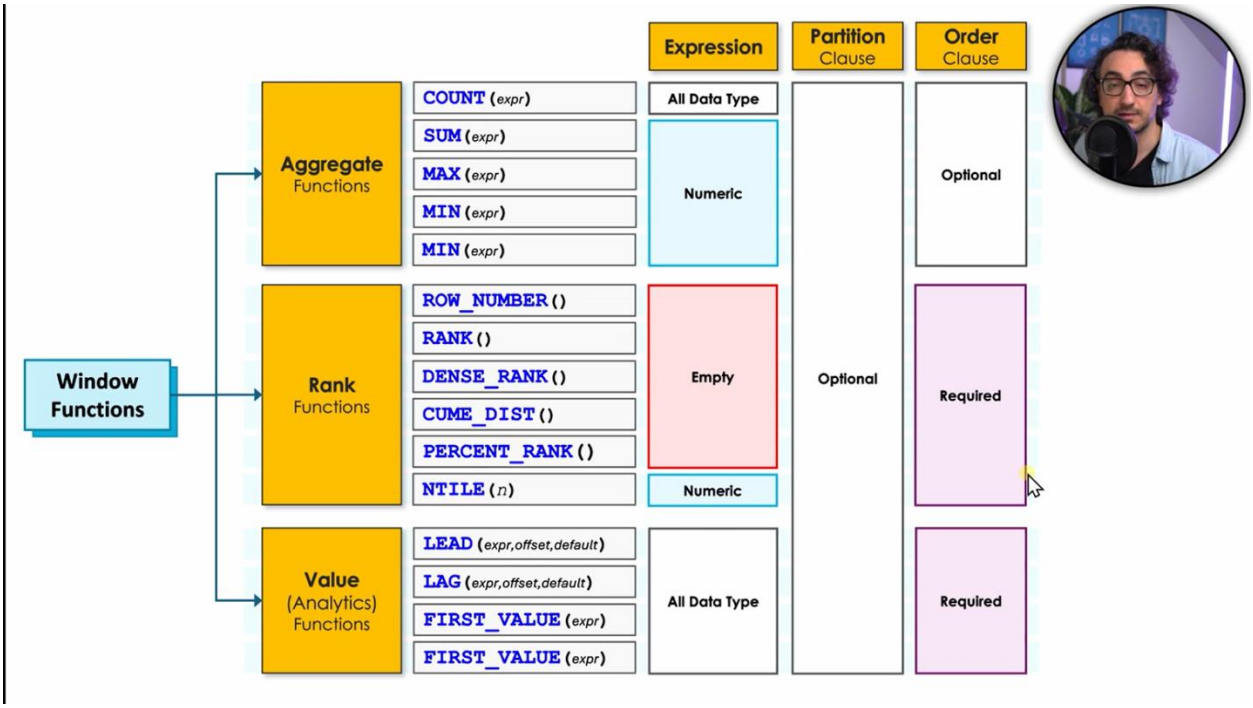
Window Functions

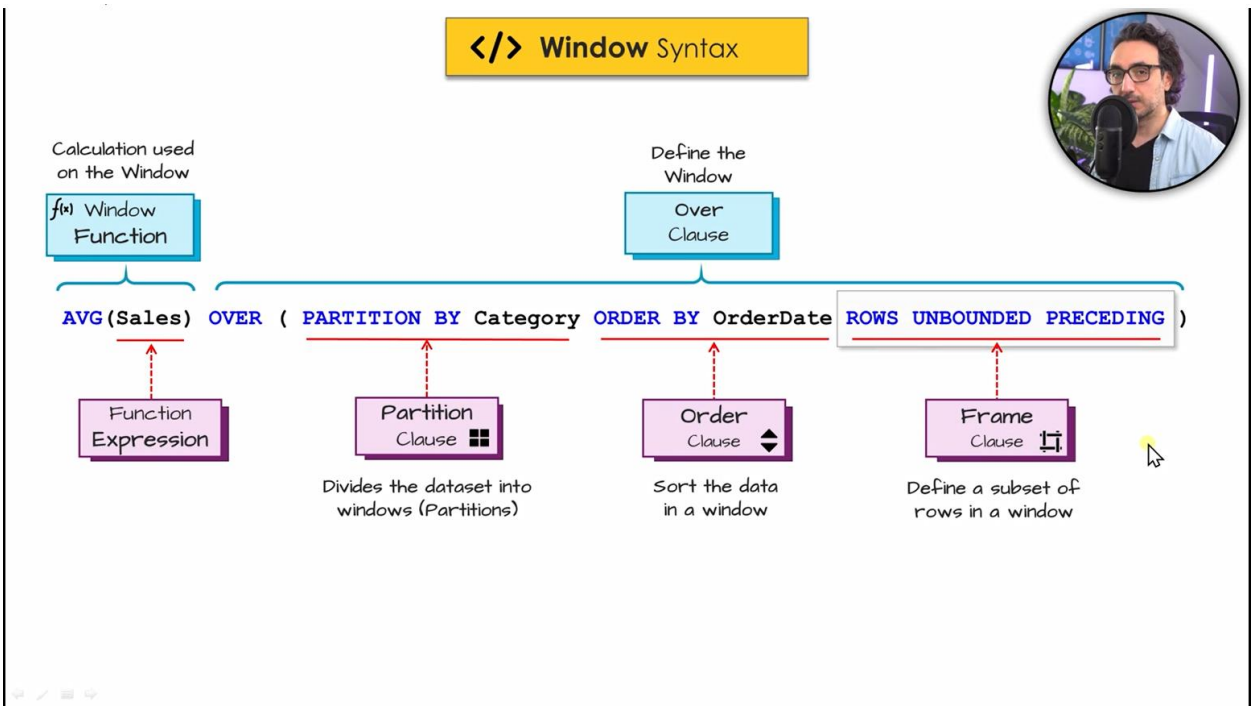
```
/*
      ======================
      == Window Functions ==
      ======================
      - perform calculations like aggregation on a specific subset of data without
losing the level of details of rows
      - Returns all original rows and adds extra columns with aggregated/calculated
data
      - we can use aggregate functions, Rank functions, value (analytics) functions


      ==============
      == GROUP BY ==
      ==============
      - Aggregate rows into groups based on one or more columns
      - returns one row per group
      - we can use aggregate functions


      -----------------------------
      -- window functions syntax --
      -----------------------------
      SELECT
    column1,
    column2,
    window_function_name(expression)
    OVER (
        [PARTITION BY column]
        [ORDER BY column]
        [ROWS/RANGE frame]
    ) AS new_column
      FROM table_name;

      ----------------------------
      -- window_function_name() --
      ----------------------------
      This is the function you want to use, like:
            ROW_NUMBER()
            RANK()
            DENSE_RANK()
            SUM()
            AVG()
            COUNT()
            LAG()
            LEAD()

      ------------------------------------------
      -- Frame Clause (optional advanced part) --
      ------------------------------------------
      - Controls which rows are included in the calculation.

      ------------------------
      -- most common syntax --
      ------------------------
```

```
	SELECT
		column,
		SUM(column) OVER (PARTITION BY column ORDER BY column) AS running_total
	FROM table;


	-----------------------------
	-- Simplest Example Syntax --
	-----------------------------
	SELECT
		name,
		score,
		ROW_NUMBER() OVER (ORDER BY score DESC) AS row_num
	FROM customers;
*/

-------------------------------------------------------------------------------

USE SalesDB;


-- find total sales across all orders
SELECT
SUM(Sales) AS total_sales
FROM Sales.Orders;


-- find total sales for each product
SELECT
	ProductID,
	SUM(Sales) AS total_sales
FROM Sales.Orders
GROUP BY ProductID;


/*
	- OVER(): tells SQL that the function used is a window function. it defines a
window or subset of data
	- PARTITION BY: divide the result set into windows (partitions)
*/


-- find the 'total sales across all orders', additionally provide details such order
id, order date
SELECT
	OrderID,
	OrderDate,
	SUM(Sales) OVER() AS [total sales]
FROM Sales.Orders;


