***PRODUCT SALES ANALYSIS***

***PHASE 4***

***NAME: NAGENTHIRA KUMAR D***

***NM\_ID: au211521106104***

***Introduction:***

In the ever-evolving landscape of modern business, data has become the lifeblood of informed decision-making. The ability to transform raw data into actionable insights has become a critical component of success. To this end, our organization has harnessed the immense capabilities of IBM Cognos, a sophisticated business intelligence and data analytics tool. In this comprehensive essay, we delve into our mission to utilize IBM Cognos to design interactive dashboards and reports, unveiling crucial insights into our business operations. These insights encompass a spectrum of crucial aspects, including the identification of top-selling products, the analysis of sales trends, and the unveiling of intricate customer preferences.

***1. Business Intelligence as a Necessity***

In the digital age, business intelligence is not merely a luxury; it is an absolute necessity. The vast pools of data at our disposal contain a wealth of information that can drive our organization to new heights of success. To tap into this potential, we have employed IBM Cognos, a cutting-edge tool that empowers us to transform raw data into actionable insights.

***2. The Holy Grail: Top-Selling Products***

Among our primary goals is the identification of top-selling products within our portfolio. These are the products that resonate most with our customer base, driving substantial revenue and profit. By using Cognos to visualize and analyze sales data, we can pinpoint these high performers with precision.

***3. Navigating the Sales Landscape: Trends Analysis***

Sales trends are not static; they evolve over time. Understanding these patterns is essential for maintaining a competitive edge. Through IBM Cognos, we can embark on a deep dive into the historical data, discerning fluctuations and growth trajectories, and paving the way for strategic planning.

***4. Seizing the Right Moments: Peak Sales Periods***

Timing is everything in business. Identifying peak sales periods empowers us to make timely adjustments to our strategies. IBM Cognos equips us with the tools to recognize when our products experience peak demand, facilitating more efficient inventory management, staffing, and marketing campaigns.

***5. Customer Preferences: A Treasure Trove of Insights***

In the customer-centric world we operate in, understanding customer preferences is akin to holding the keys to success. Cognos allows us to delve into the intricate world of customer choices. We can discern which products are favored, what features matter most, and even the preferred channels of purchase.

***6. Segmentation for Targeted Strategies***

Our approach with IBM Cognos goes beyond a broad stroke. We can segment the insights obtained based on various criteria such as product categories, geographic regions, or specific customer demographics. This level of granularity is invaluable for crafting highly targeted strategies.

***7. Real-time Analytics for Agile Decision-Making***

The ability to analyze data in real-time is one of the distinctive features of IBM Cognos. This capability allows us to react swiftly to changing market conditions and evolving customer behaviors. It ensures that our decision-making remains agile and responsive.

***8. Insights that Drive Action***

Data and insights are only as valuable as the actions they drive. The insights derived from our Cognos-driven dashboards and reports are not mere statistics but tools for decision-making. They empower our team to make informed decisions, optimize our operations, and refine our strategies.

***9. Competitive Advantage through Informed Decisions***

In the fiercely competitive marketplace, staying ahead of the curve is essential. Utilizing IBM Cognos enables us to maintain a competitive edge by interpreting data trends, understanding customer preferences, and aligning our offerings with the ever-evolving market landscape.

***10. Driving Growth, Efficiency, and Success***

In conclusion, our adoption of IBM Cognos to design interactive dashboards and reports signifies our commitment to data-driven success. By uncovering top-selling products, analyzing sales trends, and delving into customer preferences, we aim to drive growth, enhance operational efficiency, and achieve unparalleled business success. This journey is marked by the seamless fusion of technology, data, and actionable insights, which empower us to navigate the intricate landscape of modern business with confidence and competence. Stay tuned as we embark on this transformative journey into the world of data-driven decisions.

*# 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*

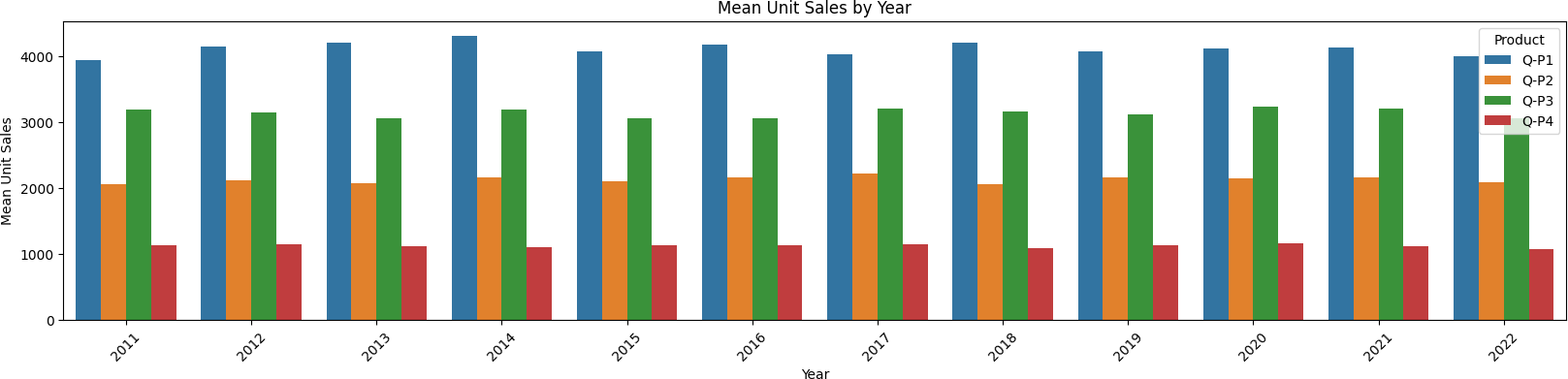
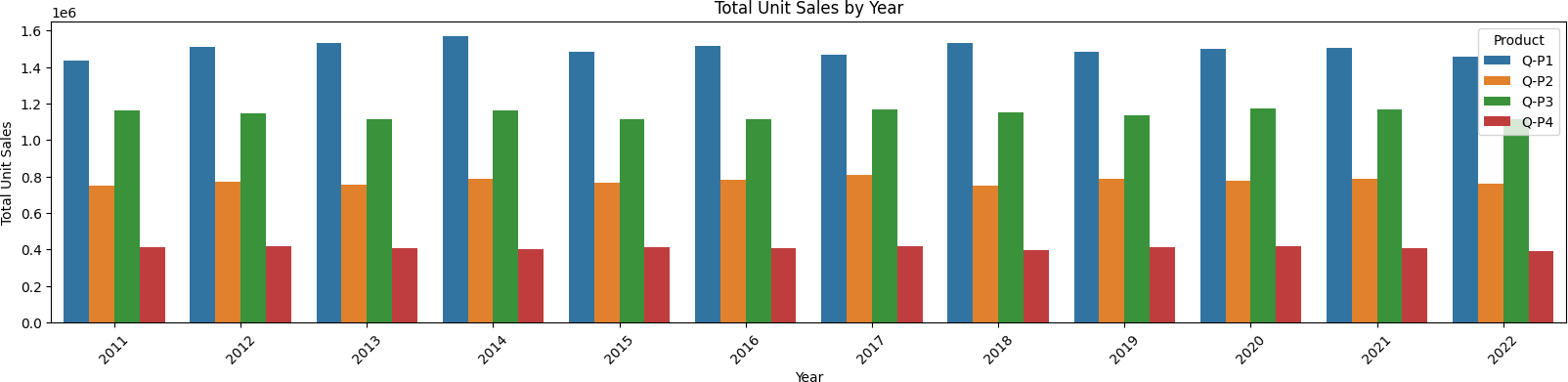
plt.figure(figsize=(20,4)) sns.barplot(data=sales\_by\_year\_melted, x='Year', y='Sales',

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')

