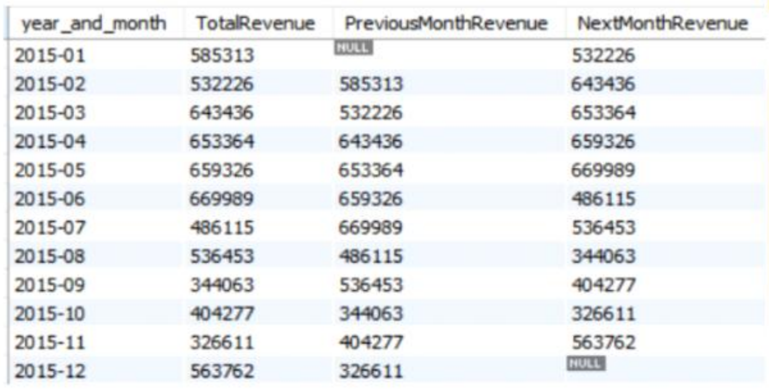
Window Functions-III

🎯 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.

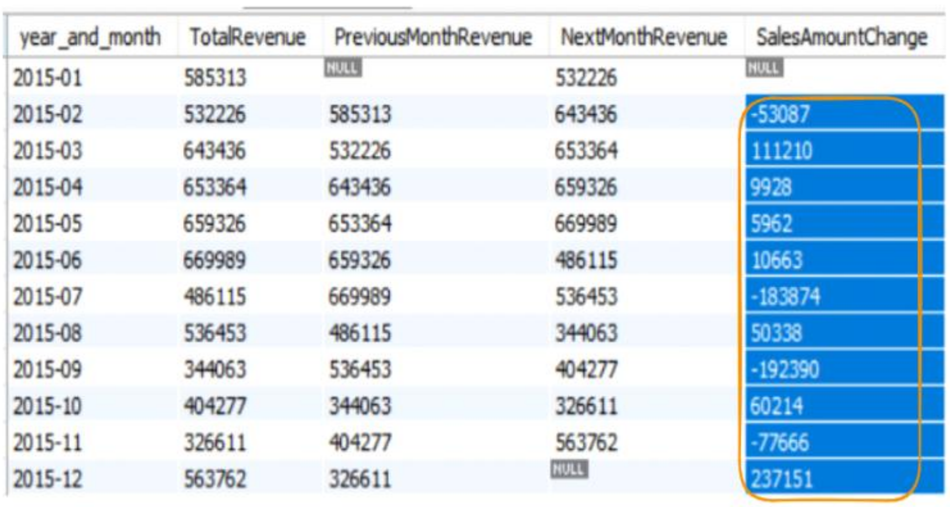
Challenge1



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

sales_amount_change

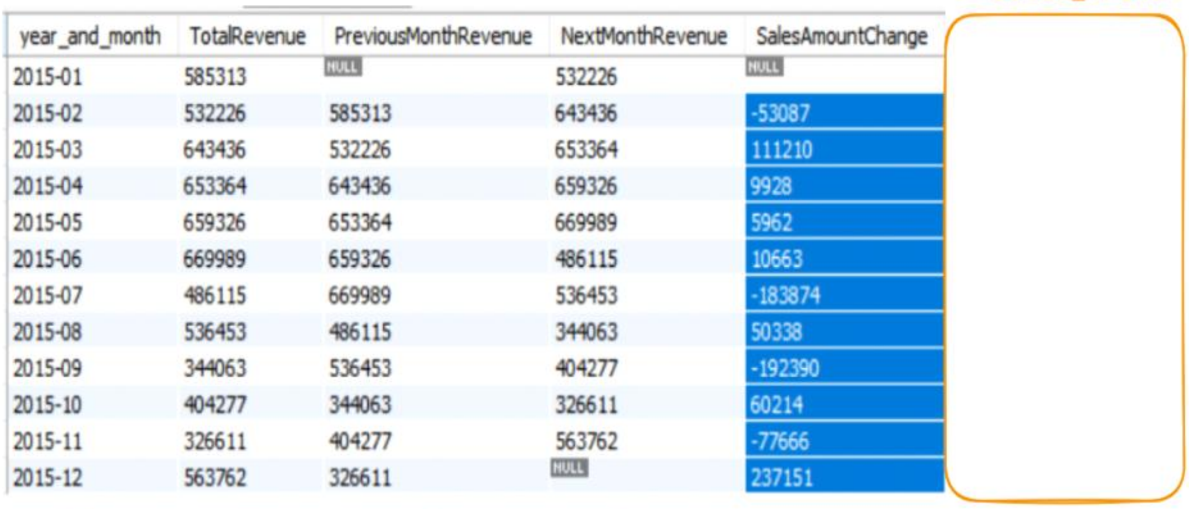
```
-- From the above code create a new column "SalesAmountChange"
WITH Understanding_Revenue AS (
    SELECT
        DATE_FORMAT(s.OrderDate, "%Y-%m") AS year_and_month,
        ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
    FROM `sales-2015` s
    JOIN Products p
    ON p.ProductKey = 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,
    TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY year_and_month) AS SalesAmountChange
FROM Understanding_Revenue;
```



