Data Analytics for Product Sales Analysis with IBM Cognos

Name: Sivaram A

Reg no - 731721106304

NM ID- au22ECLE01

Title: Innovation Phase_5

Introduction:

Data Analytics with Cognos Product Sales Analysis provides organizations with valuable insights into their sales performance. However, to enhance this analytical capability, incorporating machine learning algorithms is essential. This document explores how machine learning can be integrated to predict future sales trends and customer behaviors more accurately.

1. Problem Statement:

In traditional sales analysis, past data is used to make informed decisions about future sales and customer behaviors. While this approach is valuable, it is limited in its ability to adapt to dynamic market conditions and emerging trends. Machine learning algorithms offer the potential to predict future sales trends and customer behaviors more accurately, thereby empowering organizations to make proactive decisions.

Phase 1: Problem Definition and Design Thinking

In this phase, we will outline our approach to solving the problem of analyzing sales data for improving inventory management and marketing strategies.

Design Thinking Steps

Step 1: Analysis Objectives
- Identify top-selling products.
- Analyze sales trends.
- Understand customer preferences.

Step 2: Data Collection
We will collect data from the following sources:
- Transaction records.
- Product information.
- Customer demographics.

Step 3: Visualization Strategy
To visualize our insights, we will utilize IBM Cognos to create interactive dashboards and reports.

Step 4: Actionable Insights
The insights derived from our analysis will guide inventory management and marketing strategies.

In this phase, we will outline our approach to solving the problem of analyzing sales data for improving inventory management and marketing strategies.

Design Thinking Steps

Step 1: Analysis Objectives
- Identify top-selling products.
- Analyze sales trends.
- Understand customer preferences.

Step 2: Data Collection
We will collect data from the following sources:
- Transaction records.
- Product information.
- Customer demographics.

Step 3: Visualization Strategy
To visualize our insights, we will utilize IBM Cognos to create interactive dashboards and reports.

Step 4: Actionable Insights
The insights derived from our analysis will guide inventory management and marketing strategies.

Title: Innovation Phase_2

Task: Import the dataset and perform data cleaning & data analysis

1. Notebook

Types of Problems in Data Science

1. Classification
2. Regression
3. Clustering
4. Natural Language Processing
5. Recommendation Systems
6. Image Recognition
7. Big Data and Distributed Computing

Classification

Involves categorizing data points into predefined classes or categories.

Eg: Classifying emails as spam or not spam, identifying whether a patient has disease or not, categorizing images of animals into species

Concepts for classification:

Logistic Regression: Statistical model that predicts the probability of a binary outcome(eg:yes/no)

Decision Trees: Tree Like structure that make decisions by evaluating features at each node

Random Forests: Ensembles of multiple decision trees to improve accuracy and reduce overfitting.

Support Vector Machines (SVM): Powerful algorithm for bianry and multiclass classification by finding the optimal hyperplane taht best seperates classes.

Neural Networks: Deep Learning Models composed of layers of interconnected neurons, capable of handling complex classification tasks.

Regression

Involves preidcting a continuous numerical value. Eg: Predicting housing prices based on features, forecasting future sales, or estimating the temparature based on Historical Data.

Concepts for regression:

Linear Regression: Statistical technique that models the relationship betweena dependent variable and one or more independent variable

Polynomial Regression:Extends linear regression by fitting a polynomial equation to the data.

Ridge Regression and lasso Regression: Techniques that add regularization to linear regression models to prevent overfitting.

Neural Networks: Deep Learning Models composed of layers of interconnected neurons, capable of handling complex classification

Clustering

Involves grouping of similar data points without predefined categories.

Eg: Customer Segmentation for marketing or clustering documents by topic

Concepts for Clustering:

K-Means Clustering: A partitioning method that divides data into K clusters based on similarity.

Hierarchical Clustering: Builds a tree-like hierarchy of clusters, useful for exploring data at different levels.

DBSCAN(Density-Based Spatial Clustering of Applications with Noise): Clusters data points based on their density, suitable for irregularly shaped clusters.

Notebook Link:

https://colab.research.google.com/drive/15_IxEf7I-775x_aI30dNyvpzFauJgO83#scrollTo=I5nDp_Ww5zcO

DataSet

https://drive.google.com/file/d/1-SGBoro0m1_mbcUEkxu7K4uzeUBG4mQC/view?usp=sharing

About Dataset

Greetings , fellow analysts ! REC corp LTD. is a small-scale business venture established in India. They have been selling FOUR PRODUCTS for OVER TEN YEARS . The products are P1, P2, P3 and P4. They have collected data from their retail centers and organized it into a small csv file , which has been given to you. The excel file contains about 8 numerical parameters :

- 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

Understanding the Data

Fetching rows and columns

fetching column names

Basic info

Checking null values

Checking Dtypes

Basic statistical info

CODE

```
df.shape
```

```
df.columns
```

```
df.info()
```

```
df.isnull().sum()
```

```
df.dtypes
```

```
df.duplicated().sum()
```

```
df.describe().T
```

Cleaning the Data

Changing dtype

Filling the NaT values with average of time

fetching month,day of week, weekday

Dropping column unnamed as it is not useful for us

Code

```
df.sample(2)

from datetime import datetime as dt

df[df["Date"]=="31-9-2010"]

df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

df[df['Date'].isnull()]

df["Date"].fillna(df["Date"].mean(),inplace=True)

df['Date'].isnull().sum()

df.dtypes

df["month"]=df["Date"].dt.month_name()

df["day"]=df["Date"].dt.day_name()

df["dayoftheweek"]=df["Date"].dt.weekday

df["year"]=df["Date"].dt.year

df.sample()

df.drop(columns=["Unnamed: 0"],inplace=True)

df.sample()

df.corr().T

plt.figure(figsize=(10,10))

sns.heatmap(df.corr(),annot=True)
```

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

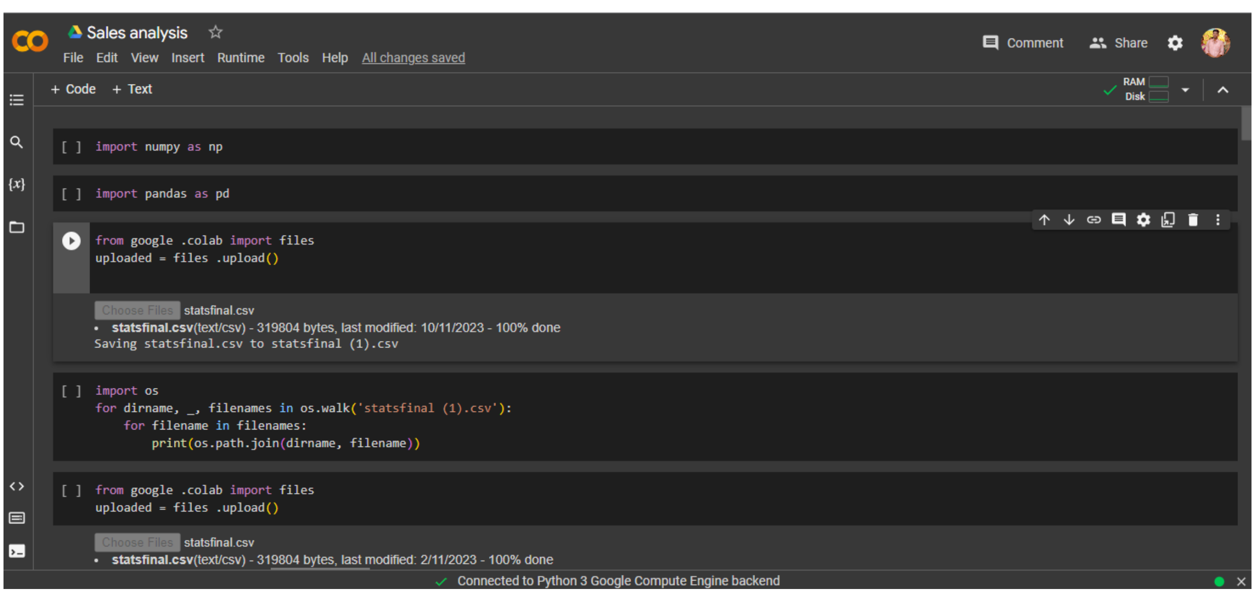
Data Analysis

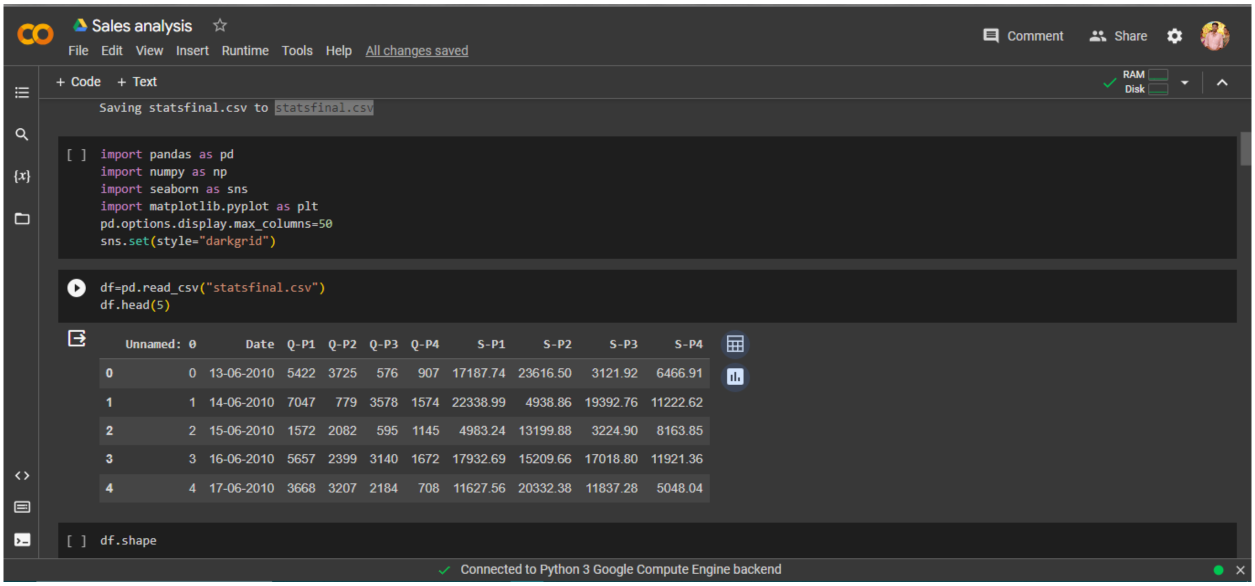
Analysis the Data through the Python code

Sales analysis

https://colab.research.google.com/drive/1d3PCu5_NhTyP80NYDCE7BUkgj3mwzrt_?usp=sharing

