**Product Sales Analysis**

**Project 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: Define the specific insights you want to extract from the sale data, such as identifying top-selling products, analyzing sales trends, and understanding customer preferences.
2. Data Collection: Determine the sources and methods for collecting sales data, including transaction records, product information, and customer demographics.
3. Visualization Strategy: Plan how to visualize the insights using IBM Cognos to create interactive dashboards and reports.
4. Actionable Insights: Identify how the derived insights can guide inventory management and marketing strategies.

**Title: Innovation Phase\_2**

**Data Analytics Design for Product Sales Analysis**

**with IBM Cognos**

**Introduction:**

In today's competitive business landscape, organizations need to stay ahead of the curve by making data-driven decisions. One of the critical aspects of this process is understanding and predicting future sales trends and customer behaviors. To achieve this, it is essential to incorporate machine learning algorithms into your data analytics strategy. In this document, we will explore the steps to leverage machine learning for product sales analysis using IBM Cognos.

**Understanding the Problem:**

To effectively integrate machine learning into product sales analysis, it's essential to understand the problem at hand. We are looking to predict future sales trends and customer behaviors. This involves various aspects:

Sales Data: Gather historical sales data, including product details, purchase history, customer demographics, and more.

Data Preprocessing: Clean and preprocess the data to ensure it's suitable for machine learning. This includes handling missing values, encoding categorical variables, and scaling numerical features.

Feature Engineering: Identify relevant features that can influence sales trends and customer behaviors. This may include variables like product type, price, promotions, customer location, and more.

Model Selection: Choose the appropriate machine learning algorithms for the task. Common choices include regression models for sales prediction and clustering/classification models for customer behavior analysis.

Training and Evaluation: Train the selected models on the historical data and evaluate their performance using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or accuracy, depending on the specific task.

Integration with IBM Cognos: Incorporate the machine learning models into IBM Cognos, a robust business intelligence tool that allows for advanced reporting and analytics.

**Implementing Machine Learning with IBM Cognos**

**Data Preparation:**

Data Collection: Gather historical sales data from various sources, ensuring that it's comprehensive and relevant.

Data Cleaning: Clean the data by handling missing values, removing duplicates, and dealing with outliers. Ensure data quality and consistency.

Feature Engineering: Identify key features that might impact sales and customer behaviors. These could include product attributes, sales channels, time of purchase, and customer demographics.

**Model Development:**

Machine Learning Algorithm Selection: Choose machine learning algorithms based on the problem. For predicting sales trends, regression algorithms like Linear Regression, Random Forest, or XGBoost can be effective. For customer ehaviour analysis, clustering algorithms like K-Means or classification algorithms like Decision Trees can be used.

Model Training: Split the data into training and testing sets. Train the selected machine learning models on the training data.

Model Evaluation: Assess the model’s performance using relevant evaluation metrics. Tune hyperparameters if necessary to improve model accuracy.

**Integration with IBM Cognos:**

IBM Cognos Connection: Establish a connection between your machine learning models and IBM Cognos. Most modern machine learning libraries support exporting models to common formats like PMML (Predictive Model Markup Language) or deploying models as web services.

Reporting and Dashboard Creation: In IBM Cognos, create reports and dashboards that display the predictions made by your machine learning models. This can include visual representations of sales trends and customer behavior insights.

Automation and Scheduling: Set up automation to regularly update your data and predictions. IBM Cognos allows you to schedule reports and dashboards for automatic generation and distribution.

**Coding:**

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error

from sklearn.cluster import KMeans

# Load your historical sales data into a pandas DataFrame

sales\_data = pd.read\_csv('sales\_data.csv')

# Data Preprocessing

# Remove rows with missing values

sales\_data.dropna(inplace=True)

# Feature selection

# Define the features that influence sales prediction

features = ['Product\_Type', 'Price', 'Promotion', 'Customer\_Location']

# Split the data into training and testing sets

X = sales\_data[features]

y = sales\_data['Sales']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Sales Prediction Model (Random Forest Regression)

sales\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)

sales\_model.fit(X\_train, y\_train)

sales\_predictions = sales\_model.predict(X\_test)

mse = mean\_squared\_error(y\_test, sales\_predictions)

print("Mean Squared Error for Sales Prediction:", mse)

# Customer Behavior Analysis (Clustering with K-Means)

# Define features for customer behavior analysis

customer\_features = ['Product\_Type', 'Customer\_Location']

# Create a DataFrame for customer behavior analysis

customer\_data = sales\_data[customer\_features]

# Define the number of clusters

n\_clusters = 3

# Apply K-Means clustering

kmeans = KMeans(n\_clusters=n\_clusters, random\_state=42)

customer\_data['Cluster'] = kmeans.fit\_predict(customer\_data)

# Display the results

print("Customer Behavior Clusters:")

print(customer\_data)

# Export the models to use with IBM Cognos

from sklearn.externals import joblib

joblib.dump(sales\_model, 'sales\_model.pkl')

joblib.dump(kmeans, 'kmeans\_model.pkl')

**Conclusion:**

Incorporating machine learning algorithms into your product sales analysis with IBM Cognos c…

**PHASE 3:**

**PRODUCT SALES ANALYSIS**

**Step 1: Define Analysis Objectives**

Before you start, it's essential to clarify your analysis objectives. What do you want to achieve with your product sales analysis? Example objectives could include:

- Understanding sales trends over time.

- Identifying the top-performing products or regions.

- Forecasting future sales.

- Analyzing the impact of marketing campaigns.

- Monitoring inventory and stock levels.

Clearly defining your objectives will help shape your analysis and determine what data you need to collect.

**Step 2: Collect Sales Data**

To collect sales data, you need to identify and access the data sources. This might include databases, spreadsheets, or other data storage systems. You should have access to data on sales transactions, which typically include details like:

- Date of sale

- Product ID or name

- Sales quantity

- Sales revenue

- Customer information (if applicable)

Ensure you have the necessary permissions and access to this data.

**Step 3: Data Preprocessing and Cleaning**

Data preprocessing is a crucial step to ensure the accuracy and reliability of your analysis. Here's what you need to do:

Data Extraction: Extract the relevant data from your data sources. This might involve querying a database, exporting data from spreadsheets, or using ETL (Extract, Transform, Load) processes.

**Data Cleaning:**

Remove Duplicates: Check for and remove duplicate records, if any.

Handle Missing Data: Address missing values in the dataset. Depending on the extent of missing data, you can choose to fill in missing values with suitable defaults or remove rows with missing values.

Data Format Standardization: Ensure that data formats (e.g., date formats, currency symbols) are consistent throughout the dataset.

Data Integrity: Check for any anomalies or outliers in the data, and decide how to handle them (e.g., removing or transforming outliers).

Data Transformation:

- Create derived features if necessary (e.g., calculate total sales, profit margins).

- Aggregate data if needed, e.g., grouping sales by month, quarter, or year.

Data Integration: If you have data from multiple sources, integrate them into a single dataset for analysis.

Data Validation: Perform sanity checks to ensure that the data aligns with your defined objectives.

**Step 4: Load Data into IBM Cognos**

Once your data is cleaned and preprocessed, you can load it into IBM Cognos for analysis and visualization. This typically involves importing the data into Cognos' data modules or connecting to your data source directly.

**Step 5: Create Visualizations and Analysis**

In IBM Cognos, you can create various visualizations to analyze your sales data. You can design reports and dashboards that help you gain insights into yourdefined objectives. Some common visualizations include bar charts, line graphs, pie charts, and tables.

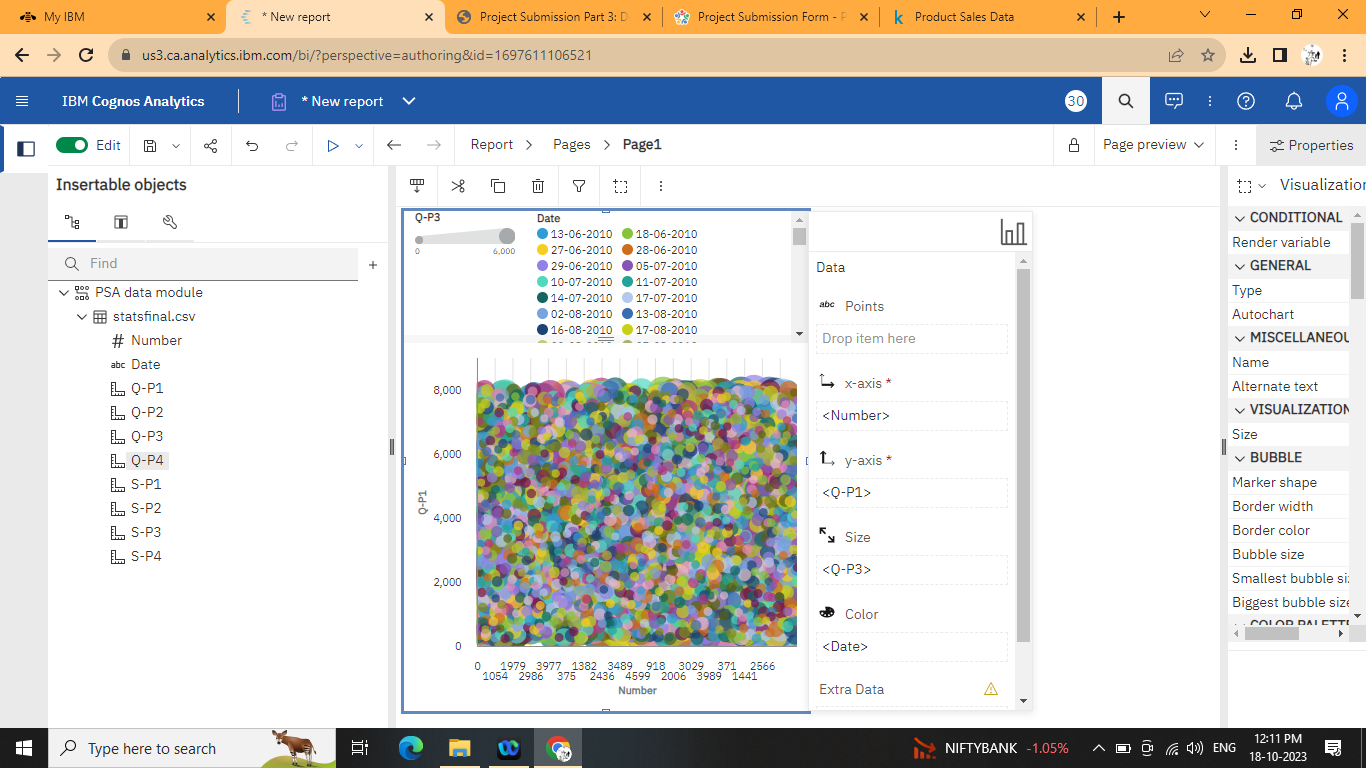
**Step 6: Interpret and Share Insights**

Finally, interpret the results and insights you've gathered from your analysis. What do the visualizations and reports tell you about your sales data and objectives? Use these insights to make informed business decisions. Share the results with stakeholders through reports, presentations, or interactive dashboards.

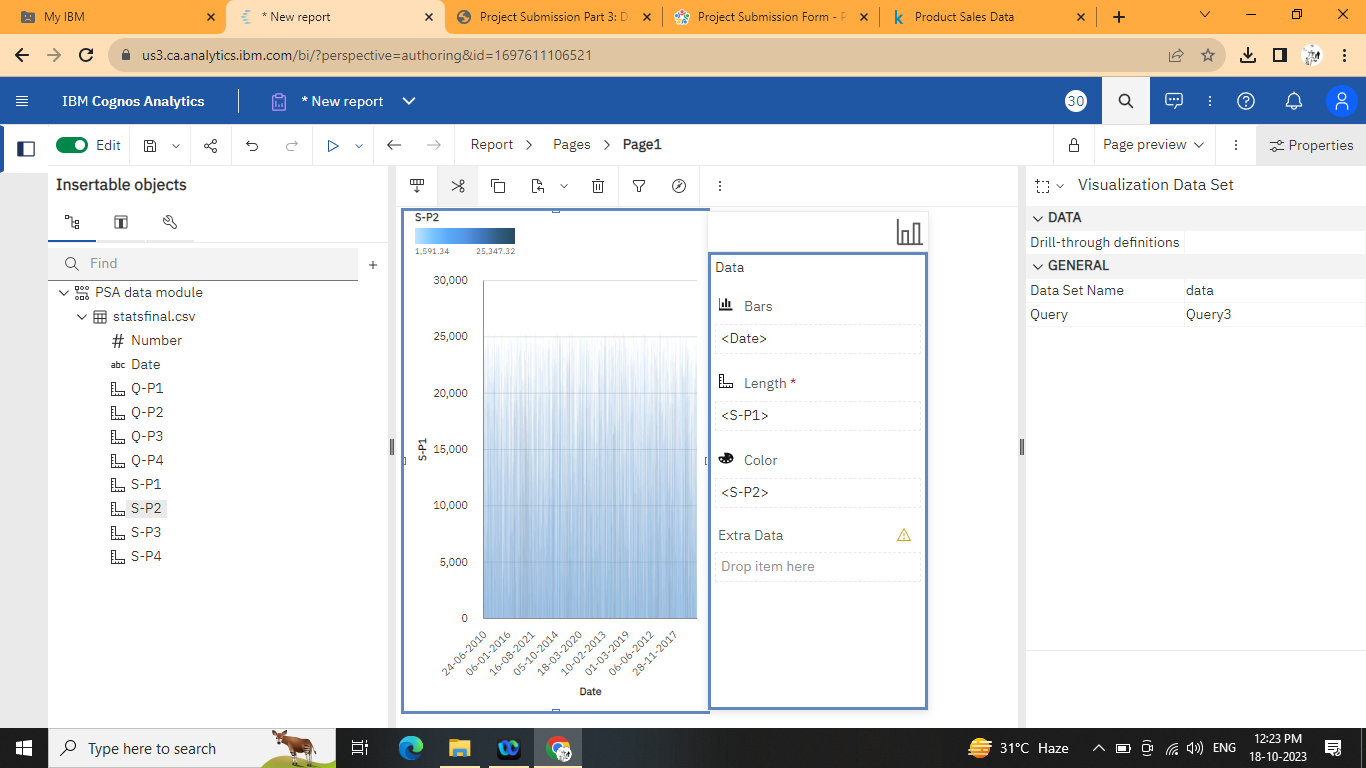
Remember to periodically update your analysis as new data becomes available to maintain the accuracy and relevance of your findings.

This process will help you begin your product sales analysis using IBM Cognos, ensuring that your data is clean and your objectives are well-defined.

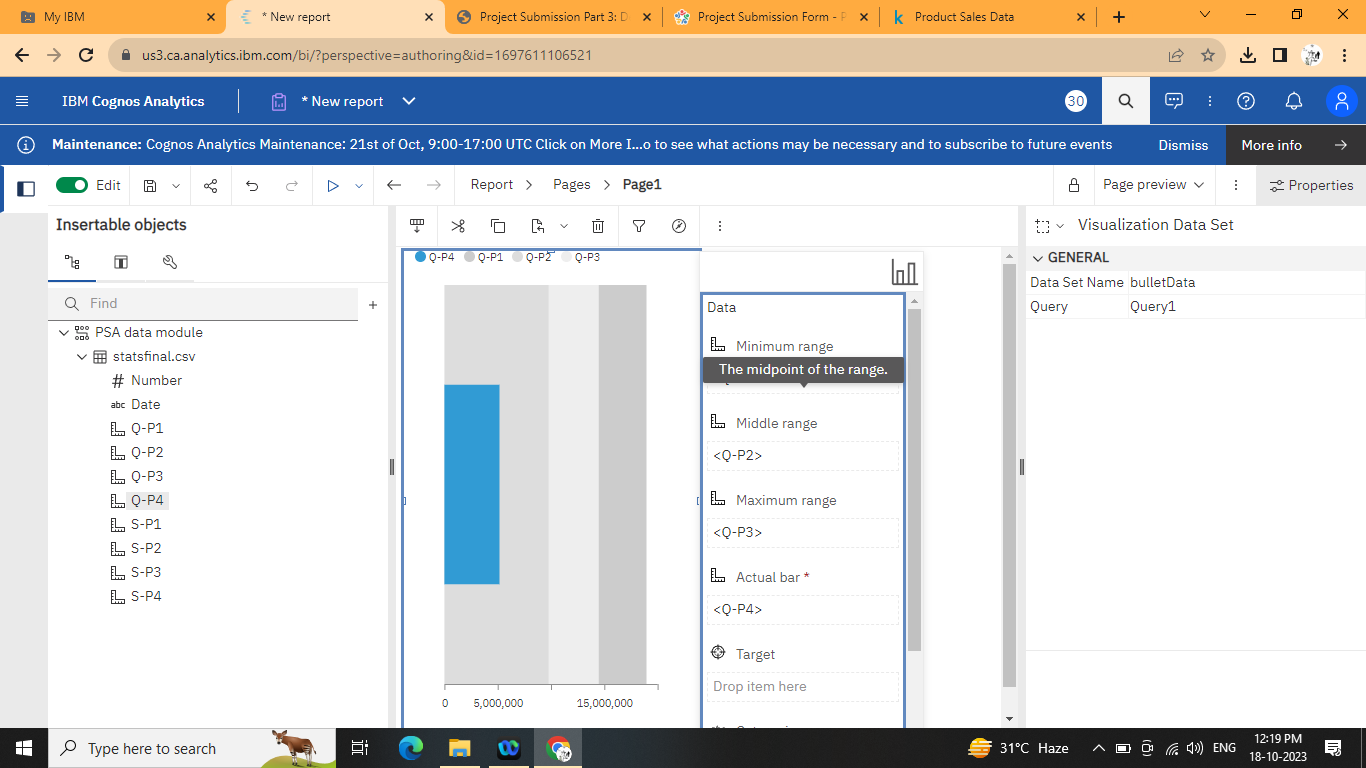
**Bubble:**



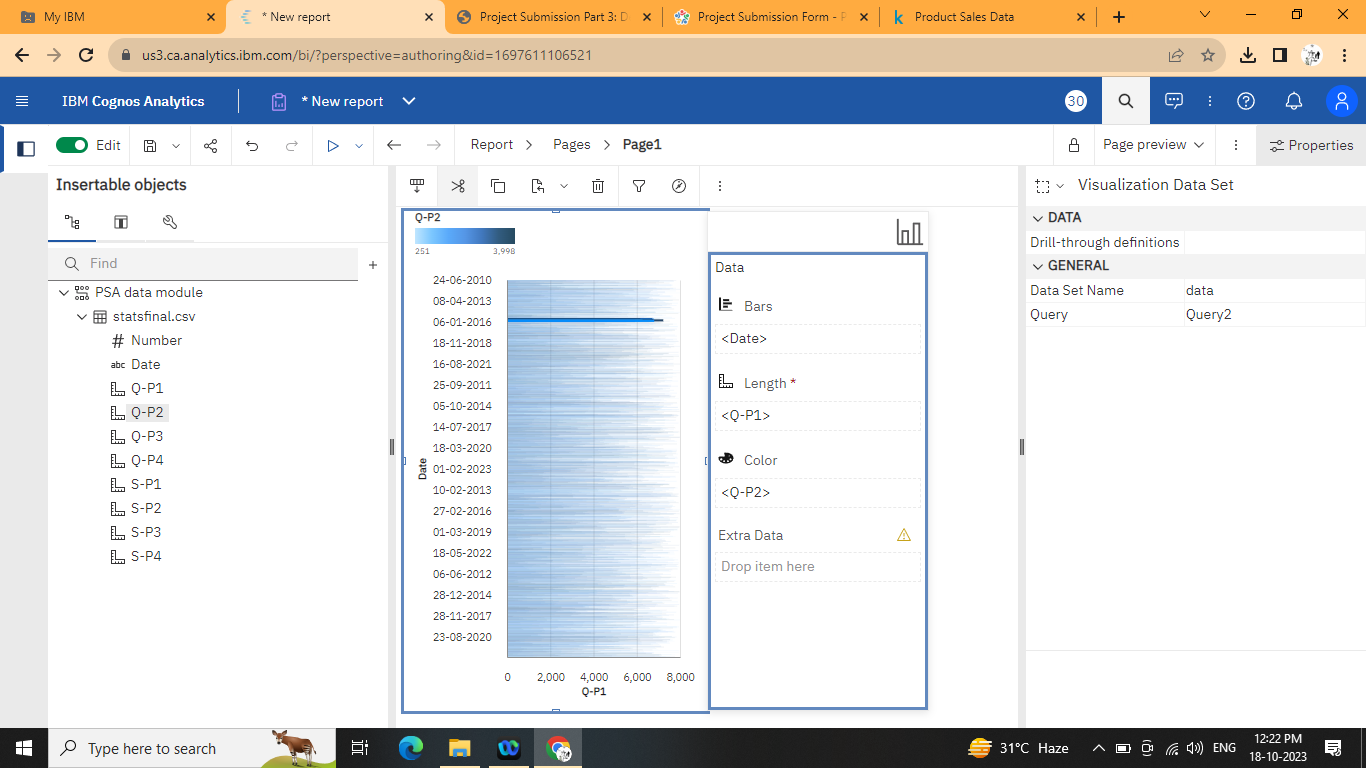
**Clustered column:**

****

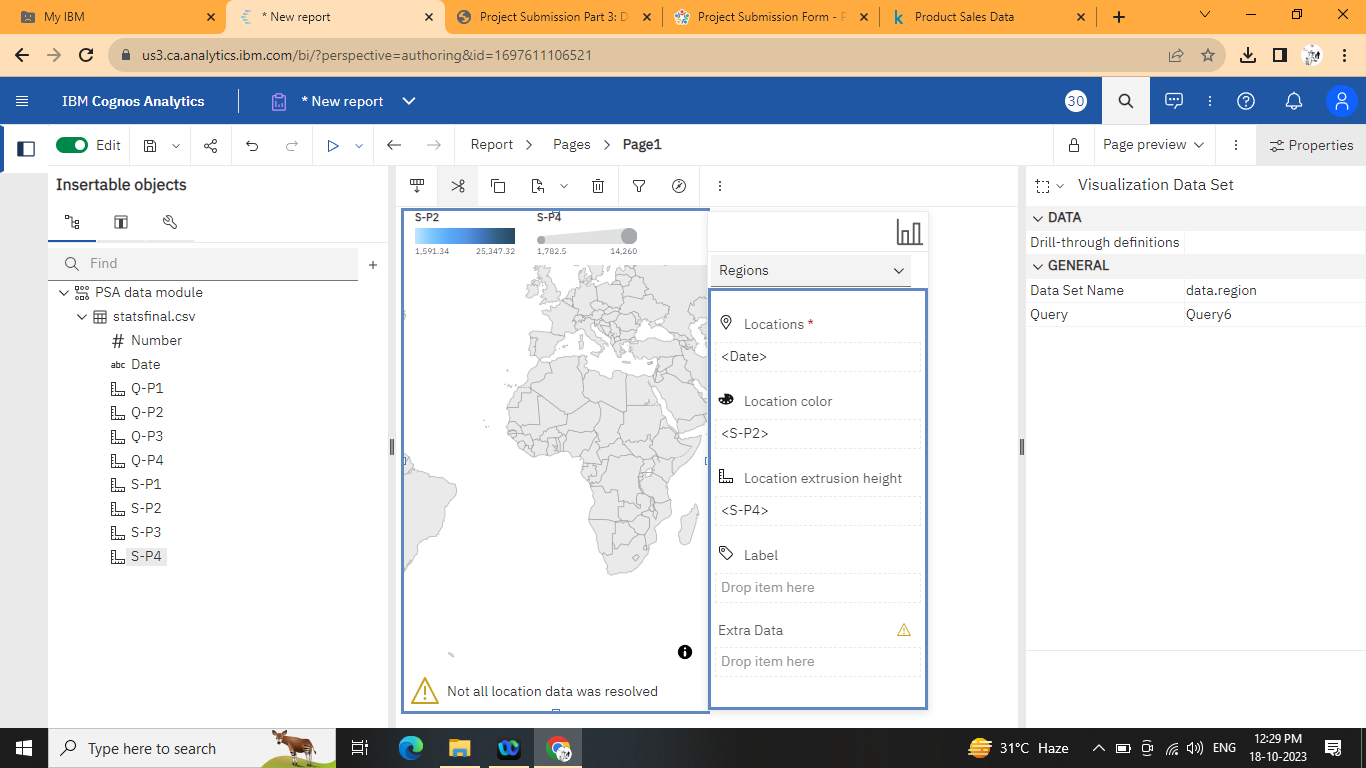
**Clustered bar:**

****

**Clustered rows:**

****

**Map:**

****

**Conclusion:**

In conclusion, embarking on a product sales analysis using IBM Cognos is a structured process that requires careful planning and execution. The key steps include defining clear analysis objectives, collecting and preprocessing sales data, and leveraging IBM Cognos for visualization and analysis.

**Data Analytics Design for Product Sales Analysis with IBM Cognos**

**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**

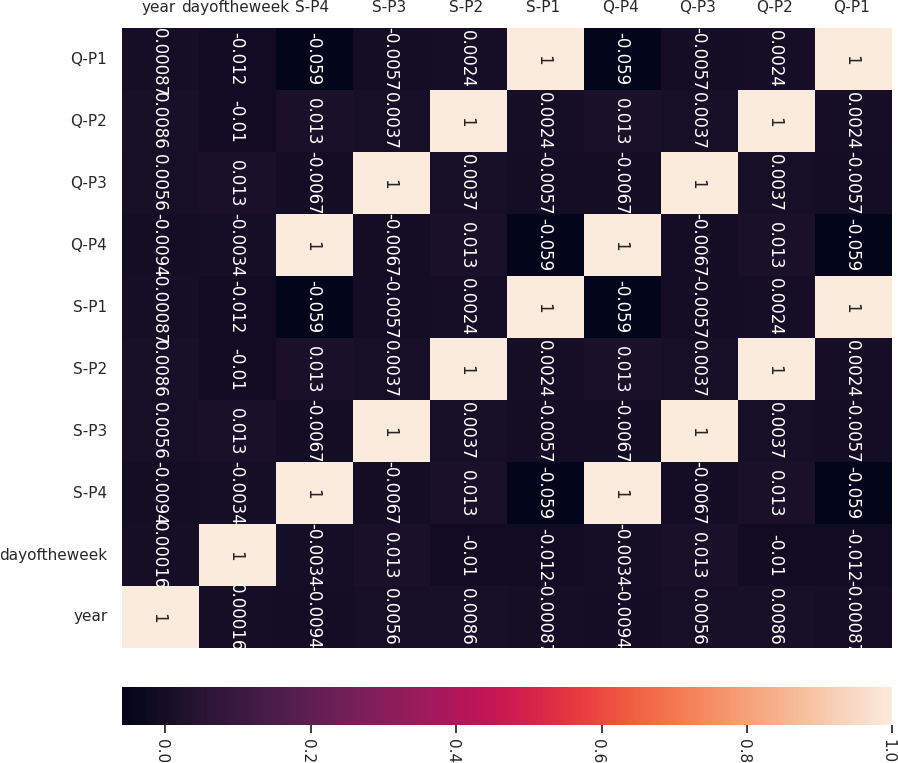
Summarize key takeaways from the visualizations.

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

**Code and Outputs**

1. **Code** plt.figure(figsize=(10,10)) sns.heatmap(df.corr(),annot=True)

**Out**



1. Code

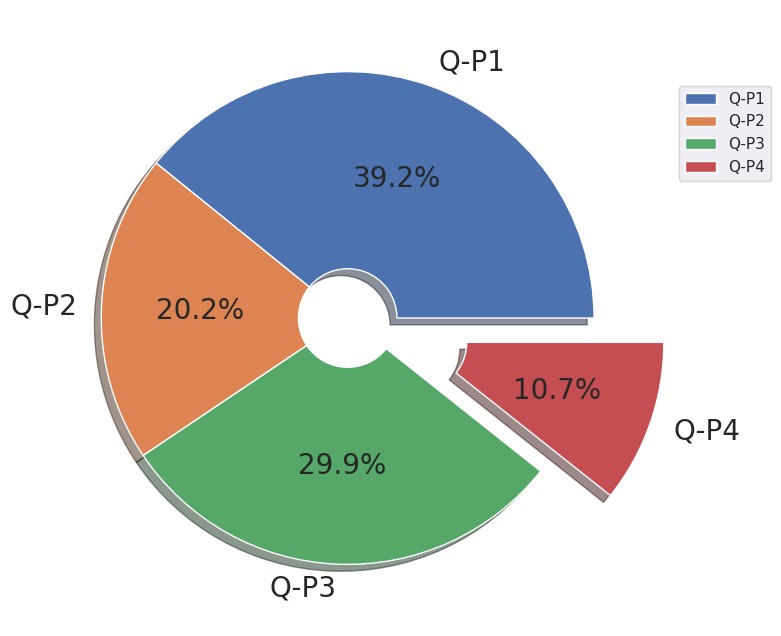
q = df[["Q-P1","Q-P2","Q-P3","Q-P4"]].sum()

