Kwanza Tukule Data Analyst Assessment

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from math import sqrt
from random import random
from pylab import rcParams

# use whitegrid style
sns.set(style="whitegrid")
import plotly.graph_objects as go

# forecasting libraries
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.stattools import adfuller
from patsy import dmatrices
#from pmdarima.arima import auto_arima
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
# import tensorflow as tf
# from tensorflow.keras.models import Sequential
# from tensorflow.keras.layers import LSTM, Dense
# from sklearn.preprocessing import MinMaxScaler
# from sklearn.metrics import mean_squared_error

## anomaly detection libraries
from scipy.stats import zscore
from sklearn.ensemble import IsolationForest
```

In [2]:
```
import warnings

warnings.filterwarnings("ignore")
```

Section 1: Data Cleaning and Preparation (20 points)

- Data Quality Assessment: Inspect the dataset for missing values, duplicates, or inconsistent data types. Provide a summary of issues identified and the steps taken to resolve them.

Reading the Dataset

In [3]:
```python
df = pd.read_csv("Case Study Data.csv")
df.head()
```

Out[3]:

	DATE	ANONYMIZED CATEGORY	ANONYMIZED PRODUCT	ANONYMIZED BUSINESS	ANONYMIZED LOCATION	QUANTIT
0	August 18, 2024, 9:32 PM	Category-106	Product-21f4	Business-de42	Location-1ba8	
1	August 18, 2024, 9:32 PM	Category-120	Product-4156	Business-de42	Location-1ba8	
2	August 18, 2024, 9:32 PM	Category-121	Product-49bd	Business-de42	Location-1ba8	
3	August 18, 2024, 9:32 PM	Category-76	Product-61dd	Business-de42	Location-1ba8	
4	August 18, 2024, 9:32 PM	Category-119	Product-66e0	Business-de42	Location-1ba8	

Dataset Exploration

In [4]:
```python
df.shape
```

Out[4]:
```
(333405, 7)
```

In [5]:
```python
df.dtypes
```

Out[5]:
```
DATE                     object
ANONYMIZED CATEGORY      object
ANONYMIZED PRODUCT       object
ANONYMIZED BUSINESS      object
ANONYMIZED LOCATION      object
QUANTITY                  int64
UNIT PRICE               object
dtype: object
```

Checking missing values

In [6]:
```python
df.isna().sum()
```

Out[6]:
```
DATE                    0
ANONYMIZED CATEGORY     0
ANONYMIZED PRODUCT      0
ANONYMIZED BUSINESS     0
ANONYMIZED LOCATION     0
QUANTITY                0
UNIT PRICE              8
dtype: int64
```

In [7]:
```
df[df['UNIT PRICE'].isna()]
```

Out[7]:

	DATE	ANONYMIZED CATEGORY	ANONYMIZED PRODUCT	ANONYMIZED BUSINESS	ANONYMIZED LOCATION
108112	July 3, 2024, 5:53 PM	Category-94	Product-3d7f	Business-4fce	Location-f37d
150961	December 16, 2024, 6:33 PM	Category-79	Product-dfc8	Business-8bbf	Location-3fc0
151142	December 22, 2024, 2:42 PM	Category-122	Product-15e0	Business-c575	Location-1979
272379	June 27, 2024, 12:15 PM	Category-92	Product-ccbc	Business-14b6	Location-1979
278284	August 14, 2024, 9:09 PM	Category-101	Product-84a5	Business-4be1	Location-bb69
278384	December 30, 2024, 2:17 PM	Category-95	Product-15f3	Business-1a74	Location-f37d
310385	March 31, 2024, 2:03 PM	Category-114	Product-9204	Business-c9dc	Location-689f
327152	August 13, 2024, 4:20 PM	Category-107	Product-7eed	Business-0d61	Location-1ba8

In [8]:
```
df[(df['ANONYMIZED PRODUCT'] == "Product-3d7f") & (df['ANONYMIZED BUSINESS']
```

Out[8]:

	DATE	ANONYMIZED CATEGORY	ANONYMIZED PRODUCT	ANONYMIZED BUSINESS	ANONYMIZED LOCATION	QU
108112	July 3, 2024, 5:53 PM	Category-94	Product-3d7f	Business-4fce	Location-f37d	
136504	July 3, 2024, 6:05 PM	Category-94	Product-3d7f	Business-4fce	Location-f37d	

In [9]:

```
df[(df['ANONYMIZED PRODUCT'] == "Product-dfc8") & (df['ANONYMIZED BUSINESS']
```

Out[9]:

	DATE	ANONYMIZED CATEGORY	ANONYMIZED PRODUCT	ANONYMIZED BUSINESS	ANONYMIZED LOCATION
150961	December 16, 2024, 6:33 PM	Category-79	Product-dfc8	Business-8bbf	Location-3fc0

In [10]:

```
df[(df['ANONYMIZED PRODUCT'] == "Product-9204") & (df['ANONYMIZED BUSINESS']
```

Out[10]:

	DATE	ANONYMIZED CATEGORY	ANONYMIZED PRODUCT	ANONYMIZED BUSINESS	ANONYMIZED LOCATION	QU
310385	March 31, 2024, 2:03 PM	Category-114	Product-9204	Business-c9dc	Location-689f	

- checking on the data duplication

In [11]:

```
df[df.duplicated()]
```

Out[11]:

	DATE	ANONYMIZED CATEGORY	ANONYMIZED PRODUCT	ANONYMIZED BUSINESS	ANONYMIZED LOCATION	C
6153	January 6, 2024, 11:52 AM	Category-91	Product-1b48	Business-20fc	Location-b125	
7554	July 9, 2024, 2:26 PM	Category-104	Product-af50	Business-476c	Location-b27b	
7555	July 9, 2024, 2:26 PM	Category-92	Product-d09a	Business-476c	Location-b27b	
12238	April 19, 2024, 3:19 PM	Category-75	Product-086d	Business-b48e	Location-03fc	
12239	April 19, 2024, 3:19 PM	Category-106	Product-21f4	Business-b48e	Location-03fc	
...	...	...	...	...	...	
333133	February 1, 2024, 9:17 AM	Category-111	Product-7fac	Business-4919	Location-3e32	
333134	February 1, 2024, 9:17 AM	Category-77	Product-d09c	Business-4919	Location-3e32	
333350	June 10, 2024, 10:08 PM	Category-76	Product-e805	Business-54ad	Location-3e32	
333399	January 9, 2024, 8:49 PM	Category-97	Product-bbdc	Business-f9ff	Location-1979	
333400	January 9, 2024, 8:49 PM	Category-119	Product-e98d	Business-f9ff	Location-1979	

3524 rows × 7 columns

In [12]:
```
len(df[df.duplicated()])
```

Out[12]: 3524

In [13]:
```
df.describe()
```

Out[13]:

	QUANTITY
count	333405.000000
mean	2.321186
std	3.790614
min	0.000000
25%	1.000000
50%	1.000000
75%	2.000000
max	359.000000

Data Issues Found

- Columns Names have spaces, this can replace by underscore.
- Date column is in a wrong data dtypes(object) instead of Datetime
- Unit price is also is a wrong data type instead of a numeric datatype(float)
- Unit Price has 8 missing values.
- There are 3524 duplicated data on the dataset.

Steps to resolve the data Issues

- Removing space from column names and replacing with '_'

In [14]:
```
# replace space with '_'
df.columns = df.columns.str.replace(' ', '_')

# check
df.columns
```

Out[14]:
```
Index(['DATE', 'ANONYMIZED_CATEGORY', 'ANONYMIZED_PRODUCT',
       'ANONYMIZED_BUSINESS', 'ANONYMIZED_LOCATION', 'QUANTITY', 'UNIT_PRIC
E'],
      dtype='object')
```

- Correcting the wrong datatypes, Date & Unit Price

In [15]:
```
# convert date to Datetime data type
df['DATE'] = pd.to_datetime(df['DATE'])

# test(check if corrected)
df['DATE'].dtype
```

Out[15]:
```
dtype('<M8[ns]')
```

In [16]:
```
df['UNIT_PRICE'].unique()[:20]
```

```
Out[16]: array(['850', '1,910', '3,670', '2,605', '1,480', '1,940', '1,460', '805',
                '1,350', '1,700', '3,650', '1,800', '4,000', '815', '2,500', '750',
                '2,255', '2,540', '1,880', '2,120'], dtype=object)
```

> from the sample above it seems the, values are integers, but the comma after 3 values, is the problem.

- First remove the comma in the unit price

```
In [17]: df['UNIT_PRICE'] = df['UNIT_PRICE'].str.replace(',', '')

         # check
         df['UNIT_PRICE'][:5]
```

```
Out[17]: 0     850
         1    1910
         2    3670
         3    2605
         4    1480
         Name: UNIT_PRICE, dtype: object
```

```
In [18]: # convert unit_price to numeric
         df['UNIT_PRICE'] = pd.to_numeric(df['UNIT_PRICE'])

         # check
         df['UNIT_PRICE'].dtype
```

```
Out[18]: dtype('float64')
```

- Removing Duplicate Values
  - Remove duplicates values(NB: I think one of the values should be kept)

```
In [135… # dropiing duplicates the parameter keep first ensure one(firt record is ret
         df = df.drop_duplicates(keep='first')

         # check
         len(df[df.duplicated()])
```

```
Out[135… 0
```

> On duplicates Values, The values need to be further check maybe from ERP to confirm on duplication Issues.

- Dealing with missing values.
  - my thoughts here is same product within same place shares common price.

In [20]:
```python
df.isna().sum()
```

Out[20]:
```
DATE                    0
ANONYMIZED_CATEGORY     0
ANONYMIZED_PRODUCT      0
ANONYMIZED_BUSINESS     0
ANONYMIZED_LOCATION     0
QUANTITY                0
UNIT_PRICE              8
dtype: int64
```

In [21]:
```python
# Fill missing UNIT_PRICE with the same price for the same product, locatior
df['UNIT_PRICE'] = df['UNIT_PRICE'].fillna(
    df.groupby(['ANONYMIZED_PRODUCT', 'ANONYMIZED_LOCATION', 'ANONYMIZED_BUS
)
```

In [22]:
```python
df[df['UNIT_PRICE'].isna()]
```

Out[22]:

	DATE	ANONYMIZED_CATEGORY	ANONYMIZED_PRODUCT	ANONYMI
150961	2024-12-16 18:33:00	Category-79	Product-dfc8	
151142	2024-12-22 14:42:00	Category-122	Product-15e0	
272379	2024-06-27 12:15:00	Category-92	Product-ccbc	
278384	2024-12-30 14:17:00	Category-95	Product-15f3	
310385	2024-03-31 14:03:00	Category-114	Product-9204	

In [23]:
```python
df[df['ANONYMIZED_PRODUCT'] == 'Product-15e0']
```

Out[23]:

	DATE	ANONYMIZED_CATEGORY	ANONYMIZED_PRODUCT	ANONYMI
11995	2024-12-29 20:42:00	Category-122	Product-15e0	
105503	2024-12-23 17:28:00	Category-122	Product-15e0	
151142	2024-12-22 14:42:00	Category-122	Product-15e0	
187599	2024-12-23 17:55:00	Category-122	Product-15e0	
199322	2024-12-23 18:00:00	Category-122	Product-15e0	
203067	2024-12-23 17:41:00	Category-122	Product-15e0	
309055	2024-12-27 13:13:00	Category-122	Product-15e0	

> Checking on the missing unit value product, I found the some product having same price across, so this I will directly replace price, but the other product not having same merge I decide to drop on those

In [24]:

