NAAN MUDHALVAN

PHASE 3 PROJECT SUBMISSION

PRODUCT SALES ANALYSIS

TEAM MEMBERS:

PARVATHY A - 2021504533

PAVITHRAN E - 2021504535

SHAMEEM AHAMED S - 2021504542

THUSHIYANTH K - 2021504555

PROBLEM DEFINITION:

This project involves using IBM Cognos to analyze sales data and extract insights about top-selling products, peak sales periods, and customer preferences. The objective is to help businesses improve inventory management and marketing strategies by understanding sales trends and customer behavior. This project includes defining analysis objectives, collecting sales data, designing relevant visualizations in IBM Cognos, and deriving actionable insights.

DATABASE LINK:

https://www.kaggle.com/datasets/ksabishek/product-sales-data

OBJECTIVES:

- Analysing data related to product sales.
- Generating valuable insights from the data.
- Based on the insights, recommendations must be formulated to address issues and optimize sales and profitability.

Cleaning and Preprocessing:

Importing the important packages:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# To ignore warnings
import warnings
warnings.filterwarnings("ignore")
```

Loading the dataset:

```
print("Load the dataset")
import pandas as pd
data = pd.read_csv('/statsfinal.csv', low_memory=False) data.shape
data.head(30)
```

Load the dataset

	Unnamed: 0	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4
0	0	13-06-2010	5422	3725	576	907	17187.74	23616.50	3121.92	6466.91
1	1	14-06-2010	7047	779	3578	1574	22338.99	4938.86	19392.76	11222.62
2	2	15-06-2010	1572	2082	595	1145	4983.24	13199.88	3224.90	8163.85
3	3	16-06-2010	5657	2399	3140	1672	17932.69	15209.66	17018.80	11921.36
4	4	17-06-2010	3668	3207	2184	708	11627.56	20332.38	11837.28	5048.04
5	5	18-06-2010	2898	2539	311	1513	9186.66	16097.26	1685.62	10787.69
6	6	19-06-2010	6912	1470	1576	1608	21911.04	9319.80	8541.92	11465.04
7	7	20-06-2010	5209	2550	3415	842	16512.53	16167.00	18509.30	6003.46
8	8	21-06-2010	6322	852	3646	1377	20040.74	5401.68	19761.32	9818.01
9	9	22-06-2010	6865	414	3902	562	21762.05	2624.76	21148.84	4007.06
10	10	23-06-2010	1287	3955	2710	1804	4079.79	25074.70	14688.20	12862.52
11	11	24-06-2010	2197	1429	2754	1299	6964.49	9059.86	14926.68	9261.87
12	12	25-06-2010	7910	1622	5574	306	25074.70	10283.48	30211.08	2181.78
13	13	26-06-2010	3855	1015	1746	608	12220.35	6435.10	9463.32	4335.04
14	14	27-06-2010	5988	3288	916	1530	18981.96	20845.92	4964.72	10908.90
15	15	28-06-2010	2653	1544	3867	652	8410.01	9788.96	20959.14	4648.76
16	16	29-06-2010	3664	2294	3244	897	11614.88	14543.96	17582.48	6395.61
17	17	30-06-2010	7077	2297	5376	1130	22434.09	14562.98	29137.92	8056.90
18	18	01-07-2010	3509	700	1175	1205	11123.53	4438.00	6368.50	8591.65

No. of rows and columns:

```
print("Load the dataset")
import pandas as pd
data = pd.read_csv('/statsfinal.csv', low_memory=False)
data.shape
data.head(30)
input_file = "/statsfinal.csv"
df = pd.read_csv(input_file)
print(df)
```

OUTPUT:

```
      Unnamed: 0        Date  Q-P1  Q-P2  Q-P3  Q-P4      S-P1      S-P2  \
0              0  13-06-2010  5422  3725   576   907  17187.74  23616.50
1              1  14-06-2010  7047   779  3578  1574  22338.99   4938.86
2              2  15-06-2010  1572  2082   595  1145   4983.24  13199.88
3              3  16-06-2010  5657  2399  3140  1672  17932.69  15209.66
4              4  17-06-2010  3668  3207  2184   708  11627.56  20332.38
...          ...         ...   ...   ...   ...   ...       ...       ...
4595        4595  30-01-2023  2476  3419   525  1359   7848.92  21676.46
4596        4596  31-01-2023  7446   841  4825  1311  23603.82   5331.94
4597        4597  01-02-2023  6289  3143  3588   474  19936.13  19926.62
4598        4598  02-02-2023  3122  1188  5899   517   9896.74   7531.92
4599        4599  03-02-2023  1234  3854  2321   406   3911.78  24434.36