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

plt.pie(q,labels=df[["Q-P1","Q-P2","Q-P3","Q-P4"]].sum().index,shado w=True,autopct="%0.01f%%",textprops={"fontsize":20},wedgeprops={'wid th': 0.8},explode=[0,0,0,0.3])

Out

plt.legend(loc='center right', bbox\_to\_anchor=(1.2, 0.8));



Q-P1

1

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

1. 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,shado w=True,autopct="%0.01f%%",textprops={"fontsize":20},wedgeprops={'wid th': 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

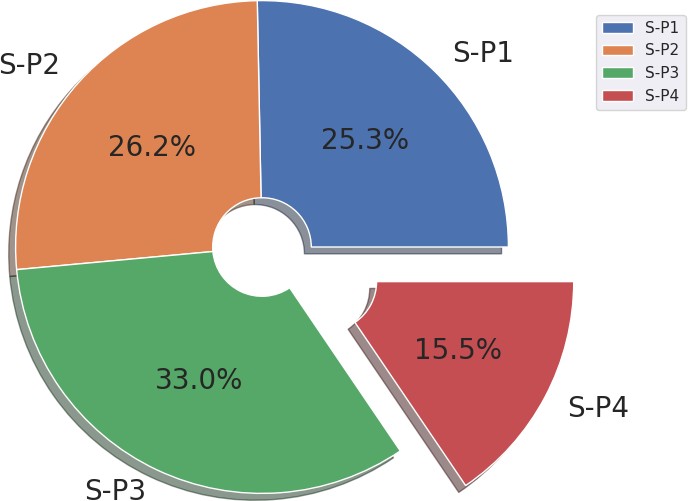
S-P2 62127530.30

S-P3 78429589.68

S-P4 36848553.00

dtype: ﬂoat64

<matplotlib.legend.Legend at 0x79ead813ff10>

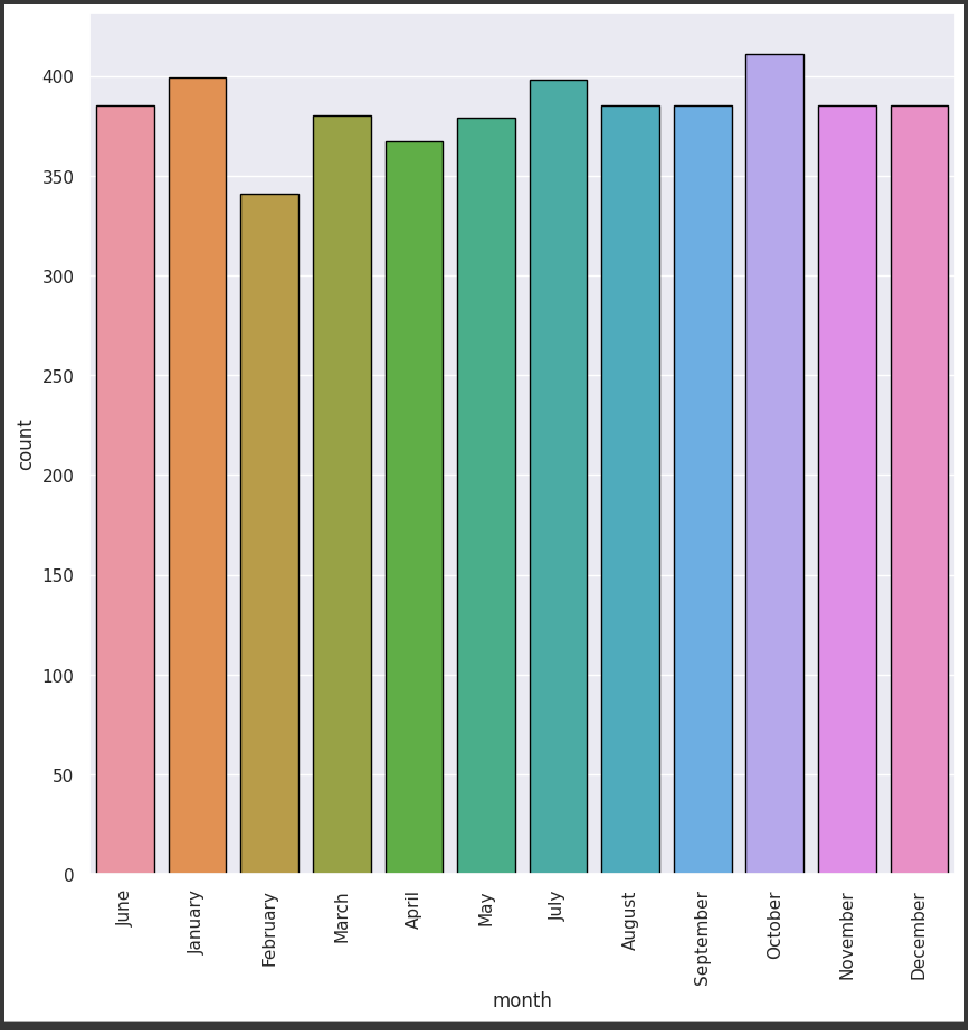


1. Code

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

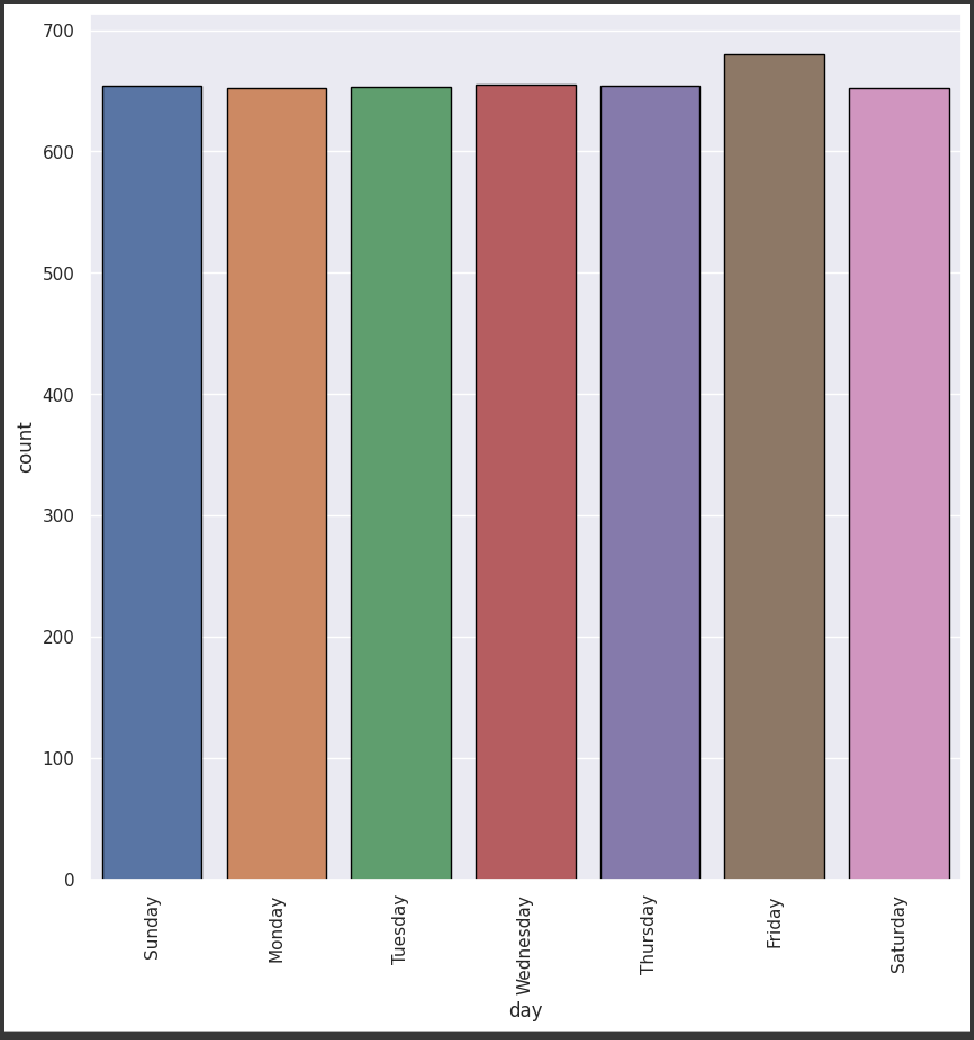


1. Code

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



1. Code

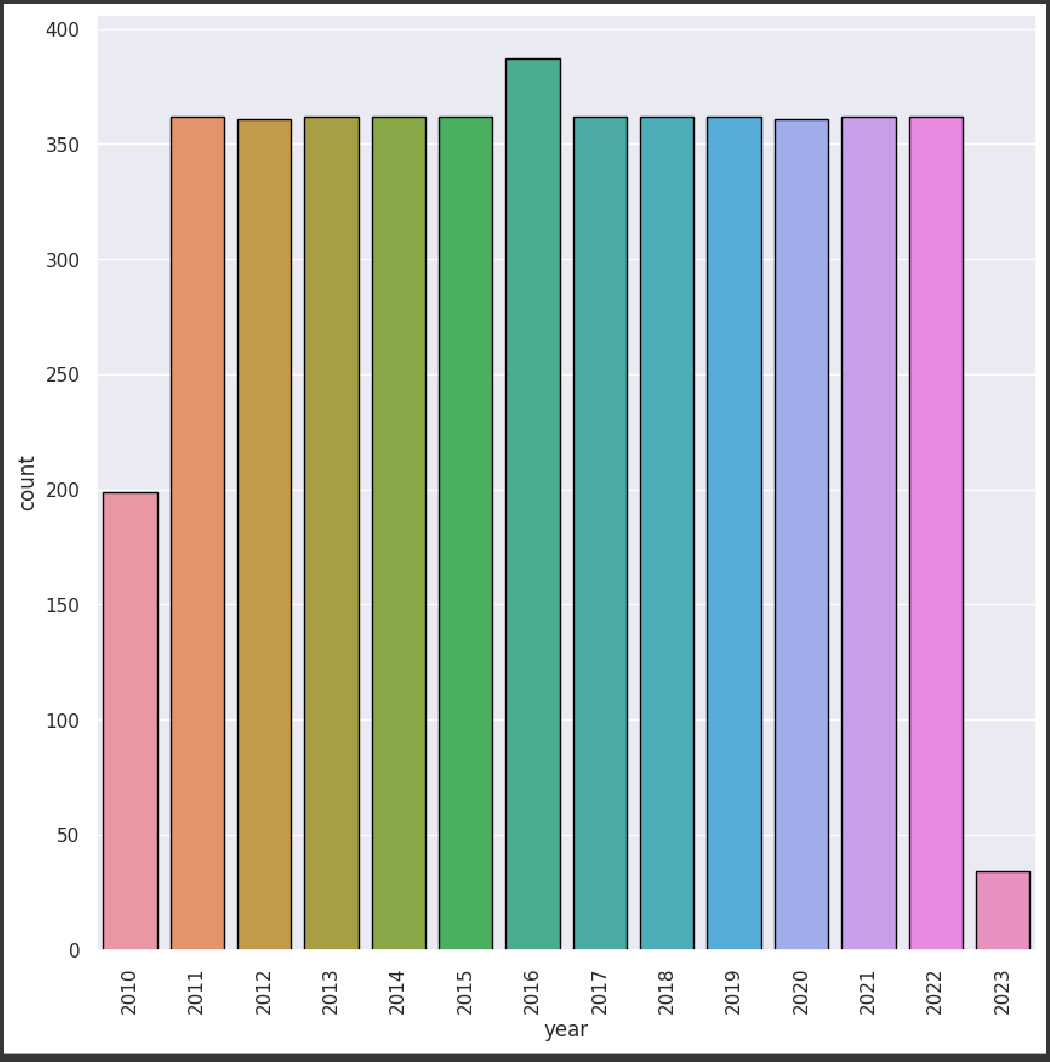