-- find the 'total sales for each product', additonally provide details such order
id & order date
SELECT
	ProductID,
	OrderID,
	OrderDate,
	SUM(Sales) OVER(PARTITION BY ProductID) AS TotalSales
```

```sql
FROM Sales.Orders
ORDER BY OrderID ASC;


-- find the total sales across all orders
-- find the total sales for each product
-- additionally provide details such as order id & order date
SELECT
	ProductID,
	OrderID,
	OrderDate,
	Sales,
	-- find the total sales across all orders
	SUM(Sales) OVER() AS [Total Sales],
	-- find the total sales for each product
	SUM(Sales) OVER(PARTITION BY ProductID) AS [Total Sales By Products]
FROM Sales.Orders;


-- fint the total sales for each combination of product and order status
SELECT
	ProductID,
	OrderID,
	OrderStatus,
	Sales,
	SUM(Sales) OVER(PARTITION BY ProductID, OrderStatus) [Sales by Products and
Status]
FROM Sales.Orders;

--------------------------------------------------------------------------------

-- rank each order based on their sales from the highest to lowest
-- additionally provide details such order id & order date
SELECT
	OrderID,
	OrderDate,
	Sales,
	RANK() OVER(ORDER BY Sales DESC) AS [RankSales]
FROM Sales.Orders;

--------------------------------------------------------------------------------

------------------
-- Window Frame --
------------------
-- define subset of rows with each window that is relevant for the calculation
-- Define which rows inside the partition are included in the calculation.
```

```sql
/*
      find running total sales across all orders, additionally provide order
details
*/
SELECT
    OrderID,
    OrderDate,
    Sales,
    SUM(Sales) OVER(
        ORDER BY OrderDate
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS RunningTotalSales
FROM Sales.Orders;

-----------------------------------------------------------------------------
```


Expression
Partition Clause
Order Clause
Window Functions
Aggregate Functions
COUNT (expr)
SUM (expr)
MAX (expr)
MIN (expr)
MIN (expr)
All Data Type
Numeric
Optional
Rank Functions
ROW_NUMBER ()
RANK ()
DENSE_RANK ()
CUME_DIST ()
PERCENT_RANK ()
NTILE (n)
Empty
Numeric
Optional
Required
Value (Analytics) Functions
LEAD (expr,offset,default)
LAG (expr,offset,default)
FIRST_VALUE (expr)
FIRST_VALUE (expr)
All Data Type
Required


Window Syntax
Calculation used on the Window
Window Function
Define the Window
Over Clause
AVG(Sales) OVER (PARTITION BY Category ORDER BY OrderDate ROWS UNBOUNDED PRECEDING)
Function Expression
Partition Clause
Divides the dataset into windows (Partitions)
Order Clause
Sort the data in a window
Frame Clause
Define a subset of rows in a window