Data Analytics with Cognos Project

PROJECT TITLE:

PRODUCT SALES ANALYSIS

PROJECT DOCUMENTATION
AND SUBMISSION PHASE

TEAM MEMBERS:

MUTHU RATHNASHRI C	2021115068
NITHYA SREE K	2021115071
MUTHAMIZH VAANJINATHAN M	2021115067
NAREN KARTHICK THIRUVENGADAM	2021115069
PRASAAD S	2021115307

OBJECTIVE

The primary goal of the project is to optimize product sales through innovative analysis methods, leveraging data-driven insights.

DESIGN THINKING PROCESS

Empathize:

Understanding the needs of customers, market trends, and sales patterns.

Define:

Defining clear objectives for the analysis, focusing on improving sales performance and customer satisfaction.

Ideate:

Generating creative ideas for data collection, preprocessing, analysis techniques, and visualization methods.

Prototype:

Developing a preliminary model for analysis, incorporating various data sources and tools.

Test:

Validating the prototype, refining the analysis techniques, and ensuring the actionable insights are meaningful and practical.

DEVELOPMENT PHASES

Data Collection and Aggregation:

Gathering sales transactions, customer demographics, and market trend data from diverse sources.

Data Cleaning and Preprocessing:

Ensuring accuracy and consistency of the collected data, preparing it for analysis.

Advanced Analytics:

Utilizing techniques like data mining and machine learning to identify patterns, correlations, and outliers.

Data Visualization:

Using IBM Cognos and other visualization tools to present the findings in a visually comprehensible format.

Predictive and Prescriptive Analytics:

Implementing predictive models for sales forecasting and prescriptive analytics to provide actionable recommendations.

MONITORING AND FEEDBACK

Establishing continuous monitoring processes and feedback loops to adapt strategies based on market dynamics.

Analysis Objectives, Data Collection Process, Data Visualization, and Actionable Insights

Analysis Objectives:

Enhancing sales performance.

Understanding customer behavior and preferences.

Identifying market trends and patterns.

Improving inventory management and marketing strategies.

Data Collection Process:

Gathering sales transactions data from POS systems.

Collecting customer demographics through surveys and online profiles.

Tracking market trends using industry reports and online databases.

Data Visualization using IBM Cognos:

Utilizing IBM Cognos to create interactive dashboards and visual representations of sales data.

Visualizing customer segmentation, product performance, and market trends for comprehensive analysis.

DERIVED ACTIONABLE INSIGHTS

Customer Segmentation:

Identifying high-value customer segments for targeted marketing efforts.

Product Performance Analysis:

Evaluating the popularity of products to optimize inventory levels.

Market Trends:

Understanding emerging trends to introduce new products or modify existing ones.

Seasonal Analysis:

Anticipating demand fluctuations based on seasonal patterns for inventory planning.

GUIDING INVENTORY MANAGEMENT AND MARKETING STRATEGIES USING INSIGHTS

Inventory Management

Optimizing Stock Levels:

Using demand forecasts to maintain optimal inventory levels, preventing overstocking or stockouts.

Introducing Seasonal Products:

Introducing seasonal products based on historical trends to capitalize on peak demand periods.

Supplier Collaboration:

Collaborating with suppliers based on demand predictions, ensuring timely restocking.

MARKETING STRATEGIES

Targeted Marketing Campaigns:

Tailoring marketing campaigns to specific customer segments identified through analysis, increasing engagement and conversions.

Personalized Offers:

Creating personalized offers for customers based on their preferences, increasing customer loyalty and sales.

Market Expansion:

Identifying untapped markets or customer segments for expanding marketing efforts, driving growth opportunities.

OVERVIEW

Innovation in product sales analysis involves a series of essential steps to optimize performance and gain valuable insights. First, data collection and aggregation is paramount, as it forms the foundation of analysis. This includes gathering data on sales transactions, customer demographics, and market trends. The next step is data cleaning and preprocessing, ensuring accuracy and consistency. Once the data is ready, advanced analytics techniques, such as data mining and machine learning, can be employed to identify patterns, correlations, and outliers. Visualization tools are then used to present these findings in a digestible format. To foster innovation, predictive and prescriptive analytics can be applied, allowing for forecasting and actionable recommendations. Regular monitoring and feedback loops complete the process, enabling continuous improvement and adaptation to changing market dynamics.

DATA CLEANING

1. Understanding the Data:

Begin by understanding the dataset's structure, column names, data types, and identifying any missing or duplicate values. This step helps in formulating a strategy for cleaning the data effectively.

2. Handling Missing Values:

Decide how to handle missing values in the dataset. Options include removing rows with missing values or filling missing values with appropriate measures like the mean or median.

3. Handling Duplicates:

Identify and remove duplicate rows to ensure that the analysis is performed on unique data points, preventing skewed results.

4. Saving Cleaned Data:

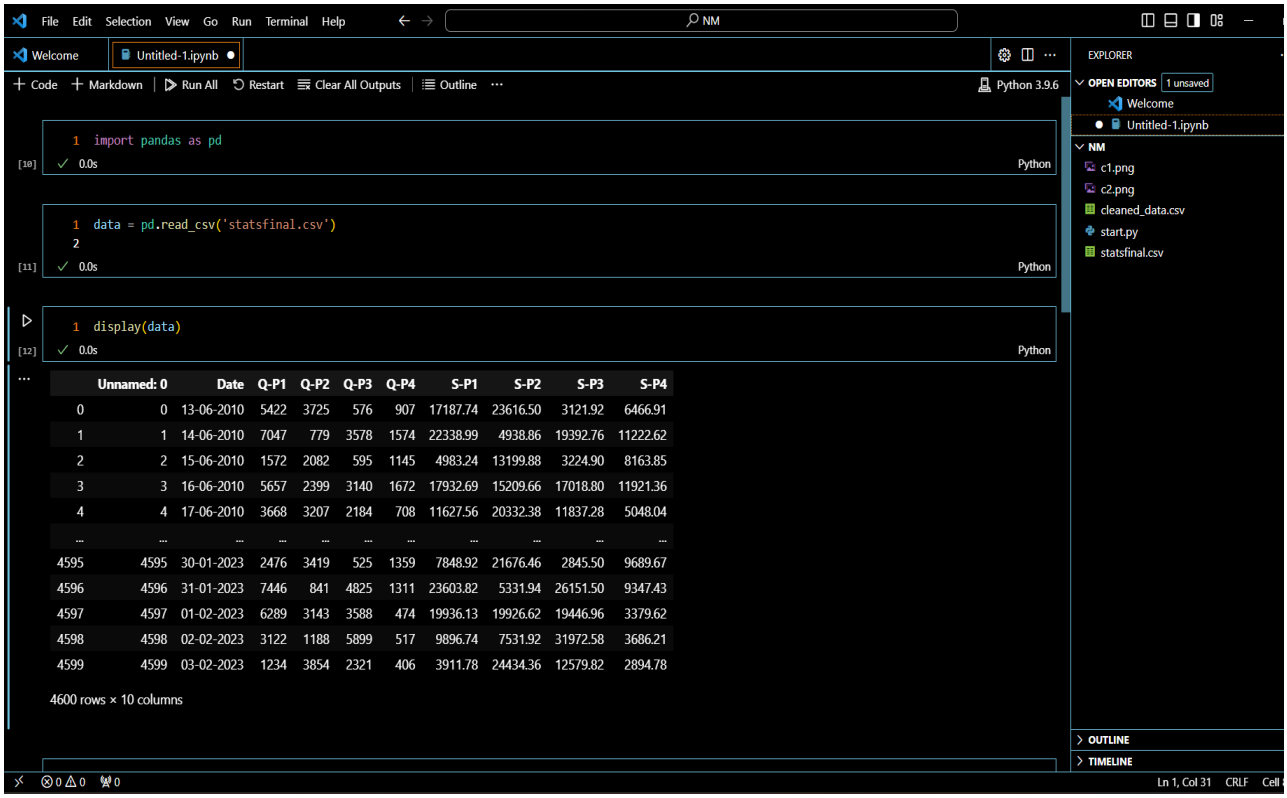
After completing the cleaning and preprocessing steps, save the cleaned dataset. This ensures that the cleaned data is available for further analysis and modeling without the need to repeat the cleaning process every time the analysis is performed.

About Dataset

- Q1- Total unit sales of product 1
- Q2- Total unit sales of product 2
- Q3- Total unit sales of product 3
- Q4- Total unit sales of product 4
- S1- Total revenue from product 1
- S2- Total revenue from product 2

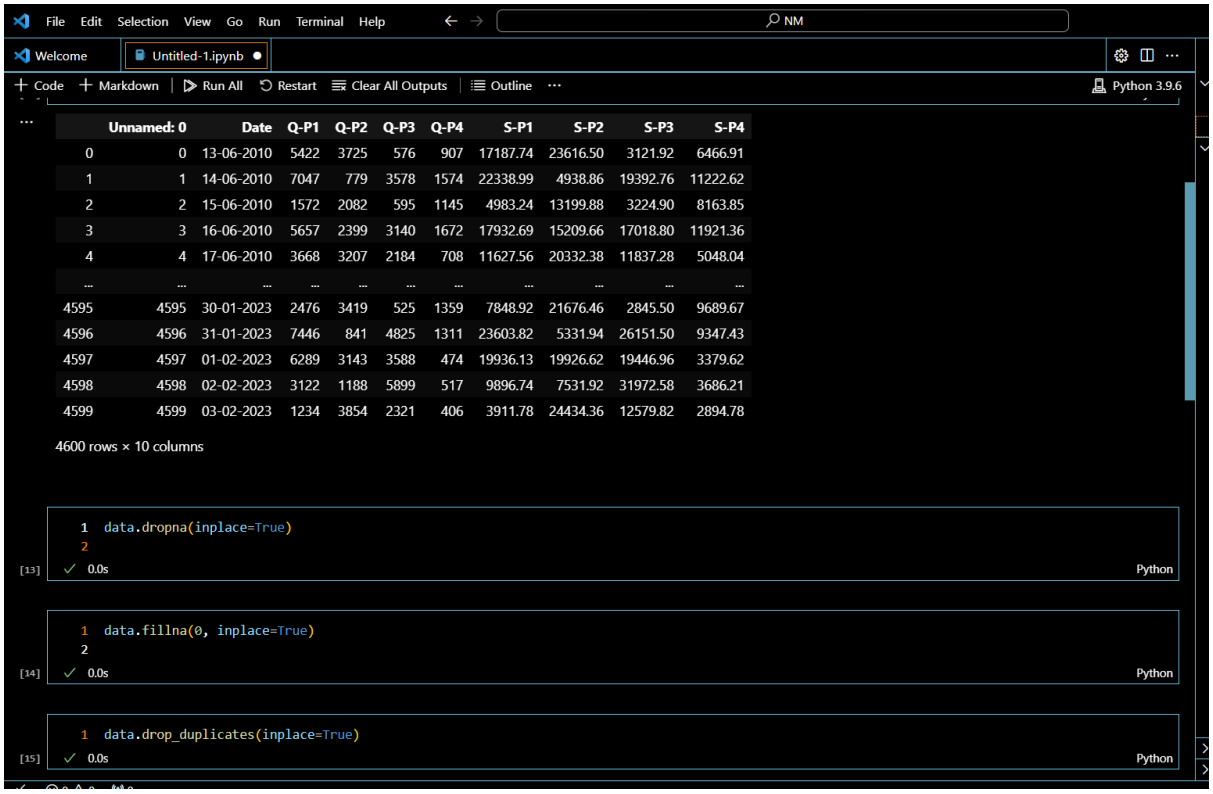
- S3- Total revenue from product 3
- S4- Total revenue from product 4

Data Cleaning



	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
...	...	...	...	...	...	...	...	...	...	...
4595	4595	30-01-2023	2476	3419	525	1359	7848.92	21676.46	2845.50	9689.67
4596	4596	31-01-2023	7446	841	4825	1311	23603.82	5331.94	26151.50	9347.43
4597	4597	01-02-2023	6289	3143	3588	474	19936.13	19926.62	19446.96	3379.62
4598	4598	02-02-2023	3122	1188	5899	517	9896.74	7531.92	31972.58	3686.21
4599	4599	03-02-2023	1234	3854	2321	406	3911.78	24434.36	12579.82	2894.78

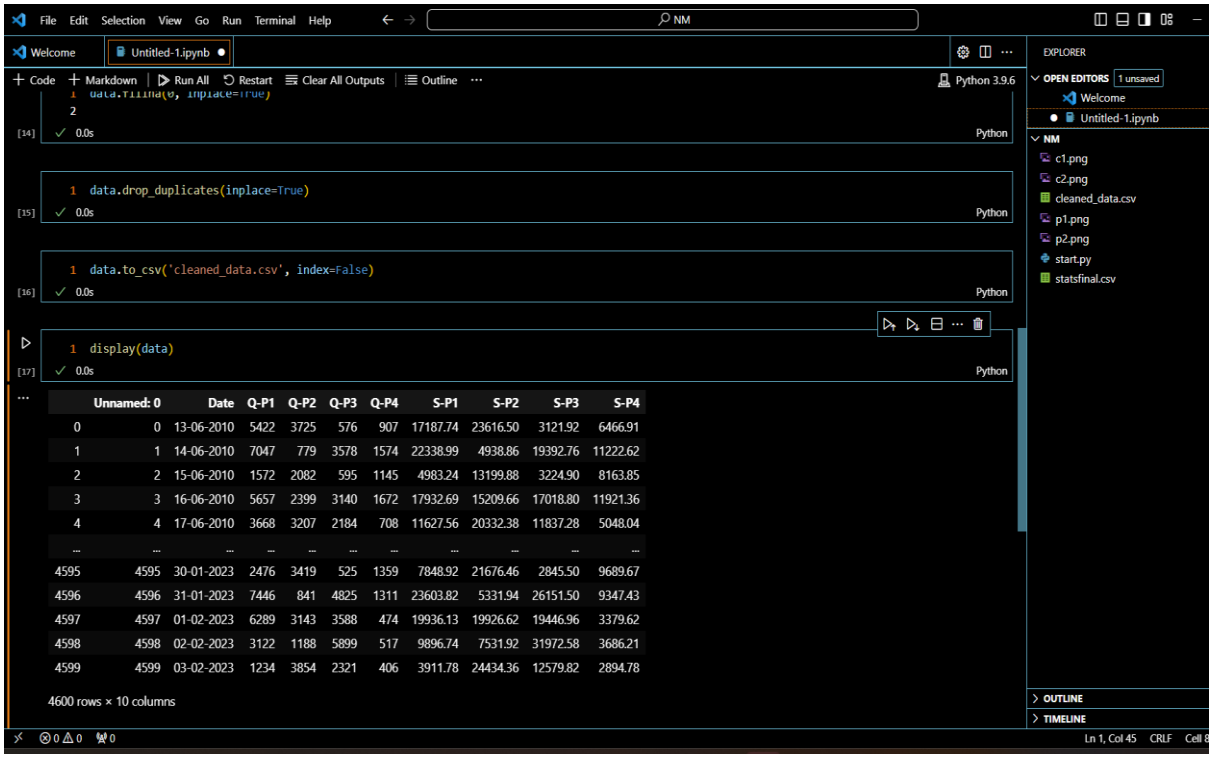
4600 rows × 10 columns



	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
...	...	...	...	...	...	...	...	...	...	...
4595	4595	30-01-2023	2476	3419	525	1359	7848.92	21676.46	2845.50	9689.67
4596	4596	31-01-2023	7446	841	4825	1311	23603.82	5331.94	26151.50	9347.43
4597	4597	01-02-2023	6289	3143	3588	474	19936.13	19926.62	19446.96	3379.62
4598	4598	02-02-2023	3122	1188	5899	517	9896.74	7531.92	31972.58	3686.21
4599	4599	03-02-2023	1234	3854	2321	406	3911.78	24434.36	12579.82	2894.78