Sample Output





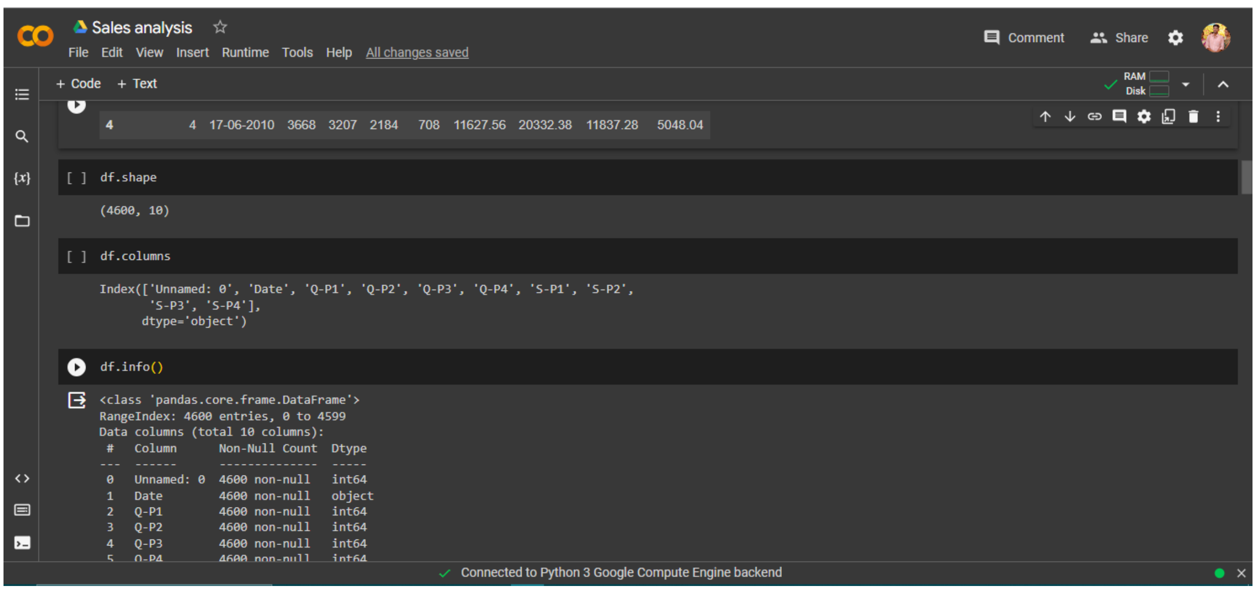

```
Saving statsfinal.csv to statsfinal.csv
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
pd.options.display.max_columns=50
sns.set(style="darkgrid")
```

```
df=pd.read_csv("statsfinal.csv")
df.head(5)
```

	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

```
df.shape
```



4	4	17-06-2010	3668	3207	2184	708	11627.56	20332.38	11837.28	5048.04

```
df.shape
```

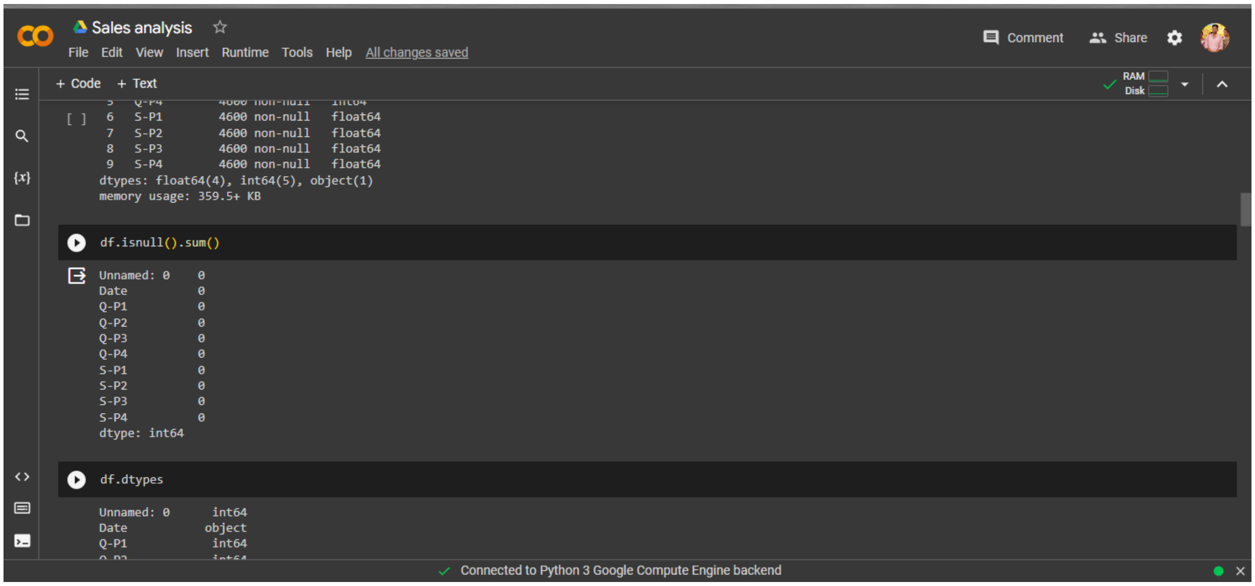
```
(4600, 10)
```

```
df.columns
```

```
Index(['Unnamed: 0', 'Date', 'Q-P1', 'Q-P2', 'Q-P3', 'Q-P4', 'S-P1', 'S-P2',
       'S-P3', 'S-P4'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  4600 non-null   int64
 1   Date        4600 non-null   object
 2   Q-P1        4600 non-null   int64
 3   Q-P2        4600 non-null   int64
 4   Q-P3        4600 non-null   int64
 5   Q-P4        4600 non-null   int64
```


Sales analysis
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
6 S-P1 4600 non-null float64
7 S-P2 4600 non-null float64
8 S-P3 4600 non-null float64
9 S-P4 4600 non-null float64
dtypes: float64(4), int64(5), object(1)
memory usage: 359.5+ KB
df.isnull().sum()
Unnamed: 0 0
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
dtype: int64
df.dtypes
Unnamed: 0 int64
Date object
Q-P1 int64
Connected to Python 3 Google Compute Engine backend

Sales analysis

File Edit View Insert Runtime Tools Help All changes saved

```
S-P2          0
S-P3          0
S-P4          0
dtype: int64
```

```
df.dtypes
```

```
Unnamed: 0      int64
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
dtype: object
```

```
df.duplicated().sum()
```

```
0
```

```
df.describe().T
```

Connected to Python 3 Google Compute Engine backend

Sales analysis

File Edit View Insert Runtime Tools Help All changes saved

```
0
```

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	4600.0	2299.500000	1328.049949	0.00	1149.750	2299.500	3449.250	4599.00
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

```
df.sample(2)
```

