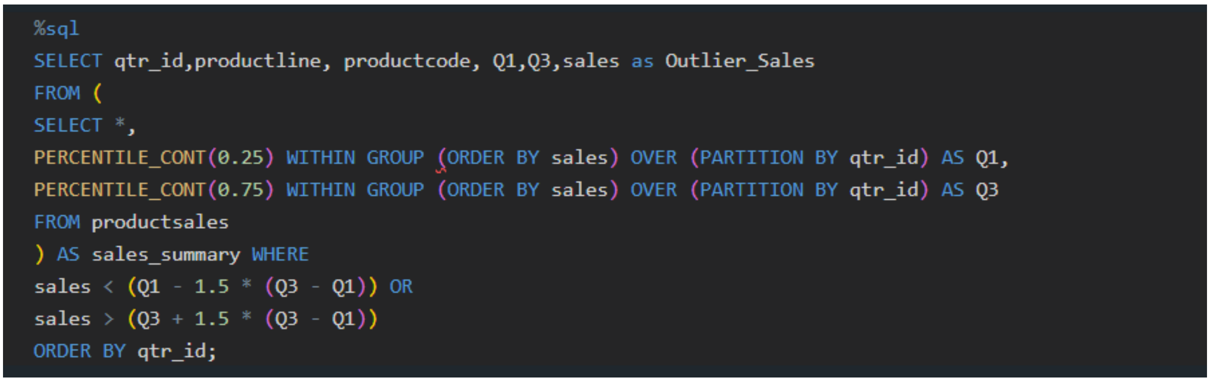
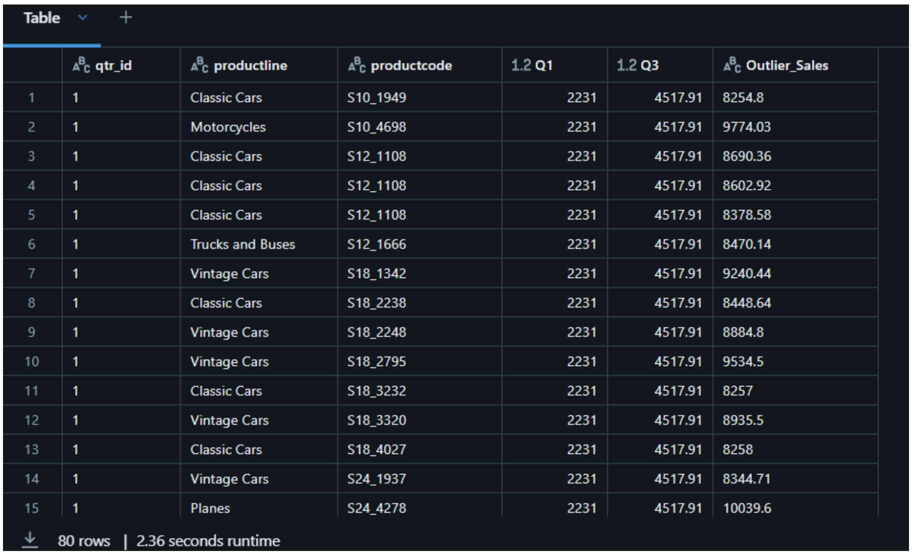
Assignment 11 – Product Sales Analysis and Visualization

1. Identify Product with outlier sales in each quarter.

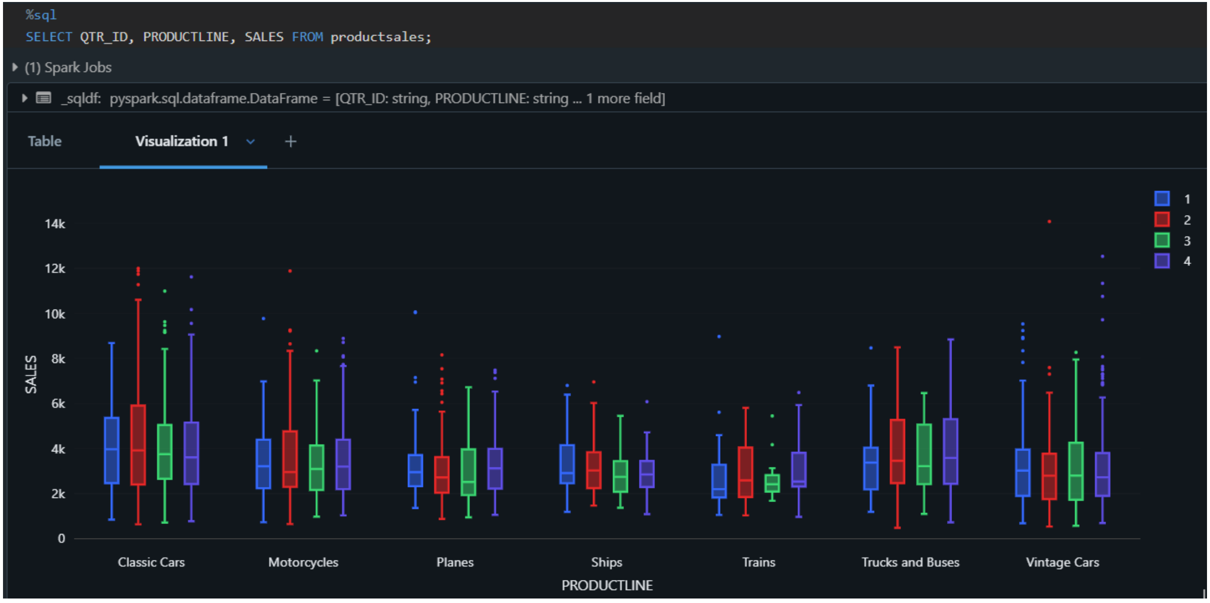


```
%sql
SELECT qtr_id,productline, productcode, Q1,Q3,sales as Outlier_Sales
FROM (
SELECT *,
PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY sales) OVER (PARTITION BY qtr_id) AS Q1,
PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY sales) OVER (PARTITION BY qtr_id) AS Q3
FROM productsales
) AS sales_summary WHERE
sales < (Q1 - 1.5 * (Q3 - Q1)) OR
sales > (Q3 + 1.5 * (Q3 - Q1))
ORDER BY qtr_id;
```



	qtr_id	productline	productcode	Q1	Q3	Outlier_Sales
1	1	Classic Cars	S10_1949	2231	4517.91	8254.8
2	1	Motorcycles	S10_4698	2231	4517.91	9774.03
3	1	Classic Cars	S12_1108	2231	4517.91	8690.36
4	1	Classic Cars	S12_1108	2231	4517.91	8602.92
5	1	Classic Cars	S12_1108	2231	4517.91	8378.58
6	1	Trucks and Buses	S12_1666	2231	4517.91	8470.14
7	1	Vintage Cars	S18_1342	2231	4517.91	9240.44
