**PRODUCT SALES ANALYSIS**

**OBJECTIVE**

Product sales analysis objectives encompass performance evaluation, sales forecasting, market segmentation, competitor benchmarking, inventory management, customer insights, pricing optimization, and more. These objectives drive decision-making, profitability, and business growth, fostering adaptability and a competitive edge in the marketplace.

**Step 1: Data Preprocessing**

1. **Data Cleaning:** Remove or fill in missing values to ensure that there are no gaps in your dataset. This may involve removing rows with missing data, imputing missing values with appropriate substitutes, or using techniques like interpolation.

2. **Handling Duplicates:** Check for and eliminate duplicate entries, which can skew analysis results. Duplicates can occur due to data entry errors or system glitches.

3. **Data Transformation:** Convert data types if necessary. For example, convert categorical variables into numerical format using encoding methods, standardize or normalize numeric data, and create new features through feature engineering.

4. **Outlier Detection and Handling:** Identify outliers in your data. Outliers are data points significantly different from the majority of the data and can distort analysis results. You can choose to remove, transform, or keep outliers based on your analysis goals.

5. **Dealing with Imbalanced Data:** If you're analyzing sales data with imbalanced classes (e.g., a small number of high-value sales and a large number of low-value sales), consider techniques like oversampling, undersampling, or using weighted models to address this imbalance.

6. **Feature Selection:** Select the most relevant features (columns) for your analysis. Some features may not contribute significantly to your analysis and can be excluded to reduce complexity.

7. **Handling Time-Series Data:** If your sales data includes timestamps, consider time-series analysis. You can resample, aggregate, or perform seasonal decomposition to reveal trends and seasonality in your data.

8. **Data Scaling:** In some cases, it might be necessary to scale your data to ensure that all features have the same influence on the analysis. This is particularly important when using machine learning algorithms.

9. **Data Splitting:** Split the data into training and testing sets to validate the performance of your analysis or predictive models.

**Step 2: Jupyter Environment Setup**

* **Environment Setup**: Ensure that your Jupyter environment is properly configured and ready for data analysis.
* **Library Installation**: Install the necessary Python libraries like Pandas, Matplotlib, Seaborn, and NumPy for data analysis and visualization.

**Step 3: Data Analysis and Visualization**

* **Chart Creation**: Design and create charts and graphs using Matplotlib and Seaborn to visually represent performance metrics and efficiency of product sales
* **Statistical Analysis**: Utilize Python's capabilities for statistical analysis, such as calculating mean values and standard deviations.

**Step 4: Insights Generation**

* **Pattern Identification**: Analyze the visualizations and statistical results to identify patterns, trends, or areas for improvement in public transportation services.
* **Insight Formulation**: Translate these patterns into actionable insights, such as suggesting improvements for specific routes or services.

**Step 5: JupyterBook Creation**

* **Installation**: Install JupyterBook, a tool to create interactive, shareable, and documentation-ready books from Jupyter notebooks.
* **Book Structure**: Define the structure of your JupyterBook, including chapters and sections.
* **Notebook Integration**: Include your Jupyter notebooks, code, visualizations, and insights into the JupyterBook

**JUPYTERBOOK**

**VISUALYISING TOP 10 STOPS BY BOARDING POINTS USING JUPYTERNOTE BOOK**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the cleaned data

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

# Line Plot

plt.figure(figsize=(12, 6))

sns.lineplot(x='Date', y='Q-P1', data=data, label='Q-P1')

sns.lineplot(x='Date', y='Q-P2', data=data, label='Q-P2')

sns.lineplot(x='Date', y='Q-P3', data=data, label='Q-P3')

plt.title("Temperature Over Time")

plt.xlabel("Date")

plt.ylabel("Temperature")

plt.xticks(rotation=45)

plt.legend()

plt.show()

# Bar Plot

plt.figure(figsize=(12, 6))

sns.barplot(x='Date', y='S-P1', data=data, color='blue', label='S-P1')

sns.barplot(x='Date', y='S-P2', data=data, color='green', label='S-P2')

plt.title("S-P1 vs. S-P2 Over Time")

plt.xlabel("Date")

plt.ylabel("Value")

plt.xticks(rotation=45)

plt.legend()

plt.show()

# Box Plot

plt.figure(figsize=(8, 6))

sns.boxplot(x='Q-P1', data=data, color='blue')

plt.title("Box Plot for Q-P1")

plt.xlabel("Q-P1")

plt.show()