```
# Identify products with a consistent price
consistent_price = df.groupby('ANONYMIZED_PRODUCT')['UNIT_PRICE'].transform(

# Fill missing UNIT_PRICE using consistent product prices
df['UNIT_PRICE'] = df['UNIT_PRICE'].fillna(
    df[consistent_price].groupby('ANONYMIZED_PRODUCT')['UNIT_PRICE'].transfo
)

# check missing values:
df.isna().sum()
```

Out[24]:

```
DATE                   0
ANONYMIZED_CATEGORY    0
ANONYMIZED_PRODUCT     0
ANONYMIZED_BUSINESS    0
ANONYMIZED_LOCATION    0
QUANTITY               0
UNIT_PRICE             3
dtype: int64
```

the remaining missing 3 values I decide to drop, since the is no correlation to tell there prices

In [25]:
```
# Drop rows where UNIT_PRICE is still missing
df = df.dropna(subset=['UNIT_PRICE'])
```

In [26]:
```
df.shape
```

Out[26]:
```
(329878, 7)
```

Feature Engineering:

- Create the following columns: “Month-Year” (e.g., August 2024) from the “DATE” column. (include a screenshot of this in your submission)

In [27]:
```
df['DATE'].dt.to_period('M')
```

Out[27]:
```
0         2024-08
1         2024-08
2         2024-08
3         2024-08
4         2024-08
            ...
333398    2024-11
333401    2024-08
333402    2024-08
333403    2024-10
333404    2024-10
Name: DATE, Length: 329878, dtype: period[M]
```

In [28]:
```
df['Month-Year'] = df['DATE'].dt.month_name() + '-' + df['DATE'].dt.year.ast
df['Month-Year'][:5]
```

Out[28]:
```
0    August-2024
1    August-2024
2    August-2024
3    August-2024
4    August-2024
Name: Month-Year, dtype: object
```

In [29]:
```
df.columns
```

Out[29]:
```
Index(['DATE', 'ANONYMIZED_CATEGORY', 'ANONYMIZED_PRODUCT',
       'ANONYMIZED_BUSINESS', 'ANONYMIZED_LOCATION', 'QUANTITY', 'UNIT_PRIC
E',
       'Month-Year'],
      dtype='object')
```

- Save Cleaned Data maybe reuse case comes in later.

In [30]:
```
df.to_csv("clean_case_study_data.csv", index=False)
```

Section 2: Exploratory Data Analysis (30 points)

- Sales Overview:
  - Calculate total Quantity and Value grouped by:
    - Anonymized Category
    - Anonymized Business
  - Provide visualizations (e.g., bar charts or tables) to support your findings.

- Getting the Sales value, then groupby and sum.

In [34]:
```
# Calculate sales value
df['SALES_VALUE'] = df['QUANTITY'] * df['UNIT_PRICE']
```

- Anonymized Category

In [35]:
```
df['ANONYMIZED_CATEGORY'].nunique()
```

Out[35]: 46

There are 46 unique category in the dataset

In [125…
```
#df.groupby('ANONYMIZED_CATEGORY')['QUANTITY'].sum().reset_index().sort_valu
```

In [36]:
```
# Group by 'ANONYMIZED_CATEGORY' and calculate total Quantity and Sales Valu
category_summary = df.groupby('ANONYMIZED_CATEGORY').agg(
    TOTAL_QUANTITY=('QUANTITY', 'sum'),
    TOTAL_SALES_VALUE=('SALES_VALUE', 'sum')
).sort_values('TOTAL_SALES_VALUE', ascending=False).reset_index()
category_summary.head(10)
```

Out[36]:

	ANONYMIZED_CATEGORY	TOTAL_QUANTITY	TOTAL_SALES_VALUE
0	Category-75	151330	544658700.0
1	Category-76	71719	344939553.0
2	Category-120	169715	319178743.0
3	Category-100	76824	134902751.0
4	Category-119	68332	103454819.0
5	Category-77	28455	76741382.0
6	Category-91	20853	44152103.0
7	Category-101	19585	35614152.0
8	Category-85	22997	33762533.0
9	Category-121	14669	22327643.0

In [31]:
```python
import plotly.io as pio
#pio.renderers.default = 'browser'  # Tto open charts on browser.
```

In [37]:
```python
import plotly.graph_objects as go
from plotly.subplots import make_subplots

def plot_separate_side_by_side(df, x_column, quantity_column, sales_column,
    # Create a subplot with 1 row and 2 columns
    fig = make_subplots(
        rows=1, cols=2,
        subplot_titles=[title_quantity, title_sales],
        column_widths=[0.5, 0.5]  # Even distribution for both plots
    )

    # Add Total Quantity plot (first subplot)
    fig.add_trace(go.Bar(
        x=df[x_column],
        y=df[quantity_column],
        name='Total Quantity',
        marker_color='skyblue'
    ), row=1, col=1)

    # Add Total Sales Value plot (second subplot)
    fig.add_trace(go.Bar(
        x=df[x_column],
        y=df[sales_column],
        name='Total Sales Value',
        marker_color='teal'
    ), row=1, col=2)

    # Update layout to improve appearance and add titles
    fig.update_layout(
        title='Comparison of Total Quantity and Sales Value',
        showlegend=False,
        xaxis=dict(title=xlabel, tickangle=-45),
        xaxis2=dict(title=xlabel, tickangle=-45),
        template='plotly_white',
        height=600  # Increase height for better visibility
    )