4600 rows × 10 columns

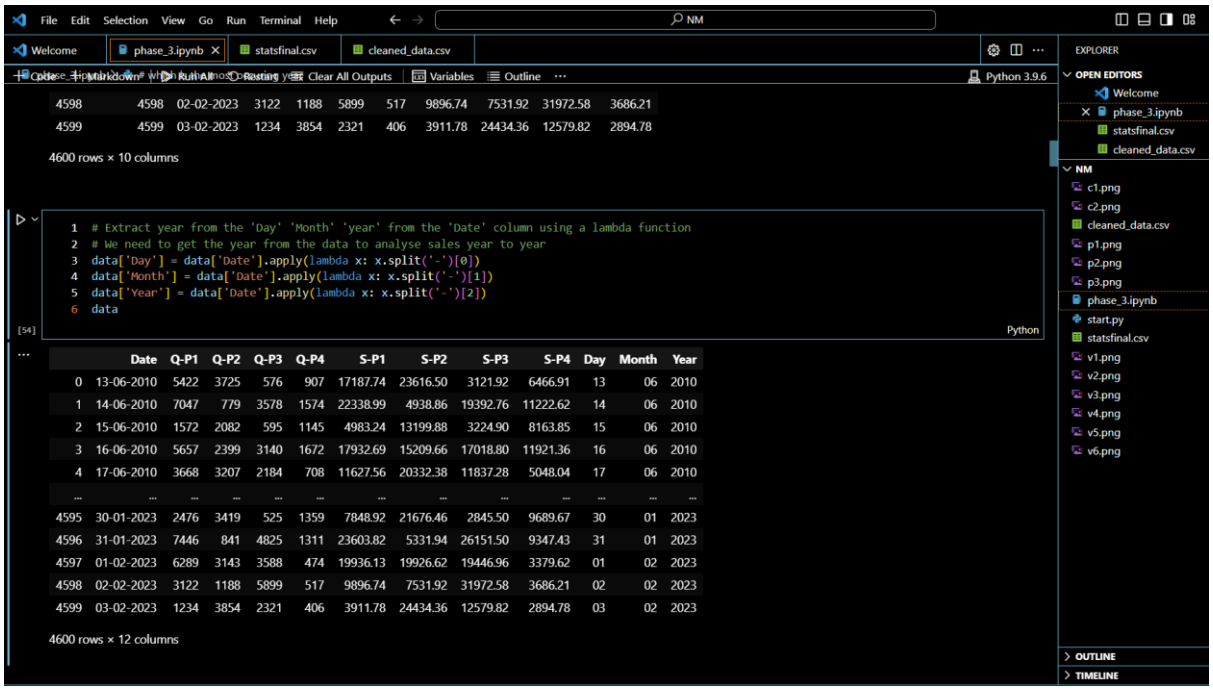
```
[13] 1 data.dropna(inplace=True)
     2
     ✓ 0.0s Python

[14] 1 data.fillna(0, inplace=True)
     2
     ✓ 0.0s Python

[15] 1 data.drop_duplicates(inplace=True)
     ✓ 0.0s Python
```

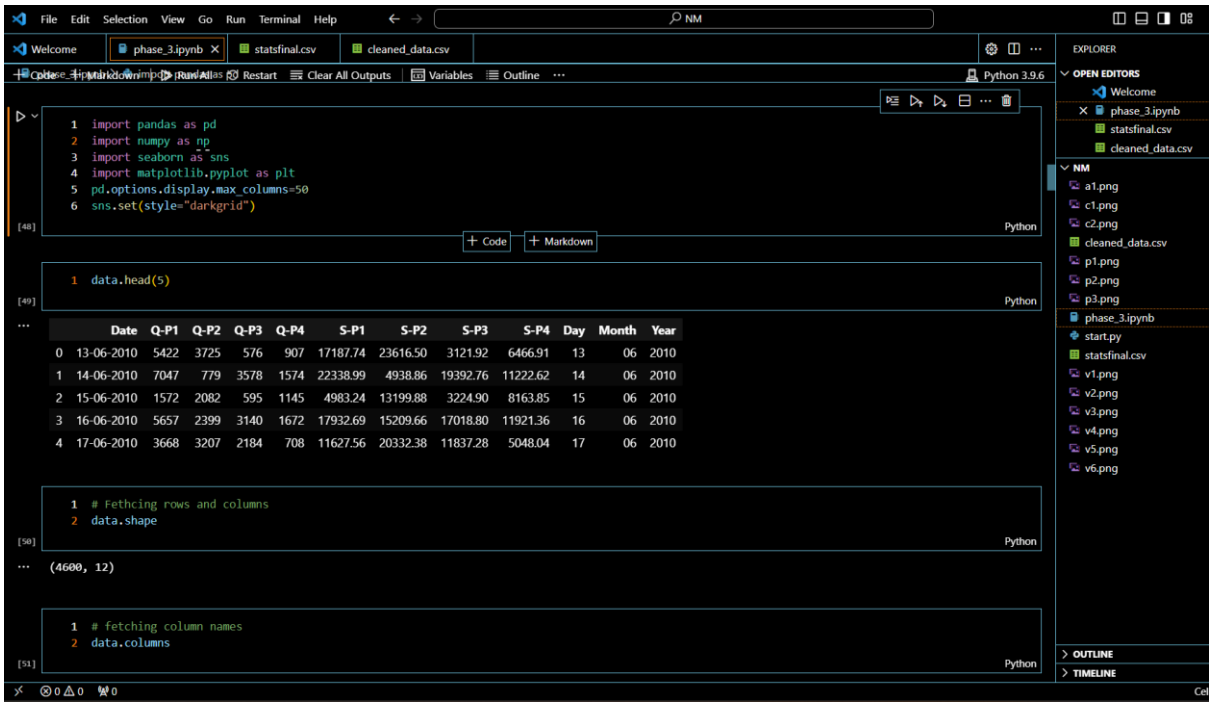


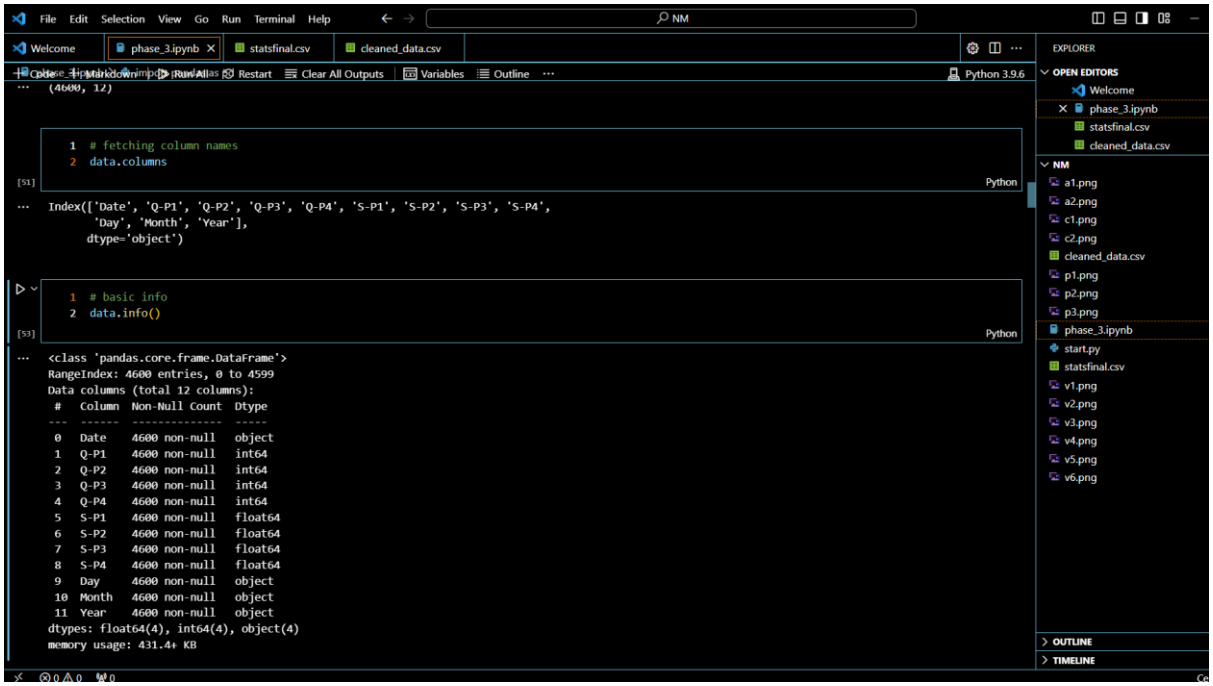

```
[14] 1 data.fillna(0, inplace=True)
     2
     ✓ 0.0s Python

[15] 1 data.drop_duplicates(inplace=True)
     ✓ 0.0s Python

[16] 1 data.to_csv('cleaned_data.csv', index=False)
     ✓ 0.0s Python

[17] 1 display(data)
     ✓ 0.0s Python
```

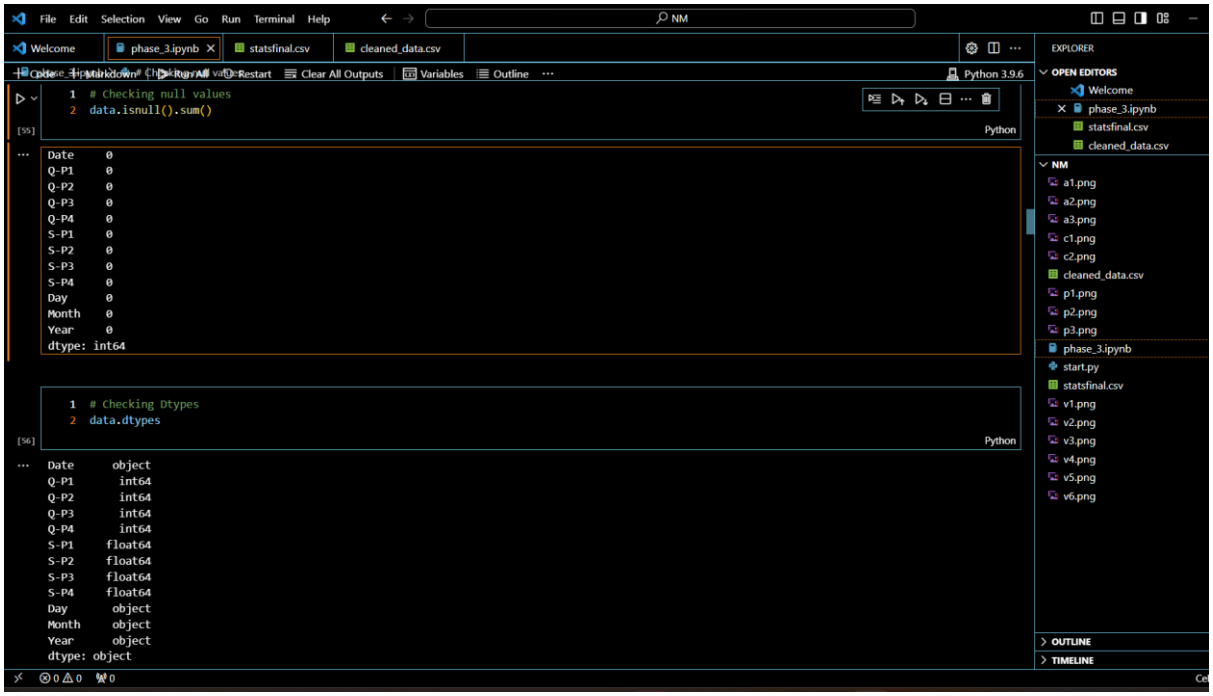
	Unnamed: 0	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4
0	0	13-06-2010	5422	3725	576	907	17187.74	23616.50	3121.92	6466.91
1	1	14-06-2010	7047	779	3578	1574	22338.99	4938.86	19392.76	11222.62
2	2	15-06-2010	1572	2082	595	1145	4983.24	13199.88	3224.90	8163.85
3	3	16-06-2010	5657	2399	3140	1672	17932.69	15209.66	17018.80	11921.36
4	4	17-06-2010	3668	3207	2184	708	11627.56	20332.38	11837.28	5048.04
...	...	...	...	...	...	...	...	...	...	...
4595	4595	30-01-2023	2476	3419	525	1359	7848.92	21676.46	2845.50	9689.67
4596	4596	31-01-2023	7446	841	4825	1311	23603.82	5331.94	26151.50	9347.43
4597	4597	01-02-2023	6289	3143	3588	474	19936.13	19926.62	19446.96	3379.62
4598	4598	02-02-2023	3122	1188	5899	517	9896.74	7531.92	31972.58	3686.21
4599	4599	03-02-2023	1234	3854	2321	406	3911.78	24434.36	12579.82	2894.78

4600 rows × 10 columns


File Edit Selection View Go Run Terminal Help
NM
Welcome
phase_3.ipynb
statsfinal.csv
cleaned_data.csv
Code Markdown Run All Restart Clear All Outputs Variables Outline
Python 3.9.6
EXPLORER
OPEN EDITORS
Welcome
phase_3.ipynb
statsfinal.csv
cleaned_data.csv
NM
c1.png
c2.png
cleaned_data.csv
p1.png
p2.png
p3.png
phase_3.ipynb
start.py
statsfinal.csv
v1.png
v2.png
v3.png
v4.png
v5.png
v6.png
OUTLINE
TIMELINE
4598 4598 02-02-2023 3122 1188 5899 517 9896.74 7531.92 31972.58 3686.21
4599 4599 03-02-2023 1234 3854 2321 406 3911.78 24434.36 12579.82 2894.78
4600 rows × 10 columns
1 # Extract year from the 'Day' 'Month' 'year' from the 'Date' column using a lambda function
2 # We need to get the year from the data to analyse sales year to year
3 data['Day'] = data['Date'].apply(lambda x: x.split('-')[0])
4 data['Month'] = data['Date'].apply(lambda x: x.split('-')[1])
5 data['Year'] = data['Date'].apply(lambda x: x.split('-')[2])
6 data
[54]
Python
Date Q-P1 Q-P2 Q-P3 Q-P4 S-P1 S-P2 S-P3 S-P4 Day Month Year
0 13-06-2010 5422 3725 576 907 17187.74 23616.50 3121.92 6466.91 13 06 2010
1 14-06-2010 7047 779 3578 1574 22338.99 4938.86 19392.76 11222.62 14 06 2010
2 15-06-2010 1572 2082 595 1145 4983.24 13199.88 3224.90 8163.85 15 06 2010
3 16-06-2010 5657 2399 3140 1672 17932.69 15209.66 17018.80 11921.36 16 06 2010
4 17-06-2010 3668 3207 2184 708 11627.56 20332.38 11837.28 5048.04 17 06 2010
4595 30-01-2023 2476 3419 525 1359 7848.92 21676.46 2845.50 9689.67 30 01 2023
4596 31-01-2023 7446 841 4825 1311 23603.82 5331.94 26151.50 9347.43 31 01 2023
4597 01-02-2023 6289 3143 3588 474 19936.13 19926.62 19446.96 3379.62 01 02 2023
4598 02-02-2023 3122 1188 5899 517 9896.74 7531.92 31972.58 3686.21 02 02 2023
4599 03-02-2023 1234 3854 2321 406 3911.78 24434.36 12579.82 2894.78 03 02 2023
4600 rows × 12 columns

