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.

Football Analogy

Aggregate

- We wanted to know the total goals per match

Window Function [Aggregate]

- Here, we need to know each and individual player performance of a match.

https://dev.mysql.com/doc/refman/8.4/en/date-and-time-functions.html

```
951 •  SELECT  AVG(AnnualIncome) FROM Customers;
952
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AVG(AnnualIncome)
57256.3353

```
953 •  SELECT
954        CustomerKey,
955        FullName,
956        AnnualIncome,
957        AnnualIncome - AVG(AnnualIncome) OVER() AS Income_Diff_From_Avg
958    FROM Customers;
```

Window Functions

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

CustomerKey	FullName	AnnualIncome	Income_Diff_From_Avg
11000	JON YANG	90000	32743.6647
11001	EUGENE HUANG	60000	2743.6647
11002	RUBEN TORRES	60000	2743.6647
11003	CHRISTY ZHU	NULL	NULL
11004	ELIZABETH JOHNSON	80000	22743.6647
11005	JULIO RUIZ	70000	12743.6647
11007	MARCO MEHTA	60000	2743.6647
11008	ROBIN VERHOFF	60000	2743.6647
11009	SHANNON CARLSON	70000	12743.6647
11010	JACQUELYN SUAREZ	70000	12743.6647
11011	CURTIS LU	60000	2743.6647
11012	LAUREN WALKER	100000	42743.6647
11013	IAN JENKINS	100000	42743.6647
11014	SYDNEY BENNETT	100000	42743.6647
11015	CHLOE YOUNG	NULL	NULL

Find the maximum income based on Prefix

```
SELECT DISTINCT Prefix FROM Customers;

SELECT
    CustomerKey,
    Prefix,
    FullName,
    AnnualIncome,
    MAX(AnnualIncome) OVER(PARTITION BY Prefix) AS max_income_by_prefix
FROM Customers;
```

Prefix
MR.
MS.
MRS.

CustomerKey	Prefix	FullName	AnnualIncome	max_income_by_prefix
12276		ALISHA SHAN	30000	130000
11082		ANGELA BUTLER	130000	130000
12790		BRANDON KUMAR	60000	130000
11035		WENDY DOMINGUEZ	10000	130000
12372		DARREN PRASAD	60000	130000
11000	MR.	JON YANG	90000	170000
11001	MR.	EUGENE HUANG	60000	170000
11002	MR.	RUBEN TORRES	60000	170000
12063	MR.	XAVIER MARTINEZ	70000	170000
12064	MR.	JOHN WHITE	70000	170000
11005	MR.	JULIO RUIZ	70000	170000
11007	MR.	MARCO MEHTA	60000	170000
12065	MR.	ISAIAH COLLINS	70000	170000
11009	MR.	SHANNON CARLSON	70000	170000
12066	MR.	WYATT POWELL	70000	170000

CustomerKey	Prefix	FullName	AnnualIncome	max_income_by_prefix
12322	MRS.	KARI MEHTA	80000	170000
12192	MRS.	JOCELYN HAYES	60000	170000
12129	MRS.	WENDY ALVAREZ	120000	170000
12325	MRS.	GLORIA MARTIN	130000	170000
13089	MRS.	ELIZABETH ALEXAN...	120000	170000
12194	MRS.	DESTINY FOSTER	70000	170000
12195	MRS.	ERIN MORRIS	70000	170000
13093	MRS.	MICHELLE JAMES	80000	170000
12196	MRS.	ALEXIS MILLER	70000	170000
11961	MS.	ANNE ALVAREZ	70000	170000
11962	MS.	ALEXANDRA ROBERTS	60000	170000
11968	MS.	KELSEY BECKER	130000	170000
11970	MS.	BAILEY COLLINS	80000	170000
11971	MS.	AMANDA ADAMS	80000	170000
11972	MS.	KATHERINE WILLIAMS	60000	170000

```
-- Find the maximum income based on Gender

SELECT DISTINCT Gender FROM Customers;

