DATA ANALYTICS USING COGNOS – GROUP 3

Project 5: Product Sales Analysis

Project Description:

The "Product Sales Analysis" project focuses on harnessing the power of IBM Cognos to analyze sales data, extract valuable insights, and drive informed business decisions. By exploring top-selling products, identifying peak sales periods, and understanding customer preferences, this initiative aims to enhance inventory management and refine marketing strategies. Key components of the project include defining analysis objectives, collecting relevant sales data, designing insightful visualizations within IBM Cognos, and translating findings into actionable insights.

Phase 5: Project Documentation and Submission

Objectives:

1. Utilize IBM Cognos to extract insights about top-selling products, peak sales periods, and customer preferences from sales data.
2. Leverage these insights to optimize inventory management and refine marketing strategies for more effective business operations.

Design thinking:

Empathize: The code starts by understanding the needs of the stakeholders, which are to analyze the product sales data and to forecast future sales.

Define: The code then defines the problem statement, which is to develop a comprehensive analysis of product sales data, including data loading and preprocessing, visualization and reporting, and forecasting future sales.

Ideate: The code then generates creative solutions to the problem statement. This includes selecting a simple moving average model for forecasting future sales, as well as using a variety of data visualization techniques to communicate the insights from the analysis.

Development Phases:

Data loading and preprocessing

Step 1: Library Imports

The code imports the required Python libraries.

• pandas (imported as pd) is a popular library for data manipulation and analysis.

• matplotlib (imported as plt) is a library used for creating data visualizations, such as plots and charts.

Step 2: Data Loading

The code reads a dataset from a CSV file named "statsfinal.csv" and loads it into a Pandas DataFrame named 'data'.

Step 3: Data Inspection:

The code prints the first few rows of the 'data' DataFrame using the .head() method. By default, .head() displays the first 5 rows of the DataFrame. This allows to quickly inspect and get an initial overview of the dataset's structure and contents.

Development Part 1: Data visualization

Using IBM Cognos, we visualize the dataset to get clear insights about the data. Here Bar plot have been used to analyse the Sales over a period of time along with the revenue generated on each date.

Using Cognos for Sales Analysis: Step-by-Step Guide

1**. Data Collection and Preparation**: Begin by gathering historical sales data from your various data sources, ensuring that the data is complete, accurate, and well-organized. Your dataset should include essential columns such as 'Date,' 'Product,' 'Sales,' and any other pertinent attributes. Make sure to save the data in a format compatible with IBM Cognos, such as CSV or Excel.

2. **IBM Cognos Setup**: Ensure that you have access to IBM Cognos Analytics and the necessary permissions to create reports and dashboards. If you are not sure about your permissions, reach out to your organization's Cognos administrator for assistance.

3. **Data Import**: After logging in to IBM Cognos Analytics, create a new data source connection and import your sales data into Cognos. Follow the platform's provided steps for

data import or consult with your organization's Cognos administrator for specific guidance on this process.

4**. Define Analysis Objectives**: Collaborate with relevant stakeholders to clearly define the analysis objectives. Ensure that these objectives align with your organization's goals. These objectives could encompass identifying top-selling products, understanding seasonality in sales, or segmenting customer preferences.

5. **Create Visualizations**: Utilize IBM Cognos Analytics to design meaningful visualizations. Start by creating a new dashboard in Cognos. If needed, construct data modules within Cognos to shape and prepare your data for visualization. Add reports to your dashboard and select appropriate chart types, such as bar charts, line graphs, or pie charts, to represent sales trends and customer behavior. Implement interactive elements, such as drill-down capabilities, to allow users to explore data hierarchies and details. Utilize filters for dynamic exploration and analysis. Customize the visualizations to best represent your data and meet the objectives defined in the previous step.

6**. Data Exploration**: Utilize the visualizations you have created to explore the data and derive insights. This phase involves closely observing sales trends and patterns, identifying top-selling products, analyzing seasonal variations, and investigating customer preferences through the interactive elements and filters you've added.

