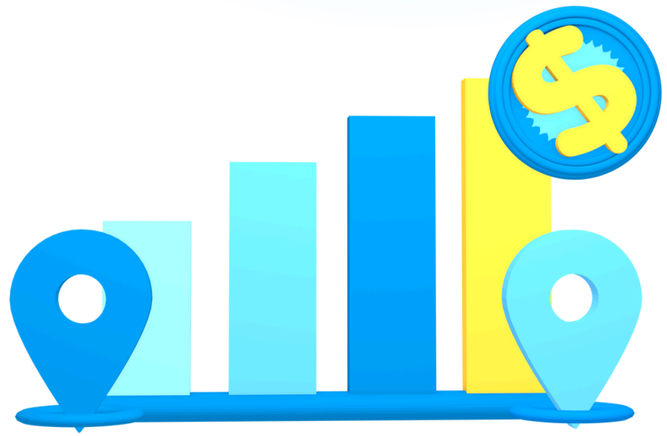
Data analysis case study

Problem Statement:

TechGear Solutions, based in Mumbai, India, is a leading supplier of computer hardware to various clients, including Prime Electronics stores across the country. However, the sales head, Arjun Mehta, is grappling with several key challenges: Based on the dataset, you need to make decisions on several problems.



Dataset analysis:

The given dataset consists of 5 tables:
Products, Customers, Markets, Date and Transactions

These tables contain the individual information about the different types of products, their customers, the different markets where these products are sold to the consumers, the date on which these items are sold, and the sales quantity and amount of products brought by different customers from different places.

All this information is scattered across the 5 tables and we need to merge them in order to find high level insights on revenue and trends.

Univariate Exploratory Data Analysis (EDA):

Tools used:
Python libraries- Pandas, Matplotlib, Seaborn

Methodology:
All the five tables were analyzed individually to understand the distributions in each individual column and to gain any useful insights from them.
Individual graphs were plotted to see the patterns in each column using python libraries.

Code:

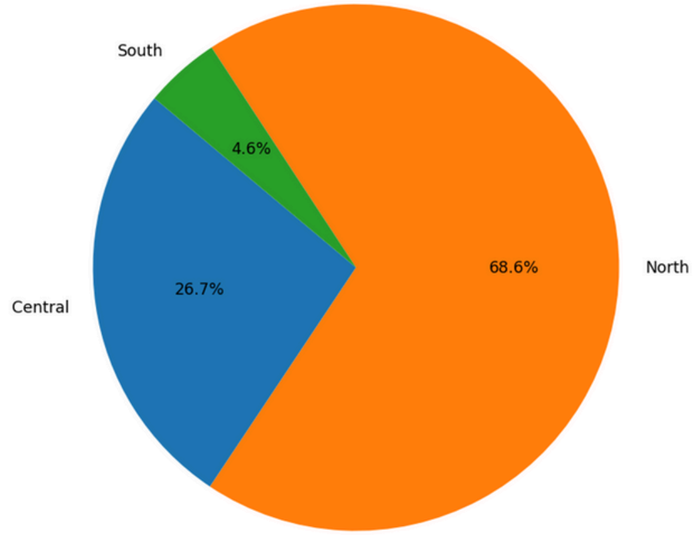
```
import pandas as pd

products = pd.read_csv("products.csv")
customers = pd.read_csv("customers.csv")
markets = pd.read_csv("markets.csv")
date = pd.read_csv("date.csv")
transactions = pd.read_csv("transactions.csv")

from ydata_profiling import ProfileReport
p1 = ProfileReport(products, title="Integrated Data Report", explorative=True)
p1.to_file("p1.html")
p2 = ProfileReport(transactions, title="Integrated Data Report", explorative=True)
p2.to_file("p2.html")
p3 = ProfileReport(date, title="Integrated Data Report", explorative=True)
p3.to_file("p3.html")
p4 = ProfileReport(customers, title="Integrated Data Report", explorative=True)
p4.to_file("p4.html")
p5 = ProfileReport(markets, title="Integrated Data Report", explorative=True)
p5.to_file("p5.html")
```

Insights from EDA :

Table 1: Product type:

Table 2: Customer type:

Table 3: Market Zones:



Data preprocessing:

All the tables were merged into one so that data can be analysed using all the tables to find insights.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

products = pd.read_csv("products.csv")
customers = pd.read_csv("customers.csv")
markets = pd.read_csv("markets.csv")
dates = pd.read_csv("date.csv")
transactions = pd.read_csv("transactions.csv")

transactions['order_date'] = pd.to_datetime(transactions['order_date'])
dates['date'] = pd.to_datetime(dates['date'])

merged_df = transactions.merge(products, on="product_code", how="left") \
                        .merge(customers, on="customer_code", how="left") \
                        .merge(markets, on="market_code", how="left") \
                        .merge(dates, left_on="order_date", right_on="date", how="left")
```

	product_code	customer_code	market_code	order_date	...	cy-date	year	month_name	date_yy_mmm
0	Prod001	Cus001	Mark001	2017-10-10	...	2017-10-01	2017	October	2022-10-17
1	Prod001	Cus002	Mark002	2018-05-08	...	2018-05-01	2018	May	2022-05-18
2	Prod002	Cus003	Mark003	2018-04-06	...	2018-04-01	2018	April	2022-04-18
3	Prod002	Cus003	Mark003	2018-04-11	...	2018-04-01	2018	April	2022-04-18
4	Prod002	Cus004	Mark003	2018-06-18	...	2018-06-01	2018	June	2022-06-18

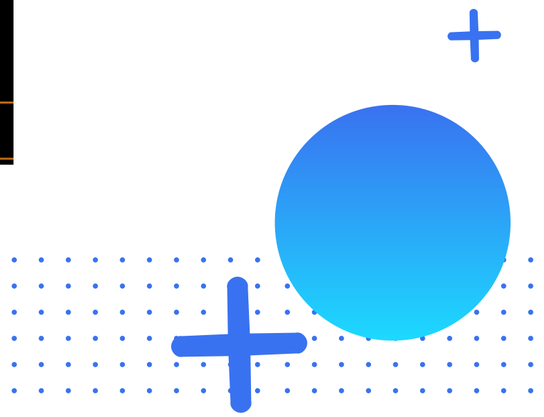
[5 rows x 17 columns]

Multivariate analysis insights:

Finding the daily, monthly and yearly sales

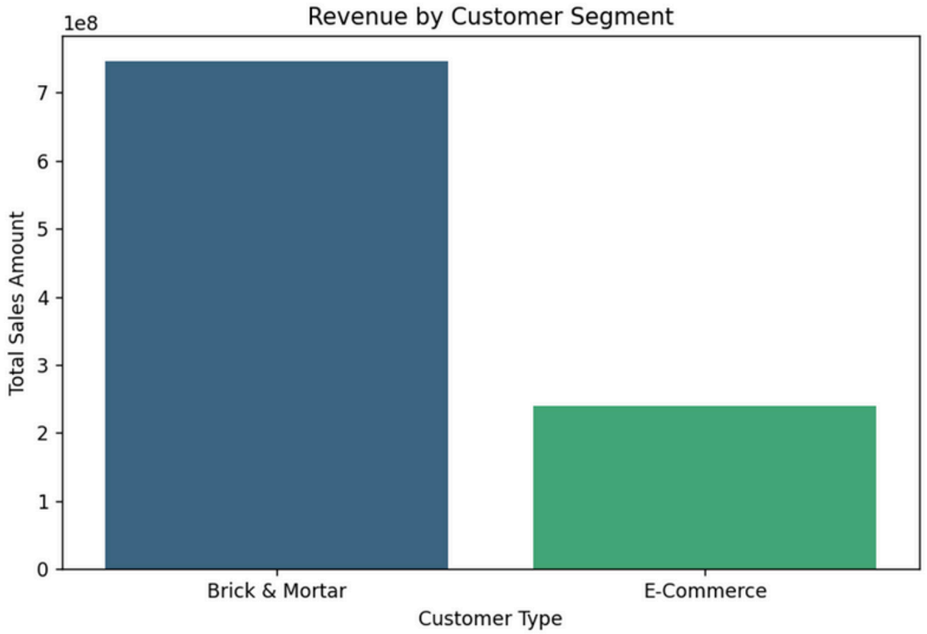
```
daily_sales = merged_df.groupby('order_date')['sales_amount'].sum().reset_index()
daily_sales.set_index('order_date', inplace=True)
plt.figure(figsize=(12,6))
plt.plot(daily_sales.index, daily_sales['sales_amount'], label='Daily Sales')
plt.title('Daily Sales Amount')
plt.xlabel('Date')
plt.ylabel('Sales Amount')
plt.legend()
plt.show()
```

