**DATA ANALYTICS**

**PRODUCT SALES ANALYSIS**

**NAME: NAGENTHIRA KUMAR D**

**NM ID:au211521106104**

**ABSTRACT:**

In an era characterized by data-driven decision-making, businesses are increasingly relying on data analysis to gain insights and make informed choices. This project delves into the realm of data analysis, specifically focusing on a dataset titled "Product Sales Analysis." The dataset comprises comprehensive information about product sales over a defined period. Through this project, we aim to extract valuable insights, uncover trends, and draw conclusions that can inform strategic decisions and optimize sales performance.

The project begins with data preprocessing and exploratory data analysis, where we clean and organize the data, identify key attributes, and perform initial visualizations. Subsequently, we delve into more advanced data analysis techniques, including trend analysis, seasonality detection, and forecasting. These techniques enable us to uncover patterns in sales data, recognize peak sales periods, and predict future trends.

Our analysis encompasses a variety of dimensions, including product categories, customer segments, and geographic regions. We identify the top-performing products and customer groups, providing insights into potential areas for expansion and improvement. Moreover, we examine the impact of external factors, such as seasonality and marketing campaigns, on sales performance.

In conclusion, this project demonstrates the power of data analysis in shedding light on product sales. It underlines the significance of understanding sales trends, customer behavior, and the impact of external factors for effective decision-making. By applying data analysis techniques, businesses can enhance their competitiveness, optimize inventory management, and tailor marketing strategies to boost sales. The findings from this analysis can guide strategic decisions, contributing to improved sales performance and long-term success.

**PROJECT OVERVIEW:**

In today's competitive business landscape, understanding product sales trends is crucial for companies aiming to make informed decisions and maximize profitability. This project delves into the intricate world of product sales analysis, where we will dissect sales data to uncover valuable insights. By leveraging data analytics and visualization tools, we aim to identify top-performing products, seasonal trends, customer preferences, and potential growth opportunities. Through this analysis, we will empower businesses with the knowledge they need to optimize their product offerings, marketing strategies, and inventory management, ultimately driving success in the dynamic marketplace.

**DESIGN THINKING:**

* **ANALYSIS OBJECTIVE:**

**1. Identifying Top-Selling Products:**

- Utilize IBM Cognos to create reports that rank products by sales revenue.

- Implement filters to view top-selling products for specific time periods or regions.

- Visualize product sales data through charts and graphs to quickly identify trends.

- Employ IBM Cognos' interactive features to drill down into detailed information about these products, including sales volume, profit margins, and customer demographics.

**2. Analyzing Sales Trends:**

- Leverage IBM Cognos to generate time-series analyses and trend reports.

- Use historical data to create forecasting models within IBM Cognos to predict future sales trends.

- Utilize geospatial analysis features to identify regional variations in sales trends.

- Enable stakeholders to interact with dynamic dashboards in IBM Cognos, allowing them to explore sales trends across different dimensions (e.g., product categories, time periods).

**3. Understanding Customer Preferences:**

- Employ IBM Cognos to segment customers based on various criteria such as age, gender, location, and purchase history.

- Create customer profiles and personas to understand their preferences and buying behaviors.

- Utilize data visualization tools in IBM Cognos to display customer preferences through charts and graphs.

- Implement predictive analytics models within IBM Cognos to suggest product recommendations and personalized marketing strategies based on individual customer preferences.

* **DATA COLLECTION:**

**1. Data Sources Identification:**

- Identify the primary sources of sales data, which may include databases, spreadsheets, point-of-sale systems, and e-commerce platforms.

- Configure IBM Cognos to connect to these data sources, allowing for seamless data extraction and integration.

**2. Transaction Records:**

- Collect detailed transaction records that capture information such as date, time, product purchased, quantity, and price.

- Use IBM Cognos to create data connectors or data models that enable the extraction, transformation, and loading (ETL) of transaction data.

- Schedule automated data refreshes in IBM Cognos to ensure that the transaction data is up-to-date for analysis.

**3. Product Information:**

- Gather comprehensive product data, including descriptions, categories, SKU numbers, and supplier details.

- Create product data hierarchies and categorizations within IBM Cognos to facilitate product-level analysis.

- Utilize IBM Cognos' data modeling capabilities to relate product information to sales transactions for more meaningful insights.

**4. Customer Demographics:**

- Collect customer demographic data, such as age, gender, location, and purchase history.

- Integrate customer demographic data with sales data in IBM Cognos to enable customer segmentation and personalization.

- Use data blending techniques in IBM Cognos to combine data from different sources, ensuring a holistic view of customer demographics.

**5. Data Validation and Cleaning:**

- Implement data validation and cleansing processes within IBM Cognos to ensure data accuracy and consistency.

- Utilize data profiling features in IBM Cognos to identify and address data quality issues, such as missing or erroneous data.

**6. Data Security and Access Control:**

- Implement role-based access control and data security measures in IBM Cognos to restrict access to sensitive sales data.

- Configure data-level permissions to ensure that only authorized users can view and manipulate specific datasets.

* **VISUALIZATION STRATEGY:**

**1. Dashboard and Report Design:**

- Use IBM Cognos to design interactive and visually appealing dashboards and reports.

- Select appropriate chart types (e.g., bar charts, line graphs, pie charts) to represent different types of insights, such as sales trends or product performance.

- Incorporate dynamic elements like filters and prompts in IBM Cognos to allow users to customize their views and focus on specific aspects of the data.

**2. Data Interaction and Exploration:**

- Implement interactive features in IBM Cognos, such as drill-down and drill-through capabilities, to enable users to explore data hierarchies and details.

- Utilize IBM Cognos' interactive sorting and filtering options to empower users to analyze data from multiple angles and dimensions.

- Integrate guided navigation within dashboards, providing step-by-step insights and recommendations based on user interactions.

**3. Accessibility and Collaboration:**

- Ensure that IBM Cognos dashboards and reports are accessible across various devices and platforms, allowing users to access insights on desktops, tablets, and smartphones.

- Foster collaboration by enabling users to share and collaborate on reports and dashboards, facilitating knowledge sharing and informed decision-making within teams.

- Leverage IBM Cognos' scheduling and distribution features to automate report delivery to stakeholders, ensuring timely access to critical insights.

* **ACTIONABLE INSIGHTS:**

**1. Inventory Optimization:**

- Utilize IBM Cognos-generated insights on top-selling products and sales trends to optimize inventory management.

- Set inventory reorder points and safety stock levels based on historical sales data and demand forecasting models created within IBM Cognos.

- Implement alerts and notifications in IBM Cognos to automatically trigger reordering when stock levels reach predefined thresholds, ensuring products are available when needed.

**2. Targeted Marketing Strategies:**

- Leverage customer preference insights derived from IBM Cognos to craft personalized marketing campaigns.

- Segment customers based on their preferences, purchase history, and behavior patterns.

- Design targeted marketing messages and promotions tailored to specific customer segments using IBM Cognos' segmentation and reporting capabilities.

**3. Performance Monitoring and Adaptation:**

- Continuously monitor the effectiveness of marketing strategies and inventory management decisions using IBM Cognos dashboards and reports.

- Set key performance indicators (KPIs) and track them in real-time to assess the impact of changes.

- Use IBM Cognos' ad-hoc reporting and analysis features to quickly adapt strategies based on emerging trends and customer responses.

**IMPLEMENTATION:**

**\*Strategic Decision-Making:** Product sales analysis is a pivotal tool for businesses to make data-driven decisions. It empowers companies to assess their product portfolio, optimize marketing strategies, and improve operational efficiency.