year_and_month	TotalRevenue	PreviousMonthRevenue	NextMonthRevenue	SalesAmountChange
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

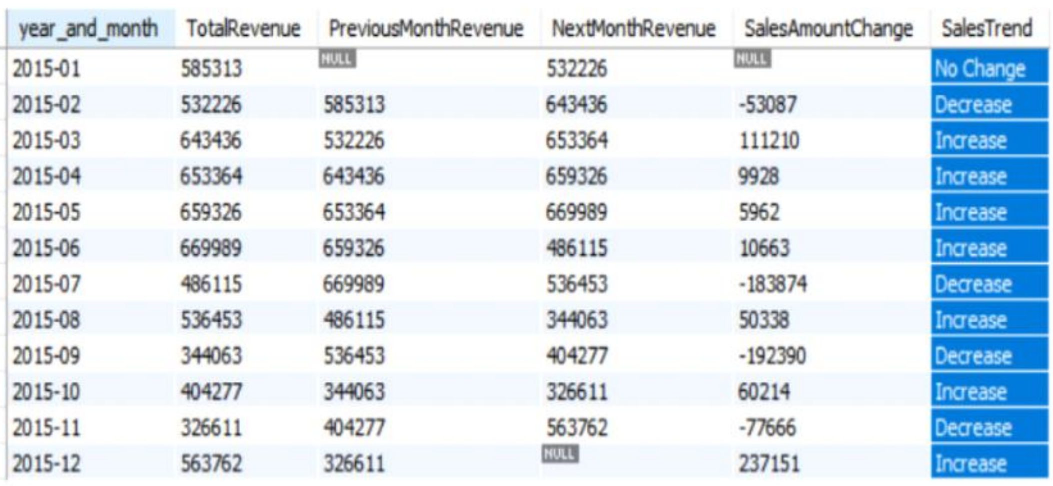
Challenge2

Sales_trend



year_and_month	TotalRevenue	PreviousMonthRevenue	NextMonthRevenue	SalesAmountChange
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

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



year_and_month	TotalRevenue	PreviousMonthRevenue	NextMonthRevenue	SalesAmountChange	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

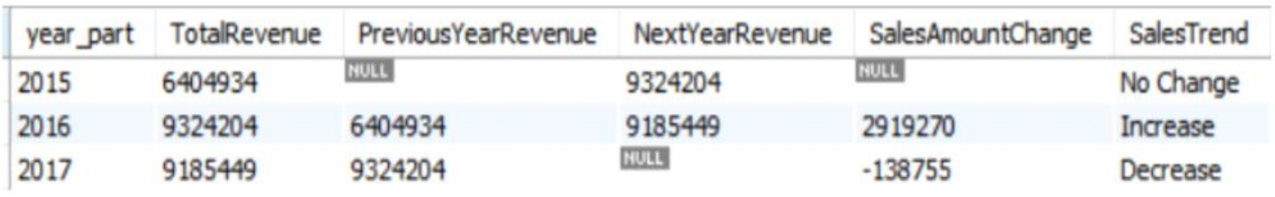
```
-- Add a Sales Trend
WITH Understanding_Revenue AS (
    SELECT
        DATE_FORMAT(s.OrderDate, "%Y-%m") AS year_and_month,
        ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
    FROM `sales-2015` s
    JOIN Products p
    ON p.ProductKey = s.ProductKey
    GROUP BY year_and_month
    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,
    TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY year_and_month) AS SalesAmountChange,
    CASE
        WHEN TotalRevenue > LAG(TotalRevenue) OVER(ORDER BY year_and_month) THEN 'Increase'
        WHEN TotalRevenue < LAG(TotalRevenue) OVER(ORDER BY year_and_month) THEN 'Decrease'
        ELSE 'No Change'
    END AS SalesTrend
FROM Understanding_Revenue;
```

```
-- Add a Sales Trend [Alternative Approach]
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,
    TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY year_and_month) AS SalesAmountChange,
    CASE
        WHEN TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY year_and_month) > 0 THEN 'Increase'
        WHEN TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY year_and_month) < 0 THEN 'Decrease'
        ELSE 'No Change'
    END AS SalesTrend
FROM Understanding_Revenue;
```