          S-P3      S-P4
0      3121.92   6466.91
1     19392.76  11222.62
2      3224.90   8163.85
3     17018.80  11921.36
4     11837.28   5048.04
...        ...       ...
4595   2845.50   9689.67
4596  26151.50   9347.43
4597  19446.96   3379.62
4598  31972.58   3686.21
4599  12579.82   2894.78

[4600 rows x 10 columns]
```

DROPPING FIRST COLUMN:

```
data = data.drop(columns=['Unnamed: 0'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    4600 non-null   object
 1   Q-P1    4600 non-null   int64
 2   Q-P2    4600 non-null   int64
 3   Q-P3    4600 non-null   int64
 4   Q-P4    4600 non-null   int64
 5   S-P1    4600 non-null   float64
 6   S-P2    4600 non-null   float64
 7   S-P3    4600 non-null   float64
 8   S-P4    4600 non-null   float64
dtypes: float64(4), int64(4), object(1)
```

Cleaned File:

```
output_file = "cleaned_file.csv"
df.to_csv(output_file, index=False)
print(output_file)
```

Output:

cleaned_file.csv

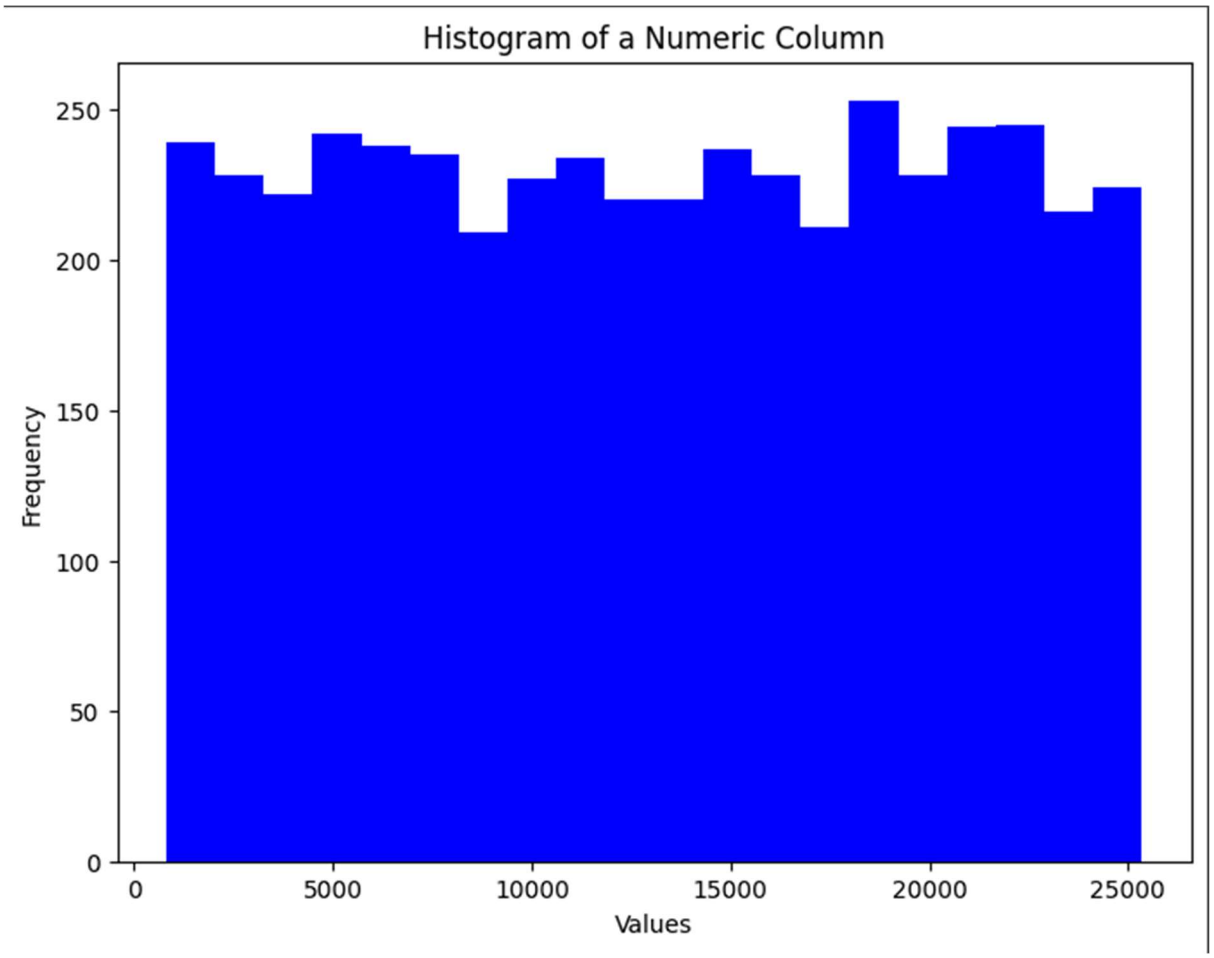
Loading the cleaned CSV file:

```
input_file = "cleaned_file.csv"
df = pd.read_csv(input_file)
```

Histogram visualization:

```
plt.figure(figsize=(8, 6))
plt.hist(df['S-P1'], bins=20, color='red')
plt.title('Histogram of a Numeric Column')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.show()
```

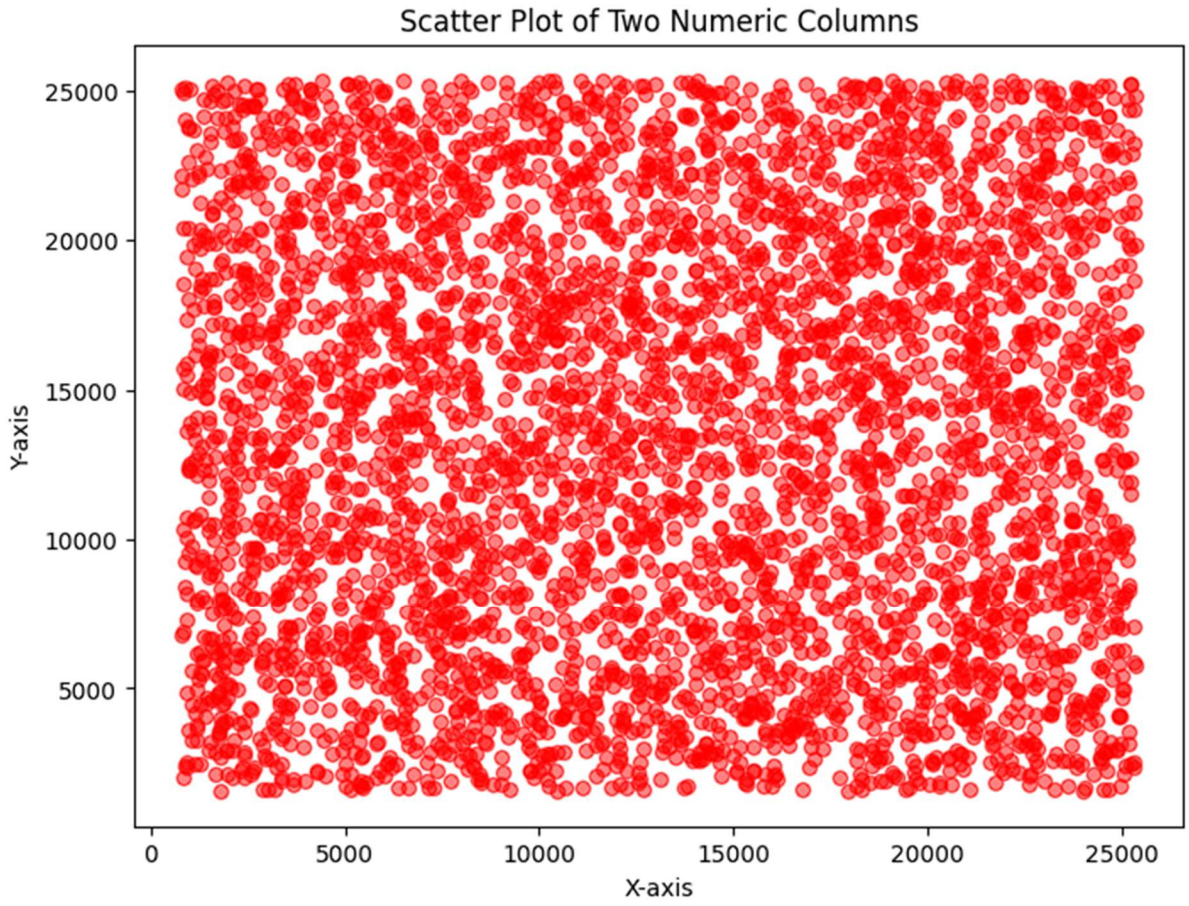
Output:



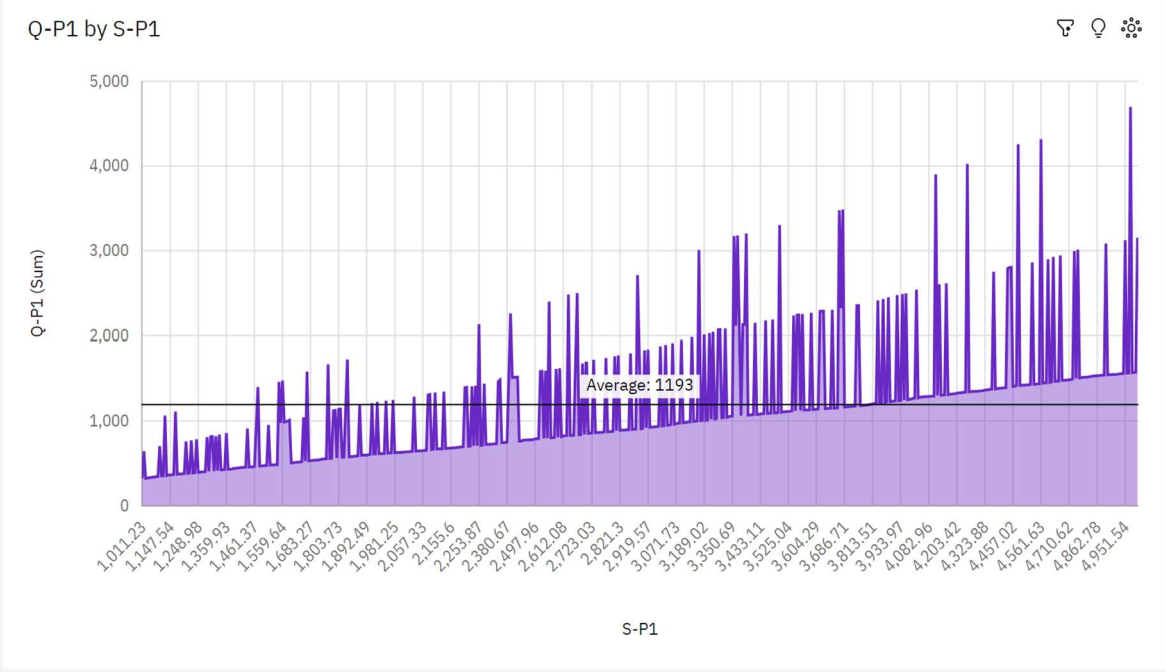
Scatter Plot Visualization:

```
plt.figure(figsize=(8, 6))
plt.scatter(df['S-P1'], df['S-P2'], color='blue', alpha=0.5)
plt.title('Scatter Plot of Two Numeric Columns')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```

Output:



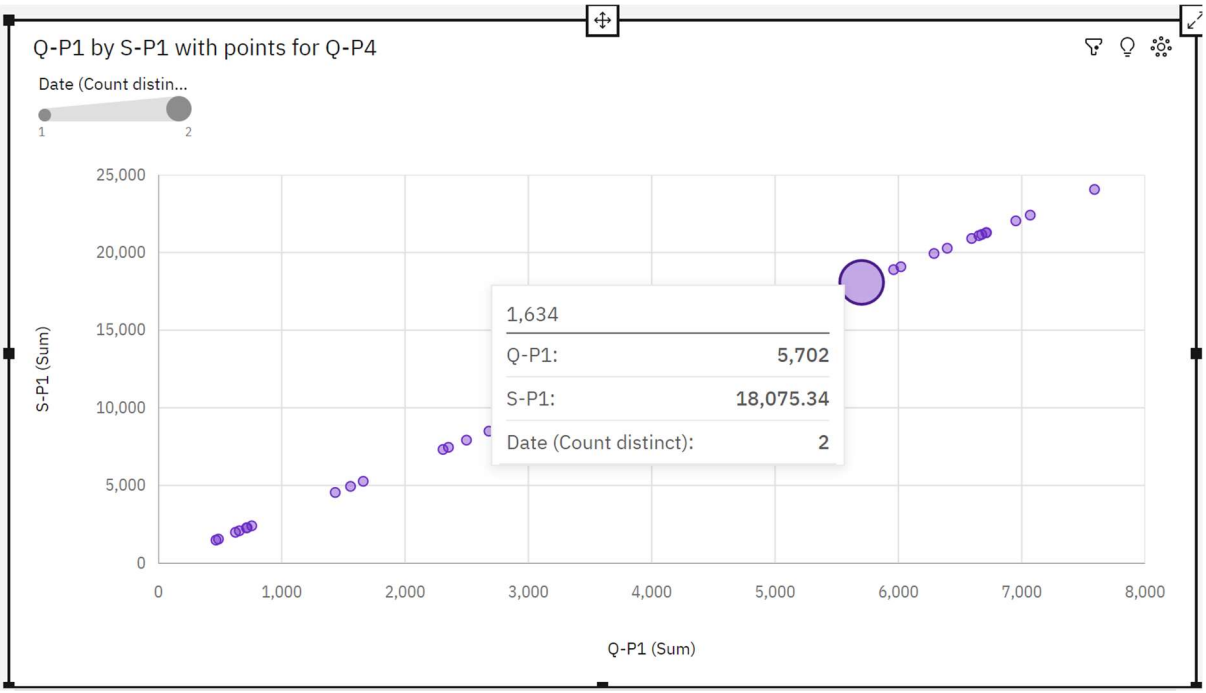
VISUALIZATION 1:



Insights:

- Over all values of S-P1, the sum of Q-P1 is nearly 678 thousand.
- **Q-P1** ranges from **319**, when **S-P1** is **1011.23**, to **over 4500**, when **S-P1** is **4964.22**.
- **S-P1 3683.54** has the highest **Total Q-P4** but is ranked #**6** in **Total Q-P1**.
- **S-P1 4964.22** has the highest **Total Q-P1** but is ranked #**238** in **Total Q-P4**.
- **Q-P4** and **Q-P1** diverged the most when **S-P1** is **1122.18**, and when **Q-P4** was **over three thousand** higher than the **Q-P1**.
- **3683.54 Q-P4** at **over 5 thousand** is **31**% higher than the **Q-P1** of **almost 3500**.

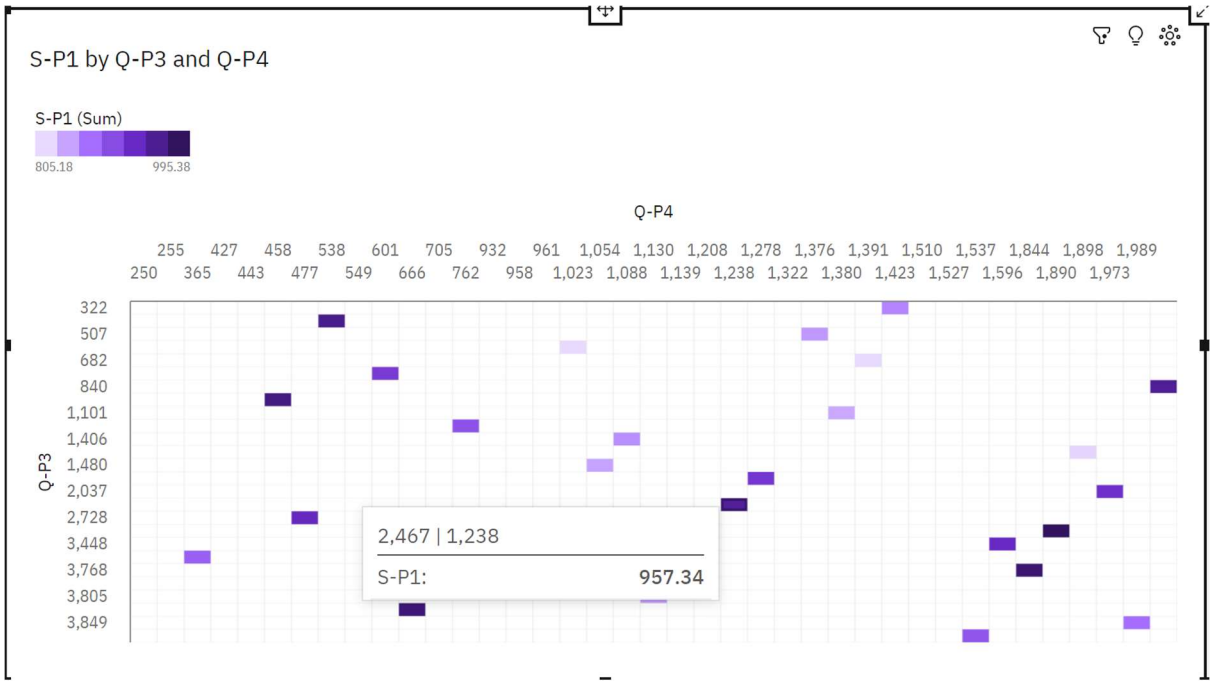
VISUALIZATION 2:



Insights:

- **S-P4** at **over 23 thousand** is **76%** higher than the **Q-P1** of **over 5500**.
- **S-P4** and **Q-P1** diverged the most when **Q-P4** is **1634**, and when **S-P4** was **nearly 18 thousand** higher than the **Q-P1**.
- **Q-P4 960** has the highest **Total Q-P1** but is ranked #**30** in **Total S-P4.**
- **Q-P4 1634 has the highest Total S-P4 but is ranked #13 in Total Q-P1.**
- **The total of S-P1 is nearly 575 thousand.**

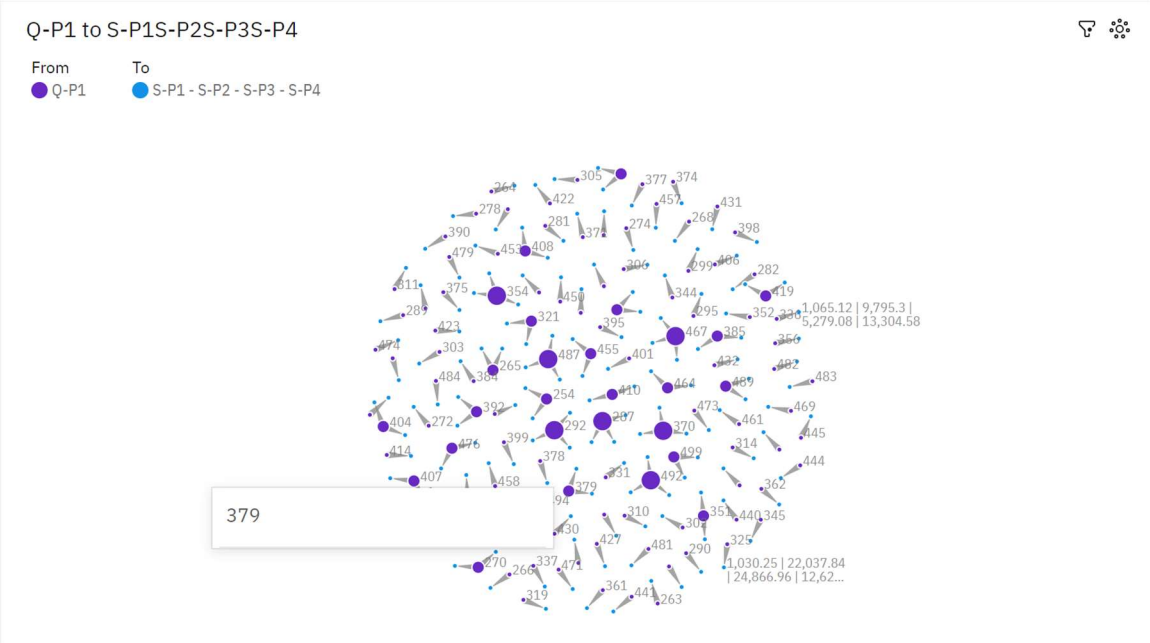
VISUALIZATION 3:



Insights:

- The summed values of **S-P1** range from **805.2** to **995.4**.
- For **S-P1**, the most significant value of **Q-P4** is **1890**, whose respective **S-P1** values add up to **995.4**, or **2.8** % of the total.
- For **S-P1**, the most significant value of **Q-P3** is **2773**, whose respective **S-P1** values add up to **995.4**, or **2.8** % of the total.
- Across all values of **Q-P3** and **Q-P4**, the sum of **S-P1** is **over 35 thousand**
- **2773 S-P4** at **13,476** is **93**% higher than the **S-P1** of **995.4**.
- **250 S-P1** at **985.9** is **68**% higher than the **Q-P1** of **311**.
- **Q-P4 250** has the highest values of both **S-P1** and **Q-P1**.
- **Q-P3 2773** has the highest values of both **S-P1** and **S-P4**.
- **Q-P3 2773** has the highest total **S-P1** due to **Q-P4 1890**.

VISUALIZATION 4:



INSIGHTS:

- **S-P1 1445.52** has the highest **Unaggregated Q-P1** but is ranked #**99** in **Total Q-P4**
- **S-P1 1122.18** has the highest **Total Q-P4** but is ranked #**108** in **Unaggregated Q-P1**.

SUMMARY: