Window Functions-III

🎯 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.

NTILE() → using this to bucket our data into 'n' group

```
NTILE(Number of Buckets) OVER(ORDER BY columnName)
```

SPLIT THE ProductPrice Columns in Quartile -> [4 Parts]

```
SELECT
    ProductName,
    ProductPrice
FROM Products
ORDER BY ProductPrice;
```

ProductName	ProductPrice
Patch Kit/8 Patches	2.29
Road Tire Tube	3.99
Water Bottle - 30 oz.	4.99
Mountain Tire Tube	4.99
Touring Tire Tube	4.99
Bike Wash - Dissolver	7.95
AWC Logo Cap	8.6442
Road Bottle Cage	8.99
Racing Socks, M	8.99
Racing Socks, L	8.99
Mountain Bike Socks, M	9.5
Mountain Bike Socks, L	9.5
Mountain Bottle Cage	9.99
Taillights - Battery-P...	13.99
Minipump	19.99
Chain	20.24

```
SELECT
    ProductName,
    ProductPrice,
    NTILE(4) OVER(ORDER BY ProductPrice) AS Price_Quartile
FROM Products;
```

ProductName	ProductPrice	Price_Quartile
Patch Kit/8 Patches	2.29	1
Road Tire Tube	3.99	1
Water Bottle - 30 oz.	4.99	1
Mountain Tire Tube	4.99	1
Touring Tire Tube	4.99	1
Bike Wash - Dissolver	7.95	1
AWC Logo Cap	8.6442	1
Road Bottle Cage	8.99	1
Racing Socks, M	8.99	1
Racing Socks, L	8.99	1
Mountain Bike Socks, M	9.5	1
Mountain Bike Socks, L	9.5	1
Mountain Bottle Cage	9.99	1
Taillights - Battery-P...	13.99	1
Minipump	19.99	1
Chain	20.24	1

ProductName	ProductPrice	Price_Quartile
Classic Vest, L	63.5	1
Women's Mountain S...	69.99	2
Women's Mountain S...	69.99	2
Women's Mountain S...	69.99	2
Women's Tights, S	74.99	2
Women's Tights, M	74.99	2
Women's Tights, L	74.99	2
HL Mountain Pedal	80.99	2
HL Road Pedal	80.99	2
Touring Pedal	80.99	2
LL Road Front Wheel	85.565	2
LL Mountain Rear W...	87.745	2
Men's Bib-Shorts, S	89.99	2
Men's Bib-Shorts, M	89.99	2
Men's Bib-Shorts, L	89.99	2
Front Derailleur	91.49	2

ProductName	ProductPrice	Price_Quartile
Road-550-W Yellow,...	1000.4375	3
HL Touring Frame - ...	1003.91	3
HL Touring Frame - ...	1003.91	3
HL Touring Frame - ...	1003.91	3
HL Touring Frame - ...	1003.91	3
HL Touring Frame - ...	1003.91	3
HL Touring Frame - ...	1003.91	3
HL Touring Frame - ...	1003.91	4
HL Touring Frame - ...	1003.91	4
Mountain-300 Black,...	1079.99	4
Mountain-300 Black,...	1079.99	4
Mountain-300 Black,...	1079.99	4
Mountain-300 Black,...	1079.99	4
HL Mountain Frame -...	1191.1739	4
HL Mountain Frame -...	1191.1739	4
HL Mountain Frame -...	1191.1739	4

Challenge

Divide each month's total sales into 3 performance tiers (terciles) to categorise months as low, medium, or high-performing.

```
With MonthlySales 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 s.ProductKey = p.ProductKey
    GROUP BY 1
    ORDER BY 1
)
SELECT * FROM MonthlySales;
```

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

```
With MonthlySales 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 s.ProductKey = p.ProductKey
    GROUP BY 1
    ORDER BY 1
)
SELECT
    YearMonth,
    TotalRevenue,
    NTILE(3) OVER(ORDER BY TotalRevenue) AS sales_tercile
FROM MonthlySales;
```