Challenge3

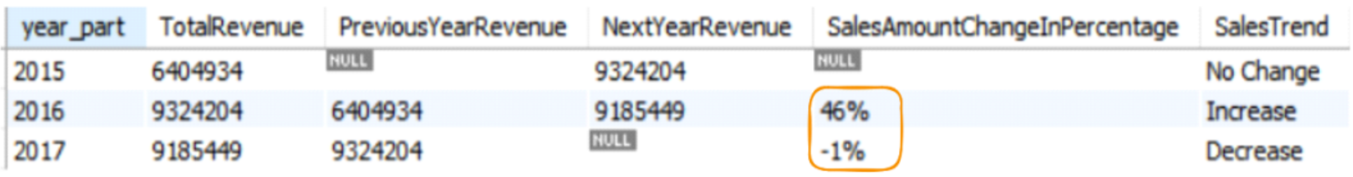
Analyse the company's year-over-year sales trend and classify each year as "Increased", "Decreased", or "No Change" compared to the previous year.

```
WITH AllSales AS (
    SELECT * FROM `sales-2015`
    UNION ALL
    SELECT * FROM `sales-2016`
    UNION ALL
    SELECT * FROM `sales-2017`
),
Understanding_Revenue AS (
    SELECT
        YEAR(s.OrderDate) AS year_part,
        ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
    FROM AllSales s
    JOIN Products p
    ON p.ProductKey = s.ProductKey
    GROUP BY year_part
    ORDER BY year_part
)
SELECT
    year_part,
    TotalRevenue,
    LAG(TotalRevenue) OVER(ORDER BY year_part) AS PreviousYearRevenue,
    LEAD(TotalRevenue) OVER(ORDER BY year_part) AS NextYearRevenue,
    TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY year_part) AS SalesAmountChange,
    CASE
        WHEN TotalRevenue > LAG(TotalRevenue) OVER(ORDER BY year_part) THEN 'Increase'
        WHEN TotalRevenue < LAG(TotalRevenue) OVER(ORDER BY year_part) THEN 'Decrease'
        ELSE 'No Change'
    END AS SalesTrend
FROM Understanding_Revenue;
```



year_part	TotalRevenue	PreviousYearRevenue	NextYearRevenue	SalesAmountChange	SalesTrend
2015	6404934	NULL	9324204	NULL	No Change
2016	9324204	6404934	9185449	2919270	Increase
2017	9185449	9324204	NULL	-138755	Decrease

```
WITH AllSales AS (
    SELECT * FROM `sales-2015`
    UNION ALL
    SELECT * FROM `sales-2016`
    UNION ALL
    SELECT * FROM `sales-2017`
),
Understanding_Revenue AS (
    SELECT
        YEAR(s.OrderDate) AS year_part,
        ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
    FROM AllSales s
    JOIN Products p
    ON p.ProductKey = s.ProductKey
    GROUP BY year_part
    ORDER BY year_part
)
SELECT
    year_part,
    TotalRevenue,
    LAG(TotalRevenue) OVER(ORDER BY year_part) AS PreviousYearRevenue,
    LEAD(TotalRevenue) OVER(ORDER BY year_part) AS NextYearRevenue,
    CONCAT(ROUND(((TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY year_part)) /
    LAG(TotalRevenue) OVER(ORDER BY year_part)) * 100,0), "%")  AS SalesAmountChangeInPercentage,
    CASE
        WHEN TotalRevenue > LAG(TotalRevenue) OVER(ORDER BY year_part) THEN 'Increase'
        WHEN TotalRevenue < LAG(TotalRevenue) OVER(ORDER BY year_part) THEN 'Decrease'
        ELSE 'No Change'
    END AS SalesTrend
FROM Understanding_Revenue;
```

