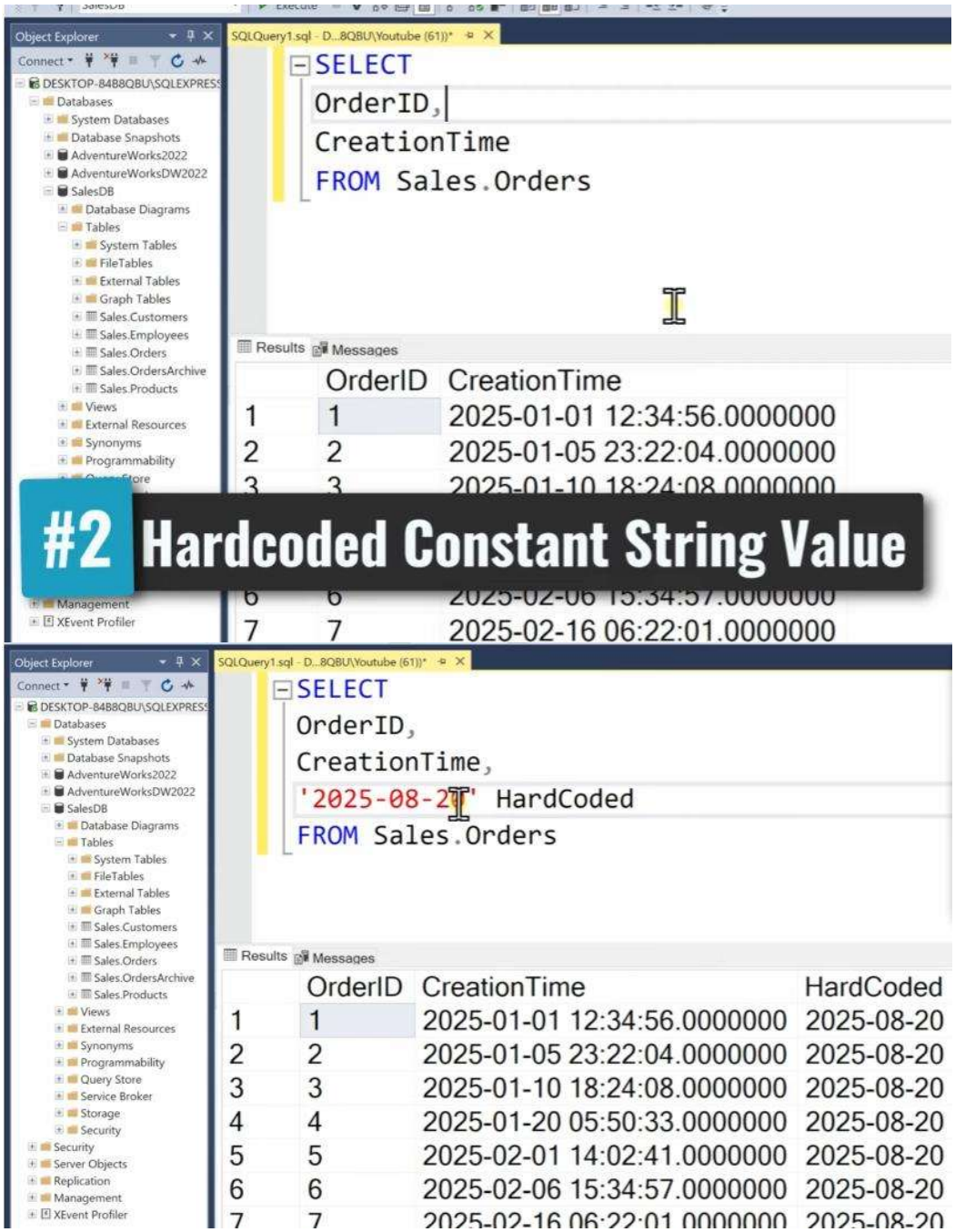
Results | Messages

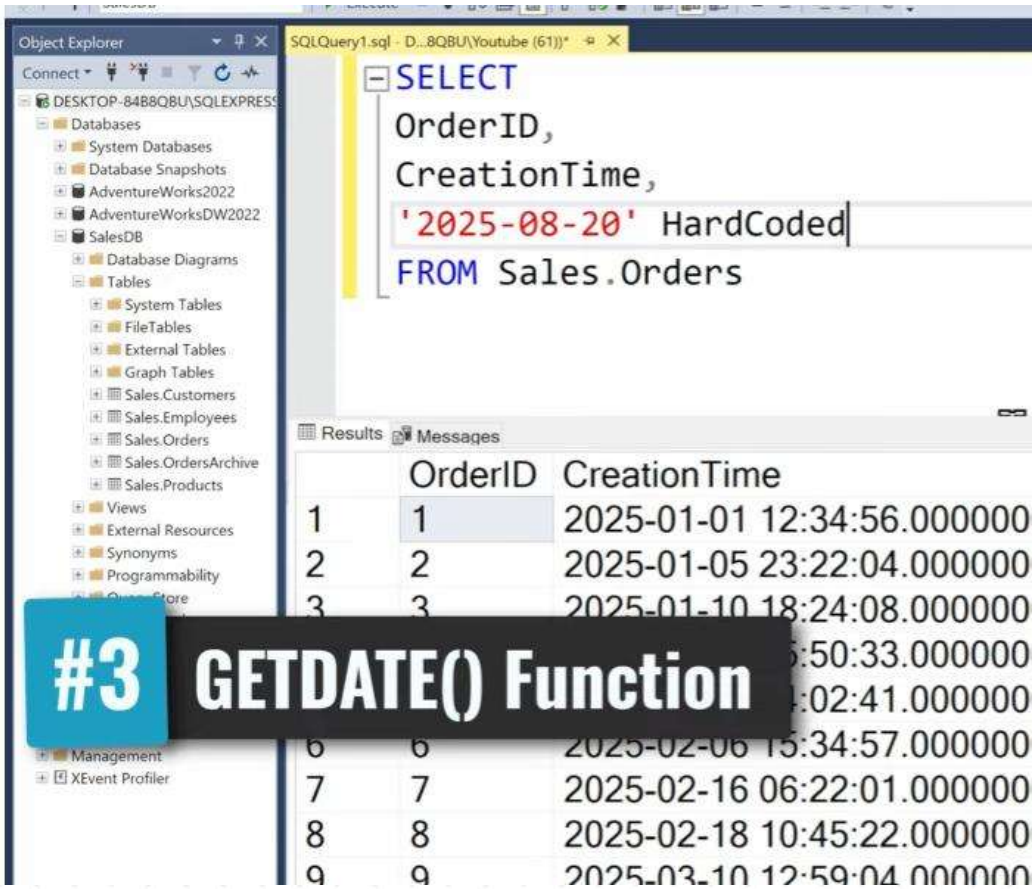
	(No column name)	(No column name)	(No column nam
1	-10	10	10