8	1	Classic Cars	S18_2238	2231	4517.91	8448.64
9	1	Vintage Cars	S18_2248	2231	4517.91	8884.8
10	1	Vintage Cars	S18_2795	2231	4517.91	9534.5
11	1	Classic Cars	S18_3232	2231	4517.91	8257
12	1	Vintage Cars	S18_3320	2231	4517.91	8935.5
13	1	Classic Cars	S18_4027	2231	4517.91	8258
14	1	Vintage Cars	S24_1937	2231	4517.91	8344.71
15	1	Planes	S24_4278	2231	4517.91	10039.6

80 rows | 2.36 seconds runtime

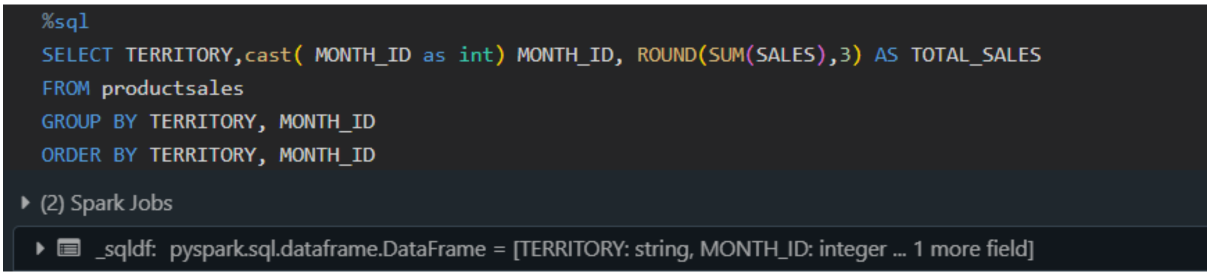


```
%sql
SELECT QTR_ID, PRODUCTLINE, SALES FROM productsales;
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [QTR_ID: string, PRODUCTLINE: string ... 1 more field]

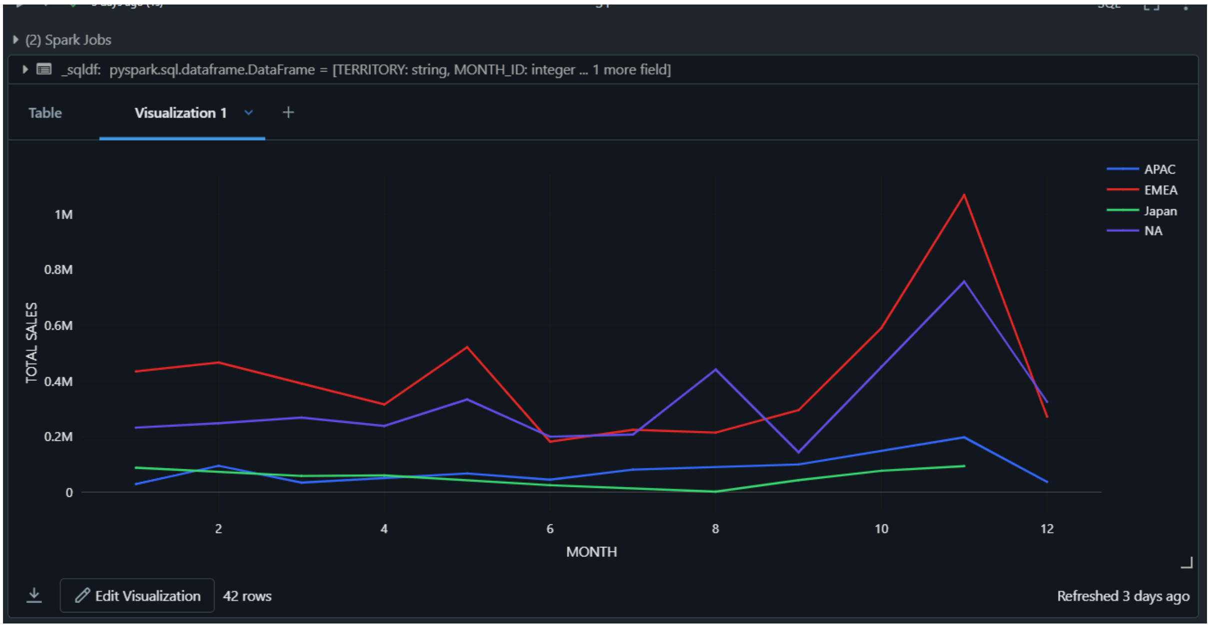
2. Monthly Sales Performance by Each Territory



```
%sql
SELECT TERRITORY,cast( MONTH_ID as int) MONTH_ID, ROUND(SUM(SALES),3) AS TOTAL_SALES
FROM productsales
GROUP BY TERRITORY, MONTH_ID
ORDER BY TERRITORY, MONTH_ID
```

▸ (2) Spark Jobs

▸ _sqldf: pyspark.sql.dataframe.DataFrame = [TERRITORY: string, MONTH_ID: integer ... 1 more field]



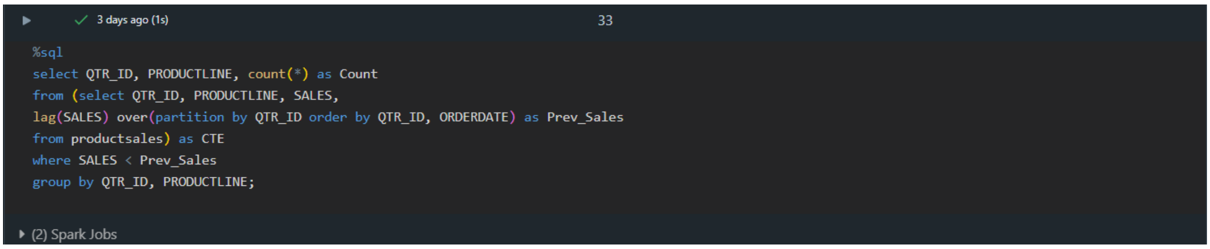
▸ (2) Spark Jobs