YearMonth	TotalRevenue	sales_tercile
2015-11	326611	1
2015-09	344063	1
2015-10	404277	1
2015-07	486115	1
2015-02	532226	2
2015-08	536453	2
2015-12	563762	2
2015-01	585313	2
2015-03	643436	3
2015-04	653364	3
2015-05	659326	3
2015-06	669989	3

YearMonth	TotalRevenue	sales_tercile
2016-01	432426	1
2016-03	471962	1
2016-02	474163	1
2016-04	494957	1
2016-06	533825	2
2016-05	545535	2
2016-08	804193	2
2016-07	815356	2
2016-09	952743	3
2016-10	1029821	3
2016-11	1133913	3
2016-12	1635309	3

YearMonth	TotalRevenue	sales_tercile
2017-01	1274379	1
2017-02	1339241	1
2017-03	1448596	2
2017-04	1527814	2
2017-05	1768433	3
2017-06	1826987	3

```sql
-- ALL Sales

With MonthlySales AS(
    SELECT
        DATE_FORMAT(s.OrderDate , '%Y-%m') AS YearMonth,
        ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
    FROM (
        SELECT * FROM Sales2015
        UNION
        SELECT * FROM Sales2016
        UNION
        SELECT * FROM Sales2017
    ) s
    JOIN Products p
    ON s.ProductKey = p.ProductKey
    GROUP BY 1
    ORDER BY 1
)
SELECT
    YearMonth,
    TotalRevenue,
    NTILE(3) OVER(ORDER BY TotalRevenue) AS sales_tercile
FROM MonthlySales;
```

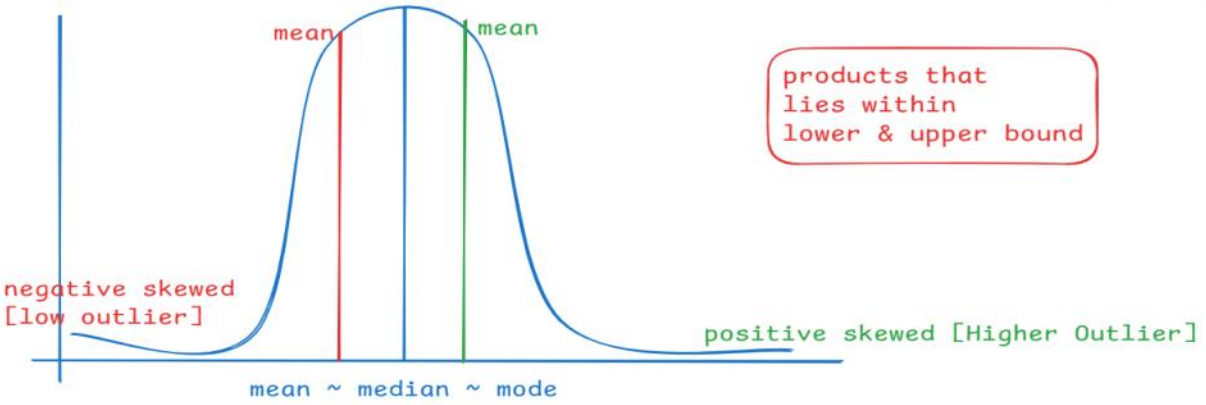
YearMonth	TotalRevenue	sales_tercile
2015-11	326611	1
2015-09	344063	1
2015-10	404277	1
2016-01	432426	1
2016-03	471962	1
2016-02	474163	1
2015-07	486115	1
2016-04	494957	1
2015-02	532226	1
2016-06	533825	1
2015-08	536453	2
2016-05	545535	2
2015-12	563762	2
2015-01	585313	2
2015-03	643436	2
2015-04	653364	2