print(df["year"].value\_counts()) plt.figure(figsize=(10,10)) sns.countplot(x="year",data=df,edgecolor="black") plt.xticks(rotation=90);

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



1. Code

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

plt.xticks(rotation=90);

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

plt.xticks(rotation=90);

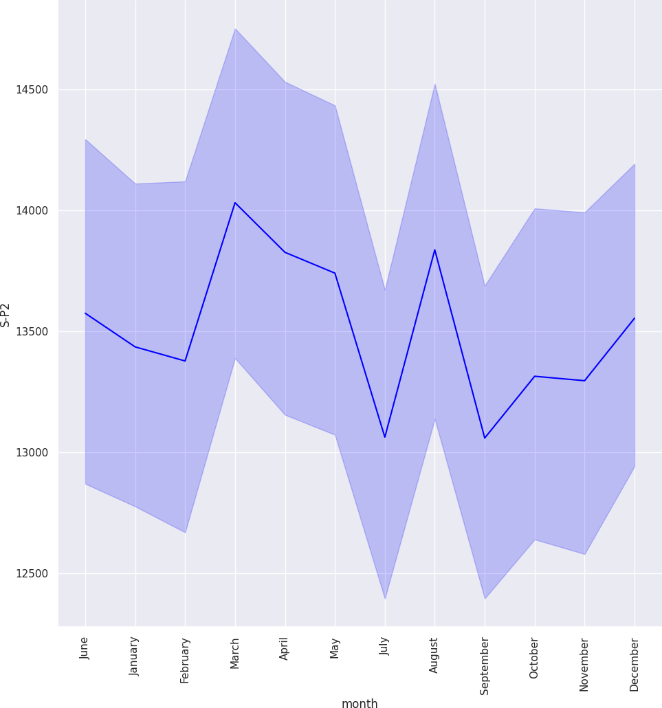
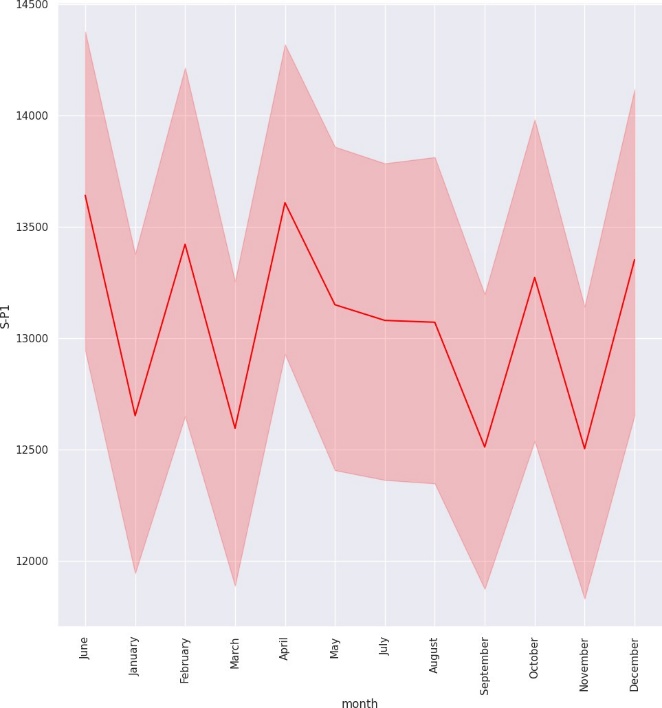
sns.relplot(x="month",y="S-P3",data=df,kind="line",height=10,c olor="green")

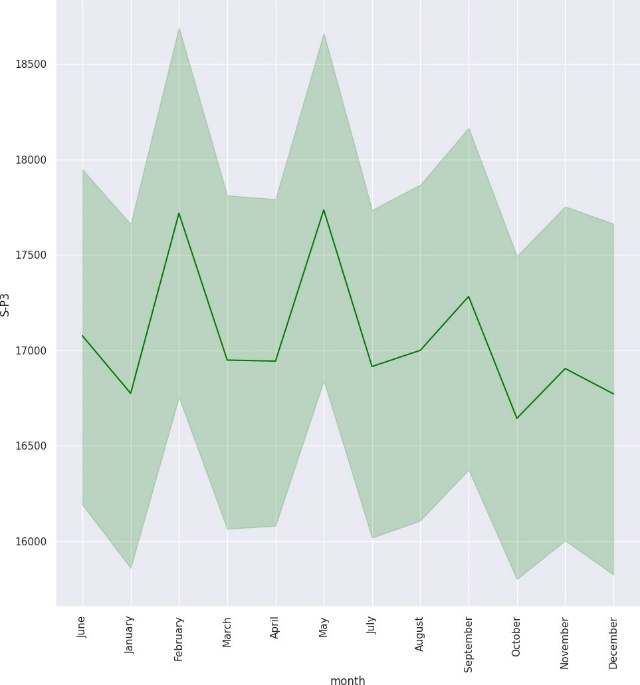
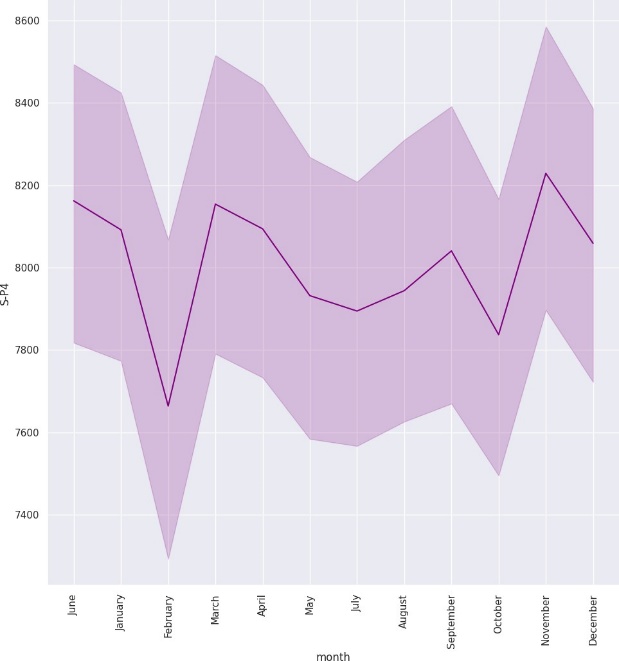
plt.xticks(rotation=90);

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

plt.xticks(rotation=90);

Out

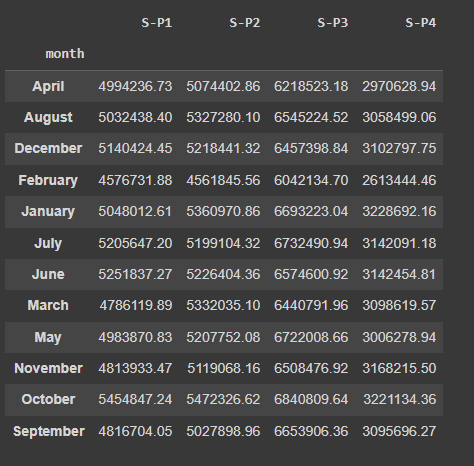


1. Code

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

Out



1. Code

plt.figure(figsize=(15,15),dpi=100) plt.subplot(2,2,1)

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

plt.xticks(rotation=90); plt.subplot(2,2,2)

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

plt.xticks(rotation=90); plt.subplot(2,2,3)

