Window Functions-II

Session Goals

- Understand what window functions are and when to use them.
- Break down and apply the syntax of common window functions like ROW_NUMBER(), SUM(), AVG(), etc.
- Differentiate window functions from regular aggregate functions.
- Use analytical window functions to extract advanced insights.
- Compare rows without self-joins.
- Partition and rank data meaningfully.
- Detect trends, group data into tiles, and calculate change over time.

CASE With Window Function

"ProductSubcategoryKey"

```
1. If the productCost = Max(ProductCost) -> "Highest"
2. If the productCost = MIN(ProductCost) -> "Lowest"
3. Else : "Middle"
```

"CostCategory"

ProductSubcategoryKey	ProductName	ProductCost	CostCategory
1	Mountain-100 Silver, 38	1912.1544	Highest
1	Mountain-100 Silver, 42	1912.1544	Highest
1	Mountain-100 Silver, 44	1912.1544	Highest
1	Mountain-100 Silver, 48	1912.1544	Highest
1	Mountain-100 Black, 38	1898.0944	Middle
1	Mountain-100 Black, 42	1898.0944	Middle
1	Mountain-100 Black, 44	1898.0944	Middle
1	Mountain-100 Black, 48	1898.0944	Middle
1	Mountain-200 Silver, 38	1117.8559	Middle
1	Mountain-200 Silver, 42	1117.8559	Middle
1	Mountain-200 Silver, 46	1117.8559	Middle
1	Mountain-200 Black, 38	1105.81	Middle

ProductSubcategoryKey	ProductName	ProductCost	CostCategory
1	Mountain-500 Silver, 44	308.2179	Middle
1	Mountain-500 Silver, 48	308.2179	Middle
1	Mountain-500 Silver, 52	308.2179	Middle
1	Mountain-500 Black, 40	294.5797	Lowest
1	Mountain-500 Black, 42	294.5797	Lowest
1	Mountain-500 Black, 44	294.5797	Lowest
1	Mountain-500 Black, 48	294.5797	Lowest
1	Mountain-500 Black, 52	294.5797	Lowest
2	Road-150 Red, 62	2171.2942	Highest
2	Road-150 Red, 44	2171.2942	Highest
2	Road-150 Red, 48	2171.2942	Highest
2	Road-150 Red, 52	2171.2942	Highest

```sql
SELECT
    ProductSubcategoryKey,
    ProductName,
    ProductCost,
    CASE
        WHEN ProductCost = MAX(ProductCost) OVER(PARTITION BY ProductSubcategoryKey) THEN 'Highest'
        WHEN ProductCost = MIN(ProductCost) OVER(PARTITION BY ProductSubcategoryKey) THEN 'Lowest'
        ELSE 'Middle'
    END AS CostCategory
FROM Products;
```

RANKING()

Within a Gender I want to the Ranking based on AnnualIncome.

Gender
M
F
NA

CustomerKey	Gender	FullName	AnnualIncome	income_rank
12645	F	AUDREY RUIZ	170000	1
12318	F	KRISTINA SCHMIDT	170000	1
11250	F	SHANNON LIU	170000	1
11244	F	ALEXIS COLEMAN	170000	1
12329	F	BONNIE SHAN	170000	1
12648	F	LORI DOMINGUEZ	170000	1
12647	F	COLLEEN ANAND	170000	1
12658	F	JOY GOMEZ	170000	1
12361	F	DANA DIAZ	160000	9
11180	F	APRIL ANAND	160000	9
11240	F	ANNE HERNANDEZ	160000	9
12654	F	BRIANA GILL	160000	9
11409	F	JACQUELINE HAYES	10000	951
12277	F	BETH ORTEGA	10000	951
12561	F	TRISHA LIN	10000	951
11130	F	CAROLINE RUSSELL	NULL	1019
11673	F	SUSAN YE	NULL	1019
11081	F	SAVANNAH BAKER	NULL	1019
11015	F	CHLOE YOUNG	NULL	1019
11003	F	CHRISTY ZHU	NULL	1019

CustomerKey	Gender	FullName	AnnualIncome	income_rank
11467	M	ARTURO ZHENG	170000	1
12190	M	JUSTIN GONZALES	170000	1
12134	M	HECTOR ALONSO	170000	1
12710	M	DALTON EVANS	170000	1
12326	M	GLENN ZHOU	170000	1
11422	M	DUSTIN DENG	170000	1
11286	M	HUNTER GRIFFIN	170000	1
12533	M	LUIS WASHINGTON	170000	1
11080	M	DAMIEN CHANDER	170000	1
11434	M	ANDRE LOPEZ	170000	1
12174	M	CHARLES PHILLIPS	170000	1
12123	M	WESLEY LIANG	170000	1

CustomerKey	Gender	FullName	AnnualIncome	income_rank
11227	M	MARSHALL CHAVEZ	NULL	1017
11082	NA	ANGELA BUTLER	130000	1
12300	NA	ADRIANA GONZALEZ	80000	2
11965	NA	KATIE SHE	70000	3
11234	NA	ANNA GRIFFIN	70000	3
12790	NA	BRANDON KUMAR	60000	5
12372	NA	DARREN PRASAD	60000	5
11643	NA	NAOMI MUNOZ	50000	7
11623	NA	ANGELA PERRY	50000	7
12446	NA	JASMINE HALL	40000	9
11526	NA	KATHERINE DIAZ	40000	9
12936	NA	RENEE MORENO	40000	9

```
SELECT
    CustomerKey,
    Gender,
    FullName,
    AnnualIncome,
    RANK() OVER (partition by Gender ORDER BY AnnualIncome DESC) AS income_rank
FROM Customers;
```

Rank customers within each MaritalStatus, and order by Birthdate(oldest to youngest).

```
DESC Customers;
SELECT
    CustomerKey,
    FullName,
    MaritalStatus,
    DateOfBirth,
    RANK() OVER (partition by MaritalStatus ORDER BY DateOfBirth) AS rank_by_birthdate
FROM Customers;
```

CustomerKey	FullName	MaritalStatus	DateOfBirth	rank_by_birthdate
12725	GABRIELLE JAMES	M	1910-08-13	1
12810	CHASE STEWART	M	1926-07-28	2
11555	ALEXANDRIA HENDERSON	M	1926-08-07	3
12823	BETHANY SHE	M	1927-11-17	4
12978	DANIELLE JAMES	M	1928-11-15	5
11503	DENNIS WU	M	1930-10-13	6
12412	MACKENZIE WRIGHT	M	1930-10-20	7
12758	STEPHANIE RAMIREZ	M	1930-11-01	8
11504	JORDAN BAKER	M	1931-04-20	9
12759	JADA SANDERS	M	1931-07-14	10
12137	JACK GREEN	M	1932-04-11	11
12136	BLAKE TURNER	M	1932-08-11	12

CustomerKey	FullName	MaritalStatus	DateOfBirth	rank_by_birthdate
12244	AMY HUANG	M	1980-11-22	1138
11131	AMANDA RIVERA	M	1980-12-03	1139
11845	NATALIE JONES	S	1924-08-18	1
11554	SYDNEY SIMMONS	S	1926-09-28	2
11251	XAVIER LONG	S	1932-04-07	3
11252	NICHOLAS THOMPSON	S	1932-07-06	4
11257	JACQUELINE POWELL	S	1933-06-01	5
11297	NOAH COLEMAN	S	1935-02-02	6
12011	MORGAN JOHNSON	S	1935-02-17	7
12539	MARC DOMINGUEZ	S	1935-05-26	8
11296	HALEY RICHARDSON	S	1935-07-10	9
11119	EVAN JAMES	S	1935-10-04	10

Rank the product and create a category with case statement
where the productcost is being ranked based on productsubcategoryKey.

= 1 Rank [Top Rank]
<= 5 Rank [Medium Rank]
Else [Bottom Rank]

```
SELECT
    ProductSubcategoryKey,
    ProductName,
    ProductCost,
    RANK() OVER(PARTITION BY ProductSubcategoryKey ORDER BY ProductCost DESC) AS ProductRank,
    CASE
        WHEN RANK() OVER(PARTITION BY ProductSubcategoryKey ORDER BY ProductCost DESC) = 1 THEN 'Top Rank'
        WHEN RANK() OVER(PARTITION BY ProductSubcategoryKey ORDER BY ProductCost DESC) <=5 THEN 'Medium Rank'
        ELSE 'Bottom Rank'
    END AS CostCategory
FROM Products;
```

```
-- Optimal Code [With CTE]

WITH ProductRanking AS (
    SELECT
        ProductSubcategoryKey,
        ProductName,
        ProductCost,
        RANK() OVER(PARTITION BY ProductSubcategoryKey ORDER BY ProductCost DESC) AS ProductRank
    FROM Products
)
SELECT
        *,
    CASE
        WHEN ProductRank = 1 THEN 'Top Rank'
        WHEN ProductRank <= 5 THEN 'Medium Rank'
        ELSE 'Bottom Rank'
    END AS CostCategory
FROM ProductRanking;
```

ProductSubcategoryKey	ProductName	ProductCost	ProductRank	CostCategory
1	Mountain-100 Silver, 38	1912.1544	1	Top Rank
1	Mountain-100 Silver, 42	1912.1544	1	Top Rank
1	Mountain-100 Silver, 44	1912.1544	1	Top Rank
1	Mountain-100 Silver, 48	1912.1544	1	Top Rank
1	Mountain-100 Black, 38	1898.0944	5	Medium Rank
1	Mountain-100 Black, 42	1898.0944	5	Medium Rank
1	Mountain-100 Black, 44	1898.0944	5	Medium Rank
1	Mountain-100 Black, 48	1898.0944	5	Medium Rank
1	Mountain-200 Silver, 38	1117.8559	9	Bottom Rank
1	Mountain-200 Silver, 42	1117.8559	9	Bottom Rank
1	Mountain-200 Silver, 46	1117.8559	9	Bottom Rank
1	Mountain-200 Black, 38	1105.81	12	Bottom Rank

DENSE_RANK()

```
SELECT
    CustomerKey,
    Gender,
    FullName,
    AnnualIncome,
    DENSE_RANK() OVER (partition by Gender ORDER BY AnnualIncome DESC) AS income_rank
FROM Customers;
```

CustomerKey	Gender	FullName	AnnualIncome	income_rank
12645	F	AUDREY RUIZ	170000	1
12318	F	KRISTINA SCHMIDT	170000	1
11250	F	SHANNON LIU	170000	1
11244	F	ALEXIS COLEMAN	170000	1
12329	F	BONNIE SHAN	170000	1
12648	F	LORI DOMINGUEZ	170000	1
12647	F	COLLEEN ANAND	170000	1
12658	F	JOY GOMEZ	170000	1
12361	F	DANA DIAZ	160000	2
11180	F	APRIL ANAND	160000	2
11240	F	ANNE HERNANDEZ	160000	2
12654	F	BRIANA GILL	160000	2
11436	F	TAYLOR COX	160000	2
11271	F	DANIELLE REED	150000	3
11243	F	ROBIN ALVAREZ	150000	3
12653	F	NICHOLE ANDERSEN	150000	3
12126	F	TAMMY RAMAN	150000	3
11291	F	JENNA WRIGHT	130000	4
12293	F	TAMMY GARCIA	130000	4

```
WITH ProductRanking AS (
    SELECT
        ProductSubcategoryKey,
        ProductName,
        ProductCost,
        DENSE_RANK() OVER(PARTITION BY ProductSubcategoryKey ORDER BY ProductCost DESC) AS ProductRank
    FROM Products
)
SELECT
        *,
    CASE
        WHEN ProductRank = 1 THEN 'Top Rank'
        WHEN ProductRank <= 3 THEN 'Medium Rank'
        ELSE 'Bottom Rank'
    END AS CostCategory
FROM ProductRanking;
```

ProductSubcategoryKey	ProductName	ProductCost	ProductRank	CostCategory
1	Mountain-100 Silver, 38	1912.1544	1	Top Rank
1	Mountain-100 Silver, 42	1912.1544	1	Top Rank
1	Mountain-100 Silver, 44	1912.1544	1	Top Rank
1	Mountain-100 Silver, 48	1912.1544	1	Top Rank
1	Mountain-100 Black, 38	1898.0944	2	Medium Rank
1	Mountain-100 Black, 42	1898.0944	2	Medium Rank
1	Mountain-100 Black, 44	1898.0944	2	Medium Rank
1	Mountain-100 Black, 48	1898.0944	2	Medium Rank
1	Mountain-200 Silver, 38	1117.8559	3	Medium Rank
1	Mountain-200 Silver, 42	1117.8559	3	Medium Rank
1	Mountain-200 Silver, 46	1117.8559	3	Medium Rank
1	Mountain-200 Black, 38	1105.81	4	Bottom Rank
1	Mountain-200 Black, 42	1105.81	4	Bottom Rank
1	Mountain-200 Black, 46	1105.81	4	Bottom Rank
1	Mountain-300 Black, 38	598.4354	5	Bottom Rank
1	Mountain-300 Black, 40	598.4354	5	Bottom Rank
1	Mountain-300 Black, 44	598.4354	5	Bottom Rank
1	Mountain-300 Black, 48	598.4354	5	Bottom Rank
1	Mountain-400-W Silver...	419.7784	6	Bottom Rank