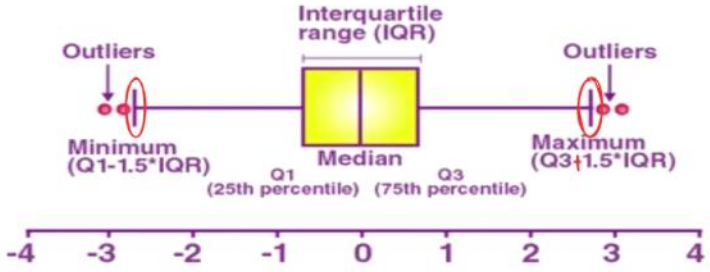
YearMonth	TotalRevenue	sales_tercile
2015-03	643436	2
2015-04	653364	2
2015-05	659326	2
2015-06	669989	2
2016-08	804193	2
2016-07	815356	2
2016-09	952743	3
2016-10	1029821	3
2016-11	1133913	3
2017-01	1274379	3
2017-02	1339241	3
2017-03	1448596	3
2017-04	1527814	3
2016-12	1635309	3
2017-05	1768433	3
2017-06	1826987	3

Challenge

<lower_bound & >upper_bound [Outlier Detection]

Find the products which are either below the lower bound or above the upper bound [Outliers]






```
-- Filter the Products lying within the lower bound and upper bound....
WITH Product_Stats AS (
    SELECT
        ProductPrice,
        NTILE(4) OVER (ORDER BY ProductPrice) AS Price_quartile
    FROM Products
),
quartiles AS (
    SELECT
        MAX(CASE WHEN Price_quartile = 1 THEN ProductPrice END) AS Q1,
        MAX(CASE WHEN Price_quartile = 3 THEN ProductPrice END) AS Q3
    FROM Product_Stats
),
iqr_bounds AS (
    SELECT
        Q1,
        Q3,
        Q3-Q1 AS IQR,
        Q1-(1.5*(Q3-Q1)) AS Lower_Bound,
        Q3+(1.5*(Q3-Q1)) AS Upper_Bound
    FROM quartiles
)
SELECT
    p.ProductKey,
    p.ProductName,
    p.ProductPrice
FROM Products p
JOIN iqr_bounds iqr
ON p.ProductPrice BETWEEN iqr.Lower_Bound AND iqr.Upper_Bound
ORDER BY ProductPrice;
```

```
SELECT * FROM iqr_bounds;
```

Result Grid | Filter Rows: | Export: | Wrap Cel

Q1	Q3	IQR	Lower_Bound	Upper_Bound
63.5	1003.91	940.41	-1347.115	2414.525

277 row(s) returned
Out of 293 Records

ProductKey	ProductName	ProductPrice
480	Patch Kit/8 Patches	2.29
529	Road Tire Tube	3.99
477	Water Bottle - 30 oz.	4.99
528	Mountain Tire Tube	4.99
530	Touring Tire Tube	4.99
484	Bike Wash - Dissolver	7.95
223	AWC Logo Cap	8.6442
479	Road Bottle Cage	8.99
481	Racing Socks, M	8.99
482	Racing Socks, L	8.99
218	Mountain Bike Socks, M	9.5
219	Mountain Bike Socks, L	9.5
478	Mountain Bottle Cage	9.99
450	Taillights - Battery-P...	13.99
448	Minipump	19.99
559	Chain	20.24
538	LL Road Tire	21.49

ProductKey	ProductName	ProductPrice
539	ML Road Tire	24.99
447	LL Mable Lock	25
515	LL Mountain Seat/Sa...	27.12
518	LL Road Seat/Saddle	27.12
521	LL Touring Seat/Saddle	27.12
541	Touring Tire	28.99
536	ML Mountain Tire	29.99
540	HL Road Tire	32.6
215	Sport-100 Helmet, Bl...	33.6442
220	Sport-100 Helmet, Blue	33.6442
394	LL Headset	34.2
214	Sport-100 Helmet, Red	34.99
451	Headlights - Dual-Beam	34.99
537	HL Mountain Tire	35
468	Full-Finger Gloves, S	37.99
469	Full-Finger Gloves, M	37.99
470	Full-Finger Gloves, L	37.99