```
df=transactions.merge(date, left_on='order_date', right_on='date', how='left')
df['daily_sales'] = df.groupby('order_date')['sales_amount'].transform('sum')
df['monthly_sales']=df.groupby('month_name')['sales_amount'].transform('sum')
df['yearly_sales']=df.groupby('year')['sales_amount'].transform('sum')
```

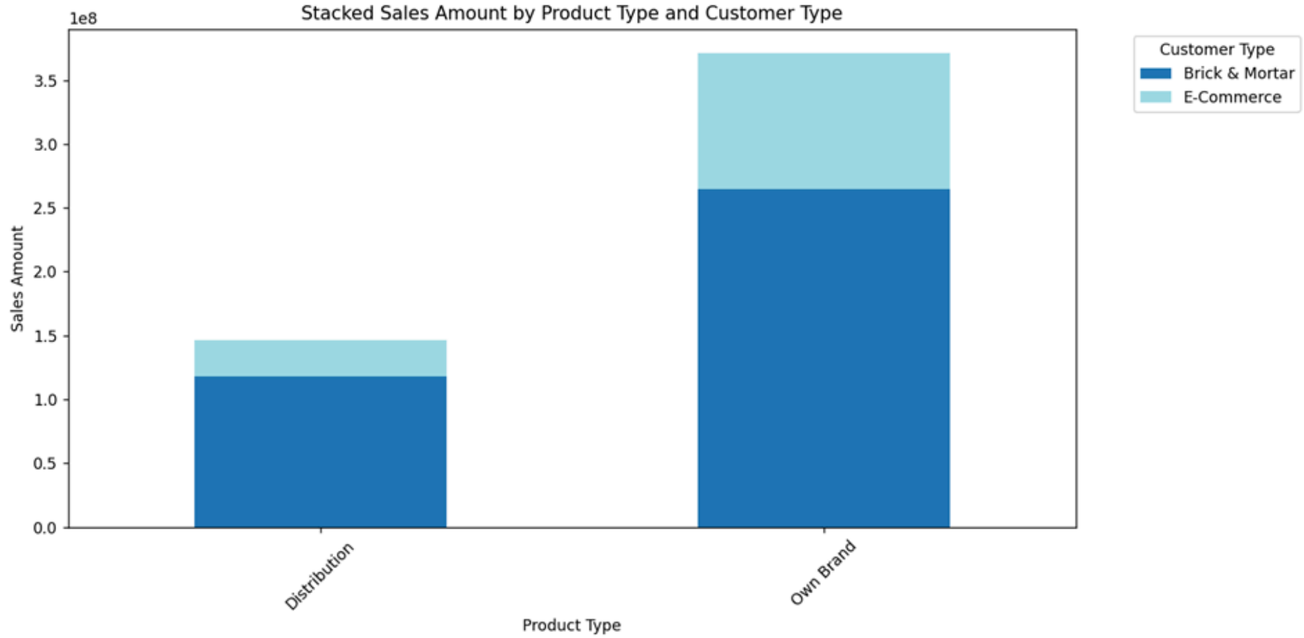


Multivariate analysis insights:

Customer revenue by segment:

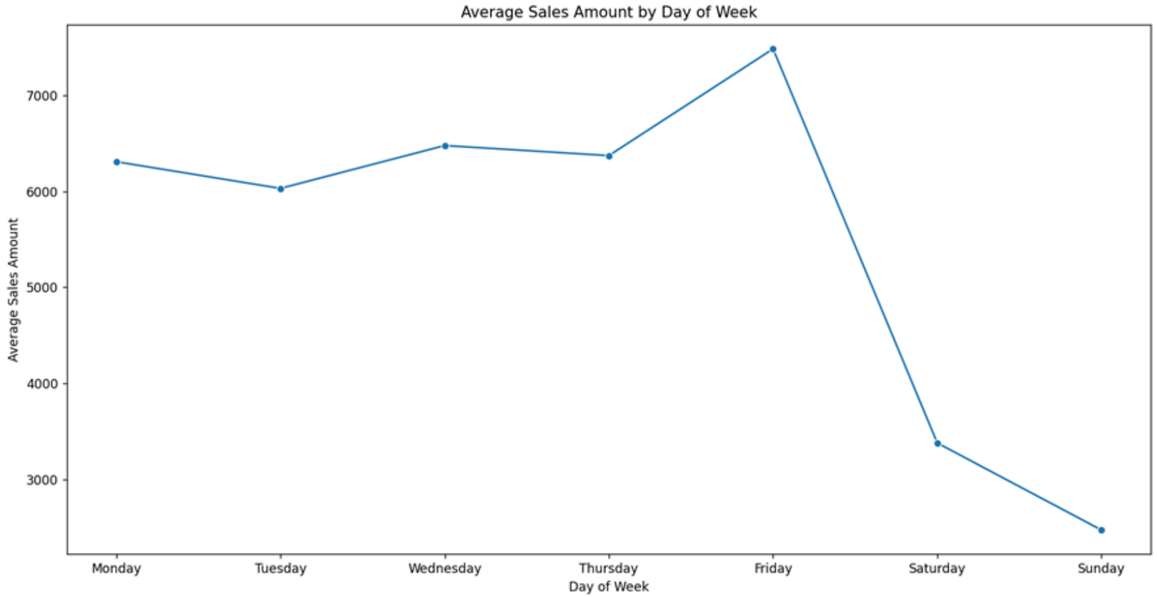


Sales amount by customer type:

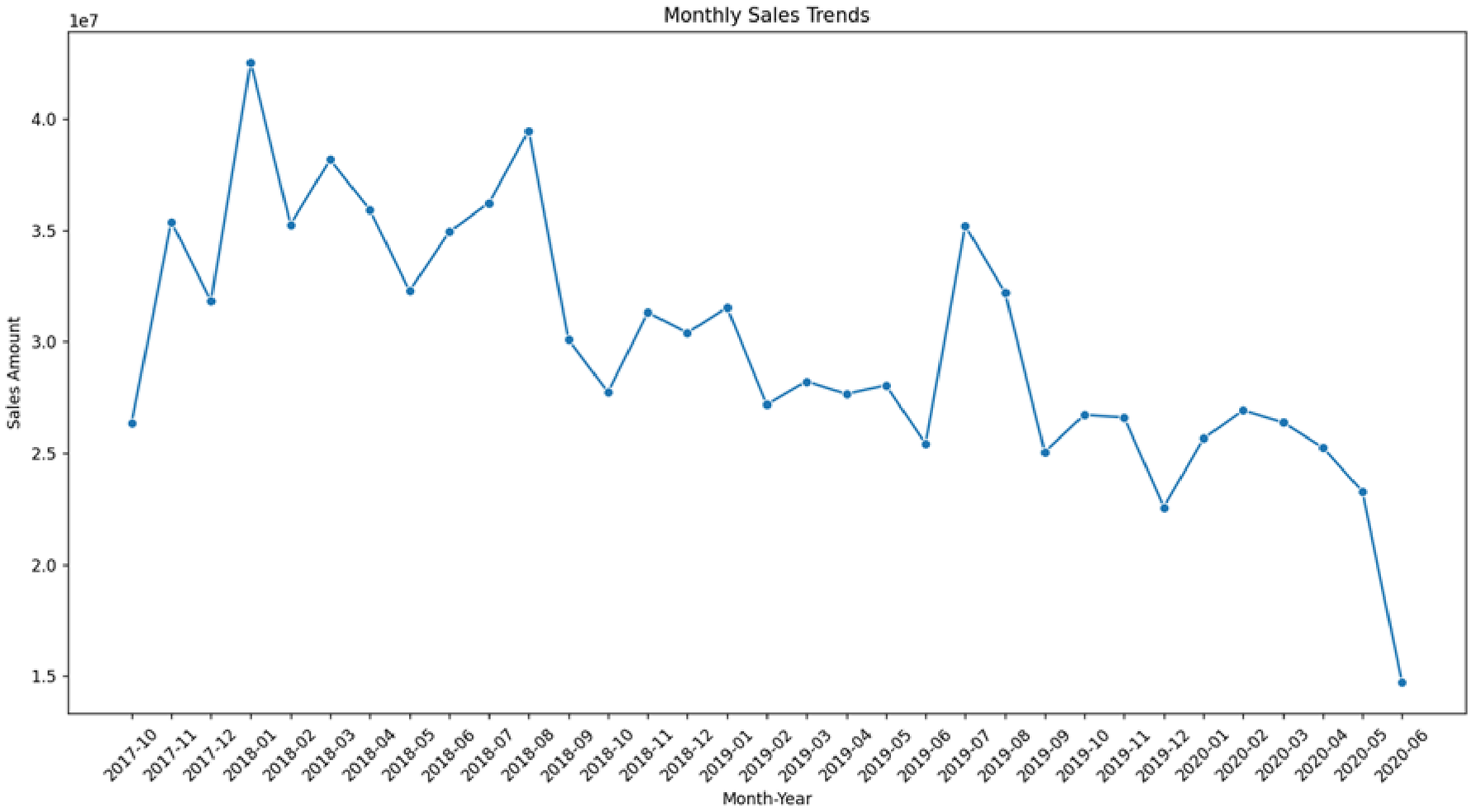


Number of transactions per zone:

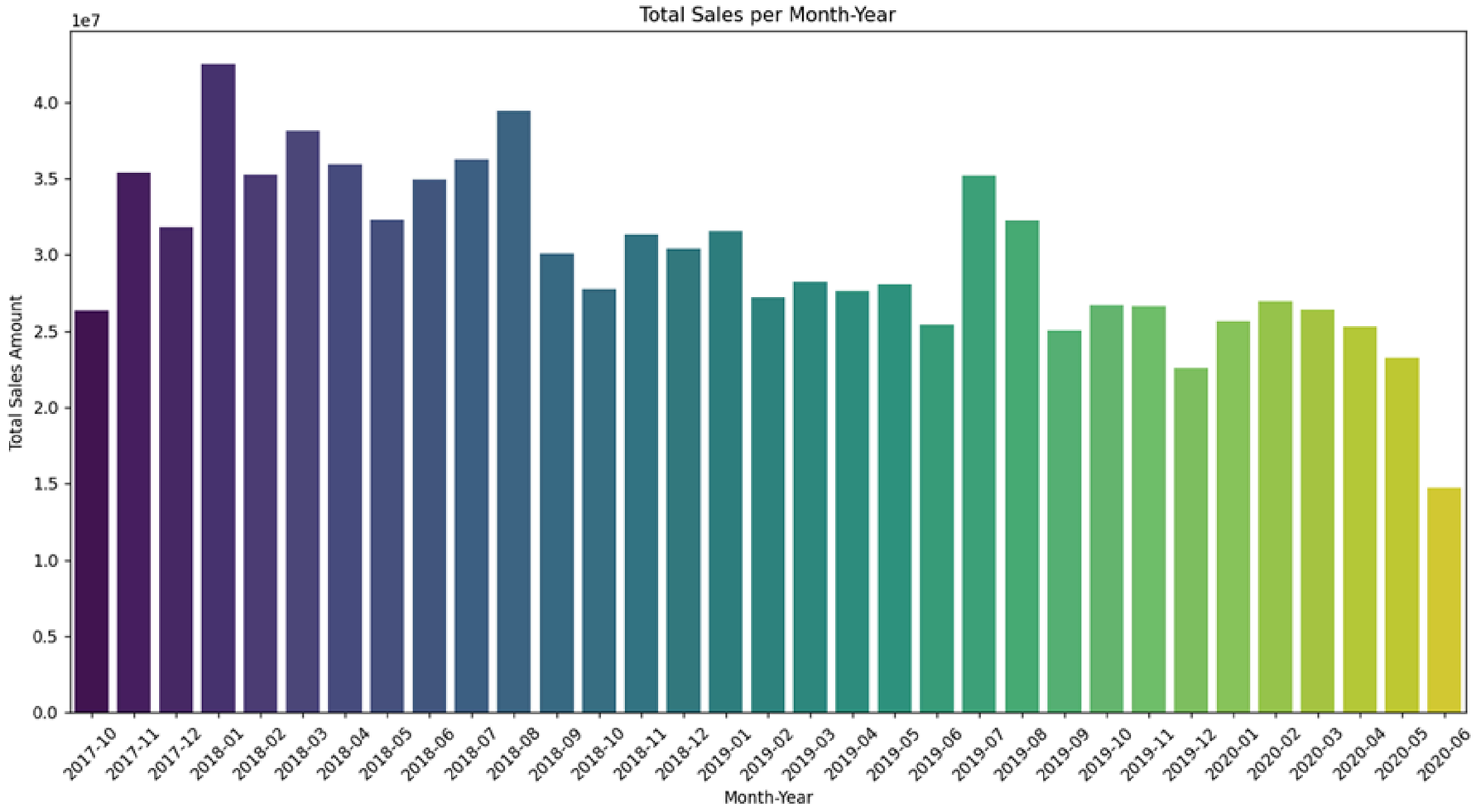
Sale trend throughout week:



Monthly sales trends:



Monthly sales trends:



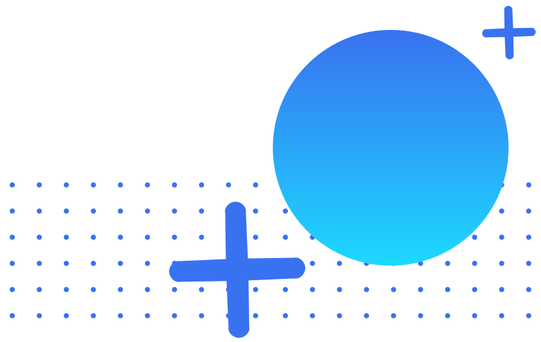
Question 1:

How can we mathematically differentiate between stable sales trends and volatile market fluctuations to ensure our strategic decisions are based on long-term patterns rather than short-term noise?

Solution: Time Series Analysis

The only way to differentiate between stable sales trends and volatile market fluctuations is to use a time series analysis. This means that we can group the date into time frames of maybe a month or a year, and then we can check the patterns in the data- so the impact of noise will be minimized.

To ensure that the decisions we make are based on the stable trends we can calculate the moving averages and find the current standard deviations- if it is beyond a certain limit we can deem it as noise and not be affected by it.

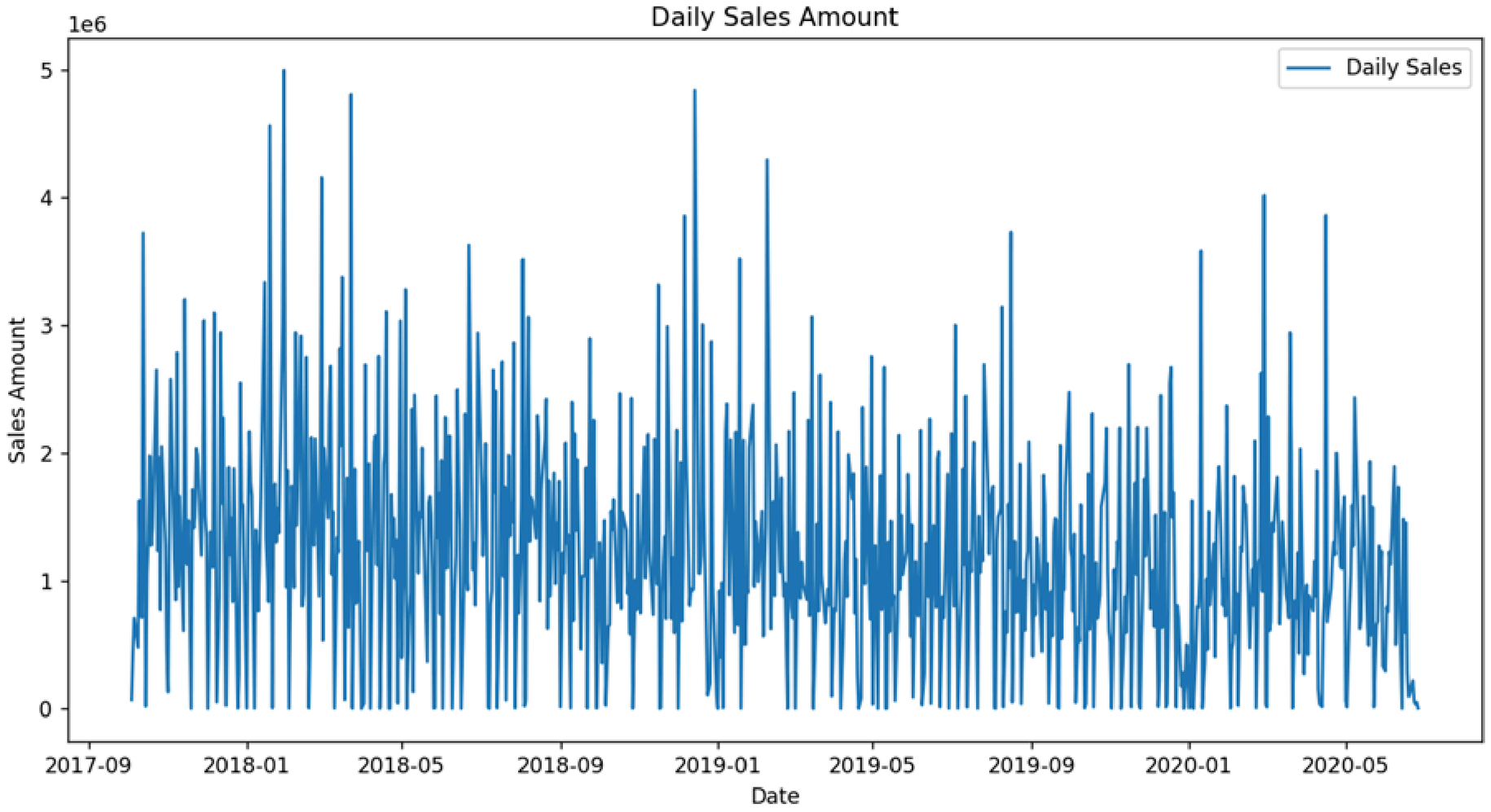


Graphical Solutions:

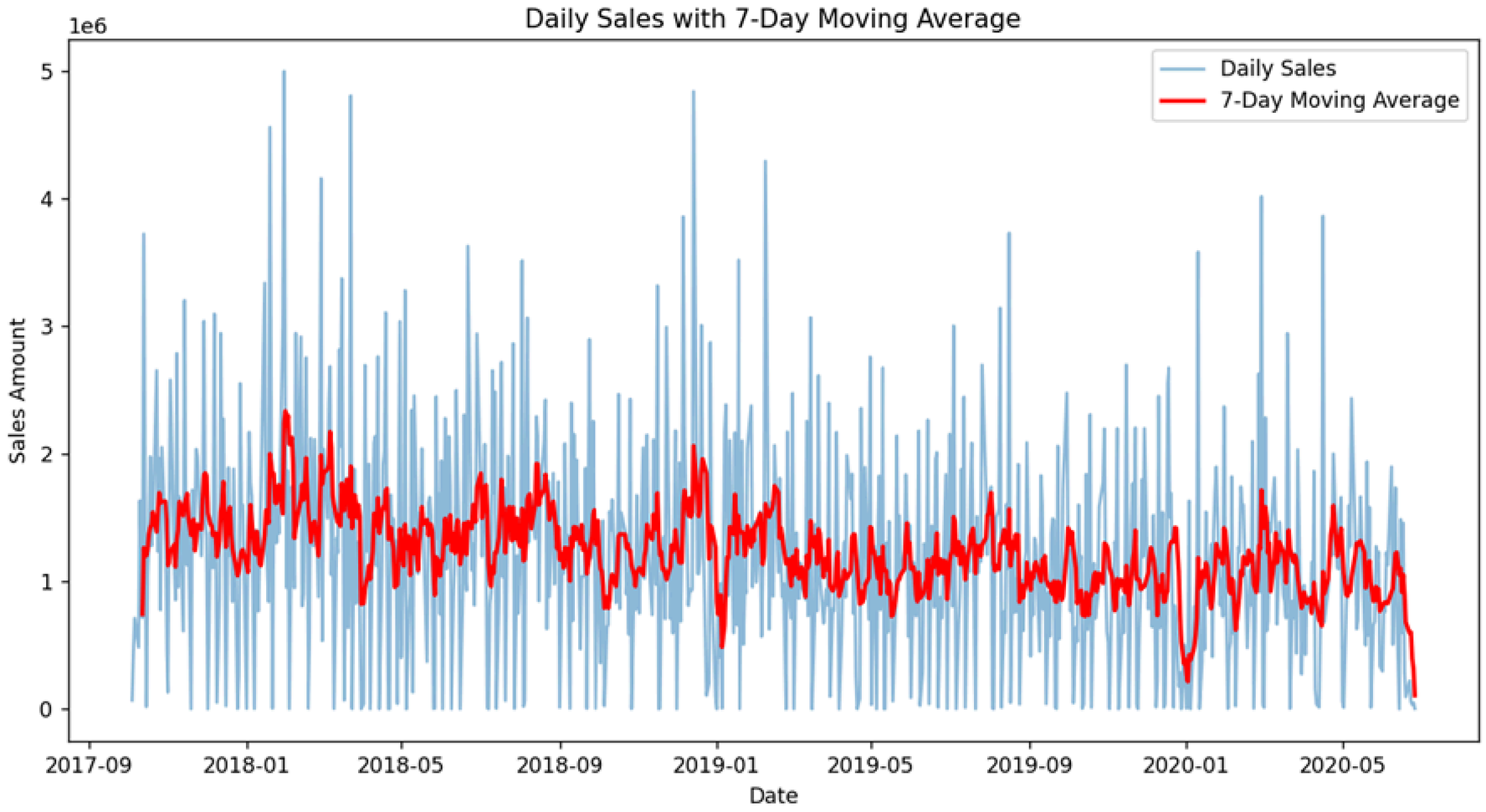
Code:

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
transactions = pd.read_csv("transactions.csv")
dates = pd.read_csv("date.csv")
transactions['order_date'] = pd.to_datetime(transactions['order_date'])
dates['date'] = pd.to_datetime(dates['date'])
merged_df = transactions.merge(dates, left_on='order_date', right_on='date', how='left')
daily_sales = merged_df.groupby('order_date')['sales_amount'].sum().reset_index()
daily_sales.set_index('order_date', inplace=True)
plt.figure(figsize=(12,6))
plt.plot(daily_sales.index, daily_sales['sales_amount'], label='Daily Sales')
plt.show()
daily_sales['7_day_MA'] = daily_sales['sales_amount'].rolling(window=7).mean()
plt.figure(figsize=(12,6))
plt.plot(daily_sales.index, daily_sales['sales_amount'], label='Daily Sales', alpha=0.5)
plt.plot(daily_sales.index, daily_sales['7_day_MA'], label='7-Day Moving Average', color='red', linewidth=2)
plt.show()
```