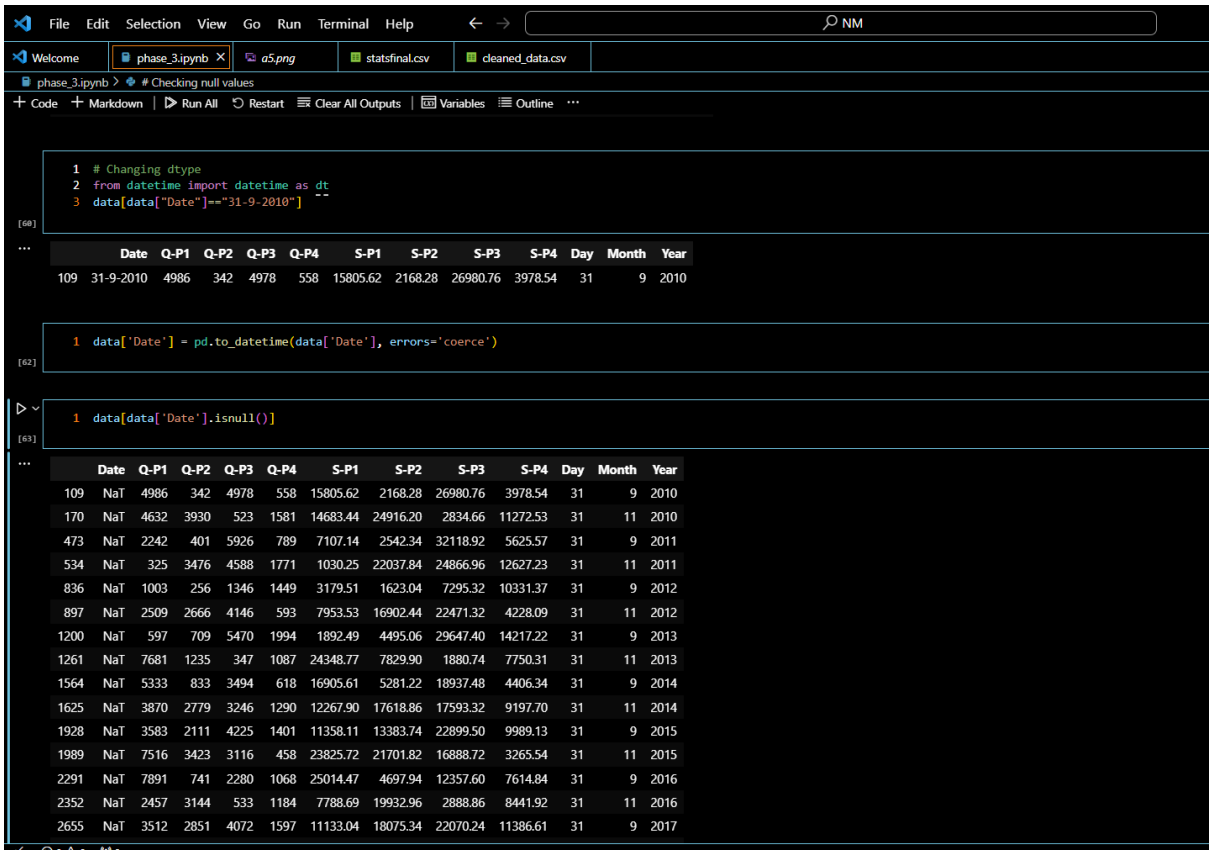

File Edit Selection View Go Run Terminal Help
NM
Welcome
phase_3.ipynb
statsfinal.csv
cleaned_data.csv
Code Markdown Run All Restart Clear All Outputs Variables Outline
Python 3.9.6
EXPLORER
OPEN EDITORS
Welcome
phase_3.ipynb
statsfinal.csv
cleaned_data.csv
NM
a1.png
c1.png
c2.png
cleaned_data.csv
p1.png
p2.png
p3.png
phase_3.ipynb
start.py
statsfinal.csv
v1.png
v2.png
v3.png
v4.png
v5.png
v6.png
OUTLINE
TIMELINE
1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 pd.options.display.max_columns=50
6 sns.set(style="darkgrid")
[48]
Python
Code Markdown
1 data.head(5)
[49]
Python
Date Q-P1 Q-P2 Q-P3 Q-P4 S-P1 S-P2 S-P3 S-P4 Day Month Year
0 13-06-2010 5422 3725 576 907 17187.74 23616.50 3121.92 6466.91 13 06 2010
1 14-06-2010 7047 779 3578 1574 22338.99 4938.86 19392.76 11222.62 14 06 2010
2 15-06-2010 1572 2082 595 1145 4983.24 13199.88 3224.90 8163.85 15 06 2010
3 16-06-2010 5657 2399 3140 1672 17932.69 15209.66 17018.80 11921.36 16 06 2010
4 17-06-2010 3668 3207 2184 708 11627.56 20332.38 11837.28 5048.04 17 06 2010
1 # Fethcing rows and columns
2 data.shape
[50]
Python
(4600, 12)
1 # fetching column names
2 data.columns
[51]
Python


File Edit Selection View Go Run Terminal Help
NM
Welcome
phase_3.ipynb
statsfinal.csv
cleaned_data.csv
Code Markdown Run All Restart Clear All Outputs Variables Outline
Python 3.9.6
(4600, 12)
1 # fetching column names
2 data.columns
[51]
Python
Index(['Date', 'Q-P1', 'Q-P2', 'Q-P3', 'Q-P4', 'S-P1', 'S-P2', 'S-P3', 'S-P4',
'Day', 'Month', 'Year'],
dtype='object')
1 # basic info
2 data.info()
[53]
Python
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 12 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
9 Day 4600 non-null object
10 Month 4600 non-null object
11 Year 4600 non-null object
dtypes: float64(4), int64(4), object(4)
memory usage: 431.4+ KB
EXPLORER
OPEN EDITORS
Welcome
phase_3.ipynb
statsfinal.csv
cleaned_data.csv
NM
a1.png
a2.png
c1.png
c2.png
cleaned_data.csv
p1.png
p2.png
p3.png
phase_3.ipynb
start.py
statsfinal.csv
v1.png
v2.png
v3.png
v4.png
v5.png
v6.png
OUTLINE
TIMELINE


File Edit Selection View Go Run Terminal Help
NM
Welcome
phase_3.ipynb
statsfinal.csv
cleaned_data.csv
Code Markdown Run All Restart Clear All Outputs Variables Outline
Python 3.9.6
1 # Checking null values
2 data.isnull().sum()
[55]
Python
Date 0
Q-P1 0
Q-P2 0
Q-P3 0
Q-P4 0
S-P1 0
S-P2 0
S-P3 0
S-P4 0
Day 0
Month 0
Year 0
dtype: int64
1 # Checking Dtypes
2 data.dtypes
[56]
Python
Date object
Q-P1 int64
Q-P2 int64
Q-P3 int64
Q-P4 int64
S-P1 float64
S-P2 float64
S-P3 float64
S-P4 float64
Day object
Month object
Year object
dtype: object
EXPLORER
OPEN EDITORS
Welcome
phase_3.ipynb
statsfinal.csv
cleaned_data.csv
NM
a1.png
a2.png
a3.png
c1.png
c2.png
cleaned_data.csv
p1.png
p2.png
p3.png
phase_3.ipynb
start.py
statsfinal.csv
v1.png
v2.png
v3.png
v4.png
v5.png
v6.png
OUTLINE
TIMELINE

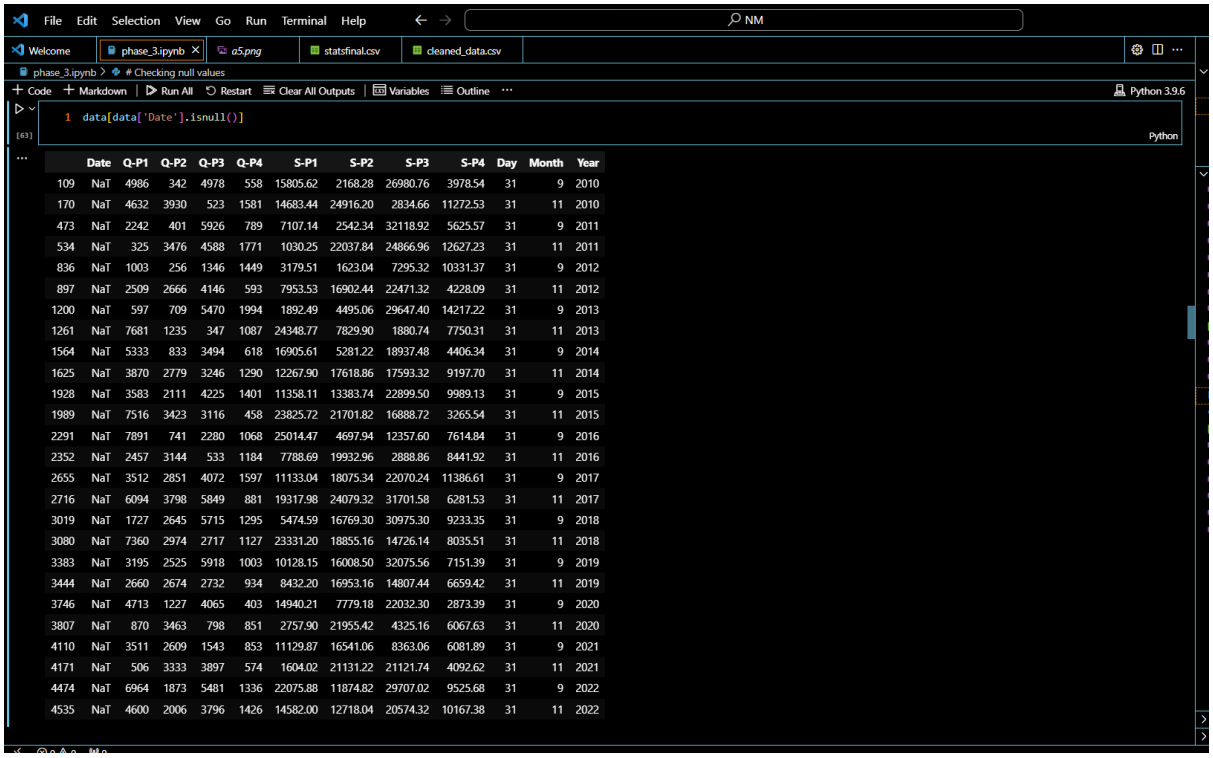
	count	mean	std	min	25%	50%	75%	max
Q-P1	4600.0	4121.849130	2244.271323	254.00	2150.500	4137.000	6072.000	7998.00
Q-P2	4600.0	2130.281522	1089.783705	251.00	1167.750	2134.000	3070.250	3998.00
Q-P3	4600.0	3145.740000	1671.832231	250.00	1695.750	3202.500	4569.000	6000.00
Q-P4	4600.0	1123.500000	497.385676	250.00	696.000	1136.500	1544.000	2000.00
S-P1	4600.0	13066.261743	7114.340094	805.18	6817.085	13114.290	19248.240	25353.66
S-P2	4600.0	13505.984848	6909.228687	1591.34	7403.535	13529.560	19465.385	25347.32
S-P3	4600.0	17049.910800	9061.330694	1355.00	9190.965	17357.550	24763.980	32520.00
S-P4	4600.0	8010.555000	3546.359869	1782.50	4962.480	8103.245	11008.720	14260.00

	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4	Day	Month	Year
1628	03-12-2014	4261	844	600	434	13507.37	5350.96	3252.00	3094.42	03	12	2014
49	01-08-2010	2062	3659	2194	803	6536.54	23198.06	11891.48	5725.39	01	08	2010



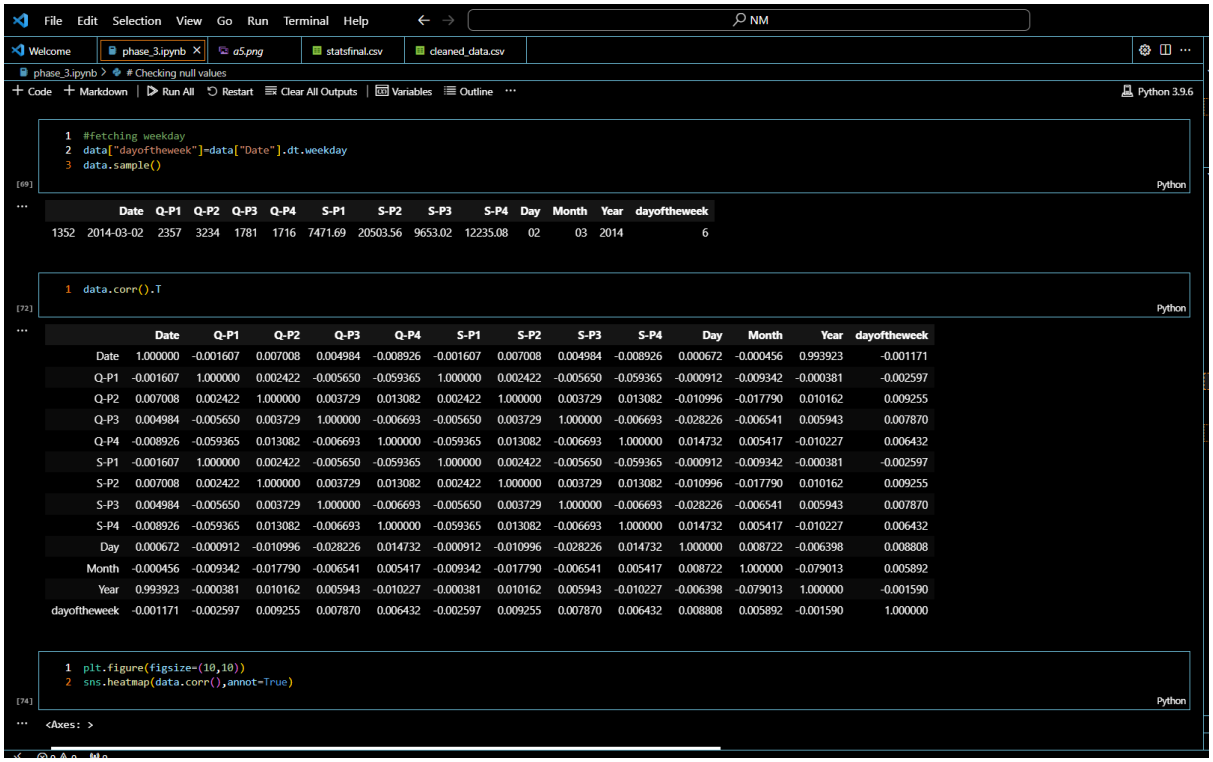
	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4	Day	Month	Year
109	31-9-2010	4986	342	4978	558	15805.62	2168.28	26980.76	3978.54	31	9	2010

	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4	Day	Month	Year
109	NaT	4986	342	4978	558	15805.62	2168.28	26980.76	3978.54	31	9	2010
170	NaT	4632	3930	523	1581	14683.44	24916.20	2834.66	11272.53	31	11	2010
473	NaT	2242	401	5926	789	7107.14	2542.34	32118.92	5625.57	31	9	2011
534	NaT	325	3476	4588	1771	1030.25	22037.84	24866.96	12627.23	31	11	2011
836	NaT	1003	256	1346	1449	3179.51	1623.04	7295.32	10331.37	31	9	2012
897	NaT	2509	2666	4146	593	7953.53	16902.44	22471.32	4228.09	31	11	2012
1200	NaT	597	709	5470	1994	1892.49	4495.06	29647.40	14217.22	31	9	2013
1261	NaT	7681	1235	347	1087	24348.77	7829.90	1880.74	7750.31	31	11	2013
1564	NaT	5333	833	3494	618	16905.61	5281.22	18937.48	4406.34	31	9	2014
1625	NaT	3870	2779	3246	1290	12267.90	17618.86	17593.32	9197.70	31	11	2014
1928	NaT	3583	2111	4225	1401	11358.11	13383.74	22899.50	9989.13	31	9	2015
1989	NaT	7516	3423	3116	458	23825.72	21701.82	16888.72	3265.54	31	11	2015
2291	NaT	7891	741	2280	1068	25014.47	4697.94	12357.60	7614.84	31	9	2016
2352	NaT	2457	3144	533	1184	7788.69	19932.96	2888.86	8441.92	31	11	2016
2655	NaT	3512	2851	4072	1597	11133.04	18075.34	22070.24	11386.61	31	9	2017



	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4	Day	Month	Year
109	NaT	4986	342	4978	558	15805.62	2168.28	26980.76	3978.54	31	9	2010
170	NaT	4632	3930	523	1581	14683.44	24916.20	2834.66	11272.53	31	11	2010
473	NaT	2242	401	5926	789	7107.14	2542.34	32118.92	5625.57	31	9	2011
534	NaT	325	3476	4588	1771	1030.25	22037.84	24866.96	12627.23	31	11	2011
836	NaT	1003	256	1346	1449	3179.51	1623.04	7295.32	10331.37	31	9	2012
897	NaT	2509	2666	4146	593	7953.53	16902.44	22471.32	4228.09	31	11	2012
1200	NaT	597	709	5470	1994	1892.49	4495.06	29647.40	14217.22	31	9	2013
1261	NaT	7681	1235	347	1087	24348.77	7829.90	1880.74	7750.31	31	11	2013
1564	NaT	5333	833	3494	618	16905.61	5281.22	18937.48	4406.34	31	9	2014
1625	NaT	3870	2779	3246	1290	12267.90	17618.86	17593.32	9197.70	31	11	2014
1928	NaT	3583	2111	4225	1401	11358.11	13383.74	22899.50	9989.13	31	9	2015
1989	NaT	7516	3423	3116	458	23825.72	21701.82	16888.72	3265.54	31	11	2015
2291	NaT	7891	741	2280	1068	25014.47	4697.94	12357.60	7614.84	31	9	2016
2352	NaT	2457	3144	533	1184	7788.69	19932.96	2888.86	8441.92	31	11	2016
2655	NaT	3512	2851	4072	1597	11133.04	18075.34	22070.24	11386.61	31	9	2017
2716	NaT	6094	3798	5849	881	19317.98	24079.32	31701.58	6281.53	31	11	2017
3019	NaT	1727	2645	5715	1295	5474.59	16769.30	30975.30	9233.35	31	9	2018
3080	NaT	7360	2974	2717	1127	23331.20	18855.16	14726.14	8035.51	31	11	2018
3383	NaT	3195	2525	5918	1003	10128.15	16008.50	32075.56	7151.39	31	9	2019
3444	NaT	2660	2674	2732	934	8432.20	16953.16	14807.44	6659.42	31	11	2019
3746	NaT	4713	1227	4065	403	14940.21	7779.18	22032.30	2873.39	31	9	2020
3807	NaT	870	3463	798	851	2757.90	21955.42	4325.16	6067.63	31	11	2020
4110	NaT	3511	2609	1543	853	11129.87	16541.06	8363.06	6081.89	31	9	2021
4171	NaT	506	3333	3897	574	1604.02	21131.22	21121.74	4092.62	31	11	2021
4474	NaT	6964	1873	5481	1336	22075.88	11874.82	29707.02	9525.68	31	9	2022
4535	NaT	4600	2006	3796	1426	14582.00	12718.04	20574.32	10167.38	31	11	2022