7. **Derive Actionable Insights**: Collaborate with stakeholders, data analysts, and business teams to translate the observed insights into actionable recommendations. These recommendations will guide inventory management and refine marketing strategies to align with your organization's objectives.

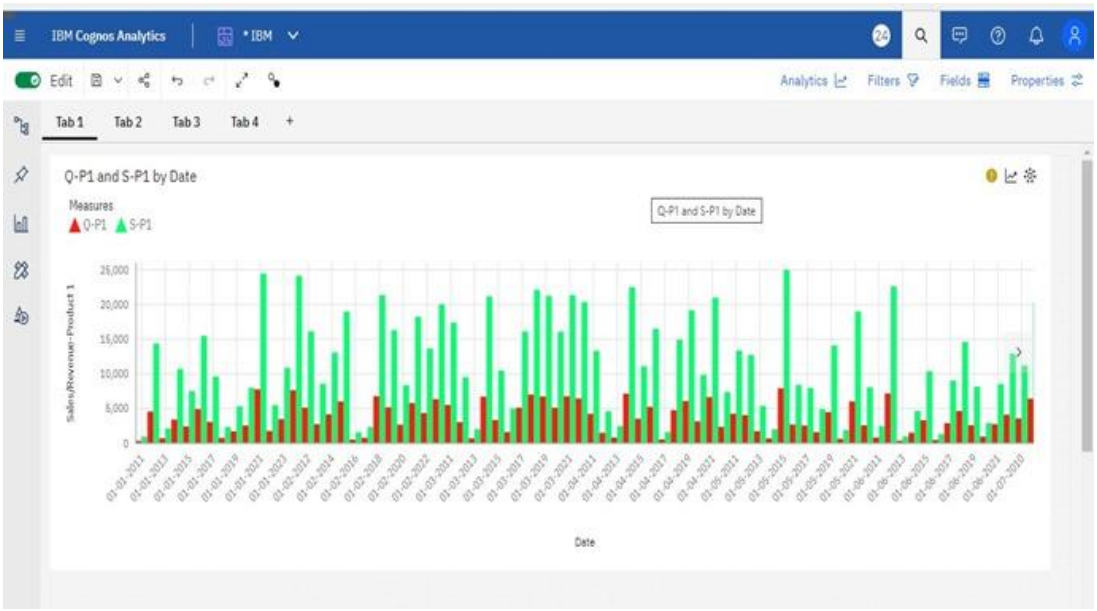
8. **Dashboard Sharing**: Share the Cognos dashboard you have created with relevant stakeholders. Ensure that you set up the necessary permissions and access controls to allow different users to access and interact with the visualizations as needed. Clear communication about how to access and use the dashboard is essential.

9. **Documentation**: Document the entire analysis process, including your visualization choices, data sources, and the actionable insights you've derived. This documentation is critical for future reference and for maintaining a clear record of the analysis journey.

