PHASE -3 – PROJECT SUBMISSION

PRODUCT SALES ANALYSIS

Loading the Dataset:

- It involves loading the dataset into the python noteebook(‘statsfinal.csv’ – downloaded from https://www.kaggle.com/datasets/ksabishek/product-sales-data)
- Now read the dataset and store it in the form of ‘Pandas DataFrame’.

```
In [1]: import numpy as np
        import pandas as pd

In [2]: data = pd.read_csv('statsfinal.csv')
```

Cleaning the dataset:

- Cleaning the data, also known as data preprocessing or data wrangling, is a crucial step in any data analytics project.
- It involves the process of identifying and handling issues or imperfections in the dataset to ensure that the data is of high quality, accurate, and ready for analysis.
- Here's a description of the data cleaning process in a data analytics project:

Various stages of cleaning:

1. Data Inspection
2. Handling Missing Values
3. Handling Outliers

1. Data Inspection:

- Started by examining the dataset to get an initial understanding of its structure and contents.
- This includes checking the data types, column names, and the first few rows of data.

```
In [24]: print(data.shape)
         print(data.info())
```

```
(4600, 10)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  4600 non-null   int64
 1   Date        4600 non-null   object
 2   Q-P1        4600 non-null   int64
 3   Q-P2        4600 non-null   int64
 4   Q-P3        4600 non-null   int64
 5   Q-P4        4600 non-null   int64
 6   S-P1        4600 non-null   float64
 7   S-P2        4600 non-null   float64
 8   S-P3        4600 non-null   float64
 9   S-P4        4600 non-null   float64
dtypes: float64(4), int64(5), object(1)
memory usage: 359.5+ KB
None
```

```
In [25]: print(data.columns)
```

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

```
In [30]: print(data[0:5])
```

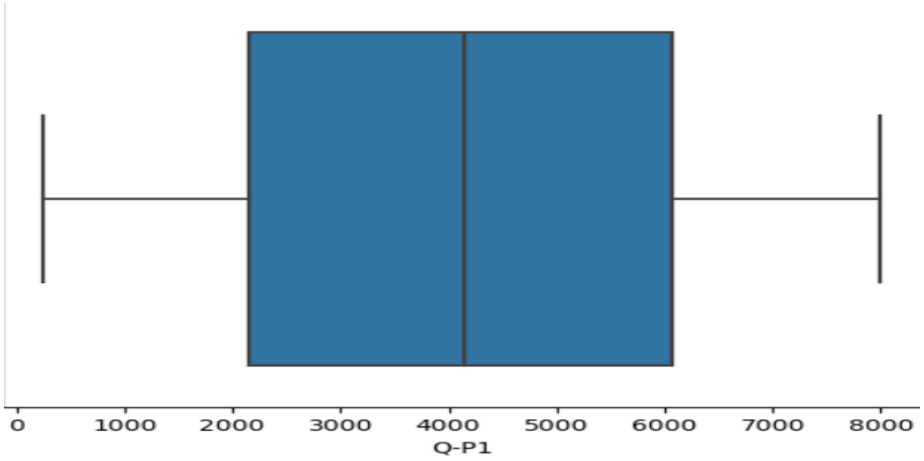
```
   Unnamed: 0        Date  Q-P1  Q-P2  Q-P3  Q-P4      S-P1      S-P2  \
0           0  13-06-2010  5422  3725   576   907  17187.74  23616.50
1           1  14-06-2010  7047   779  3578  1574  22338.99   4938.86
2           2  15-06-2010  1572  2082   595  1145   4983.24  13199.88
3           3  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
```

2. Handling Missing Values:

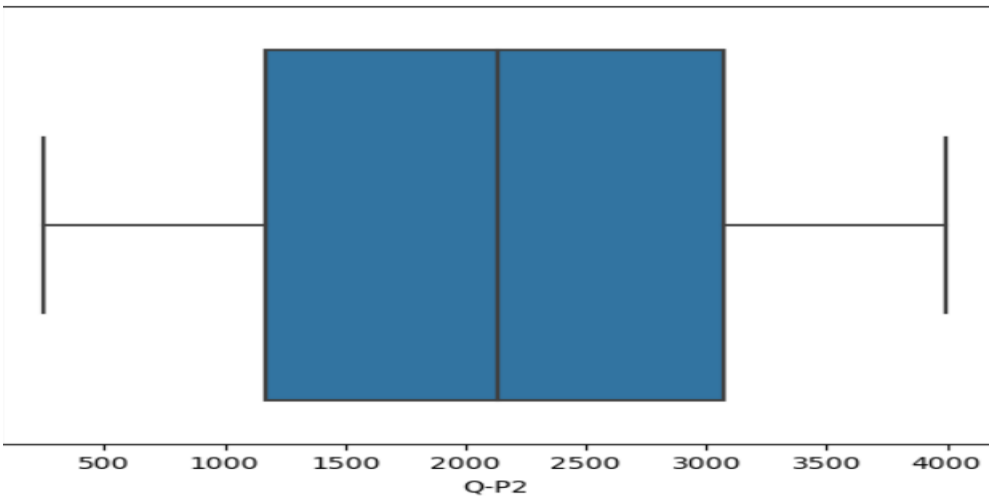
- Identify and handle missing data points. Common strategies include:
- Removing rows with missing values: If the missing data is minimal and the rows are not critical.
- Imputing missing values: Replacing missing values with a specific value (e.g., mean, median, or mode of the column) or using predictive modeling.
- **There were no null values or any missing values in the given dataset**

4. Handling Outliers:

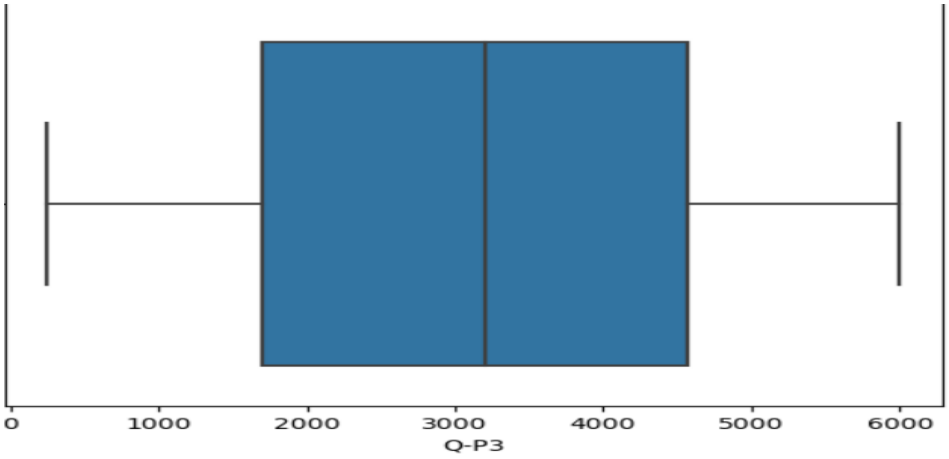
- Detected and address outliers, which are data points significantly different from the majority.
- Box plot was constructed to check outliers
- But the dataset did not contain any outliers



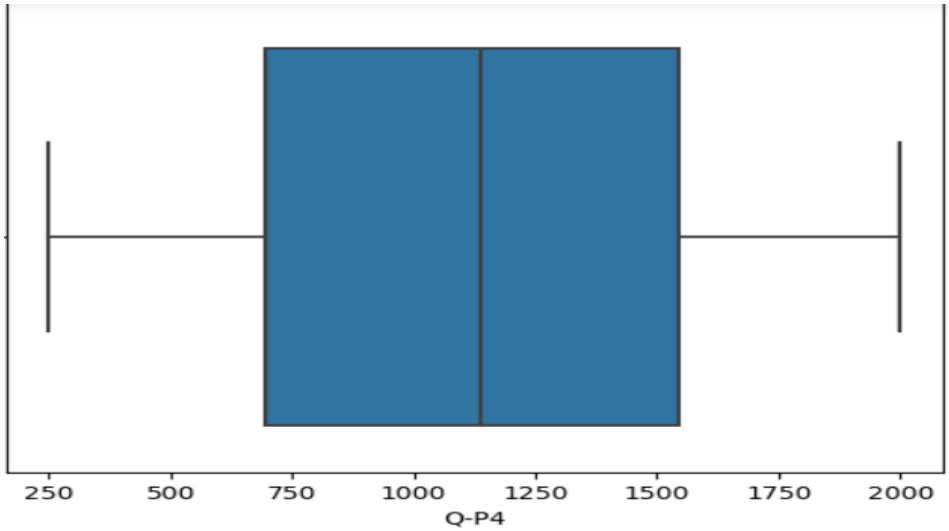
(Boxplot for the Q-P1 indicating no outliers)



(Boxplot for Q-P2 – indicating no outlier)



(Boxplot for Q-P3– indicating no outlier)



(Boxplot for Q-P4 – indicating no outlier)

Data Exploration from DataSet:

- At this stage, the dataset contained 8 columns with 4 columns the total selling cost of the respective four products and other 4 ,revenue from the each product .
- The ranges of each column and their corresponding paramaters are found.

```
In [26]: print(data.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
```

- From the ‘Date’ attribute of the dataset, years and month are obtained.
- From the year, the distribution of sales of each product for each year can be visualised.

```
In [5]: date =[]
        month = []
        year = []
```

```
In [27]: for i in data['Date']:
             l = i.split('-')
             date.append(l[0])
             month.append(l[1])
             year.append(l[2])
```

(Splitting the day , month and year from given dates)

```
In [8]: y1 = np.unique(year)
        ind3 = []
        for i in y1:
            ind3.append(year.index(i))
        print(ind3)
```

```
[0, 201, 565, 928, 1292, 1656, 2020, 2383, 2747, 3111, 3475, 3838, 4202, 4566]
```

(Finding the starting index of each year in the dataset)

In [14]:
```
import pandas as pd

qp1_year = {}
qp2_year = {}
qp3_year = {}
qp4_year = {}

QP_1 = data['Q-P1'].tolist()
QP_2 = data['Q-P2'].tolist()
QP_3 = data['Q-P3'].tolist()
QP_4 = data['Q-P4'].tolist()

for i in range(1, len(ind3)):
    qp1_year[y1[i-1]] = sum(QP_1[ind3[i-1]:ind3[i]+1])
    qp2_year[y1[i-1]] = sum(QP_2[ind3[i-1]:ind3[i]+1])
    qp3_year[y1[i-1]] = sum(QP_3[ind3[i-1]:ind3[i]+1])
    qp4_year[y1[i-1]] = sum(QP_4[ind3[i-1]:ind3[i]+1])

qp1_year['2023'] = sum(QP_1[4566:])
qp2_year['2023'] = sum(QP_2[4566:])
qp3_year['2023'] = sum(QP_3[4566:])
qp4_year['2023'] = sum(QP_4[4566:])

# Convert dictionaries to DataFrames
qp1_df = pd.DataFrame(list(qp1_year.items()), columns=['Year', 'QP1'])
qp2_df = pd.DataFrame(list(qp2_year.items()), columns=['Year', 'QP2'])
qp3_df = pd.DataFrame(list(qp3_year.items()), columns=['Year', 'QP3'])
qp4_df = pd.DataFrame(list(qp4_year.items()), columns=['Year', 'QP4'])
```

(Creating a dictionary for each product and filling it with the products sold in each year and converting it into a dataframe)

In [28]:
```
print(qp1_df)
```

	Year	QP1
0	2010	812252
1	2011	1440142
2	2012	1509267
3	2013	1536451
4	2014	1574496
5	2015	1489050
6	2016	1520619
7	2017	1470429
8	2018	1533080
9	2019	1485121
10	2020	1506050
11	2021	1506701
12	2022	1463237
13	2023	150310

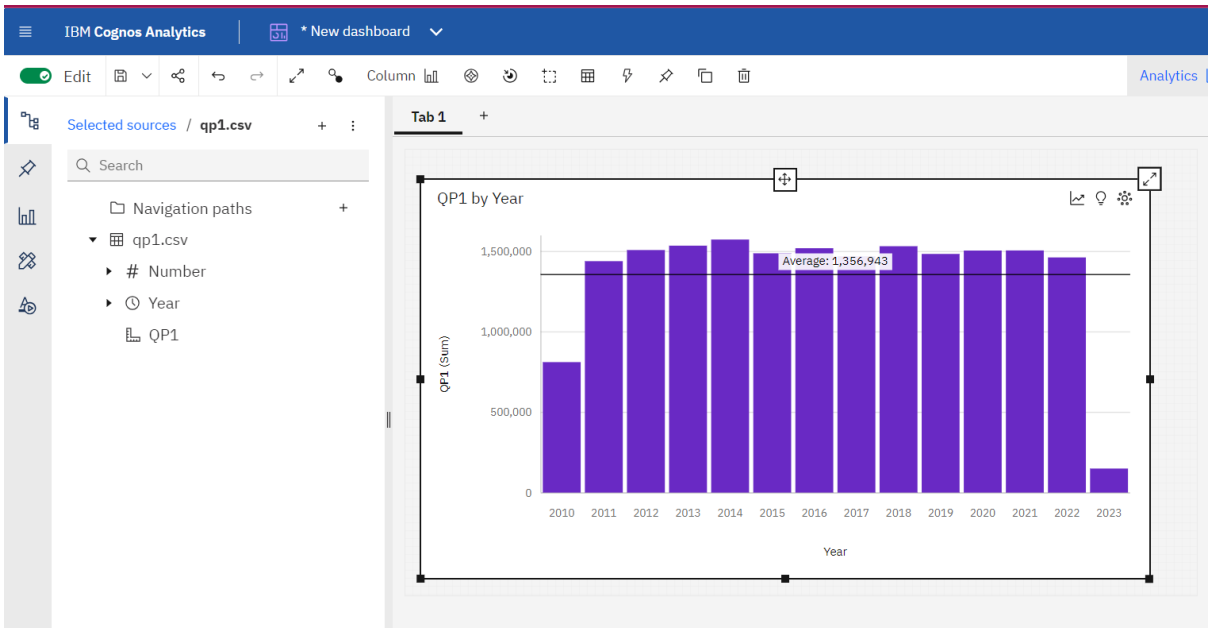
(Displaying the values)

```
In [16]: qp1_df.to_csv('qp1.csv')
         qp2_df.to_csv('qp2.csv')
         qp3_df.to_csv('qp3.csv')
         qp4_df.to_csv('qp4.csv')
```

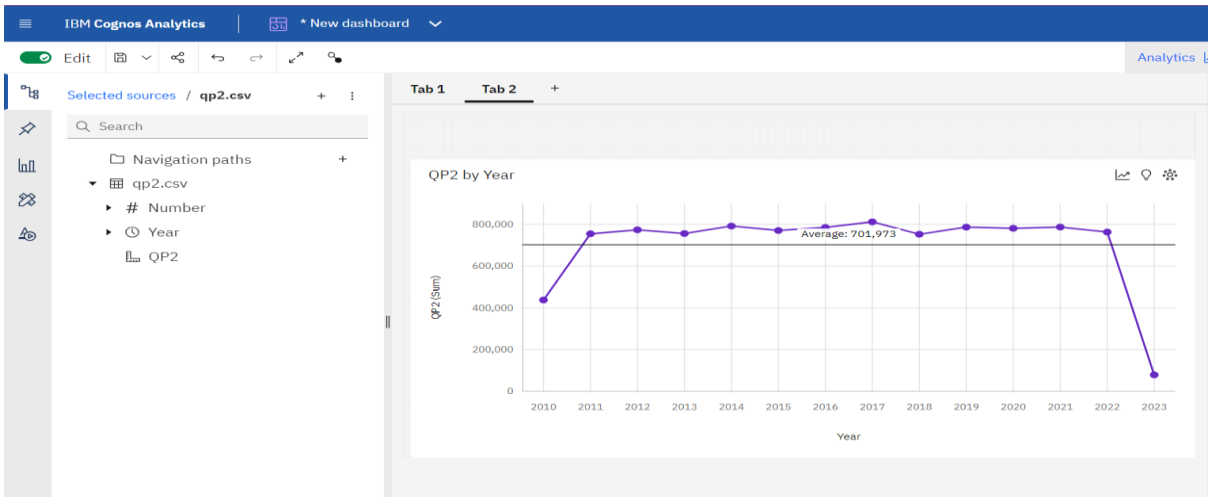
(Creating dataset from the dataframe)

Visualization in Cognos Analytics:

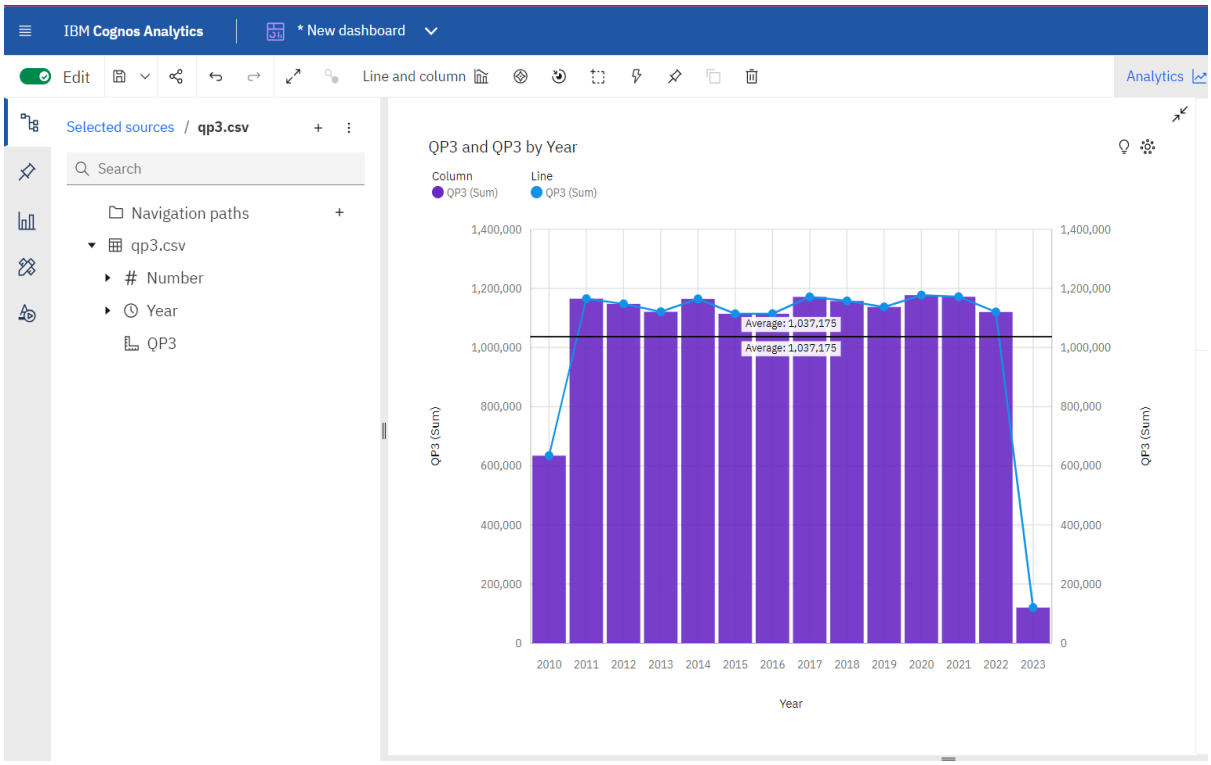
- Creating dashboard in IBM Cognos Analytics.
- Uploading the datasets created in the previous step
- Plotting various graphs for each product.