▸ _sqldf: pyspark.sql.dataframe.DataFrame = [TERRITORY: string, MONTH_ID: integer ... 1 more field]

42 rows

Refreshed 3 days ago

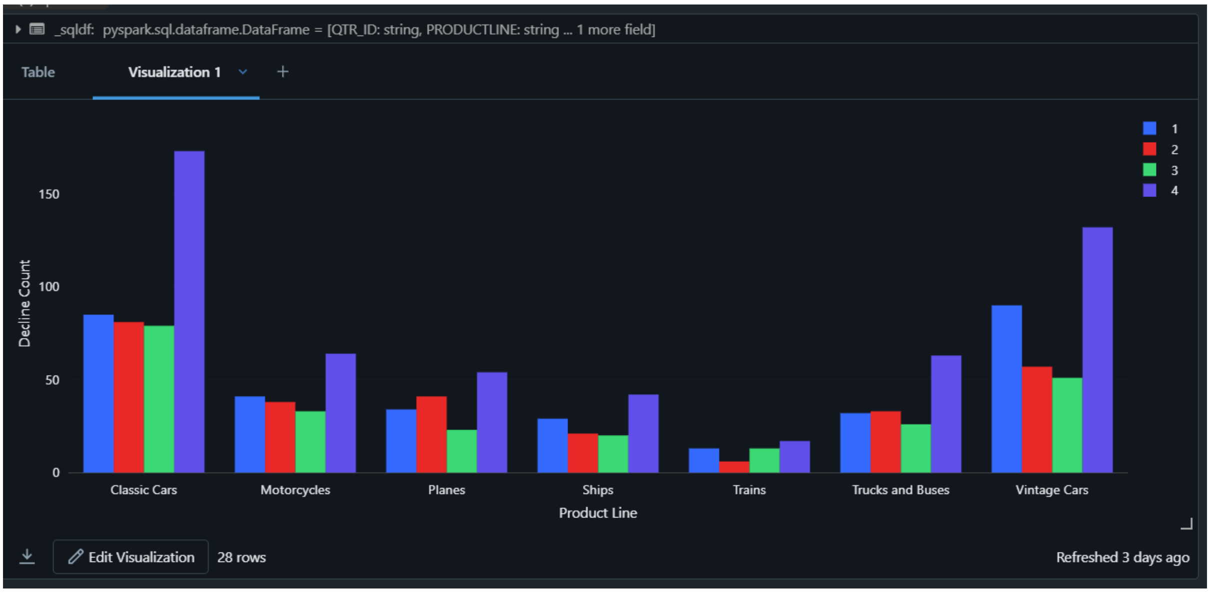
3. Identify Consecutive Quarters of Sales Decline for Each Product



✓ 3 days ago (1s) 33

```
%sql
select QTR_ID, PRODUCTLINE, count(*) as Count
from (select QTR_ID, PRODUCTLINE, SALES,
lag(SALES) over(partition by QTR_ID order by QTR_ID, ORDERDATE) as Prev_Sales
from productsales) as CTE
where SALES < Prev_Sales
group by QTR_ID, PRODUCTLINE;
```

▸ (2) Spark Jobs

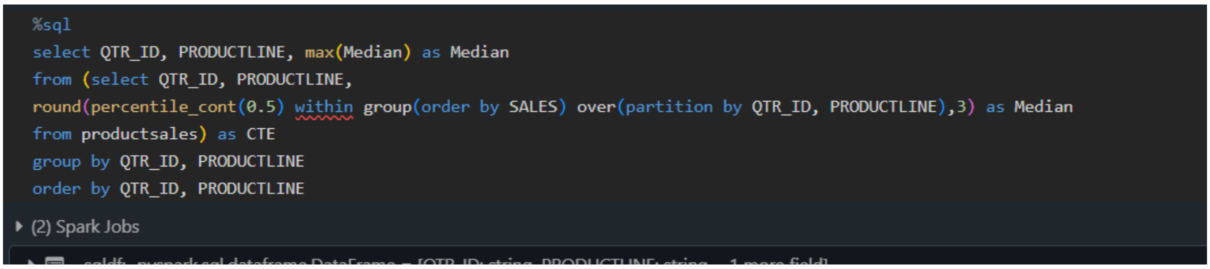


```
_sqldf: pyspark.sql.dataframe.DataFrame = [QTR_ID: string, PRODUCTLINE: string ... 1 more field]
```

28 rows

Refreshed 3 days ago

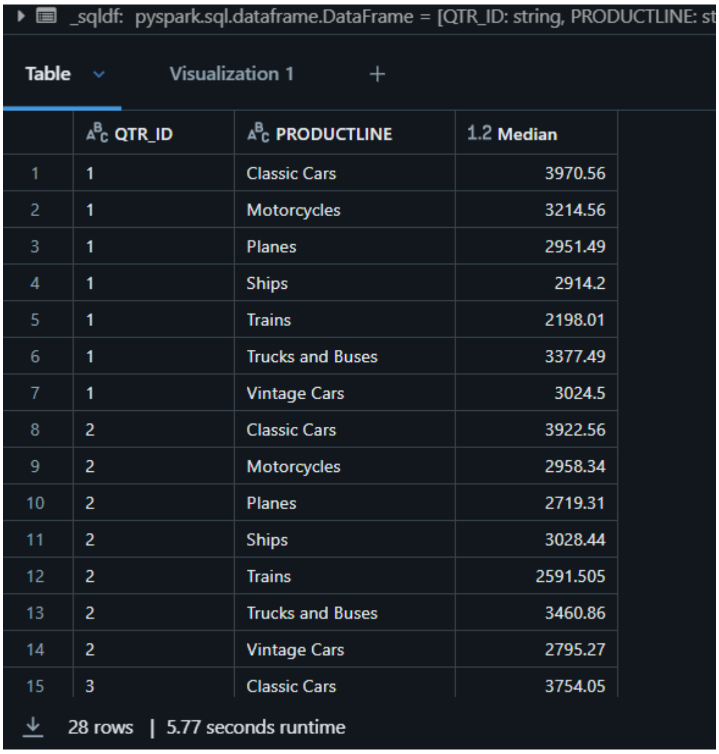
4. Median Sales for Each Product Line in Every Quarter



```
%sql
select QTR_ID, PRODUCTLINE, max(Median) as Median
from (select QTR_ID, PRODUCTLINE,
round(percentile_cont(0.5) within group(order by SALES) over(partition by QTR_ID, PRODUCTLINE),3) as Median
from productsales) as CTE