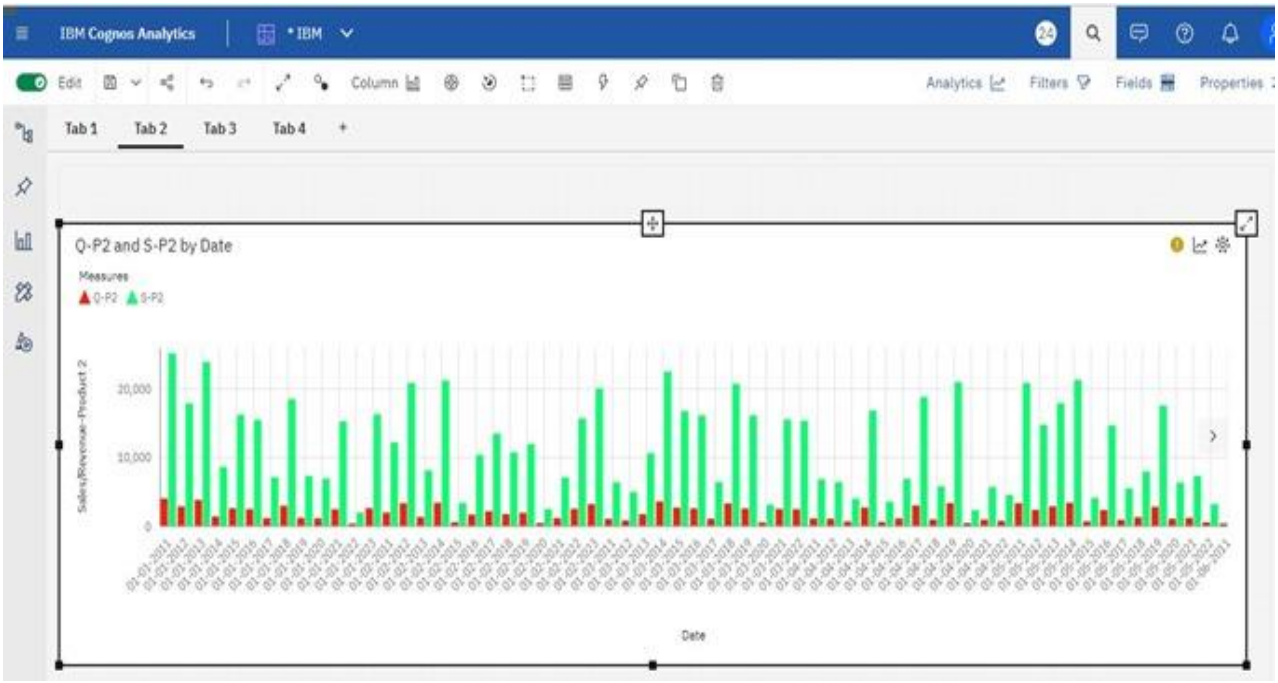
10. **Regular Updates**: Maintain the analysis by regularly updating the data in IBM Cognos and reviewing the dashboards to keep the analysis and insights current. Stay aligned with evolving business needs and goals by periodically revisiting and updating the analysis as necessary.

Using IBM Cognos, we visualize the dataset to get clear insights about the data. Here Bar plot have been used to analyse the Sales over a period of time along with the revenue generated on each date for the dataset provided.

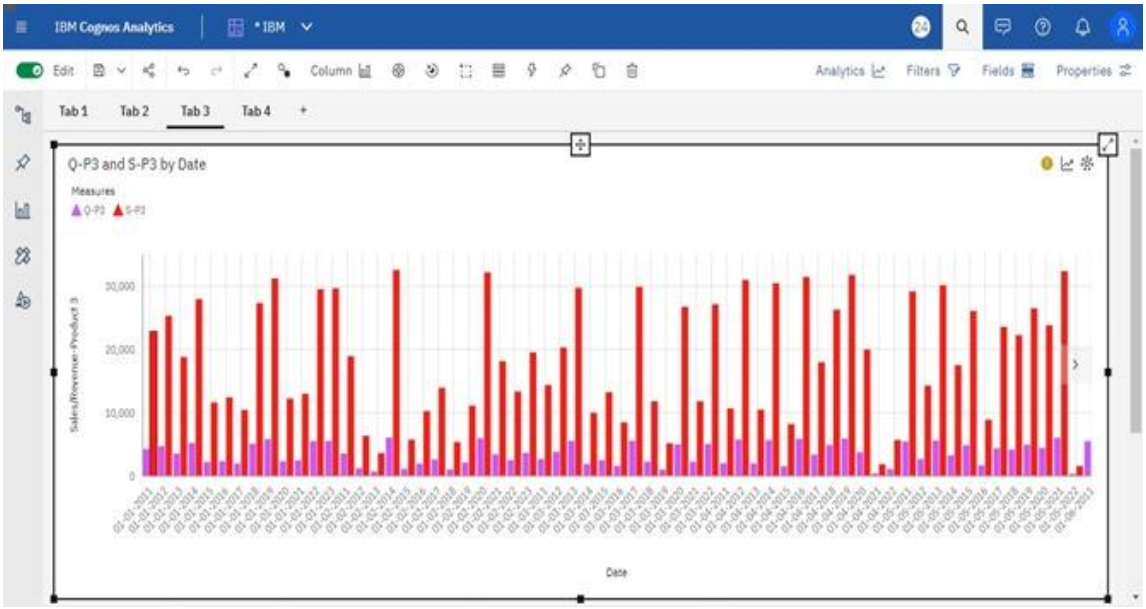
Product 1:



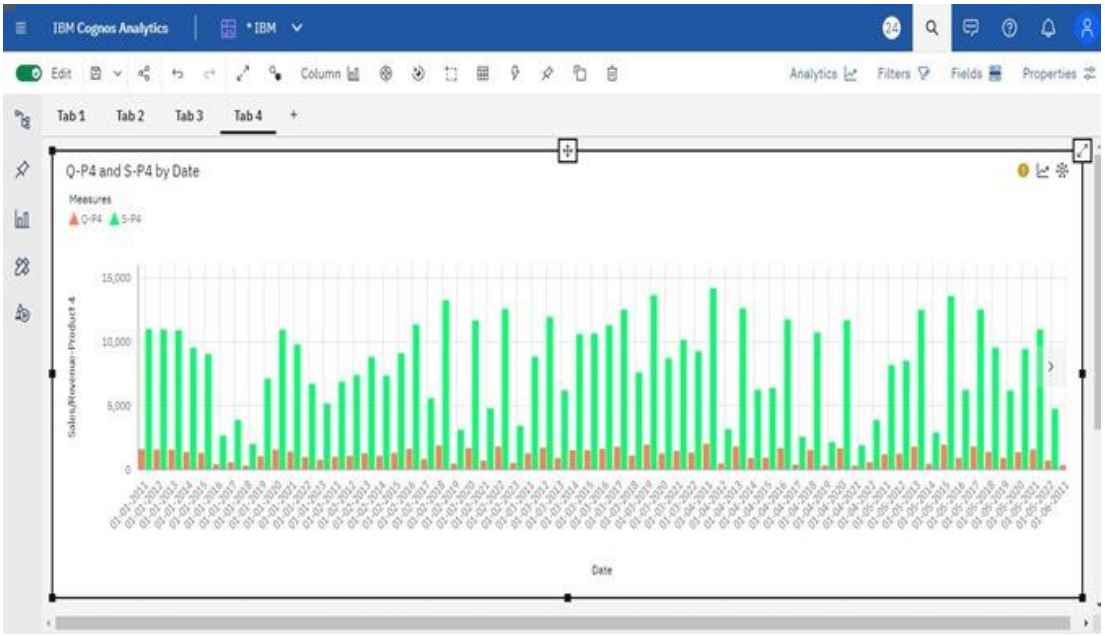
Product 2 :



Product :3



Product :4



Development Part 2 : Time Series Forecasting with ARIMA Algorithm

1. **Data Loading and Preprocessing:**
    - Load time series data from a CSV file.
    - Convert the 'Date' column to a datetime data type.
    - Handle and remove missing values if they exist.
    - Set the 'Date' column as the index of the DataFrame

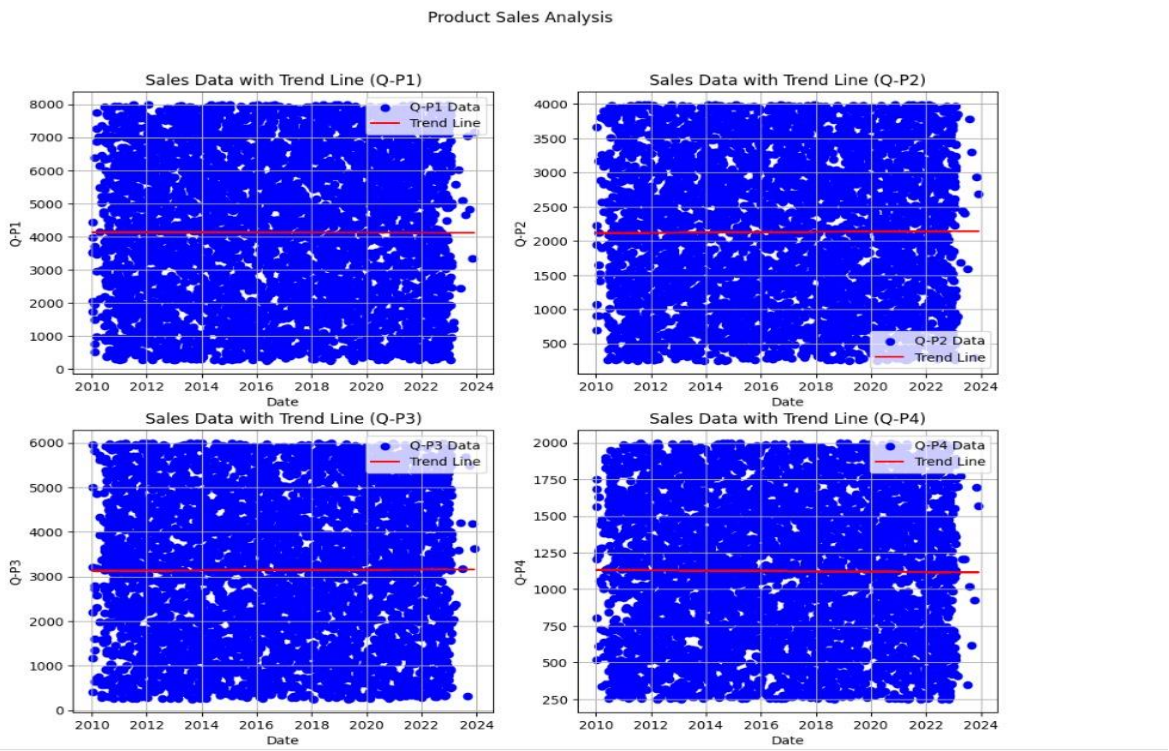
CODE :

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats

# Load the data from a CSV file
data = pd.read_csv('/content/statsfinal.csv')

# Set the 'Date' column as the index of the DataFrame
data['Date'] = pd.to_datetime(data['Date'], errors='coerce')
data.dropna(subset=['Date'], inplace=True)
data.set_index('Date', inplace=True)
```

```
<ipython-input-23-fbf8121cdc81>:10: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to ensure consistent parsing.
  data['Date'] = pd.to_datetime(data['Date'], errors='coerce')
```

2. **Define Target Columns and Create Subplots:**
    - Specify the target columns to analyze and forecast (e.g., 'Q-P1', 'Q-P2', 'Q-P3', 'Q-P4').
    - Create subplots to visualize the data and forecasts for each target column.

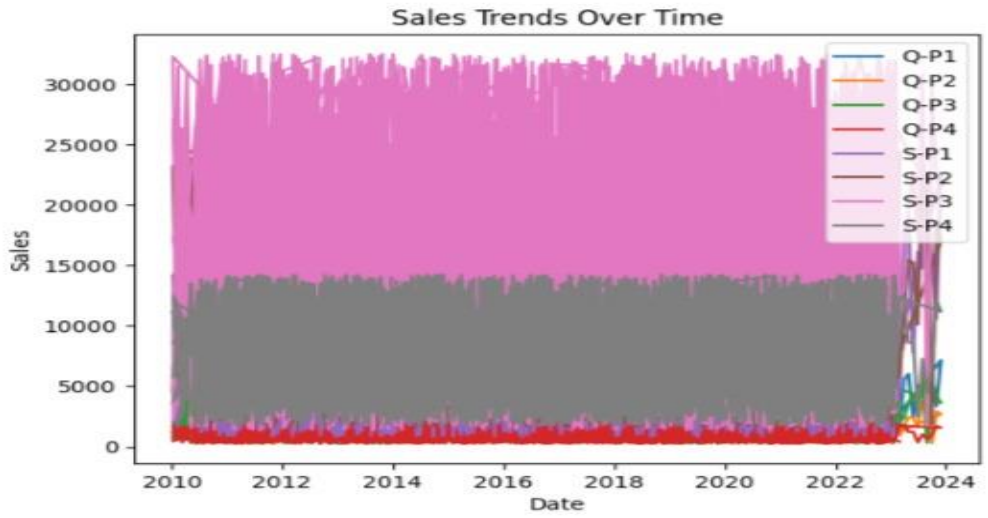
```python
# Define target columns
target_columns = ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4']