SELECT
    CustomerKey,
    FullName,
    Gender,
    AnnualIncome,
    MAX(AnnualIncome) OVER(PARTITION BY Gender) AS max_income_by_gender
FROM Customers;
```

CustomerKey	FullName	Gender	AnnualIncome	max_income_by_prefix
13074	GRACE HUGHES	F	60000	170000
13078	MICHELE CHANDE	F	70000	170000
13080	DENISE MEHTA	F	80000	170000
13081	KRISTA RUIZ	F	80000	170000
13083	JORDYN BUTLER	F	70000	170000
13088	MEGAN POWELL	F	60000	170000
13089	ELIZABETH ALEXAND...	F	120000	170000
13091	DEANNA MARTIN	F	130000	170000
13093	MICHELLE JAMES	F	80000	170000
12039	RICARDO NATH	M	70000	170000
12040	RICHARD TORRES	M	40000	170000
12041	SETH MARTIN	M	40000	170000
12042	ALEX CARTER	M	40000	170000
12043	CARSON PATTERSON	M	40000	170000
12045	LUIS RUSSELL	M	40000	170000
12050	AARON YOUNG	M	110000	170000
12051	ISAIAH SCOTT	M	120000	170000
11000	JON YANG	M	90000	170000
12053	CEDRIC GAO	M	130000	170000

gender

```
986    -- Find the maximum income based on EducationLevel , Occupation
987 •  SELECT DISTINCT EducationLevel , Occupation FROM Customers; -- 25 row(s) returned
988
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EducationLevel	Occupation
Bachelors	Professional
Bachelors	Management
Partial College	Skilled Manual
High School	Skilled Manual
Partial College	Clerical
Partial High School	Clerical
Graduate Degree	Management
Partial College	Professional
High School	Professional
Partial High School	Skilled Manual
Graduate Degree	Manual
Graduate Degree	Clerical
Bachelors	Manual
Partial College	Manual
Bachelors	Clerical
High School	Manual
Partial College	Management
High School	Management
Partial High School	Professional
Partial High School	Management
Bachelors	Skilled Manual

Find the maximum income based on EducationLevel , Occupation

```
SELECT DISTINCT EducationLevel , Occupation FROM Customers; -- 25 row(s) returned

SELECT
    CustomerKey,
    FullName,
    EducationLevel,
    Occupation,
    AnnualIncome,
    MAX(AnnualIncome) OVER(PARTITION BY EducationLevel, Occupation) AS max_income_by_Edu_Occupation
FROM Customers;
```

CustomerKey	FullName	EducationLevel	Occupation	AnnualIncome	max_income_by_Edu_Occupation
12810	CHASE STEWART	Bachelors	Clerical	30000	40000
12280	THERESA ALVAREZ	Bachelors	Clerical	30000	40000
11399	BRENDA MEHTA	Bachelors	Clerical	30000	40000
11398	COLIN NATH	Bachelors	Clerical	30000	40000
12284	CRISTINA BECK	Bachelors	Clerical	30000	40000
11395	BETH GUTIERREZ	Bachelors	Clerical	30000	40000
12815	DOMINIQUE MEHTA	Bachelors	Clerical	30000	40000
12821	CESAR MCDONALD	Bachelors	Clerical	40000	40000
12471	LACEY ZENG	Bachelors	Clerical	40000	40000
12472	JEFFERY ZHANG	Bachelors	Clerical	30000	40000
12799	GEORGE CHANDRA	Bachelors	Clerical	30000	40000
11549	CRYSTAL LIANG	Bachelors	Clerical	40000	40000
12459	STEFANIE RODRIG...	Bachelors	Clerical	30000	40000
11545	REGINALD DOMIN...	Bachelors	Clerical	30000	40000