group by QTR_ID, PRODUCTLINE
order by QTR_ID, PRODUCTLINE
```

(2) Spark Jobs

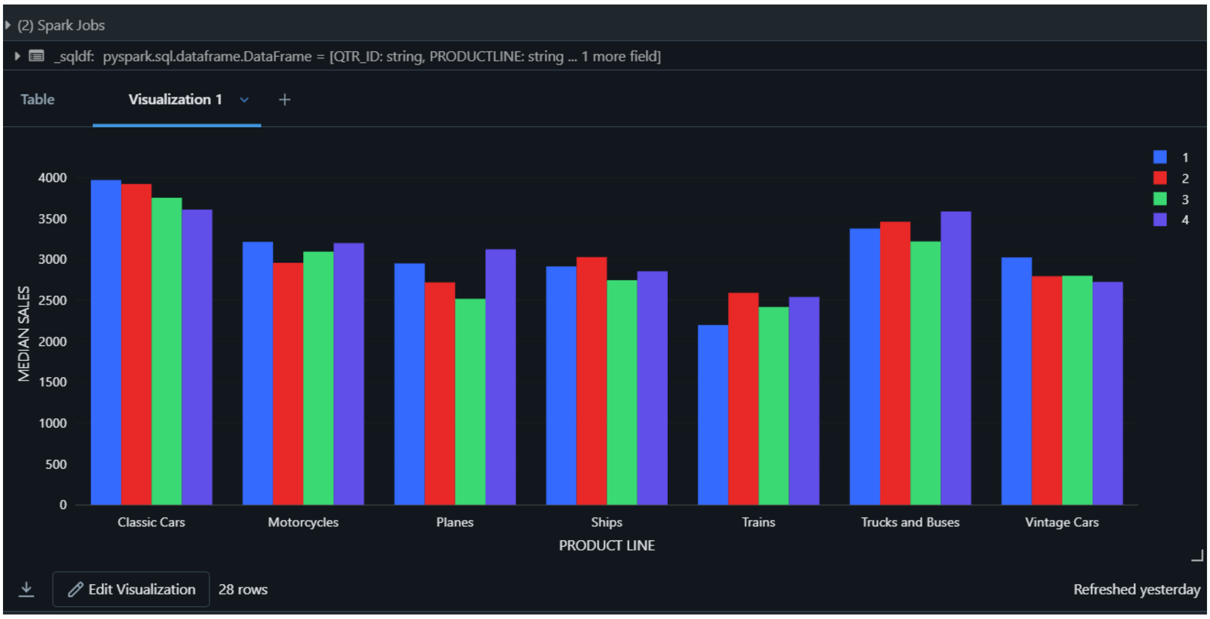
```
_sqldf: pyspark.sql.dataframe.DataFrame = [QTR_ID: string, PRODUCTLINE: string ... 1 more field]
```



```
_sqldf: pyspark.sql.dataframe.DataFrame = [QTR_ID: string, PRODUCTLINE: st
```

	QTR_ID	PRODUCTLINE	1.2 Median
1	1	Classic Cars	3970.56
2	1	Motorcycles	3214.56
3	1	Planes	2951.49
4	1	Ships	2914.2
5	1	Trains	2198.01
6	1	Trucks and Buses	3377.49
7	1	Vintage Cars	3024.5
8	2	Classic Cars	3922.56
9	2	Motorcycles	2958.34
10	2	Planes	2719.31
11	2	Ships	3028.44
12	2	Trains	2591.505
13	2	Trucks and Buses	3460.86
14	2	Vintage Cars	2795.27
15	3	Classic Cars	3754.05

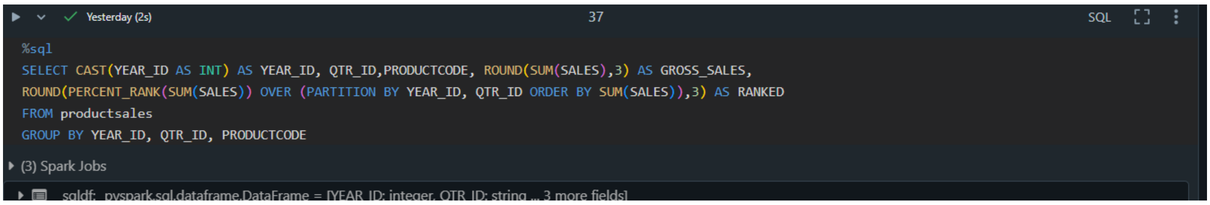
28 rows | 5.77 seconds runtime



(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [QTR_ID: string, PRODUCTLINE: string ... 1 more field]

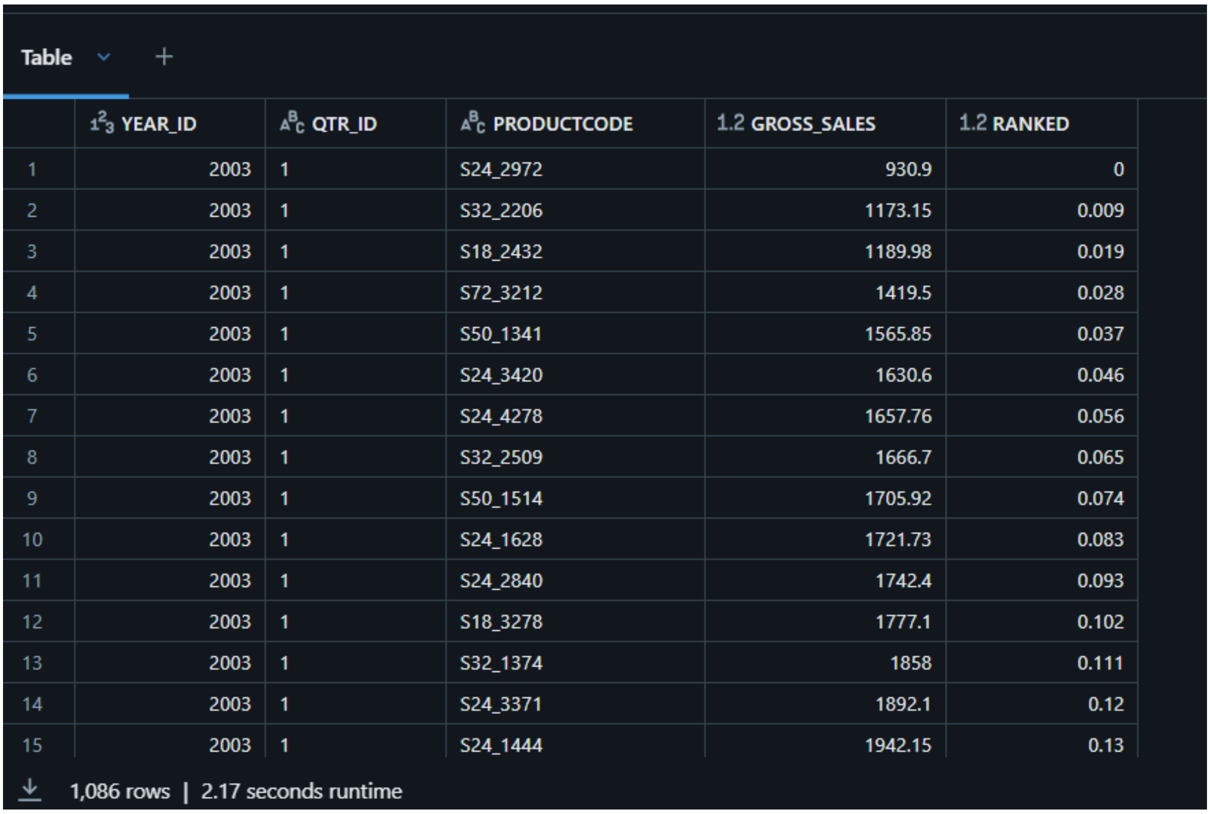
5. Percent Rank of Products Based on Gross Sales Amount in Each Quarter



```
%sql
SELECT CAST(YEAR_ID AS INT) AS YEAR_ID, QTR_ID,PRODUCTCODE, ROUND(SUM(SALES),3) AS GROSS_SALES,