Unnamed: 0	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4

Connected to Python 3 Google Compute Engine backend

Title: Innovation Phase_3

Task: Perform Data Visualization

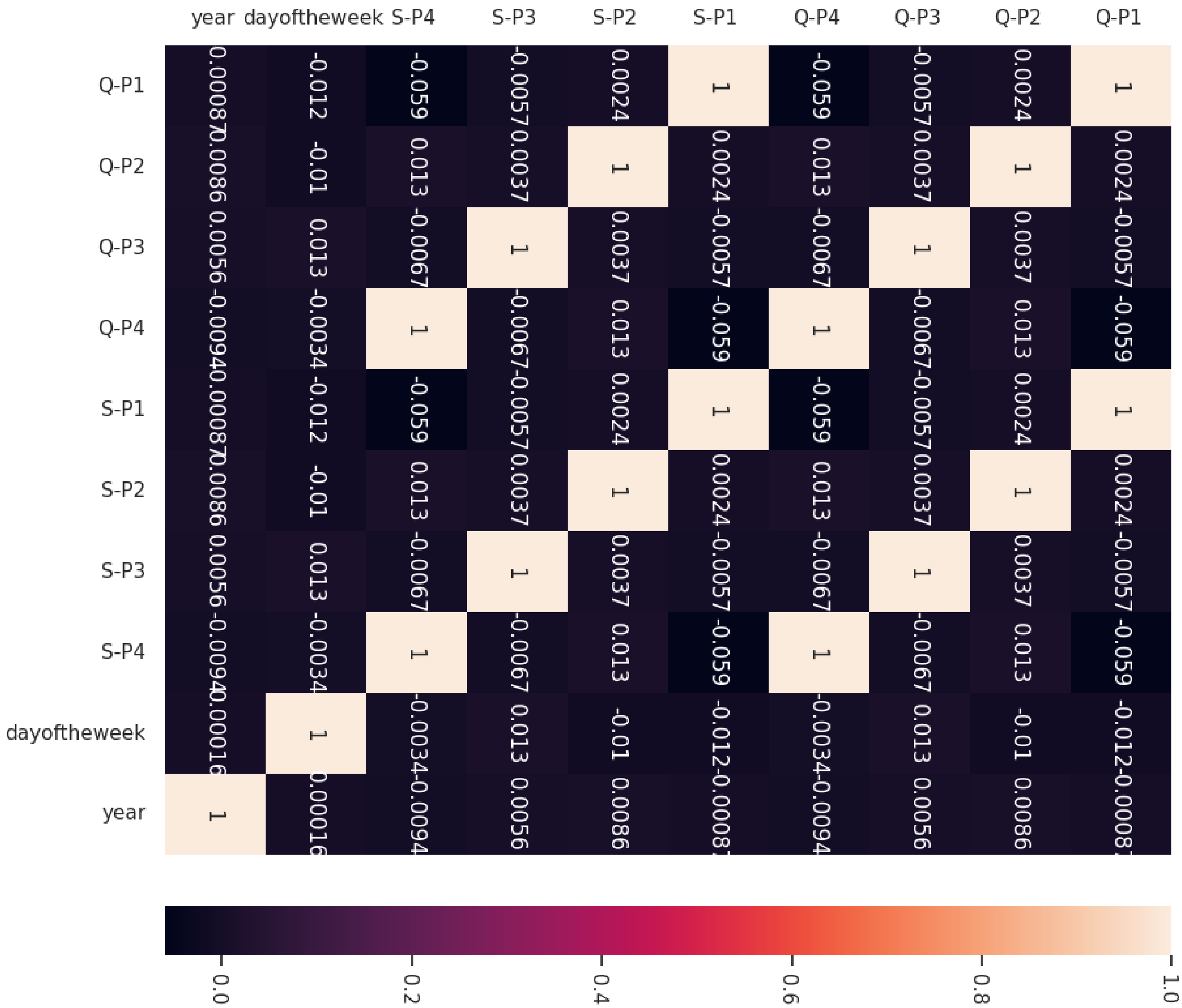
1. **Data Visualization**

   Code and Outputs

   1. **Code**

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

Out



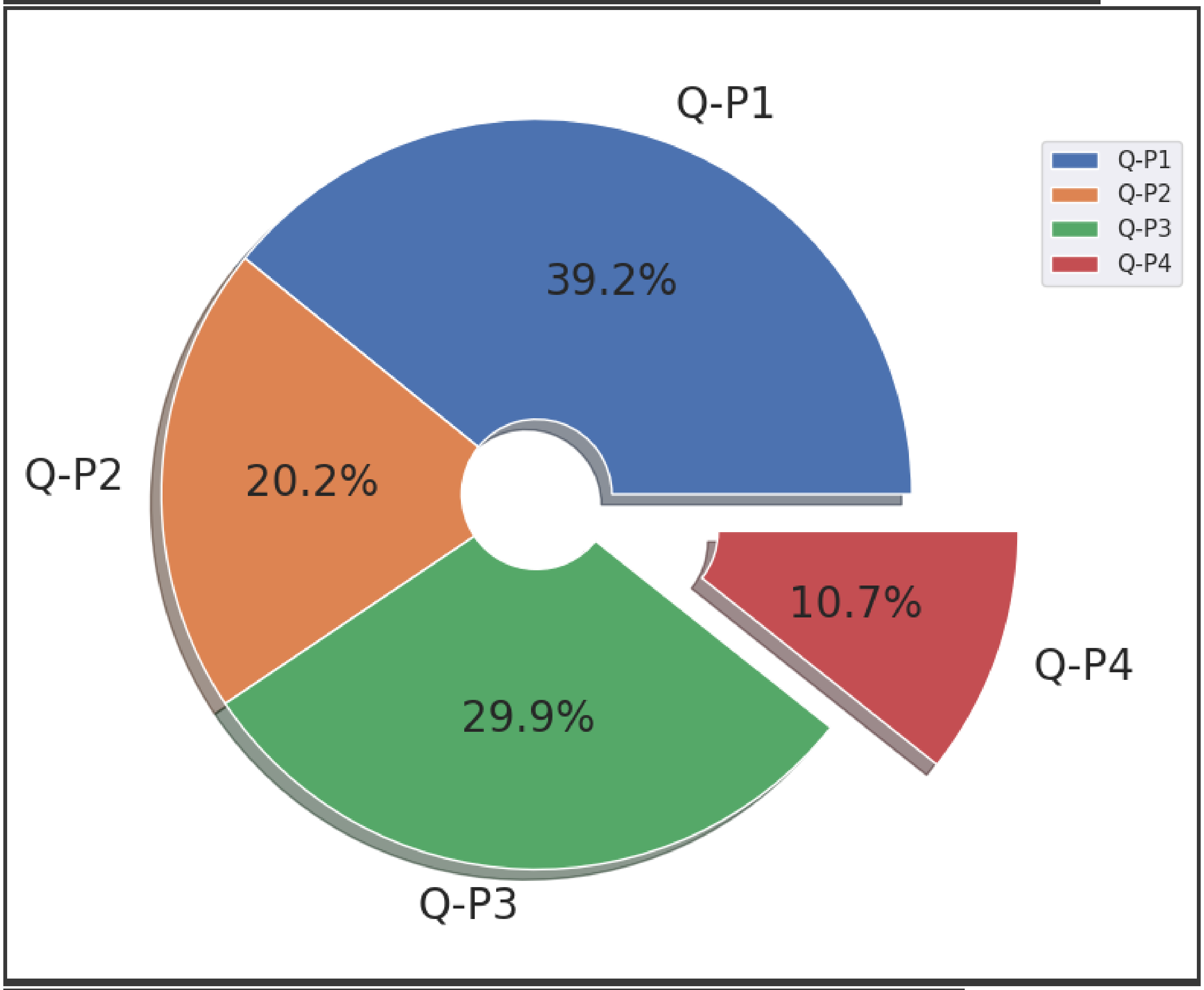
2. Code

```
q = df[["Q-P1","Q-P2","Q-P3","Q-P4"]].sum()

print(q)

plt.figure(figsize=(8,8))

plt.pie(q,labels=df[["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));
```

Out

```
Q-P1                                                                              1
```




```
8960506 Q-P2 9799295 Q-P3 14470404 Q-P4 5168100 dtype: int64
```

3. Code

```
s=df[["S-P1","S-P2","S-P3","S-P4"]].sum()

print(s)

plt.figure(figsize=(8,8))

plt.pie(s,labels=df[["S-P1","S-P2","S-P3","S-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))
```

Out

S-P1 60104804.02

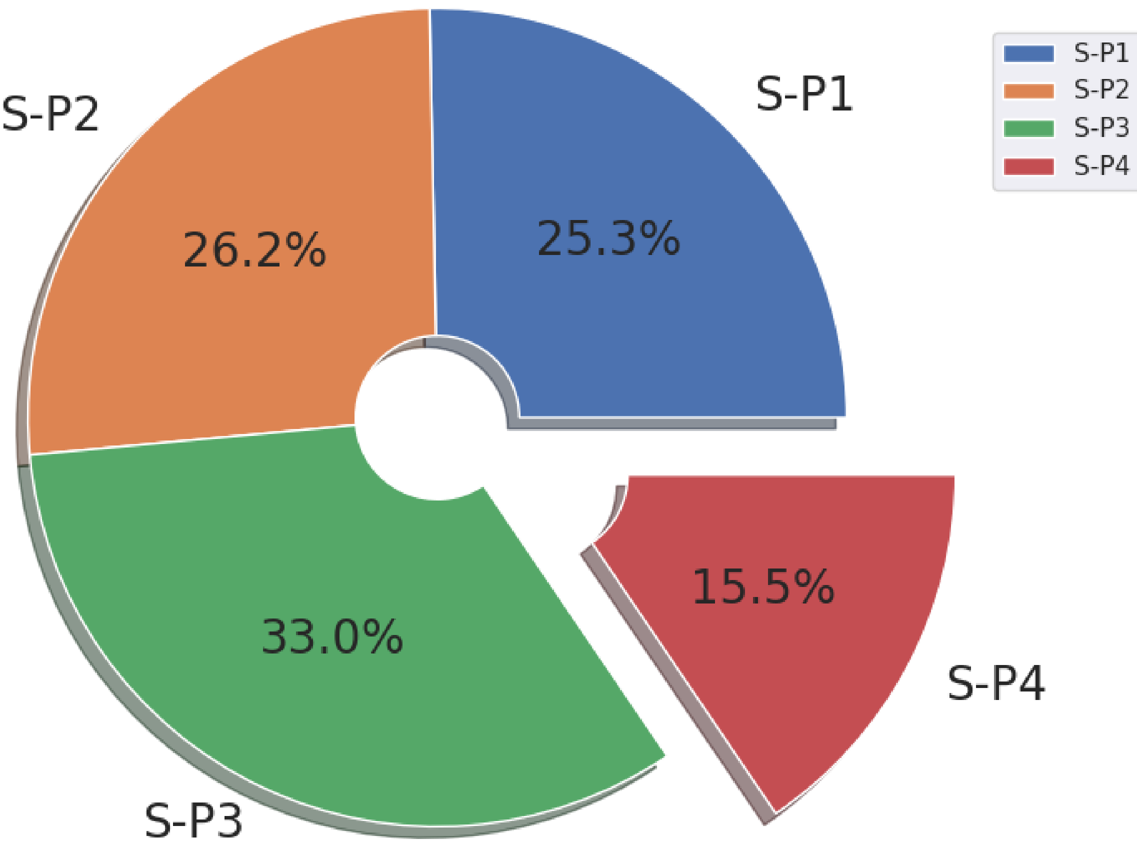
S-P2 62127530.30

S-P3 78429589.68

S-P4 36848553.00

dtype: float64

<matplotlib.legend.Legend at 0x79ead813ff10>

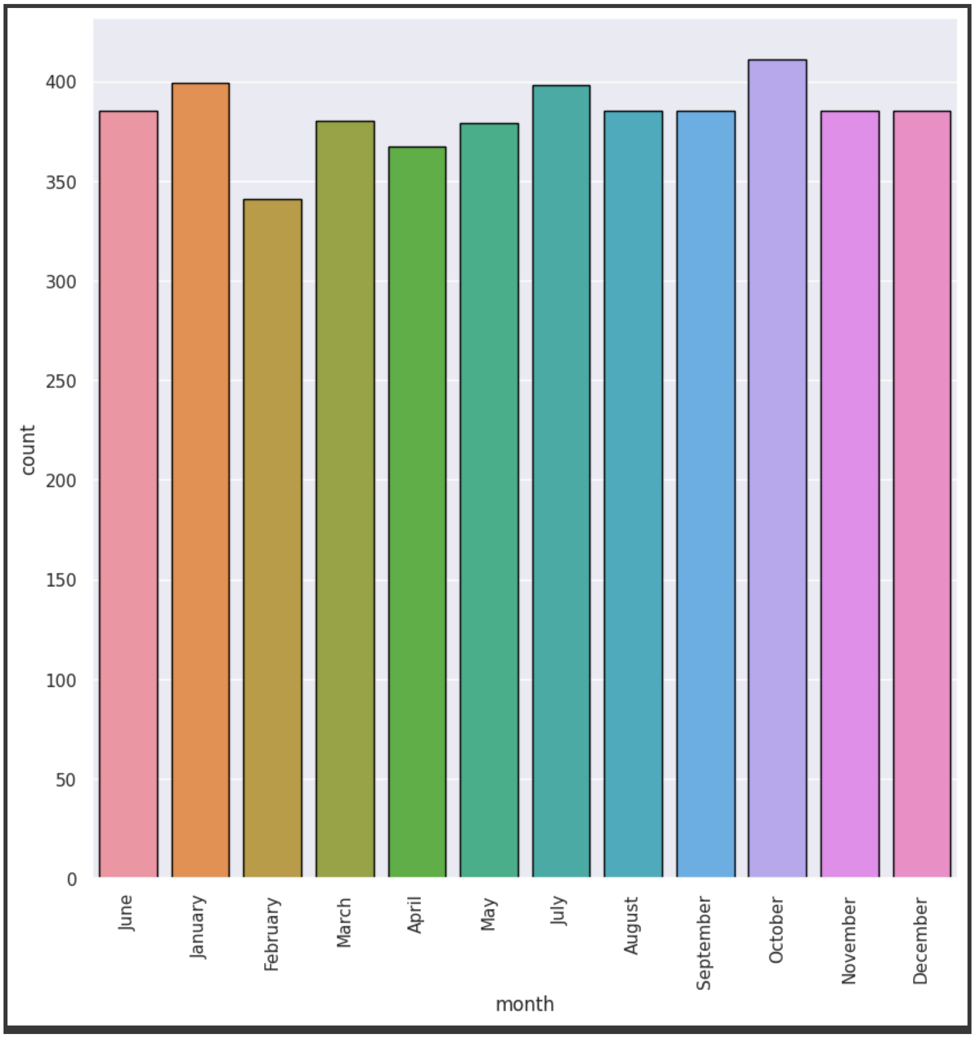


4. Code

```
print(df["month"].value_counts())

plt.figure(figsize=(10,10))

sns.countplot(x="month",data=df,edgecolor="black")

plt.xticks(rotation=90);
```

Out

```
October 411 January 399 July 398 June 385 August 385 September 385
November 385 December 385 March 380 May 379 April 367 February 341 Name:
month, dtype: int64
```