Date
Time
2025 - 08 - 20
18 : 55 : 45
Year
Month
Day
Hours
Minutes
Seconds
Timestamp
(oracle, Postgres, MySQL)
Datetime2
(SQL Server)


Graph Tables
Sales.Customers
Sales.Employees
Sales.Orders
Columns
OrderID (PK, int, not null)
ProductID (int, null)
CustomerID (int, null)
SalesPersonID (int, null)
OrderDate (date, null)
ShipDate (date, null)
OrderStatus (varchar(50), null)
ShipAddress (varchar(255), null)
BillAddress (varchar(255), null)
Quantity (int, null)
Sales (int, null)
CreationTime (datetime2(7), null)
Keys
Constraints

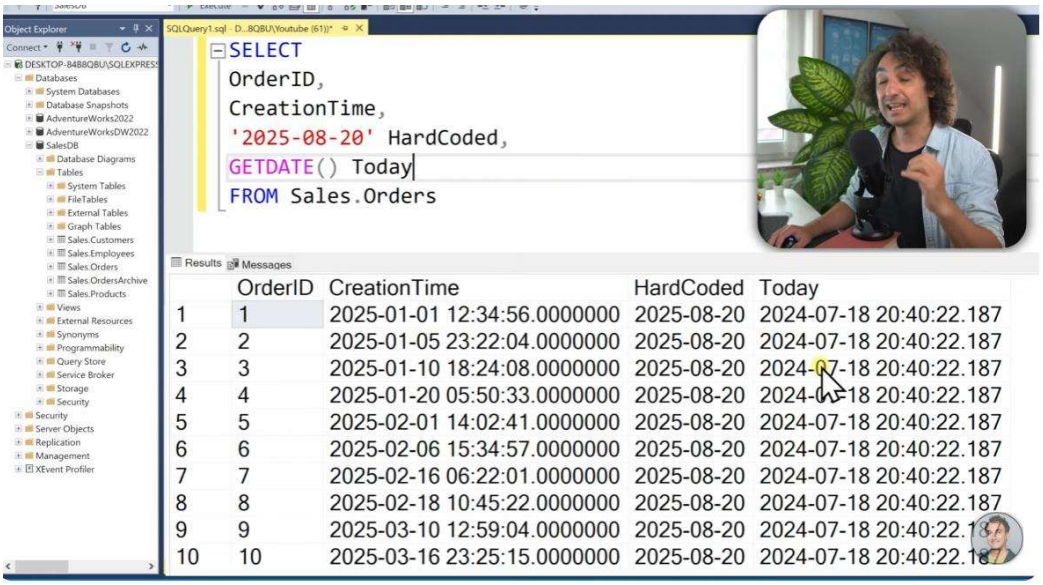

Object Explorer
Connect
DESKTOP-84B8QBU\SQLEXPRESS (SQL Server 16.0.1000
Databases
System Databases
Database Snapshots
AdventureWorks2022
AdventureWorksDW2022
SalesDB
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
Sales.Customers
Sales.Employees
Sales.Orders
Columns
OrderID (PK, int, not null)
ProductID (int, null)
CustomerID (int, null)
SalesPersonID (int, null)
OrderDate (date, null)
ShipDate (date, null)
OrderStatus (varchar(50), null)
ShipAddress (varchar(255), null)
BillAddress (varchar(255), null)
Quantity (int, null)
Sales (int, null)
CreationTime (datetime2(7), null)
Keys
Constraints
Triggers
Indexes
Statistics
Sales.OrdersArchive
Sales.Products
Views
SQLQuery2.sql - D...8QBU\Youtube (65))*
SELECT
OrderID,
OrderDate,
ShipDate,
CreationTime
FROM Sales.Orders
224 %
Results
Messages
OrderID OrderDate ShipDate CreationTime
1 1 2025-01-01 2025-01-05 2025-01-01 12:34:56.0000000
2 2 2025-01-05 2025-01-10 2025-01-05 23:22:04.0000000
3 3 2025-01-10 2025-01-25 2025-01-10 18:24:08.0000000
4 4 2025-01-20 2025-01-25 2025-01-20 05:50:33.0000000
5 5 2025-02-01 2025-02-05 2025-02-01 14:02:41.0000000
6 6 2025-02-05 2025-02-10 2025-02-06 15:34:57.0000000
7 7 2025-02-15 2025-02-27 2025-02-16 06:22:01.0000000
8 8 2025-02-18 2025-02-27 2025-02-18 10:45:22.0000000
9 9 2025-03-10 2025-03-15 2025-03-10 12:59:04.0000000
10 10 2025-03-15 2025-03-20 2025-03-16 23:25:15.0000000
Query executed successfully.
DESKTOP-84B8QBU\SQLEX
SalesDB
Execute
Object Explorer
Connect
DESKTOP-84B8QBU\SQLEXPRESS
Databases
System Databases
Database Snapshots
AdventureWorks2022
AdventureWorksDW2022
SalesDB
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
Sales.Customers
Sales.Employees
Sales.Orders
Sales.OrdersArchive
Sales.Products
Views
External Resources
Synonyms
Programmability
Management
XEvent Profiler
SQLQuery1.sql - D...8QBU\Youtube (61))*
SELECT
OrderID,
OrderDate,
Shipdate,
CreationTime
FROM Sales.Orders
Results
Messages
OrderID OrderDate Shipdate Cr
1 1 2025-01-01 2025-01-05 20
2 2 2025-01-05 2025-01-10 20
3 3 2025-01-10 2025-01-25 20
7 7 2025-02-15 2025-02-27 20
8 8 2025-02-18 2025-02-27 20
#1 Date Column From a Table