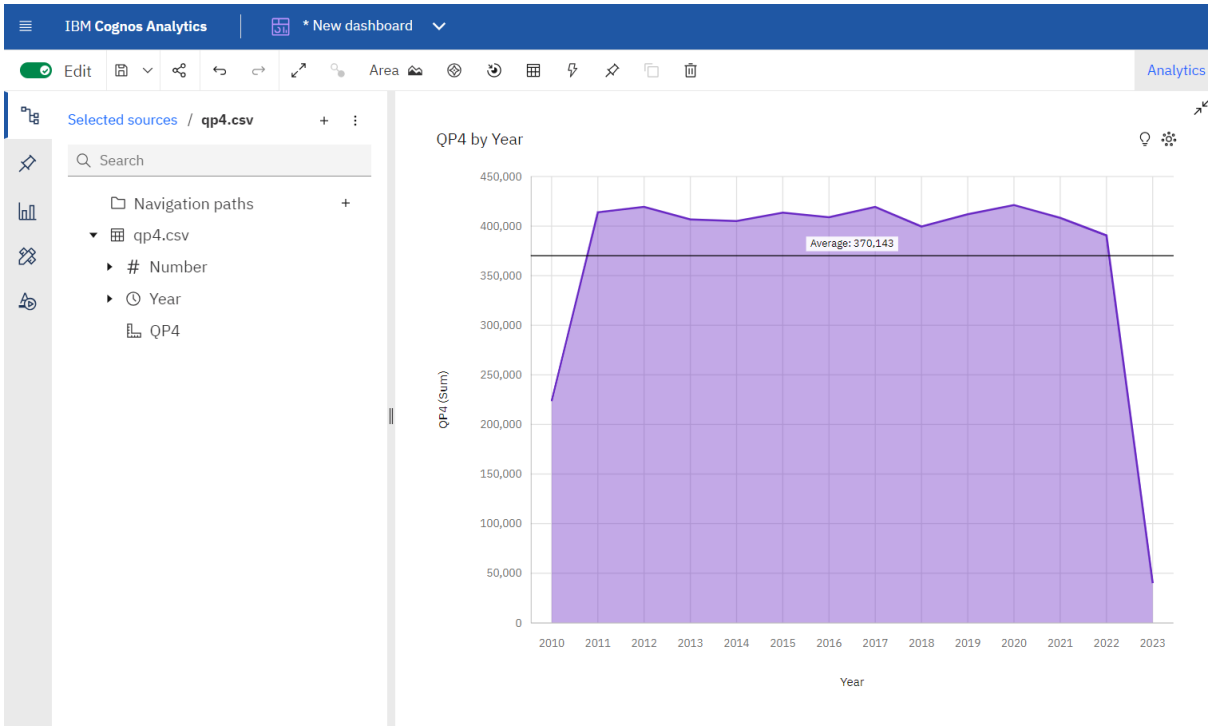
(Barplot of the product sales of – Q-P1 with the year)



(Lineplot of the product sales of – Q-P2 with the year)



(Barplot along with lineplot of the product – Q-P3 with the year)



(Area plot of the product – Q_P4 with the year)