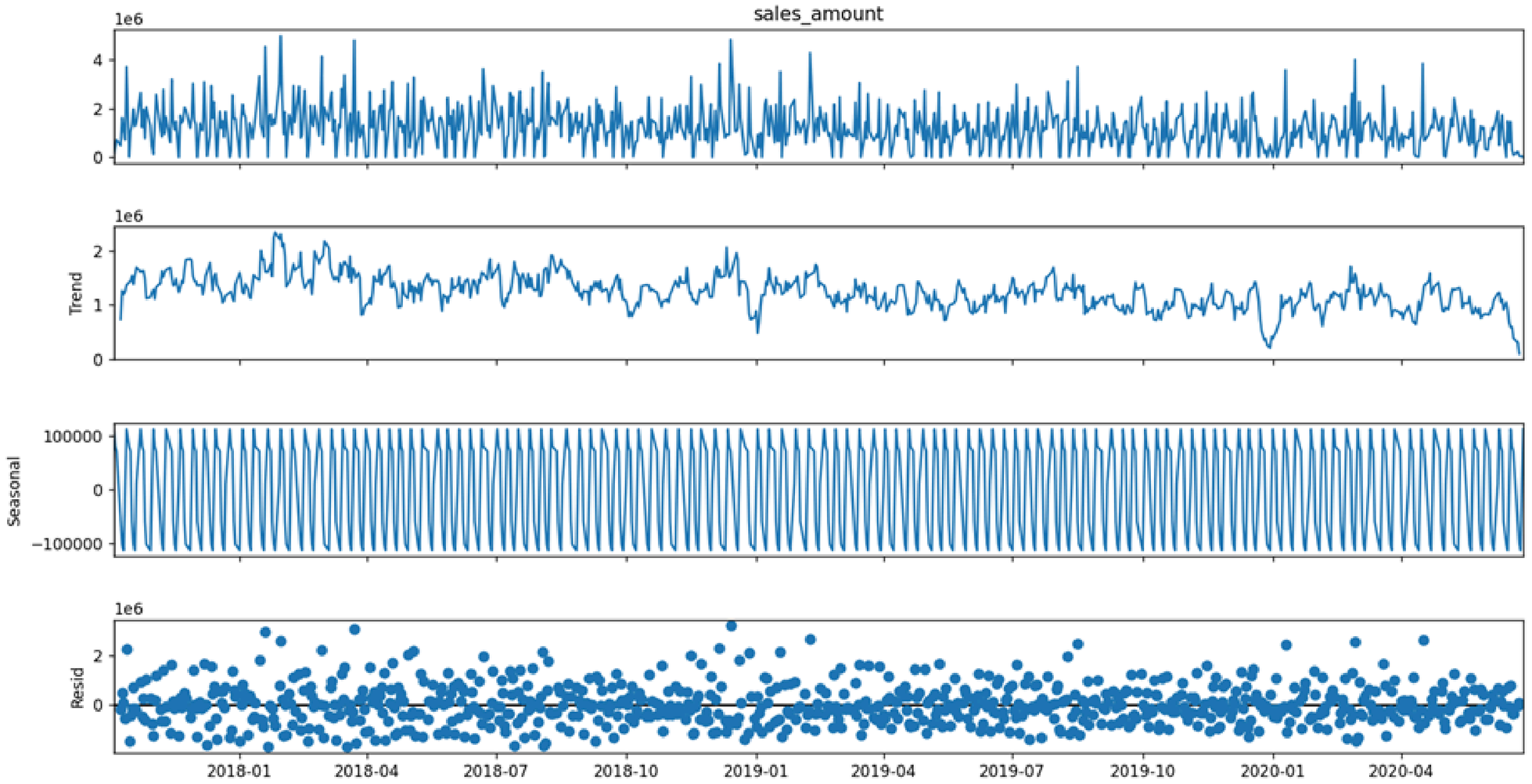
Daily sales graph without time series analysis:



Daily sales graph with time series analysis:



Analysis of trend v/s noise:



Question 2:

Market Pulse or Data Mirage? Given the rapidly evolving market, how would you design a system to differentiate between actual sales trends and short-term fluctuations, ensuring that business strategies align with genuine market demand?

Solution: Dual averages with anomaly detection

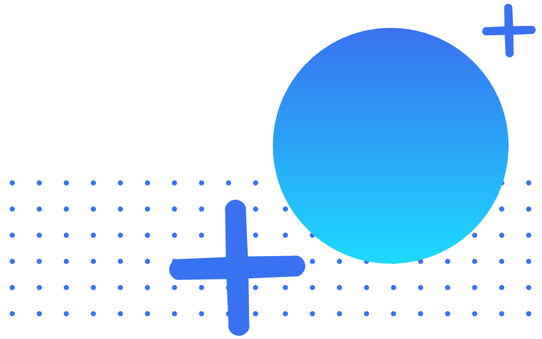
Looking at sales over time (daily, weekly, or monthly) helps us to spot patterns. Use of averages and stats check if sales are steady or bouncing around a lot. Breaking sales data into long-term trends and short-term ups and downs to see the trends.
Comparing short-term and long-term sales averages—big differences might mean temporary changes or noise, which can be avoided. Also, building a system (like a dashboard) that tracks sales in real-time and sends alerts is also a useful strategy.

Graphical Solutions:

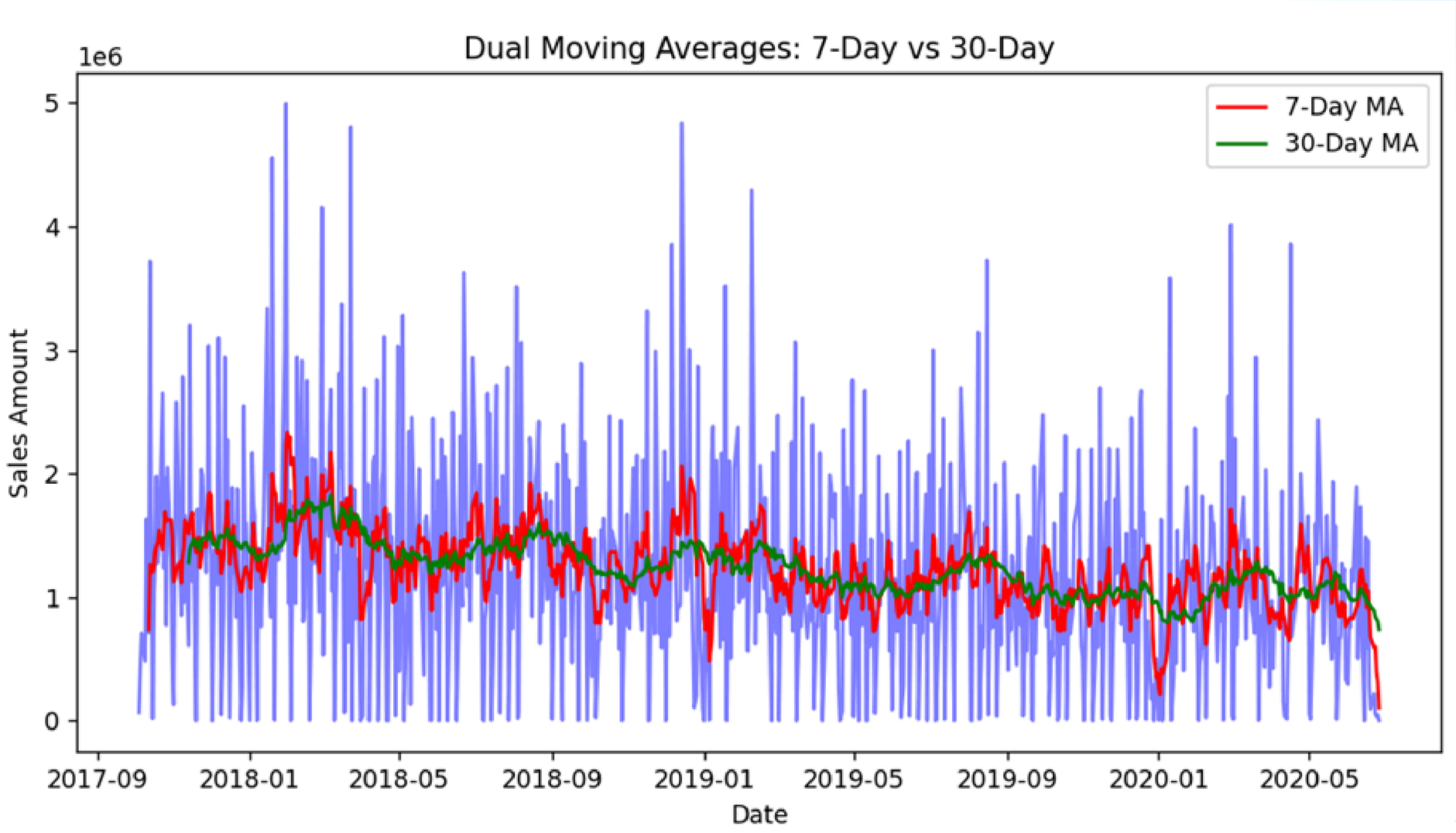
Code:

```
daily_sales['30_day_MA'] = daily_sales['sales_amount'].rolling(window=30).mean()
plt.figure(figsize=(10,5))
plt.plot(daily_sales.index, daily_sales['sales_amount'], color='blue', alpha=0.5)
plt.plot(daily_sales.index, daily_sales['7_day_MA'], color='red', label='7-Day MA')
plt.plot(daily_sales.index, daily_sales['30_day_MA'], color='green', label='30-Day MA')
plt.title("Dual Moving Averages: 7-Day vs 30-Day")
plt.xlabel("Date")
plt.ylabel("Sales Amount")
plt.legend()
plt.show()
```

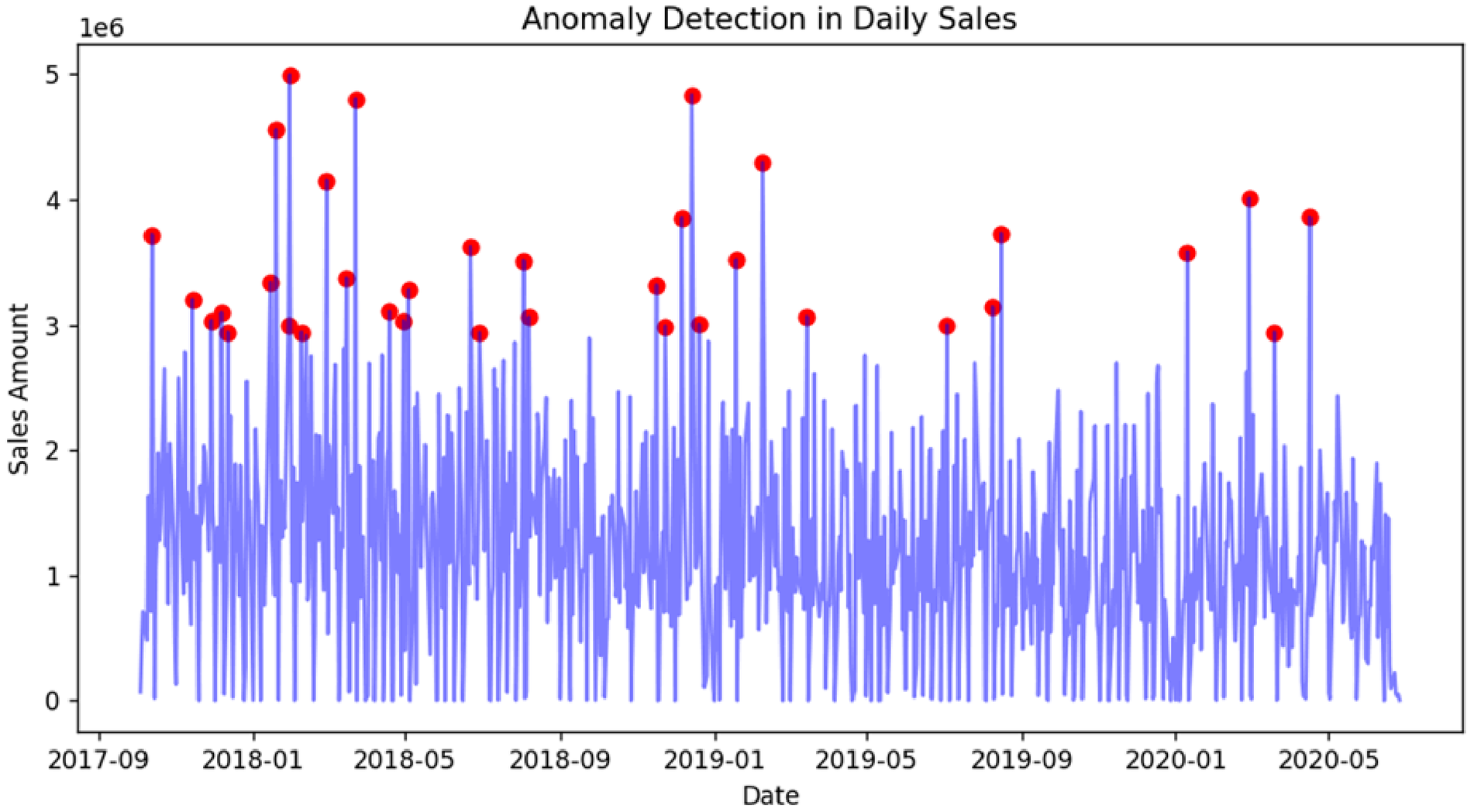
```
daily_sales['z_score'] = (daily_sales['sales_amount'] - daily_sales['sales_amount'].mean())
anomalies = daily_sales[np.abs(daily_sales['z_score']) > 2]
plt.figure(figsize=(10,5))
plt.plot(daily_sales.index, daily_sales['sales_amount'], color='blue', alpha=0.5)
plt.scatter(anomalies.index, anomalies['sales_amount'], color='red')
plt.title("Anomaly Detection in Daily Sales")
plt.xlabel("Date")
plt.ylabel("Sales Amount")
plt.show()
```



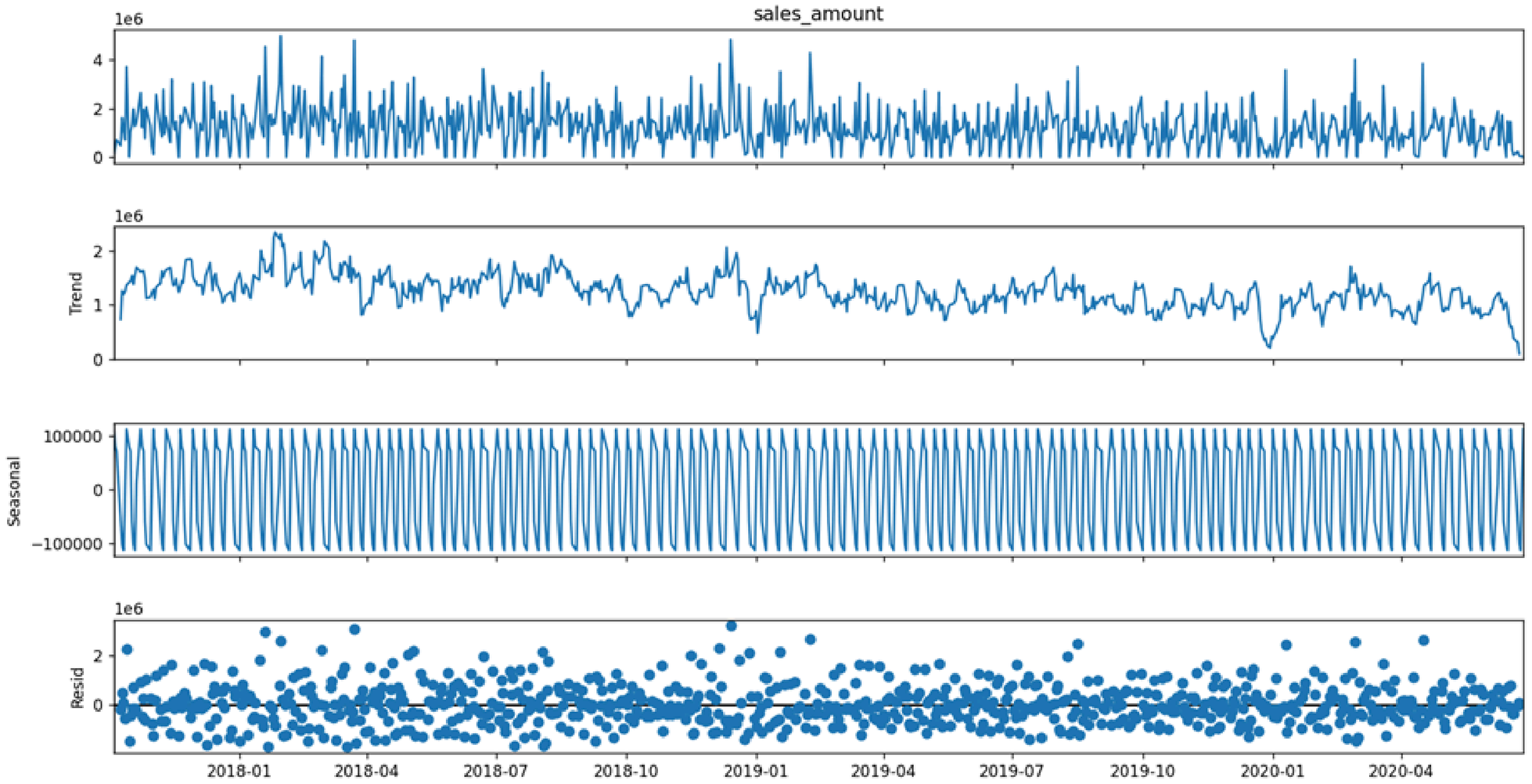
Daily sales graph with dual averages:



Anomaly detection in sales:



Graph of Seasonal v/s Trend v/s Residual:



Practical solutions based on insights:

- Promotions timed with seasons: Sales and promotions should be planned to match busy times of the year, like holidays or certain months when more purchases are made. Special deals should be offered during slower times to encourage purchases.

- Growth trend followed: If business growth is observed, stock, team, and marketing efforts should be increased to meet demand. If sales are flat or declining, products or prices should be reviewed to spark growth.

- Unexpected changes reacted to: Sudden changes in sales should be monitored, as they may be caused by events like competitor sales or supply problems. Adjustments in pricing or flash deals should be made quickly.

- Customers targeted specifically: Knowledge about customers should be used to offer the right deals at the right times, especially during busy or slow seasons.

- Regional differences checked: sales patterns in different regions should be observed. Adjustments should be made based on performance in various areas.

Question 3:

Which customer segments contribute the highest revenue, and how can we identify customer clusters that require different pricing, discounting, or marketing strategies?

Solution: RFM segments, KMeans Clustering

To find out which customers bring in the most money, three simple factors are seen: how recently they made a purchase (Recency), how often they buy (Frequency), and how much money they spend (Monetary). By collecting this information for every customer, a unique profile is created for each one.

The clusters have been formed of the customers by using the famous KMeans method (it is a machine learning algorithm) which divides the given data into a number of segments based on the parameters provided. Here clustering has been done on the basis of this unique profile of RFM parameters.

Code:

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

cust = pd.read_csv("customers.csv")
sales = pd.read_csv("transactions.csv")
data = sales.merge(cust, on="customer_code", how="left")
data['order_date'] = pd.to_datetime(data['order_date'])
today = data['order_date'].max() + pd.Timedelta(days=1)
rfm_table = data.groupby('customer_code').agg({
    'order_date': lambda d: (today - d.max()).days,
    'customer_code': 'count',
    'sales_amount': 'sum'
}).rename(columns={'order_date': 'Recency', 'customer_code': 'Frequency', 'sales_amount': 'M
rfm_table = rfm_table.merge(cust[['customer_code', 'customer_type']], on="customer_code", ho
seg_rev = rfm_table.groupby('customer_type')['Monetary'].sum().reset_index()
print(seg_rev)
scaler = StandardScaler()
scaled_vals = scaler.fit_transform(rfm_table[['Recency', 'Frequency', 'Monetary']])
kmeans = KMeans(n_clusters=3, random_state=42)
rfm_table['Cluster'] = kmeans.fit_predict(scaled_vals)
print(rfm_table.sample(15))
```

Divided segments:

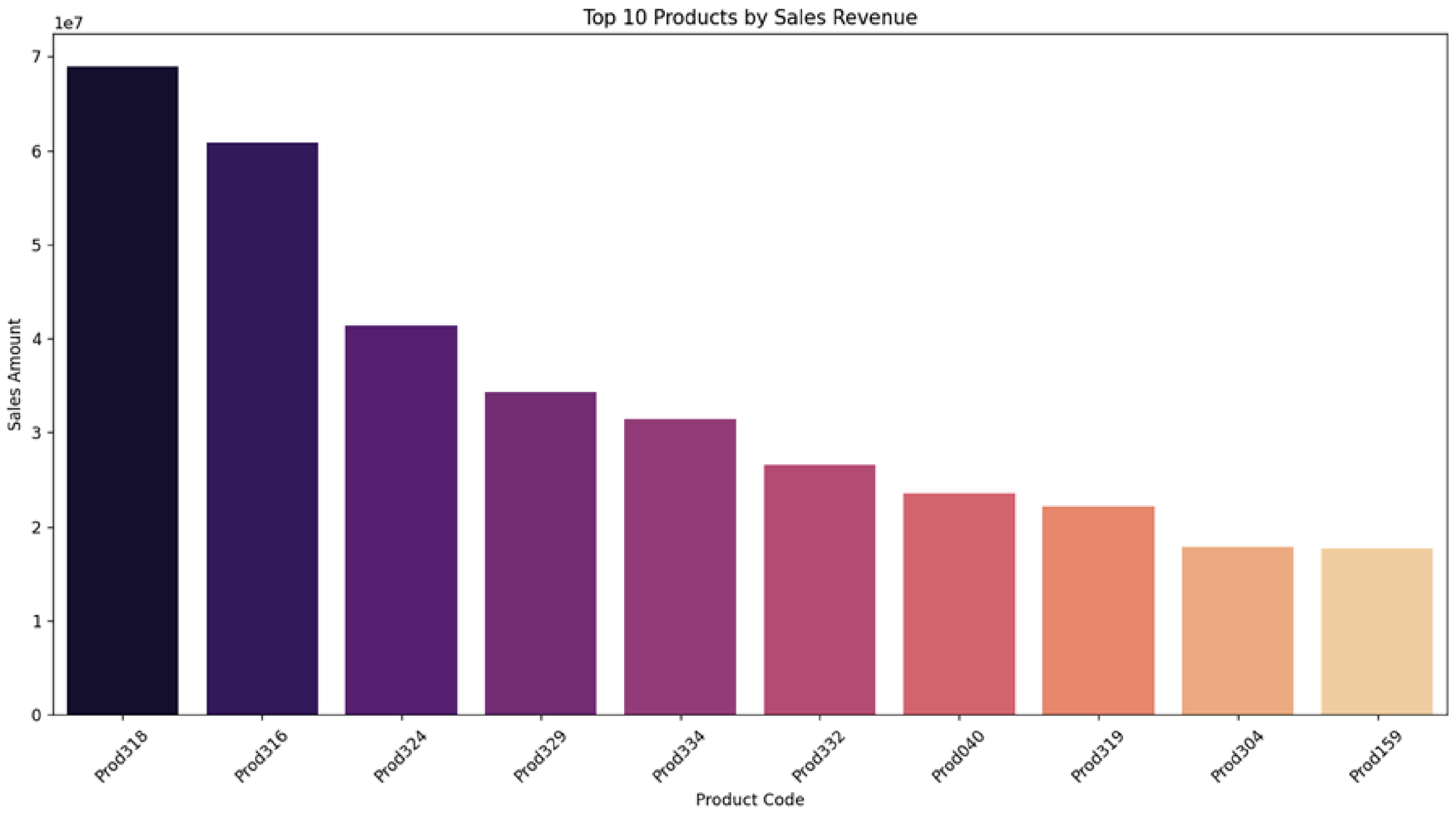
	customer_code	Recency	Frequency	Monetary	customer_type	Cluster
21	Cus022	3	4792	49644189	E-Commerce	1
17	Cus018	4	3143	1868461	Brick & Mortar	1
32	Cus033	12	1446	6068432	E-Commerce	1
34	Cus035	11	2755	5230158	E-Commerce	1
19	Cus020	3	17340	43916981	E-Commerce	2
26	Cus027	4	2204	31771997	E-Commerce	1
0	Cus001	5	4983	28833717	Brick & Mortar	1
25	Cus026	8	1223	3342051	E-Commerce	1
4	Cus005	10	20003	45258250	Brick & Mortar	2
7	Cus008	9	1387	21198041	Brick & Mortar	1
8	Cus009	11	410	1333393	Brick & Mortar	1
2	Cus003	1	9559	49175285	Brick & Mortar	1
22	Cus023	5	2147	4966707	E-Commerce	1
18	Cus019	2	11726	10310851	Brick & Mortar	1
36	Cus037	10	4226	4183862	E-Commerce	1

Graphical Solutions:

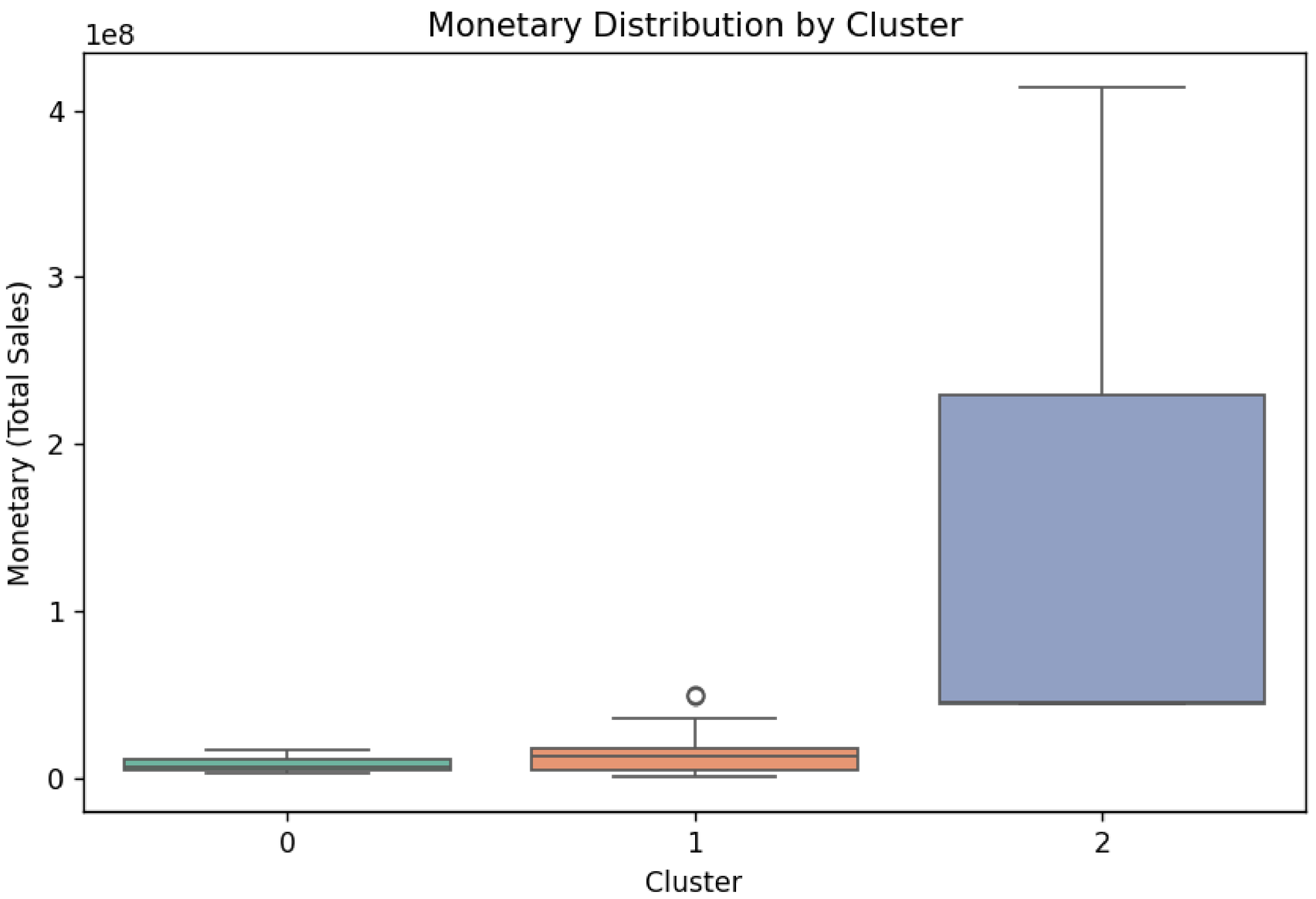
Code:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
top_customers = merged_df.groupby('customer_code')['sales_amount'].sum().reset_index()
top_customers = top_customers.sort_values('sales_amount', ascending=False).head(30)
plt.figure(figsize=(8,5))
sns.barplot(x='customer_code', y='sales_amount', data=top_customers, palette='Greens_d')
plt.title("Top 10 Customers by Revenue")
plt.xlabel("Customer Code")
plt.ylabel("Total Revenue")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
snapshot_date = merged_df['order_date'].max() + pd.Timedelta(days=1)
rfm = merged_df.groupby('customer_code').agg({
    'order_date': lambda x: (snapshot_date - x.max()).days,
    'customer_code': 'count',
    'sales_amount': 'sum'
}).rename(columns={'order_date': 'Recency', 'customer_code': 'Frequency', 'sales_amount': 'M
scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(rfm[['Recency','Frequency','Monetary']])
kmeans = KMeans(n_clusters=3, random_state=42)
rfm['Cluster'] = kmeans.fit_predict(rfm_scaled)
plt.figure(figsize=(8,5))
sns.boxplot(x='Cluster', y='Monetary', data=rfm, palette='Set2')
plt.title("Monetary Distribution by Cluster")
plt.xlabel("Cluster")
plt.ylabel("Monetary (Total Sales)")
plt.show()

# print(merged_df.head())
```