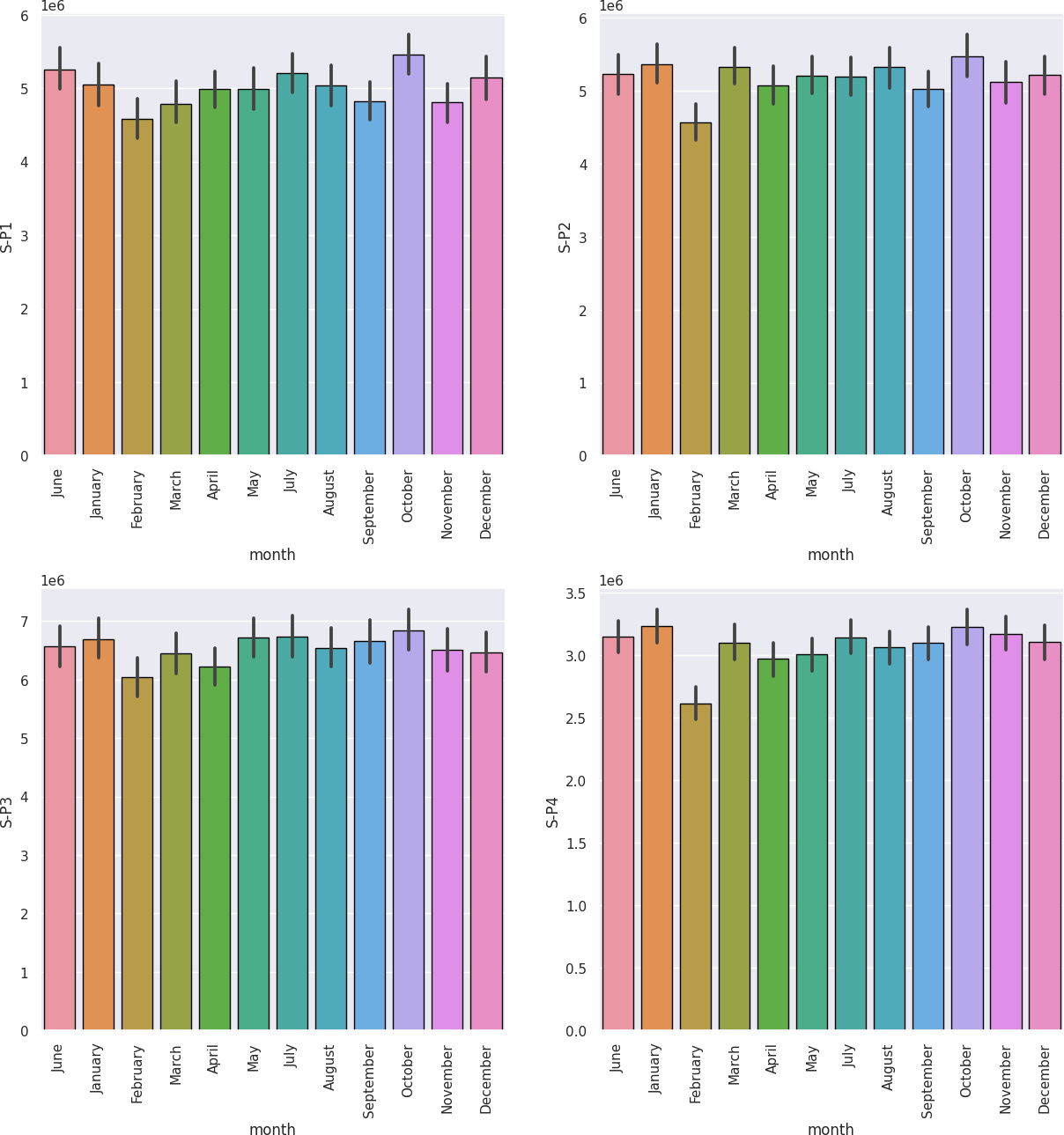
sns.barplot(x="month",y="S-P3",data=df,edgecolor="black",estim ator=sum)

plt.xticks(rotation=90); plt.subplot(2,2,4)

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

plt.xticks(rotation=90) plt.subplots\_adjust(hspace=0.3);

Out



1. Code

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

Out



1. 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",estim ator=sum)

plt.xticks(rotation=90); plt.subplot(2,2,3)

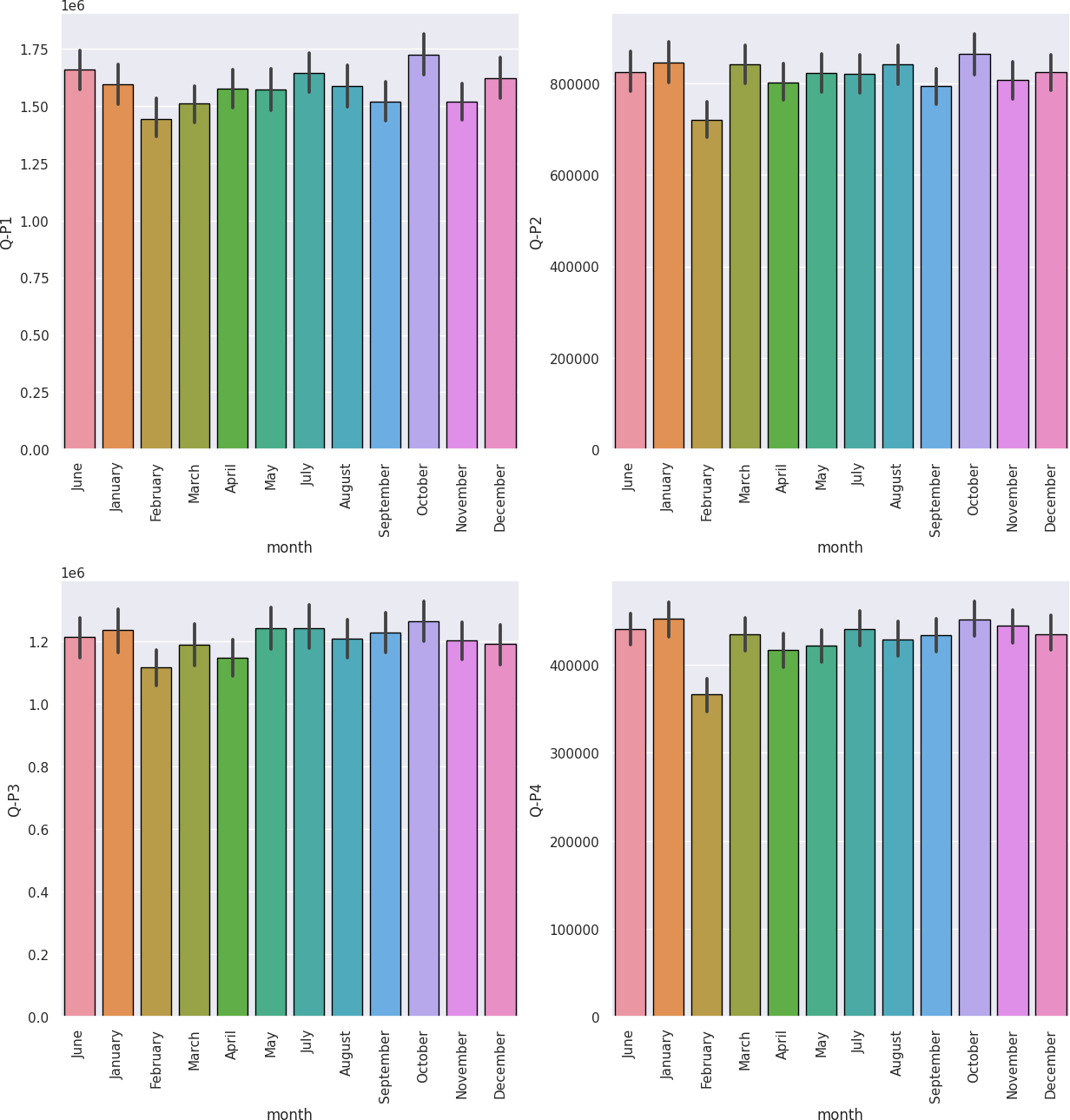
sns.barplot(x="month",y="Q-P3",data=df,edgecolor="black",estim ator=sum)

plt.xticks(rotation=90); plt.subplot(2,2,4)

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

plt.xticks(rotation=90) plt.subplots\_adjust(hspace=0.3);

Out



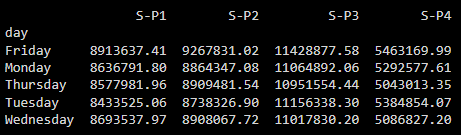
1. Code

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



1. 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)

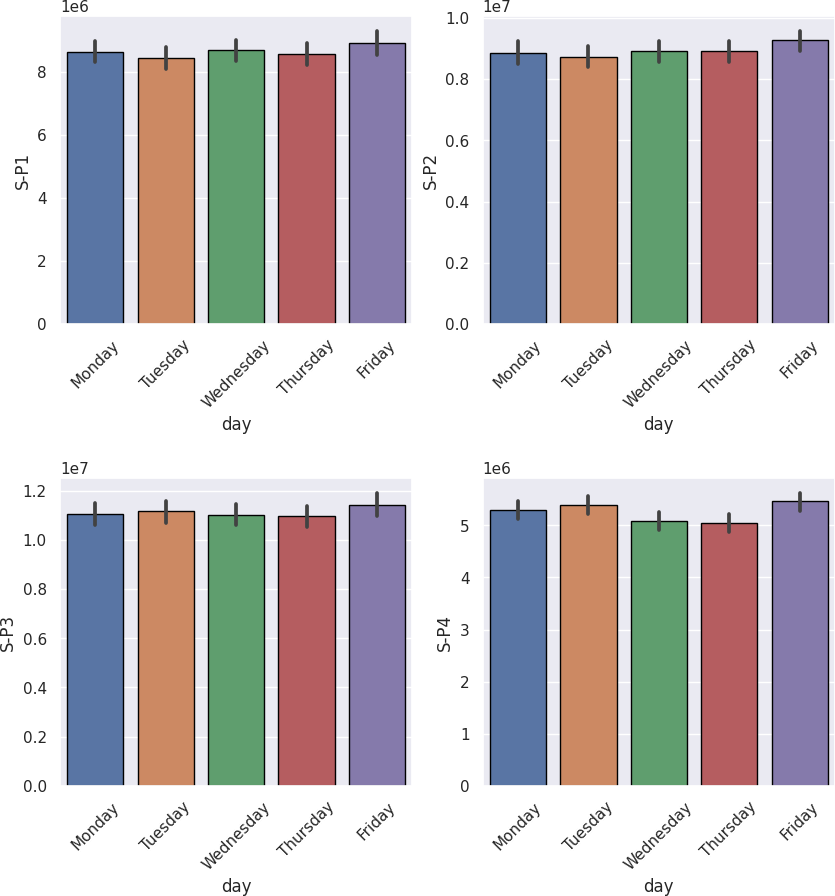
sns.barplot(x="day",y="S-P3",data=week\_t,edgecolor="black",est imator=sum)

plt.xticks(rotation=45); plt.subplot(2,2,4)

sns.barplot(x="day",y="S-P4",data=week\_t,edgecolor="black",est imator=sum)

plt.xticks(rotation=45) plt.subplots\_adjust(hspace=0.5);