CustomerKey	FullName	EducationLevel	Occupation	AnnualIncome	max_income_by_Edu_Occupation
11329	ANDY ALVAREZ	Bachelors	Management	90000	170000
11328	JULIAN GRIFFIN	Bachelors	Management	90000	170000
12085	TYLER RODRIGUEZ	Bachelors	Management	60000	170000
12086	EMILY CLARK	Bachelors	Management	60000	170000
11327	JAIME MORENO	Bachelors	Management	90000	170000
11971	AMANDA ADAMS	Bachelors	Management	80000	170000
11970	BAILEY COLLINS	Bachelors	Management	80000	170000
12362	THOMAS HARRISON	Bachelors	Management	90000	170000
12087	SEAN COOK	Bachelors	Management	70000	170000
12071	ROBERT ROBINSON	Bachelors	Management	70000	170000
12088	LAUREN THOMPSON	Bachelors	Management	70000	170000
12089	NATHANIEL RICHA...	Bachelors	Management	70000	170000
12094	BAILEY BAILEY	Bachelors	Management	60000	170000
12095	AUSTIN GRIFFIN	Bachelors	Management	60000	170000

CustomerKey	FullName	EducationLevel	Occupation	AnnualIncome	max_income_by_Edu_Occupation
11335	CARLA RAMAN	Bachelors	Manual	10000	10000
11337	JEROME ROMERO	Bachelors	Manual	10000	10000
11343	ARTHUR CARLSON	Bachelors	Manual	10000	10000
12543	RUBEN ALVAREZ	Bachelors	Manual	10000	10000
12537	TERRENCE LUO	Bachelors	Manual	10000	10000
12536	JAVIER DOMINGUEZ	Bachelors	Manual	10000	10000
11901	STACY ALVAREZ	Bachelors	Professional	60000	90000
12341	KATIE KUMAR	Bachelors	Professional	80000	90000
12340	STACY DIAZ	Bachelors	Professional	70000	90000
11810	ANTONIO WASHIN...	Bachelors	Professional	40000	90000
11902	DREW PAL	Bachelors	Professional	70000	90000
12339	CLAYTON JAI	Bachelors	Professional	70000	90000
12338	MONICA VANCE	Bachelors	Professional	70000	90000
12337	DUSTIN GOLDSTEIN	Bachelors	Professional	70000	90000

CASE With Window Functions

```
If the ProductCost = Max(ProductCost) Within each 'ProductSubcategory' -> Highest [Cost Category]
If the ProductCost = Min(ProductCost) Within each 'ProductSubcategory' -> Lowest [Cost Category]
Else : 'Medium'
```

```
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 'Medium'
    END AS CostCategory
FROM Products;
```

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	Medium
1	Mountain-100 Black, 42	1898.0944	Medium
1	Mountain-100 Black, 44	1898.0944	Medium
1	Mountain-100 Black, 48	1898.0944	Medium
1	Mountain-200 Silver, 38	1117.8559	Medium
1	Mountain-200 Silver, 42	1117.8559	Medium
1	Mountain-200 Silver, 46	1117.8559	Medium
1	Mountain-200 Black, 38	1105.81	Medium
1	Mountain-200 Black, 42	1105.81	Medium
1	Mountain-200 Black, 46	1105.81	Medium
1	Mountain-300 Black, 38	598.4354	Medium
1	Mountain-300 Black, 40	598.4354	Medium
1	Mountain-300 Black, 44	598.4354	Medium
1	Mountain-300 Black, 48	598.4354	Medium
1	Mountain-400-W Silver...	419.7784	Medium
1	Mountain-400-W Silver...	419.7784	Medium
1	Mountain-400-W Silver...	419.7784	Medium

ProductSubcategoryKey	ProductName	ProductCost	CostCategory
1	Mountain-500 Silver, 48	308.2179	Medium
1	Mountain-500 Silver, 52	308.2179	Medium
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
2	Road-150 Red, 56	2171.2942	Highest
2	Road-450 Red, 58	884.7083	Medium
2	Road-450 Red, 60	884.7083	Medium
2	Road-450 Red, 44	884.7083	Medium
2	Road-450 Red, 48	884.7083	Medium
2	Road-450 Red, 52	884.7083	Medium
2	Road-650 Red, 58	413.1463	Medium
2	Road-650 Red, 60	413.1463	Medium
2	Road-650 Red, 62	413.1463	Medium
2	Road-650 Red, 44	413.1463	Medium

