PRODUCT SALES ANALYSIS

Data Analytics with cognos – Phase 3

DOCUMENTATION

Team Members:

1.R.AARTHY(au613021205001)

2.G.KANIMOZHI (au613021205021)

3.V.SRINITHI (au613021205052)

4.L.ELAKKIYA(au613021205011)

5.N.THIRSHA au613021205058)

Problem Definition:

Start the data analysis by loading and preprocessing the dataset. Load the dataset using python and data manipulation libraries (e.g., pandas).

Dataset Link:

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

Overview of the process:

1.Import Libraries:

Begin by importing the necessary libraries, such as pandas for data manipulation.

2.Load the Dataset:

Use pd.read_csv() or other appropriate methods to load your dataset into a pandas DataFrame.

3.Explore the Dataset:

Display the initial rows, check for missing values, and explore basic statistics to understand the structure and content of the data.

4.Handle Missing Values:

Decide on an appropriate strategy for dealing with missing values, such as dropping rows or filling values based on a specific strategy.

5.Additional Preprocessing Steps:

Depending on the nature of your data, consider additional preprocessing steps such as feature scaling, handling outliers, processing date-time features, dealing with text data, feature engineering, or discretization.

6.Save Preprocessed Dataset (Optional):

Save the preprocessed dataset to a new file if significant changes have been made.

Loading the dataset:

1.*Importing libraries*

Here, for preprocessing the dataset and manipulate the data, pandas is the library used to frame the data.

Code:

import pandas as pd

2.*Loading the dataset*

In this step, we are framing the data into the table using DataFrame in pandas, and display the head or 5 rows of the dataset.

Code:

Replace with the actual filename

file_path= "C:/Users/91962/Documents/phase3.csv"

df = pd.read_csv(file_path)

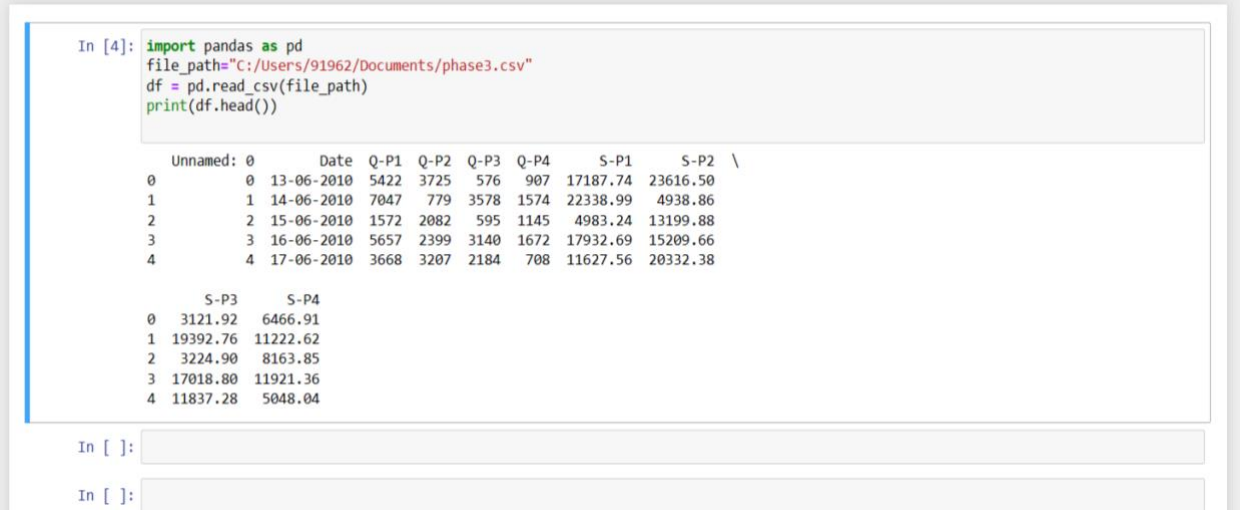
Preprocessing the dataset

3.*Explore the dataset:*

After framing data, the first few or five rows of the data in displayed using the head() function.

Code:

print(df.head())



```
In [4]: import pandas as pd
        file_path="C:/Users/91962/Documents/phase3.csv"
        df = pd.read_csv(file_path)
        print(df.head())

           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

              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

In [ ]:

In [ ]:
```

OUTPUT:

```
   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

      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
                                                              In [ ]:
```

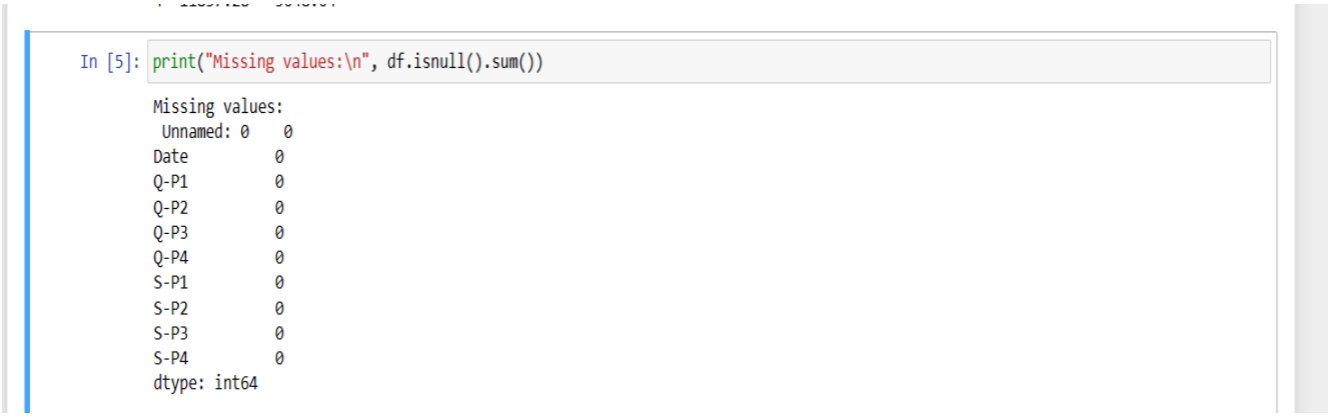
4.Check for missing values:

In this step, the missing values or null values, if it present in the data are separated and number of null values are shown through this code.

Code:

print("Missing values:\n", df.isnull().sum())

Output:




```
Missing values:
 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
```

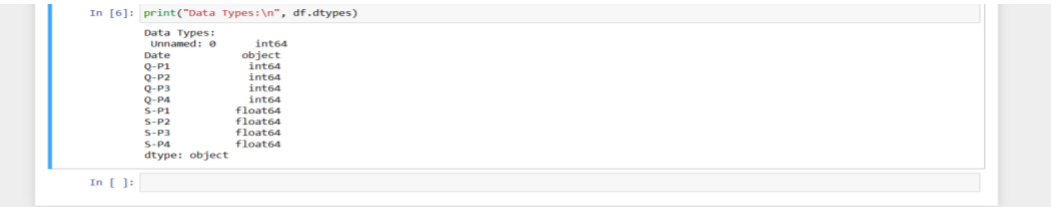
5.Check datatype:

In this step, the data type of the columns are discussed

Code:

print("Data Types:\n", df.dtypes)

OUTPUT




```
Data Types:
 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
```

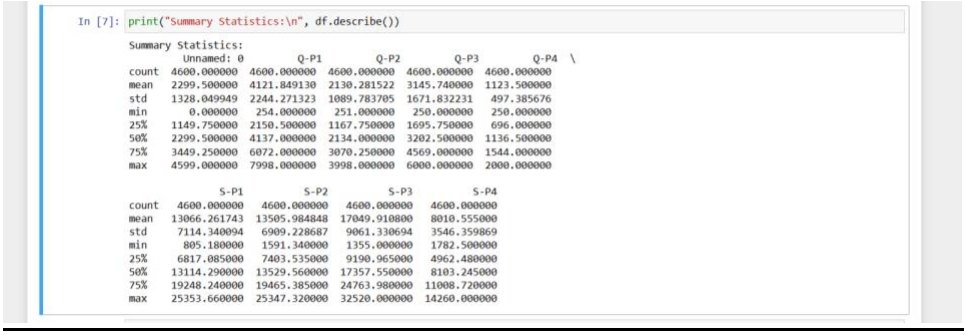
6.Check basic statistics:

The statistics of the columns such as count, mean, std, min, max, 25%, 50%, 75% are shown through the describe() function command.

Code:

print("Summary Statistics:\n", df.describe())

OUTPUT:




```
Summary Statistics:
         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

7.*Additional Preprocessing steps:*

Perform any other preprocessing steps that are specific to your dataset a nd analysis goals. This may include scaling numeric features, handling outliers, or creating new features.

8.*Saving Preprocessed dataset:*

In this step, if we made substantial changes to the dataset and want to sa ve the preprocessed version, you can use the following Code.

Code:

```
# Save the preprocessed dataset to a new CSV file
df.to_csv('preprocessed_dataset.csv', index=False)
```

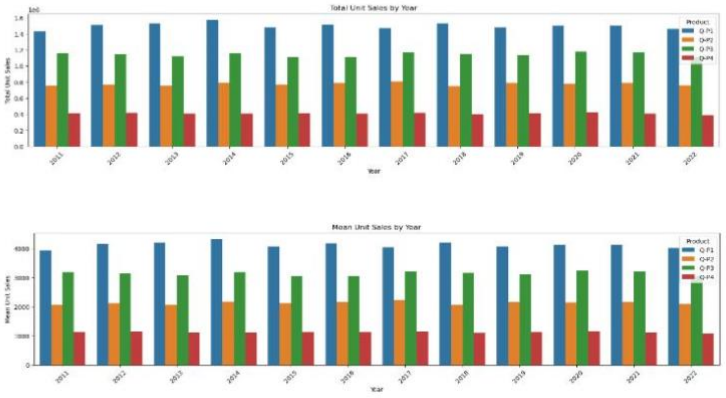
Data virtualization

Graph our TOTAL & MEAN unit sold for each product using a histogram.

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()
```

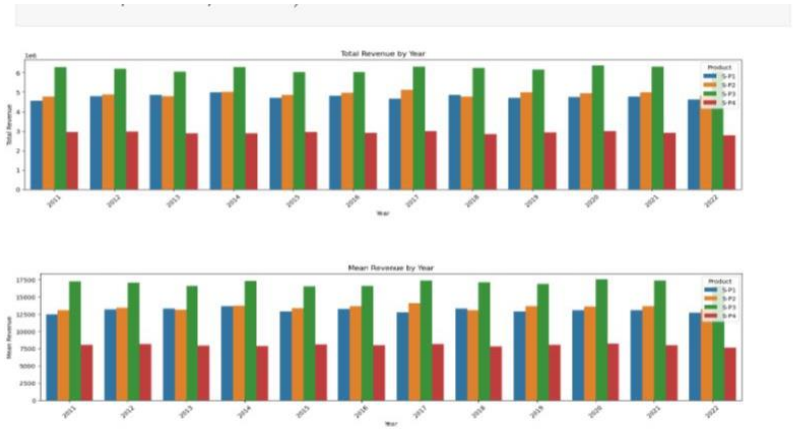
```
    Elif val == 'mean':
        Sales_by_year = df.groupby('Year')[columns].mean().reset_index()

    # 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="ci
vidis")
    Plt.xlabel('Year')
    Plt.ylabel(stri)
    Plt.title(f'{stri} by {str1}')
    Plt.xticks(rotation=45)
    Plt.show()

Plot_bar_chart(data_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'],'Total Unit Sales', 'Year', 'su
m')

Plot_bar_chart(data_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'],'Mean Unit Sales', 'Year', '
mean')
```



Graph our TOTAL & MEAN revenue of sales for each product u sing a historgram.

```
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’, ‘mea
n’)



Data

In [12]:
```
data
```

Out[12]:

	Date	Q-P1	Q-P2	Q-P3	Q-P4	S-P1	S-P2	S-P3	S-P4	Day	Month
0	13-06-2010	5422	3725	576	907	17187.74	23616.50	3121.92	6466.91	13	06
1	14-06-2010	7047	779	3578	1574	22338.99	4938.86	19392.76	11222.62	14	06
2	15-06-2010	1572	2082	595	1145	4983.24	13199.88	3224.90	8163.85	15	06
3	16-06-2010	5657	2399	3140	1672	17932.69	15209.66	17018.80	11921.36	16	06
4	17-06-2010	3668	3207	2184	708	11627.56	20332.38	11837.28	5048.04	17	06
...	...	...	...	...	...	...	...	...	...	...	...
4595	30-01-2023	2476	3419	525	1359	7848.92	21676.46	2845.50	9689.67	30	01
4596	31-01-2023	7446	841	4825	1311	23603.82	5331.94	26151.50	9347.43	31	01
4597	01-02-2023	6289	3143	3588	474	19936.13	19926.62	19446.96	3379.62	01	02
4598	02-02-2023	3122	1188	5899	517	9896.74	7531.92	31972.58	3686.21	02	02
4599	03-02-2023	1234	3854	2321	406	3911.78	24434.36	12579.82	2894.78	03	02

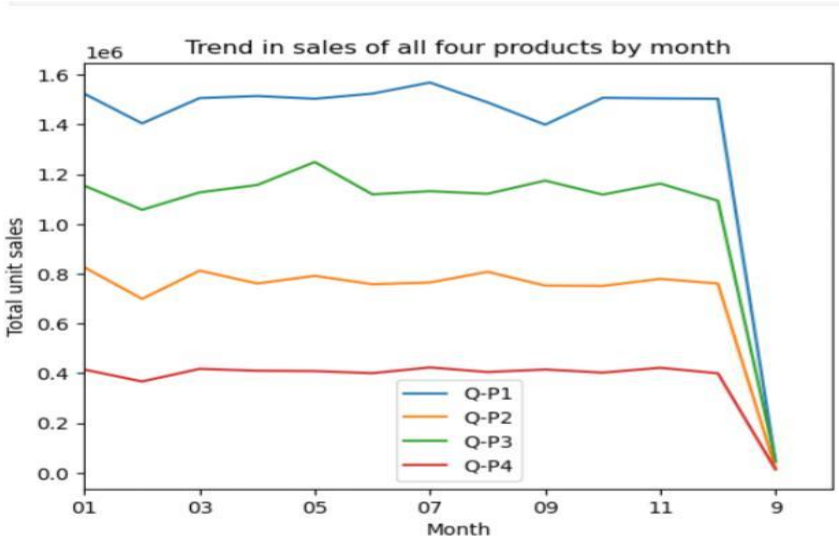
Trend in sales of all four products during certain months

Month_plot():
    Fig, ax = plt.subplots()

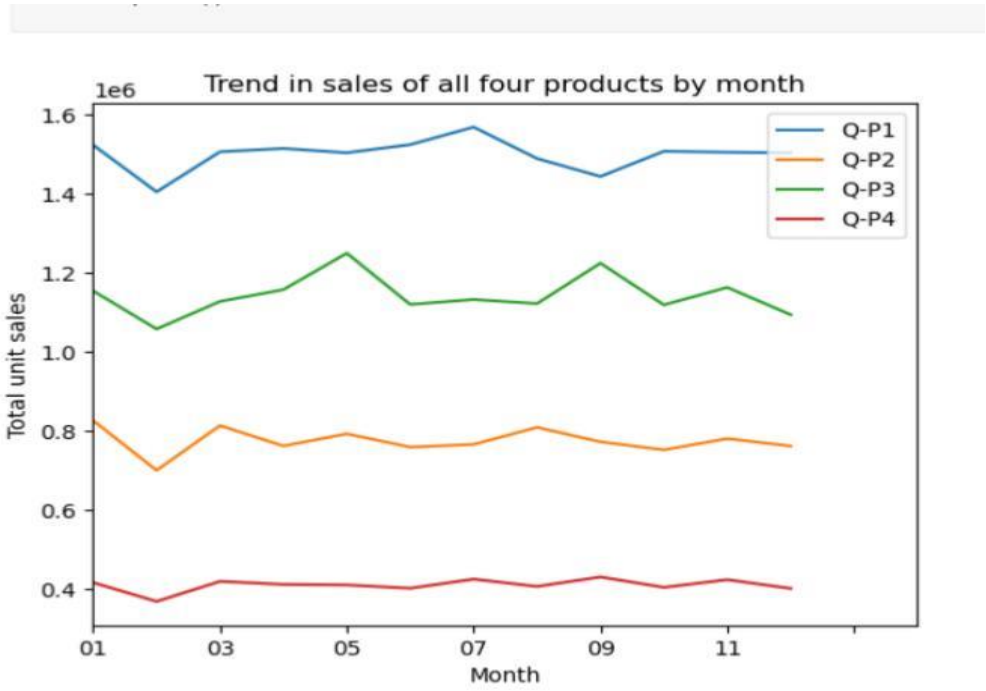
    # Plot the sales data for each product by month
    Data_reduced.groupby(‘Month’)[[‘Q-P1’, ‘Q-P2’, ‘Q-P3’, ‘Q-P4’]].sum().plot(ax=ax)

    # 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()
```




```
Data_reduced['Month'] = data['Month'].replace('9', '09')
Month_plot()
```



CONCLUSION:

In conclusion, the outlined data loading and preprocessing steps provide a foundational framework for preparing a dataset for analysis in Python using th e pandas library. By following these steps, you can ensure that your data is in a suitable format and quality for further exploration and visualization tasks.