	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4	Day	Month	Year	dayoftheweek
1352	2014-03-02	2357	3234	1781	1716	7471.69	20503.56	9653.02	12235.08	02	03	2014	6



```
#fetching weekday
data["dayoftheweek"]=data["Date"].dt.weekday
data.sample()
```

	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4	Day	Month	Year	dayoftheweek
1352	2014-03-02	2357	3234	1781	1716	7471.69	20503.56	9653.02	12235.08	02	03	2014	6

```
data.corr().T
```

	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4	Day	Month	Year	dayoftheweek
Date	1.000000	-0.001607	0.007008	0.004984	-0.008926	-0.001607	0.007008	0.004984	-0.008926	0.000672	-0.000456	0.993923	-0.001171
Q-P1	-0.001607	1.000000	0.002422	-0.005650	-0.059365	1.000000	0.002422	-0.005650	-0.059365	-0.000912	-0.009342	-0.000381	-0.002597
Q-P2	0.007008	0.002422	1.000000	0.003729	0.013082	0.002422	1.000000	0.003729	0.013082	-0.010996	-0.017790	0.010162	0.009255
Q-P3	0.004984	-0.005650	0.003729	1.000000	-0.006693	-0.005650	0.003729	1.000000	-0.006693	-0.028226	-0.006541	0.005943	0.007870
Q-P4	-0.008926	-0.059365	0.013082	-0.006693	1.000000	-0.059365	0.013082	-0.006693	1.000000	0.014732	0.005417	-0.010227	0.006432
S-P1	-0.001607	1.000000	0.002422	-0.005650	-0.059365	1.000000	0.002422	-0.005650	-0.059365	-0.000912	-0.009342	-0.000381	-0.002597
S-P2	0.007008	0.002422	1.000000	0.003729	0.013082	0.002422	1.000000	0.003729	0.013082	-0.010996	-0.017790	0.010162	0.009255
S-P3	0.004984	-0.005650	0.003729	1.000000	-0.006693	-0.005650	0.003729	1.000000	-0.006693	-0.028226	-0.006541	0.005943	0.007870
S-P4	-0.008926	-0.059365	0.013082	-0.006693	1.000000	-0.059365	0.013082	-0.006693	1.000000	0.014732	0.005417	-0.010227	0.006432
Day	0.000672	-0.000912	-0.010996	-0.028226	0.014732	-0.000912	-0.010996	-0.028226	0.014732	1.000000	0.008722	-0.006398	0.008808
Month	-0.000456	-0.009342	-0.017790	-0.006541	0.005417	-0.009342	-0.017790	-0.006541	0.005417	0.008722	1.000000	-0.079013	0.005892
Year	0.993923	-0.000381	0.010162	0.005943	-0.010227	-0.000381	0.010162	0.005943	-0.010227	-0.006398	-0.079013	1.000000	-0.001590
dayoftheweek	-0.001171	-0.002597	0.009255	0.007870	0.006432	-0.002597	0.009255	0.007870	0.006432	0.008808	0.005892	-0.001590	1.000000

```
plt.figure(figsize=(10,10))
sns.heatmap(data.corr(),annot=True)
```

<Axes: >

Visualization

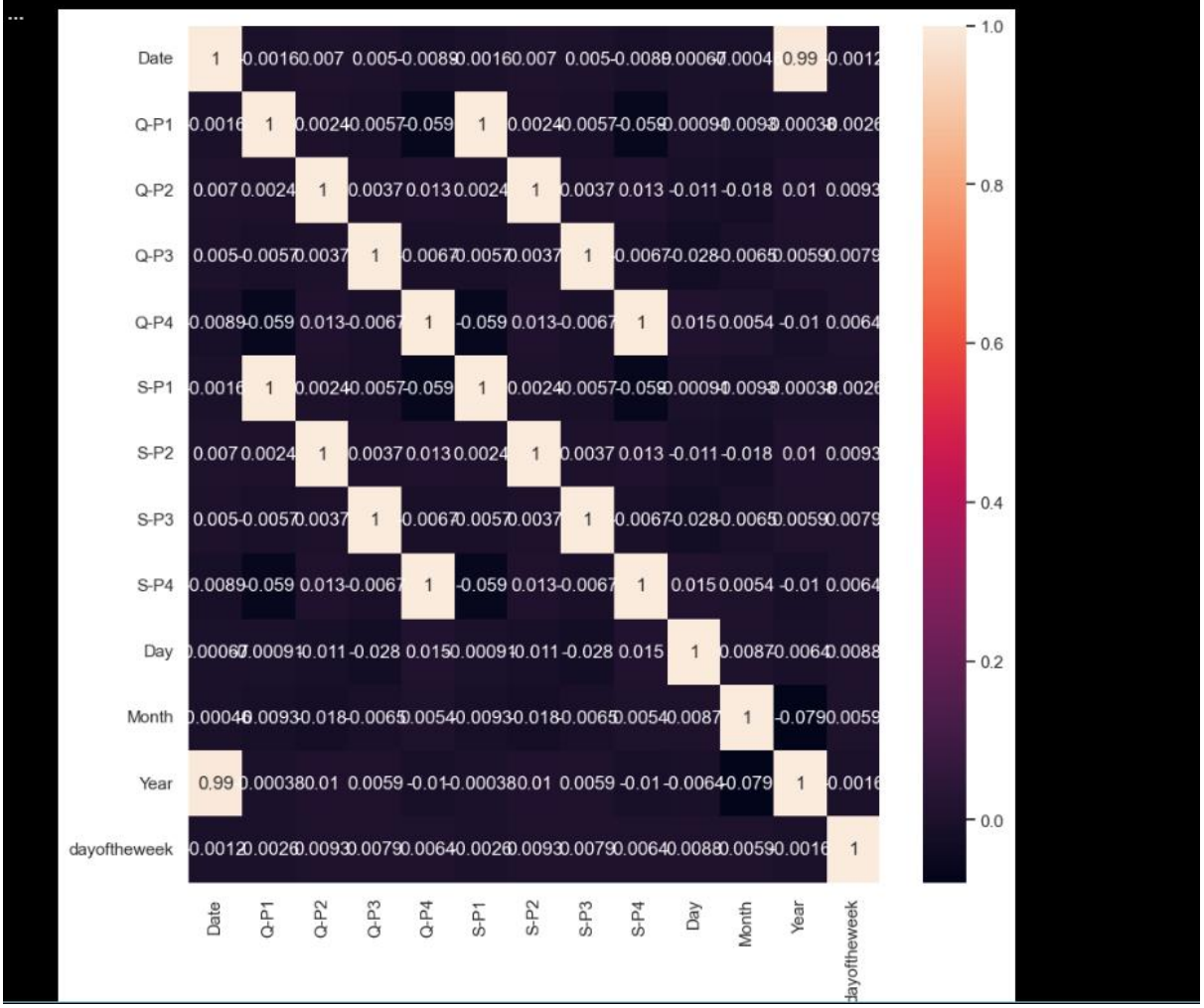
In the product sales analysis project, visualization plays a pivotal role in transforming complex data into clear, actionable insights. Utilizing IBM Cognos, a powerful business intelligence tool, enables the creation of intuitive and interactive visualizations. Through compelling charts, graphs, and dashboards, patterns and trends in product sales data become immediately apparent. Visualizations allow stakeholders to grasp sales performance across products, regions, and time periods effortlessly. This visual representation not only enhances data-driven decision-making but also facilitates the communication of key findings, empowering businesses to strategize effectively, optimize inventory, identify market opportunities, and enhance overall sales performance.

phase_3.ipynb > plt.figure(figsize=(10,10))

+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ...

```
[74]
1  plt.figure(figsize=(10,10))
2  sns.heatmap(data.corr(),annot=True)
```

```
<Axes: >
```



```
for i in data.columns:
    print(i,"---------",data[i].unique())
```

```
Date --------- <DatetimeArray>
['2010-06-13 00:00:00', '2010-06-14 00:00:00', '2010-06-15 00:00:00',
 '2010-06-16 00:00:00', '2010-06-17 00:00:00', '2010-06-18 00:00:00',
 '2010-06-19 00:00:00', '2010-06-20 00:00:00', '2010-06-21 00:00:00',
 '2010-06-22 00:00:00',
 ...
 '2023-01-25 00:00:00', '2023-01-26 00:00:00', '2023-01-27 00:00:00',
 '2023-01-28 00:00:00', '2023-01-29 00:00:00', '2023-01-30 00:00:00',
 '2023-01-31 00:00:00', '2023-02-01 00:00:00', '2023-02-02 00:00:00',
 '2023-02-03 00:00:00']
Length: 4575, dtype: datetime64[ns]
Q-P1 --------- [5422 7047 1572 ... 1227 3122 1234]
Q-P2 --------- [3725  770 2082 ... 3404  841 3143]
Q-P3 --------- [ 576 3578  595 ... 4825 3588 5899]
Q-P4 --------- [ 907 1574 1145 ... 1161 1151 1112]
S-P1 --------- [17187.74 22338.99  4983.24 ...  3889.59  9896.74  3911.78]
S-P2 --------- [23616.5   4938.86 13199.88 ... 21581.36  5331.94 19926.62]
S-P3 --------- [ 3121.92 19392.76  3224.9  ... 26151.5  19446.96 31972.58]
S-P4 --------- [ 6466.91 11222.62  8163.85 ...  8277.93  8286.63  7928.56]
Day --------- ['13' '14' '15' '16' '17' '18' '19' '20' '21' '22' '23' '24' '25' '26'
 '27' '28' '29' '30' '01' '02' '03' '04' '05' '06' '07' '08' '09' '10'
 '11' '12' '31']
Month --------- ['06' '07' '08' '09' '9' '10' '11' '12' '01' '02' '03' '04' '05']
Year --------- ['2010' '2011' '2012' '2013' '2014' '2015' '2016' '2017' '2018' '2019'
 '2020' '2021' '2022' '2023']
dayoftheweek --------- [6 0 1 2 3 4 5]
```

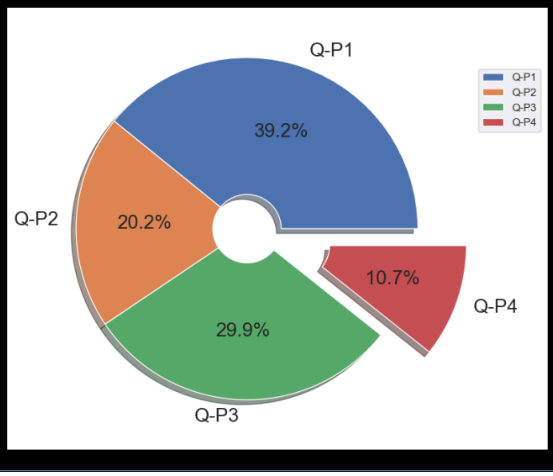
```
# Total unit sales Product 1, Product 2, Product 3, Product 4
q = data[["Q-P1","Q-P2","Q-P3","Q-P4"]].sum()
print(q)
plt.figure(figsize=(8,8))
plt.pie(q,labels=data[["Q-P1","Q-P2","Q-P3","Q-P4"]].sum().index,shadow=True,autopct="%0.01f%%",textprops={"fontsize":20},wedgeprops={'width': 0.8},explode=[0,0,0,0.3])
plt.legend(loc='center right', bbox_to_anchor=(1.2, 0.8));
```

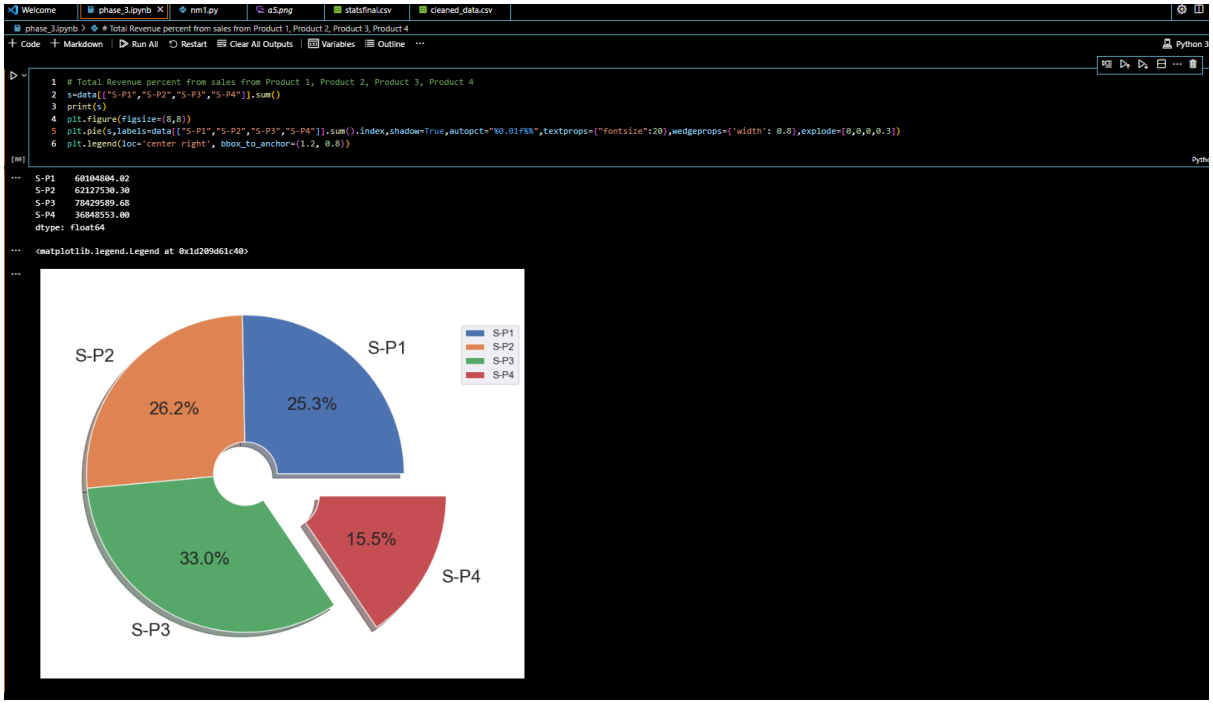
```
Q-P1    18960506
Q-P2     9799295
Q-P3    14470404
Q-P4     5168100
dtype: int64
```


```
'2020' '2021' '2022' '2023']
dayoftheweek --------- [6 0 1 2 3 4 5]
```

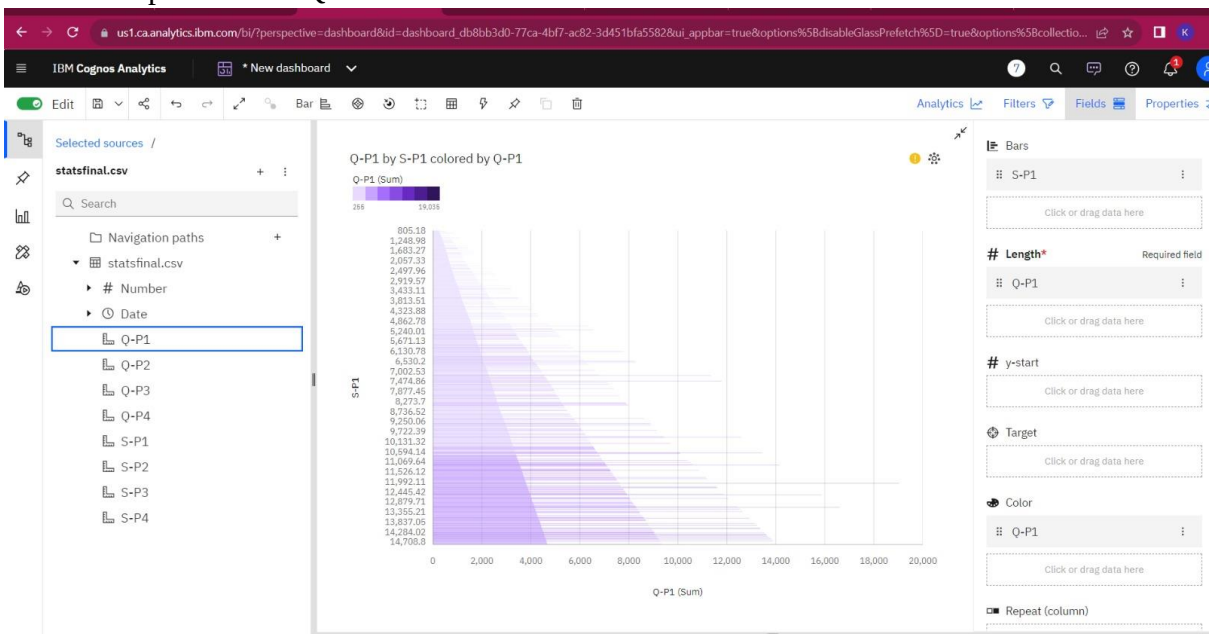
```
# Total unit sales Product 1, Product 2, Product 3, Product 4
q = data[["Q-P1","Q-P2","Q-P3","Q-P4"]].sum()
print(q)
plt.figure(figsize=(8,8))
plt.pie(q,labels=data[["Q-P1","Q-P2","Q-P3","Q-P4"]].sum().index,shadow=True,autopct="%0.01f%%",textprops={"fontsize":20},wedgeprops={'width': 0.8},explode=[0,0,0,0.3])
plt.legend(loc='center right', bbox_to_anchor=(1.2, 0.8));
```

```
Q-P1    18960506
Q-P2     9799295
Q-P3    14470404
Q-P4     5168100
dtype: int64
```