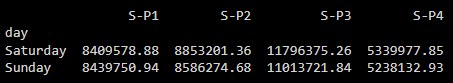
Out



1. Code

print(weekend\_t.groupby("day")[["S-P1","S-P2","S-P3","S-P4"]]. sum())

Out



1. 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)

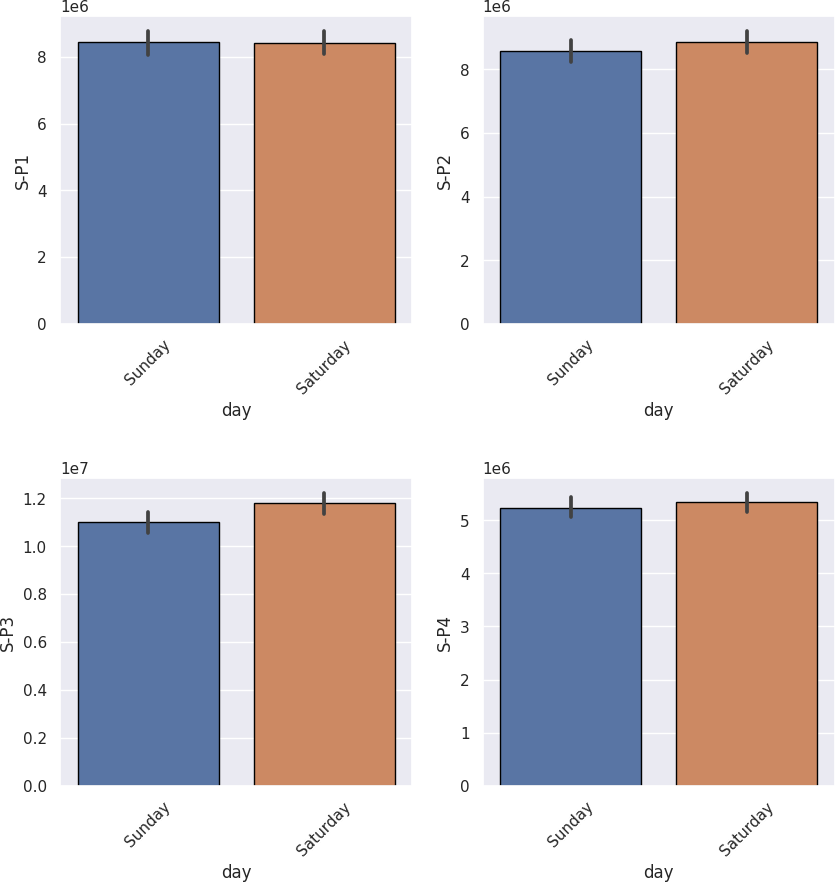
sns.barplot(x="day",y="S-P3",data=weekend\_t,edgecolor="black", estimator=sum)

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);

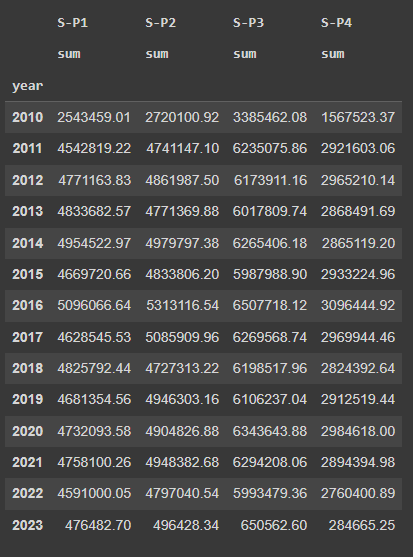
Out



1. Code

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

Out



1. 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)

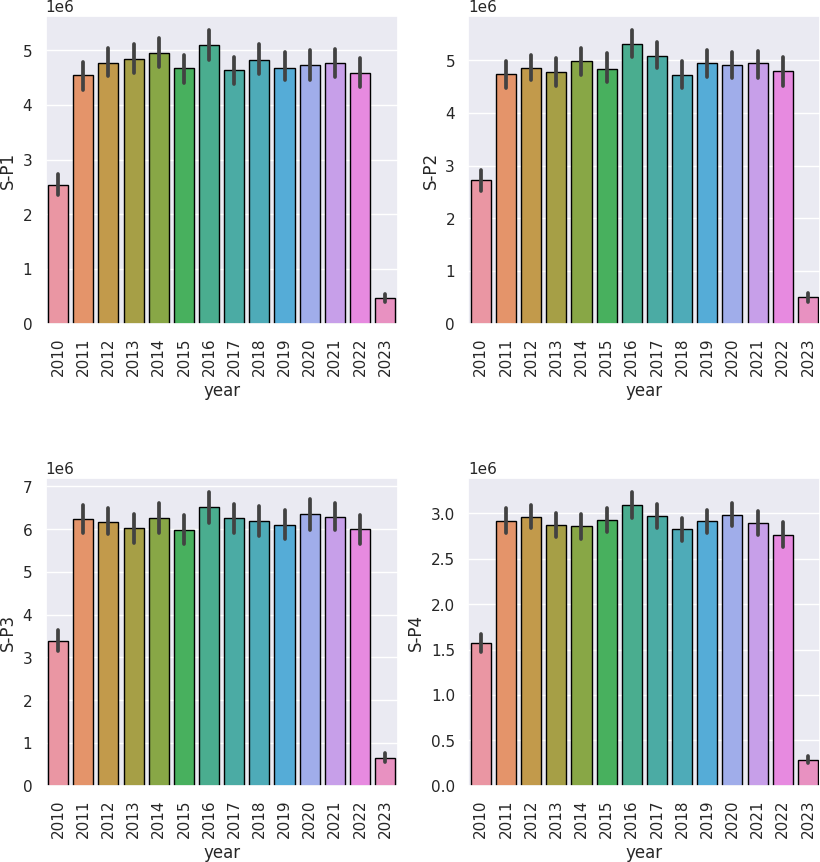
sns.barplot(x="year",y="S-P3",data=df,edgecolor="black",estima tor=sum)

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);

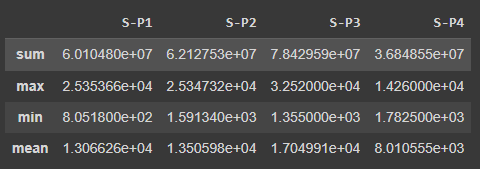
Out



1. Code

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

Out



1. 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)

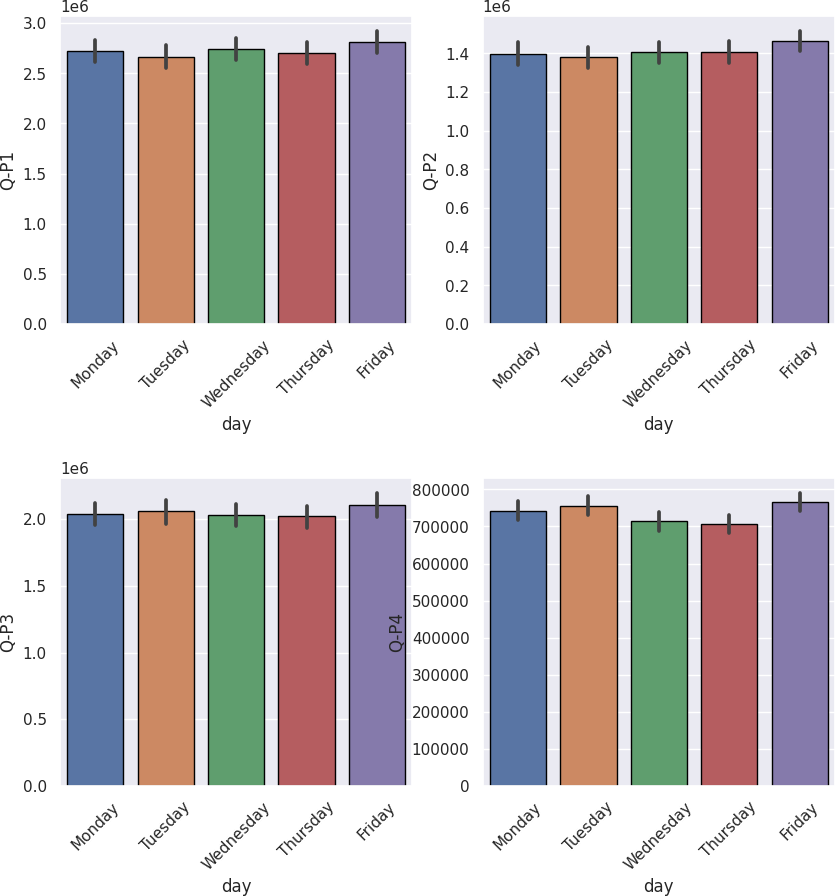
sns.barplot(x="day",y="Q-P3",data=week\_t,edgecolor="black",est imator=sum)

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



1. 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)