**\*Market Insights and Trends:** This analysis helps in understanding market trends and customer preferences, enabling companies to anticipate demands, align product development, and maintain a competitive edge.

**\*Enhanced Profitability:** By harnessing the power of data and analytics, product sales analysis aids in allocating resources effectively, enhancing customer satisfaction, and ultimately increasing profitability, making it indispensable in today's dynamic business landscape.

**OVERVIEW OF THE PROJECT:**

* In this phase-2 of the project,we have innovated the design of the project (PRODUCT SALES ANALYSIS) using MACHINE LEARNING .
* Here we have taken “REC corp LTD. is small-scaled business venture established in India” as an example company and done the analysed the data of that company .
* Dataset link:[**https://www.kaggle.com/datasets/ksabishek/product-sales-data**](https://www.kaggle.com/datasets/ksabishek/product-sales-data)

INPUT[1]: *# import the important packages*

import pandas as pd *# library used for data manipulation and analysis*

import numpy as np *# library used for working with arrays* import matplotlib.pyplot as plt *# library for plots and visualizations*

import seaborn as sns *# library for visualizations*

%matplotlib inline

*# To ignore warnings*

import warnings warnings.filterwarnings("ignore")

INPUT[2]:#*if you open in Kaggle editor*

data = pd.read\_csv('/content/statsfinal.csv')

*#if you open in juypter notebook*

*# data = pd.read\_csv('statsfinal.csv')*

INPUT[3]:*# Checking the first 5 and last 5 rows of the dataset*

data.head(-1)

OUTPUT 3

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

P2 \

0 0 13-06-2010 5422 3725 576 907 17187.74

23616.50

1 1 14-06-2010 7047 779 3578 1574 22338.99

4938.86

2 2 15-06-2010 1572 2082 595 1145 4983.24

13199.88

3 3 16-06-2010 5657 2399 3140 1672 17932.69

15209.66

4 4 17-06-2010 3668 3207 2184 708 11627.56

20332.38

... ... ... ... ... ... ... ... .

..

4594 4594 29-01-2023 1227 3044 5510 1896 3889.59

19298.96

4595 4595 30-01-2023 2476 3419 525 1359 7848.92

21676.46

4596 4596 31-01-2023 7446 841 4825 1311 23603.82

5331.94

4597 4597 01-02-2023 6289 3143 3588 474 19936.13

19926.62

4598 4598 02-02-2023 3122 1188 5899 517 9896.74

7531.92

|  |  |  |
| --- | --- | --- |
|  | S-P3 | S-P4 |
| 0 | 3121.92 | 6466.91 |
| 1 | 19392.76 | 11222.62 |
| 2 | 3224.90 | 8163.85 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 3 | 17018.80 | 11921.36 |
| 4 | 11837.28 | 5048.04 |
| ... | ... | ... |
| 4594 | 29864.20 | 13518.48 |
| 4595 | 2845.50 | 9689.67 |
| 4596 | 26151.50 | 9347.43 |
| 4597 | 19446.96 | 3379.62 |
|  | 4598 | 31972.58 | 3686.21 |

[4599 rows x 10 columns]

INPUT[4]:*# drop the first column*

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

INPUT[5]:<class 'pandas.core.frame.DataFrame'> RangeIndex: 4600 entries, 0 to 4599 Data columns (total 9 columns):

# Column Non-Null Count Dtype

0 Date 4600 non-null object

1. Q-P1 4600 non-null int64
2. Q-P2 4600 non-null int64
3. Q-P3 4600 non-null int64
4. Q-P4 4600 non-null int64
5. S-P1 4600 non-null float64
6. S-P2 4600 non-null float64
7. S-P3 4600 non-null float64
8. S-P4 4600 non-null float64 dtypes: float64(4), int64(4), object(1) memory usage: 323.6+ KB

[6]data.isnull().s

OUTPUT[6]:

um() 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

INPUT[7] *# Extract year from the 'Day' 'Month' 'year' from the 'Date' column using a lambda function*

*# We need to get the year from the data to analyse sales year to year*

data['Day'] = data['Date'].apply(lambda x: x.split('-')[0])

data['Month'] = data['Date'].apply(lambda x: x.split('-')[1])

OUTPUT[7]:

data['Year'] = data['Date'].apply(lambda x: x.split('-')[2]) data

Date Q-P1 Q-P2 Q-P3 Q-P4 S-P1 S-P2 S-P3

\

0 13-06-2010 5422 3725 576 907 17187.74 23616.50 3121.92

1 14-06-2010 7047 779 3578 1574 22338.99 4938.86 19392.76

2 15-06-2010 1572 2082 595 1145 4983.24 13199.88 3224.90

3 16-06-2010 5657 2399 3140 1672 17932.69 15209.66 17018.80

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

... ... ... ... ... ... ... ... ...

4595 30-01-2023 2476 3419 525 1359 7848.92 21676.46 2845.50

4596 31-01-2023 7446 841 4825 1311 23603.82 5331.94 26151.50

4597 01-02-2023 6289 3143 3588 474 19936.13 19926.62 19446.96

4598 02-02-2023 3122 1188 5899 517 9896.74 7531.92 31972.58

4599 03-02-2023 1234 3854 2321 406 3911.78 24434.36 12579.82

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | S-P4 | Day | Month | Year |
| 0 | 6466.91 | 13 | 06 | 2010 |
| 1 | 11222.62 | 14 | 06 | 2010 |
| 2 | 8163.85 | 15 | 06 | 2010 |
| 3 | 11921.36 | 16 | 06 | 2010 |
| 4 | 5048.04 | 17 | 06 | 2010 |
| ... | ... | .. | ... | ... |
| 4595 | 9689.67 | 30 | 01 | 2023 |
| 4596 | 9347.43 | 31 | 01 | 2023 |
| 4597 | 3379.62 | 01 | 02 | 2023 |
| 4598 | 3686.21 | 02 | 02 | 2023 |
| 4599 | 2894.78 | 03 | 02 | 2023 |

[4600 rows x 12 columns]

INPUT[8]

data\_reduced = data.query("Year != '2010' and Year != '2023'")

INPUT[9]

*#Create a function that allows us to plot a bar chart for the 4 products*

def plot\_bar\_chart(df, columns, stri, str1, val):

*# Aggregate sales for each product by year, by sum or mean*

if val == 'sum':

sales\_by\_year = df.groupby('Year')

[columns].sum().reset\_index() elif val == 'mean':

sales\_by\_year = df.groupby('Year') [columns].mean().reset\_index()

*# Melt the data to make it easier to plot*

sales\_by\_year\_melted = pd.melt(sales\_by\_year, id\_vars='Year', value\_vars=columns, var\_name='Product', value\_name='Sales')

*# Create a bar chart*

plt.figure(figsize=(20,4)) sns.barplot(data=sales\_by\_year\_melted, x='Year', y='Sales',