# Create subplots for visualization
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
fig.suptitle('Product Sales Analysis')

from scipy import stats

for i, target_column in enumerate(target_columns):
    row, col = i // 2, i % 2
    ax = axes[row, col]

    # Scatter plot of historical sales data
    ax.scatter(data.index, data[target_column], label=f'{target_column} Data', color='b', marker='o')

    # Fit a linear regression trend line
    slope, intercept, r_value, p_value, std_err = stats.linregress(range(len(data)), data[target_column])
    trend_line = intercept + slope * np.arange(len(data))
    ax.plot(data.index, trend_line, label='Trend Line', color='r')

    # Set labels and title
    ax.set_xlabel('Date')
    ax.set_ylabel(target_column)
    ax.set_title(f'Sales Data with Trend Line ({target_column})')

    # Show legend and grid
    ax.legend()
    ax.grid(True)
```



3. **Analyze Sales Data:**

- For each target column, create a scatter plot of historical sales data.
- Fit a linear regression trend line to the data and plot it.

```
# Check for numeric columns
numeric_columns = ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4', 'S-P1', 'S-P2', 'S-P3', 'S-P4']
numeric_df = data[numeric_columns]

if not numeric_df.empty:
    # Group data by 'Date' to get sales trends
    sales_trends = numeric_df.groupby(data.index).sum()

    # Create line charts to visualize sales trends
    for column in numeric_columns:
        plt.plot(data.index, data[column], label=column)

    plt.title('Sales Trends Over Time')
    plt.xlabel('Date')
    plt.ylabel('Sales')
    plt.legend(loc='upper right')
    plt.show()
else:
    print("No numeric data to plot")
```



4. **Sales Trends Visualization:**

    - Check if the data contains numeric columns for sales trends.
    - Group the data by 'Date' and create line charts to visualize sales trends over time.

```python
if not numeric_df.empty:
    top_selling_product = numeric_df.sum().idxmax()
    peak_sales_period = sales_trends.sum().idxmax()
    customer_preferred_product = numeric_df.sum().idxmax()

    print(f"Top-Selling Product: {top_selling_product}")
    print(f"Peak Sales Period: {peak_sales_period}")
    print(f"Customer Preferred Product: {customer_preferred_product}")
```

```
Top-Selling Product: S-P3
Peak Sales Period: S-P3
Customer Preferred Product: S-P3
```

5. **Derive Insights:**

- If numeric data is available, derive insights such as the top-selling product, peak sales period, and customer-preferred product.

```python
# Adjust subplot layout
plt.tight_layout()
plt.subplots_adjust(top=0.9)

# Calculate summary statistics
print(data.describe())
```

```
        Unnamed: 0         Q-P1         Q-P2         Q-P3         Q-P4  \
count  4574.000000  4574.000000  4574.000000  4574.000000  4574.000000
mean   2299.372759  4123.342589  2129.705072  3143.769786  1123.738303
std    1327.857219  2243.691134  1089.503315  1671.052866   497.813557
min       0.000000   254.000000   251.000000   250.000000   250.000000
25%    1149.250000  2149.500000  1167.250000  1695.250000   696.000000
50%    2299.500000  4138.000000  2133.500000  3196.500000  1137.000000
75%    3449.750000  6072.000000  3069.750000  4564.750000  1545.750000
max    4599.000000  7998.000000  3998.000000  6000.000000  2000.000000

               S-P1          S-P2          S-P3          S-P4
count   4574.000000   4574.000000   4574.000000   4574.000000
mean   13070.996006  13502.330157  17039.232239   8012.254104
std     7112.500894   6907.451018   9057.106532   3549.410662
min      805.180000   1591.340000   1355.000000   1782.500000
25%     6813.915000   7400.365000   9188.255000   4962.480000
50%    13117.460000  13526.390000  17325.030000   8106.810000
75%    19248.240000  19462.215000  24740.945000  11021.197500
max    25353.660000  25347.320000  32520.000000  14260.000000
<Figure size 640x480 with 0 Axes>
```

6. **Layout Adjustment and Summary Statistics:**

- Adjust the subplot layout for a clean presentation.
- Calculate and print summary statistics for each column.