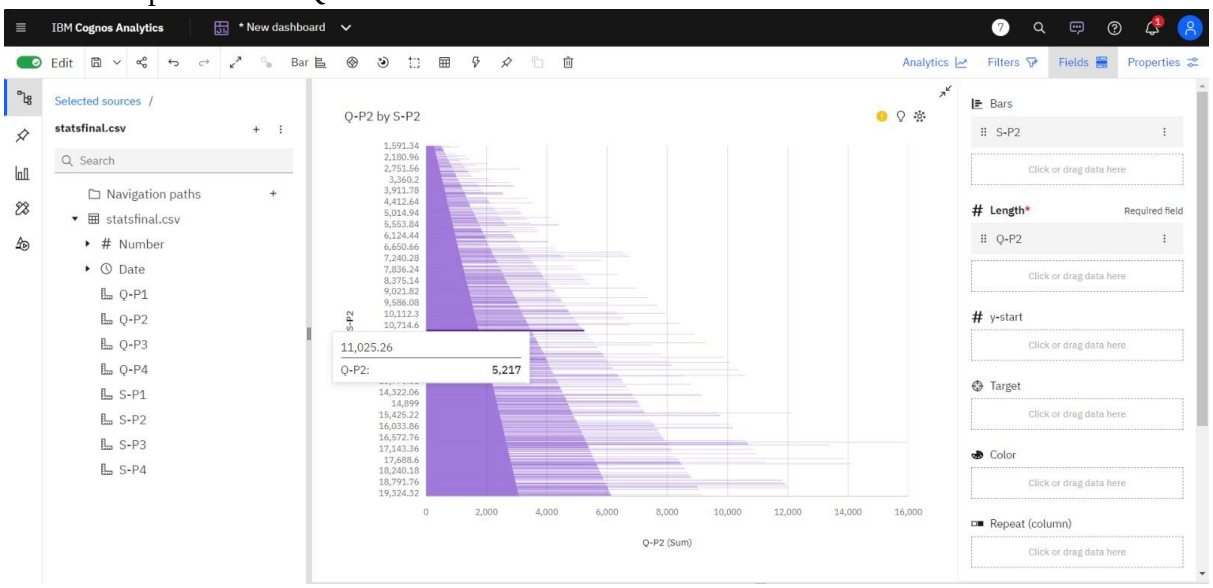





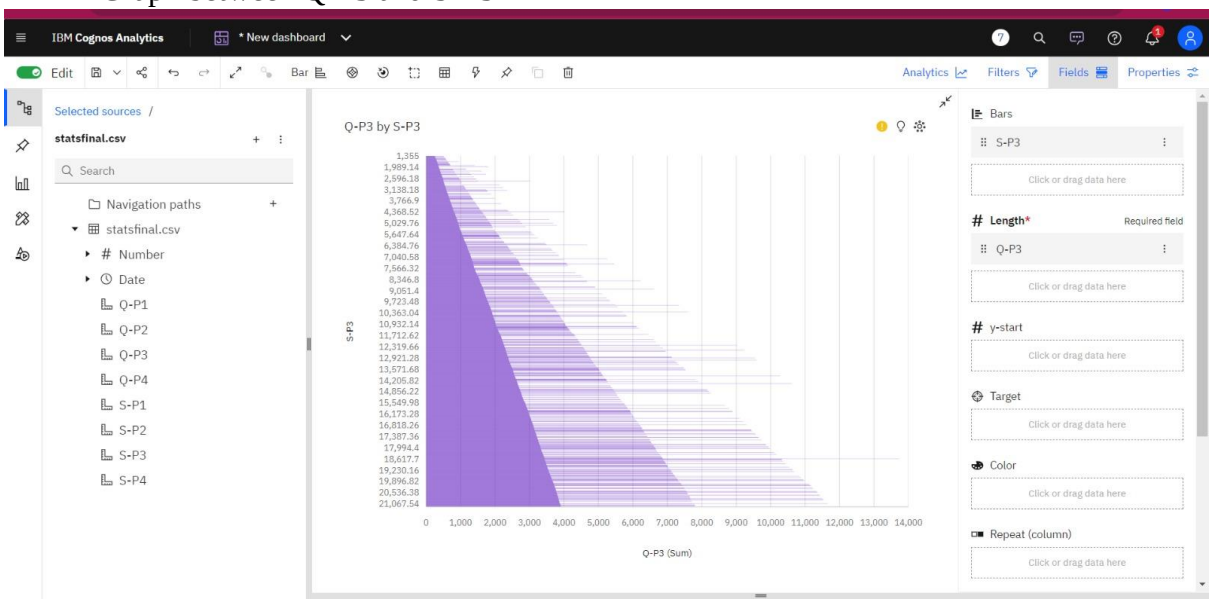
- Graph between Q-P1 and S-P1



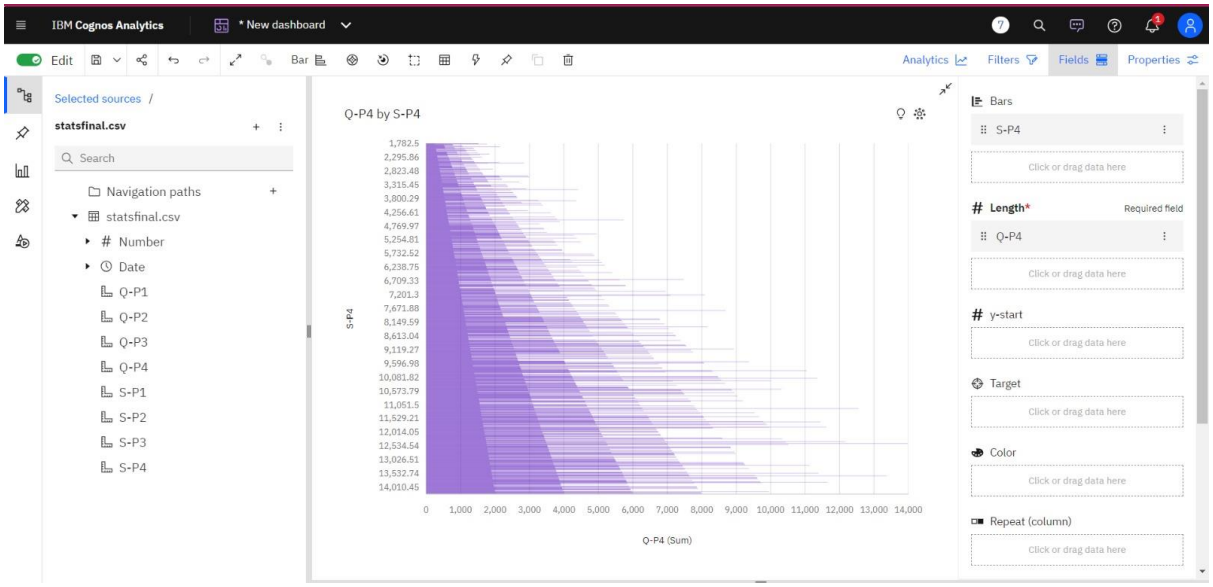
- Graph between Q-P2 and S-P2



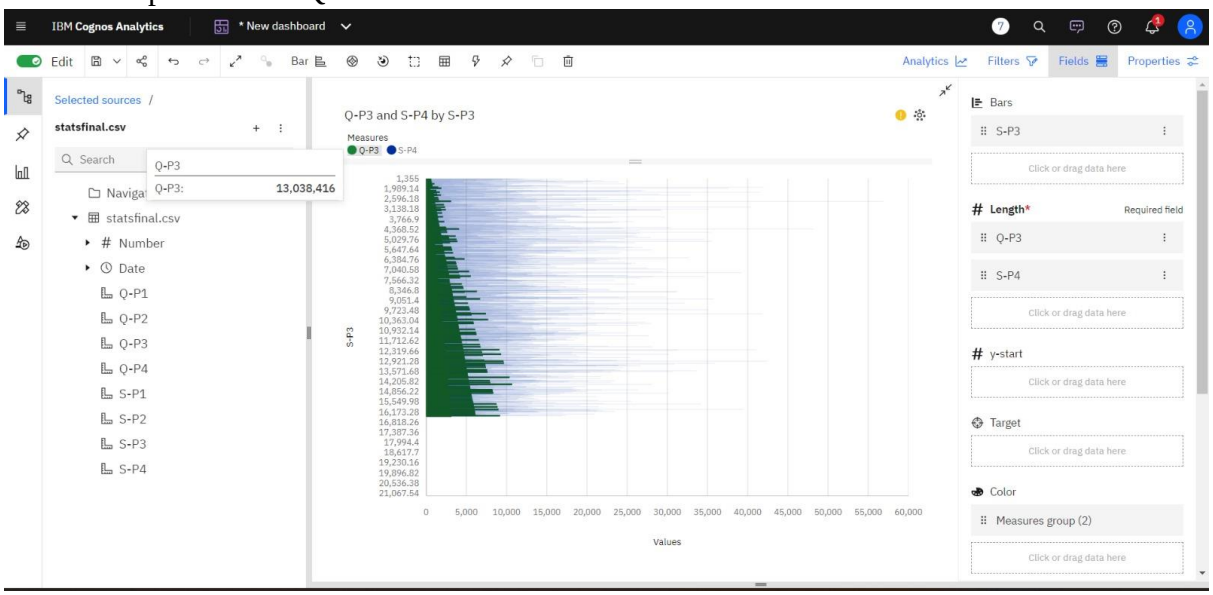
- Graph between Q-P3 and S-P3

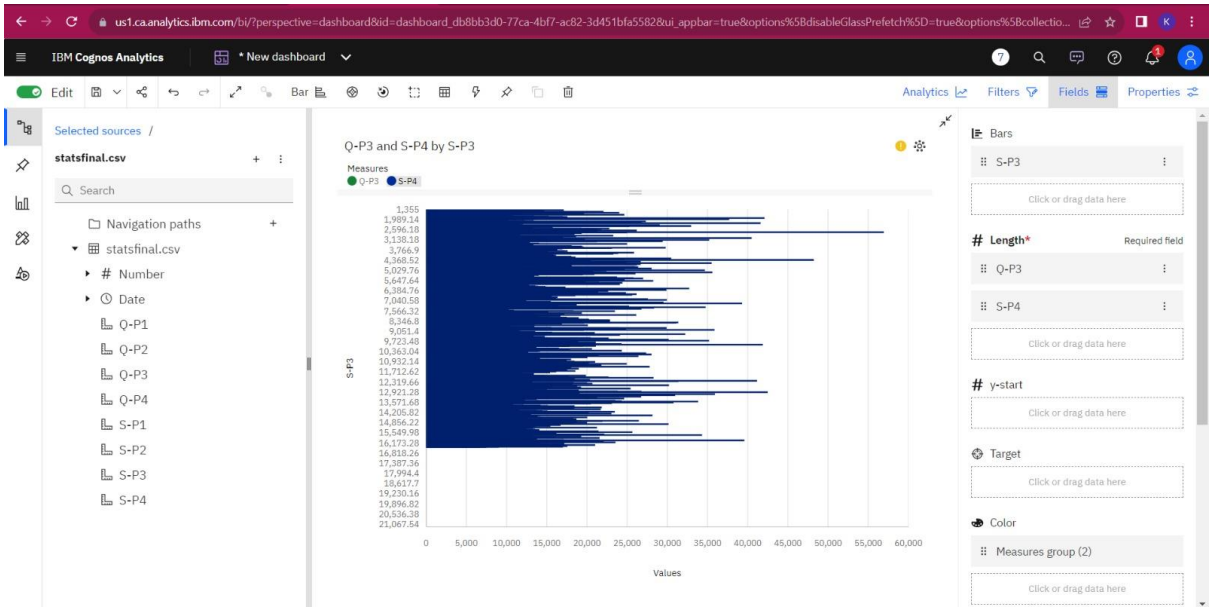


- Graph between Q-P4 and S-P4



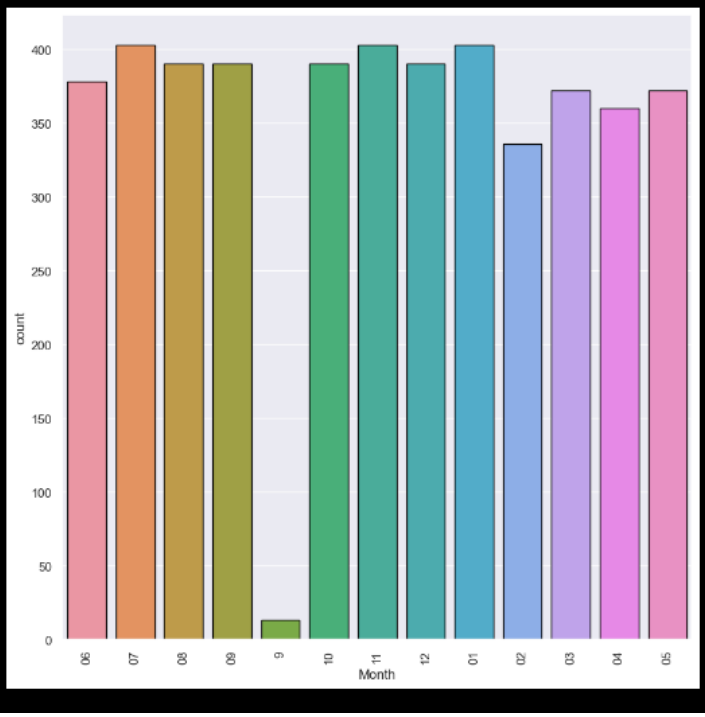
- Graph between Q-P3 and S-P3






```
# which is the most occuring month
print(data["Month"].value_counts())
plt.figure(figsize=(10,10))
sns.countplot(x="Month",data=data,edgecolor="black")
plt.xticks(rotation=90);
```

```
Month
07    403
11    403
01    403
08    390
09    390
10    390
12    390
06    378
03    372
05    372
04    360
02    336
9      13
Name: count, dtype: int64
```



```python
# which is the most occuring Day
print(data["Day"].value_counts())
plt.figure(figsize=(10,10))
sns.countplot(x="Day",data=data,edgecolor="black")
plt.xticks(rotation=90);
```

Python

```
Day
13    152
23    152
03    152
02    152
01    152
14    152
27    152
25    152
24    152
26    152
22    152
18    152
15    152
16    152
21    152
17    152
19    152
20    152
12    151
11    151
10    151
04    151
05    151
06    151
...
29    140
30    127
31     89
Name: count, dtype: int64
```

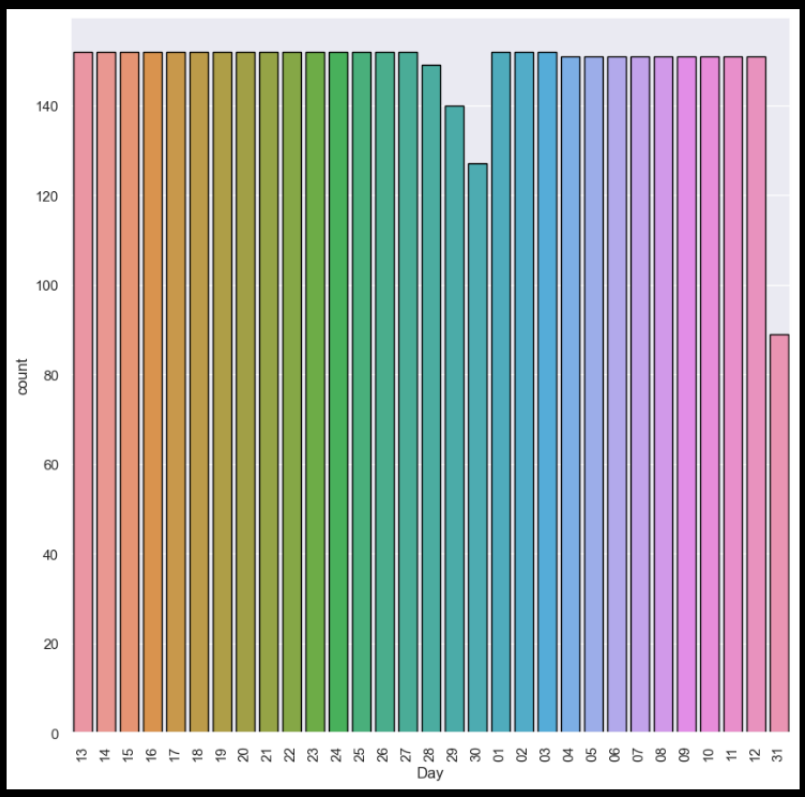
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

phase_3.ipynb > # which is the most occuring Day

Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ...