ProductKey	ProductName	ProductPrice
362	Mountain-200 Black,...	2049.0982
352	Mountain-200 Silver,...	2071.4196
354	Mountain-200 Silver,...	2071.4196
356	Mountain-200 Silver,...	2071.4196
371	Road-250 Red, 58	2181.5625
373	Road-250 Black, 44	2181.5625
375	Road-250 Black, 48	2181.5625
377	Road-250 Black, 52	2181.5625
379	Road-250 Black, 58	2181.5625
561	Touring-1000 Yellow...	2384.07
562	Touring-1000 Yellow...	2384.07
563	Touring-1000 Yellow...	2384.07
564	Touring-1000 Yellow...	2384.07
573	Touring-1000 Blue, 46	2384.07
574	Touring-1000 Blue, 50	2384.07
575	Touring-1000 Blue, 54	2384.07
576	Touring-1000 Blue, 60	2384.07

Find the products which are either below the lower bound or above the upper bound [Outliers]

16 Outliers
Detected

```
WITH Product_Stats AS (
    SELECT
        ProductPrice,
        NTILE(4) OVER (ORDER BY ProductPrice) AS Price_quartile
    FROM Products
),
quartiles AS (
    SELECT
        MAX(CASE WHEN Price_quartile = 1 THEN ProductPrice END) AS Q1,
        MAX(CASE WHEN Price_quartile = 3 THEN ProductPrice END) AS Q3
    FROM Product_Stats
),
iqr_bounds AS (
    SELECT
        Q1,
        Q3,
        Q3-Q1 AS IQR,
        Q1-(1.5*(Q3-Q1)) AS Lower_Bound,
        Q3+(1.5*(Q3-Q1)) AS Upper_Bound
    FROM quartiles
)
SELECT
    p.ProductKey,
    p.ProductName,
    p.ProductPrice
FROM Products p
JOIN iqr_bounds iqr    Low_Outlier                    High_Outlier
ON p.ProductPrice < iqr.Lower_Bound OR p.ProductPrice > iqr.Upper_Bound
ORDER BY ProductPrice; -- 16 row(s) returned
```

First_Value() → Earliest

Syntax:

```
First_Value(Column) OVER(Partition BY.... ORDER BY .....)
```

Syntax:

```
SELECT
  window_function(...) OVER (
    PARTITION BY column_name
    ORDER BY column_name
    ROWS/RANGE ...
  ) AS result_column
FROM table_name;
```

→ Unbounded

1. Find the customers who purchased each products based on the earliest date.

```
SELECT
    s.ProductKey,
    c.FullName,
    s.OrderDate,
    FIRST_VALUE(c.FullName) OVER(PARTITION BY s.ProductKey ORDER BY s.OrderDate)
    AS FirstPurchase
FROM Sales2015 s
JOIN Customers c
ON c.CustomerKey = s.CustomerKey;
```

ProductKey	FullName	OrderDate	FirstPurchase
310	JARED COOK	2015-01-23	JARED COOK
310	TIFFANY LI	2015-02-07	JARED COOK
310	TREVOR BRYANT	2015-02-07	JARED COOK
310	NATALIE ADAMS	2015-02-10	JARED COOK

ProductKey	FullName	OrderDate	FirstPurchase
311	MEREDITH ALVAREZ	2015-01-10	MEREDITH ALVAREZ
311	NICHOLAS THOM...	2015-02-14	MEREDITH ALVAREZ
311	DONNA XIE	2015-02-18	MEREDITH ALVAREZ
311	STEVE WAGNER	2015-02-28	MEREDITH ALVAREZ
311	KYLE FOSTER	2015-03-01	MEREDITH ALVAREZ

ProductKey	FullName	OrderDate	FirstPurchase
312	AMANDA PEREZ	2015-01-04	AMANDA PEREZ
312	VERONICA SUBRAM	2015-01-12	AMANDA PEREZ
312	KRISTI PEREZ	2015-02-11	AMANDA PEREZ
312	JASMINE TORRES	2015-02-15	AMANDA PEREZ