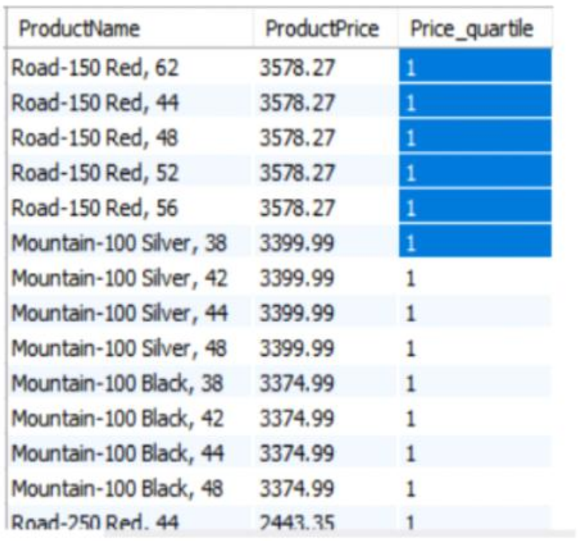


year_part	TotalRevenue	PreviousYearRevenue	NextYearRevenue	SalesAmountChangeInPercentage	SalesTrend
2015	6404934	NULL	9324204	NULL	No Change
2016	9324204	6404934	9185449	46%	Increase
2017	9185449	9324204	NULL	-1%	Decrease

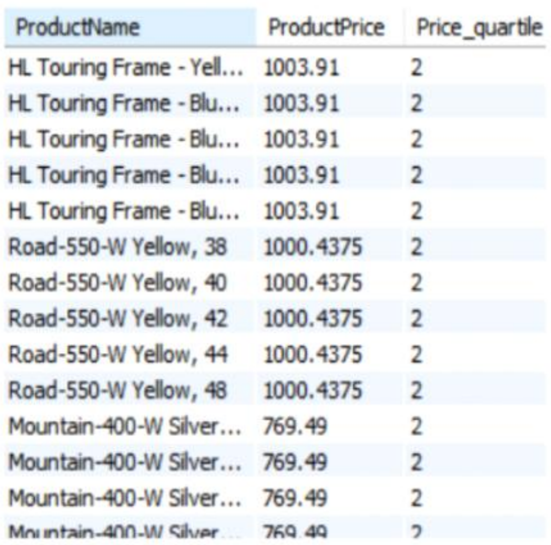
$\boxed{\text{NTILE()}} \longrightarrow$ using this to bucket our data into 'n' group

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

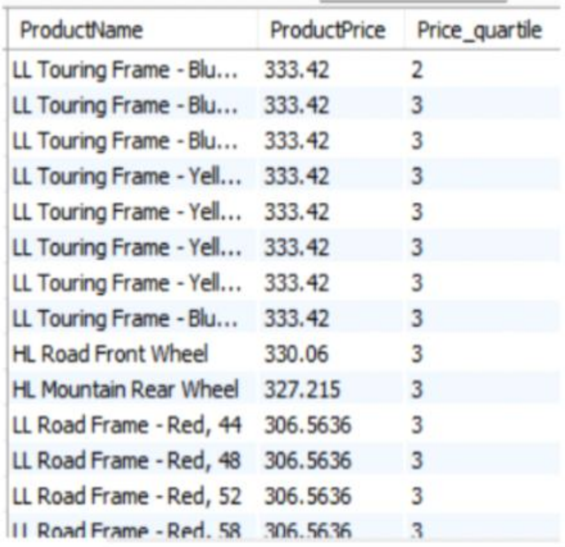
```
- SPLIT THE ProductPrice Column into quartile ->
SELECT
    ProductName,
    ProductPrice,
    NTILE(4) OVER(ORDER BY ProductPrice DESC) AS Price_quartile
FROM Products;
```