    # Show the figure
    fig.show()

# Side-by-Side Plots for Anonymized Category
plot_separate_side_by_side(
    category_summary.head(20),
    x_column='ANONYMIZED_CATEGORY',
    quantity_column='TOTAL_QUANTITY',
    sales_column='TOTAL_SALES_VALUE',
    title_quantity='Total Quantity by Anonymized Category',
    title_sales='Total Sales Value by Anonymized Category',
    xlabel='Anonymized Category'
)
```

This graphs plotted on the browsers I'll share a screenshot on the report

- ANONYMIZED_BUSINESS

In [38]:
```python
# Group by 'ANONYMIZED_BUSINESS' and calculate total Quantity and Value
business_summary = df.groupby('ANONYMIZED_BUSINESS').agg(
    TOTAL_QUANTITY=('QUANTITY', 'sum'),
    TOTAL_SALES_VALUE=('SALES_VALUE', 'sum')
).sort_values('TOTAL_SALES_VALUE', ascending=False).reset_index()
business_summary.head(20)
```

Out[38]:

	ANONYMIZED_BUSINESS	TOTAL_QUANTITY	TOTAL_SALES_VALUE
0	Business-978e	13991	28037358.0
1	Business-fe7d	6743	26997121.0
2	Business-6068	8214	16464195.0
3	Business-07de	6065	16258068.0
4	Business-7a03	6318	13968451.0
5	Business-ba13	5533	13650016.0
6	Business-1e3e	4981	13192967.0
7	Business-468e	5450	12546597.0
8	Business-f4f4	3852	11952941.0
9	Business-5613	4089	11895552.0
10	Business-8119	3788	11727274.0
11	Business-e672	3242	10758445.0
12	Business-cb1f	4636	9602700.0
13	Business-a8bd	3494	9344121.0
14	Business-d72e	3835	9280411.0
15	Business-80b3	4303	9275497.0
16	Business-2197	3327	9102473.0
17	Business-0e5b	4289	9062164.0
18	Business-2efb	3625	8498555.0
19	Business-3215	3495	8489805.0

In [39]:
```python
# Side-by-Side Plots for Anonymized Category
plot_separate_side_by_side(
    business_summary.head(20),
    x_column='ANONYMIZED_BUSINESS',
    quantity_column='TOTAL_QUANTITY',
    sales_column='TOTAL_SALES_VALUE',
```

```
    title_quantity='Total Quantity by Anonymized Business',
    title_sales='Total Sales Value by Anonymized Business',
    xlabel='Anonymized Business'
)
```

2. Trends Over Time:

- Analyze sales trends (Value and Quantity) by Month-Year. Create a time series plot to show seasonal patterns or changes in sales performance.

- **Note** for plotting a trend line on any foreacting data has to sorted using date

In [40]:
```
df['DATE'].min(), df['DATE'].max()
```

Out[40]: (Timestamp('2024-01-01 05:54:00'), Timestamp('2024-12-31 18:24:00'))

In [41]:
```
## check if the data is sorted according to date
df = df.sort_values('DATE').reset_index(drop=True)
df.head()
```

Out[41]:

	DATE	ANONYMIZED_CATEGORY	ANONYMIZED_PRODUCT	ANONYMIZED_B
0	2024-01-01 05:54:00	Category-75	Product-086d	Busi
1	2024-01-01 05:54:00	Category-85	Product-0c64	Busi
2	2024-01-01 07:18:00	Category-120	Product-4156	Busi
3	2024-01-01 07:18:00	Category-75	Product-2175	Busi
4	2024-01-01 10:45:00	Category-85	Product-6859	Busi

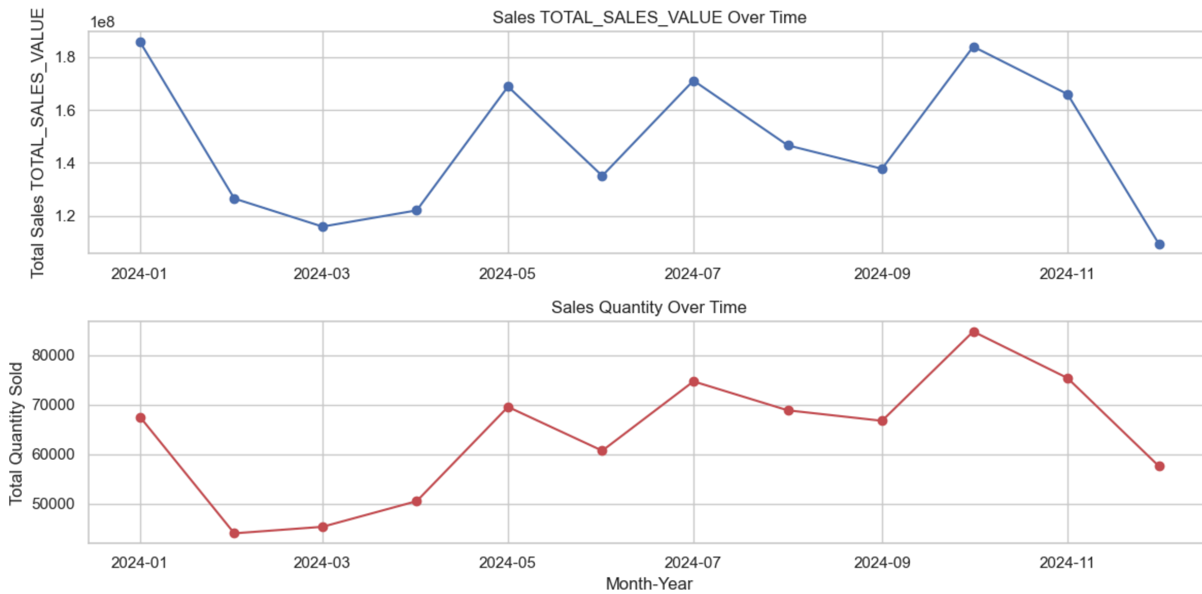
In [42]:
```
# Calculate the total TOTAL_SALES_VALUE (QUANTITY * UNIT_PRICE)
df['TOTAL_SALES_VALUE'] = df['QUANTITY'] * df['UNIT_PRICE']