#2 Hardcoded Constant String Value
Object Explorer
Connect
DESKTOP-84B8QBU\SQLEXPRESS
Databases
System Databases
Database Snapshots
AdventureWorks2022
AdventureWorksDW2022
SalesDB
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
Sales.Customers
Sales.Employees
Sales.Orders
Sales.OrdersArchive
Sales.Products
Views
External Resources
Synonyms
Programmability
Management
XEvent Profiler
SQLQuery1.sql - D...8QBU\Youtube (61))*
SELECT
OrderID,
CreationTime
FROM Sales.Orders
Results
Messages
OrderID
CreationTime
1 1 2025-01-01 12:34:56.0000000
2 2 2025-01-05 23:22:04.0000000
7 7 2025-02-16 06:22:01.0000000
SELECT
OrderID,
CreationTime,
'2025-08-20' HardCoded
FROM Sales.Orders
HardCoded
1 1 2025-01-01 12:34:56.0000000 2025-08-20
2 2 2025-01-05 23:22:04.0000000 2025-08-20
3 3 2025-01-10 18:24:08.0000000 2025-08-20
4 4 2025-01-20 05:50:33.0000000 2025-08-20
5 5 2025-02-01 14:02:41.0000000 2025-08-20
6 6 2025-02-06 15:34:57.0000000 2025-08-20

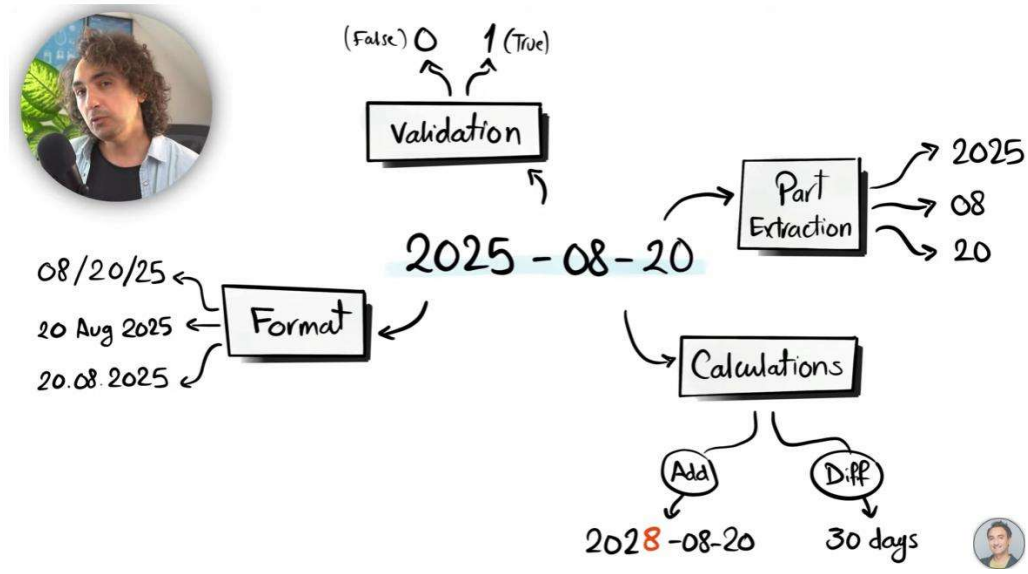


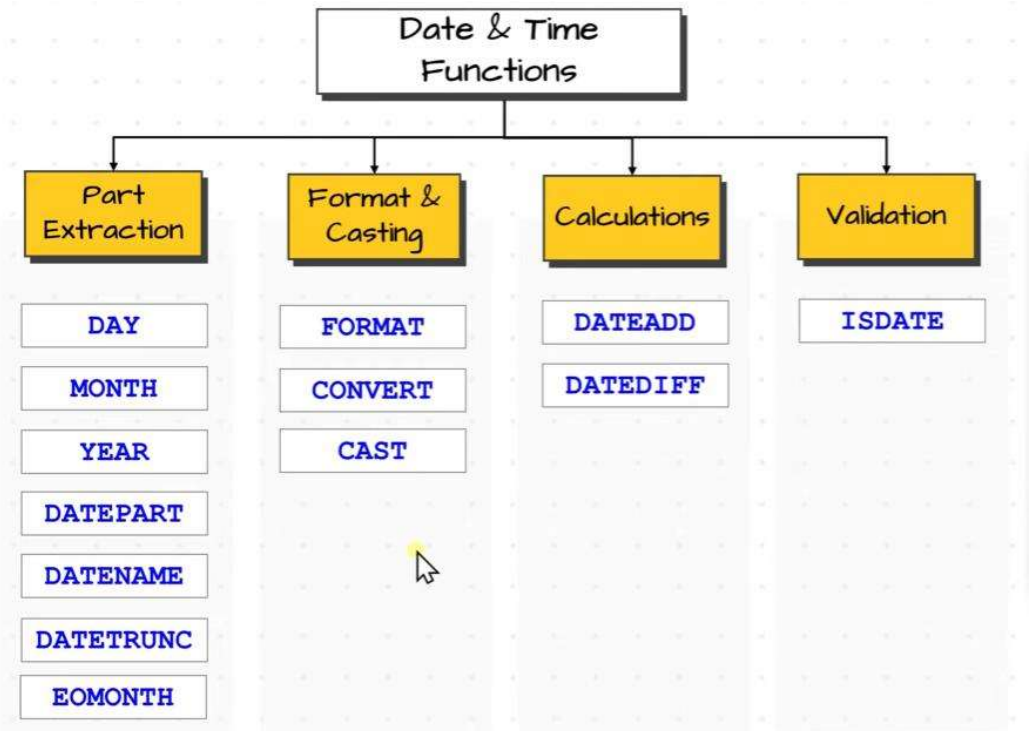
GETDATE ()