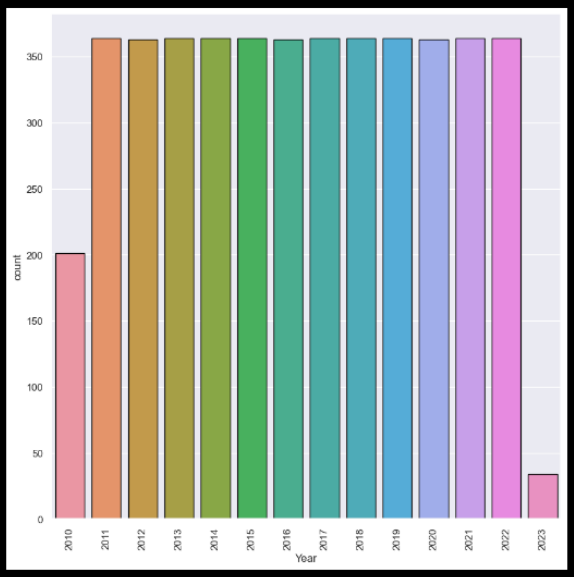
31 89
Name: count, dtype: int64

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



```python
# which is the most occuring year
print(data["Year"].value_counts())
plt.figure(figsize=(10,10))
sns.countplot(x="Year",data=data,edgecolor="black")
plt.xticks(rotation=90);
```

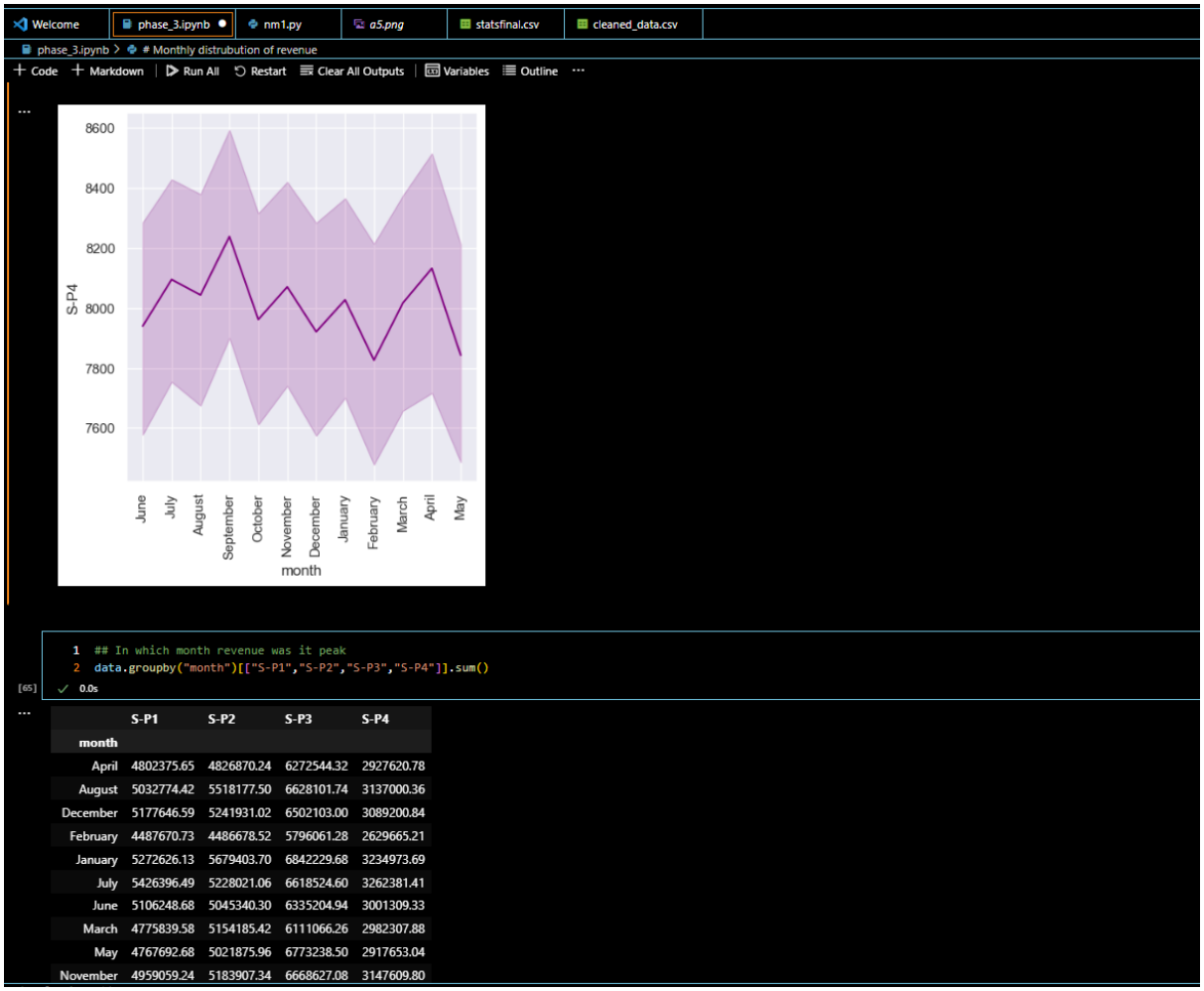
```
Year
2011    364
2013    364
2014    364
2015    364
2017    364
2018    364
2019    364
2021    364
2022    364
2012    363
2016    363
2020    363
2010    201
2023     34
Name: count, dtype: int64
```



Visualization :

Monthly distribution of revenue


Welcome
phase_3.ipynb
nm1.py
a5.png
statsfinal.csv
cleaned_data.csv
phase_3.ipynb > # Monthly distrubution of revenue
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
month
S-P2
14500
14000
13500
13000
12500
June
July
August
September
October
November
December
January
February
March
April
May
month
S-P3
19000
18500
18000
17500
17000
16500
16000
15500
June
July
August
September
October
November
December
January
February
March
April
May
month



month	S-P1	S-P2	S-P3	S-P4
April	4802375.65	4826870.24	6272544.32	2927620.78
August	5032774.42	5518177.50	6628101.74	3137000.36
December	5177646.59	5241931.02	6502103.00	3089200.84
February	4487670.73	4486678.52	5796061.28	2629665.21
January	5272626.13	5679403.70	6842229.68	3234973.69
July	5426396.49	5228021.06	6618524.60	3262381.41
June	5106248.68	5045340.30	6335204.94	3001309.33
March	4775839.58	5154185.42	6111066.26	2982307.88
May	4767692.68	5021875.96	6773238.50	2917653.04
November	4959059.24	5183907.34	6668627.08	3147609.80

Which month revenue was it peak

Welcome | phase_3.ipynb | nm1.py | a5.png | statsfinal.csv | cleaned_data.csv

phase_3.ipynb > # Monthly distrubution of revenue

+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ...


```
## In which month revenue was it peak
data.groupby("month")[["S-P1","S-P2","S-P3","S-P4"]].sum()
```

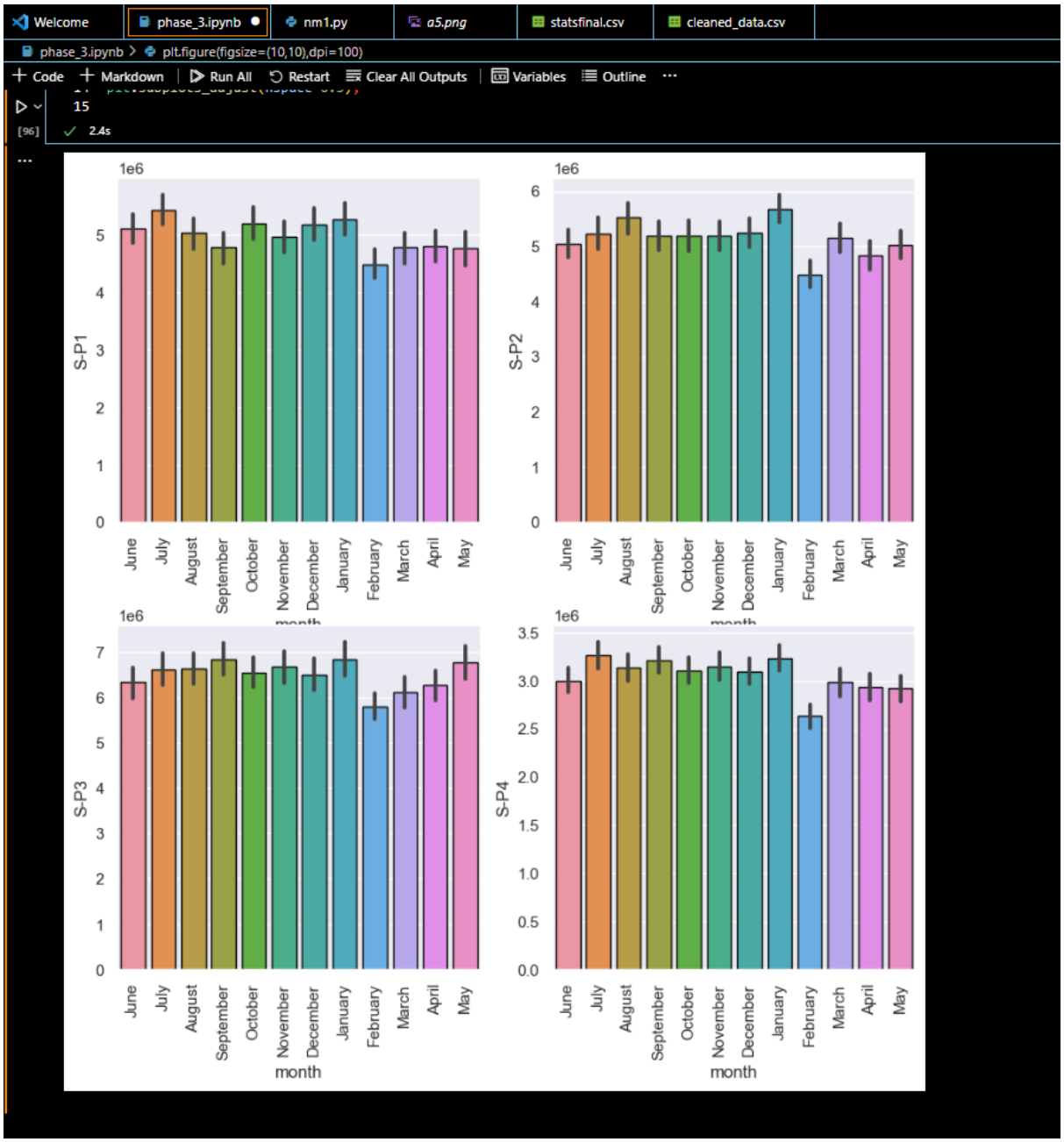
[65] ✓ 0.0s

month	S-P1	S-P2	S-P3	S-P4
April	4802375.65	4826870.24	6272544.32	2927620.78
August	5032774.42	5518177.50	6628101.74	3137000.36
December	5177646.59	5241931.02	6502103.00	3089200.84
February	4487670.73	4486678.52	5796061.28	2629665.21
January	5272626.13	5679403.70	6842229.68	3234973.69
July	5426396.49	5228021.06	6618524.60	3262381.41
June	5106248.68	5045340.30	6335204.94	3001309.33
March	4775839.58	5154185.42	6111066.26	2982307.88
May	4767692.68	5021875.96	6773238.50	2917653.04
November	4959059.24	5183907.34	6668627.08	3147609.80
October	5192694.58	5182607.64	6552210.90	3105264.73
September	4785710.96	5190659.44	6837535.96	3213063.20

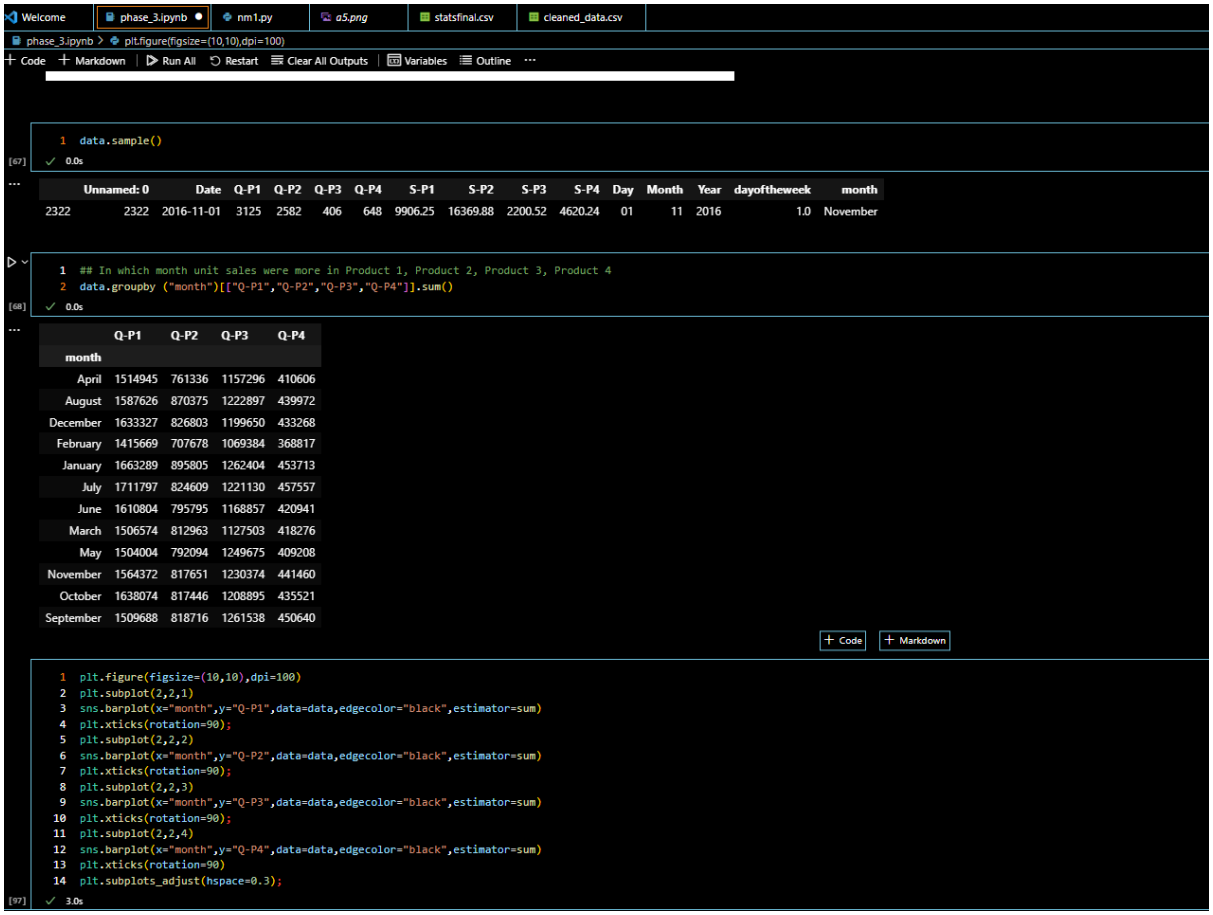
```
plt.figure(figsize=(10,10),dpi=100)
plt.subplot(2,2,1)
sns.barplot(x="month",y="S-P1",data=data,edgecolor="black",estimator=sum)
plt.xticks(rotation=90);
plt.subplot(2,2,2)
sns.barplot(x="month",y="S-P2",data=data,edgecolor="black",estimator=sum)
plt.xticks(rotation=90);
plt.subplot(2,2,3)
sns.barplot(x="month",y="S-P3",data=data,edgecolor="black",estimator=sum)
plt.xticks(rotation=90);
plt.subplot(2,2,4)
sns.barplot(x="month",y="S-P4",data=data,edgecolor="black",estimator=sum)
plt.xticks(rotation=90)
plt.subplots_adjust(hspace=0.3);

