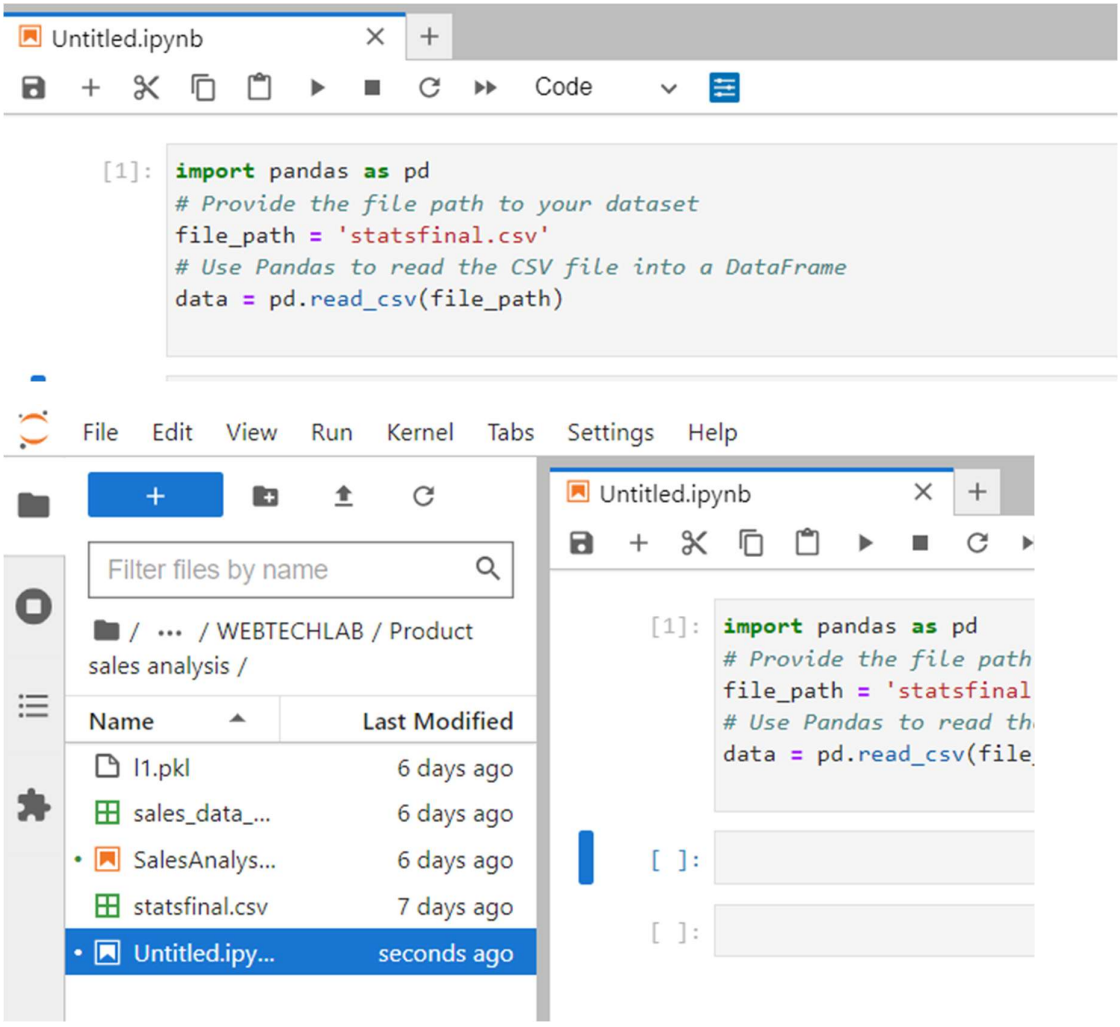
Phase3(Development part 1)

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

Building our project by loading and preprocessing the dataset :

Loading the dataset:



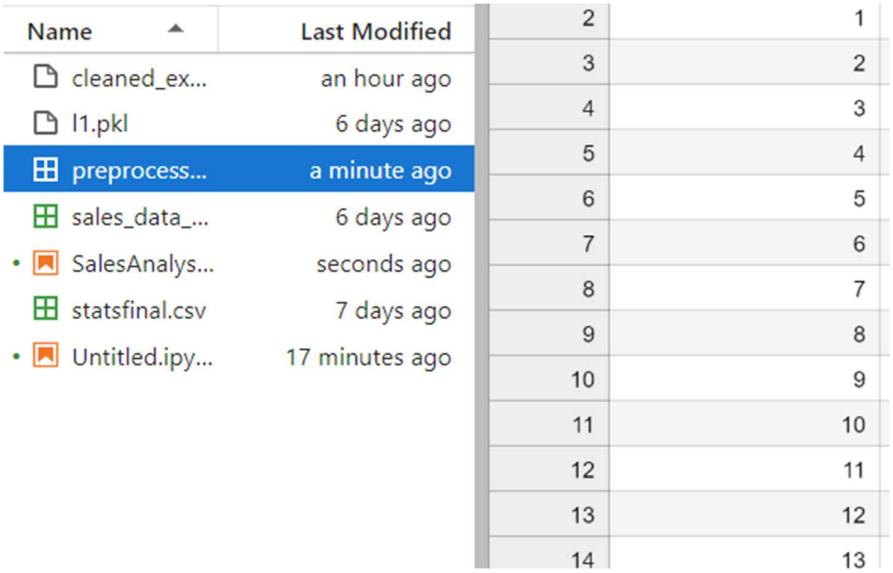
Preprocessing the data:

Source Code:

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

```
# Load the sales data from a CSV file
data = pd.read_csv('statsfinal.csv')  # Adjust the filename as needed
# 1: Data Preprocessing
# Remove rows with missing values or replace them with appropriate values
data.dropna(subset=['S-P1', 'Q-P1'], inplace=True)
scaler = StandardScaler()
data[['S-P1', 'Q-P1']] = scaler.fit_transform(data[['S-P1', 'Q-P1']])
# Save Preprocessed Data
# Save the preprocessed data to a new CSV file
data.to_csv('preprocessed_sales_data.csv', index=False)
```

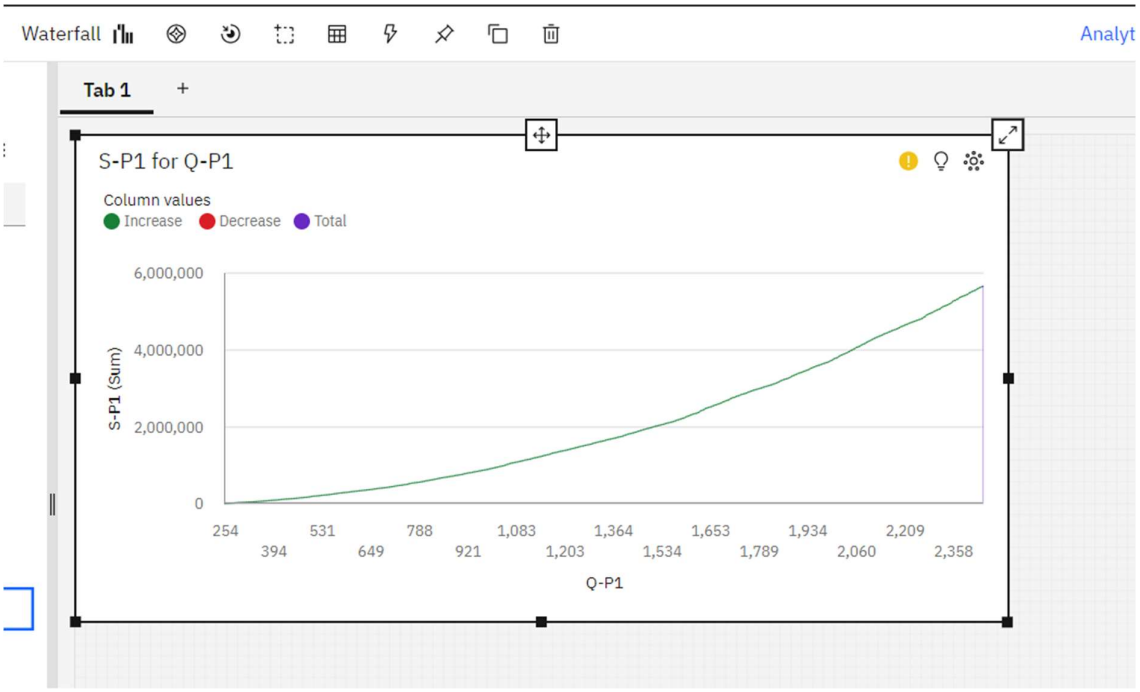
Output:



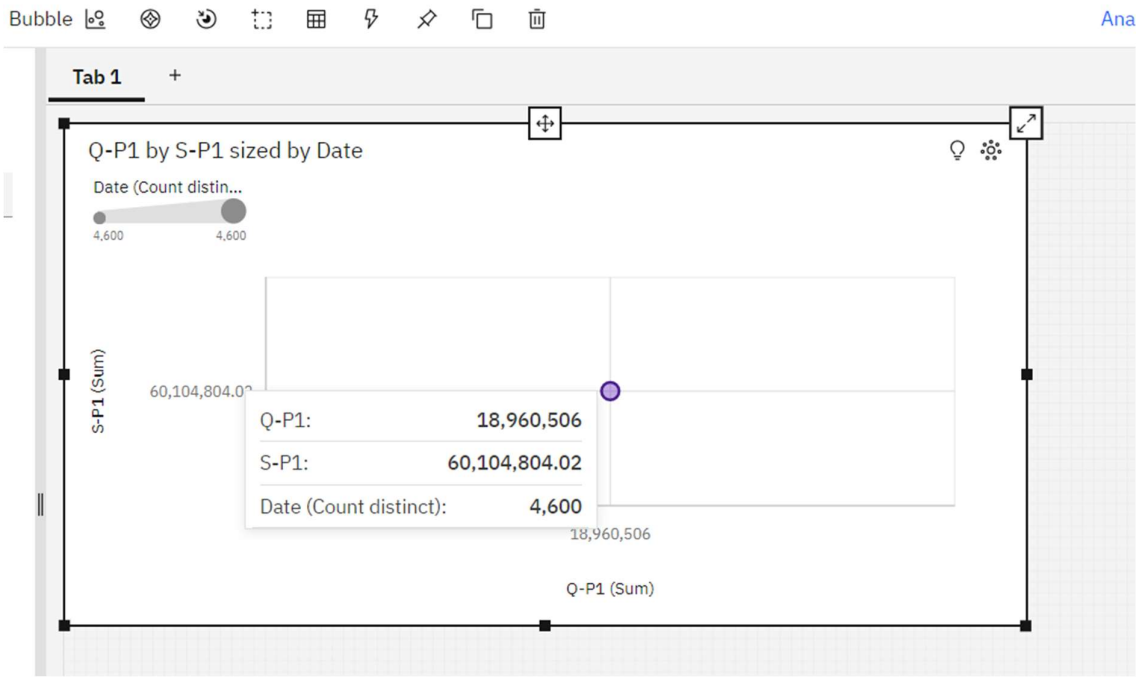
The product sales analysis using IBM Cognos for visualization.

Source: Given Dataset.

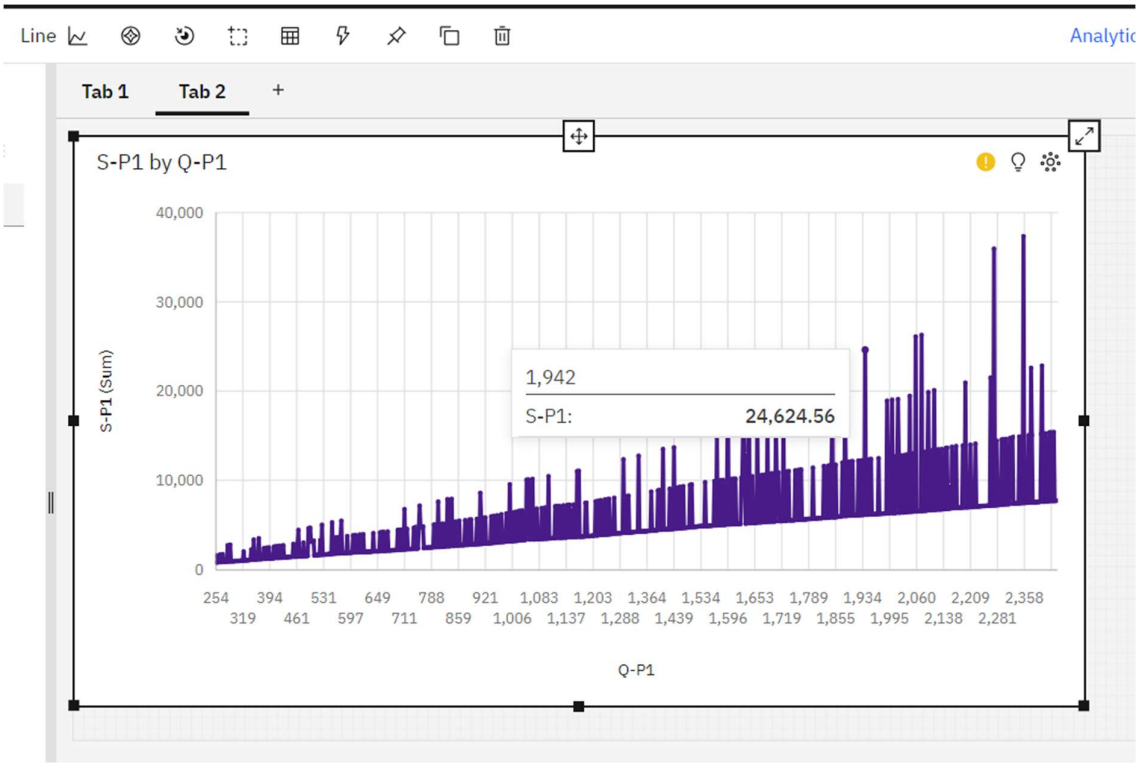
Waterfall : (Betweeen S-P1 and Q-P1)