# Heatmap

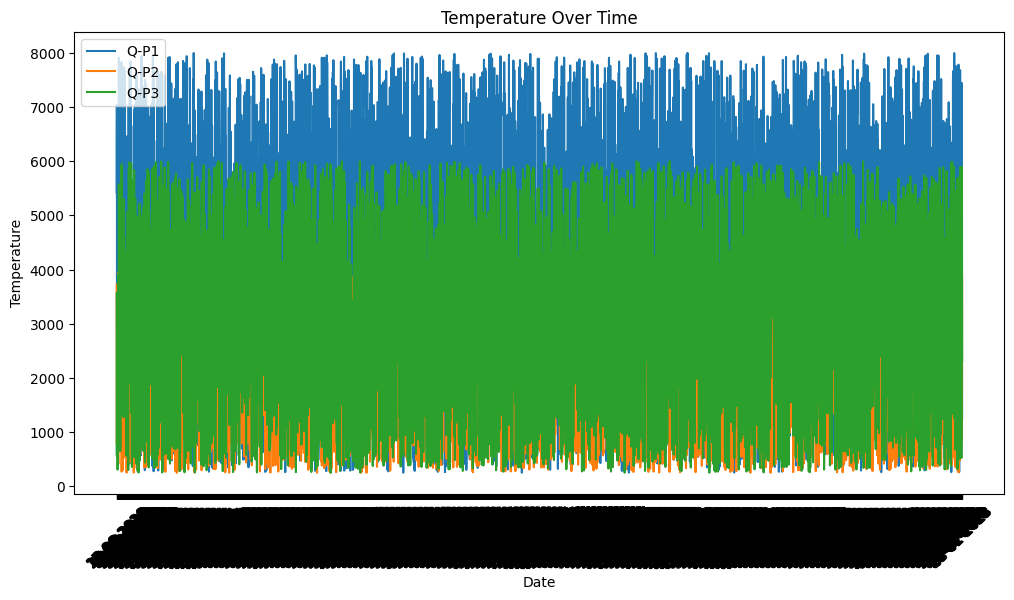
pivot\_data = data.pivot\_table(values='Q-P1', index='Date', columns='Q-P2')

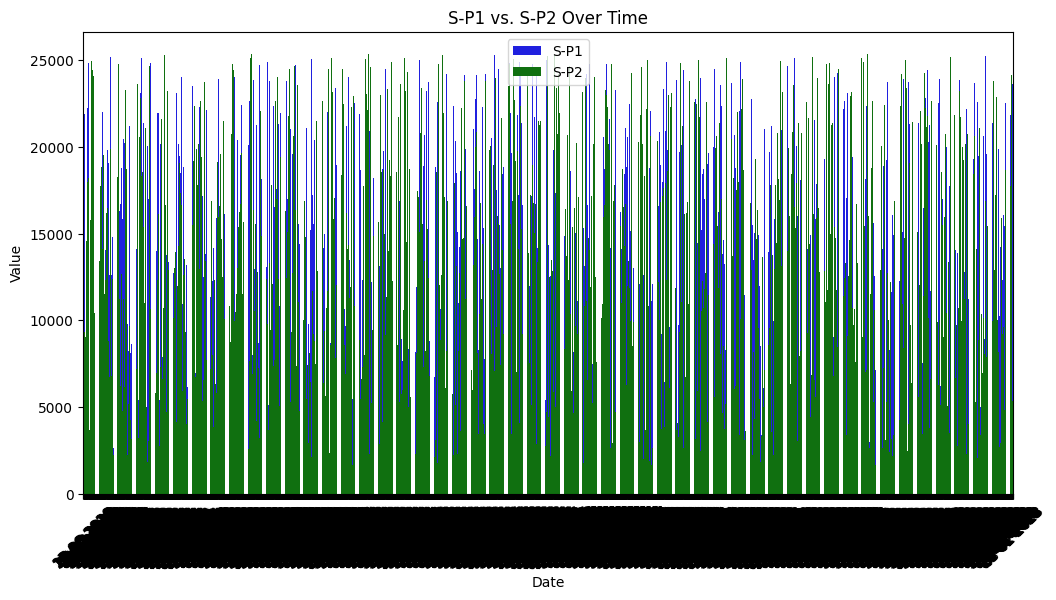
plt.figure(figsize=(10, 8))

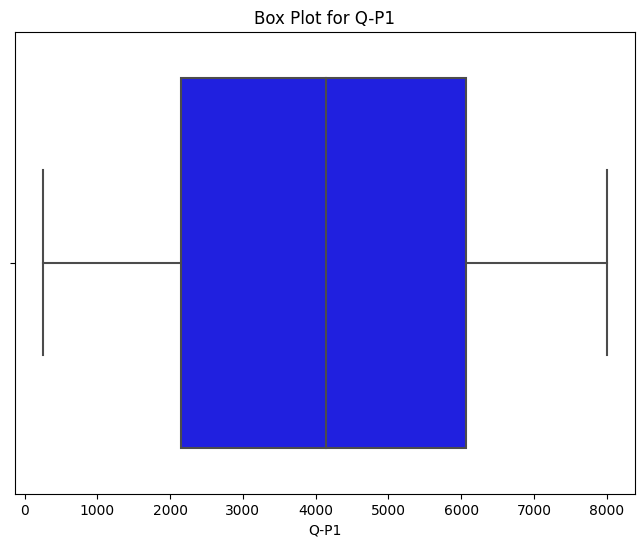
sns.heatmap(pivot\_data, cmap='coolwarm')

plt.title("Heatmap of Q-P1 vs. Q-P2")

plt.show()

**OUTPUT**

****

****