# Aggregate by Month-Year
df['MONTH_YEAR'] = df['DATE'].dt.to_period('M')  # Create a Month-Year colum
monthly_sales = df.groupby('MONTH_YEAR').agg({'TOTAL_SALES_VALUE': 'sum', 'Q

# Convert MONTH_YEAR back to datetime for plotting
monthly_sales['MONTH_YEAR'] = monthly_sales['MONTH_YEAR'].dt.to_timestamp()
monthly_sales
```

Out[42]:

	MONTH_YEAR	TOTAL_SALES_VALUE	QUANTITY
0	2024-01-01	185626186.0	67526
1	2024-02-01	126579702.0	44063
2	2024-03-01	116000676.0	45381
3	2024-04-01	122110750.0	50554
4	2024-05-01	168781502.0	69551
5	2024-06-01	135140164.0	60717
6	2024-07-01	171042631.0	74691
7	2024-08-01	146618918.0	68859
8	2024-09-01	137791455.0	66747
9	2024-10-01	183840551.0	84739
10	2024-11-01	165933104.0	75361
11	2024-12-01	109557214.0	57629

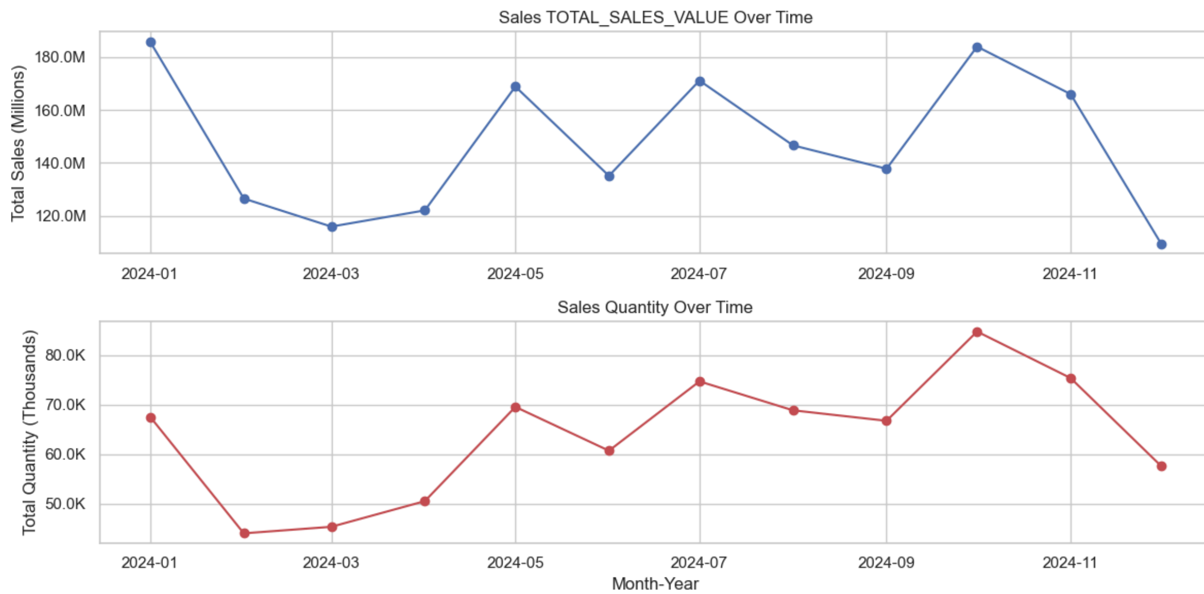
In [134…

```
# Plotting the time series
plt.figure(figsize=(12, 6))

# Plot for TOTAL_SALES_VALUE
plt.subplot(2, 1, 1)
plt.plot(monthly_sales['MONTH_YEAR'], monthly_sales['TOTAL_SALES_VALUE'], ma
plt.title('Sales TOTAL_SALES_VALUE Over Time')
plt.ylabel('Total Sales TOTAL_SALES_VALUE')
plt.grid(True)

# Plot for Quantity
plt.subplot(2, 1, 2)
plt.plot(monthly_sales['MONTH_YEAR'], monthly_sales['QUANTITY'], marker='o',
plt.title('Sales Quantity Over Time')
plt.ylabel('Total Quantity Sold')
plt.xlabel('Month-Year')
plt.grid(True)

plt.tight_layout()
plt.show()
```



In [43]:

```
# Convert sales to millions and quantity to thousands
monthly_sales['TOTAL_SALES_VALUE'] /= 1_000_000  # Convert to Millions
monthly_sales['QUANTITY'] /= 1_000  # Convert to Thousands

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

# Plot for TOTAL_SALES_VALUE (in Millions)
plt.subplot(2, 1, 1)
plt.plot(monthly_sales['MONTH_YEAR'], monthly_sales['TOTAL_SALES_VALUE'], ma
plt.title('Sales TOTAL_SALES_VALUE Over Time')
plt.ylabel('Total Sales (Millions)')
plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _: f'{x:.1
plt.grid(True)

# Plot for Quantity (in Thousands)
plt.subplot(2, 1, 2)
plt.plot(monthly_sales['MONTH_YEAR'], monthly_sales['QUANTITY'], marker='o',
plt.title('Sales Quantity Over Time')
plt.ylabel('Total Quantity (Thousands)')
plt.xlabel('Month-Year')
plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _: f'{x:.1
plt.grid(True)

plt.tight_layout()
plt.show()
```



3. Performance Analysis:

- Identify the top 5 most frequently purchased products (based on Quantity).
- Identify the top 5 most valuable products (based on Value).

In [44]:
```
# group product, thn to get total sales value and quantity of each product
product_summary = df.groupby('ANONYMIZED_PRODUCT').agg(
    Total_Quantity=('QUANTITY', 'sum'),
    Total_Sales_Value=('TOTAL_SALES_VALUE', 'sum')
).reset_index()

# pick the top 5 product by total Quantity
top_5_frequent_products = product_summary.nlargest(5, 'Total_Quantity').rese
top_5_frequent_products
```

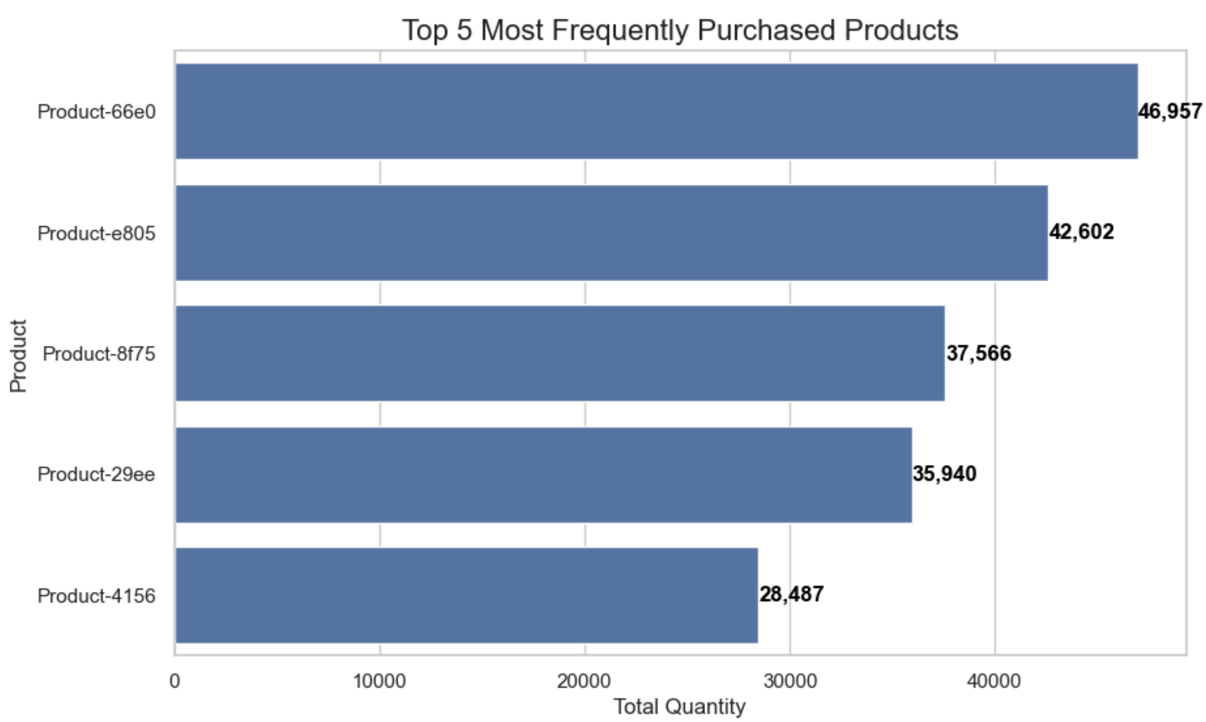
Out[44]:

	ANONYMIZED_PRODUCT	Total_Quantity	Total_Sales_Value
0	Product-66e0	46957	70704225.0
1	Product-e805	42602	262787281.0
2	Product-8f75	37566	158797460.0
3	Product-29ee	35940	68248274.0
4	Product-4156	28487	56956007.0

In [137…
```
# # Plot top 5 most frequently purchased products
# plt.figure(figsize=(10, 6))
# sns.barplot(
#     x='Total_Quantity',
#     y='ANONYMIZED_PRODUCT',
#     data=top_5_frequent_products.sort_values(by='Total_Quantity', ascendir
#     # palette='viridis'
# )
# plt.title('Top 5 Most Frequently Purchased Products', fontsize=16)
# plt.xlabel('Total Quantity', fontsize=12)
```

```
# plt.ylabel('Product', fontsize=12)
# plt.show()
```

In [45]:
```
plt.figure(figsize=(10, 6))
ax = sns.barplot(
    x='Total_Quantity',
    y='ANONYMIZED_PRODUCT',
    data=top_5_frequent_products.sort_values(by='Total_Quantity', ascending=
)

plt.title('Top 5 Most Frequently Purchased Products', fontsize=16)
plt.xlabel('Total Quantity', fontsize=12)
plt.ylabel('Product', fontsize=12)

# Annotate values on top of each bar
for bar in ax.patches:
    plt.text(
        bar.get_width() + 0.5,  # Adjust placement slightly to the right
        bar.get_y() + bar.get_height() / 2,  # Center the text vertically
        f'{int(bar.get_width()):,}',  # Format number with thousands separat
        ha='left',  # Align left for clarity
        va='center',
        fontsize=12,
        fontweight='bold',
        color='black'
    )