ROUND(PERCENT_RANK(SUM(SALES)) OVER (PARTITION BY YEAR_ID, QTR_ID ORDER BY SUM(SALES)),3) AS RANKED
FROM productsales
GROUP BY YEAR_ID, QTR_ID, PRODUCTCODE
```

(3) Spark Jobs

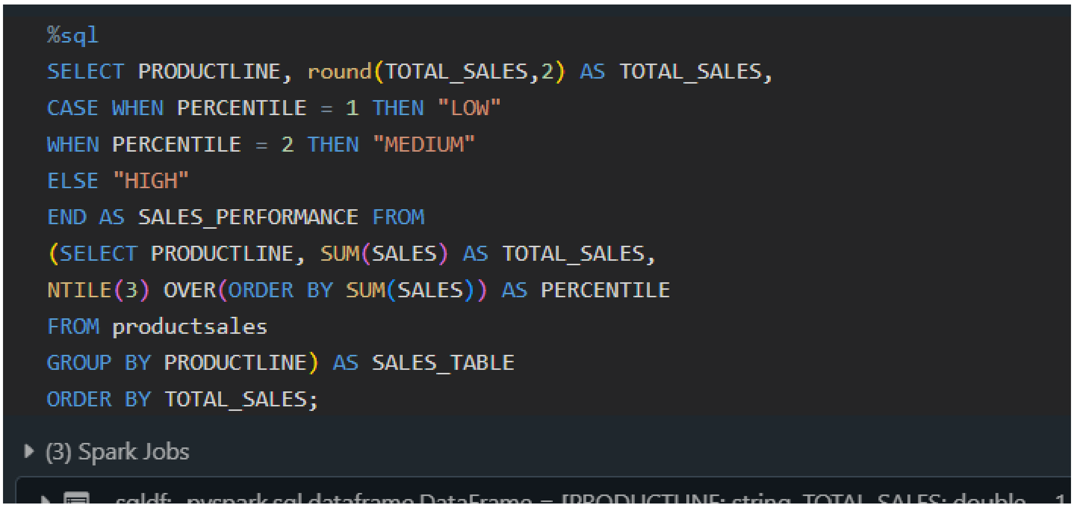
_sqldf: pyspark.sql.dataframe.DataFrame = [YEAR_ID: integer, QTR_ID: string ... 3 more fields]



	YEAR_ID	QTR_ID	PRODUCTCODE	GROSS_SALES	RANKED
1	2003	1	S24_2972	930.9	0
2	2003	1	S32_2206	1173.15	0.009
3	2003	1	S18_2432	1189.98	0.019
4	2003	1	S72_3212	1419.5	0.028
5	2003	1	S50_1341	1565.85	0.037
6	2003	1	S24_3420	1630.6	0.046
7	2003	1	S24_4278	1657.76	0.056
8	2003	1	S32_2509	1666.7	0.065
9	2003	1	S50_1514	1705.92	0.074
10	2003	1	S24_1628	1721.73	0.083
11	2003	1	S24_2840	1742.4	0.093
12	2003	1	S18_3278	1777.1	0.102
13	2003	1	S32_1374	1858	0.111
14	2003	1	S24_3371	1892.1	0.12
15	2003	1	S24_1444	1942.15	0.13

1,086 rows | 2.17 seconds runtime

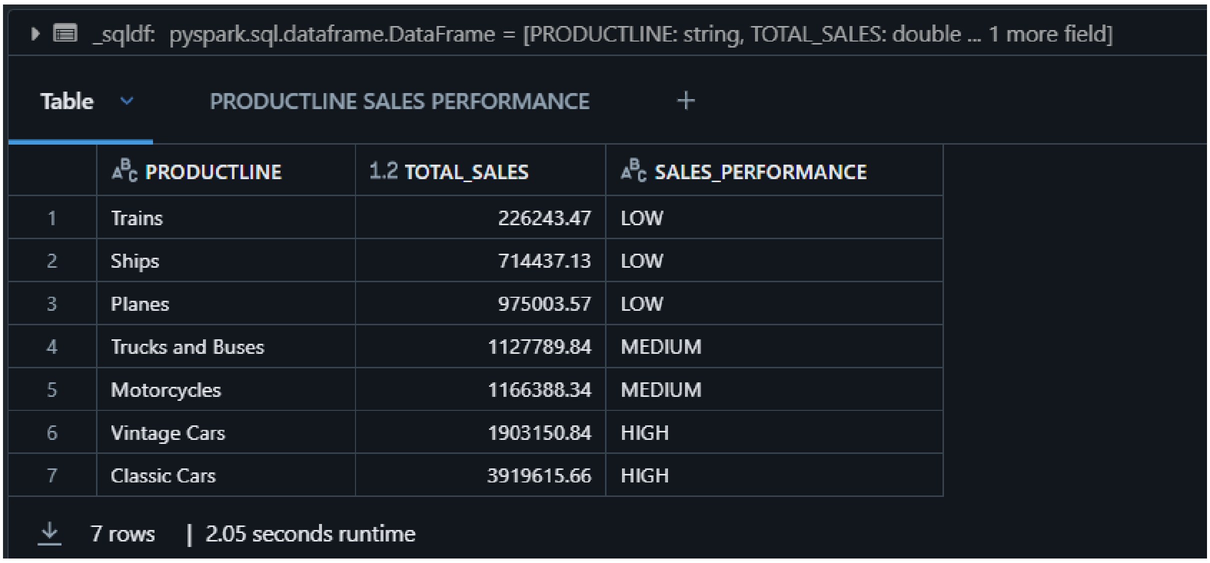
6. Categorize Product Category Sales Performance as 'High', 'Medium', or 'Low' Based on Percentiles