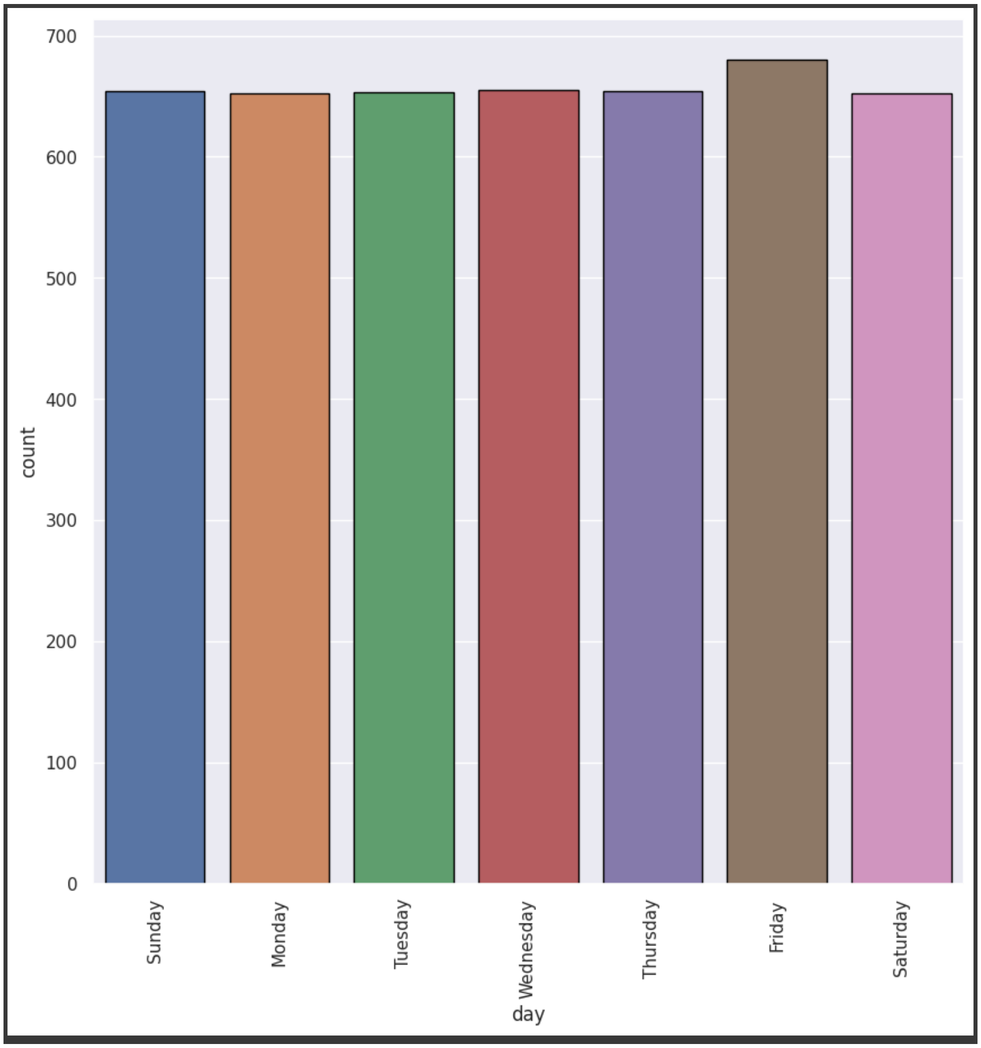


5. Code

```
print(df["day"].value_counts())

plt.figure(figsize=(10,10))

sns.countplot(x="day",data=df,edgecolor="black")

plt.xticks(rotation=90);
```

Out

```
Friday 680 Wednesday 655 Sunday 654 Thursday 654 Tuesday 653
Monday 652 Saturday 652 Name: day, dtype: int64
```

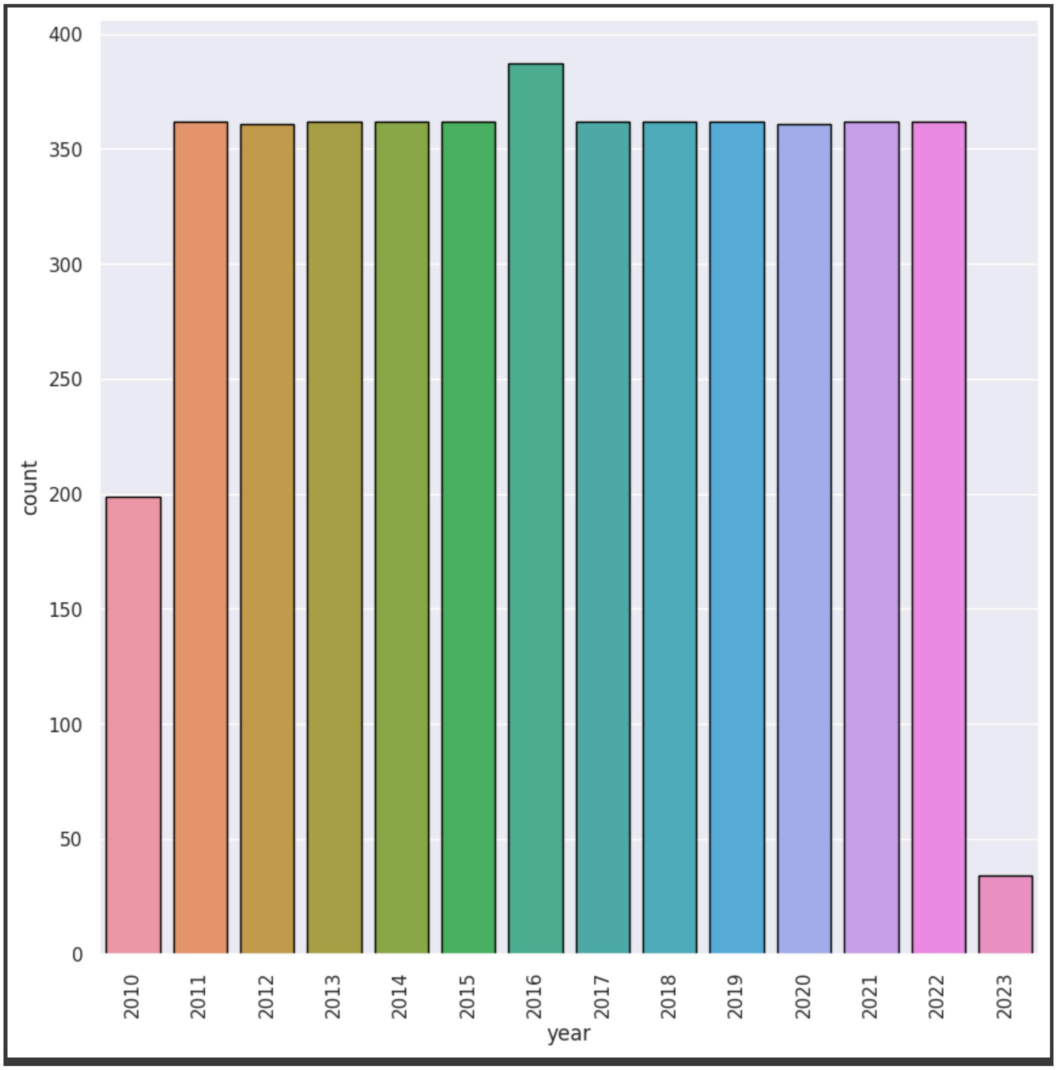


6. Code

```
print(df["year"].value_counts())

plt.figure(figsize=(10,10))

sns.countplot(x="year",data=df,edgecolor="black")

plt.xticks(rotation=90);
```

Out

```
2016 387 2011 362 2013 362 2014 362 2015 362 2017 362 2018 362
2019 362 2021 362 2022 362 2012 361 2020 361 2010 199 2023 34
Name: year, dtype: int64
```

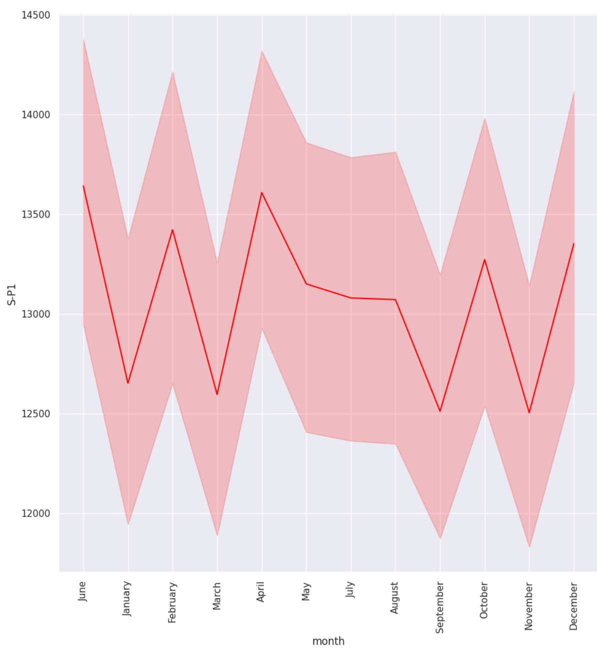


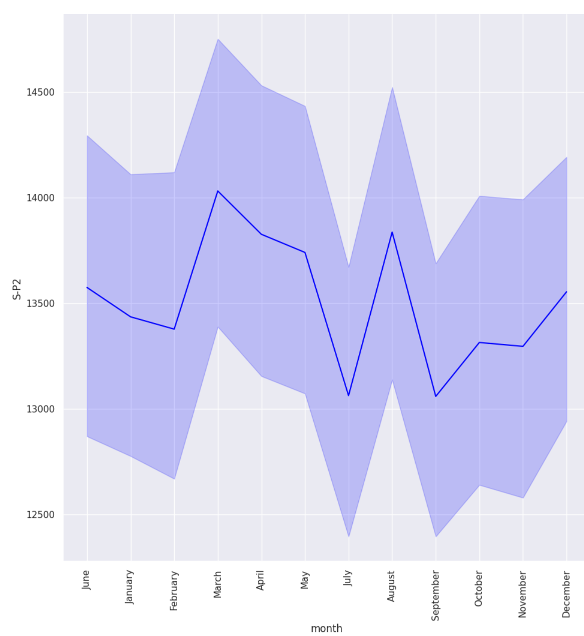
7. Code

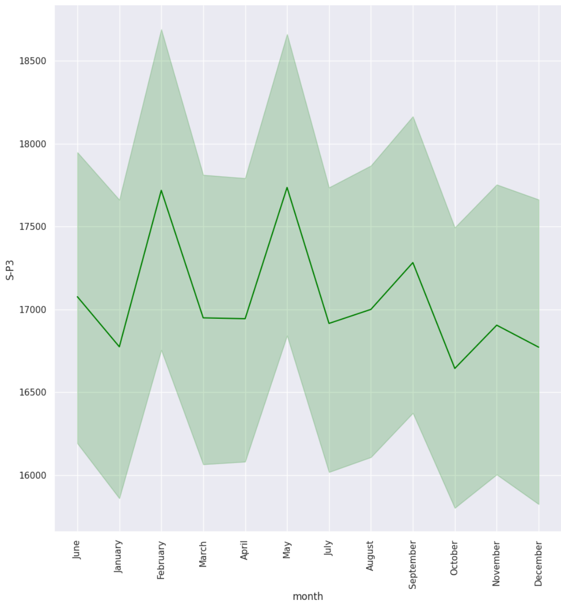
```
sns.relplot(x="month",y="S-P1",data=df,kind="line",height=10,color="red")

plt.xticks(rotation=90);

sns.relplot(x="month",y="S-P2",data=df,kind="line",height=10,color="blue")

plt.xticks(rotation=90);

sns.relplot(x="month",y="S-P3",data=df,kind="line",height=10,color="green")

plt.xticks(rotation=90);
```