Returns the current date and time
at the moment when the query is executed.



How to manipulate date time information





DAY ( )

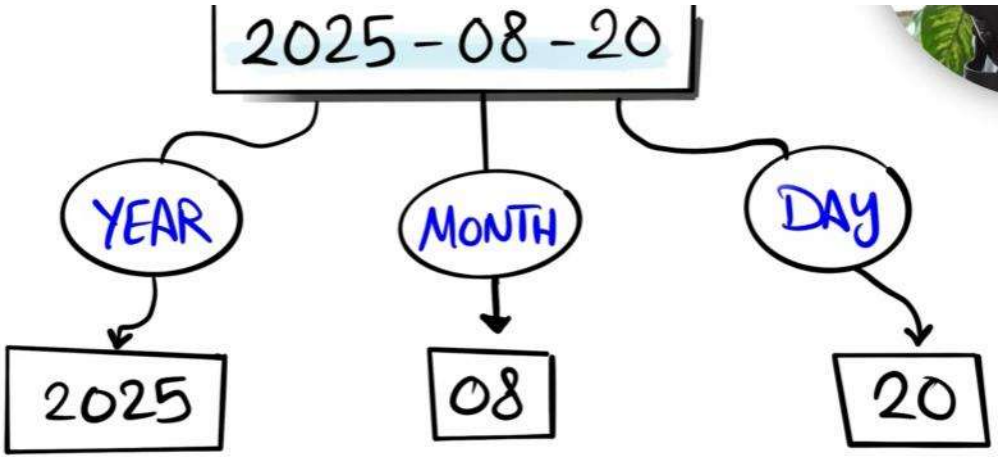
returns the day from a date

MONTH ( )

returns the month from a date

YEAR ( )