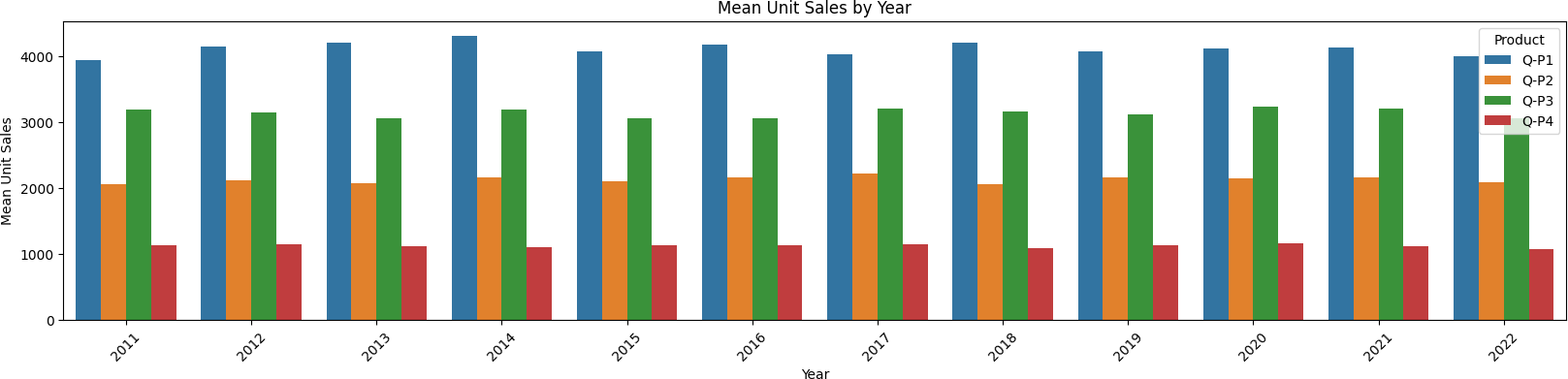
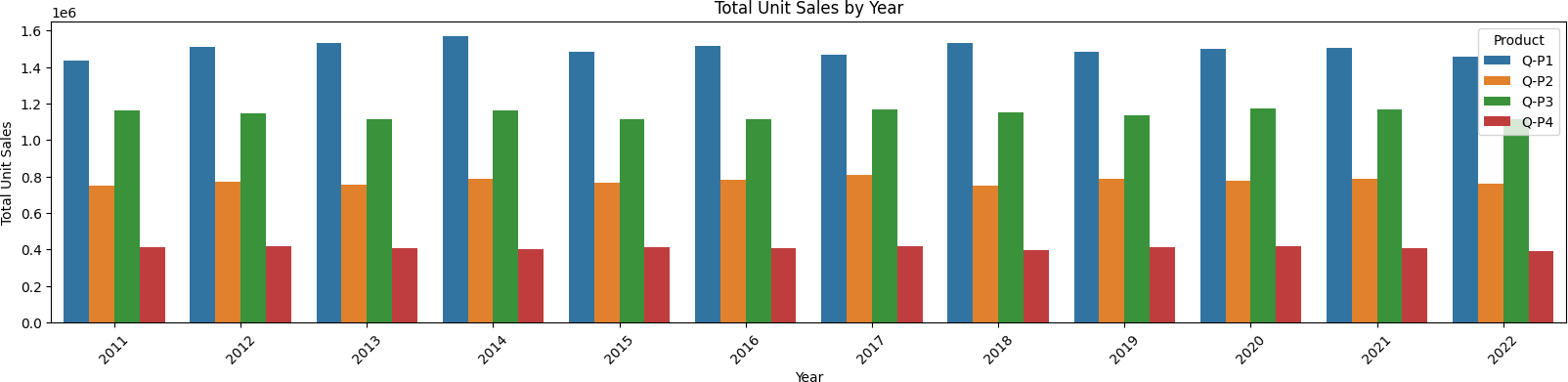
hue='Product') *#,palette="cividis")* plt.xlabel('Year') plt.ylabel(stri) plt.title(f'{stri} by {str1}') plt.xticks(rotation=45) plt.show()

INPUT[10]:

*#use the plot\_bar\_chart function, enter the Unit Sales Columns and the Unit Sales string*

plot\_bar\_chart(data\_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'],'Total Unit Sales', 'Year', 'sum')

plot\_bar\_chart(data\_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'],'Mean Unit Sales', 'Year', 'mean')



*# Import necessary libraries*

import pandas as pd

*# Load your dataset into a pandas DataFrame*

df = pd.read\_csv('/content/statsfinal.csv')

*# Drop duplicate rows*

df = df.drop\_duplicates()

*# Handle missing values*

df = df.dropna() *# Drop rows with any NaN values # OR fill missing values with a specific value*

*# df = df.fillna(value)*

*# Remove unwanted columns*

*# df = df.drop(columns=['Q-P1'])*

*# Convert data types if needed*

*# df['Q-P1'] = df['Q-P1'].astype('desired\_data\_type')*

*# Remove leading/trailing whitespaces from string columns # df['Q-P1'] = df['Q-P1'].str.strip()*

*# Perform other data cleaning operations as per your specific requirements*

*# Save the cleaned data to a new CSV file* df.to\_csv('cleaned\_data.csv', index=False) from google.colab import files

*# Assuming your cleaned data file is named 'cleaned\_data.csv' # Replace 'cleaned\_data.csv' with the actual file name if it's different*

files.download('cleaned\_data.csv')

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object> df.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 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

1. Q-P2 4600 non-null int64
2. Q-P3 4600 non-null int64
3. Q-P4 4600 non-null int64
4. S-P1 4600 non-null float64
5. S-P2 4600 non-null float64
6. S-P3 4600 non-null float64
7. S-P4 4600 non-null float64 dtypes: float64(4), int64(5), object(1) memory usage: 395.3+ KB

df.tail(3)

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

P2 \

4597 4597 01-02-2023 6289 3143 3588 474 19936.13

19926.62

4598 4598 02-02-2023 3122 1188 5899 517 9896.74

7531.92

4599 4599 03-02-2023 1234 3854 2321 406 3911.78

24434.36

|  |  |  |
| --- | --- | --- |
|  | S-P3 | S-P4 |
| 4597 | 19446.96 | 3379.62 |
| 4598 | 31972.58 | 3686.21 |
| 4599 | 12579.82 | 2894.78 |

df.head(3)

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

\

0 0 13-06-2010 5422 3725 576 907 17187.74 23616.50

1 1 14-06-2010 7047 779 3578 1574 22338.99 4938.86

2 2 15-06-2010 1572 2082 595 1145 4983.24 13199.88

S-P3 S-P4 0 3121.92 6466.91

1 19392.76 11222.62

2 3224.90 8163.85

df.describe()

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

\

count 4600.000000 4600.000000 4600.000000 4600.000000 4600.000000

mean 2299.500000 4121.849130 2130.281522 3145.740000 1123.500000

std 1328.049949 2244.271323 1089.783705 1671.832231 497.385676

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| min | 0.000000 | 254.000000 | 251.000000 | 250.000000 | 250.000000 |
|  |  |  |  |  |  |
| 25% | 1149.750000 | 2150.500000 | 1167.750000 | 1695.750000 | 696.000000 |
|  |  |  |  |  |  |
| 50% | 2299.500000 | 4137.000000 | 2134.000000 | 3202.500000 | 1136.500000 |
|  |  |  |  |  |  |
| 75% | 3449.250000 | 6072.000000 | 3070.250000 | 4569.000000 | 1544.000000 |
|  |  |  |  |  |  |
| max | 4599.000000 | 7998.000000 | 3998.000000 | 6000.000000 | 2000.000000 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | S-P1 | S-P2 | S-P3 | S-P4 |
| count | 4600.000000 | 4600.000000 | 4600.000000 | 4600.000000 |
| mean | 13066.261743 | 13505.984848 | 17049.910800 | 8010.555000 |
| std | 7114.340094 | 6909.228687 | 9061.330694 | 3546.359869 |
| min | 805.180000 | 1591.340000 | 1355.000000 | 1782.500000 |
| 25% | 6817.085000 | 7403.535000 | 9190.965000 | 4962.480000 |
| 50% | 13114.290000 | 13529.560000 | 17357.550000 | 8103.245000 |
| 75% | 19248.240000 | 19465.385000 | 24763.980000 | 11008.720000 |
| max | 25353.660000 | 25347.320000 | 32520.000000 | 14260.000000 |

df.isna().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.dropna(inplace=True) df.isna().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