Bubble: (Q-P1 by S-P1 sized by Date)

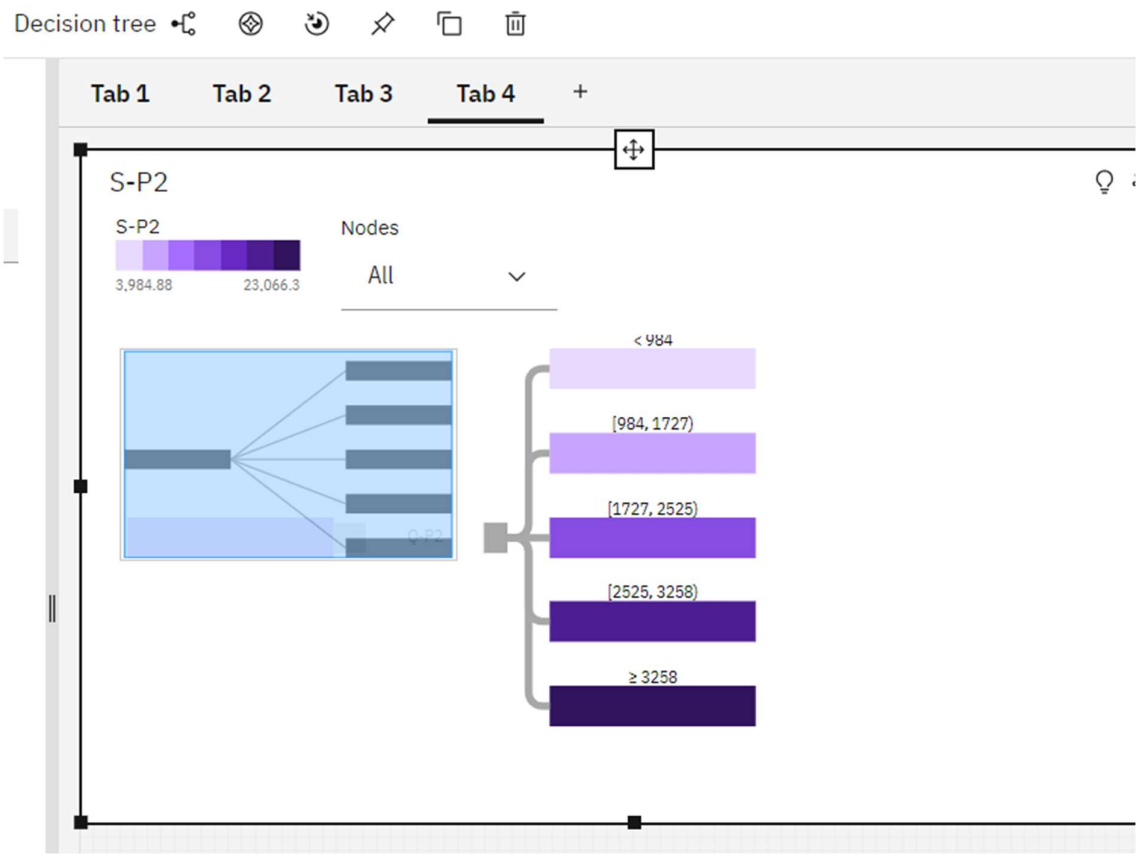


Line : (S-P1 by Q-P1)