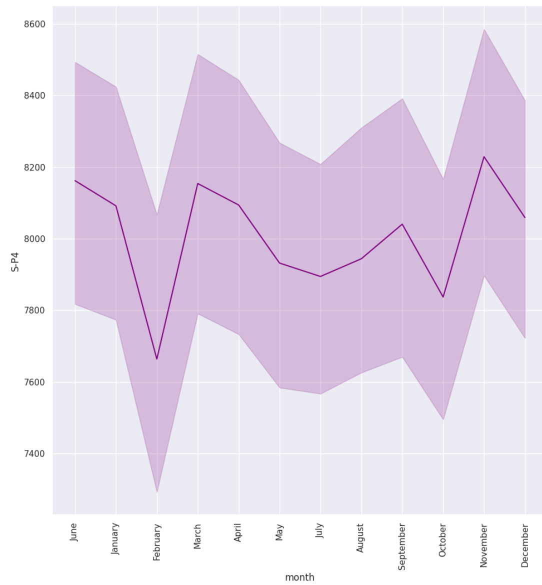
```
sns.relplot(x="month",y="S-P4",data=df,kind="line",height=10,c
olor="purple")

plt.xticks(rotation=90);
```

Out









8. Code

```
df.groupby("month")[["S-P1","S-P2","S-P3","S-P4"]].sum()
```

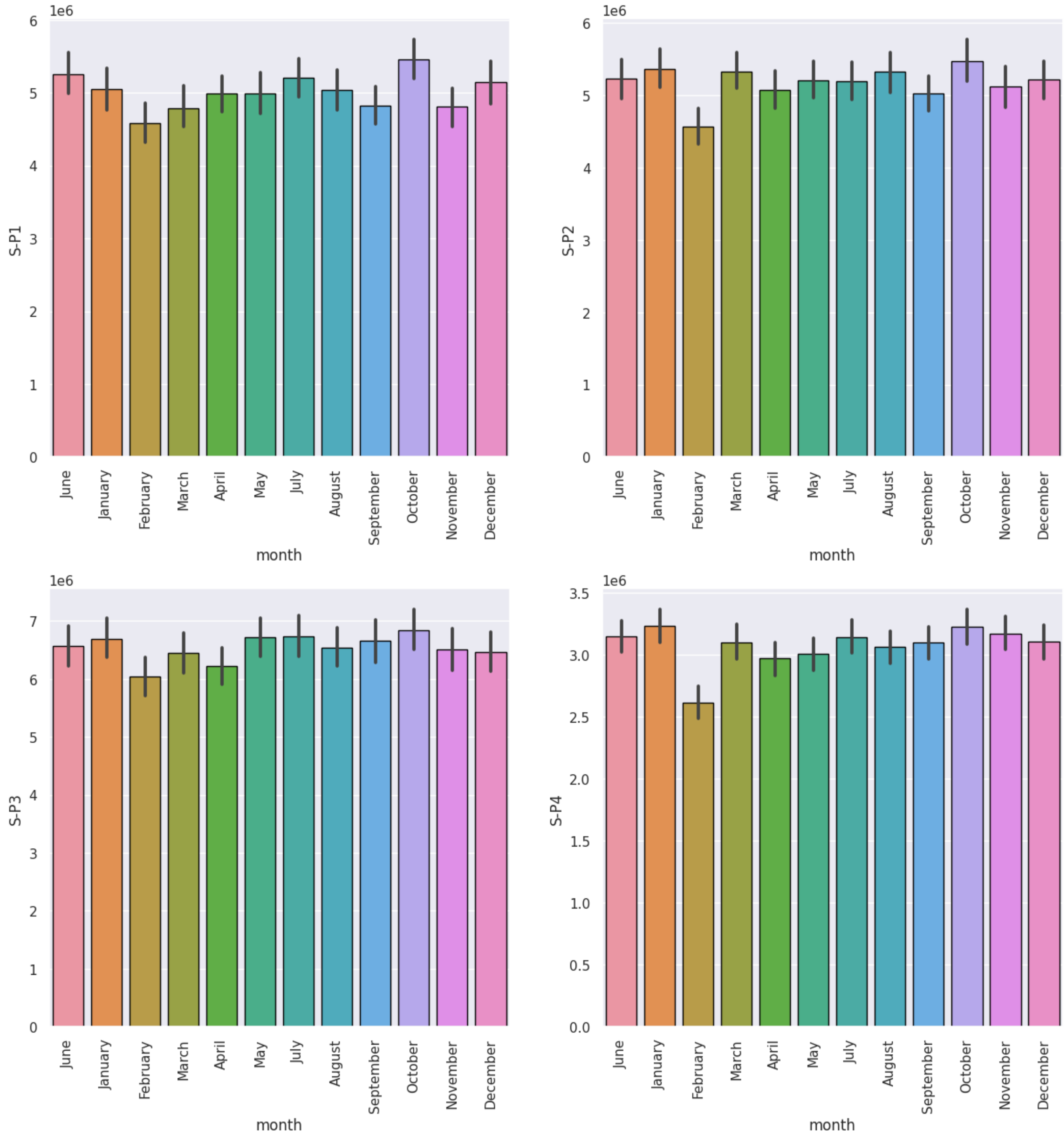
Out

	S-P1	S-P2	S-P3	S-P4
month				
April	4994236.73	5074402.86	6218523.18	2970628.94
August	5032438.40	5327280.10	6545224.52	3058499.06
December	5140424.45	5218441.32	6457398.84	3102797.75
February	4576731.88	4561845.56	6042134.70	2613444.46
January	5048012.61	5360970.86	6693223.04	3228692.16
July	5205647.20	5199104.32	6732490.94	3142091.18
June	5251837.27	5226404.36	6574600.92	3142454.81
March	4786119.89	5332035.10	6440791.96	3098619.57
May	4983870.83	5207752.08	6722008.66	3006278.94
November	4813933.47	5119068.16	6508476.92	3168215.50
October	5454847.24	5472326.62	6840809.64	3221134.36
September	4816704.05	5027898.96	6653906.36	3095696.27

9. Code

```
plt.figure(figsize=(15,15),dpi=100)

plt.subplot(2,2,1)

sns.barplot(x="month",y="S-P1",data=df,edgecolor="black",estimator=sum)

plt.xticks(rotation=90);

plt.subplot(2,2,2)

sns.barplot(x="month",y="S-P2",data=df,edgecolor="black",estimator=sum)

plt.xticks(rotation=90);

plt.subplot(2,2,3)

sns.barplot(x="month",y="S-P3",data=df,edgecolor="black",estimator=sum)

plt.xticks(rotation=90);

plt.subplot(2,2,4)

sns.barplot(x="month",y="S-P4",data=df,edgecolor="black",estimator=sum)

plt.xticks(rotation=90)

plt.subplots_adjust(hspace=0.3);
```

Out


1e6
S-P1
S-P2
S-P3
S-P4
June
January
February
March
April
May
July
August
September
October
November
December
month

10. Code

```
df.groupby ("month")[["Q-P1","Q-P2","Q-P3","Q-P4"]].sum()
```

Out

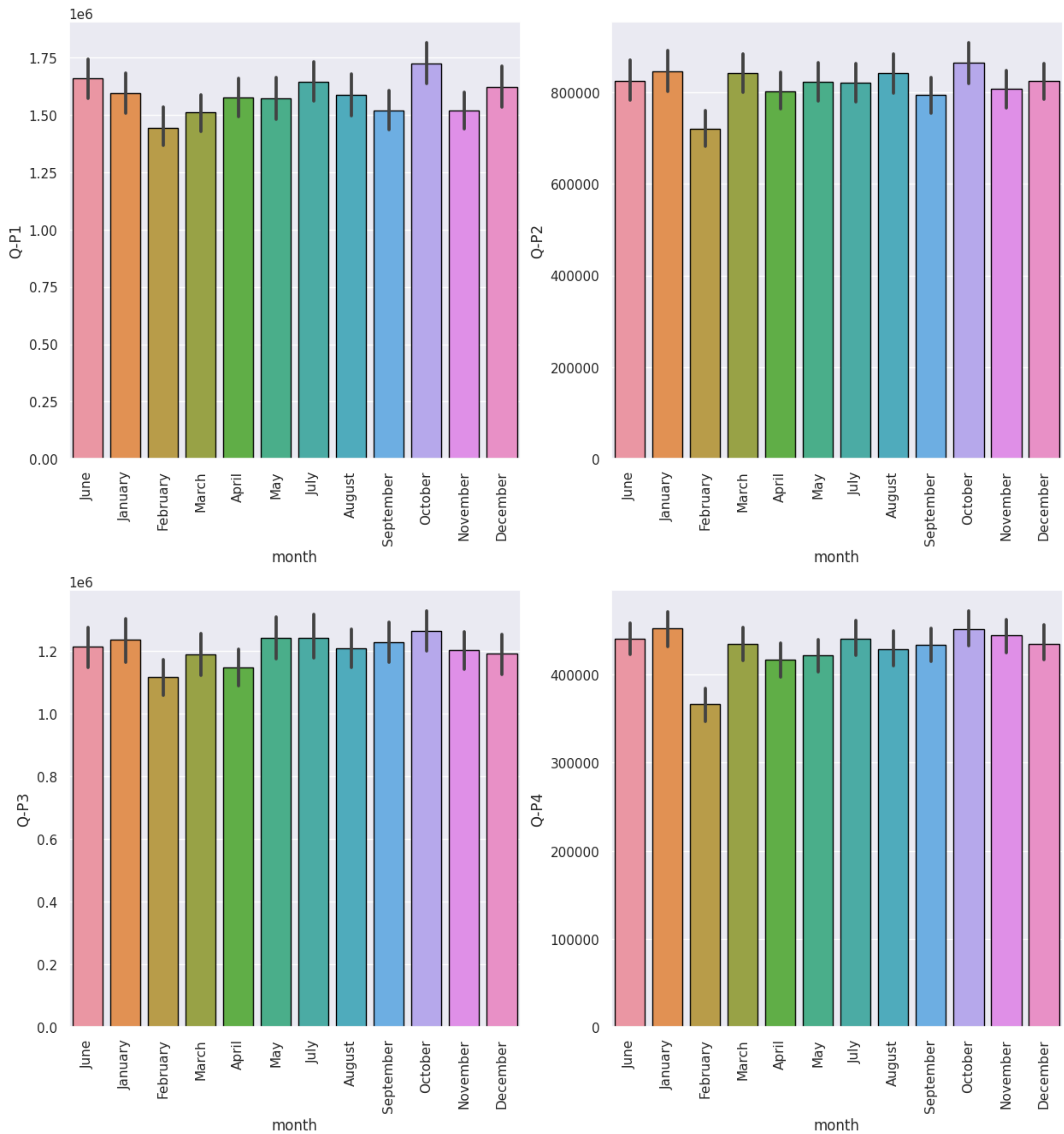
	Q-P1	Q-P2	Q-P3	Q-P4
month				
April	1575469	800379	1147329	416638
August	1587520	840265	1207606	428962
December	1621585	823098	1191402	435175
February	1443764	719534	1114785	366542
January	1592433	845579	1234912	452832
July	1642160	820048	1242157	440686
June	1656731	824354	1213026	440737
March	1509817	841015	1188338	434589
May	1572199	821412	1240223	421638
November	1518591	807424	1200826	444350
October	1720772	863143	1262142	451772
September	1519465	793044	1227658	434179