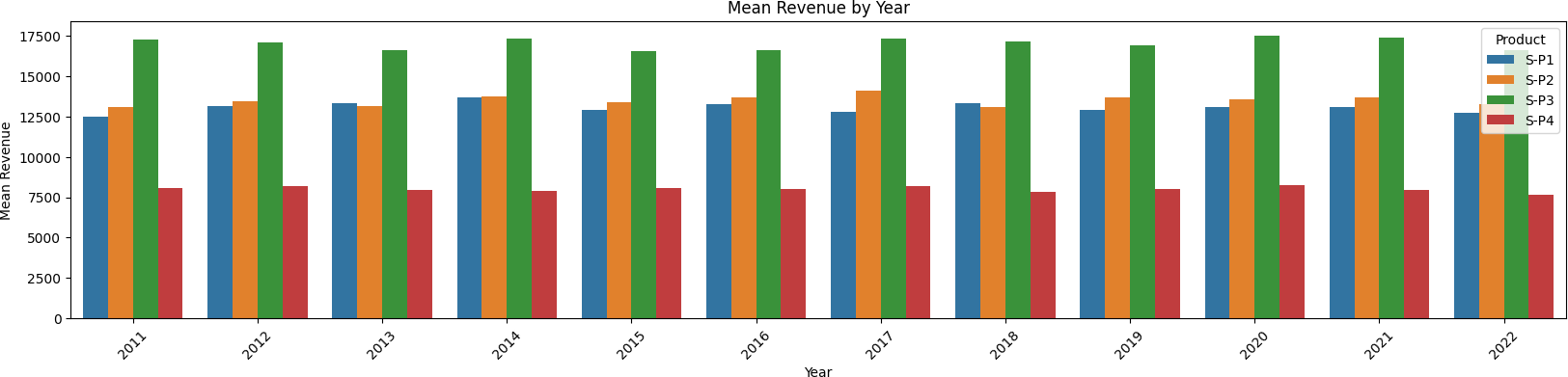
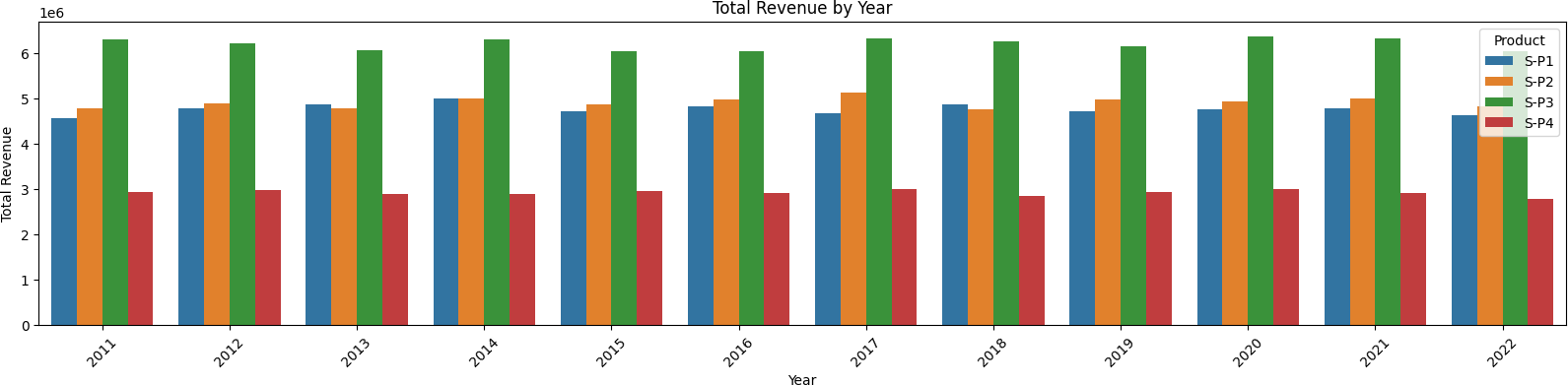


*#use the plot\_bar\_chart function, enter the Revenue Columns and the Revenue string*

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')

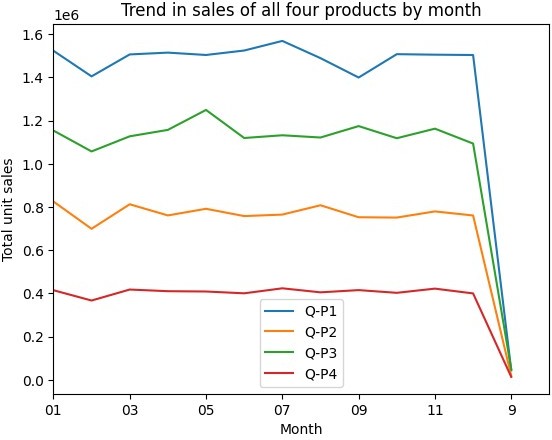
ax.set\_title('Trend in sales of all four products by month')

*# Show the plot*

plt.show()

month\_plot()

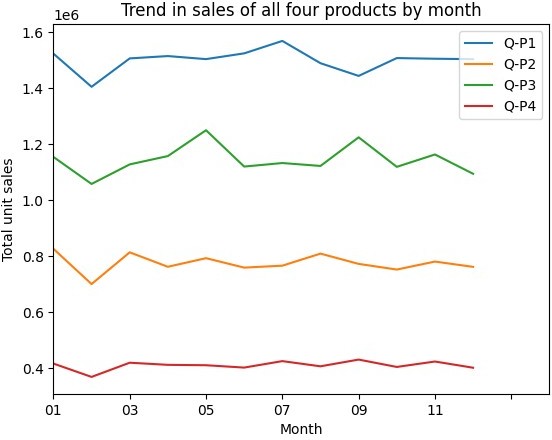
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | S-P4 | Day | Month | Year |
| 0 | 6466.91 | 13 | 06 | 2010 |
| 1 | 11222.62 | 14 | 06 | 2010 |
| 2 | 8163.85 | 15 | 06 | 2010 |
| 3 | 11921.36 | 16 | 06 | 2010 |
| 4 | 5048.04 | 17 | 06 | 2010 |
| ... | ... | .. | ... | ... |
| 4595 | 9689.67 | 30 | 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:**