```
%sql
SELECT PRODUCTLINE, round(TOTAL_SALES,2) AS TOTAL_SALES,
CASE WHEN PERCENTILE = 1 THEN "LOW"
WHEN PERCENTILE = 2 THEN "MEDIUM"
ELSE "HIGH"
END AS SALES_PERFORMANCE FROM
(SELECT PRODUCTLINE, SUM(SALES) AS TOTAL_SALES,
NTILE(3) OVER(ORDER BY SUM(SALES)) AS PERCENTILE
FROM productsales
GROUP BY PRODUCTLINE) AS SALES_TABLE
ORDER BY TOTAL_SALES;
```

(3) Spark Jobs

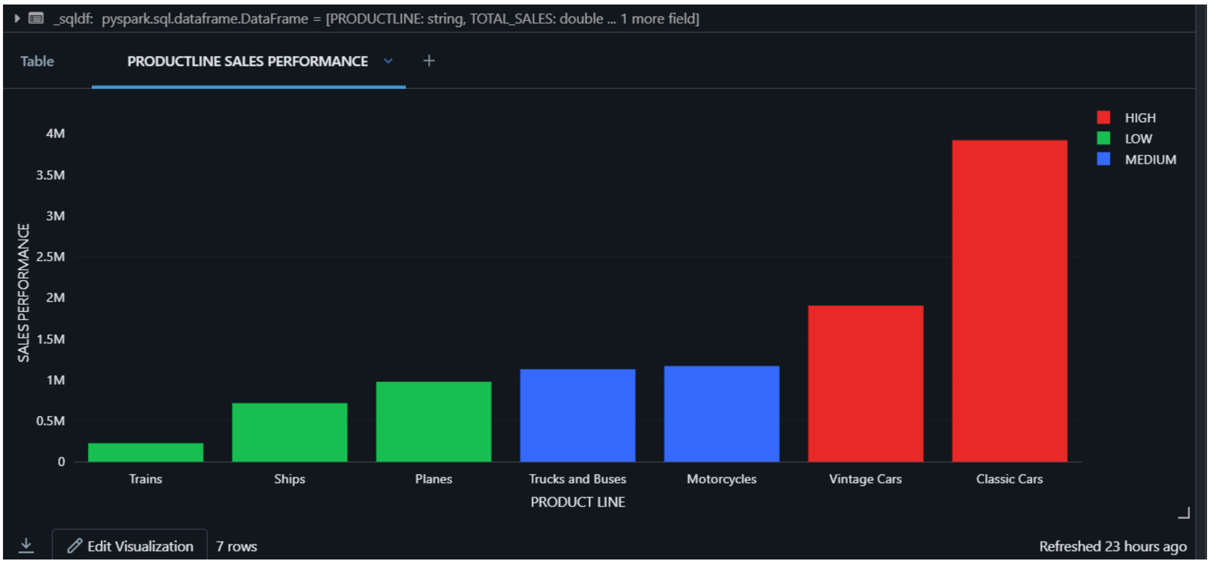
_sqldf: pyspark.sql.dataframe.DataFrame = [PRODUCTLINE: string, TOTAL_SALES: double ... 1



_sqldf: pyspark.sql.dataframe.DataFrame = [PRODUCTLINE: string, TOTAL_SALES: double ... 1 more field]

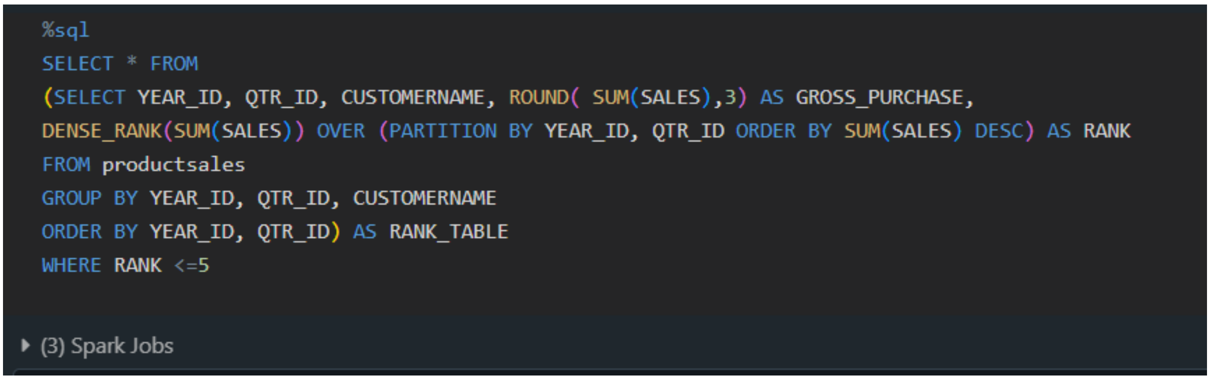
	PRODUCTLINE	TOTAL_SALES	SALES_PERFORMANCE
1	Trains	226243.47	LOW
2	Ships	714437.13	LOW
3	Planes	975003.57	LOW
4	Trucks and Buses	1127789.84	MEDIUM
5	Motorcycles	1166388.34	MEDIUM
6	Vintage Cars	1903150.84	HIGH
7	Classic Cars	3919615.66	HIGH

7 rows | 2.05 seconds runtime

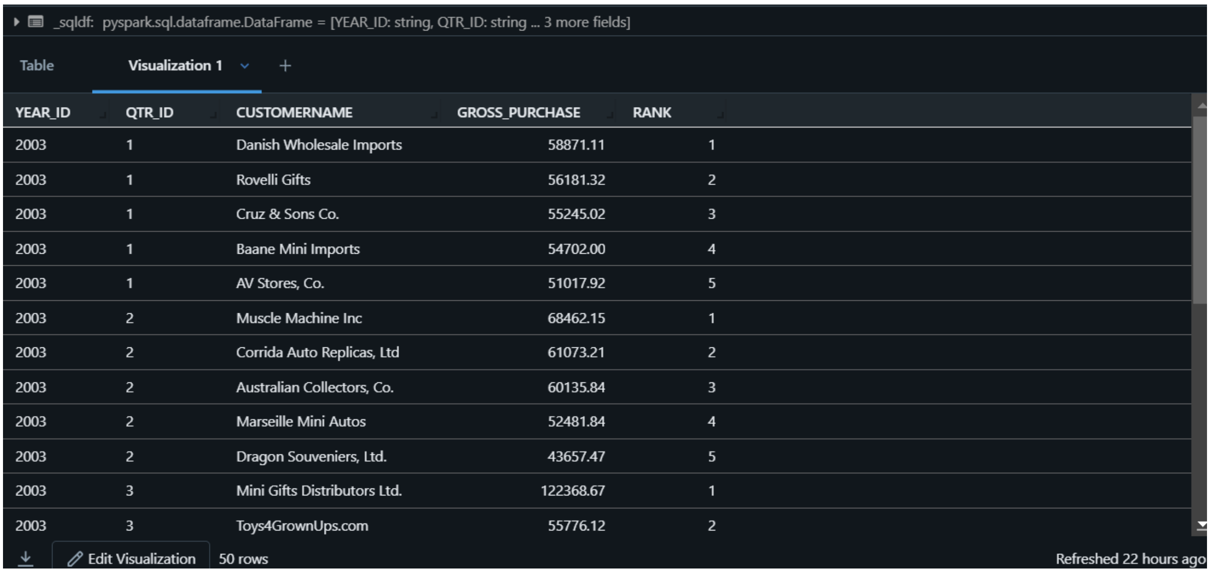


_sqldf: pyspark.sql.dataframe.DataFrame = [PRODUCTLINE: string, TOTAL_SALES: double ... 1 more field]