ProductSubcategoryKey	Max(ProductCost)	MIN(ProductCost)
1	1912.1544	294.5797
2	2171.2942	343.6496
3	1481.9379	461.4448
4	48.5453	17.978
5	53.9416	23.9716
6	47.286	47.286
7	8.9866	8.9866
8	179.8156	77.9176
9	53.9282	40.6216
10	101.8936	65.8097
11	55.3801	15.1848

Rank the Customers [based on AnnualIncome] Partition By EducationLevel , Occupation

```
SELECT DISTINCT EducationLevel , Occupation FROM Customers; -- 25 row(s) returned

SELECT
    CustomerKey,
    FullName,
    EducationLevel,
    Occupation,
    AnnualIncome,
    DENSE_RANK() OVER(PARTITION BY EducationLevel, Occupation ORDER BY AnnualIncome DESC) AS IncomeRank
FROM Customers;
```

CustomerKey	FullName	EducationLevel	Occupation	AnnualIncome	IncomeRank
11566	APRIL SHAN	Bachelors	Clerical	40000	1
11549	CRYSTAL LIANG	Bachelors	Clerical	40000	1
11555	ALEXANDRIA HENDERSON	Bachelors	Clerical	40000	1
12822	ALEXANDRA JAMES	Bachelors	Clerical	40000	1
11550	DEB TORRES	Bachelors	Clerical	40000	1
12808	MARC SCHMIDT	Bachelors	Clerical	40000	1
12471	LACEY ZENG	Bachelors	Clerical	40000	1
12821	CESAR MCDONALD	Bachelors	Clerical	40000	1
11567	CARLA PEREZ	Bachelors	Clerical	40000	1
12567	CARMEN ARTHUR	Bachelors	Clerical	30000	2
11353	CARRIE ORTEGA	Bachelors	Clerical	30000	2

CustomerKey	FullName	EducationLevel	Occupation	AnnualIncome	IncomeRank
12061	BRYCE BROOKS	Bachelors	Management	160000	2
12706	LOUIS LIANG	Bachelors	Management	160000	2
12317	CARL SHE	Bachelors	Management	150000	3
11271	DANIELLE REED	Bachelors	Management	150000	3
12164	NOAH BUTLER	Bachelors	Management	150000	3
12163	AARON WANG	Bachelors	Management	150000	3
12653	NICHOLE ANDERSEN	Bachelors	Management	150000	3
12705	ANDRE GARCIA	Bachelors	Management	150000	3
11238	MAYRA PRASAD	Bachelors	Management	130000	4
11237	CLARENCE ANAND	Bachelors	Management	130000	4
12059	JASON JENKINS	Bachelors	Management	130000	4

CustomerKey	FullName	EducationLevel	Occupation	AnnualIncome	IncomeRank
12347	SUMMER MADAN	Bachelors	Professional	90000	1
12674	JIMMY TRAVERS	Bachelors	Professional	90000	1
12349	LACEY YUAN	Bachelors	Professional	90000	1
11454	MELINDA NAVARRO	Bachelors	Professional	80000	2
11459	TASHA DENG	Bachelors	Professional	80000	2
12693	RUTH GONZALEZ	Bachelors	Professional	80000	2
11004	ELIZABETH JOHNSON	Bachelors	Professional	80000	2
11765	MARC TORRES	Bachelors	Professional	80000	2
12681	CHRISTY CHOW	Bachelors	Professional	80000	2
11102	JULIA NELSON	Bachelors	Professional	80000	2
12680	GEORGE SANCHEZ	Bachelors	Professional	80000	2

Rank the Customers with each MaritalStatus and order by BirthDate (oldest To Youngest).

```
DESC Customers;
SELECT
    CustomerKey,
    FullName,
    MaritalStatus,
    DateOfBirth,
    DENSE_RANK() OVER(PARTITION BY MaritalStatus ORDER BY DateOfBirth) AS Age_Ranking
FROM Customers;
```

CustomerKey	FullName	MaritalStatus	DateOfBirth	Age_Ranking
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
11253	JOSÃ‰ HERNANDEZ	M	1933-02-28	13
11254	JOHNATHAN VANCE	M	1933-03-03	14
12139	FRANKLIN YANG	M	1933-03-15	15
11255	COLIN LIN	M	1933-04-04	16

CustomerKey	FullName	MaritalStatus	DateOfBirth	Age_Ranking
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
12179	ALEXIS JONES	S	1935-10-10	11
12180	HOLLY MEHTA	S	1936-09-17	12
12205	MADISON LONG	S	1938-02-10	13
11323	JOSE PATTERSON	S	1938-06-20	14
11139	TANYA MORENO	S	1938-09-11	15
11325	ELIJAH ROSS	S	1939-02-07	16
11351	ANNE RAMOS	S	1939-06-04	17

Rank the Product and create a category with a case statement where the ProductCost is being ranked based on ProductSubcategory.