returns the year from a date

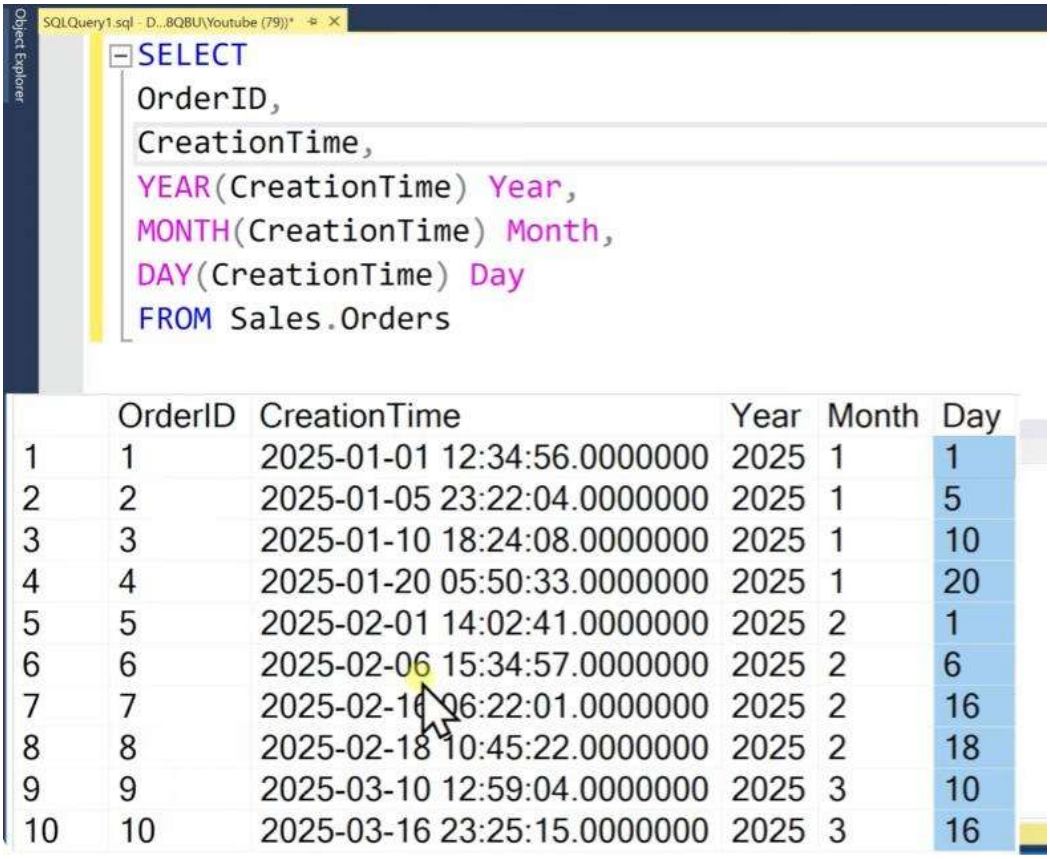


Syntax

```
DAY(date)
```

```
MONTH(date)
```

```
YEAR(date)
```

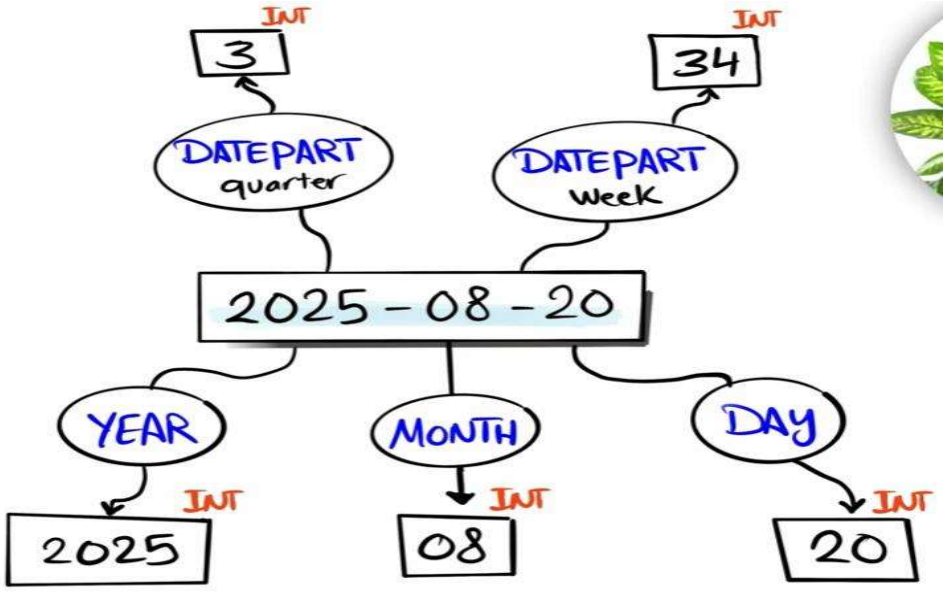


	OrderID	CreationTime	Year	Month	Day
1	1	2025-01-01 12:34:56.0000000	2025	1	1
2	2	2025-01-05 23:22:04.0000000	2025	1	5
3	3	2025-01-10 18:24:08.0000000	2025	1	10
4	4	2025-01-20 05:50:33.0000000	2025	1	20
5	5	2025-02-01 14:02:41.0000000	2025	2	1
6	6	2025-02-06 15:34:57.0000000	2025	2	6
7	7	2025-02-16 06:22:01.0000000	2025	2	16
8	8	2025-02-18 10:45:22.0000000	2025	2	18
9	9	2025-03-10 12:59:04.0000000	2025	3	10
10	10	2025-03-16 23:25:15.0000000	2025	3	16

Datepart Function -
Returns specific part of date or a number

DATEPART ( )

Returns a specific part of a date as a number.



Syntax

```
DATEPART(part, date)
```

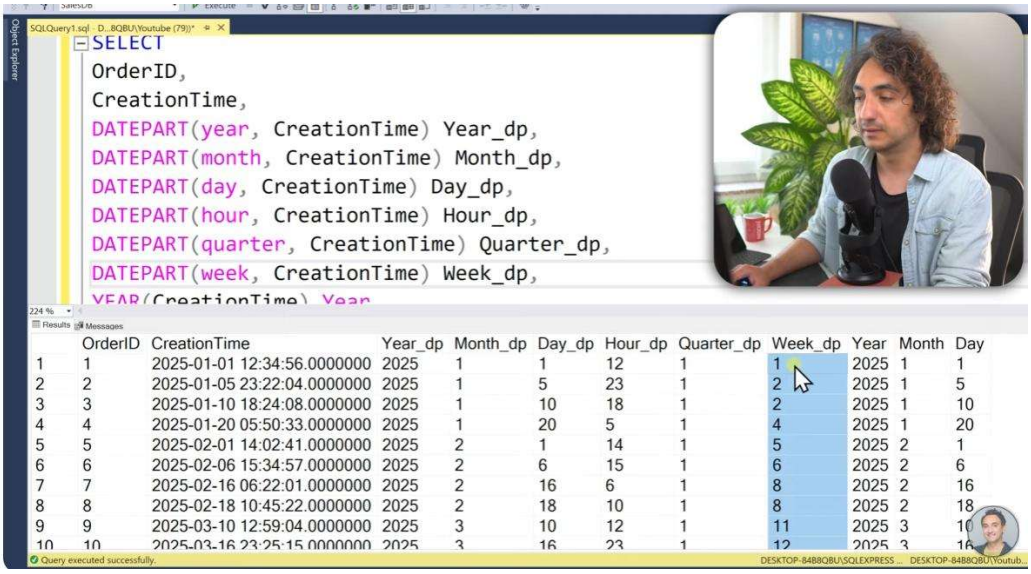
Examples

```
DATEPART(month, OrderDate)
```