7. Rank 5 Customers by Total Purchase in every Quarter of Each Year.



```
%sql
SELECT * FROM
(SELECT YEAR_ID, QTR_ID, CUSTOMERNAME, ROUND( SUM(SALES),3) AS GROSS_PURCHASE,
DENSE_RANK(SUM(SALES)) OVER (PARTITION BY YEAR_ID, QTR_ID ORDER BY SUM(SALES) DESC) AS RANK
FROM productsales
GROUP BY YEAR_ID, QTR_ID, CUSTOMERNAME
ORDER BY YEAR_ID, QTR_ID) AS RANK_TABLE
WHERE RANK <=5
```



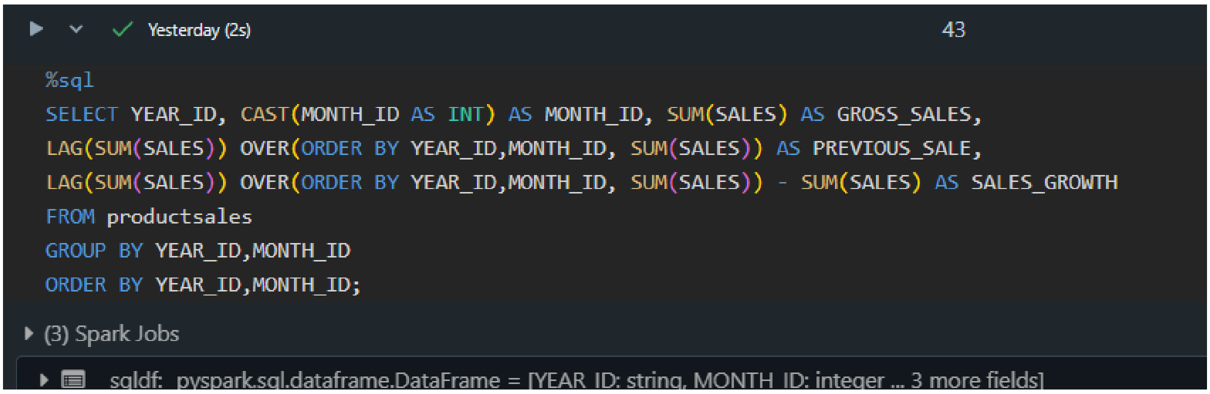
_sqldf: pyspark.sql.dataframe.DataFrame = [YEAR_ID: string, QTR_ID: string ... 3 more fields]

YEAR_ID	QTR_ID	CUSTOMERNAME	GROSS_PURCHASE	RANK
2003	1	Danish Wholesale Imports	58871.11	1
2003	1	Rovelli Gifts	56181.32	2
2003	1	Cruz & Sons Co.	55245.02	3
2003	1	Baane Mini Imports	54702.00	4
2003	1	AV Stores, Co.	51017.92	5
2003	2	Muscle Machine Inc	68462.15	1
2003	2	Corrida Auto Replicas, Ltd	61073.21	2
2003	2	Australian Collectors, Co.	60135.84	3
2003	2	Marseille Mini Autos	52481.84	4
2003	2	Dragon Souveniers, Ltd.	43657.47	5
2003	3	Mini Gifts Distributors Ltd.	122368.67	1