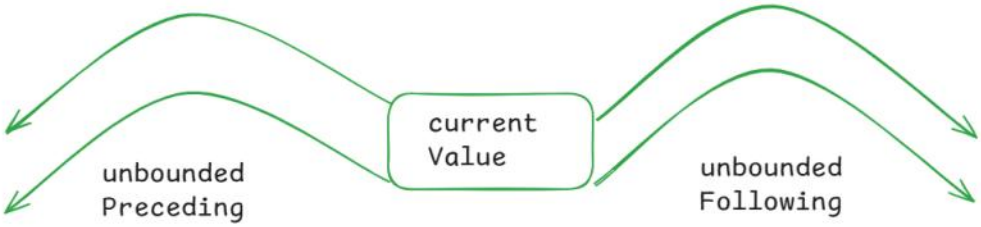
```sql
WITH first_Customer AS (
    SELECT
        s.ProductKey,
        c.FullName,
        s.OrderDate,
        FIRST_VALUE(c.FullName) OVER(PARTITION BY s.ProductKey ORDER BY s.OrderDate)
        AS FirstPurchase
    FROM Sales2015 s
    JOIN Customers c
    ON c.CustomerKey = s.CustomerKey
)
SELECT
    DISTINCT ProductKey, FirstPurchase
FROM first_Customer;
```

ProductKey	FirstPurchase
310	JARED COOK
311	MEREDITH ALVAREZ
312	AMANDA PEREZ
313	TAMMY CHANDRA
314	WAYNE NATH
344	GEORGE MCDONALD
345	COLIN NATH
346	TASHA DENG
347	JON GAO
348	JACLYN ANDERSEN
349	ERIKA RUBIO
350	ROSS SANZ
351	LAURA LIN
352	BLAKE COLLINS
354	JENNIFER GREEN
356	ERICK SAI
358	MONIQUE ORTEGA
360	CASEY MUNOZ
362	LACEY YUAN
368	TRISHA MA
369	KEVIN BAKER

LAST_VALUE() → Latest Record

UNBOUNDED PRECEDING & UNBOUNDED FOLLOWING

Find the last region where each product was sold based on the latest order date?




```sql
SELECT
    s.ProductKey,
    t.Region,
    s.OrderDate,
    LAST_VALUE(t.Region) OVER(PARTITION BY s.ProductKey ORDER BY s.OrderDate
        RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
    ) AS last_region_product_sold
FROM Sales2015 s
JOIN Territories t
ON s.TerritoryKey = t.SalesTerritoryKey;
```

ProductKey	Region	OrderDate	last_region_product_sold
310	Southwest	2015-06-24	Southwest
310	Canada	2015-06-27	Southwest
310	Australia	2015-06-27	Southwest
310	Southwest	2015-06-27	Southwest
310	Northwest	2015-06-29	Southwest
310	Southwest	2015-06-30	Southwest

ProductKey	Region	OrderDate	last_region_product_sold
311	Germany	2015-06-24	Germany
311	Southwest	2015-06-25	Germany
311	Northwest	2015-06-25	Germany
311	Northwest	2015-06-27	Germany
311	Southwest	2015-06-27	Germany
311	Germany	2015-06-30	Germany

ProductKey	Region	OrderDate	last_region_product_sold
312	Northwest	2015-06-25	Australia
312	Southwest	2015-06-26	Australia
312	United King...	2015-06-29	Australia
312	Canada	2015-06-29	Australia
312	Australia	2015-06-30	Australia

ProductKey	Region	OrderDate	last_region_product_sold
313	Northwest	2015-06-25	Canada
313	France	2015-06-26	Canada
313	United King...	2015-06-26	Canada
313	Northwest	2015-06-28	Canada
313	Canada	2015-06-29	Canada

NTH-VALUE()

```
NTH-VALUE(Column,Nth) OVER(PARTITION BY ..... ORDER BY ..... RANGES .....)
```

From the Sales Record, Retrieve the 5th Customer from every product using Nth_value.

```
SELECT
    s.ProductKey,
    c.FullName,
    s.OrderDate,
    NTH_VALUE(c.FullName , 5) OVER
    (PARTITION BY s.ProductKey ORDER BY s.OrderDate) AS FifthPurchase
FROM Sales2015 s
JOIN Customers c
ON c.CustomerKey = s.CustomerKey;
```