Part

```
DATEPART(mm , '2025-08-20')
```

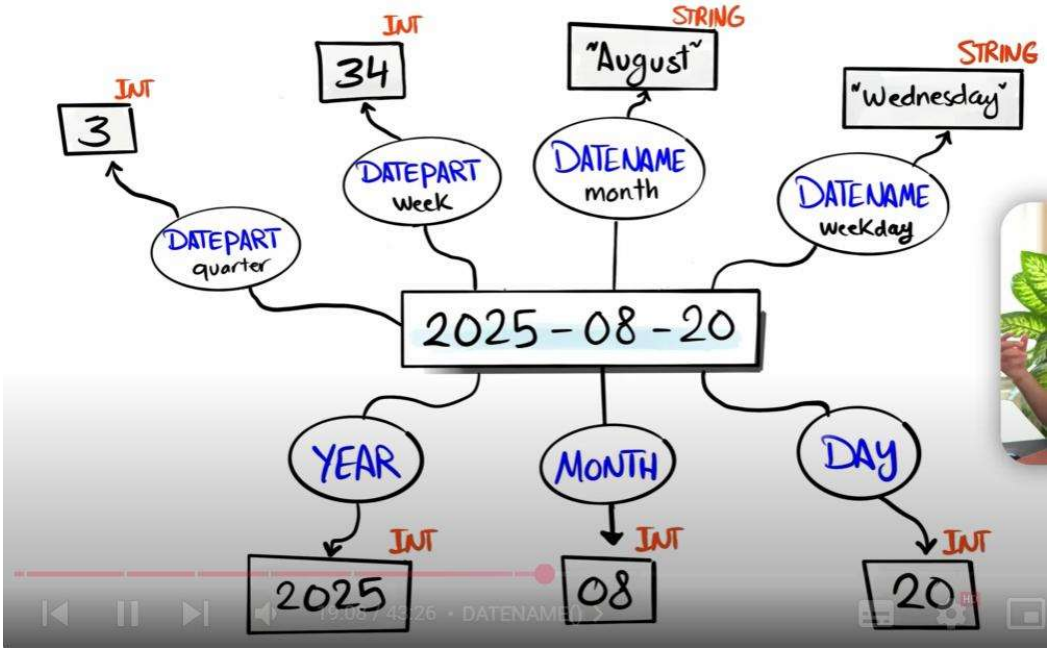
Abbreviation of Part



	OrderID	CreationTime	Year_dp	Month_dp	Day_dp	Hour_dp	Quarter_dp	Week_dp	Year	Month	Day
1	1	2025-01-01 12:34:56.0000000	2025	1	1	12	1	1	2025	1	1
2	2	2025-01-05 23:22:04.0000000	2025	1	5	23	1	2	2025	1	5
3	3	2025-01-10 18:24:08.0000000	2025	1	10	18	1	2	2025	1	10
4	4	2025-01-20 05:50:33.0000000	2025	1	20	5	1	4	2025	1	20
5	5	2025-02-01 14:02:41.0000000	2025	2	1	14	1	5	2025	2	1
6	6	2025-02-06 15:34:57.0000000	2025	2	6	15	1	6	2025	2	6
7	7	2025-02-16 06:22:01.0000000	2025	2	16	6	1	8	2025	2	16
8	8	2025-02-18 10:45:22.0000000	2025	2	18	10	1	8	2025	2	18
9	9	2025-03-10 12:59:04.0000000	2025	3	10	12	1	11	2025	3	10
10	10	2025-03-16 23:25:15.0000000	2025	3	16	23	1	12	2025	3	16

DATENAME ( )

Returns the name of a specific part of a date.

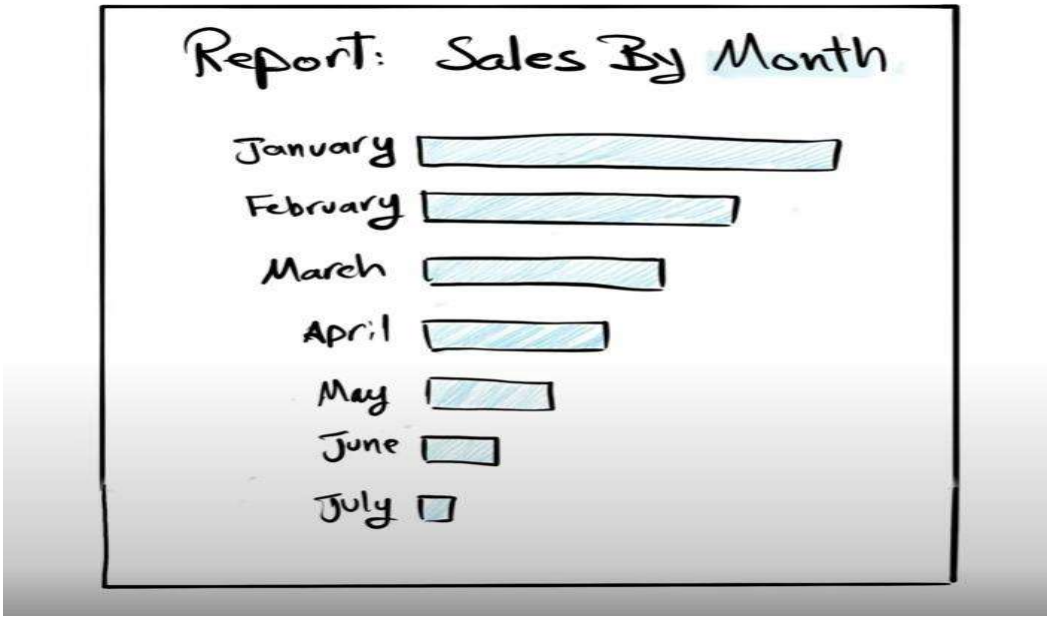


Date and Year is in the form of String data Type here

```sql
SELECT
OrderID,
CreationTime,
DATENAME(month, CreationTime) Month_dn,
DATENAME(weekday, CreationTime) weekday_dn,
DATENAME(day, CreationTime) Day_dn,
DATENAME(year, CreationTime) Year_dn,
-- DATEPART Examples
DATEPART(year, CreationTime) Year_dp,
DATEPART(month, CreationTime) Month_dp,
DATEPART(day, CreationTime) Day_dp,
```