2003	3	Toys4GrownUps.com	55776.12	2

50 rows

Refreshed 22 hours ago

8. Identify and Present the Trends of Sales Growth by Month. (Bar Graph)



```
%sql
SELECT YEAR_ID, CAST(MONTH_ID AS INT) AS MONTH_ID, SUM(SALES) AS GROSS_SALES,
LAG(SUM(SALES)) OVER(ORDER BY YEAR_ID,MONTH_ID, SUM(SALES)) AS PREVIOUS_SALE,
LAG(SUM(SALES)) OVER(ORDER BY YEAR_ID,MONTH_ID, SUM(SALES)) - SUM(SALES) AS SALES_GROWTH
FROM productsales
GROUP BY YEAR_ID,MONTH_ID
ORDER BY YEAR_ID,MONTH_ID;
```

_sqldf: pyspark.sql.dataframe.DataFrame = [YEAR_ID: string, MONTH_ID: integer ... 3 more fields]

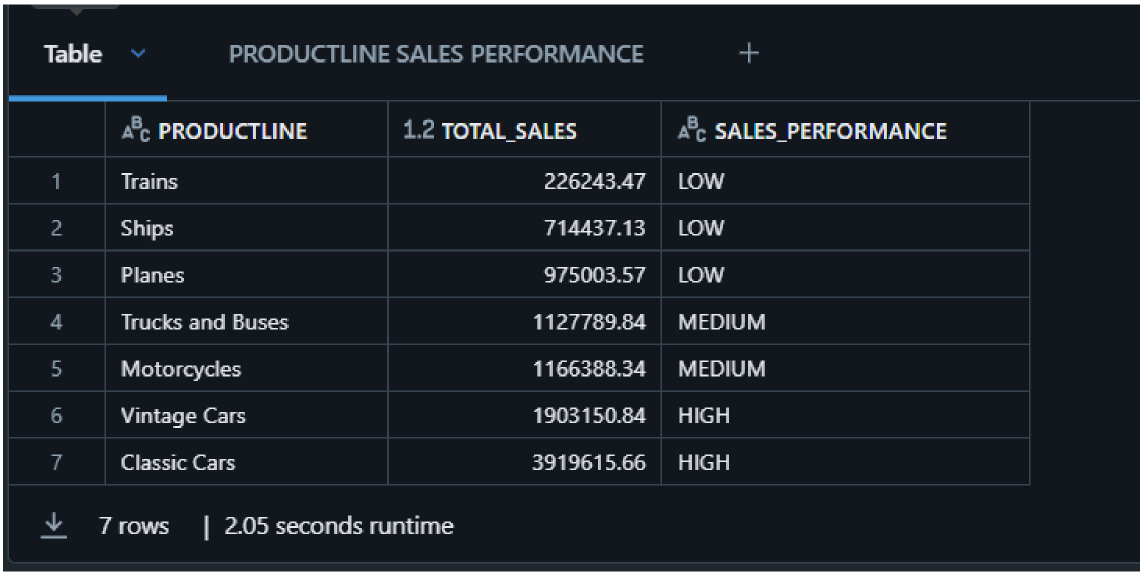
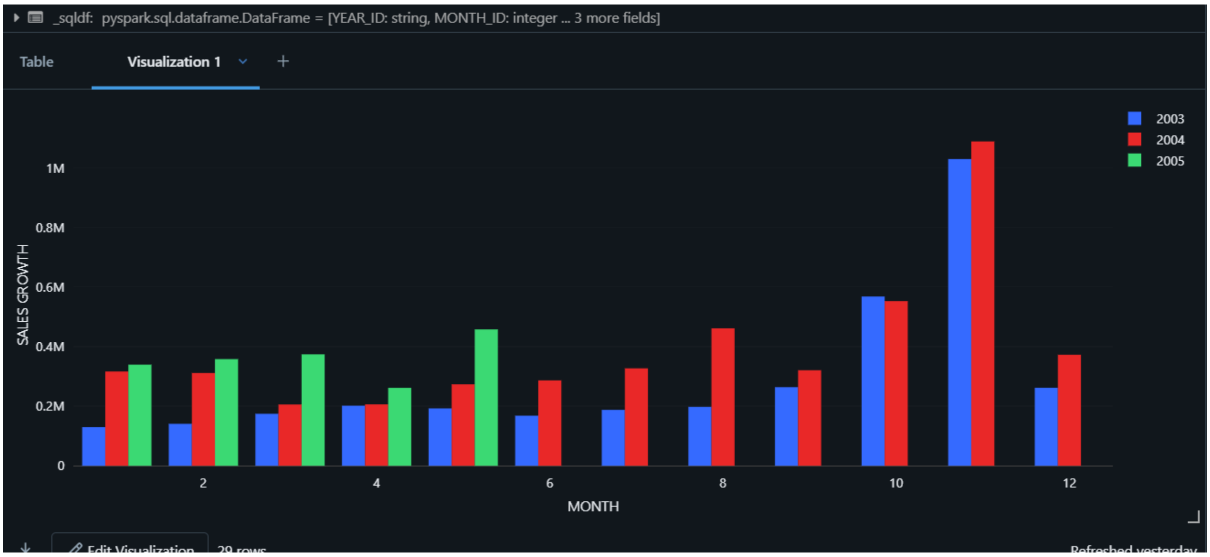


Table PRODUCTLINE SALES PERFORMANCE +

	PRODUCTLINE	TOTAL_SALES	SALES_PERFORMANCE
1	Trains	226243.47	LOW
2	Ships	714437.13	LOW
3	Planes	975003.57	LOW
4	Trucks and Buses	1127789.84	MEDIUM
5	Motorcycles	1166388.34	MEDIUM
6	Vintage Cars	1903150.84	HIGH
7	Classic Cars	3919615.66	HIGH

7 rows | 2.05 seconds runtime



_sqldf: pyspark.sql.dataframe.DataFrame = [YEAR_ID: string, MONTH_ID: integer ... 3 more fields]