ROW_NUMBER()

```
-- ROW_NUMBER()
SELECT
    CustomerKey,
    Gender,
    FullName,
    AnnualIncome,
    ROW_NUMBER() OVER (partition by Gender ORDER BY AnnualIncome DESC) AS income_rank
FROM Customers;
```

row_number_by_income

CustomerKey	Gender	FullName	AnnualIncome	income_rank
11081	F	SAVANNAH BAKER	NULL	1021
11015	F	CHLOE YOUNG	NULL	1022
11003	F	CHRISTY ZHU	NULL	1023
11467	M	ARTURO ZHENG	170000	1
12190	M	JUSTIN GONZALES	170000	2
12134	M	HECTOR ALONSO	170000	3
12710	M	DALTON EVANS	170000	4
12326	M	GLENN ZHOU	170000	5
11422	M	DUSTIN DENG	170000	6

LEAD() & LAG()

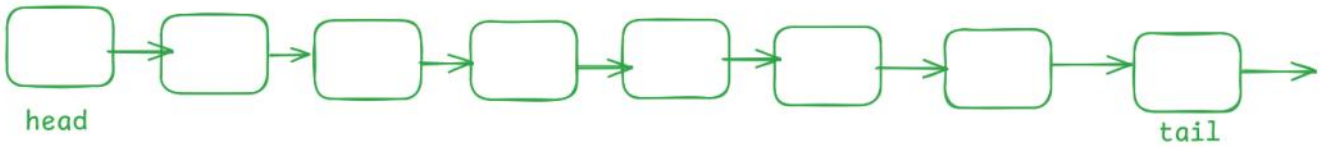
next Value

previous value

Field	Type	Null	Key	Default	Extra
SaleID	int	YES		NULL	
Salesperson	text	YES		NULL	
SaleAmount	int	YES		NULL	
SaleDate	text	YES		NULL	

```
SELECT
    SaleID,
    Salesperson,
    SaleAmount,
    saleDate,
    LAG(SaleAmount) OVER (PARTITION BY Salesperson ORDER BY SaleDate) AS PreviousSale,
    LEAD(SaleAmount) OVER (PARTITION BY Salesperson ORDER BY SaleDate) AS NextSale
FROM Sales;
```

SaleID	Salesperson	SaleAmount	saleDate	PreviousSale	NextSale
1	Alice	300	2023-01-01	NULL	200
3	Alice	200	2023-01-03	300	100
6	Alice	100	2023-01-06	200	450
8	Alice	450	2023-01-08	100	150
11	Alice	150	2023-01-11	450	350
14	Alice	350	2023-01-14	150	NULL
2	Bob	150	2023-01-02	NULL	300
5	Bob	300	2023-01-05	150	200
9	Bob	200	2023-01-09	300	250
12	Bob	250	2023-01-12	200	100
15	Bob	100	2023-01-15	250	NULL
4	Charlie	250	2023-01-04	NULL	350
7	Charlie	350	2023-01-07	250	400
10	Charlie	400	2023-01-10	350	300
13	Charlie	300	2023-01-13	400	NULL




```
SELECT
    DATE_FORMAT(s.OrderDate, "%Y-%m") AS year_and_month,

    ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue,

    LAG(ROUND(SUM(p.ProductPrice * s.OrderQuantity),0))
    OVER(ORDER BY DATE_FORMAT(s.OrderDate, "%Y-%m")) AS PreviousMonthRevenue,

    LEAD(ROUND(SUM(p.ProductPrice * s.OrderQuantity),0))
    OVER(ORDER BY DATE_FORMAT(s.OrderDate, "%Y-%m")) AS NextMonthRevenue

FROM `sales-2015` s
JOIN Products p
ON p.ProductKey = s.ProductKey
GROUP BY DATE_FORMAT(s.OrderDate, "%Y-%m")
ORDER BY year_and_month;
```

year_and_month	TotalRevenue	PreviousMonthRevenue	NextMonthRevenue
2015-01	585313	NULL	532226
2015-02	532226	585313	643436
2015-03	643436	532226	653364
2015-04	653364	643436	659326
2015-05	659326	653364	669989
2015-06	669989	659326	486115
2015-07	486115	669989	536453
2015-08	536453	486115	344063
2015-09	344063	536453	404277
2015-10	404277	344063	326611
2015-11	326611	404277	563762
2015-12	563762	326611	NULL

```sql
-- SAME ABOVE CODE WITH CTE
WITH Understanding_Revenue AS (
    SELECT
        DATE_FORMAT(s.OrderDate, "%Y-%m") AS year_and_month,
        ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
    FROM `sales-2015` s
    JOIN Products p
    ON p.ProductKey = s.ProductKey
    GROUP BY year_and_month
    ORDER BY year_and_month
)
SELECT
    year_and_month,
    TotalRevenue,
    LAG(TotalRevenue) OVER(ORDER BY year_and_month) AS PreviousMonthRevenue,
    LEAD(TotalRevenue) OVER(ORDER BY year_and_month) AS NextMonthRevenue
FROM Understanding_Revenue;
```