ProductKey	FullName	OrderDate	FifthPurchase
310	JARED COOK	2015-01-23	NULL
310	TIFFANY LI	2015-02-07	NULL
310	TREVOR BRYANT	2015-02-07	NULL
310	NATALIE ADAMS	2015-02-10	NULL
310	BRADLEY NARA	2015-02-14	BRADLEY NARA
310	BRANDY SANCHEZ	2015-02-24	BRADLEY NARA
310	ROBERTO SANZ	2015-02-27	BRADLEY NARA
310	NINA YUAN	2015-03-02	BRADLEY NARA
310	DEVIN NELSON	2015-03-05	BRADLEY NARA
310	JULIA COLEMAN	2015-03-05	BRADLEY NARA
310	VICTORIA STEW...	2015-03-07	BRADLEY NARA
310	JADE BAILEY	2015-03-12	BRADLEY NARA
310	AIDAN ROSS	2015-03-15	BRADLEY NARA
310	JONATHAN PHILL...	2015-03-19	BRADLEY NARA
310	ELIJAH ROSS	2015-03-22	BRADLEY NARA

ProductKey	FullName	OrderDate	FifthPurchase
311	MEREDITH ALVAREZ	2015-01-10	NULL
311	NICHOLAS THOM...	2015-02-14	NULL
311	DONNA XIE	2015-02-18	NULL
311	STEVE WAGNER	2015-02-28	NULL
311	KYLE FOSTER	2015-03-01	KYLE FOSTER
311	JOEL PRASAD	2015-03-02	KYLE FOSTER
311	DESTINY ROGERS	2015-03-04	KYLE FOSTER
311	DAWN LAL	2015-03-06	KYLE FOSTER
311	SPENCER RUSSELL	2015-03-09	KYLE FOSTER
311	TAYLOR HOWARD	2015-03-13	KYLE FOSTER
311	SETH PHILLIPS	2015-03-18	KYLE FOSTER
311	NICOLE TAYLOR	2015-03-19	KYLE FOSTER
311	LAUREN MARTINEZ	2015-03-22	KYLE FOSTER
311	KATELYN SANCHEZ	2015-03-23	KYLE FOSTER
311	DAKOTA ROSS	2015-03-24	KYLE FOSTER

```sql
SELECT
    s.ProductKey,
    c.FullName,
    s.OrderDate,
    NTH_VALUE(c.FullName , 5) OVER(PARTITION BY s.ProductKey ORDER BY s.OrderDate
    RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS FifthPurchase
FROM Sales2015 s
JOIN Customers c
ON c.CustomerKey = s.CustomerKey;
```

ProductKey	FullName	OrderDate	FifthPurchase
310	JARED COOK	2015-01-23	BRADLEY NARA
310	TIFFANY LI	2015-02-07	BRADLEY NARA
310	TREVOR BRYANT	2015-02-07	BRADLEY NARA
310	NATALIE ADAMS	2015-02-10	BRADLEY NARA
310	BRADLEY NARA	2015-02-14	BRADLEY NARA
310	BRANDY SANCHEZ	2015-02-24	BRADLEY NARA
310	ROBERTO SANZ	2015-02-27	BRADLEY NARA
310	NINA YUAN	2015-03-02	BRADLEY NARA
310	DEVIN NELSON	2015-03-05	BRADLEY NARA
310	JULIA COLEMAN	2015-03-05	BRADLEY NARA
310	VICTORIA STEW...	2015-03-07	BRADLEY NARA
310	JADE BAILEY	2015-03-12	BRADLEY NARA
310	AIDAN ROSS	2015-03-15	BRADLEY NARA
310	JONATHAN PHILL...	2015-03-19	BRADLEY NARA
310	ELIJAH ROSS	2015-03-22	BRADLEY NARA

🔶 Question 1: Avg Days Between Orders for Same Subcategory

🧩 Problem Statement :