11. Code

```
plt.figure(figsize=(15,15),dpi=100)

plt.subplot(2,2,1)

sns.barplot(x="month",y="Q-P1",data=df,edgecolor="black",estim
ator=sum)

plt.xticks(rotation=90);

plt.subplot(2,2,2)
```

```
sns.barplot(x="month",y="Q-P2",data=df,edgecolor="black",estimator=sum)

plt.xticks(rotation=90);

plt.subplot(2,2,3)

sns.barplot(x="month",y="Q-P3",data=df,edgecolor="black",estimator=sum)

plt.xticks(rotation=90);

plt.subplot(2,2,4)

sns.barplot(x="month",y="Q-P4",data=df,edgecolor="black",estimator=sum)

plt.xticks(rotation=90)

plt.subplots_adjust(hspace=0.3);
```

Out



1e6
Q-P1
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
June
January
February
March
April
May
July
August
September
October
November
December
month
Q-P2
800000
600000
400000
200000
0
month
1e6
Q-P3
1.2
1.0
0.8
0.6
0.4
0.2
0.0
month
Q-P4
400000
300000
200000
100000
0
month

12. Code

```
week_t=df[df["dayoftheweek"]<5]

weekend_t=df[df["dayoftheweek"]>=5]

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

Out

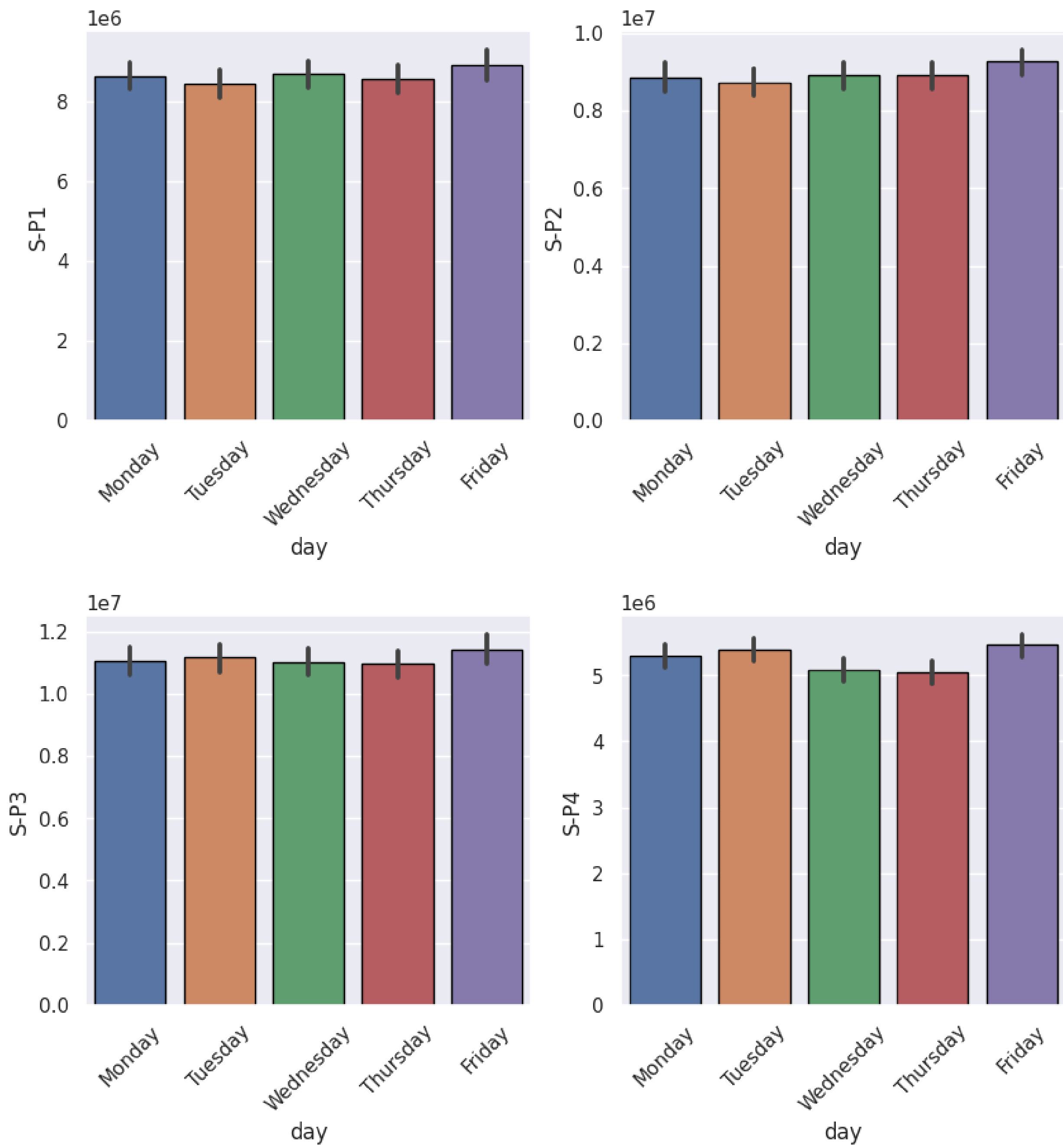
```
                S-P1        S-P2         S-P3        S-P4
day
Friday     8913637.41  9267831.02  11428877.58  5463169.99
Monday     8636791.80  8864347.08  11064892.06  5292577.61
Thursday   8577981.96  8909481.54  10951554.44  5043013.35
Tuesday    8433525.06  8738326.90  11156338.30  5384854.07
Wednesday  8693537.97  8908067.72  11017830.20  5086827.20
```

13. Code

```
plt.figure(figsize=(10,10),dpi=100)

plt.subplot(2,2,1)

sns.barplot(x="day",y="S-P1",data=week_t,edgecolor="black",est
imator=sum)

plt.xticks(rotation=45);

plt.subplot(2,2,2)

sns.barplot(x="day",y="S-P2",data=week_t,edgecolor="black",est
imator=sum)

plt.xticks(rotation=45);

plt.subplot(2,2,3)

sns.barplot(x="day",y="S-P3",data=week_t,edgecolor="black",est
imator=sum)

plt.xticks(rotation=45);

plt.subplot(2,2,4)
```

```
sns.barplot(x="day",y="S-P4",data=week_t,edgecolor="black",estimator=sum)

plt.xticks(rotation=45)

plt.subplots_adjust(hspace=0.5);
```

Out



14. Code

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

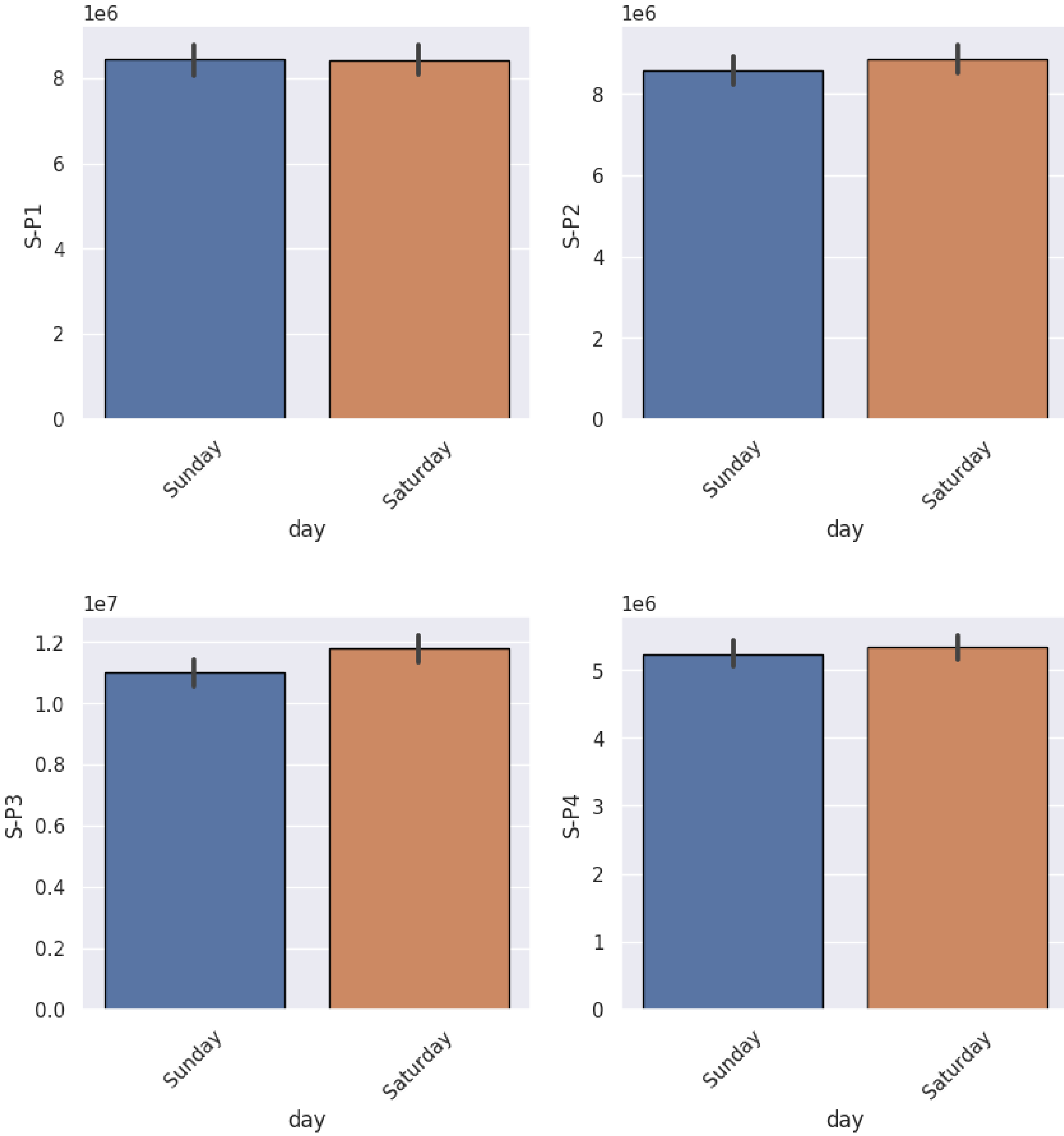
Out

```
                S-P1        S-P2         S-P3        S-P4
day
Saturday  8409578.88  8853201.36  11796375.26  5339977.85
Sunday    8439750.94  8586274.68  11013721.84  5238132.93
```

15. Code