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

```sql
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 'TopRank'
        WHEN RANK() OVER(PARTITION BY ProductSubcategoryKey ORDER BY ProductCost DESC) <= 5 THEN 'MediumRank'
        ELSE 'BottomRank'
    END AS CostCategory
FROM Products;
```

```sql
-- Same Above Code With CTE [Optimized]

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 'TopRank'
        WHEN ProductRank <= 5 THEN 'MediumRank'
        ELSE 'BottomRank'
    END AS CostCategory
FROM ProductRanking;
```

ProductSubcategoryKey	ProductName	ProductCost	ProductRank	CostCategory
1	Mountain-100 Silver, 44	1912.1544	1	TopRank
1	Mountain-100 Silver, 48	1912.1544	1	TopRank
1	Mountain-100 Black, 38	1898.0944	5	MediumRank
1	Mountain-100 Black, 42	1898.0944	5	MediumRank
1	Mountain-100 Black, 44	1898.0944	5	MediumRank
1	Mountain-100 Black, 48	1898.0944	5	MediumRank
1	Mountain-200 Silver, 38	1117.8559	9	BottomRank
1	Mountain-200 Silver, 42	1117.8559	9	BottomRank
1	Mountain-200 Silver, 46	1117.8559	9	BottomRank
1	Mountain-200 Black, 38	1105.81	12	BottomRank
1	Mountain-200 Black, 42	1105.81	12	BottomRank
1	Mountain-200 Black, 46	1105.81	12	BottomRank
1	Mountain-300 Black, 38	598.4354	15	BottomRank
1	Mountain-300 Black, 40	598.4354	15	BottomRank

ROW_NUMBER()

```
-- ROW_NUMBER() -> Index [1 till the Partition Count]

SELECT
    CustomerKey,
    FullName,
    EducationLevel,
    Occupation,
    AnnualIncome,
    ROW_NUMBER() OVER(PARTITION BY EducationLevel, Occupation ORDER BY AnnualIncome DESC) AS Row_Index
FROM Customers;
```

CustomerKey	FullName	EducationLevel	Occupation	AnnualIncome	Row_Index
12247	BRANDY RAMAN	Bachelors	Clerical	20000	84
12556	MANUEL KAPOOR	Bachelors	Clerical	10000	85
12226	FAITH WARD	Bachelors	Clerical	10000	86
12557	PHILLIP LOPEZ	Bachelors	Clerical	10000	87
12555	EDGAR MALHOTRA	Bachelors	Clerical	10000	88
11244	ALEXIS COLEMAN	Bachelors	Management	170000	1
12318	KRISTINA SCHMIDT	Bachelors	Management	170000	2
12123	WESLEY LIANG	Bachelors	Management	170000	3
12645	AUDREY RUIZ	Bachelors	Management	170000	4
11422	DUSTIN DENG	Bachelors	Management	170000	5
11180	APRIL ANAND	Bachelors	Management	160000	6
12061	BRYCE BROOKS	Bachelors	Management	160000	7
12706	LOUIS LIANG	Bachelors	Management	160000	8
12317	CARL SHE	Bachelors	Management	150000	9

Lead() & Lag()

next_val

previous_val

```
099 •  DESC Sale;
```

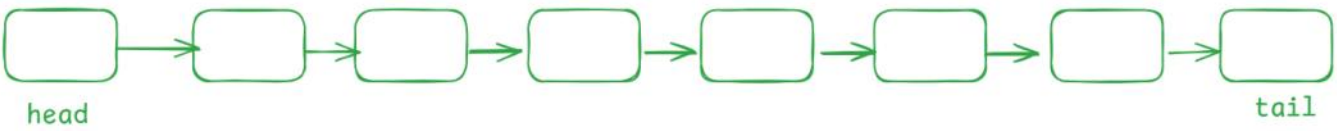
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