For each customer and product subcategory, calculate the average number of days between consecutive orders.

```sql
WITH subcategory_orders AS (
    SELECT
        s.CustomerKey,
        p.ProductSubcategoryKey,
        s.OrderDate,
        LAG(s.OrderDate) OVER(PARTITION BY s.CustomerKey, p.ProductSubcategoryKey
        ORDER BY s.OrderDate) AS PreviousDate
    FROM Products p
    JOIN Sales2017 s
    ON p.ProductKey = s.ProductKey -- 29481 row(s) returned
)
SELECT
    CustomerKey,
    ProductSubcategoryKey,
    ROUND(AVG(DATEDIFF(OrderDate , PreviousDate)),0) AS AvgOrderDays
FROM subcategory_orders
WHERE PreviousDate IS NOT NULL
GROUP BY 1,2
HAVING COUNT(*) > 1
ORDER BY 3 DESC; -- 832 row(s) returned
```

CustomerKey	ProductSubcategoryKey	AvgOrderDays
11086	37	86
11276	29	76
11507	21	75
11253	31	74
12203	28	72
15059	28	69
16497	37	69
11185	21	68
11310	37	67
11300	31	65
11500	21	64
11724	28	63
16167	37	62
15340	37	61
11417	3	60
11420	3	59
11420	31	59

🔶 Question 2: Longest Out-of-Stock Duration

🧩 Problem Statement :

Find the product that remained out of stock for the longest time and calculate how many days it was unavailable.

```
WITH Stock_Changes AS (
    SELECT
        ProductKey,
        StockDate,
        LEAD(StockDate) OVER(PARTITION BY ProductKey ORDER BY StockDate) AS NextStockDate
    FROM Sales2017
)
SELECT * FROM Stock_Changes;
```

ProductKey	StockDate	NextStockDate
214	2003-09-05	2003-09-06
214	2003-09-06	2003-09-09
214	2003-09-09	2003-09-12
214	2003-09-12	2003-09-12
214	2003-09-12	2003-09-14
214	2003-09-14	2003-09-15
214	2003-09-15	2003-09-15
214	2003-09-15	2003-09-15
214	2003-09-15	2003-09-17
214	2003-09-17	2003-09-18
214	2003-09-18	2003-09-19
214	2003-09-19	2003-09-19
214	2003-09-19	2003-09-19
214	2003-09-19	2003-09-20
214	2003-09-20	2003-09-21
214	2003-09-21	2003-09-21
214	2003-09-21	2003-09-22
214	2003-09-22	2003-09-23
214	2003-09-23	2003-09-24
214	2003-09-24	2003-09-24

ProductKey	StockDate	NextStockDate
215	2003-12-19	2003-12-19
215	2003-12-19	2003-12-19
215	2003-12-19	2003-12-20
215	2003-12-20	2003-12-20
215	2003-12-20	2003-12-20
215	2003-12-20	2003-12-20
215	2003-12-20	2003-12-20
215	2003-12-20	2003-12-21
215	2003-12-21	2003-12-21
215	2003-12-21	2003-12-21
215	2003-12-21	2003-12-21
215	2003-12-21	2003-12-21
215	2003-12-21	2003-12-22
215	2003-12-22	2003-12-22
215	2003-12-22	2003-12-22
215	2003-12-22	2003-12-22
215	2003-12-22	2003-12-22
215	2003-12-22	2003-12-23
215	2003-12-23	2003-12-23
215	2003-12-23	2003-12-23

```sql
WITH Stock_Changes AS (
    SELECT
        ProductKey,
        StockDate,
        LEAD(StockDate) OVER(PARTITION BY ProductKey ORDER BY StockDate) AS NextStockDate
    FROM Sales2017
)
SELECT
    ProductKey,
    MAX(DATEDIFF(NextStockDate,StockDate)) AS days_out_of_stock
FROM Stock_Changes
GROUP BY 1
ORDER BY 2 DESC;
```

ProductKey	days_out_of_stock
566	54
571	50
591	43
570	42
586	42
565	39
572	38
588	37
600	34
568	33
590	31
593	31
592	28
596	28
379	27
577	27
597	27
569	26
574	26
604	26
375	25

🔶 Question 3: Yearly Sales Trend with % Change
🧩 Problem Statement :
Show total sales per year and calculate the percentage growth or decline from the previous year.

🔶 Question 4: Row-wise Value Insights
🧩 Problem Statement :
For each sale, show the first, last, and third sale amount within its product's transaction history.