**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**

Data preprocessing is a crucial and often labor-intensive step in data analysis and machine learning. It involves cleaning, transforming, and organizing raw data into a structured format that is suitable for analysis. This process ensures that the data is accurate, consistent, and ready for modeling. Data preprocessing typically includes:

Data Cleaning: This step involves handling missing values, correcting inconsistencies, and removing outliers. Data cleaning ensures that the dataset is free of errors and inaccuracies that could lead to incorrect conclusions.

Data Transformation: Data transformation includes normalizing or standardizing data to bring different features to a common scale, encoding categorical variables, and applying mathematical transformations to make the data suitable for modeling.

Feature Selection: Feature selection involves identifying and selecting the most relevant features or variables for analysis. It helps in reducing dimensionality and improving model performance.

Data Integration: In cases where data comes from multiple sources, data integration combines them into a single, cohesive dataset. This step is essential for comprehensive analysis.

Data Reduction: Data reduction techniques, such as principal component analysis (PCA), can be used to reduce the dimensionality of the dataset while retaining its most important information.

Data Splitting: Data is often split into training and testing sets to assess model performance. This ensures that the model's generalization is evaluated on unseen data.

**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

# Read the data from the provided CSV or DataFrame

data = pd.read\_csv('/content/statsfinal.csv') # Replace 'your\_data.csv' with the actual file path or DataFrame

# Extract relevant columns

date = data['Date']

product\_sales = data[['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4']]

# Create a bar chart for product sales

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

for product in product\_sales.columns:

plt.bar(date, product\_sales[product], label=product)

plt.xlabel('Date')

plt.ylabel('Sales')

plt.title('Product Sales Analysis - Bar Chart')

plt.xticks(rotation=45)

plt.legend()

# Create a line chart for a specific product (e.g., 'Q-P1')

selected\_product = 'Q-P1'

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

plt.plot(date, product\_sales[selected\_product], marker='o', label=selected\_product, color='green')

plt.xlabel('Date')

plt.ylabel('Sales')

plt.title(f'{selected\_product} Sales Analysis - Line Chart')

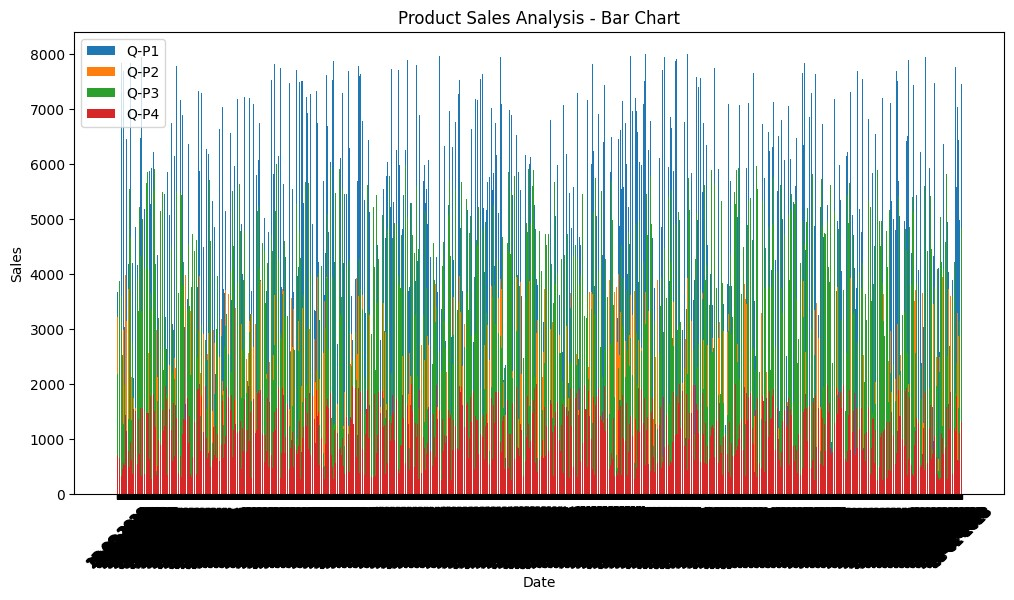
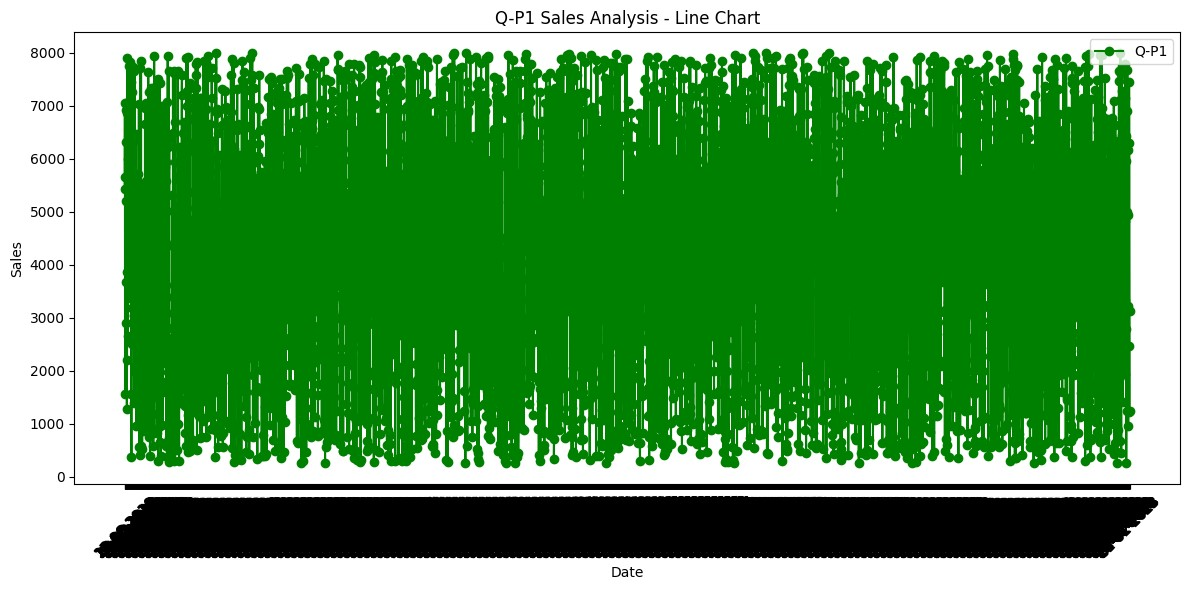
plt.xticks(rotation=45)

plt.legend()

# Show both charts

plt.tight\_layout()

plt.show()

**OUTPUT**