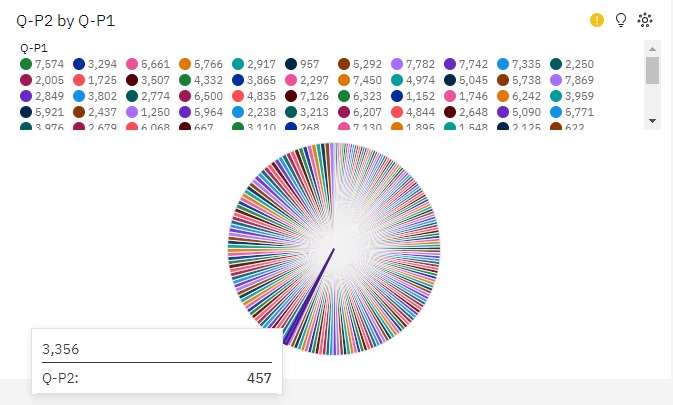
dtype: int64

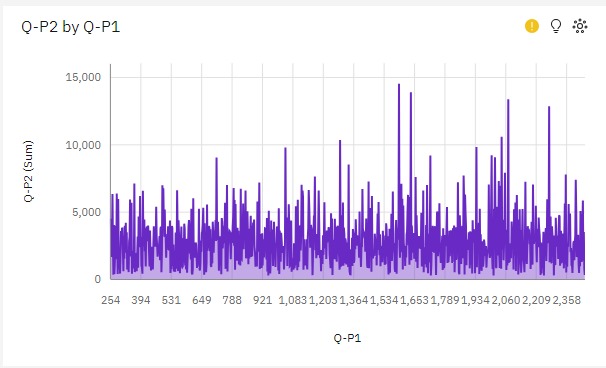
df[df.duplicated()]

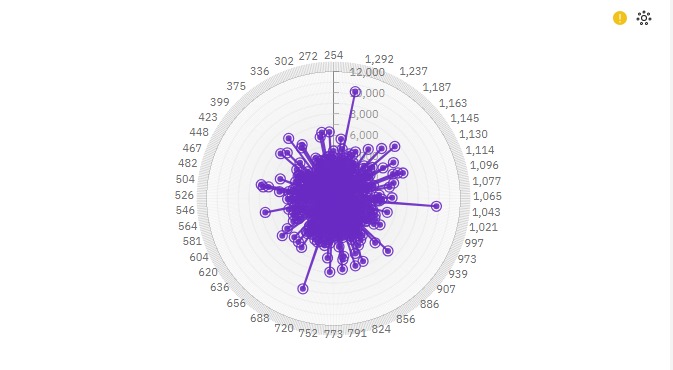
Empty DataFrame

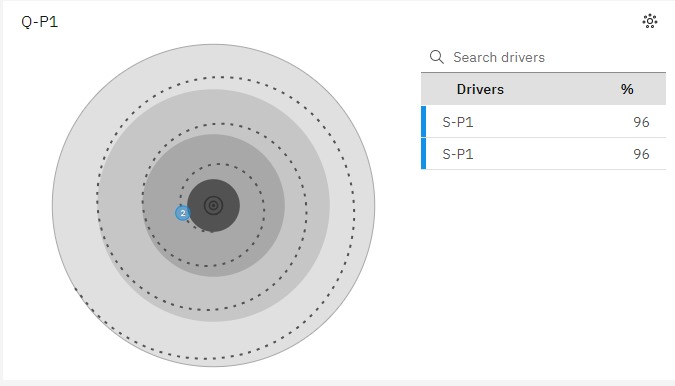
Columns: [Unnamed: 0, Date, Q-P1, Q-P2, Q-P3, Q-P4, S-P1, S-P2, S-P3, S-P4]

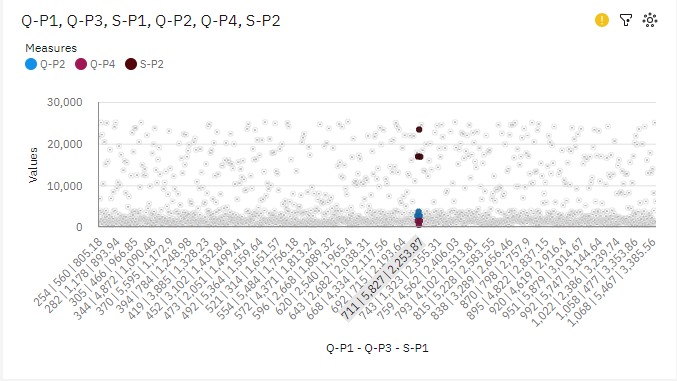
Index: []



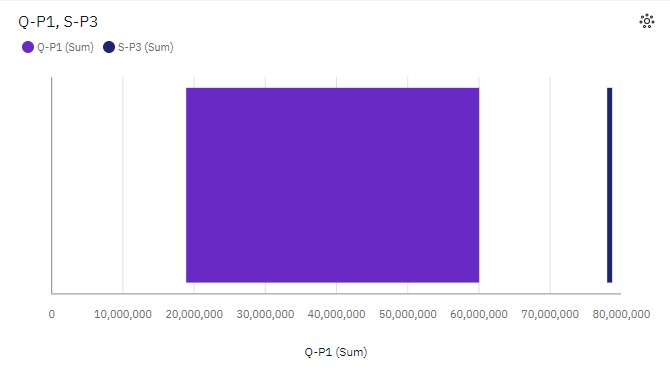


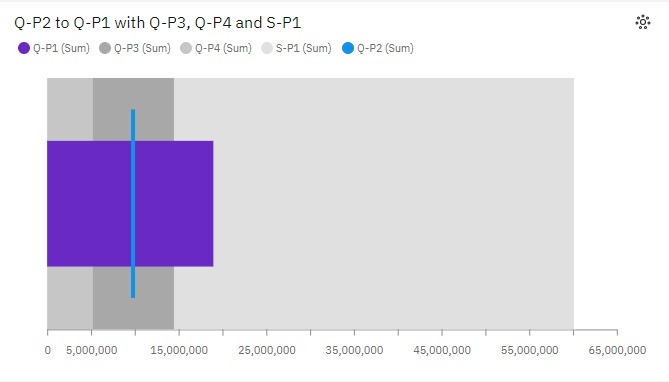


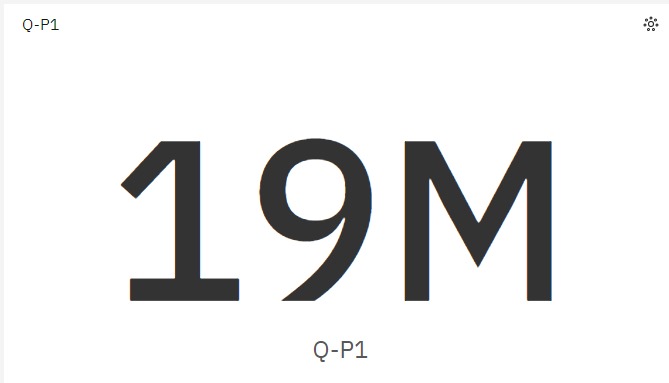












*# import the important packages*

import pandas as pd *# library used for data manipulation and analysis*

import numpy as np *# library used for working with arrays* import matplotlib.pyplot as plt *# library for plots and visualizations*

import seaborn as sns *# library for visualizations*

%matplotlib inline

*# To ignore warnings*

import warnings warnings.filterwarnings("ignore")

*#if you open in Kaggle editor*

data = pd.read\_csv('/content/statsfinal.csv')

*#if you open in juypter notebook*

*# data = pd.read\_csv('statsfinal.csv')*

*# Checking the first 5 and last 5 rows of the dataset*

data.head(-1)

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

P2 \

0 0 13-06-2010 5422 3725 576 907 17187.74

23616.50

1 1 14-06-2010 7047 779 3578 1574 22338.99

4938.86

2 2 15-06-2010 1572 2082 595 1145 4983.24

13199.88

3 3 16-06-2010 5657 2399 3140 1672 17932.69

15209.66

4 4 17-06-2010 3668 3207 2184 708 11627.56

20332.38

... ... ... ... ... ... ... ... .