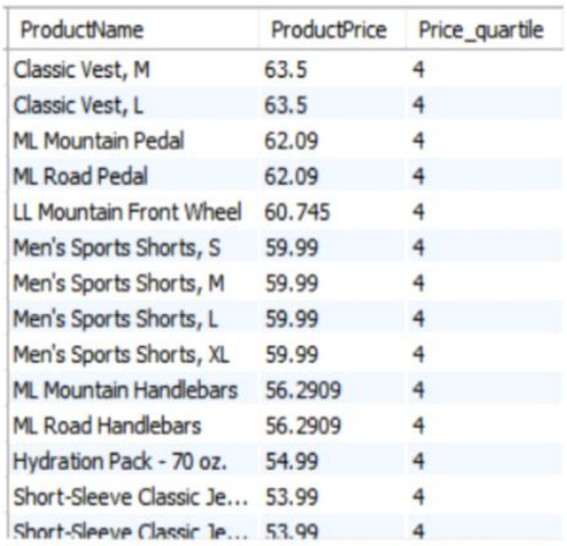
ProductName	ProductPrice	Price_quartile
Road-150 Red, 62	3578.27	1
Road-150 Red, 44	3578.27	1
Road-150 Red, 48	3578.27	1
Road-150 Red, 52	3578.27	1
Road-150 Red, 56	3578.27	1
Mountain-100 Silver, 38	3399.99	1
Mountain-100 Silver, 42	3399.99	1
Mountain-100 Silver, 44	3399.99	1
Mountain-100 Silver, 48	3399.99	1
Mountain-100 Black, 38	3374.99	1
Mountain-100 Black, 42	3374.99	1
Mountain-100 Black, 44	3374.99	1
Mountain-100 Black, 48	3374.99	1
Road-250 Red, 44	2443.35	1



ProductName	ProductPrice	Price_quartile
HL Touring Frame - Yell...	1003.91	2
HL Touring Frame - Blu...	1003.91	2
HL Touring Frame - Blu...	1003.91	2
HL Touring Frame - Blu...	1003.91	2
HL Touring Frame - Blu...	1003.91	2
Road-550-W Yellow, 38	1000.4375	2
Road-550-W Yellow, 40	1000.4375	2
Road-550-W Yellow, 42	1000.4375	2
Road-550-W Yellow, 44	1000.4375	2
Road-550-W Yellow, 48	1000.4375	2
Mountain-400-W Silver...	769.49	2
Mountain-400-W Silver...	769.49	2
Mountain-400-W Silver...	769.49	2
Mountain-400-W Silver...	769.49	2



ProductName	ProductPrice	Price_quartile
LL Touring Frame - Blu...	333.42	2
LL Touring Frame - Blu...	333.42	3
LL Touring Frame - Blu...	333.42	3
LL Touring Frame - Yell...	333.42	3
LL Touring Frame - Yell...	333.42	3
LL Touring Frame - Yell...	333.42	3
LL Touring Frame - Yell...	333.42	3
LL Touring Frame - Blu...	333.42	3
HL Road Front Wheel	330.06	3
HL Mountain Rear Wheel	327.215	3
LL Road Frame - Red, 44	306.5636	3
LL Road Frame - Red, 48	306.5636	3
LL Road Frame - Red, 52	306.5636	3
LL Road Frame - Red, 58	306.5636	3

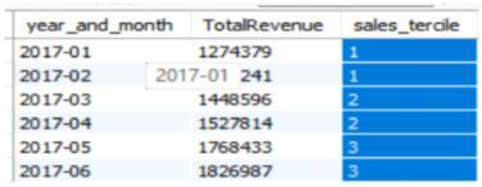


ProductName	ProductPrice	Price_quartile
Classic Vest, M	63.5	4
Classic Vest, L	63.5	4
ML Mountain Pedal	62.09	4
ML Road Pedal	62.09	4
LL Mountain Front Wheel	60.745	4
Men's Sports Shorts, S	59.99	4
Men's Sports Shorts, M	59.99	4
Men's Sports Shorts, L	59.99	4
Men's Sports Shorts, XL	59.99	4
ML Mountain Handlebars	56.2909	4
ML Road Handlebars	56.2909	4
Hydration Pack - 70 oz.	54.99	4
Short-Sleeve Classic Je...	53.99	4
Short-Sleeve Classic Je...	53.99	4