```

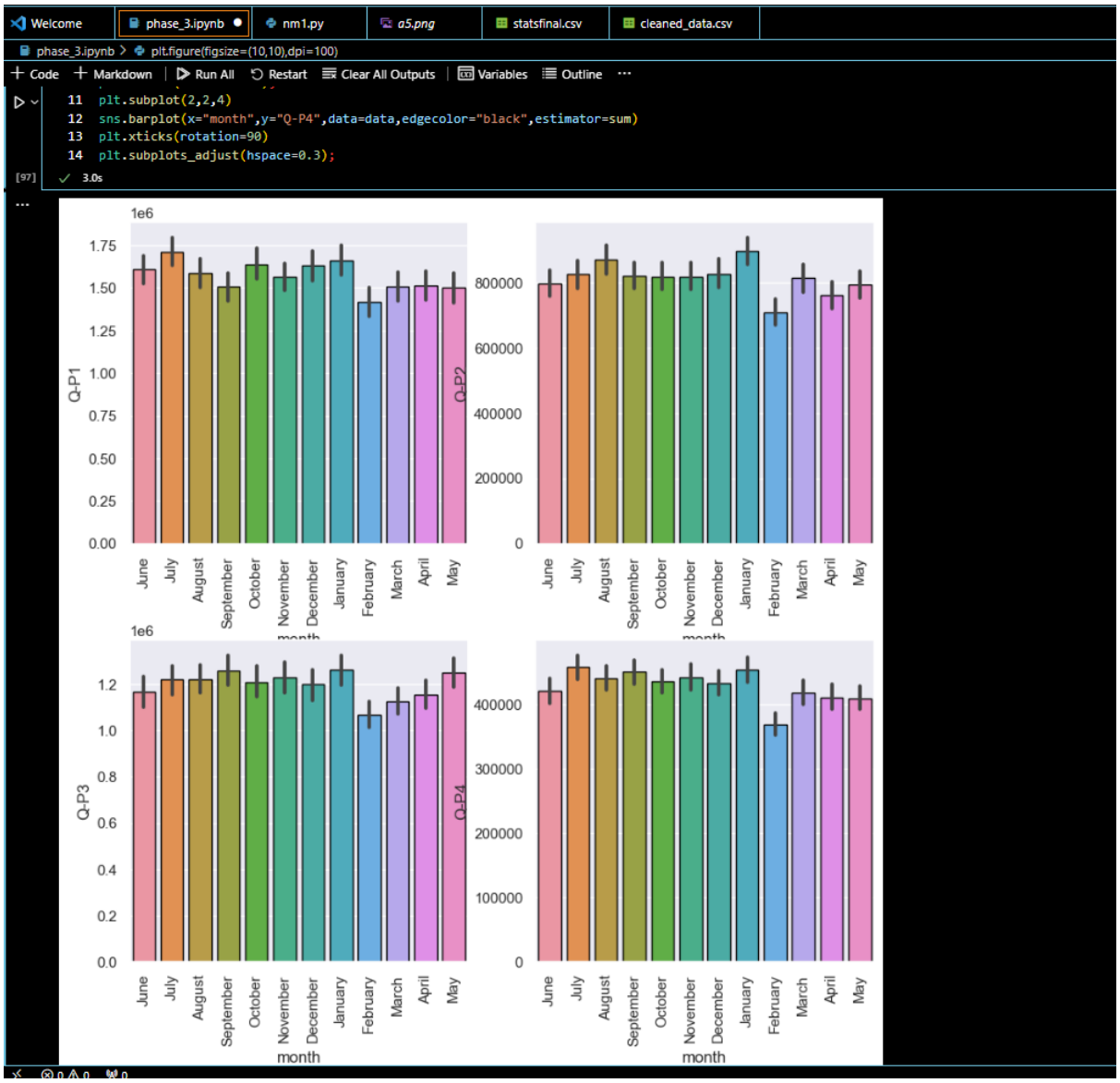
[96] ✓ 2.4s

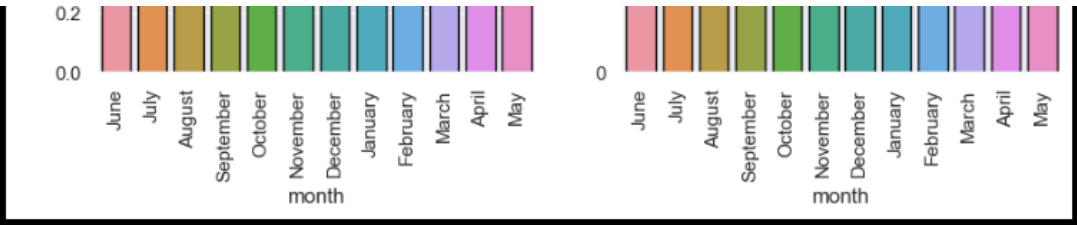
Dashboards:


Welcome
phase_3.ipynb
nm1.py
a5.png
statsfinal.csv
cleaned_data.csv
phase_3.ipynb > plt.figure(figsize=(10,10),dpi=100)
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
15
[96]
2.4s
S-P1
S-P2
S-P3
S-P4
1e6
June
July
August
September
October
November
December
January
February
March
April
May
month

Which month unit sales were more in product 1,2,3,4




Welcome
phase_3.ipynb
nm1.py
a5.png
statsfinal.csv
cleaned_data.csv
phase_3.ipynb > plt.figure(figsize=(10,10),dpi=100)
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
11 plt.subplot(2,2,4)
12 sns.barplot(x="month",y="Q-P4",data=data,edgecolor="black",estimator=sum)
13 plt.xticks(rotation=90)
14 plt.subplots_adjust(hspace=0.3);
[97] 3.0s
1e6
Q-P1
Q-P2
Q-P3
Q-P4
June
July
August
September
October
November
December
January
February
March
April
May
month

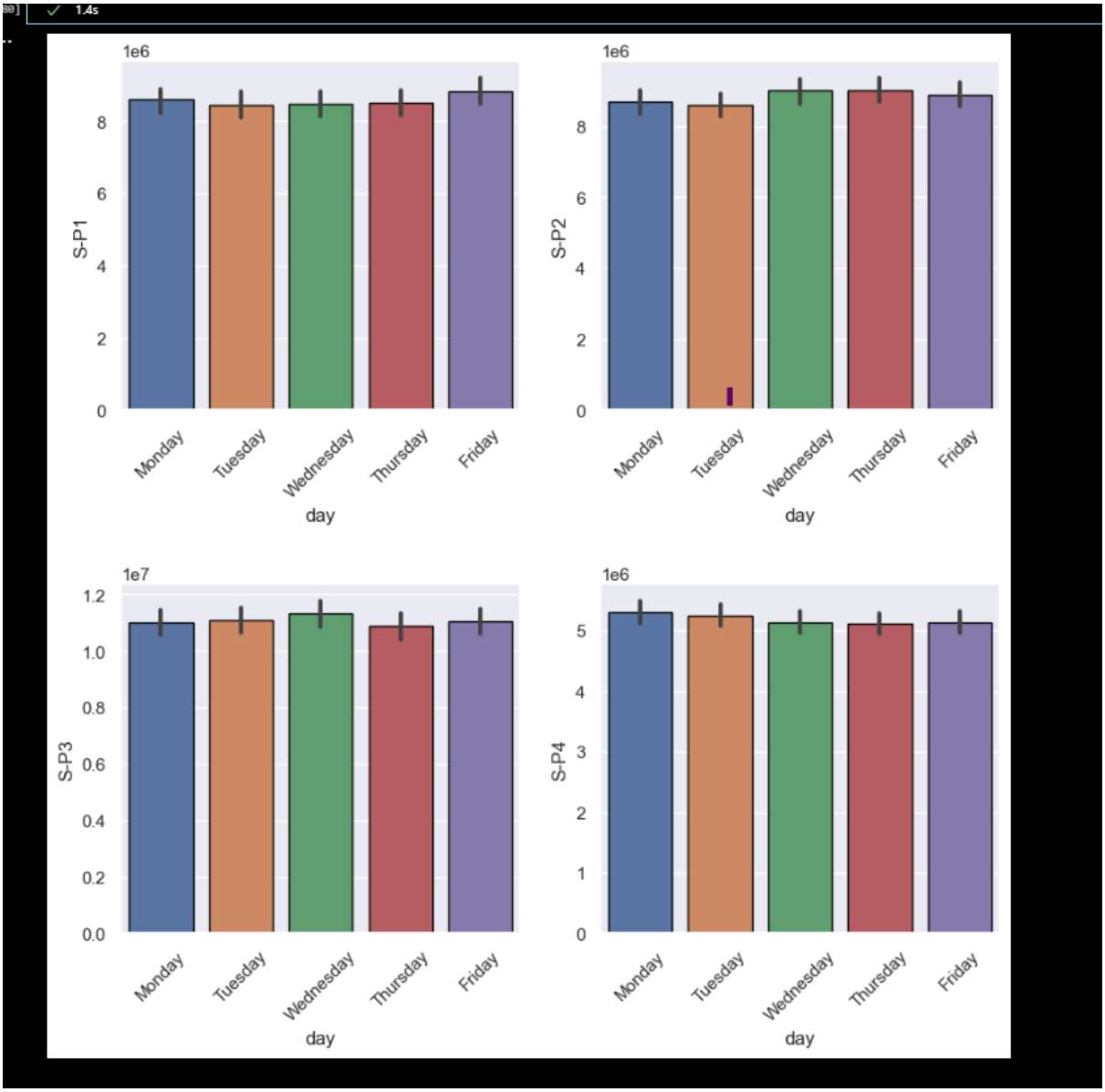



```
[72]  1  data["day"]=data["Date"].dt.day_name()
      ✓ 0.0s
```

```
[79]  1  week_t=data[data["dayoftheweek"]<5]
      2  weekend_t=data[data["dayoftheweek"]>=5]
      3  print(week_t.groupby("day")[["S-P1","S-P2","S-P3","S-P4"]].sum())
      ✓ 0.0s
```

```
                S-P1        S-P2         S-P3        S-P4
day
Friday     8838549.62  8898088.56  11056312.20  5142733.53
Monday     8590452.74  8675681.36  11016047.02  5305654.03
Thursday   8499752.70  9010725.00  10895462.86  5112751.88
Tuesday    8458961.14  8590928.24  11084062.60  5253576.51
Wednesday  8476478.56  8995331.48  11328873.16  5139403.82
```

```
[80]  1  plt.figure(figsize=(10,10),dpi=100)
      2  plt.subplot(2,2,1)
      3  sns.barplot(x="day",y="S-P1",data=week_t,edgecolor="black",estimator=sum)
      4  plt.xticks(rotation=45);
      5  plt.subplot(2,2,2)
      6  sns.barplot(x="day",y="S-P2",data=week_t,edgecolor="black",estimator=sum)
      7  plt.xticks(rotation=45);
      8  plt.subplot(2,2,3)
      9  sns.barplot(x="day",y="S-P3",data=week_t,edgecolor="black",estimator=sum)
     10  plt.xticks(rotation=45);
     11  plt.subplot(2,2,4)
     12  sns.barplot(x="day",y="S-P4",data=week_t,edgecolor="black",estimator=sum)
     13  plt.xticks(rotation=45)
     14  plt.subplots_adjust(hspace=0.5);
      ✓ 1.4s
```


1.4s
1e6
S-P1
0
2
4
6
8
Monday
Tuesday
Wednesday
Thursday
Friday
day
1e6
S-P2
0
2
4
6
8
Monday
Tuesday
Wednesday
Thursday
Friday
day
1e7
S-P3
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Monday
Tuesday
Wednesday
Thursday
Friday
day
1e6
S-P4
0
1
2
3
4
5
Monday
Tuesday
Wednesday
Thursday
Friday
day

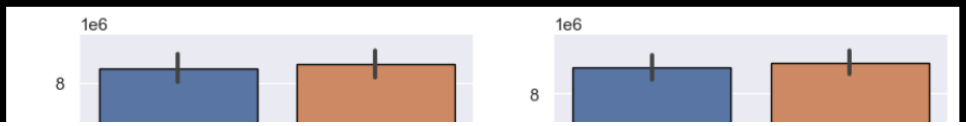

```
[81] ✓ 0.0s
```

```
print(weekend_t.groupby("day")[["S-P1","S-P2","S-P3","S-P4"]].sum())
```

```
              S-P1        S-P2         S-P3        S-P4
day
Saturday  8527781.84  8865500.96  11572057.72  5411691.39
Sunday    8394759.13  8723402.54  10984632.70  5282239.11
```

```
[82] ✓ 1.2s
```

```
plt.figure(figsize=(10,10),dpi=100)
plt.subplot(2,2,1)
sns.barplot(x="day",y="S-P1",data=weekend_t,edgecolor="black",estimator=sum)
plt.xticks(rotation=45);
plt.subplot(2,2,2)
sns.barplot(x="day",y="S-P2",data=weekend_t,edgecolor="black",estimator=sum)
plt.xticks(rotation=45);
plt.subplot(2,2,3)
sns.barplot(x="day",y="S-P3",data=weekend_t,edgecolor="black",estimator=sum)
plt.xticks(rotation=45);
plt.subplot(2,2,4)
sns.barplot(x="day",y="S-P4",data=weekend_t,edgecolor="black",estimator=sum)
plt.xticks(rotation=45)
plt.subplots_adjust(hspace=0.5);
```



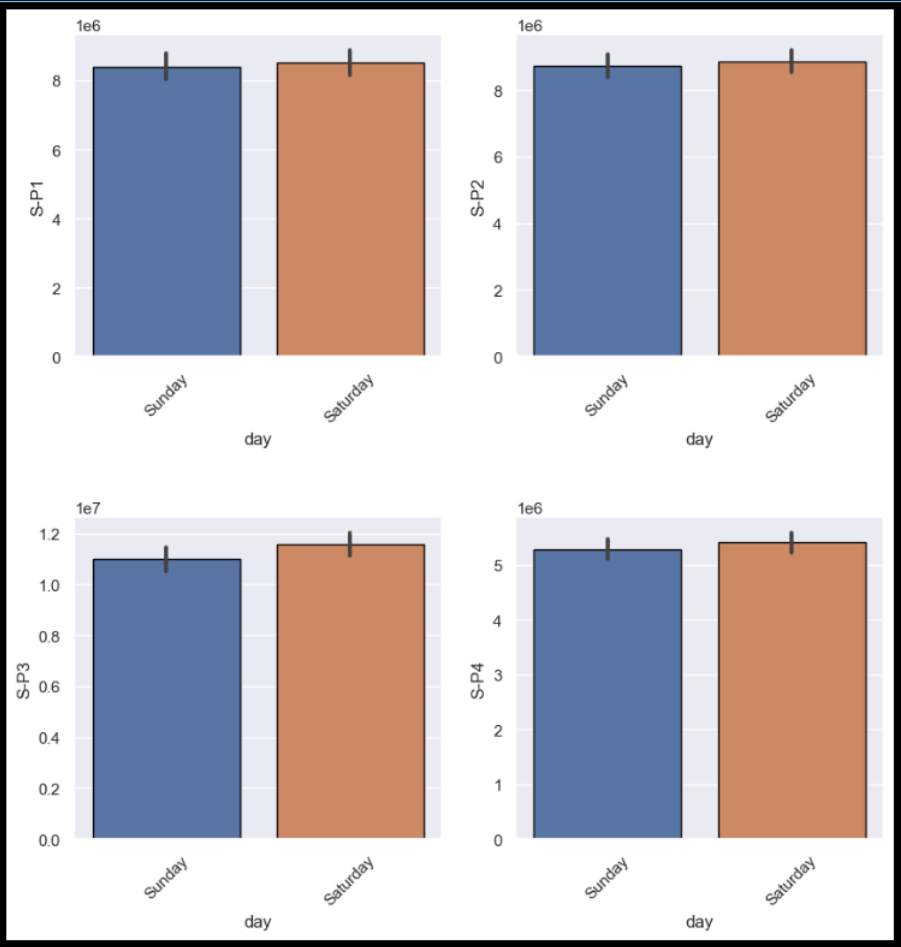
Welcome | phase_3.ipynb | nm1.py | a5.png | statsfinal.csv | cleaned_data.csv

phase_3.ipynb > plt.figure(figsize=(10,10),dpi=100)

+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ...

```
12  sns.barplot(x="day",y="S-P4",data=weekend_t,edgecolor="black",estimator=sum)
13  plt.xticks(rotation=45)
14  plt.subplots_adjust(hspace=0.5);
```

[82] ✓ 1.2s



Welcome | phase_3.ipynb | nm1.py | a5.png | statsfinal.csv | cleaned_data.csv

phase_3.ipynb > plt.figure(figsize=(10,10),dpi=100)

+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ...