..

4594 4594 29-01-2023 1227 3044 5510 1896 3889.59

19298.96

4595 4595 30-01-2023 2476 3419 525 1359 7848.92

21676.46

4596 4596 31-01-2023 7446 841 4825 1311 23603.82

5331.94

4597 4597 01-02-2023 6289 3143 3588 474 19936.13

19926.62

4598 4598 02-02-2023 3122 1188 5899 517 9896.74

7531.92

|  |  |  |
| --- | --- | --- |
|  | S-P3 | S-P4 |
| 0 | 3121.92 | 6466.91 |
| 1 | 19392.76 | 11222.62 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 2 | 3224.90 | 8163.85 |
| 3 | 17018.80 | 11921.36 |
| 4 | 11837.28 | 5048.04 |
| ... | ... | ... |
| 4594 | 29864.20 | 13518.48 |
| 4595 | 2845.50 | 9689.67 |
| 4596 | 26151.50 | 9347.43 |
| 4597 | 19446.96 | 3379.62 |
|  | 4598 | 31972.58 | 3686.21 |

[4599 rows x 10 columns] data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 4600 entries, 0 to 4599 Data columns (total 9 columns):

# Column Non-Null Count Dtype

0 Date 4600 non-null object

1. Q-P1 4600 non-null int64
2. Q-P2 4600 non-null int64
3. Q-P3 4600 non-null int64
4. Q-P4 4600 non-null int64
5. S-P1 4600 non-null float64
6. S-P2 4600 non-null float64
7. S-P3 4600 non-null float64
8. S-P4 4600 non-null float64 dtypes: float64(4), int64(4), object(1) memory usage: 323.6+ KB

data.isnull().sum() 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

*# Extract year from the 'Day' 'Month' 'year' from the 'Date' column using a lambda function*

*# We need to get the year from the data to analyse sales year to year*

data['Day'] = data['Date'].apply(lambda x: x.split('-')[0])

data['Month'] = data['Date'].apply(lambda x: x.split('-')[1])

data['Year'] = data['Date'].apply(lambda x: x.split('-')[2]) data

Date Q-P1 Q-P2 Q-P3 Q-P4 S-P1 S-P2 S-P3

\

0 13-06-2010 5422 3725 576 907 17187.74 23616.50 3121.92

1 14-06-2010 7047 779 3578 1574 22338.99 4938.86 19392.76

2 15-06-2010 1572 2082 595 1145 4983.24 13199.88 3224.90

3 16-06-2010 5657 2399 3140 1672 17932.69 15209.66 17018.80

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

... ... ... ... ... ... ... ... ...

4595 30-01-2023 2476 3419 525 1359 7848.92 21676.46 2845.50

4596 31-01-2023 7446 841 4825 1311 23603.82 5331.94 26151.50

4597 01-02-2023 6289 3143 3588 474 19936.13 19926.62 19446.96

4598 02-02-2023 3122 1188 5899 517 9896.74 7531.92 31972.58

4599 03-02-2023 1234 3854 2321 406 3911.78 24434.36 12579.82

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | S-P4 | Day | Month | Year |
| 0 | 6466.91 | 13 | 06 | 2010 |
| 1 | 11222.62 | 14 | 06 | 2010 |
| 2 | 8163.85 | 15 | 06 | 2010 |
| 3 | 11921.36 | 16 | 06 | 2010 |
| 4 | 5048.04 | 17 | 06 | 2010 |
| ... | ... | .. | ... | ... |
| 4595 | 9689.67 | 30 | 01 | 2023 |
| 4596 | 9347.43 | 31 | 01 | 2023 |
| 4597 | 3379.62 | 01 | 02 | 2023 |
| 4598 | 3686.21 | 02 | 02 | 2023 |
| 4599 | 2894.78 | 03 | 02 | 2023 |

[4600 rows x 12 columns]

data\_reduced = data.query("Year != '2010' and Year != '2023'")

*#Create a function that allows us to plot a bar chart for the 4 products*

def plot\_bar\_chart(df, columns, stri, str1, val):

*# Aggregate sales for each product by year, by sum or mean*

if val == 'sum':

sales\_by\_year = df.groupby('Year')

[columns].sum().reset\_index() elif val == 'mean':

sales\_by\_year = df.groupby('Year') [columns].mean().reset\_index()

*# Melt the data to make it easier to plot*

sales\_by\_year\_melted = pd.melt(sales\_by\_year, id\_vars='Year', value\_vars=columns, var\_name='Product', value\_name='Sales')

*# Create a bar chart*

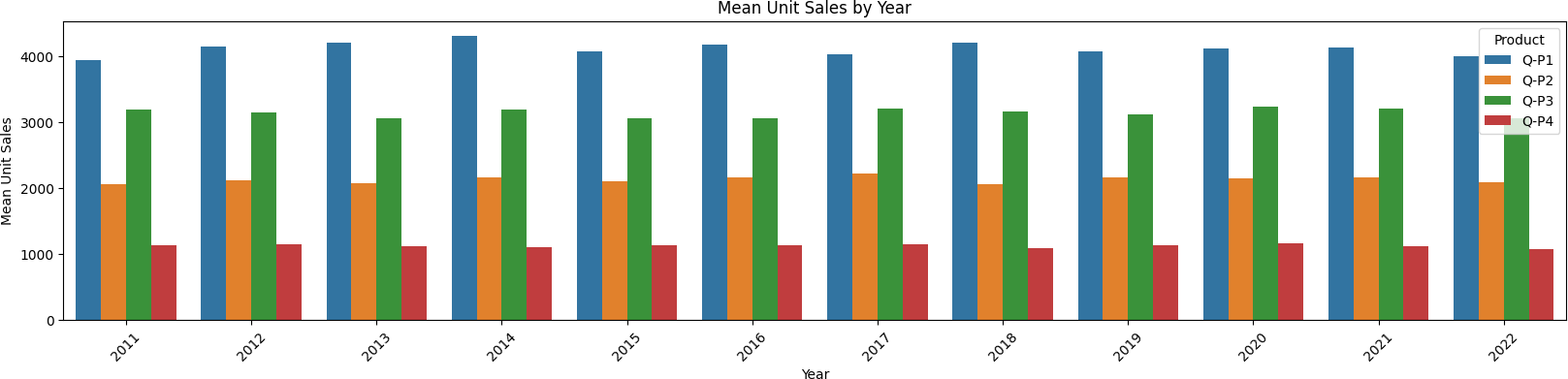
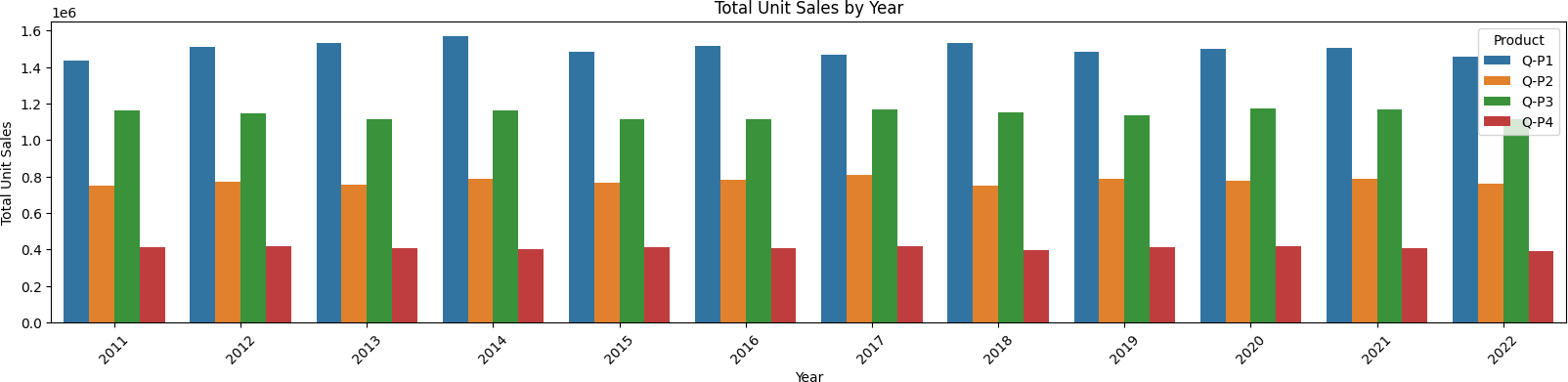
plt.figure(figsize=(20,4)) sns.barplot(data=sales\_by\_year\_melted, x='Year', y='Sales',

hue='Product') *#,palette="cividis")* plt.xlabel('Year') plt.ylabel(stri) plt.title(f'{stri} by {str1}') plt.xticks(rotation=45) plt.show()

*#use the plot\_bar\_chart function, enter the Unit Sales Columns and the Unit Sales string*

plot\_bar\_chart(data\_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'],'Total Unit Sales', 'Year', 'sum')

plot\_bar\_chart(data\_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'],'Mean Unit Sales', 'Year', 'mean')

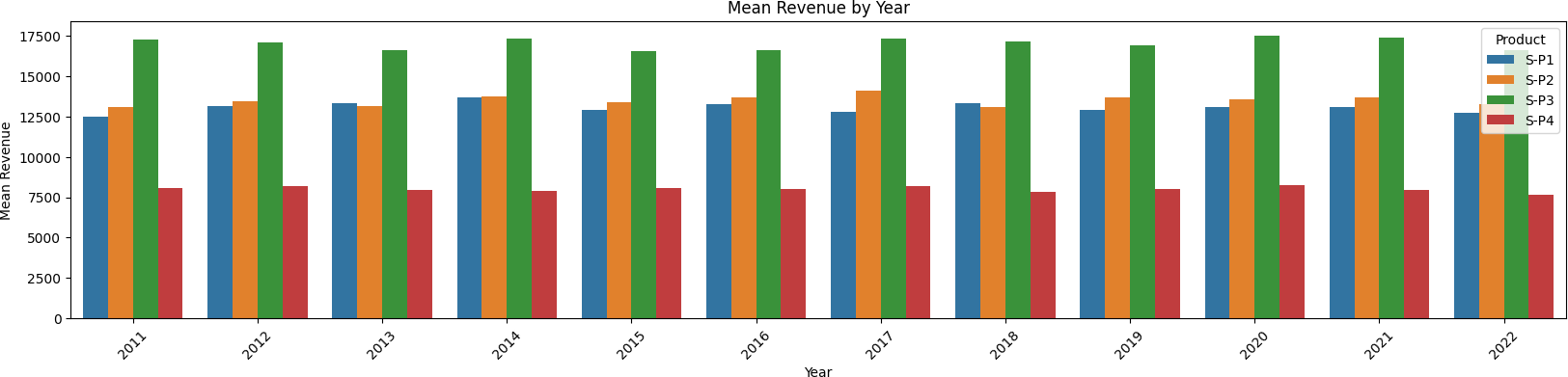
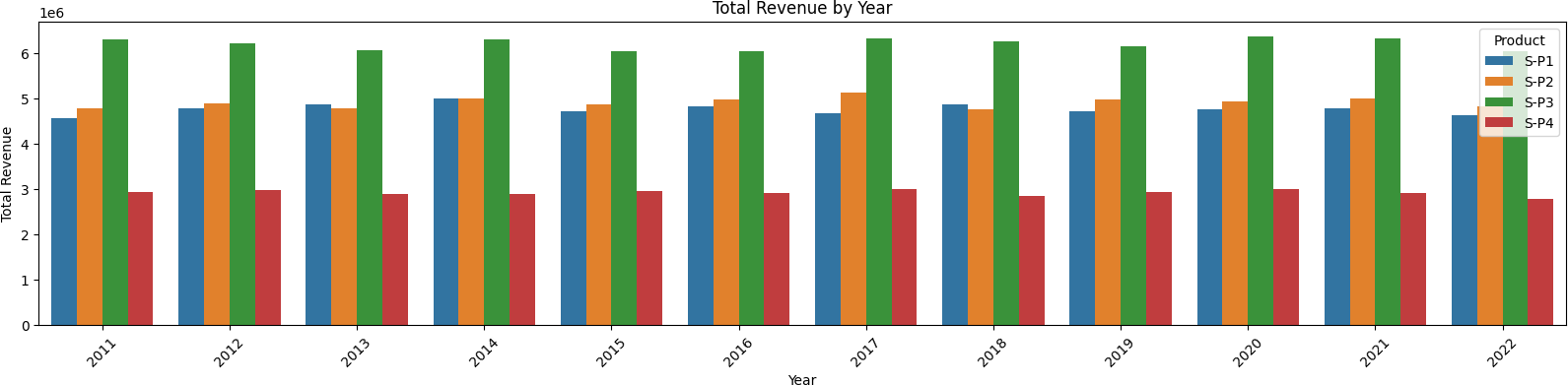


*#use the plot\_bar\_chart function, enter the Revenue Columns and the Revenue string*

plot\_bar\_chart(data\_reduced, ['S-P1', 'S-P2', 'S-P3', 'S-P4'], 'Total

Revenue', 'Year', 'sum')

plot\_bar\_chart(data\_reduced, ['S-P1', 'S-P2', 'S-P3', 'S-P4'], 'Mean Revenue', 'Year', 'mean')



data

Date Q-P1 Q-P2 Q-P3 Q-P4 S-P1 S-P2 S-P3

\

0 13-06-2010 5422 3725 576 907 17187.74 23616.50 3121.92

1 14-06-2010 7047 779 3578 1574 22338.99 4938.86 19392.76

2 15-06-2010 1572 2082 595 1145 4983.24 13199.88 3224.90

3 16-06-2010 5657 2399 3140 1672 17932.69 15209.66 17018.80

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

... ... ... ... ... ... ... ... ...