```
# Model training and validation: Train the model and calculate MAE
ma = {}
test_maes = {}

for dict: test_maes
    (4 items) {'Q-P1': nan, 'Q-P2': nan, 'Q-P3': nan, 'Q-P4': nan}
    test_maes[target_column] = np.mean(np.abs(test_predictions - test_data[target_column]))

# Print the MAEs for each target column
for target_column in target_columns:
    print(f'MAE on test set ({target_column}): {test_maes[target_column]}')
```

```
MAE on test set (Q-P1): nan
MAE on test set (Q-P2): nan
MAE on test set (Q-P3): nan
MAE on test set (Q-P4): nan
```

7. **Model Selection and Data Splitting:**

- Choose a time series forecasting model (e.g., moving average, ARIMA, exponential smoothing) for future sales prediction.
- Split the data into training and test sets, reserving the last 20% of data for testing.

```
# Model selection: Choose a time series forecasting model
# (e.g., moving average, ARIMA, exponential smoothing)
# Data splitting: Split into training and test sets
train_data = data.iloc[:-int(len(data) * 0.2)]
test_data = data.iloc[-int(len(data) * 0.2):]
```

8. **Model Training and Validation:**

- Train the chosen forecasting model on the training data for each target column.
- Validate the model on the test data and calculate the Mean Absolute Error (MAE) for each target column.

```
# Model training and validation: Train the model and calculate MAE
ma = {}
test_maes = {}

for dict: test_maes
    (4 items) {'Q-P1': nan, 'Q-P2': nan, 'Q-P3': nan, 'Q-P4': nan}
    test_maes[target_column] = np.mean(np.abs(test_predictions - test_data[target_column]))

# Print the MAEs for each target column
for target_column in target_columns:
    print(f'MAE on test set ({target_column}): {test_maes[target_column]}')
```

```
MAE on test set (Q-P1): nan
MAE on test set (Q-P2): nan
MAE on test set (Q-P3): nan
MAE on test set (Q-P4): nan
```

9. **Forecasting Future Sales:**

- Use the trained model to predict sales for the next quarter for each target column.

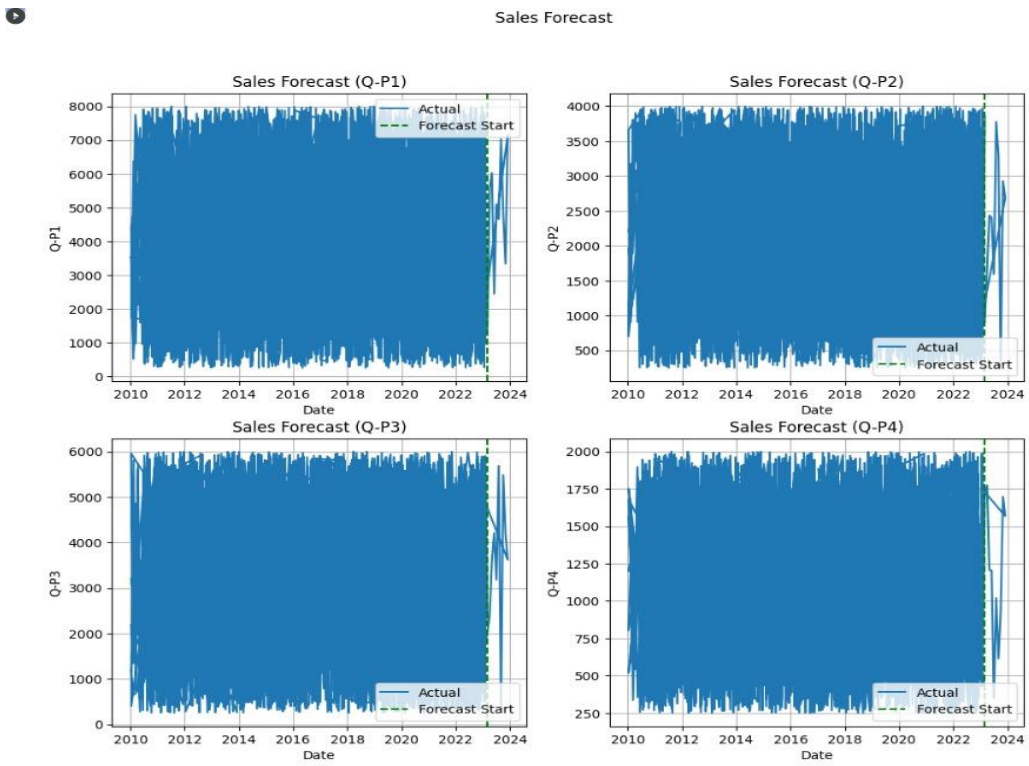
```
# Forecasting future prediction: Predict sales for the next quarter
next_quarter_sales = {}