	OrderID	CreationTime	Month_dn	weekday_dn	Day_dn	Year_dn	Year_dp	Month_dp	Day_dp	Hour_dp	Q
1	1	2025-01-01 12:34:56.0000000	January	Wednesday	1	2025	2025	1	1	12	1
2	2	2025-01-05 23:22:04.0000000	January	Sunday	5	2025	2025	1	5	23	1
3	3	2025-01-10 18:24:08.0000000	January	Friday	10	2025	2025	1	10	18	1
4	4	2025-01-20 05:50:33.0000000	January	Monday	20	2025	2025	1	20	5	1
5	5	2025-02-01 14:02:41.0000000	February	Saturday	1	2025	2025	2	1	14	1
6	6	2025-02-06 15:34:57.0000000	February	Thursday	6	2025	2025	2	6	15	1
7	7	2025-02-16 06:22:01.0000000	February	Sunday	16	2025	2025	2	16	6	1
8	8	2025-02-18 10:45:22.0000000	February	Tuesday	18	2025	2025	2	18	10	1
9	9	2025-03-10 12:59:04.0000000	March	Monday	10	2025	2025	3	10	12	1
10	10	2025-03-16 23:25:15.0000000	March	Sunday	16	2025	2025	3	16	23	1

Usecase of datename



DATETRUNC ( )

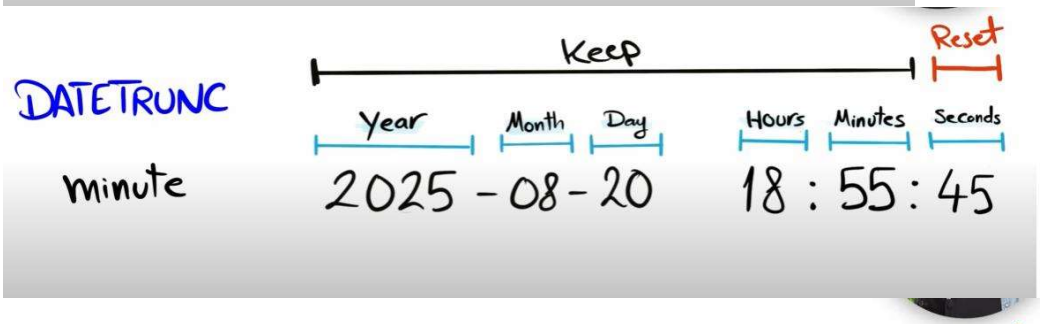
Truncates the date to the specific part.