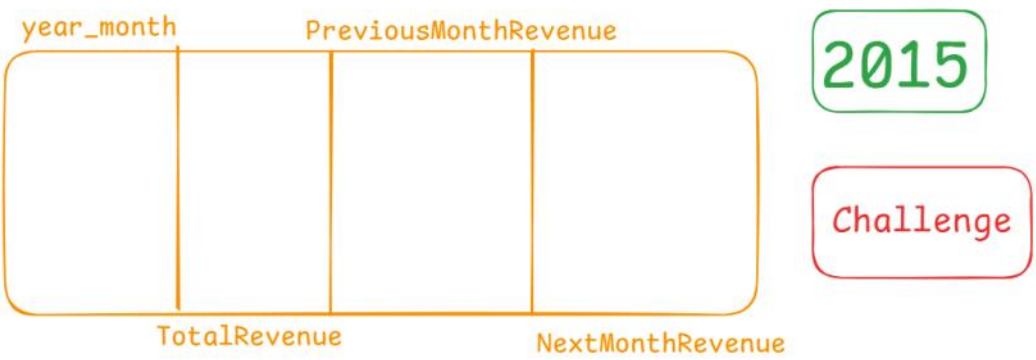
Field	Type	Null	Key	Default	Extra
SaleID	int	YES		NULL	
Salesperson	text	YES		NULL	
SaleAmount	int	YES		NULL	
SaleDate	text	YES		NULL	

```sql
DESC Sale;

SELECT
    *,
    LAG(SaleAmount) OVER (PARTITION BY Salesperson ORDER BY SaleDate) AS PreviousSale,
    LEAD(SaleAmount) OVER (PARTITION BY Salesperson ORDER BY SaleDate) AS NextSale
FROM Sale;
```

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





2015

Challenge

```sql
SELECT * FROM Sales2015;

SELECT
    DATE_FORMAT(s.OrderDate , '%Y-%m') AS YearMonth,
    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 Sales2015 s
JOIN Products p
ON p.ProductKey = s.ProductKey
GROUP BY 1
ORDER BY 1;
```

YearMonth	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 CODE AS ABOVE USING CTE [Optimized]
WITH Understanding_Revenue AS (
    SELECT
        DATE_FORMAT(s.OrderDate , '%Y-%m') AS YearMonth,
        ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
    FROM Sales2015 s
    JOIN Products p
    ON p.ProductKey = s.ProductKey
    GROUP BY 1
    ORDER BY 1
)
SELECT
    *,
    LAG(TotalRevenue) OVER(ORDER BY YearMonth) AS PreviousMonthRevenue,
    LEAD(TotalRevenue) OVER(ORDER BY YearMonth) AS NextMonthRevenue
FROM Understanding_Revenue;
```

ALL Sales
[Multiple CTE]
HomeWork

Challenge

YearMonth	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

SaleAmountChange

TotalRevenue - PreviousMonthRevenue