```
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);
```

Out



16. Code

```
df.groupby("year")[["S-P1","S-P2","S-P3","S-P4"]].agg(["sum"])
```

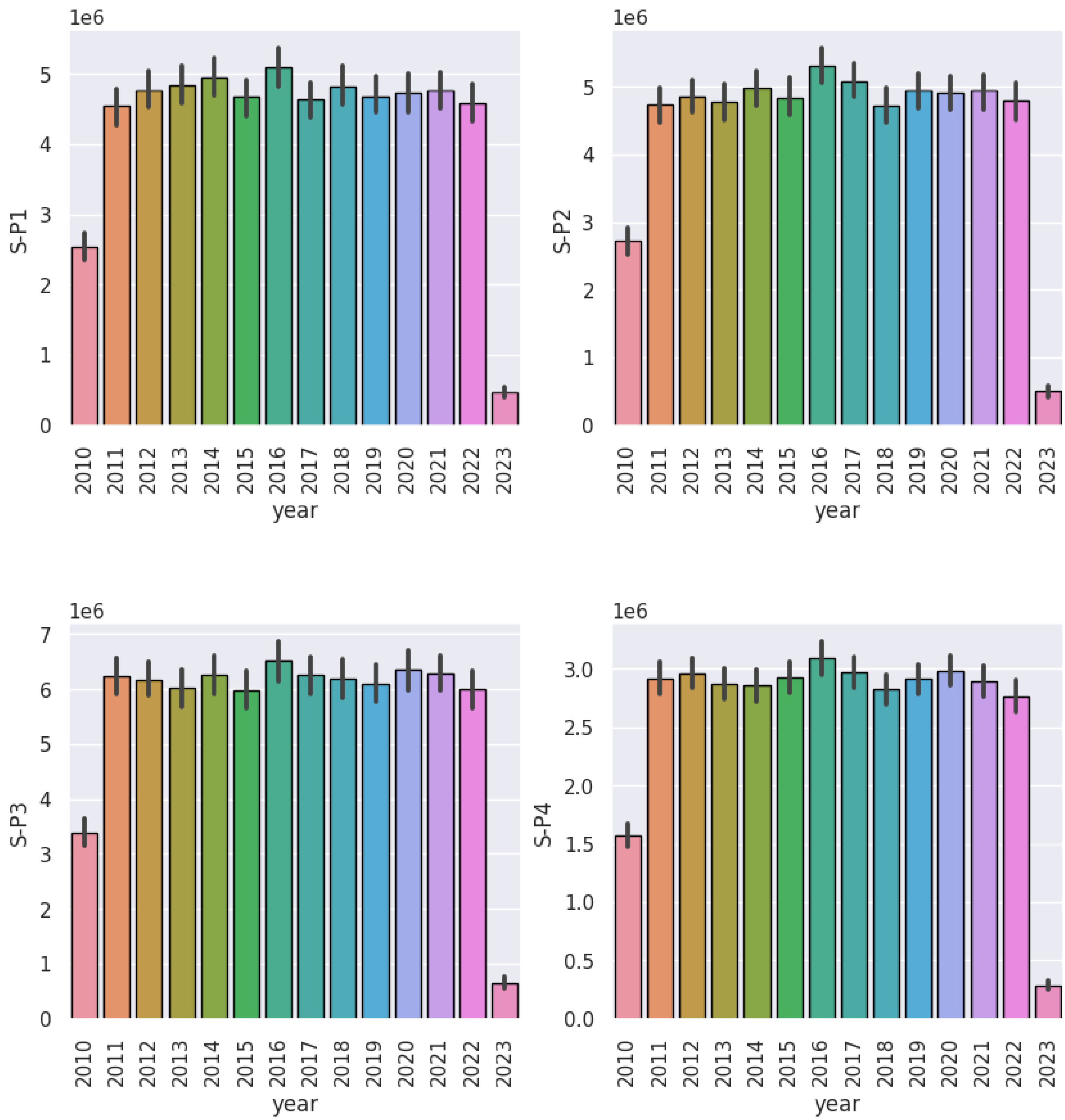
Out

	S-P1	S-P2	S-P3	S-P4
	sum	sum	sum	sum
year				
2010	2543459.01	2720100.92	3385462.08	1567523.37
2011	4542819.22	4741147.10	6235075.86	2921603.06
2012	4771163.83	4861987.50	6173911.16	2965210.14
2013	4833682.57	4771369.88	6017809.74	2868491.69
2014	4954522.97	4979797.38	6265406.18	2865119.20
2015	4669720.66	4833806.20	5987988.90	2933224.96
2016	5096066.64	5313116.54	6507718.12	3096444.92
2017	4628545.53	5085909.96	6269568.74	2969944.46
2018	4825792.44	4727313.22	6198517.96	2824392.64
2019	4681354.56	4946303.16	6106237.04	2912519.44
2020	4732093.58	4904826.88	6343643.88	2984618.00
2021	4758100.26	4948382.68	6294208.06	2894394.98
2022	4591000.05	4797040.54	5993479.36	2760400.89
2023	476482.70	496428.34	650562.60	284665.25

17. Code

```
plt.figure(figsize=(10,10),dpi=100)

plt.subplot(2,2,1)

sns.barplot(x="year",y="S-P1",data=df,edgecolor="black",estima
tor=sum)

plt.xticks(rotation=90);

plt.subplot(2,2,2)

sns.barplot(x="year",y="S-P2",data=df,edgecolor="black",estima
tor=sum)

plt.xticks(rotation=90);

plt.subplot(2,2,3)

sns.barplot(x="year",y="S-P3",data=df,edgecolor="black",estima
tor=sum)

plt.xticks(rotation=90);

plt.subplot(2,2,4)

sns.barplot(x="year",y="S-P4",data=df,edgecolor="black",estima
tor=sum)

plt.xticks(rotation=90)

plt.subplots_adjust(hspace=0.5);
```

Out



18. Code

```
df[["S-P1","S-P2","S-P3","S-P4"]].agg(["sum","max","min","mean
"])
```

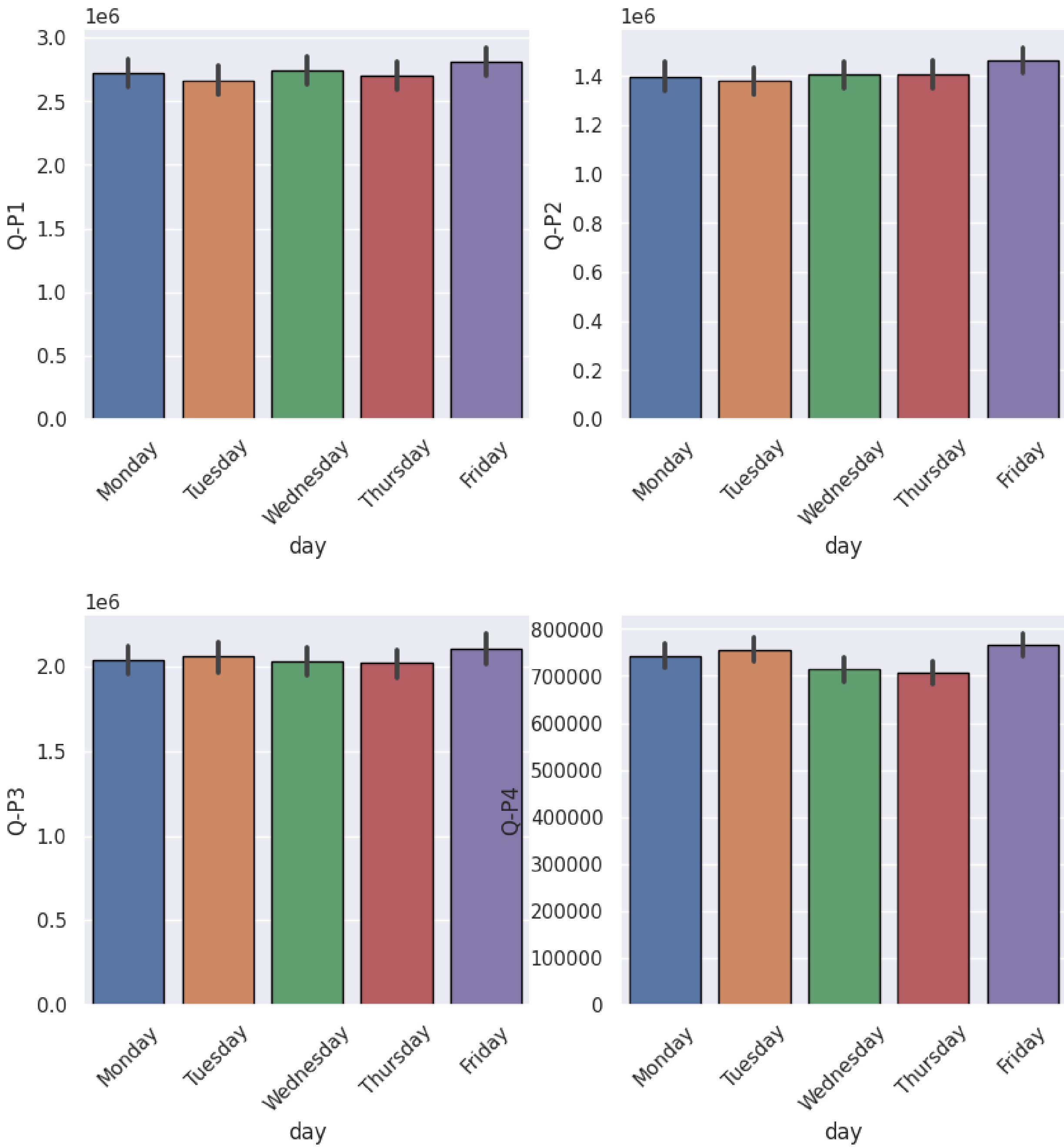
Out

	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

19. Code

```
plt.figure(figsize=(10,10),dpi=100)

plt.subplot(2,2,1)

sns.barplot(x="day",y="Q-P1",data=week_t,edgecolor="black",est
imator=sum)

plt.xticks(rotation=45);

plt.subplot(2,2,2)

sns.barplot(x="day",y="Q-P2",data=week_t,edgecolor="black",est
imator=sum)

plt.xticks(rotation=45);

plt.subplot(2,2,3)

sns.barplot(x="day",y="Q-P3",data=week_t,edgecolor="black",est
imator=sum)

plt.xticks(rotation=45);

plt.subplot(2,2,4)

sns.barplot(x="day",y="Q-P4",data=week_t,edgecolor="black",est
imator=sum)

plt.xticks(rotation=45)