4595 30-01-2023 2476 3419 525 1359 7848.92 21676.46 2845.50

4596 31-01-2023 7446 841 4825 1311 23603.82 5331.94 26151.50

4597 01-02-2023 6289 3143 3588 474 19936.13 19926.62 19446.96

4598 02-02-2023 3122 1188 5899 517 9896.74 7531.92 31972.58

4599 03-02-2023 1234 3854 2321 406 3911.78 24434.36 12579.82

[4600 rows x 12 columns]

*# Create a figure and axis*

def month\_plot():

fig, ax = plt.subplots()

*# Plot the sales data for each product by month*

data\_reduced.groupby('Month')[['Q-P1', 'Q-P2', 'Q-P3', 'Q- P4']].sum().plot(ax=ax)

*# Set the x-axis limits to only show up to December*

ax.set\_xlim(left=0, right=13)

*# Set the axis labels and title* ax.set\_xlabel('Month') ax.set\_ylabel('Total unit sales')

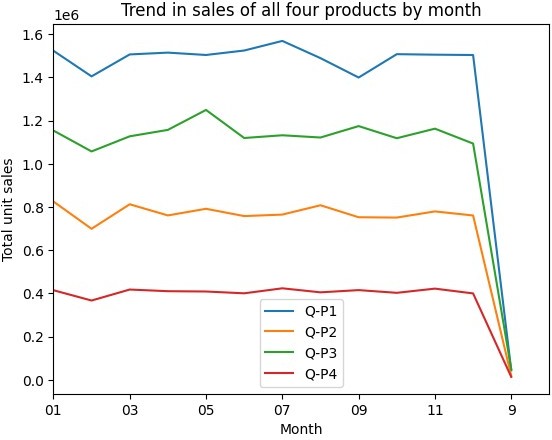
ax.set\_title('Trend in sales of all four products by month')

*# Show the plot*

plt.show()

month\_plot()

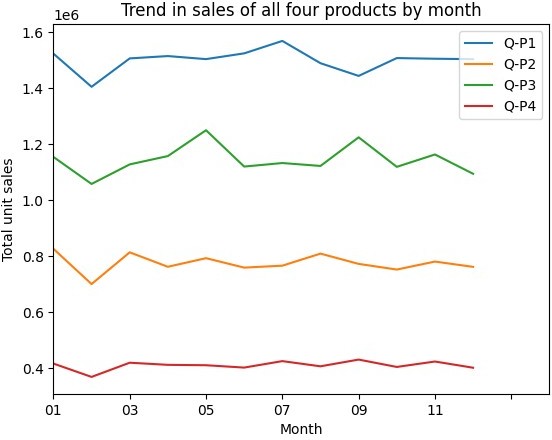
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | S-P4 | Day | Month | Year |
| 0 | 6466.91 | 13 | 06 | 2010 |
| 1 | 11222.62 | 14 | 06 | 2010 |
| 2 | 8163.85 | 15 | 06 | 2010 |
| 3 | 11921.36 | 16 | 06 | 2010 |
| 4 | 5048.04 | 17 | 06 | 2010 |
| ... | ... | .. | ... | ... |
| 4595 | 9689.67 | 30 | 01 | 2023 |
| 4596 | 9347.43 | 31 | 01 | 2023 |
| 4597 | 3379.62 | 01 | 02 | 2023 |
| 4598 | 3686.21 | 02 | 02 | 2023 |
| 4599 | 2894.78 | 03 | 02 | 2023 |



*# Replace all entries of '9' in the Month column with '09'*

data\_reduced['Month'] = data['Month'].replace('9', '09')

month\_plot()



*#get the 31st day for each month in each year. Note: not every month has 31 days*

def month\_31\_data(df, months):

m31\_data = df[df['Month'].isin(months) & (df['Day'] == '31')] return m31\_data

\_31\_months = month\_31\_data(data\_reduced, ['01', '02', '03', '04',

'05', '06', '07', '08', '09', '10', '11', '12'])

\_31\_months

Date Q-P1 Q-P2 Q-P3 Q-P4 S-P1 S-P2 S-P3

\

231 31-01-2011 939 3325 1863 1612 2976.63 21080.50 10097.46

290 31-03-2011 464 2220 421 1663 1470.88 14074.80 2281.82

351 31-05-2011 1507 2980 3816 1202 4777.19 18893.20 20682.72

412 31-07-2011 4336 744 4717 667 13745.12 4716.96 25566.14

442 31-08-2011 4548 1484 1596 1974 14417.16 9408.56 8650.32

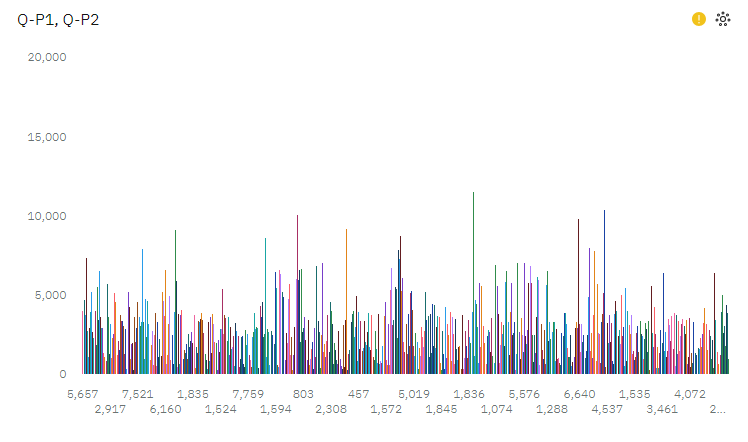
... ... ... ... ... ... ... ... ...

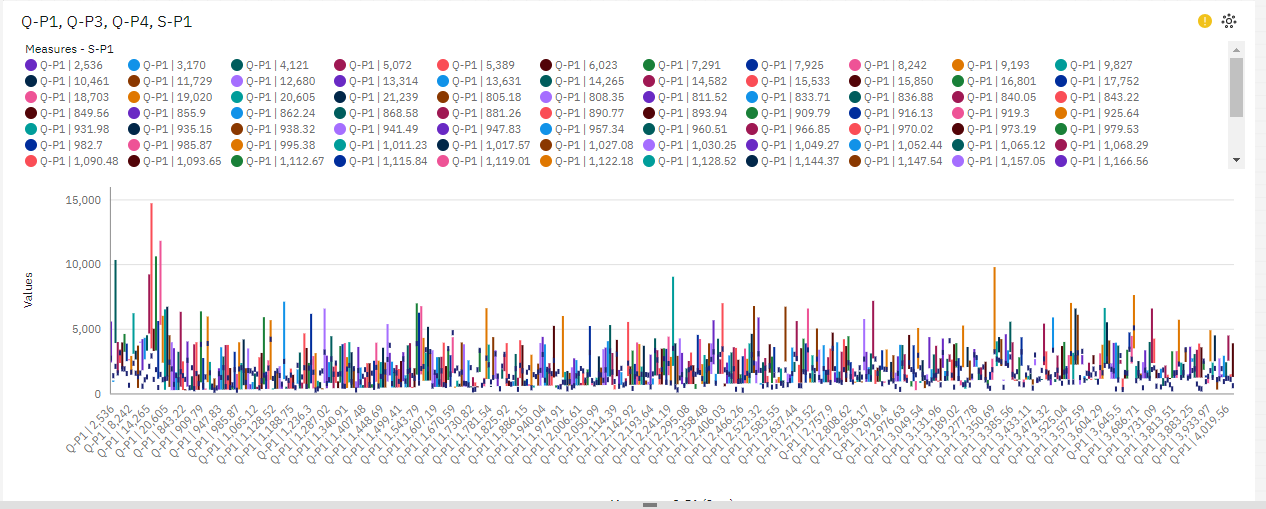
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
| 4352 | 31-05-2022 | 3669 | 2710 | 3067 | 1593 | 11630.73 | 17181.40 | 16623.14 |
|  |  |  |  |  |  |  |  |  |
| 4413 | 31-07-2022 | 1437 | 833 | 1867 | 1270 | 4555.29 | 5281.22 | 10119.14 |
|  |  |  |  |  |  |  |  |  |
| 4443 | 31-08-2022 | 1035 | 1639 | 3658 | 841 | 3280.95 | 10391.26 | 19826.36 |
|  |  |  |  |  |  |  |  |  |
| 4474 | 31-9-2022 | 6964 | 1873 | 5481 | 1336 | 22075.88 | 11874.82 | 29707.02 |
|  |  |  |  |  |  |  |  |  |  |
|  | 4535 | 31-11-2022 | 4600 | 2006 | 3796 | 1426 | 14582.00 | 12718.04 | 20574.32 |