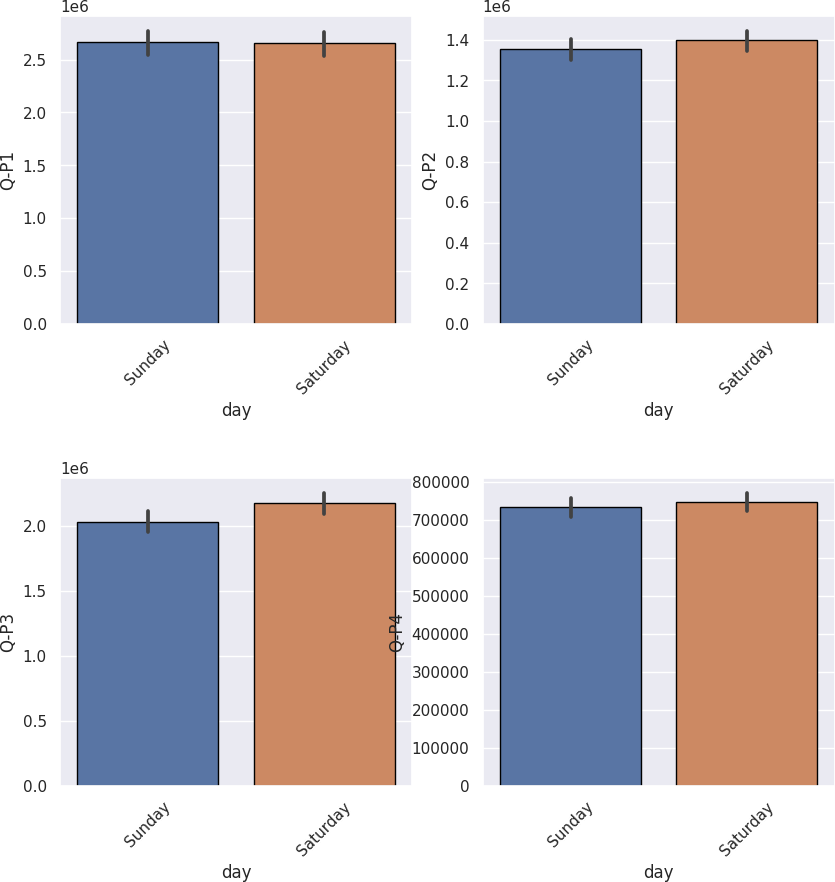
sns.barplot(x="day",y="Q-P3",data=weekend\_t,edgecolor="black", estimator=sum)

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



1. Code

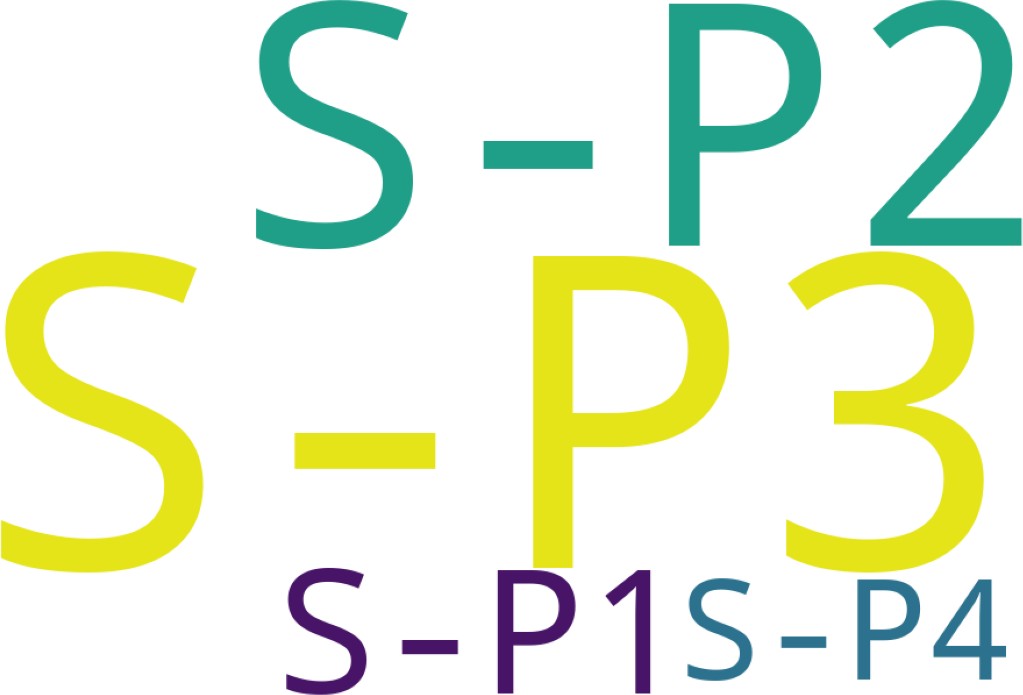
from wordcloud import WordCloud as word d=df[["S-P1","S-P2","S-P3","S-P4"]].sum()

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



1. Code

Out

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()



**SUBMISSION:**

1. <https://github.com/JTSubhashini/Product-sales-analysis-.git>
2. **Provide instructions on how to replicate the analysis and generate visualizations using IBM Cognos.**

1. \*\*Access IBM Cognos\*\*:

- Launch your IBM Cognos environment and log in with your credentials.

2. \*\*Data Source Connection\*\*:

- Connect to your data source, which could be a database, Excel file, or other data repository. This often involves setting up a data connection or importing the data into Cognos.

3. \*\*Create a New Report or Dashboard\*\*:

- Create a new report or dashboard where you want to replicate your analysis. The process may vary depending on your specific Cognos version.

4. \*\*Select Data\*\*:

- Add the data you want to analyze to your report or dashboard. This typically involves selecting the data source you connected to in step 2.

5. \*\*Create Data Items\*\*:

- Define data items or fields you want to use in your analysis. These could be dimensions (e.g., date, product) and measures (e.g., sales, profit). You may need to create calculated data items if your analysis requires them.

6. \*\*Build Queries\*\*:

- Construct queries to extract and filter the data you need. This can involve specifying conditions and criteria to narrow down your dataset.

7. \*\*Build Visualizations\*\*:

- Add visualizations to your report or dashboard. Cognos offers various visualization options such as charts, tables, and cross-tabs. Select the appropriate visualization type for your analysis.

8. \*\*Customize Visualizations\*\*:

- Customize the appearance of your visualizations by specifying chart types, colors, labels, legends, and other formatting options.

9. \*\*Add Filters and Prompts\*\*:

- If necessary, add filters and prompts to allow users to interact with your analysis. Filters enable users to dynamically control what data they see.

10. \*\*Create Calculations and Aggregations\*\*:

- If your analysis requires calculations or aggregations (e.g., sums, averages), use Cognos functions and expressions to create these calculations.

11. \*\*Apply Sorting and Grouping\*\*:

- Arrange your data by applying sorting and grouping options. This is important for making your analysis more understandable.

12. \*\*Test Your Analysis\*\*:

- Test the report or dashboard to ensure that the visualizations and data are correct and meet your requirements.

13. \*\*Save and Publish\*\*:

- Save your analysis for future use and publish it to a location where users can access it.

14. \*\*Share and Collaborate\*\*:

- Share the analysis with relevant stakeholders and collaborate with them as needed.

15. \*\*Schedule and Automate\*\*:

- If you need to generate and share the analysis regularly, set up schedules and automation within Cognos.

16. \*\*Monitor and Maintain\*\*:

- Continuously monitor and maintain your analysis to ensure it remains accurate and up to date.

Please note that the specific steps may vary based on your version of IBM Cognos, and the terminology used in the software might be slightly different. It's important to refer to the official documentation for your specific Cognos version for more detailed instructions and guidance. Additionally, it's a good practice to receive training or work with a Cognos expert if you are new to the platform, as it can be complex and powerful.

**3.Include example outputs of the visualizations and derived insights.**

1. \*\*Bar Chart: Sales by Product Category\*\*

Visualization: A bar chart showing the total sales for different product categories.

Derived Insight: The bar chart reveals that Electronics is the highest-selling category, followed by Clothing and Home Appliances. This suggests that focusing on the Electronics category could be profitable.

2. \*\*Line Chart: Website Traffic Over Time\*\*

Visualization: A line chart displaying website traffic over a period of one year.

Derived Insight: The line chart shows a significant increase in website traffic from April to July, followed by a decline. This insight could prompt further investigation into the reasons behind this trend and the effectiveness of marketing efforts during this period.

3. \*\*Pie Chart: Market Share of Smartphone Manufacturers\*\*

Visualization: A pie chart illustrating the market share of different smartphone manufacturers.

Derived Insight: The pie chart indicates that Apple holds the largest market share, followed by Samsung and Xiaomi. This information is crucial for understanding the competitive landscape in the smartphone industry.

4. \*\*Scatter Plot: Relationship Between Advertising Spending and Sales\*\*

Visualization: A scatter plot with advertising spending on the x-axis and sales on the y-axis for various months.

Derived Insight: The scatter plot shows a positive correlation between advertising spending and sales. As spending increases, sales tend to increase as well. This suggests that investing more in advertising could lead to higher sales.

5. \*\*Heatmap: Customer Churn by Demographic Segments\*\*

Visualization: A heatmap showing the customer churn rate for different demographic segments (e.g., age groups and income levels).

Derived Insight: The heatmap highlights that customers in the 18-24 age group with lower income levels have the highest churn rate. This insight can guide targeted marketing efforts to retain this particular customer segment.

6. \*\*Histogram: Distribution of Product Prices\*\*

Visualization: A histogram displaying the distribution of product prices in a catalog.

Derived Insight: The histogram reveals that the majority of products are priced between $50 and $100. Understanding the price distribution can help with pricing strategy and product positioning.

7. \*\*Box Plot: Employee Salaries by Department\*\*

Visualization: A box plot showing the distribution of salaries by department in a company.

Derived Insight: The box plot demonstrates that the Marketing department has a wider salary range compared to other departments. This could indicate variations in roles or experience levels within the Marketing team.