Challenge 4

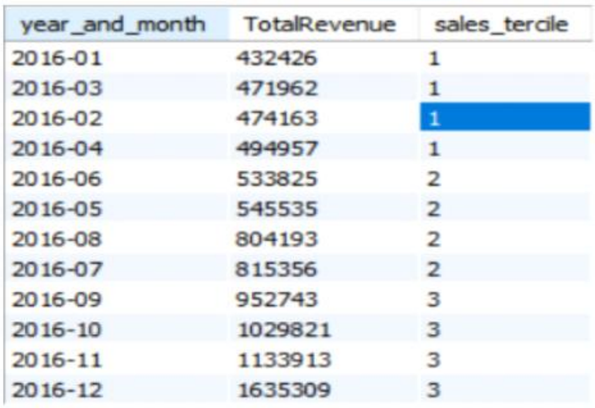
Divide each month's total sales into 3 performance tiers (terciles) to categorise months as low, medium, or high-performing.

```
WITH MontlySales AS (
        SELECT
            DATE_FORMAT(s.OrderDate, "%Y-%m") AS year_and_month,
            ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
        FROM `sales-2017` s
        JOIN Products p
        ON p.ProductKey = s.ProductKey
    GROUP BY year_and_month
    ORDER BY year_and_month
)
SELECT
    year_and_month,
    TotalRevenue,
    NTILE(3) OVER(ORDER BY TotalRevenue) AS sales_tercile
FROM MontlySales;
```



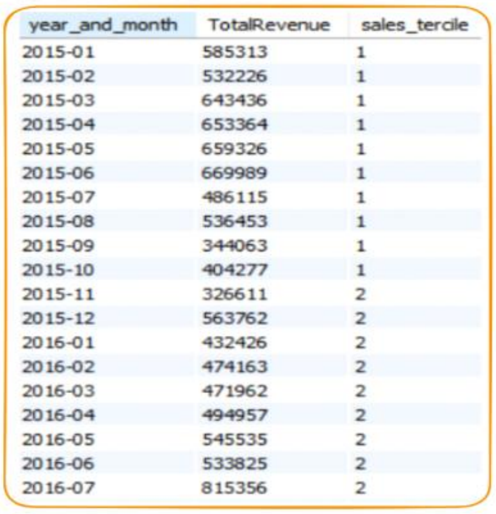
year_and_month	TotalRevenue	sales_tercile
2017-01	1274379	1
2017-02	[illegible]241	1
2017-03	1448596	2
2017-04	1527814	2
2017-05	1768433	3
2017-06	1826987	3

sales-2016



year_and_month	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

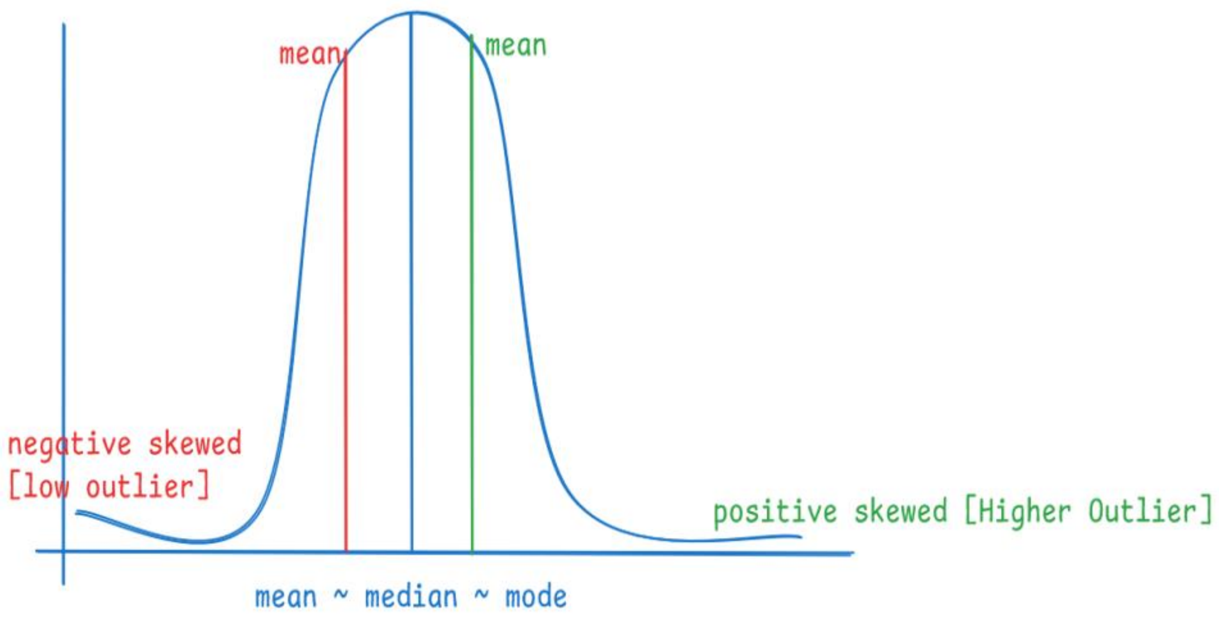
```
WITH MontlySales AS (
    SELECT
        DATE_FORMAT(s.OrderDate, "%Y-%m") AS year_and_month,
        ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
    FROM (
        SELECT * FROM `sales-2015`
        UNION ALL
        SELECT * FROM `sales-2016`
        UNION ALL
        SELECT * FROM `sales-2017`
    ) AS s
    JOIN Products p
    ON p.ProductKey = s.ProductKey
    GROUP BY year_and_month
    ORDER BY year_and_month
)
SELECT
    year_and_month,
    TotalRevenue,
    NTILE(3) OVER() AS sales_tercile
FROM MontlySales;
```