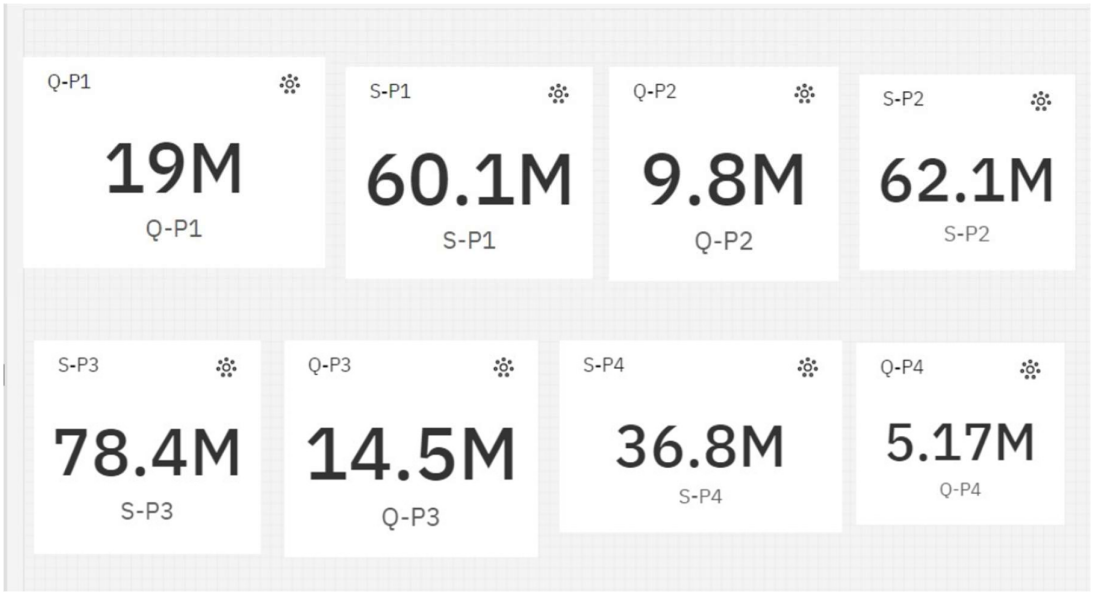


Scattered Column:(S-P4 by Q-P4 Colored by DATE)

Decision Tree :(S-P2)



Summary :



Define the analysis objectives and collect sales data from source shared:

1. Define the Objective:

- Our objective is to use clustering to analyze the sales data based on the product count, revenue count, and date.

2. Data Collection:

- Collecting the sales data from the sources shared or our internal sales records.Checks the data includes the following columns:
    1. Four columns for product counts.
    2. Four columns for revenue counts.
    3. One column for the date.

3. Data Preprocessing:

- Follow the steps for data preprocessing to clean, transform, and prepare our data for clustering analysis. This includes handling missing data, data cleaning, feature selection, feature engineering, and more.

4. Feature Selection:

- Selecting the relevant columns for our clustering analysis. In our case, we would include the product name, revenue average, and time columns.

5. Feature Engineering:

- Creating additional features if necessary based on your analysis objectives. For instance, you can extract relevant time components or calculate aggregated metrics.

6. Apply Clustering Algorithm:

- Use a clustering algorithm like K-Means to group our data into clusters based on the selected features.

7. Interpretation and Insights:

- Analyzing the results of the clustering and extracting meaningful insights. Understanding the characteristics of each cluster and how they relate to our sales data.

8. Visualize Results:

- Visualizing the clusters using plots and graphs to make the insights more accessible.

9. Report and Share:

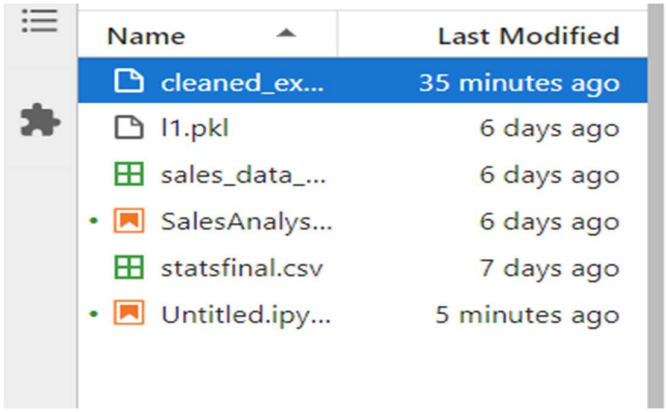
- Creating a report summarizing the analysis, insights, and any actions or recommendations derived from the clustering results.

Cleaning the dataset:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df=pd.read_csv('statsfinal.csv')

date_column_name = 'Date'  # Replace with the name of your date column

df[date_column_name] = pd.to_datetime(df[date_column_name], errors='coerce')

# Step 3: Handle errors or inconsistencies (e.g., fill NaN values or drop invalid rows)

# Example: Drop rows with invalid dates

df = df.dropna(subset=[date_column_name])

# Step 4: Save the cleaned DataFrame back to a new Excel file

output_file_path = 'cleaned_excel_file.xlsx'  # Replace with the desired output file path

df.to_excel(output_file_path, index=False)

print("Cleaned data saved to", output_file_path)
```

Output:

```
Cleaned data saved to cleaned_excel_file.xlsx

[ ]:
```



Process and clean the collected data to ensure its accuracy and reliability :

Data Summary Statistics:

Calculating summary statistics, such as mean, median, standard deviation, and range for numerical columns. This helps us identify outliers and unusual values that may affect data accuracy.

Source Code:

```
# Calculate summary statistics for numerical columns

summary_stats = df.describe()

print(summary_stats)
```

Output:

```
       Unnamed: 0         Q-P1         Q-P2         Q-P3         Q-P4  \
count  4575.000000  4575.000000  4575.000000  4575.000000  4575.000000
mean   2298.893989  4123.531148  2129.314317  3144.170710  1123.614645
std    1328.106921  2243.482107  1089.704783  1671.090234   497.829405
min       0.000000   254.000000   251.000000   250.000000   250.000000
25%    1148.500000  2150.000000  1166.000000  1695.500000   695.500000
50%    2299.000000  4138.000000  2133.000000  3197.000000  1137.000000
75%    3449.500000  6072.000000  3069.500000  4567.000000  1545.500000
max    4599.000000  7998.000000  3998.000000  6000.000000  2000.000000

              S-P1          S-P2          S-P3          S-P4  \
count   4575.000000   4575.000000   4575.000000   4575.000000
mean   13071.593738  13499.852769  17041.405250   8011.372417
std     7111.838279   6908.728326   9057.309068   3549.523656
min      805.180000   1591.340000   1355.000000   1782.500000
25%     6815.500000   7392.440000   9189.610000   4958.915000
50%    13117.460000  13523.220000  17327.740000   8106.810000
75%    19248.240000  19460.630000  24753.140000  11019.415000
max    25353.660000  25347.320000  32520.000000  14260.000000

       marketing_strategy
count                 0.0
mean                  NaN
std                   NaN
min                   NaN
25%                   NaN
50%                   NaN
75%                   NaN
max                   NaN
```

Data Integrity Checks:

Check for data integrity issues, such as missing values, duplicates, and inconsistent data types.

Source code :

```
# Check for missing values

missing_values = df.isnull().sum()

# Check for duplicates

duplicate_rows = df[df.duplicated(keep='first')]

print("Missing Values:")

print(missing_values)

print("Duplicate Rows:")

print(duplicate_rows)
```

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
marketing_strategy    4575
dtype: int64
Duplicate Rows:
Empty DataFrame
Columns: [Unnamed: 0, Date, Q-P1, Q-P2, Q-P3, Q-P4, S-P1, S-P2, S-P3, S-P4, marketing_strategy]
Index: []
```

Data Consistency Checks:

Examine data consistency, including unique values in categorical columns, date format consistency, and values that should follow a specific pattern.