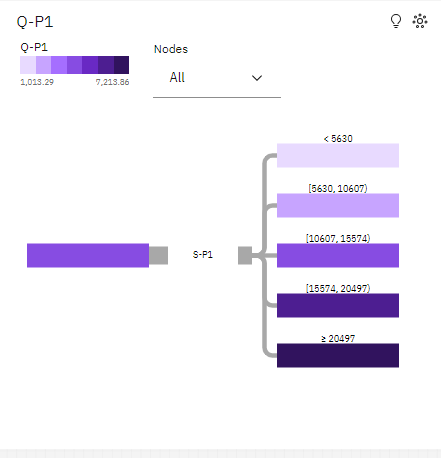
[84 rows x 12 columns]

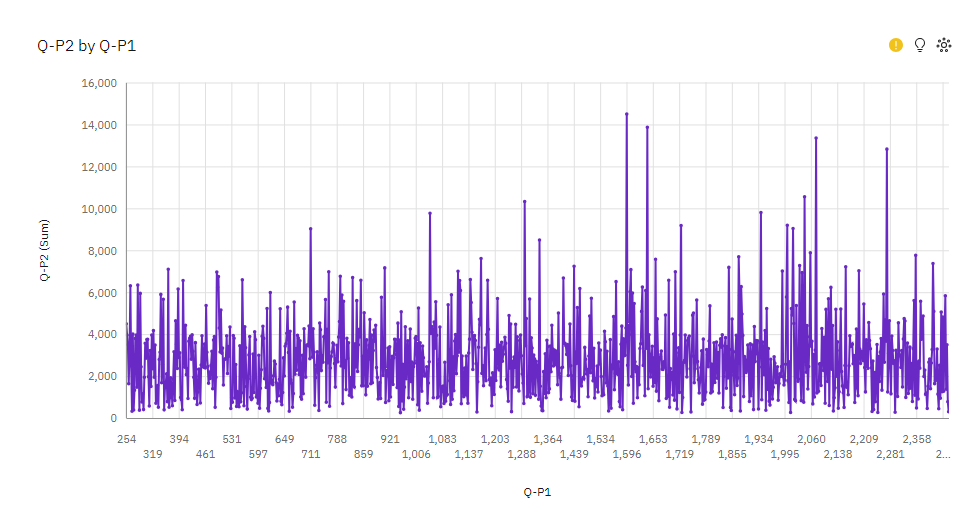
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | S-P4 | Day | Month | Year |
| 231 | 11493.56 | 31 | 01 | 2011 |
| 290 | 11857.19 | 31 | 03 | 2011 |
| 351 | 8570.26 | 31 | 05 | 2011 |
| 412 | 4755.71 | 31 | 07 | 2011 |
| 442 | 14074.62 | 31 | 08 | 2011 |
| ... | ... | .. | ... | ... |
| 4352 | 11358.09 | 31 | 05 | 2022 |
| 4413 | 9055.10 | 31 | 07 | 2022 |
| 4443 | 5996.33 | 31 | 08 | 2022 |
| 4474 | 9525.68 | 31 | 09 | 2022 |
| 4535 | 10167.38 | 31 | 11 | 2022 |

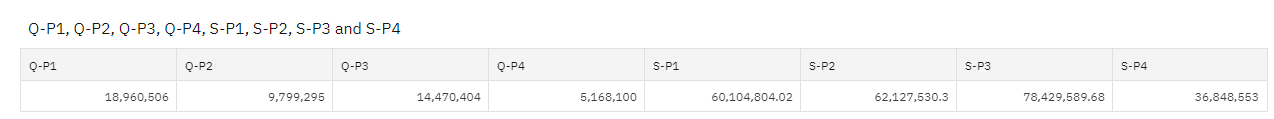
* **SALE TREND PATTERNS USING IBM COGNOS ANALYTICS:**

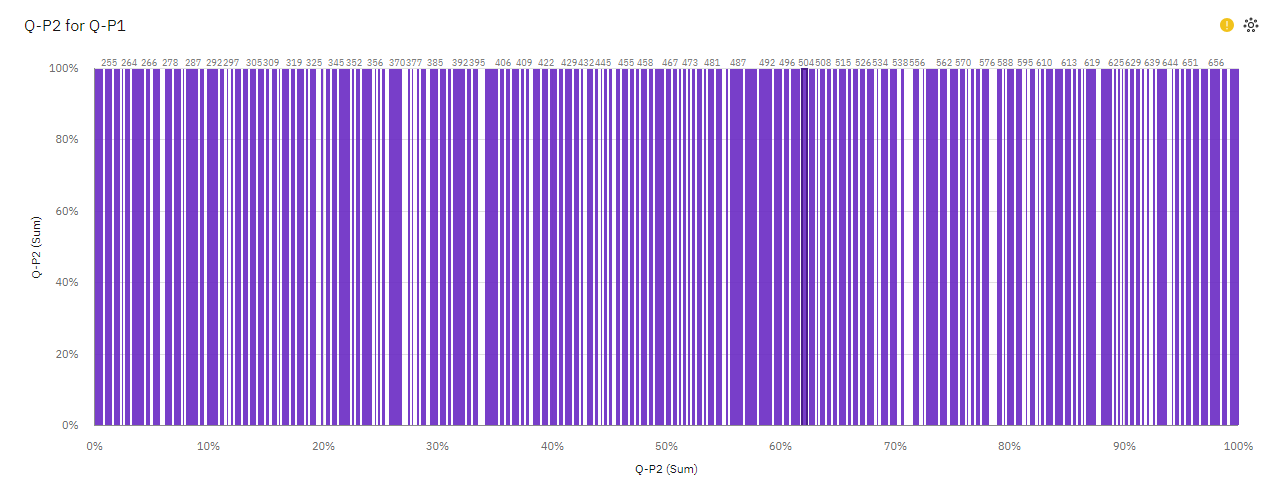












**CONCLUSION:**

The "Product Sales Analysis" project has provided valuable insights into the world of data-driven decision-making in the context of product sales. Through comprehensive data analysis, we have uncovered key trends and drawn significant conclusions that can inform and transform business strategies.

One of the major takeaways from this project is the importance of data preprocessing and exploratory data analysis. Cleaning and organizing the data, as well as identifying key attributes, lay the foundation for more in-depth analysis. Visualizations and initial explorations allowed us to understand the overall sales patterns and identify the strengths and weaknesses within the dataset.

Moreover, advanced analysis techniques, including trend analysis, seasonality detection, and forecasting, unveiled hidden patterns and insights. This information is crucial for businesses to make informed decisions about product inventory, marketing campaigns, and resource allocation. By understanding when and how sales peak, organizations can optimize their operations and capitalize on opportunities.

The project also highlighted the significance of considering different dimensions in data analysis, such as product categories, customer segments, and geographic regions. This level of granularity allows for a more nuanced understanding of sales performance and provides a basis for targeted strategies.

In conclusion, the "Product Sales Analysis" project reinforces the idea that data analysis is an indispensable tool for businesses striving for success in today's competitive market. It demonstrates that informed decisions, rooted in data, can lead to improved sales performance, increased customer satisfaction, and a more efficient allocation of resources. By harnessing the power of data, businesses can adapt to changing market conditions, enhance their strategic decision-making, and ultimately achieve sustainable growth and success.