```
-- Challenge - Adding SaleAmountChange in Above Table
WITH Understanding_Revenue AS (
    SELECT
        DATE_FORMAT(s.OrderDate , '%Y-%m') AS YearMonth,
        ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
    FROM Sales2015 s
    JOIN Products p
    ON p.ProductKey = s.ProductKey
    GROUP BY 1
    ORDER BY 1
)
SELECT
    *,
    LAG(TotalRevenue) OVER(ORDER BY YearMonth) AS PreviousMonthRevenue,
    LEAD(TotalRevenue) OVER(ORDER BY YearMonth) AS NextMonthRevenue,
    TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY YearMonth) AS SaleAmountChange
FROM Understanding_Revenue;
```

YearMonth	TotalRevenue	PreviousMonthRevenue	NextMonthRevenue	SaleAmountChange
2015-01	585313	NULL	532226	NULL
2015-02	532226	585313	643436	-53087
2015-03	643436	532226	653364	111210
2015-04	653364	643436	659326	9928
2015-05	659326	653364	669989	5962
2015-06	669989	659326	486115	10663
2015-07	486115	669989	536453	-183874
2015-08	536453	486115	344063	50338
2015-09	344063	536453	404277	-192390
2015-10	404277	344063	326611	60214
2015-11	326611	404277	563762	-77666
2015-12	563762	326611	NULL	237151

Challenge

SaleTrend

YearMonth	TotalRevenue	PreviousMonthRevenue	NextMonthRevenue	SaleAmountChange
2015-01	585313	NULL	532226	NULL
2015-02	532226	585313	643436	-53087
2015-03	643436	532226	653364	111210
2015-04	653364	643436	659326	9928
2015-05	659326	653364	669989	5962
2015-06	669989	659326	486115	10663
2015-07	486115	669989	536453	-183874
2015-08	536453	486115	344063	50338
2015-09	344063	536453	404277	-192390
2015-10	404277	344063	326611	60214
2015-11	326611	404277	563762	-77666
2015-12	563762	326611	NULL	237151

CASE with Window Function

```
TotalRevenue > PreviousMonthRevenue -> Increase
TotalRevenue < PreviousMonthRevenue -> Decrease
ELSE : No Change
```

```sql
-- Adding SalesTrend on the above query:

WITH Understanding_Revenue AS (
    SELECT
        DATE_FORMAT(s.OrderDate , '%Y-%m') AS YearMonth,
        ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
    FROM Sales2015 s
    JOIN Products p
    ON p.ProductKey = s.ProductKey
    GROUP BY 1
    ORDER BY 1
)
SELECT
    *,
    LAG(TotalRevenue) OVER(ORDER BY YearMonth) AS PreviousMonthRevenue,
    LEAD(TotalRevenue) OVER(ORDER BY YearMonth) AS NextMonthRevenue,
    TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY YearMonth) AS SaleAmountChange,
    CASE
        WHEN TotalRevenue > LAG(TotalRevenue) OVER(ORDER BY YearMonth) THEN 'Increase'
        WHEN TotalRevenue < LAG(TotalRevenue) OVER(ORDER BY YearMonth) THEN 'Decrease'
        ELSE 'No Change'
    END AS SalesTrend
FROM Understanding_Revenue;
```

YearMonth	TotalRevenue	PreviousMonthRevenue	NextMonthRevenue	SaleAmountChange	SalesTrend
2015-01	585313	NULL	532226	NULL	No Change
2015-02	532226	585313	643436	-53087	Decrease
2015-03	643436	532226	653364	111210	Increase
2015-04	653364	643436	659326	9928	Increase
2015-05	659326	653364	669989	5962	Increase
2015-06	669989	659326	486115	10663	Increase
2015-07	486115	669989	536453	-183874	Decrease
2015-08	536453	486115	344063	50338	Increase
2015-09	344063	536453	404277	-192390	Decrease
2015-10	404277	344063	326611	60214	Increase
2015-11	326611	404277	563762	-77666	Decrease
2015-12	563762	326611	NULL	237151	Increase

```sql
-- ALTERNATIVE Solution

WITH Understanding_Revenue AS (
    SELECT
        DATE_FORMAT(s.OrderDate , '%Y-%m') AS YearMonth,
        ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
    FROM Sales2015 s
    JOIN Products p
    ON p.ProductKey = s.ProductKey
    GROUP BY 1
    ORDER BY 1
)
SELECT
    *,
    LAG(TotalRevenue) OVER(ORDER BY YearMonth) AS PreviousMonthRevenue,
    LEAD(TotalRevenue) OVER(ORDER BY YearMonth) AS NextMonthRevenue,
    TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY YearMonth) AS SaleAmountChange,
    CASE
        WHEN TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY YearMonth) > 0 THEN 'Increase'
        WHEN TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY YearMonth) < 0 THEN 'Decrease'
        ELSE 'No Change'
    END AS SalesTrend
FROM Understanding_Revenue;
```

All Sales

Group By Year
Group By Year-Month