plt.subplots_adjust(hspace=0.5);
```

Out

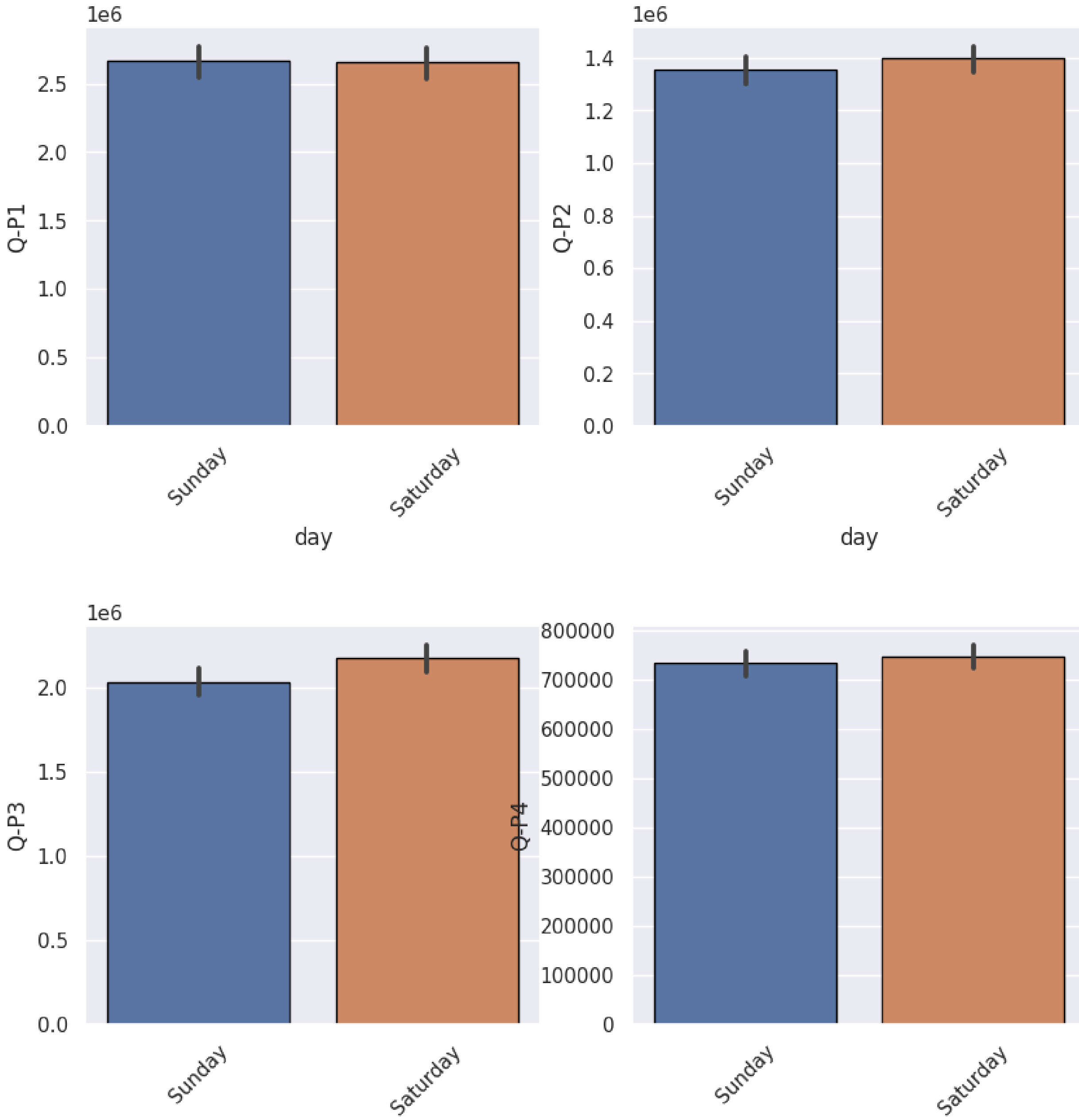


1e6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Q-P1
Monday
Tuesday
Wednesday
Thursday
Friday
day
1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Q-P2
Monday
Tuesday
Wednesday
Thursday
Friday
day
1e6
2.0
1.5
1.0
0.5
0.0
Q-P3
Monday
Tuesday
Wednesday
Thursday
Friday
day
800000
700000
600000
500000
400000
300000
200000
100000
0
Q-P4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

20. Code

```
plt.figure(figsize=(10,10),dpi=100)

plt.subplot(2,2,1)

sns.barplot(x="day",y="Q-P1",data=weekend_t,edgecolor="black",
estimator=sum)

plt.xticks(rotation=45);

plt.subplot(2,2,2)

sns.barplot(x="day",y="Q-P2",data=weekend_t,edgecolor="black",
estimator=sum)

plt.xticks(rotation=45);

plt.subplot(2,2,3)

sns.barplot(x="day",y="Q-P3",data=weekend_t,edgecolor="black",
estimator=sum)

plt.xticks(rotation=45);

plt.subplot(2,2,4)

sns.barplot(x="day",y="Q-P4",data=weekend_t,edgecolor="black",
estimator=sum)

plt.xticks(rotation=45)

plt.subplots_adjust(hspace=0.5);
```

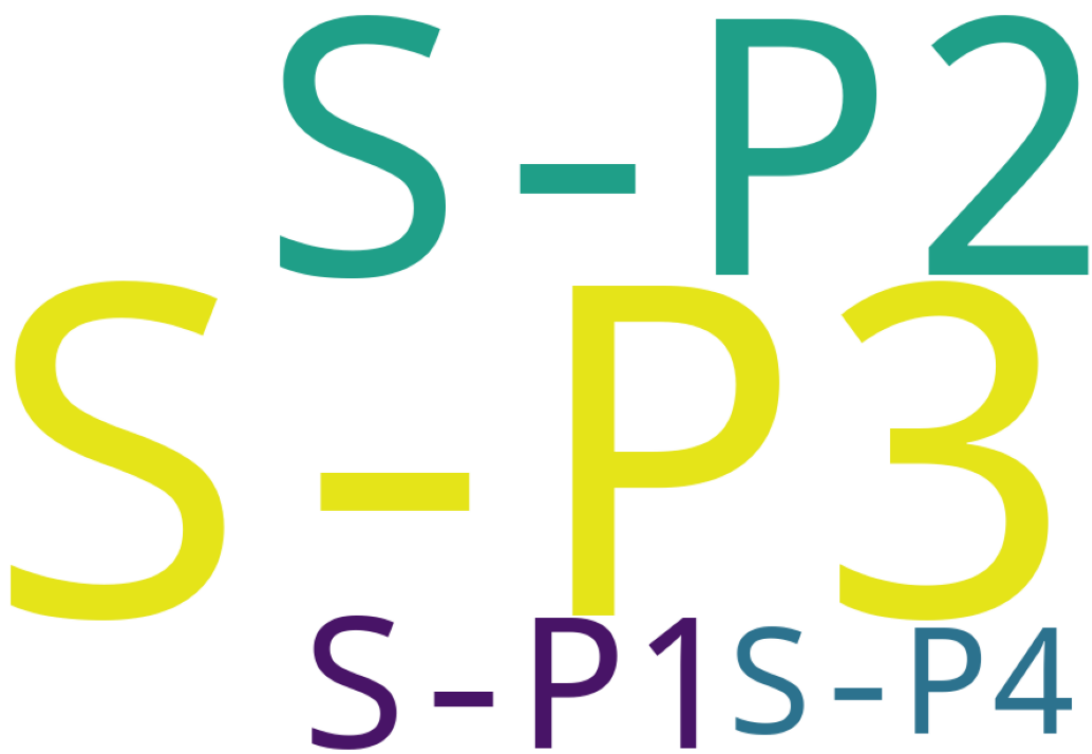
Out


1e6
2.5
2.0
1.5
1.0
0.5
0.0
Q-P1
Sunday
Saturday
day
1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Q-P2
Sunday
Saturday
day
1e6
2.0
1.5
1.0
0.5
0.0
Q-P3
Sunday
Saturday
day
800000
700000
600000
500000
400000
300000
200000
100000
0
Q-P4
Sunday
Saturday
day

21. Code

```
from wordcloud import WordCloud as word
d=df[["S-P1","S-P2","S-P3","S-P4"]].sum()
wc = word(background_color='white', width=1000, height=600)
wc.generate_from_frequencies(d)
plt.figure(figsize=(15,15),dpi=100)
plt.imshow(wc)
plt.axis('off')
plt.show()
```

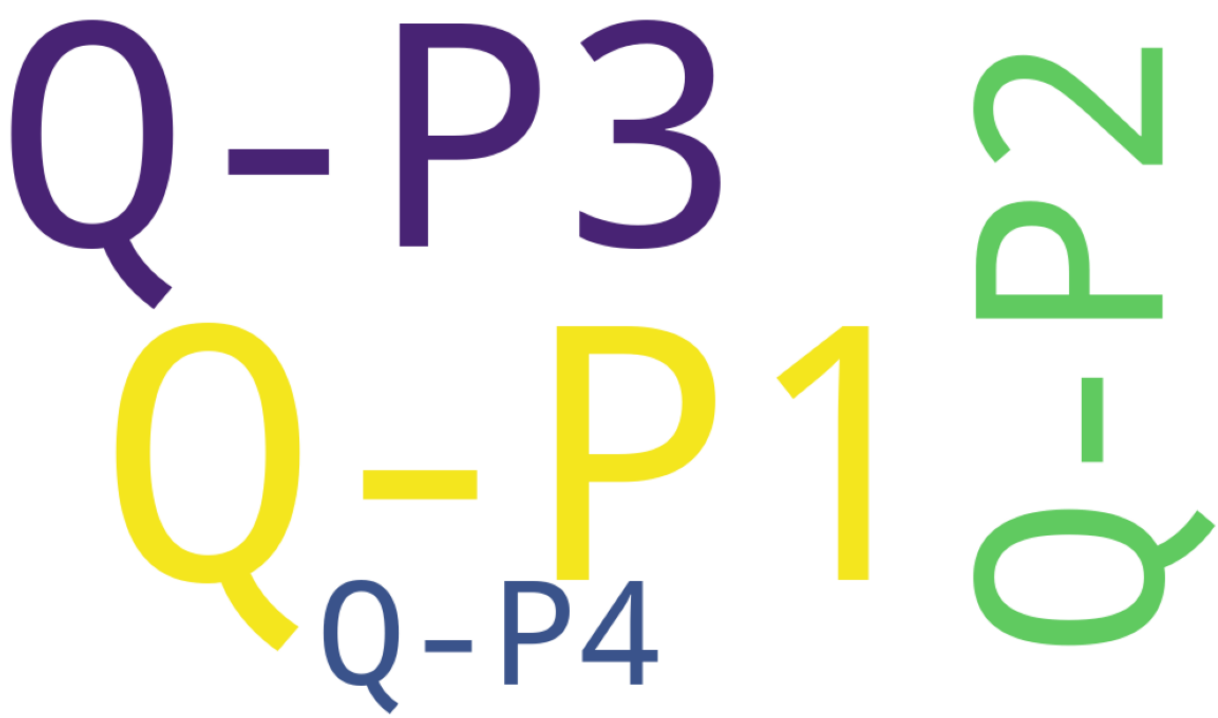
Out



22. Code

```
q=df[["Q-P1","Q-P2","Q-P3","Q-P4"]].sum()

wc = word(background_color='white', width=1000, height=600)

wc.generate_from_frequencies(q)

plt.figure(figsize=(15,15),dpi=100)

plt.imshow(wc)

plt.axis('off')

plt.show()
```

Out



Title: Innovation Phase_4

Introduction

Briefly introduce the purpose of the report and its focus on insights derived from IBM Cognos Analytics.

Top-Selling Products

Present a dashboard highlighting the products with the highest sales.

Include interactive charts and tables for easy exploration.

Sales Trends

Showcase a trend analysis report displaying sales patterns over time.

Identify peak sales periods and provide a clear visualization.

Customer Preferences

Create a dashboard that reveals customer preferences for specific products.

Utilize filters for users to customize their preferences.

Actionable Insights

Summarize key takeaways from the visualizations.

Emphasize the need to focus on top-selling products and peak sales periods.

https://colab.research.google.com/drive/1d3PCu5_NhTyP80NYDCE7BUkgj3mwzrt_?usp=sharing

IBM Cognos Link:

https://us1.ca.analytics.ibm.com/bi/?perspective=dashboard&pathRef=.my_folders%2FProduct%2Bsales%2BAnalysis%2BDashboard&action=view&mode=dashboard&subView=model0000018b660aecda_00000000