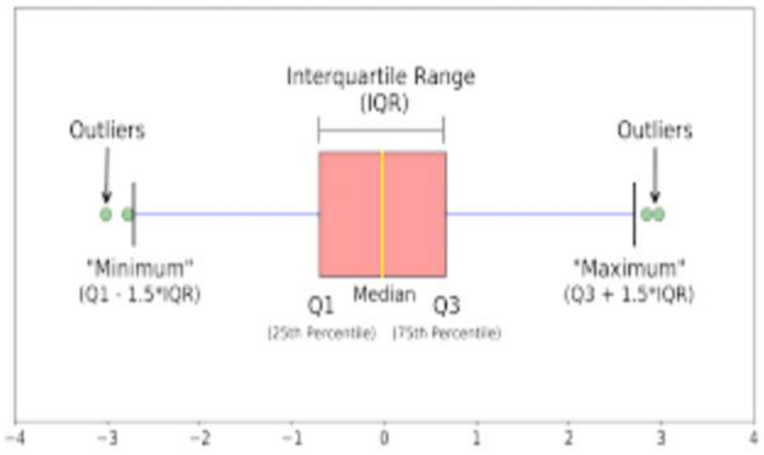


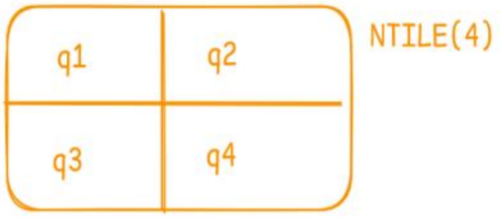
year_and_month	TotalRevenue	sales_tercile
2015-01	585313	1
2015-02	532226	1
2015-03	643436	1
2015-04	653364	1
2015-05	659326	1
2015-06	669989	1
2015-07	486115	1
2015-08	536453	1
2015-09	344063	1
2015-10	404277	1
2015-11	326611	2
2015-12	563762	2
2016-01	432426	2
2016-02	474163	2
2016-03	471962	2
2016-04	494957	2
2016-05	545535	2
2016-06	533825	2
2016-07	815356	2