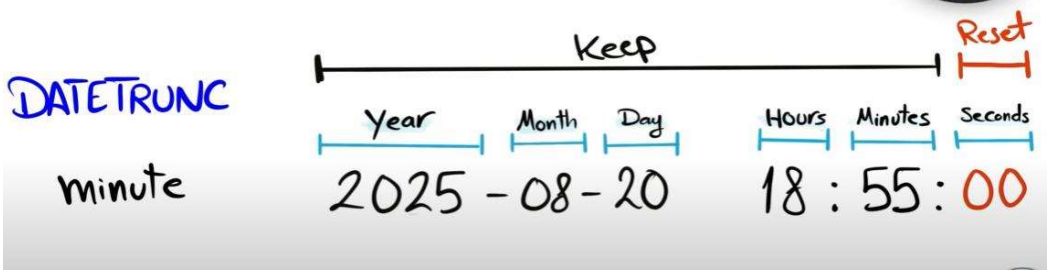
Syntax

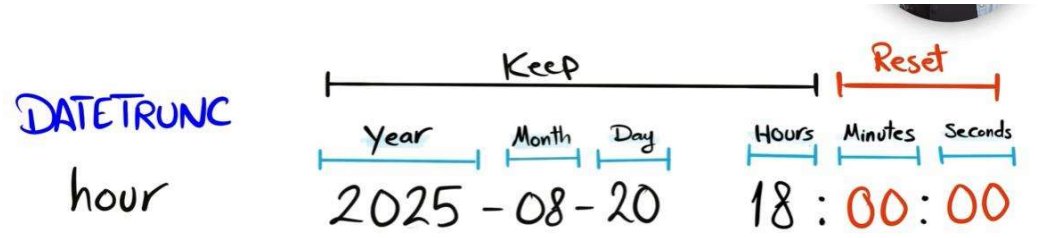
```
DATEPART (part, date)
```

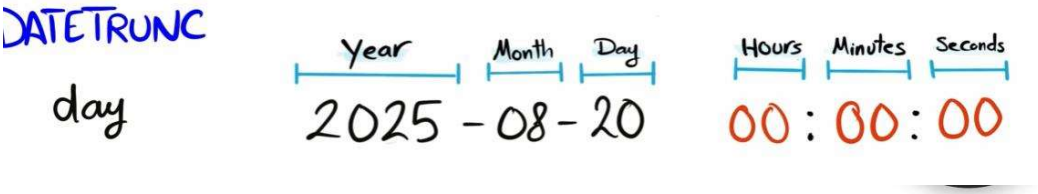
```
DATENAME (part, date)
```

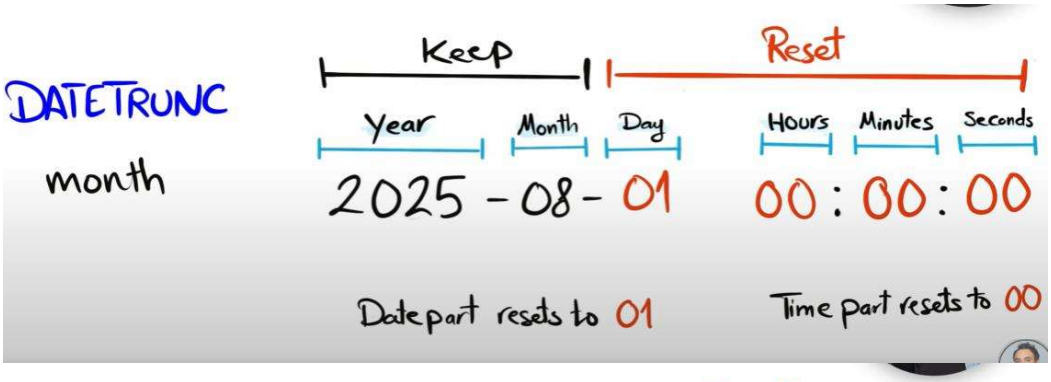
```
DATETRUNC (part, date)
```

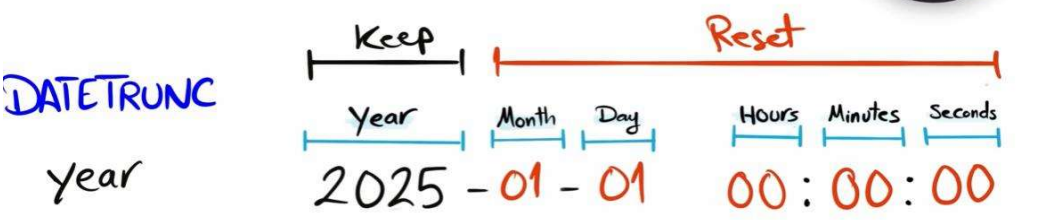


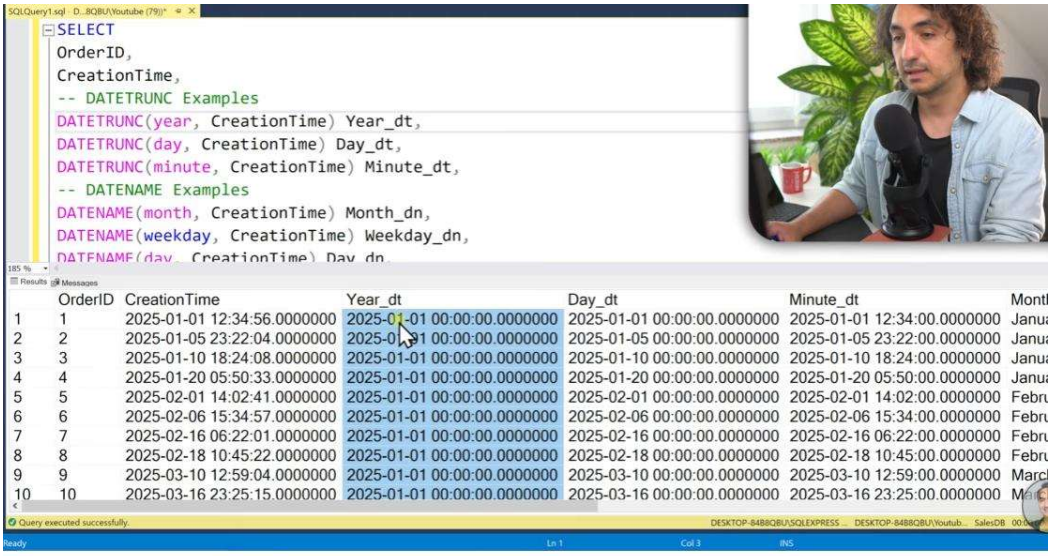



DATETRUNC
hour
Keep
Reset
Year
Month
Day
Hours
Minutes
Seconds
2025 - 08 - 20
18 : 00 : 00


DATETRUNC
day
Year
Month
Day
Hours
Minutes
Seconds
2025 - 08 - 20
00 : 00 : 00


DATETRUNC
month
Keep
Reset
Year
Month
Day
Hours
Minutes
Seconds
2025 - 08 - 01
00 : 00 : 00
Datepart resets to 01
Time part resets to 00


DATETRUNC
year
Keep
Reset
Year
Month
Day
Hours
Minutes
Seconds
2025 - 01 - 01
00 : 00 : 00


SELECT
OrderID,
CreationTime,
-- DATETRUNC Examples
DATETRUNC(year, CreationTime) Year_dt,
DATETRUNC(day, CreationTime) Day_dt,
DATETRUNC(minute, CreationTime) Minute_dt,
-- DATENAME Examples
DATENAME(month, CreationTime) Month_dn,
DATENAME(weekday, CreationTime) Weekday_dn,
OrderID CreationTime Year_dt Day_dt Minute_dt
1 1 2025-01-01 12:34:56.0000000 2025-01-01 00:00:00.0000000 2025-01-01 00:00:00.0000000 2025-01-01 12:34:00.0000000
2 2 2025-01-05 23:22:04.0000000 2025-01-01 00:00:00.0000000 2025-01-05 00:00:00.0000000 2025-01-05 23:22:00.0000000
3 3 2025-01-10 18:24:08.0000000 2025-01-01 00:00:00.0000000 2025-01-10 00:00:00.0000000 2025-01-10 18:24:00.0000000
4 4 2025-01-20 05:50:33.0000000 2025-01-01 00:00:00.0000000 2025-01-20 00:00:00.0000000 2025-01-20 05:50:00.0000000
5 5 2025-02-01 14:02:41.0000000 2025-01-01 00:00:00.0000000 2025-02-01 00:00:00.0000000 2025-02-01 14:02:00.0000000
6 6 2025-02-06 15:34:57.0000000 2025-01-01 00:00:00.0000000 2025-02-06 00:00:00.0000000 2025-02-06 15:34:00.0000000
7 7 2025-02-16 06:22:01.0000000 2025-01-01 00:00:00.0000000 2025-02-16 00:00:00.0000000 2025-02-16 06:22:00.0000000
8 8 2025-02-18 10:45:22.0000000 2025-01-01 00:00:00.0000000 2025-02-18 00:00:00.0000000 2025-02-18 10:45:00.0000000
9 9 2025-03-10 12:59:04.0000000 2025-01-01 00:00:00.0000000 2025-03-10 00:00:00.0000000 2025-03-10 12:59:00.0000000
10 10 2025-03-16 23:25:15.0000000 2025-01-01 00:00:00.0000000 2025-03-16 00:00:00.0000000 2025-03-16 23:25:00.0000000
Query executed successfully.