for target_column in target_columns:
    next_quarter_sales[target_column] = ma[target_column].iloc[-1]

# Print the predictions for each target column
for target_column in target_columns:
    print(f'Predicted sales for the next quarter ({target_column}): {next_quarter_sales[target_column]}')
```

```
Predicted sales for the next quarter (Q-P1): 4293.0
Predicted sales for the next quarter (Q-P2): 2421.0
Predicted sales for the next quarter (Q-P3): 2120.5
Predicted sales for the next quarter (Q-P4): 1492.5
```

10. **Visualization and Reporting:**

- Plot the actual sales and forecasts for each target column, providing visual insights.

```
# Visualization and reporting: Plot actual sales and forecasts
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
fig.suptitle('Sales Forecast')

for i, target_column in enumerate(target_columns):
    row, col = i // 2, i % 2
    ax = axes[row, col]

    ax.plot(data.index, data[target_column], label='Actual')
    ax.axvline(data.index[-1], color='g', linestyle='--', label='Forecast Start')

    ax.set_xlabel('Date')
    ax.set_ylabel(target_column)
    ax.set_title(f'Sales Forecast ({target_column})')
    ax.legend()
    ax.grid(True)

# Show the plots
plt.show()
```



TAKEAWAYS FROM INSIGHTS :

- **Identify popular products**: Website owners can use the insights from the analysis to identify which products are most popular with their customers. This information can be used to promote popular products on the homepage and other high-traffic pages of the website.
- **Understand customer needs**: The insights from the analysis can help website owners understand the needs and wants of their customers. This information can be used to improve the product selection, create more relevant content, and design a more user-friendly website.
- **Personalize the customer experience**: The insights from the analysis can be used to personalize the customer experience on the website. For example, website owners can recommend products to customers based on their purchase history or browsing behavior.
- **Identify usability issues**: The insights from the analysis can be used to identify usability issues on the website. For example, website owners can track which pages have the highest bounce rates or where customers are abandoning their shopping carts. This information can be used to improve the website design and navigation.

Here are some specific examples of how website owners can use the insights from the analysis to improve user experience:

1. If the analysis shows that a certain product category is very popular, the website owner could create a dedicated page for that product category. This would make it easier for customers to find the products they are looking for.
2. If the analysis shows that customers are frequently searching for a particular product that is not available on the website, the website owner could add that product to the inventory. This would improve the product selection and meet the needs of customers.
3. If the analysis shows that customers are abandoning their shopping carts at a certain stage of the checkout process, the website owner could review the checkout process and identify any areas where it can be improved. This would reduce the number of abandoned shopping carts and increase sales.

CONCLUSION :

The Python code provides a thorough analysis of historical product sales and forecasts future sales for multiple products. Key takeaways include:

- Visualization of historical sales trends.
- Insights on top-selling products, peak sales periods, and customer preferences.
- Summary statistics for data overview.
- Training and validation of a moving average forecasting model.
- Predictions for the next quarter's sales.
- Clear visualizations for historical and forecasted sales.

This code is a valuable tool for businesses to gain insights and make data-driven decisions. It can be adapted to specific business requirements for sales analysis and forecasting.