Source Code:

```
# Check unique values in a column

unique_values = df['Date'].unique()
```

```
# Check date format consistency (assuming the 'Date' column is in datetime format)

date_format_consistency = pd.to_datetime(df['Date'], errors='coerce').notna()

print("Unique Values in Category:")

print(unique_values)

print("Date Format Consistency:")

print(date_format_consistency)
```

Output:

```
Unique Values in Category:
['2010-06-13T00:00:00.000000000' '2010-06-14T00:00:00.000000000'
 '2010-06-15T00:00:00.000000000' ... '2023-01-02T00:00:00.000000000'
 '2023-02-02T00:00:00.000000000' '2023-03-02T00:00:00.000000000']
Date Format Consistency:
0       True
1       True
2       True
3       True
4       True
       ...
4595    True
4596    True
4597    True
4598    True
4599    True
Name: Date, Length: 4575, dtype: bool
```

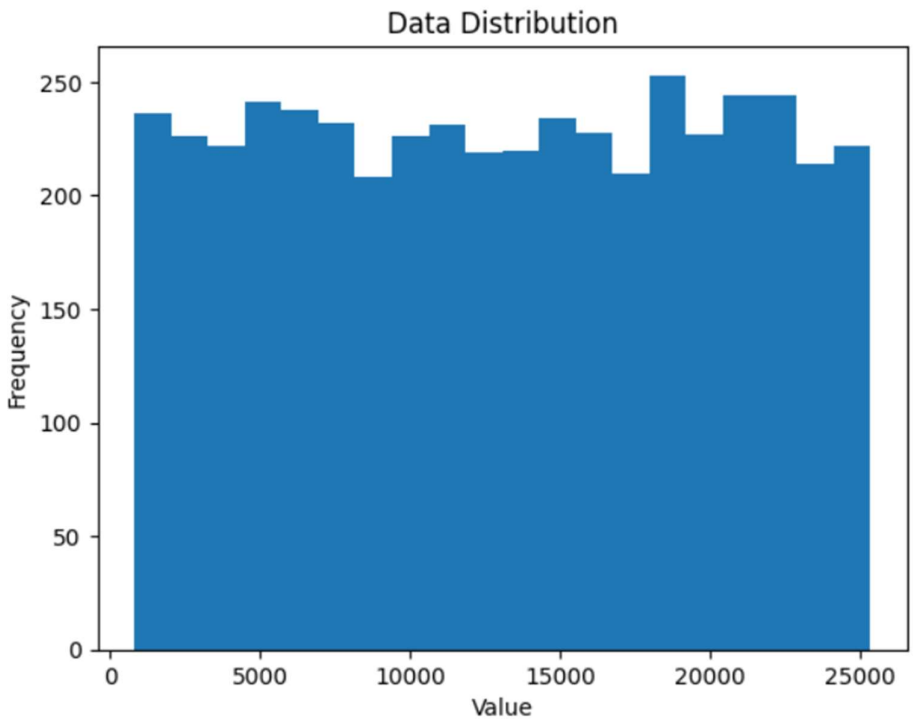
Data Visualization:

Create plots and visualizations to better understand the data distribution, which can reveal anomalies.

Source code :

```
import matplotlib.pyplot as plt

# Example: Histogram of a numerical column

plt.hist(df['S-P1'], bins=20)

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.title('Data Distribution')

plt.show()
```

Output:




```
missing_values = df.isnull().sum().sum()
# Step 3: Handle errors or inconsistencies
# Example: Filling missing values or drop rows with missing data
df.dropna(inplace=True)
# Step 4: Apply statistical analysis
# Example: Calculate descriptive statistics
statistics = df.describe()
# Step 5: Calculate the accuracy and reliability metrics
accuracy = "High" if df['S-P2'].max() - df['S-P1'].min() < 10 else "Low"
reliability = "High" if df['S-P3'].std() < 2 else "Low"
# Step 6: Report the findings
print("Data Accuracy:", accuracy)
print("Data Reliability:", reliability)
print("Missing Values:", missing_values)
```

```
print("Descriptive Statistics:\n", statistics)
# Step 7: Save the cleaned data to a new Excel file
output_file_path = 'cleaned_excel_file.xlsx'
df.to_excel(output_file_path, index=False)
print("Cleaned data saved to", output_file_path)
```

Output:

```
Data Accuracy: Low
Data Reliability: Low
Missing Values: 0.0
Descriptive Statistics:
        Unnamed: 0  Q-P1  Q-P2  Q-P3  Q-P4  S-P1  S-P2  S-P3  S-P4  \
count          0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
mean           NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
std            NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
min            NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
25%            NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
50%            NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
75%            NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
max            NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN

       marketing_strategy
count                 0.0
mean                  NaN
std                   NaN
min                   NaN
25%                   NaN
50%                   NaN
75%                   NaN
max                   NaN
Cleaned data saved to cleaned_excel_file.xlsx
```