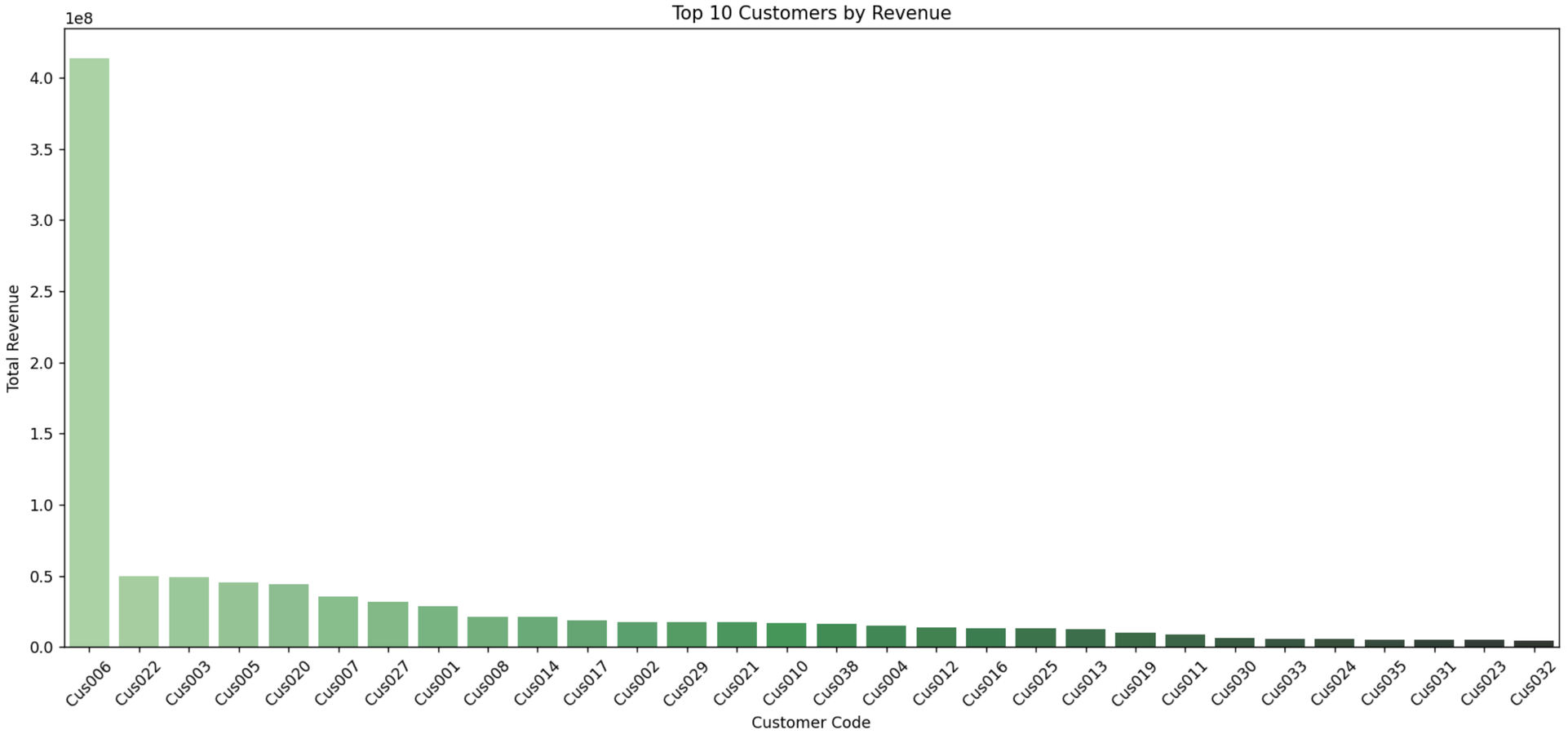
Best selling products:



Monetary contribution of clusters:



Top customers:



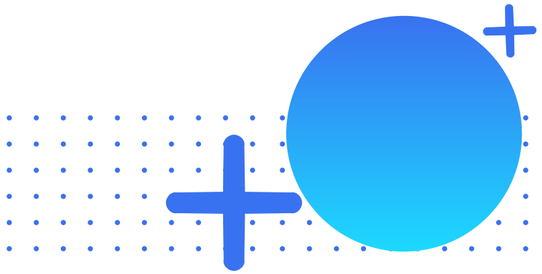
Practical solutions based on insights:

- For the first group, offering special promotions or personalized services is a good way to deepen their loyalty. Since they are responsible for most of the revenue of the company, it is only fair that they get exclusive deals, early access to new products, personalized recommendations, and premium rewards that acknowledge their value.

- For the moderate cluster, encouraging increased spending by offering targeted promotions, bundled deals, and personalized discounts is an effective way.

- For the least spending categories, the focus should be to spark their interest to boost sales by offering introductory offers, re-engagement campaigns, and incentives to try new products.

In this way we can deal with all the customer by the simple method of divide and conquer- by dividing them into clusters and tailoring our marketing strategies according to their type to maximise revenue.



Question 4:

While Excel files have been the backbone of TechGear Solutions' sales data management, they also lead to excessive manual effort. How would you transition the company to a more automated, error-free data consolidation process without disrupting existing workflows?