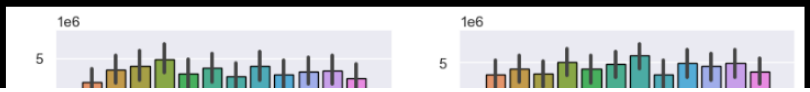
```
# In which year revenue was the highest
data.groupby("Year")[["S-P1","S-P2","S-P3","S-P4"]].agg(["sum"])
```

[85] ✓ 0.0s

	S-P1	S-P2	S-P3	S-P4
	sum	sum	sum	sum
Year				
2010	2573948.07	2747185.40	3415277.50	1582774.44
2011	4550956.61	4765727.28	6292061.74	2939855.86
2012	4782296.87	4880512.98	6203677.80	2979769.60
2013	4859923.83	4783694.84	6049337.88	2890459.22
2014	4983696.48	5002697.46	6301936.98	2878723.24
2015	4704904.49	4868891.76	6027777.12	2946479.63
2016	4810801.51	4969875.28	6030823.16	2911998.95
2017	4658996.55	5128064.62	6323340.56	2987612.60
2018	4854598.23	4762937.68	6244219.40	2841661.50
2019	4699914.91	4979264.82	6153120.04	2926330.25
2020	4749791.69	4934561.48	6370001.34	2993559.02
2021	4770834.15	4986054.96	6323692.86	2904569.49
2022	4627657.93	4821633.40	6043760.70	2780093.95
2023	476482.70	496428.34	650562.60	284665.25

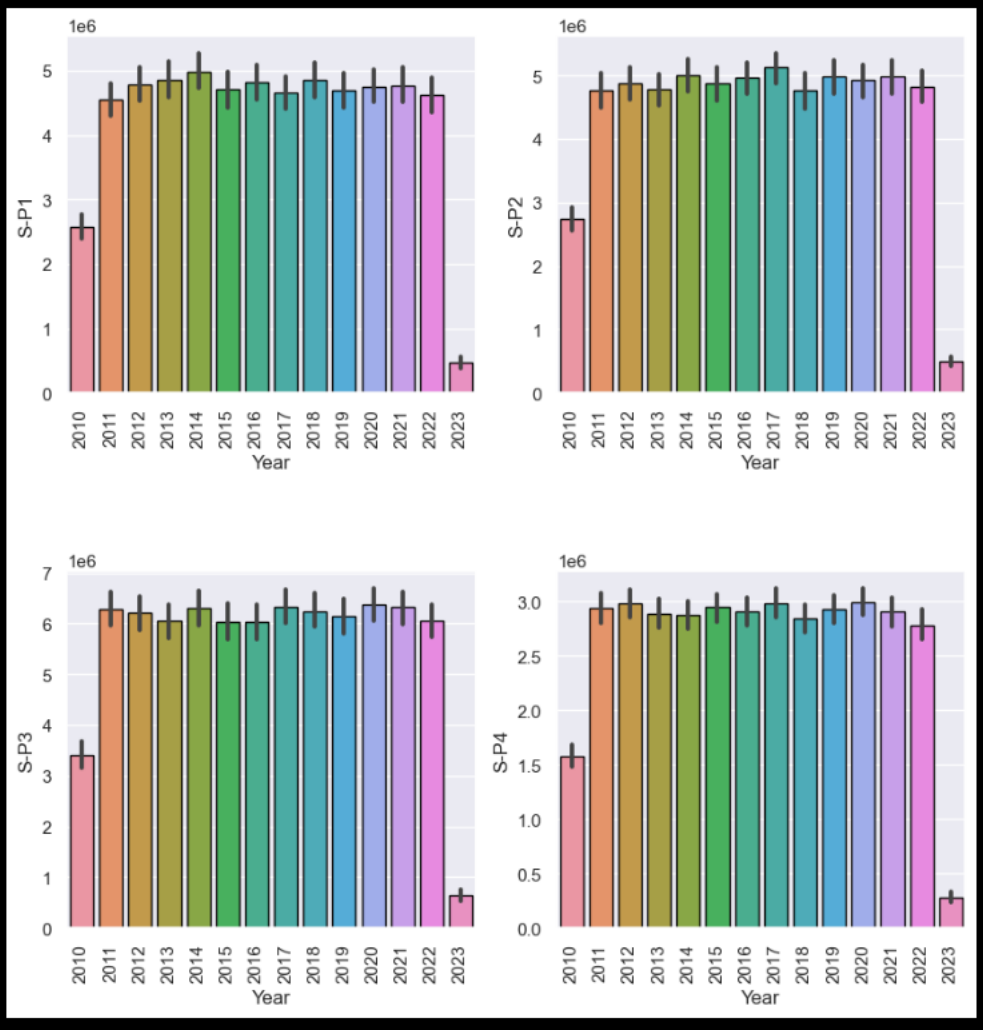
```
plt.figure(figsize=(10,10),dpi=100)
plt.subplot(2,2,1)
sns.barplot(x="Year",y="S-P1",data=data,edgecolor="black",estimator=sum)
plt.xticks(rotation=90);
plt.subplot(2,2,2)
sns.barplot(x="Year",y="S-P2",data=data,edgecolor="black",estimator=sum)
plt.xticks(rotation=90);
plt.subplot(2,2,3)
sns.barplot(x="Year",y="S-P3",data=data,edgecolor="black",estimator=sum)
plt.xticks(rotation=90);
plt.subplot(2,2,4)
sns.barplot(x="Year",y="S-P4",data=data,edgecolor="black",estimator=sum)
plt.xticks(rotation=90)
plt.subplots_adjust(hspace=0.5);
```

[86] ✓ 2.9s



```
plt.subplot(2,2,4)
sns.barplot(x="Year",y="S-P4",data=data,edgecolor="black",estimator=sum)
plt.xticks(rotation=90)
plt.subplots_adjust(hspace=0.5);
```

[86] ✓ 2.9s



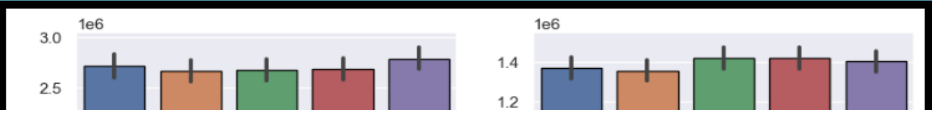
```
## What was the avg revenue, maximum and min
data[["S-P1","S-P2","S-P3","S-P4"]].agg(["sum","max","min","mean"])
```

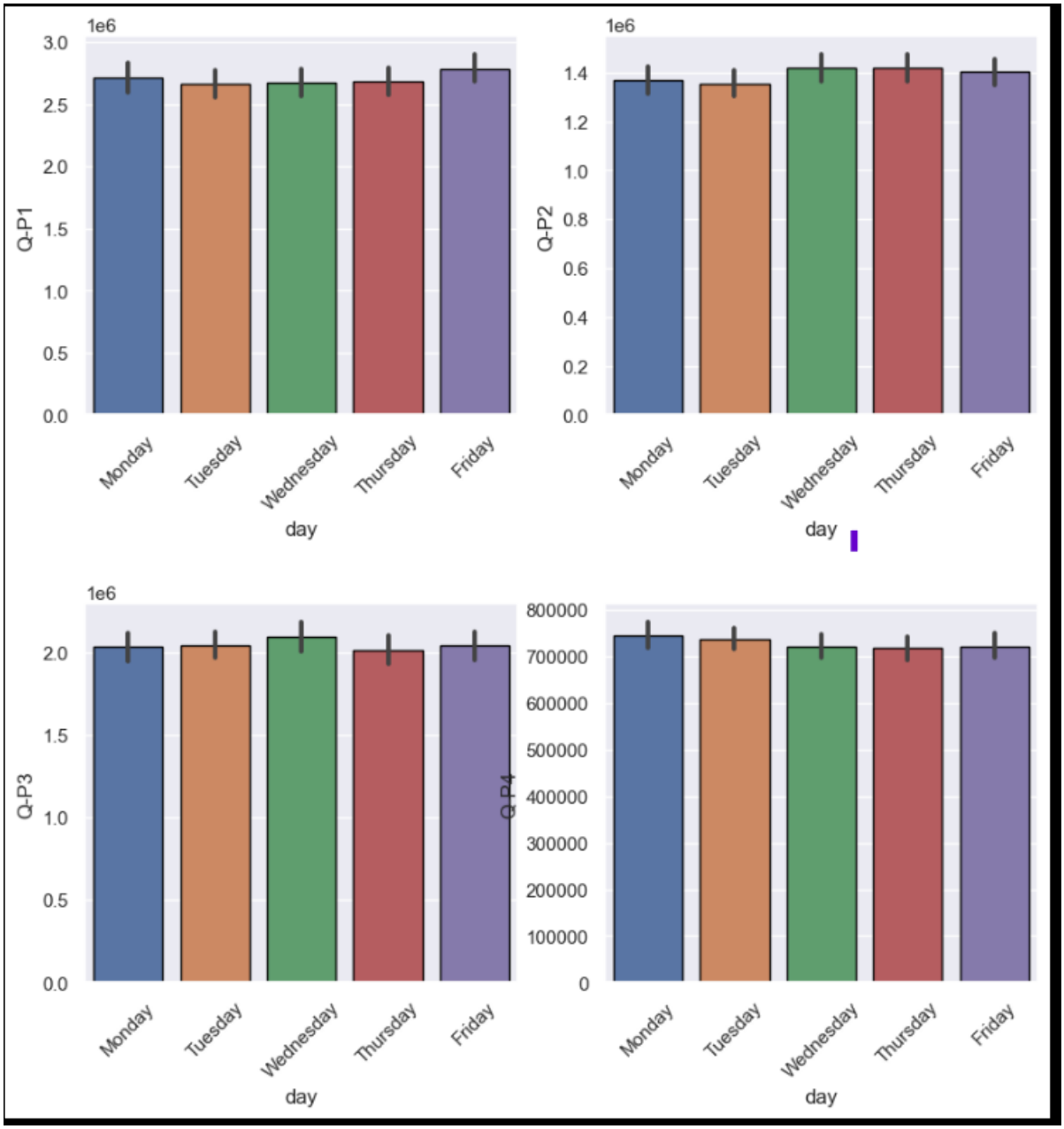
[87] ✓ 0.0s

	S-P1	S-P2	S-P3	S-P4
sum	6.010480e+07	6.212753e+07	7.842959e+07	3.684855e+07
max	2.535366e+04	2.534732e+04	3.252000e+04	1.426000e+04
min	8.051800e+02	1.591340e+03	1.355000e+03	1.782500e+03
mean	1.306626e+04	1.350598e+04	1.704991e+04	8.010555e+03

```
plt.figure(figsize=(10,10),dpi=100)
plt.subplot(2,2,1)
sns.barplot(x="day",y="Q-P1",data=week_t,edgecolor="black",estimator=sum)
plt.xticks(rotation=45);
plt.subplot(2,2,2)
sns.barplot(x="day",y="Q-P2",data=week_t,edgecolor="black",estimator=sum)
plt.xticks(rotation=45);
plt.subplot(2,2,3)
sns.barplot(x="day",y="Q-P3",data=week_t,edgecolor="black",estimator=sum)
plt.xticks(rotation=45);
plt.subplot(2,2,4)
sns.barplot(x="day",y="Q-P4",data=week_t,edgecolor="black",estimator=sum)
plt.xticks(rotation=45)
plt.subplots_adjust(hspace=0.5);
```

[88] ✓ 1.3s




1e6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Q-P1
Monday
Tuesday
Wednesday
Thursday
Friday
day
1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Q-P2
Monday
Tuesday
Wednesday
Thursday
Friday
day
1e6
2.0
1.5
1.0
0.5
0.0
Q-P3
Monday
Tuesday
Wednesday
Thursday
Friday
day
800000
700000
600000
500000
400000
300000
200000
100000
0
Q-P4
Monday
Tuesday
Wednesday
Thursday
Friday
day

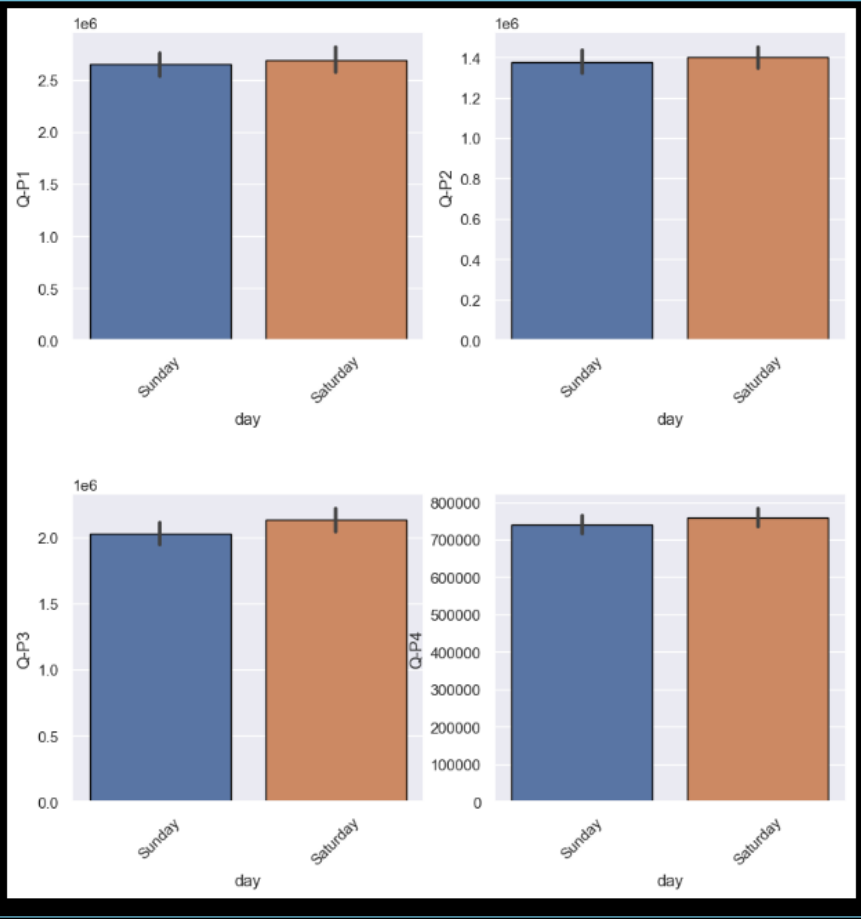
Welcome | phase_3.ipynb | nm1.py | a5.png | statsfinal.csv | cleaned_data.csv

phase_3.ipynb > plt.figure(figsize=(10,10),dpi=100)

+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ...

```
1  plt.figure(figsize=(10,10),dpi=100)
2  plt.subplot(2,2,1)
3  sns.barplot(x="day",y="Q-P1",data=weekend_t,edgecolor="black",estimator=sum)
4  plt.xticks(rotation=45);
5  plt.subplot(2,2,2)
6  sns.barplot(x="day",y="Q-P2",data=weekend_t,edgecolor="black",estimator=sum)
7  plt.xticks(rotation=45);
8  plt.subplot(2,2,3)
9  sns.barplot(x="day",y="Q-P3",data=weekend_t,edgecolor="black",estimator=sum)
10 plt.xticks(rotation=45);
11 plt.subplot(2,2,4)
12 sns.barplot(x="day",y="Q-P4",data=weekend_t,edgecolor="black",estimator=sum)
13 plt.xticks(rotation=45)
14 plt.subplots_adjust(hspace=0.5);
15
```

[89] ✓ 0.7s



Conclusion

Unit Sales 2011 - 2022

- P1 has the highest unit sales for each year. And it's highest is in year 2014.
- We can observe that P4 has the lowest unit sales of all the products.

Revenues 2011 - 2022

- We can observe that P3 brought in the most revenue. This could be as a result of multiple things:
  - P3 was sold for higher than the rest, as it had the second highest unit sales for each year.
- We can observe than P1 and P2 brought in similar revenues for each year. With P2 bringing in slightly more.
- P1 despite having the most unit sold, brought in the second lowest revenue each year.

Average Month Sales 2011 - 2022

- We can observe that all Products unit sales drop in Feb.
- We can observe that Feb and Dec have the lowest sales for each product
- For P1 We can observe Mar - Jul having the highest unit sales
- For P2 We can observe Jan, Mar - Aug having the highest unit sales
- For P3 We can observe May & Sep having the highest unit sales
- For P4 We can observe uniform sales from Jan - Dec

Estimated Unit Sales for 31st of Dec

This value can not be properly estimated with out Machine Learning. Currently we used the average for all the 31st days across all years for each product.

- Overall we can see that P1 has the highest unit sales on the 31st for each year, except for 2021 and 2022. (These could be as a result to Covid and other economy issues.)
- P3 has the second highest unit sales for all the 31st in each year.
- We can see that our previous observation correlate as Q-P1 has the higest estimate, followed by Q-P3
- We can approxiamte that the company will make:
    - Q-P1: 3850.73
    - Q-P2: 2076.51
    - Q-P3: 3196.45
    - Q-P4: 1112.13

Insights

- Added columns month, day and day of the week and changing the dtype of date from object to datetime64 through feature engineering.
- Drop columns unnamed as it was not providing any usefull information.
- S-P3 has gained the most revenue but the unit sale of Q-P1 is more.
- In 2016 most revenue most revenue generated and on fridays and staurdays most revenue generated.
- On Weekdays and weekend the S-P3 has the highest revenue whereas on weekend and weekday the Q-P1 has more unit sales.
- In month of October unit sale and revenue was at peak.