COURSE: DATA ANALYTICS WITH COGNOS

PHASE V : PROJECT SUBMISSION

TITLE : PRODUCT SALES ANALYSIS

TEAM MEMBERS:

- ABISHEK KUMAR S
  810721243004
  abishekkumar.s@care.ac.in

- ADEEB M
  810721243005
  adeeb.m@care.ac.in

- DINESH BABU J
  810721243018
  dineshbabu@care.
  ac.in

- JEEVA RAJ K
  810721243025
  jeevaraj.k@care.ac.in

- RAHUL M
  810721243042
  rahul.m@care.ac.in

TABLE OF CONTENT

Project Title: Product Sales Analysis

OBJECTIVES

Problem Definition:

The 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.

Design Thinking:

1. Analysis Objectives:

- Identifying Top-Selling Products:
  - Determine the top-selling products based on total sales revenue or units sold.
  - Identify any seasonal variations in the sales of these products.
  - Discover the geographical regions or customer segments where these products perform exceptionally well.

- Analyzing sales trends:
  - Identify overall sales trends over time (e.g., monthly, quarterly, or annually).
  - Detect any sudden spikes or dips in sales and investigate their causes.
  - Determine if there are specific products or product categories that exhibit consistent growth or decline in sales.

- Understanding customer preferences:
  - Segment customers based on demographics (age, gender, location), purchase history, and behavior (e.g., frequent shoppers, one-time buyers).

- Identify the most preferred products or product categories for each customer segment.
- Analyze how customer preferences change over time and in response to marketing campaigns.

2. Data Collection:

- Sales Data:
  - **Point of Sale (POS) System:** Collect transaction-level data from the point-of-sale system if available. This should include details such as product IDs, transaction dates and times, quantities sold, and prices.
  - **Historical Sales Records:** Gather historical sales records covering an extended period to identify long-term trends and seasonality.
  - **Sales Channels:** Include data from all sales channels, whether it's in-store, online, or through third-party retailers.
- Product Data:
  - **Product Catalog**: Obtain a comprehensive product catalog that includes product names, descriptions, categories, and attributes.
  - **SKU Information:** If applicable, collect data on stock-keeping units (SKUs) for each product, which can help in tracking inventory at a granular level.
  - **Pricing Information:** Include pricing data for each product, including regular prices, discounts, and any special pricing structures.
- customer demographics:
  - For data collection in this project, customer demographics encompass a comprehensive set of information about the individuals who engage in transactions.
  - This includes but is not limited to age, gender, location, income level, occupation, marital status, educational background, and household size. Gathering data on these demographic attributes allows for segmentation and analysis of the customer base, aiding in the identification of target customer groups, understanding their preferences, and tailoring marketing strategies.
  - Moreover, it provides insights into the potential impact of demographic factors on purchasing behavior and sales trends, facilitating data-driven decisions for inventory management and marketing efforts.

- Transaction records:
    - The transaction record for the data collection in this project comprises a set of crucial details pertaining to a sales transaction. Each record is uniquely identified by a Transaction ID and includes essential information such as the Transaction Date and Time, Customer ID, Customer Name, Payment Method, and the Total Amount of the purchase.
    - It also encompasses a comprehensive list of the Products Purchased, with each product identified by a unique Product ID, Product Name, Quantity, Unit Price, and Subtotal.
    - Additionally, the record features both the Shipping Address and Billing Address, offering insights into customer locations and payment details, and specifies the Sales Channel through which the transaction was conducted, providing context for the sales channel's performance and effectiveness in driving revenue.

3. Visualization Strategy:
    - To visualize the insights using IBM Cognos and create interactive dashboards and reports, we will follow a structured plan. First, we will import the cleaned and prepared sales data into IBM Cognos. Next, we'll design a set of interactive dashboards that cater to various user needs, incorporating dynamic filters and drill-down options for in-depth exploration.
    - Key insights, such as top-selling products, sales trends, and customer preferences, will be presented using appropriate chart types, including bar charts, line graphs, pie charts, and heatmaps. We'll ensure that visualizations are clear, labeled, and use a consistent color scheme for readability.
    - Additionally, we will integrate geospatial data if applicable, providing regional insights on a map. For time series analysis, time-based visualizations will be created to capture trends over different periods. Throughout the process, we'll emphasize responsiveness and accessibility, ensuring that the dashboards and reports work well on various devices and are accessible to all users.
    - Finally, thorough documentation and training materials will be provided to support users in effectively utilizing the interactive dashboards and reports to make data-driven decisions for optimizing inventory management and marketing strategies.

4. Actionable Insights:
    - The derived insights from the analysis of sales data in IBM Cognos provide actionable guidance for optimizing both inventory management and marketing strategies.
    - By identifying top-selling products and understanding customer preferences, businesses can prioritize stock levels and tailor marketing campaigns to promote these high-demand items. Analysis of sales trends and peak sales periods informs inventory stocking strategies to ensure adequate supply during surges in demand, reducing stockouts and improving customer satisfaction.
    - Insights into the effectiveness of marketing campaigns and channels help allocate resources more efficiently, targeting the right audience with the right promotions.
    - Additionally, by forecasting future sales trends, businesses can make data-driven decisions on inventory replenishment and marketing planning.
    - Overall, these insights enable businesses to reduce carrying costs, minimize lost sales opportunities, enhance customer experiences, and maximize ROI on marketing investments.

INNOVATION:

1. Data Collection and Preparation:

Gather historical sales data, which should include information on sales volume, date/time, product details, pricing, marketing campaigns, and any other relevant factors. Clean and preprocess the data to handle missing values, outliers, and inconsistencies. Ensure that the data is in a format suitable for machine learning

2. Feature Engineering:

Create relevant features from the data that can help your machine learning model make accurate predictions. These features may include lagged sales, seasonality, holidays, and economic indicators.

3. Data Splitting:

Split your data into training, validation, and test sets. The training set is used to train the model, the validation set helps tune hyperparameters, and the test set evaluates the model's performance.

4. Selecting Machine Learning Algorithms:

Choose machine learning algorithms suitable for time-series forecasting. Common choices include:

- Linear Regression: Simple and interpretable, but may not capture complex patterns.
- Decision Trees and Random Forests: Effective for capturing non- linear relationships and feature importance.
- ARIMA (Auto Regressive Integrated Moving Average): A traditional time-series forecasting method.
- Prophet: Developed by Facebook, designed for forecasting with seasonality and holidays.

5. Model Training:

Train your selected models using the training data. Experiment with different algorithms and hyperparameters to find the best-performing model.

6. Model Evaluation:

Use the validation set to assess the model's performance. Common evaluation metrics for time-series forecasting include Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

7. Monitoring and Maintenance:

Continuously monitor the model's performance in the production environment. Update the model as needed to adapt to changing sales trends

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

VISUALIZATION STRATEGY:

Python:

Matplotlib: A versatile 2D plotting library that offers a wide range of chart types and customization options.

Seaborn: Built on top of Matplotlib, Seaborn provides a high- level interface for creating attractive statistical visualizations.

Pandas Plotting: Pandas, a data manipulation library, offers built-in plotting capabilities that are convenient for quickly visualizing data.

Plotly: An interactive plotting library that supports a wide range of chart types, including interactive web-based visualizations.

About Dataset:

Greetings, fellow analyst ! REC corp LTD. is small-scaled 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

Import **Libraries**:

```
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")
```

Import Data:

```
In [3]: df = pd.read_csv('Z:\PSA.csv')
        df.head()
```

Out[3]:

	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

Workflow:

- Understanding the data
- Data cleaning
- Exploratory Data Analysis
- Insights

Understanding the data

```
In [4]: # Fethcing rows and columns
        df.shape
```

Out[4]: (4600, 10)

```
In [5]: # fetching column names
        df.columns
```

Out[5]:
```
Index(['Unnamed: 0', 'Date', 'Q-P1', 'Q-P2', 'Q-P3', 'Q-P4', 'S-P1', 'S-P2',
       'S-P3', 'S-P4'],
      dtype='object')
```

In [6]:
```
# basic info
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  
 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: 341.5+ KB
```

In [7]:
```
# Checking null values
df.isnull().sum()
```

Out[7]:
```
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
```

In [8]:
```
# Checking Dtypes
df.dtypes
```

Out[8]:
```
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
```

In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
## Basic statistical info
df.describe().T
```

Out[10]:

	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

Data Cleaning

In [11]:
```
df.sample(2)
```

Out[11]:

	Unnamed: 0	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4
918	918	21-12-2012	4304	3211	3210	1031	13643.68	20357.74	17398.20	7351.03
1160	1160	21-08-2013	4323	2593	1663	717	13703.91	16439.62	9013.46	5112.21

In [12]:
```
# Changing dtype
from datetime import datetime as dt
df[df["Date"]=="31-9-2010"]
```

Out[12]:

	Unnamed: 0	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4
109	109	31-9-2010	4986	342	4978	558	15805.62	2168.28	26980.76	3978.54

In [13]:
```
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df[df['Date'].isnull()]
```

Out[13]:

	Unnamed: 0	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4
109	109	NaT	4986	342	4978	558	15805.62	2168.28	26980.76	3978.54
170	170	NaT	4632	3930	523	1581	14683.44	24916.20	2834.66	11272.53
473	473	NaT	2242	401	5926	789	7107.14	2542.34	32118.92	5625.57
534	534	NaT	325	3476	4588	1771	1030.25	22037.84	24866.96	12627.23
836	836	NaT	1003	256	1346	1449	3179.51	1623.04	7295.32	10331.37
897	897	NaT	2509	2666	4146	593	7953.53	16902.44	22471.32	4228.09
1200	1200	NaT	597	709	5470	1994	1892.49	4495.06	29647.40	14217.22
1261	1261	NaT	7681	1235	347	1087	24348.77	7829.90	1880.74	7750.31
1564	1564	NaT	5333	833	3494	618	16905.61	5281.22	18937.48	4406.34
1625	1625	NaT	3870	2779	3246	1290	12267.90	17618.86	17593.32	9197.70
1928	1928	NaT	3583	2111	4225	1401	11358.11	13383.74	22899.50	9989.13
1989	1989	NaT	7516	3423	3116	458	23825.72	21701.82	16888.72	3265.54
2291	2291	NaT	7891	741	2280	1068	25014.47	4697.94	12357.60	7614.84
2352	2352	NaT	2457	3144	533	1184	7788.69	19932.96	2888.86	8441.92
2655	2655	NaT	3512	2851	4072	1597	11133.04	18075.34	22070.24	11386.61
2716	2716	NaT	6094	3798	5849	881	19317.98	24079.32	31701.58	6281.53
3019	3019	NaT	1727	2645	5715	1295	5474.59	16769.30	30975.30	9233.35
3080	3080	NaT	7360	2974	2717	1127	23331.20	18855.16	14726.14	8035.51
3383	3383	NaT	3195	2525	5918	1003	10128.15	16008.50	32075.56	7151.39
3444	3444	NaT	2660	2674	2732	934	8432.20	16953.16	14807.44	6659.42
3746	3746	NaT	4713	1227	4065	403	14940.21	7779.18	22032.30	2873.39
3807	3807	NaT	870	3463	798	851	2757.90	21955.42	4325.16	6067.63
4110	4110	NaT	3511	2609	1543	853	11129.87	16541.06	8363.06	6081.89
4171	4171	NaT	506	3333	3897	574	1604.02	21131.22	21121.74	4092.62
4474	4474	NaT	6964	1873	5481	1336	22075.88	11874.82	29707.02	9525.68
4535	4535	NaT	4600	2006	3796	1426	14582.00	12718.04	20574.32	10167.38

In [14]:
```
## Filling the NaT  values with average of time
df["Date"].fillna(df["Date"].mean(),inplace=True)
df['Date'].isnull().sum()
```

Out[14]:
```
0
```

In [17]:
```
#fetching month,day of week, weekday
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()
```

Out[17]:

	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4	month	day	dayoftheweek	year
2015	2015-12-26	6685	3320	4771	526	21191.45	21048.8	25858.82	3750.38	December	Saturday	5	2015

In [16]:
```
## Droping column unnamed as it is not usefull for us
df.drop(columns=["Unnamed: 0"],inplace=True)
df.sample()
```

Out[16]:

	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4
2010	2015-12-21	4549	3393	1757	1351	14420.33	21511.62	9522.94	9632.63

In [18]:
```
df.corr().T
```

Out[18]:

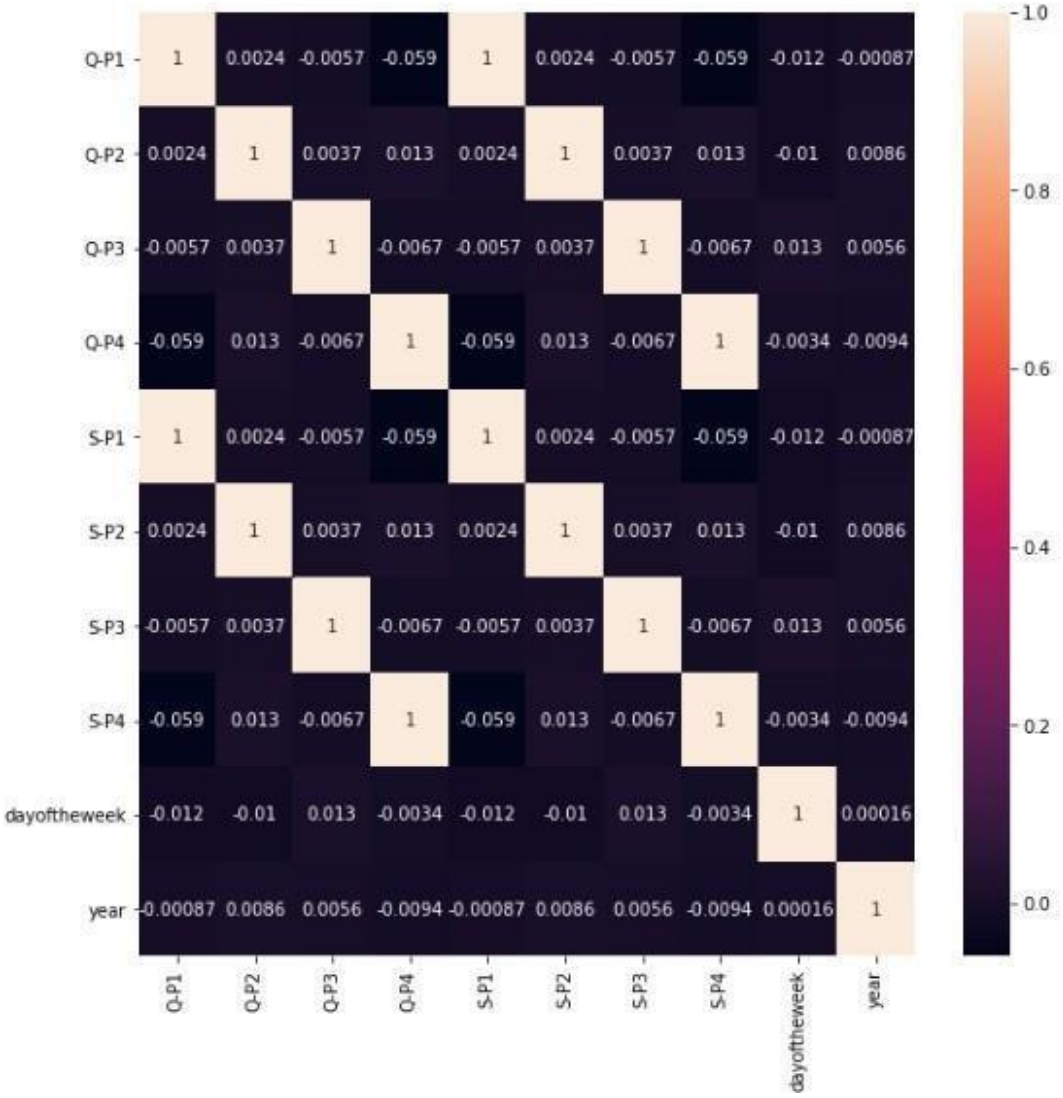
	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4	dayoftheweek	year
Q-P1	1.000000	0.002422	-0.005650	-0.059365	1.000000	0.002422	-0.005650	-0.059365	-0.012221	-0.000866
Q-P2	0.002422	1.000000	0.003729	0.013082	0.002422	1.000000	0.003729	0.013082	-0.010037	0.008556
Q-P3	-0.005650	0.003729	1.000000	-0.006693	-0.005650	0.003729	1.000000	-0.006693	0.012546	0.005632
Q-P4	-0.059365	0.013082	-0.006693	1.000000	-0.059365	0.013082	-0.006693	1.000000	-0.003351	-0.009436
S-P1	1.000000	0.002422	-0.005650	-0.059365	1.000000	0.002422	-0.005650	-0.059365	-0.012221	-0.000866
S-P2	0.002422	1.000000	0.003729	0.013082	0.002422	1.000000	0.003729	0.013082	-0.010037	0.008556
S-P3	-0.005650	0.003729	1.000000	-0.006693	-0.005650	0.003729	1.000000	-0.006693	0.012546	0.005632
S-P4	-0.059365	0.013082	-0.006693	1.000000	-0.059365	0.013082	-0.006693	1.000000	-0.003351	-0.009436
dayoftheweek	-0.012221	-0.010037	0.012546	-0.003351	-0.012221	-0.010037	0.012546	-0.003351	1.000000	0.000159
year	-0.000866	0.008556	0.005632	-0.009436	-0.000866	0.008556	0.005632	-0.009436	0.000159	1.000000

In [20]:
```
for i in df.columns:
    print(i,"---------",df[i].unique())
```

```
Date --------- ['2010-06-13T00:00:00.000000000' '2010-06-14T00:00:00.000000000'
 '2010-06-15T00:00:00.000000000' ... '2023-01-02T00:00:00.000000000'
 '2023-02-02T00:00:00.000000000' '2023-03-02T00:00:00.000000000']
Q-P1 --------- [5422 7047 1572 ... 1227 3122 1234]
Q-P2 --------- [3725  779 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  8206.63  7928.56]
month --------- ['June' 'January' 'February' 'March' 'April' 'May' 'July' 'August'
 'September' 'October' 'November' 'December']
day --------- ['Sunday' 'Monday' 'Tuesday' 'Wednesday' 'Thursday' 'Friday' 'Saturday']
dayoftheweek --------- [6 0 1 2 3 4 5]
year --------- [2010 2016 2011 2012 2013 2014 2015 2017 2018 2019 2020 2021 2022 2023]
```

```
In [19]: plt.figure(figsize=(10,10))
         sns.heatmap(df.corr(),annot=True)
```

```
Out[19]: <AxesSubplot:>
```



```
In [3]: df = pd.read_csv('Z:\PSA.csv')
        df.head()
```

Out[3]:

	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

Exploratory Data Analysis

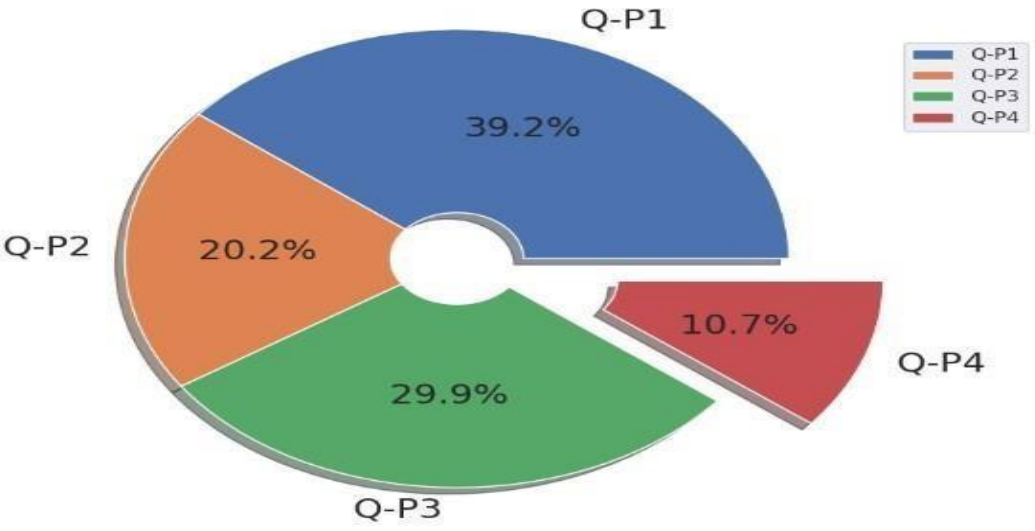
```
In [23]: df.sample()
```

Out[23]:

	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4	month	day	dayoftheweek	year
2659	2017-04-10	2964	3484	4267	1140	9395.88	22088.56	23127.14	8128.2	April	Monday	0	2017

```
In [24]: # Total unit sales Product 1, Product 2, Product 3, Product 4
         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));
```

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



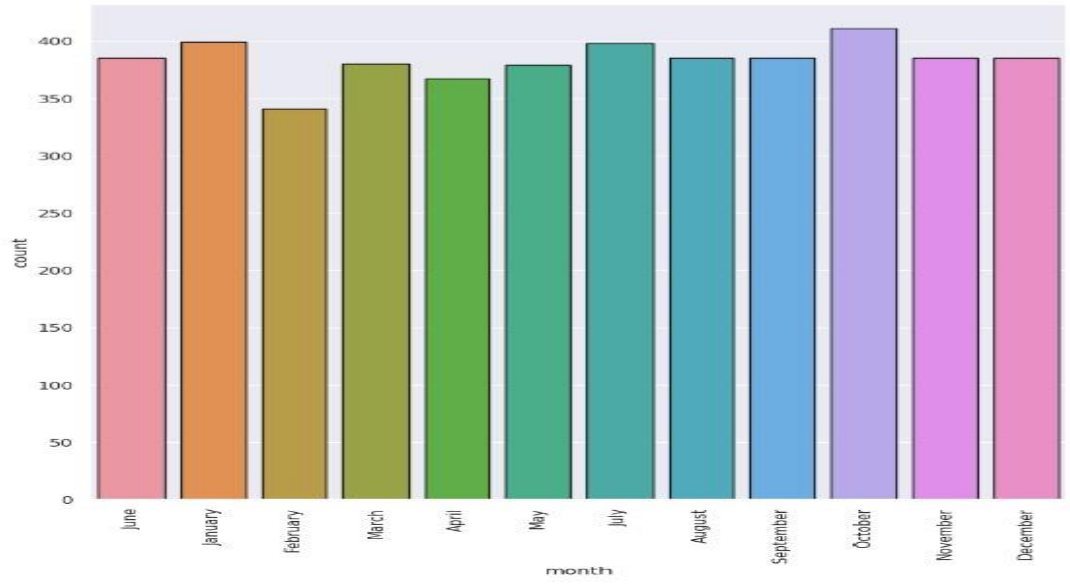
In [25]:
```
# Total Revenue percent from sales from Product 1, Product 2, Product 3, Product
s=df[["S-P1","S-P2","S-P3","S-P4"]].sum()
print(s)
plt.figure(figsize=(8,8))
plt.pie(s,labels=df[["S-P1","S-P2","S-P3","S-P4"]].sum().index,
        shadow=True,autopct="%0.01f%%",textprops={"fontsize":20},
        wedgeprops={'width': 0.8},explode=[0,0,0,0.3])
plt.legend(loc='center right', bbox_to_anchor=(1.2, 0.8))
```

```
S-P1     60104804.02
S-P2     62127530.30
S-P3     78429589.68
S-P4     36848553.00
dtype: float64
```

Out[25]: `<matplotlib.legend.Legend at 0x7fb0c914e090>`

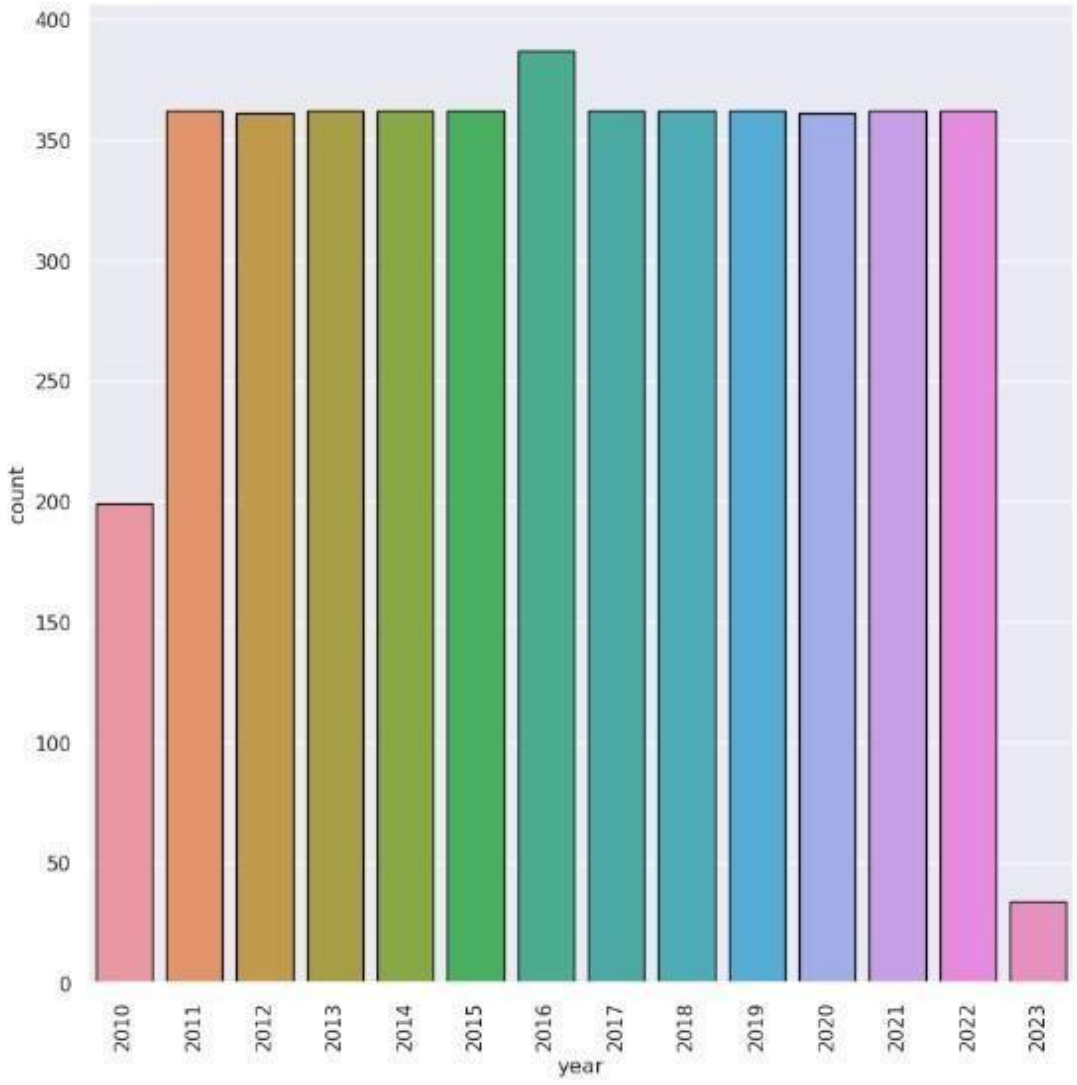
In [26]:
```
# which is the most occuring month
print(df["month"].value_counts())
plt.figure(figsize=(10,10))
sns.countplot(x="month",data=df,edgecolor="black")
plt.xticks(rotation=90);
```

```
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
```



In [28]:
```
# which is the most occuring year
print(df["year"].value_counts())
plt.figure(figsize=(10,10))
sns.countplot(x="year",data=df,edgecolor="black")
plt.xticks(rotation=90);
```

```
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
```

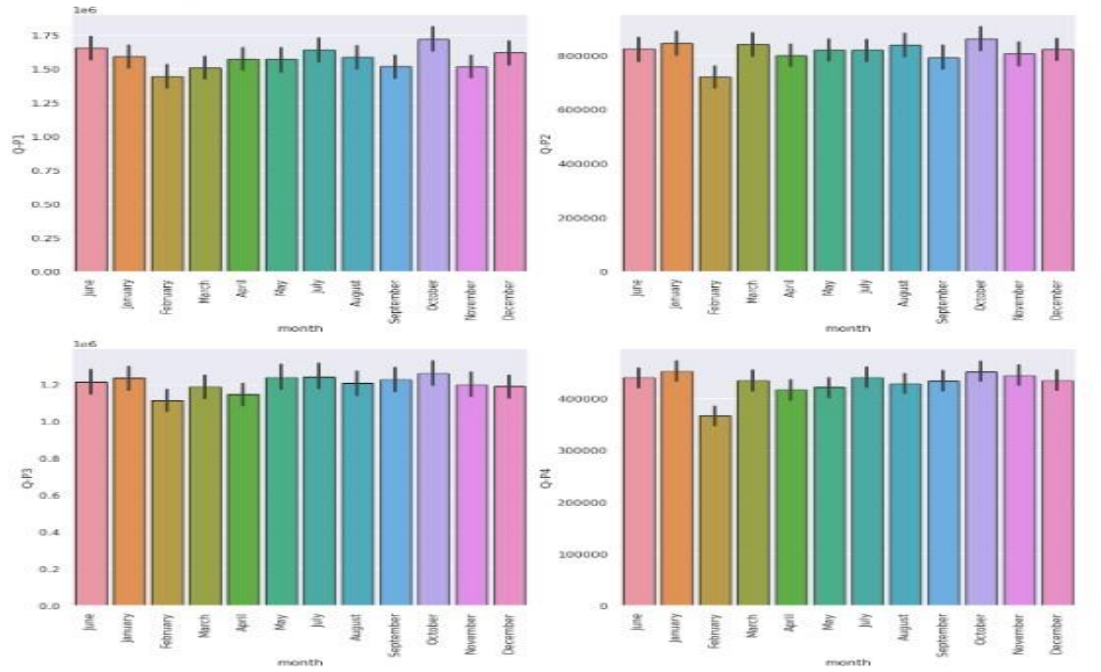


In [30]:
```
## In which month revenue was it peak
df.groupby("month")[["S-P1","S-P2","S-P3","S-P4"]].sum()
```

Out[30]:

month	S-P1	S-P2	S-P3	S-P4
April	4994236.73	5074402.86	6218623.18	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	5206647.20	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	6608476.92	3168215.50
October	5454847.24	5472326.82	6840809.84	3221134.36
September	4816704.05	5027898.96	6653906.36	3095696.27

In [34]:
```
plt.figure(figsize=(15,15),dpi=100)
plt.subplot(2,2,1)
sns.barplot(x="month",y="Q-P1",data=df,edgecolor="black",estimator=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);
```



In [32]:
```
df.sample()
```

Out[32]:

	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4	month	day	dayoftheweek	year
3525	2020-02-20	1270	889	885	274	4025.9	5636.26	4796.7	1953.62	February	Thursday	3	2020

In [33]:
```
## In which month unit sales were more in Product 1, Product 2, Product 3, Product 4
df.groupby ("month")[["Q-P1","Q-P2","Q-P3","Q-P4"]].sum()
```

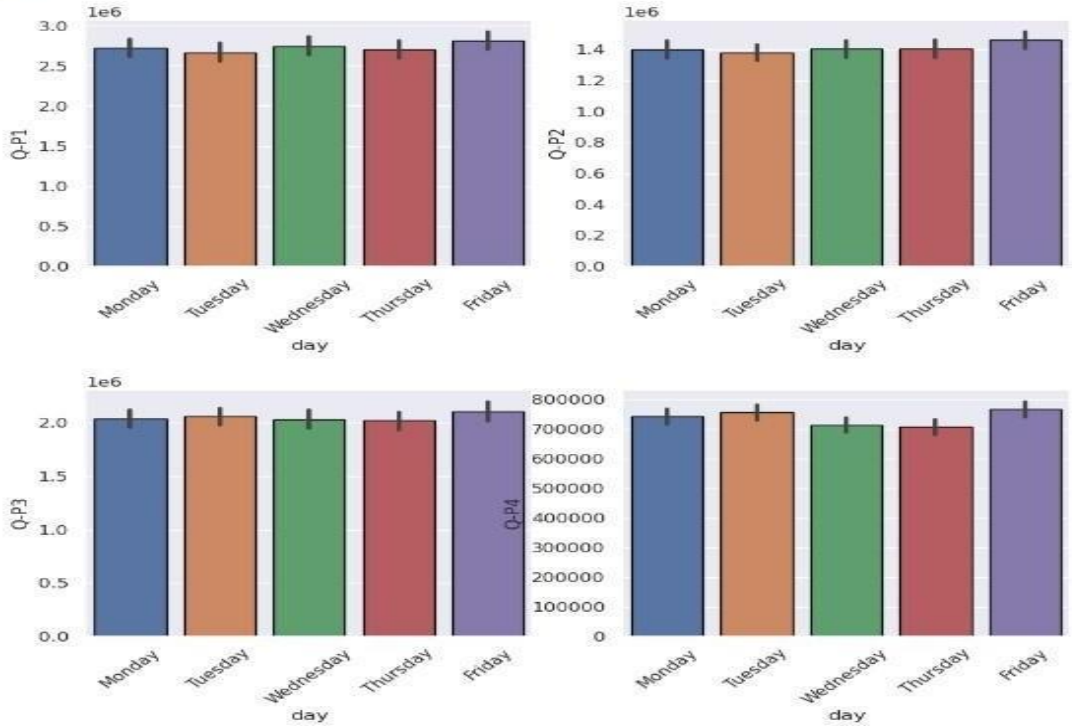
Out[33]:

month	Q-P1	Q-P2	Q-P3	Q-P4
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

In [35]:
```
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())
```

```
                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
```

In [42]:
```
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);
```

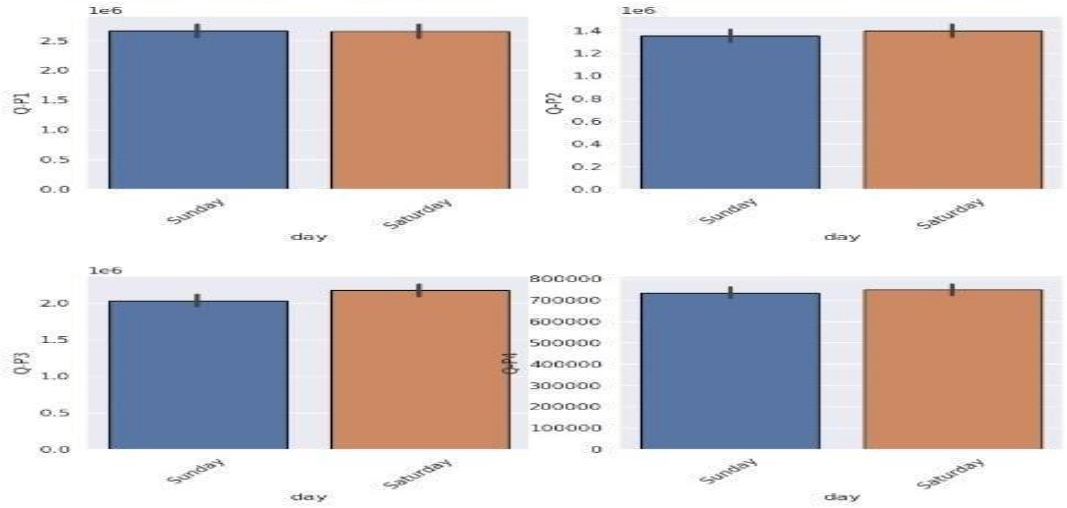


In [39]:
```
# In which year revenue was the highest
df.groupby("year")[["S-P1","S-P2","S-P3","S-P4"]].agg(["sum"])
```

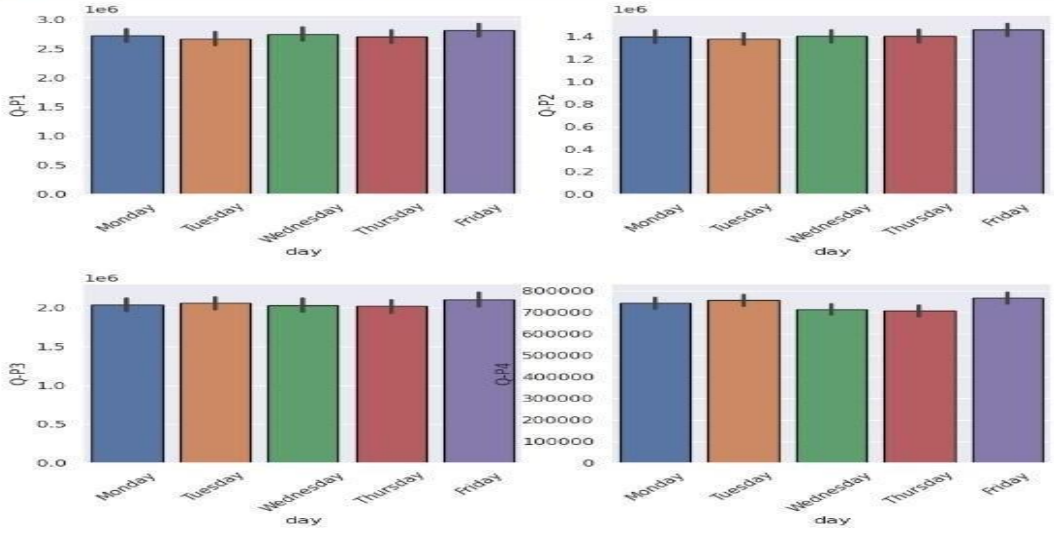
Out[39]:

	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

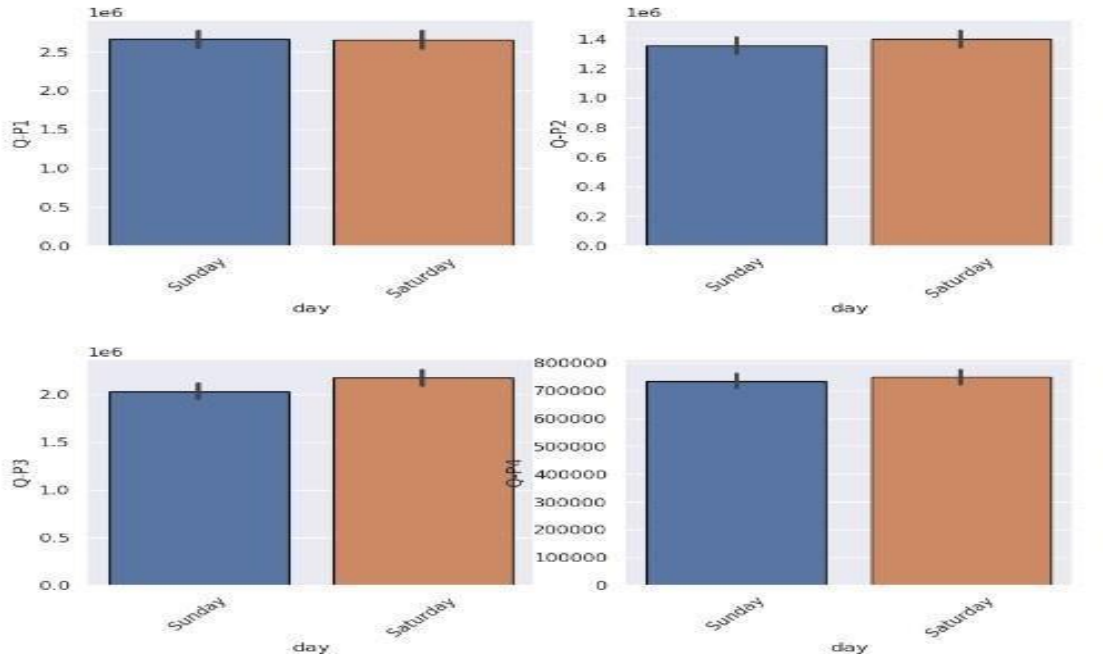
In [43]:
```
plt.figure(figsize=(10,10),dpi=100)
plt.subplot(2,2,1)
sns.barplot(x="day",y="Q-P1",data=weekend_t,edgecolor="black",estimator=sum)
plt.xticks(rotation=45);
plt.subplot(2,2,2)
sns.barplot(x="day",y="Q-P2",data=weekend_t,edgecolor="black",estimator=sum)
plt.xticks(rotation=45);
plt.subplot(2,2,3)
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);
```



In [42]:
```
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);
```

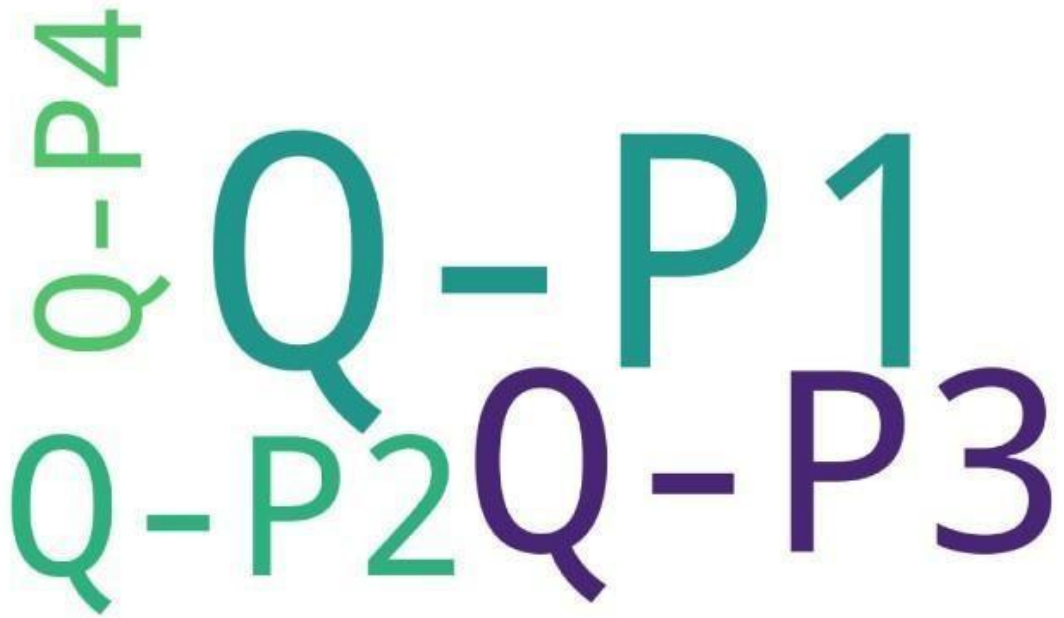


In [43]:
```
plt.figure(figsize=(10,10),dpi=100)
plt.subplot(2,2,1)
sns.barplot(x="day",y="Q-P1",data=weekend_t,edgecolor="black",estimator=sum)
plt.xticks(rotation=45);
plt.subplot(2,2,2)
sns.barplot(x="day",y="Q-P2",data=weekend_t,edgecolor="black",estimator=sum)
plt.xticks(rotation=45);
plt.subplot(2,2,3)
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);
```



In [44]:
```
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()
```

```
In [45]: 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()
```



INSIGHTS:

- Added columns month, day and day of the week and changing the type of date from object to datetime64 through feature engineering.
- Drop columns unnamed as it was not providing any useful 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 Saturdays 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.

CONCLUSION:

The product sales analysis workflow is a vital component of any business success. By meticulously tracking and analyzing sales data, companies can make informed decisions, identify trends, and adapt their strategies to maximize revenue and customer satisfaction. This process enables businesses to understand their products performance, customer preferences, and market dynamics, ultimately leading to improved profitability and competitiveness.