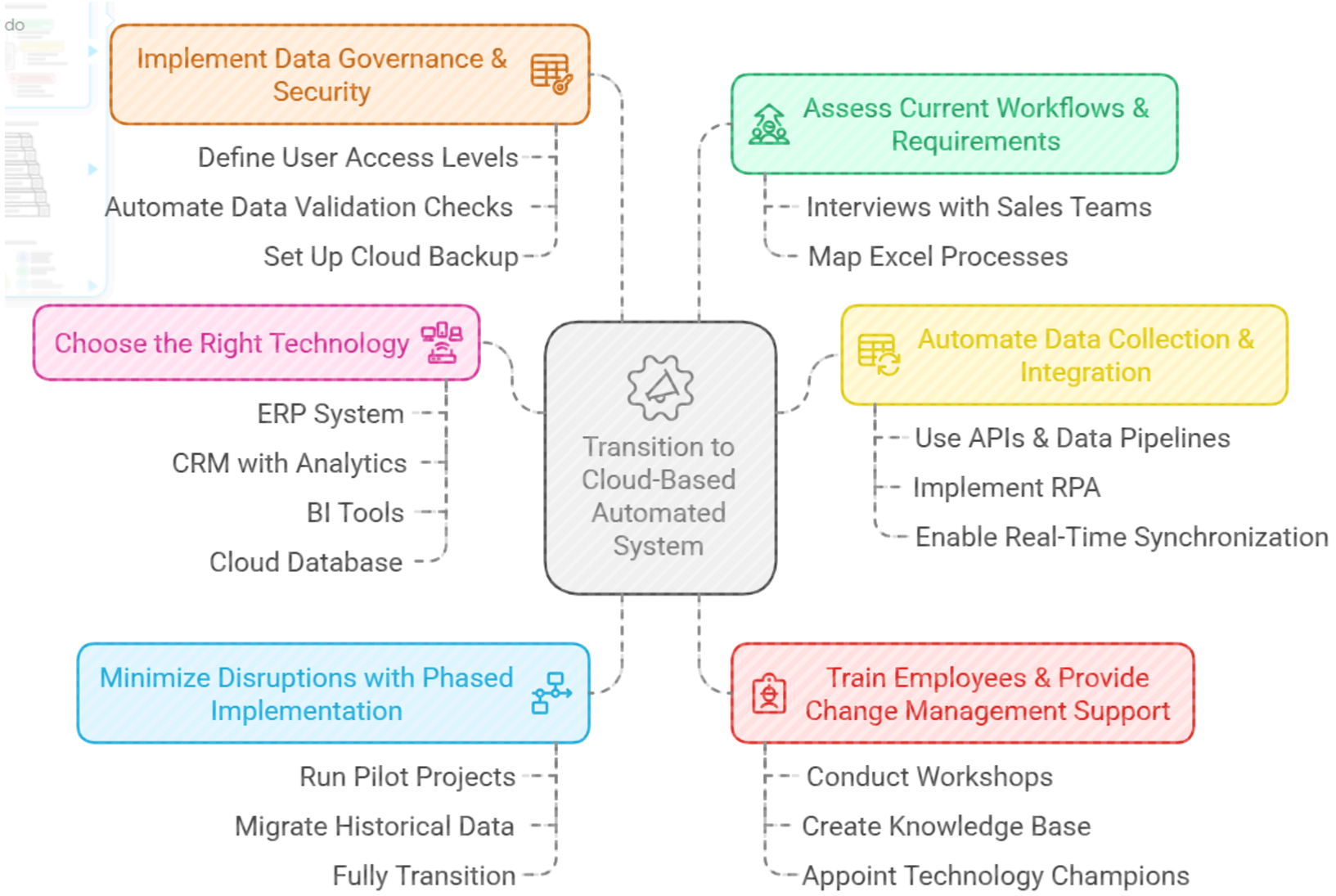
Solution: Pipelining and Automation

Yes, working in excel often becomes tedious and inefficient- and this can be tackled by creating pipelines for the data. Instead of manually combining Excel files, use Python to automatically read all the Excel files from a specific folder, merge them together, and save the combined data in one place (like a CSV file or a database). Then, build a simple dashboard (using a tool like Streamlit or a Business Intelligence tool like Power BI) that helps analyze the trends in important values in real time helps managers quickly see up-to-date information and make decisions based on the latest data.
The creation of ETL pipelines to automate the data extraction, preprocessing and uploading should be set up as a regular cycle.

Key Challenges with Excel-Based Sales Data Management:

- Manual Effort & Inefficiency: Employees spend excessive time updating and consolidating data.
- High Error Rates: Manual data entry leads to frequent mistakes.
- Limited Scalability: Managing large datasets in Excel is cumbersome and slow.
- Data Inconsistency: Different sales teams might use varying formats, leading to confusion.
- Lack of Real-Time Insights: Decision-making is delayed due to lack of automation.

The solution and how it works:



Expected Benefits of automation:

- 80% Reduction in Manual Effort through automation.
- Faster, Data-Driven Decision-Making with real-time dashboards.
- Improved Accuracy & Standardization across sales teams.
- Scalability for Future Growth without operational bottlenecks.

Question 5:

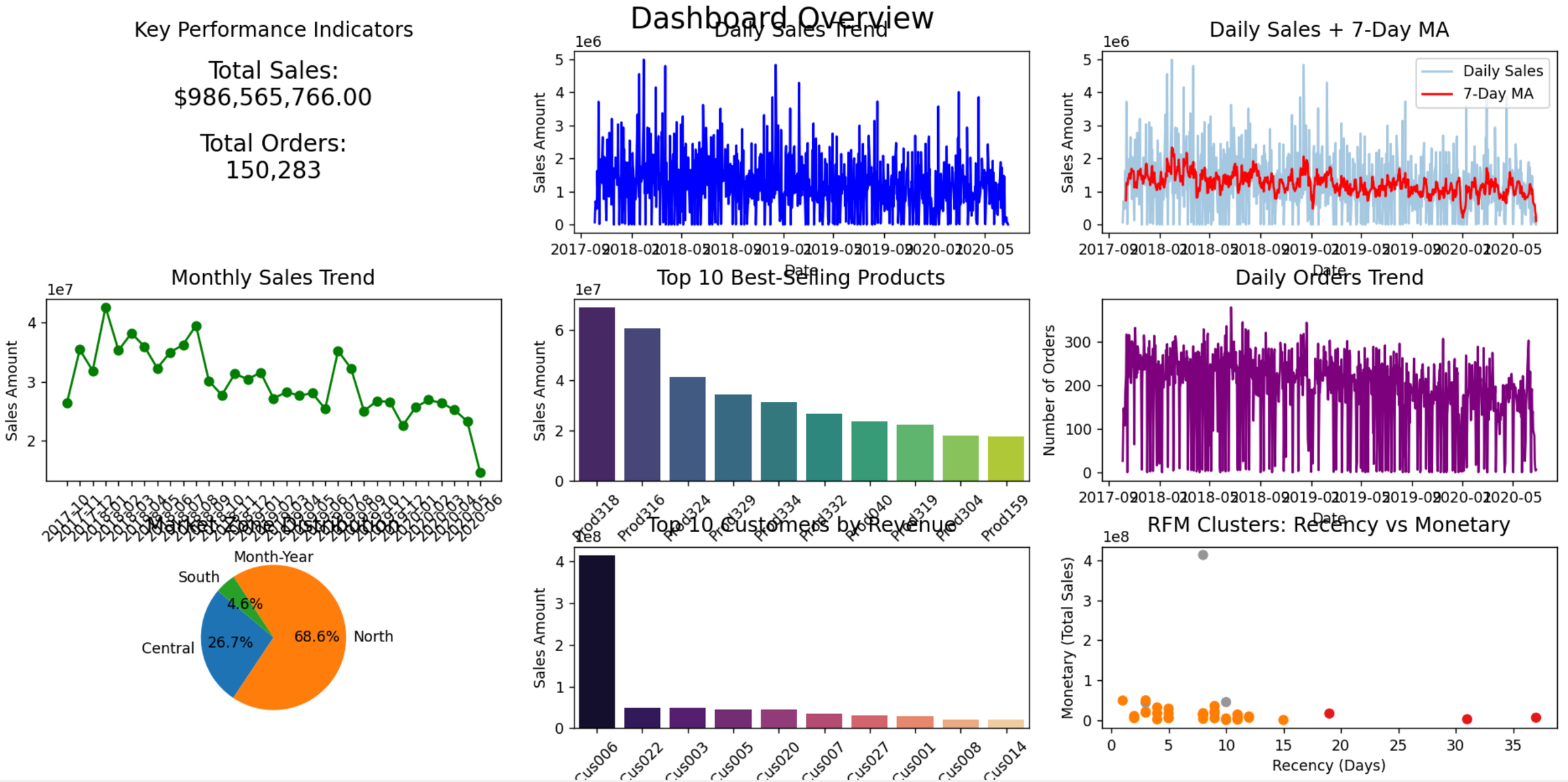
Without centralized, real-time insights, the leadership team struggles to make quick, informed decisions. If you were designing a dashboard for Arjun Mehta, what metrics would you prioritize, and how would you ensure that the insights are actionable?

Solution: Dashboarding

Instead of using multiple Excel files, you can set up a process where Python automatically grabs all the files, combines them into one dataset, and saves that in one place. Then, create a simple dashboard that shows key numbers. This dashboard will update automatically so that managers can quickly see the latest information and make fast decisions.

The columns to be kept in the final dashboard are decided by how useful they are in analyzing the data- those columns which are irrelevant should be removed from the dashboard to keep it simple and minimized. In the present dataset, the useful columns are- total sales, number of orders, best-selling products, and monthly sales trends. Some vestigial columns include market number, customer number, cy-date etc. These do not add any real value to the output, so these are not required there.

Example dashboard:



Thank you