Challenge 5

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



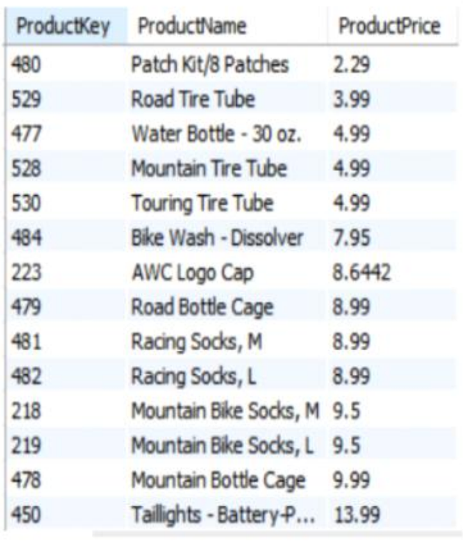




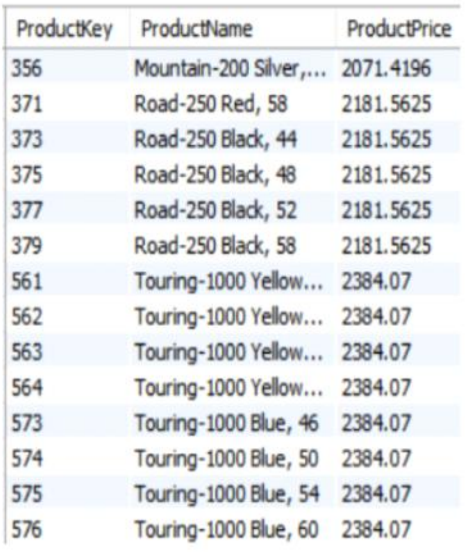
293 row(s) returned

277 row(s) returned

```
-- Filter the products lying within the lower bound & 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 iqb
ON p.ProductPrice BETWEEN iqb.lower_bound AND iqb.upper_bound
ORDER BY ProductPrice;
```



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

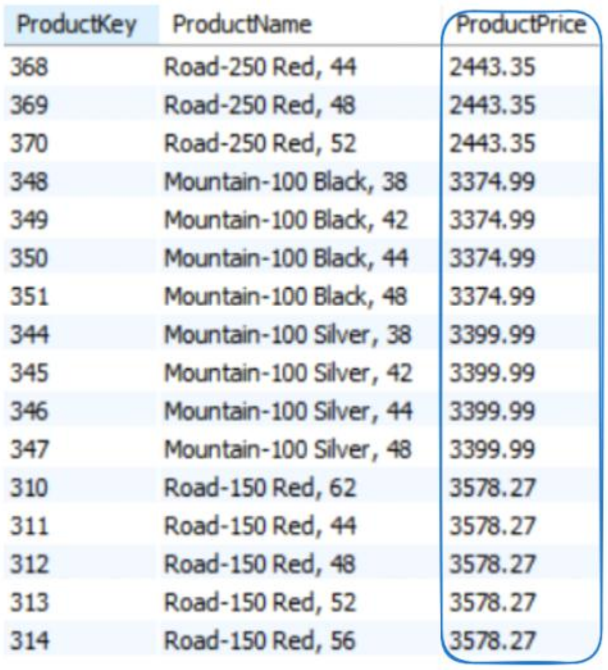


ProductKey	ProductName	ProductPrice
356	Mountain-200 Silver,...	2071.4196
371	Road-250 Red, 58	2181.5625
373	Road-250 Black, 44	2181.5625
375	Road-250 Black, 48	2181.5625
377	Road-250 Black, 52	2181.5625
379	Road-250 Black, 58	2181.5625
561	Touring-1000 Yellow...	2384.07
562	Touring-1000 Yellow...	2384.07
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

```
-- Detecting The Outlier
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 iqb
ON p.ProductPrice < iqb.lower_bound OR p.ProductPrice > iqb.upper_bound
ORDER BY ProductPrice;
```

low outliers

high outliers



ProductKey	ProductName	ProductPrice
368	Road-250 Red, 44	2443.35
369	Road-250 Red, 48	2443.35
370	Road-250 Red, 52	2443.35
348	Mountain-100 Black, 38	3374.99
349	Mountain-100 Black, 42	3374.99
350	Mountain-100 Black, 44	3374.99
351	Mountain-100 Black, 48	3374.99
344	Mountain-100 Silver, 38	3399.99
345	Mountain-100 Silver, 42	3399.99
346	Mountain-100 Silver, 44	3399.99
347	Mountain-100 Silver, 48	3399.99
310	Road-150 Red, 62	3578.27
311	Road-150 Red, 44	3578.27
312	Road-150 Red, 48	3578.27
313	Road-150 Red, 52	3578.27
314	Road-150 Red, 56	3578.27

Product Table is Positive Skewed by detecting the high outliers.