plt.show()
```



In [191…
```
# Identify top 5 most valuable products(sales value)
top_5_valuable_products = product_summary.nlargest(5, 'Total_Sales_Value').r
top_5_valuable_products
```

Out[191…

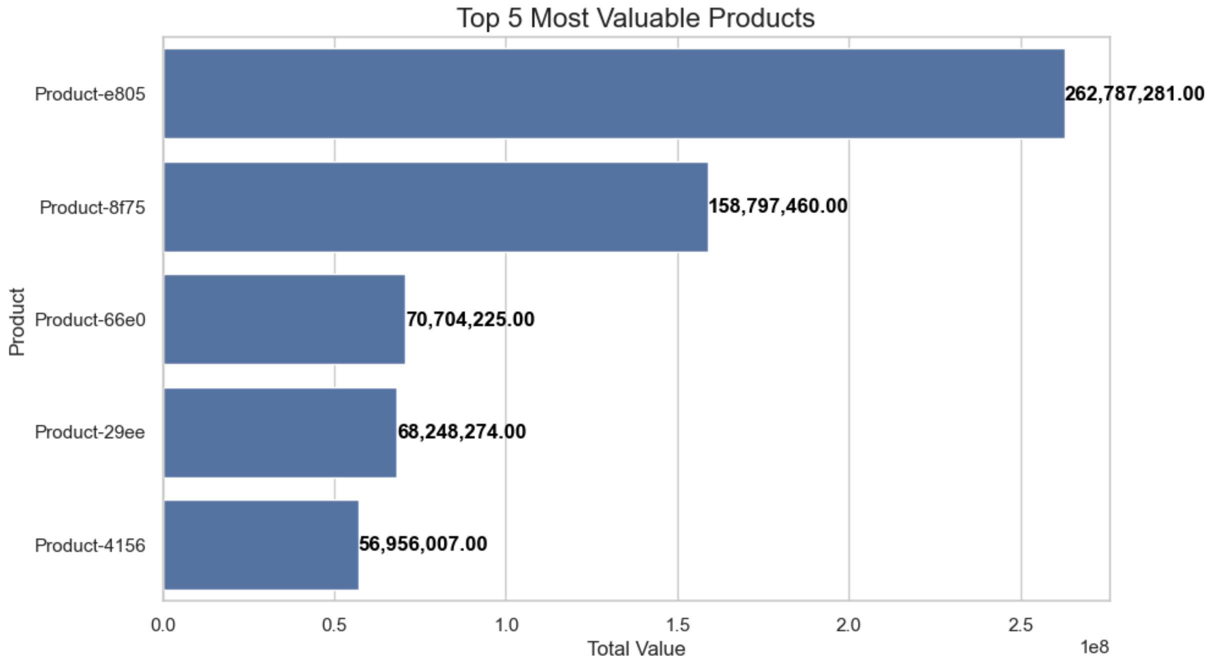
	ANONYMIZED_PRODUCT	Total_Quantity	Total_Sales_Value
0	Product-e805	42602	262787281.0
1	Product-8f75	37566	158797460.0
2	Product-66e0	46957	70704225.0
3	Product-29ee	35940	68248274.0
4	Product-4156	28487	56956007.0

In [140…

```
# # Plot top 5 most valuable products
# plt.figure(figsize=(10, 6))
# sns.barplot(
#     x='Total_Sales_Value',
#     y='ANONYMIZED_PRODUCT',
#     data=top_5_valuable_products.sort_values(by='Total_Sales_Value', ascen
#     # palette='coolwarm'
# )
# plt.title('Top 5 Most Valuable Products', fontsize=16)
# plt.xlabel('Total Value', fontsize=12)
# plt.ylabel('Product', fontsize=12)
# plt.show()
```

In [192…

```
plt.figure(figsize=(10, 6))
ax = sns.barplot(
    x='Total_Sales_Value',
    y='ANONYMIZED_PRODUCT',
    data=top_5_valuable_products.sort_values(by='Total_Sales_Value', ascendi
    #palette='coolwarm'
)

plt.title('Top 5 Most Valuable Products', fontsize=16)
plt.xlabel('Total Value', fontsize=12)
plt.ylabel('Product', fontsize=12)

# Annotate values on top of each bar
for bar in ax.patches:
    plt.text(
        bar.get_width() + 0.5,  # Adjust placement slightly to the right
        bar.get_y() + bar.get_height() / 2,  # Center text vertically
        f'{bar.get_width():,.2f}',  # Format as currency (e.g., 10,000.00)
        ha='left',  # Align left for better visibility
        va='center',
        fontsize=12,
        fontweight='bold',
        color='black'
    )

plt.show()
```



Classify businesses into 3 groups (e.g., High Value, Medium Value, Low Value) and provide recommendations for engagement with each group.

In [51]:
```
# Aggregate metrics for each business # feature engineering for clustering
business_summary = df.groupby('ANONYMIZED_BUSINESS').agg(
    Total_Quantity=('QUANTITY', 'sum'),
    Total_Value=('TOTAL_SALES_VALUE', 'sum'),
    Frequency=('DATE', 'count') # frequency on daily
).reset_index()
```

In [52]:
```
# Standardizing the metrics
scaler = StandardScaler()
scaled_data = scaler.fit_transform(business_summary[['Total_Quantity', 'Tota
```

In [53]:
```
# Apply K-Means clustering, tool to the standard data, k= 3 groups
kmeans = KMeans(n_clusters=3, random_state=42)
business_summary['Segment'] = kmeans.fit_predict(scaled_data)
business_summary
```

Out[53]:

	ANONYMIZED_BUSINESS	Total_Quantity	Total_Value	Frequency	Segme
0	Business-0000	8	10445.0	8	
1	Business-0005	1	2645.0	1	
2	Business-0029	26	77340.0	6	
3	Business-003d	98	221761.0	31	
4	Business-0072	127	225056.0	101	
...	...	...	...	...	
4795	Business-ffa9	3	6740.0	3	
4796	Business-ffae	6	10530.0	5	
4797	Business-ffb1	266	438115.0	105	
4798	Business-ffd2	37	67723.0	22	
4799	Business-ffff	110	110285.0	107	

4800 rows × 5 columns

In [54]:

```
# Clusterinng results
print("Segmented Business Summary:")
business_summary['Segment'].value_counts()
```

```
Segmented Business Summary:
```

Out[54]:

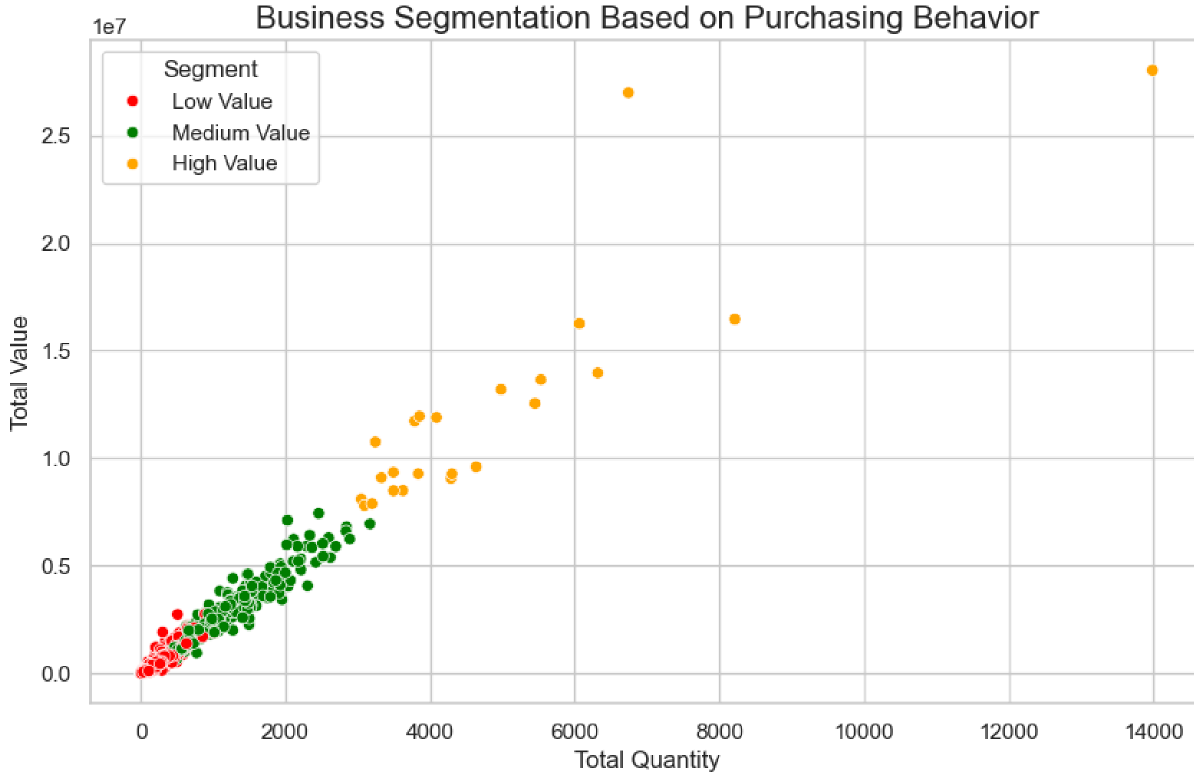
```
Segment
0    4472
2     305
1      23
Name: count, dtype: int64
```

In [55]:

```
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x='Total_Quantity',
    y='Total_Value',
    hue='Segment',
    data=business_summary,
    palette={0: 'red', 1: 'orange', 2: 'green'}  # Use numeric keys
)
plt.title('Business Segmentation Based on Purchasing Behavior', fontsize=16)
plt.xlabel('Total Quantity', fontsize=12)
plt.ylabel('Total Value', fontsize=12)
plt.legend(title='Segment')
plt.show()
```

In [56]:

```
# Map numeric segment values to categorical labels
segment_mapping = {0: 'Low Value', 2: 'Medium Value', 1: 'High Value'}
business_summary['Segment'] = business_summary['Segment'].map(segment_mappir

# plotting clustering results output
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x='Total_Quantity',
    y='Total_Value',
    hue='Segment',
    data=business_summary,
    palette={'Low Value': 'red', 'Medium Value': 'green', 'High Value': 'ora
)
plt.title('Business Segmentation Based on Purchasing Behavior', fontsize=16)
plt.xlabel('Total Quantity', fontsize=12)
plt.ylabel('Total Value', fontsize=12)
plt.legend(title='Segment')
plt.show()
```



In [57]:
```
df.columns
```

Out[57]:
```
Index(['DATE', 'ANONYMIZED_CATEGORY', 'ANONYMIZED_PRODUCT',
       'ANONYMIZED_BUSINESS', 'ANONYMIZED_LOCATION', 'QUANTITY', 'UNIT_PRIC
E',
       'Month-Year', 'SALES_VALUE', 'TOTAL_SALES_VALUE', 'MONTH_YEAR'],
      dtype='object')
```

Forecasting:

- Using the provided data, forecast the total sales (Value) for the next 3 months. Use an appropriate time-series forecasting method (e.g., ARIMA, moving average, or exponential smoothing).

In [58]:
```
monthly_sales.columns
```

Out[58]:
```
Index(['MONTH_YEAR', 'TOTAL_SALES_VALUE', 'QUANTITY'], dtype='object')
```

In [59]:
```
monthly_sales.head()
```

Out[59]:

	MONTH_YEAR	TOTAL_SALES_VALUE	QUANTITY
0	2024-01-01	185.626186	67.526
1	2024-02-01	126.579702	44.063
2	2024-03-01	116.000676	45.381
3	2024-04-01	122.110750	50.554
4	2024-05-01	168.781502	69.551

- select 'TOTAL_SALES_VALUE' only

In [60]:

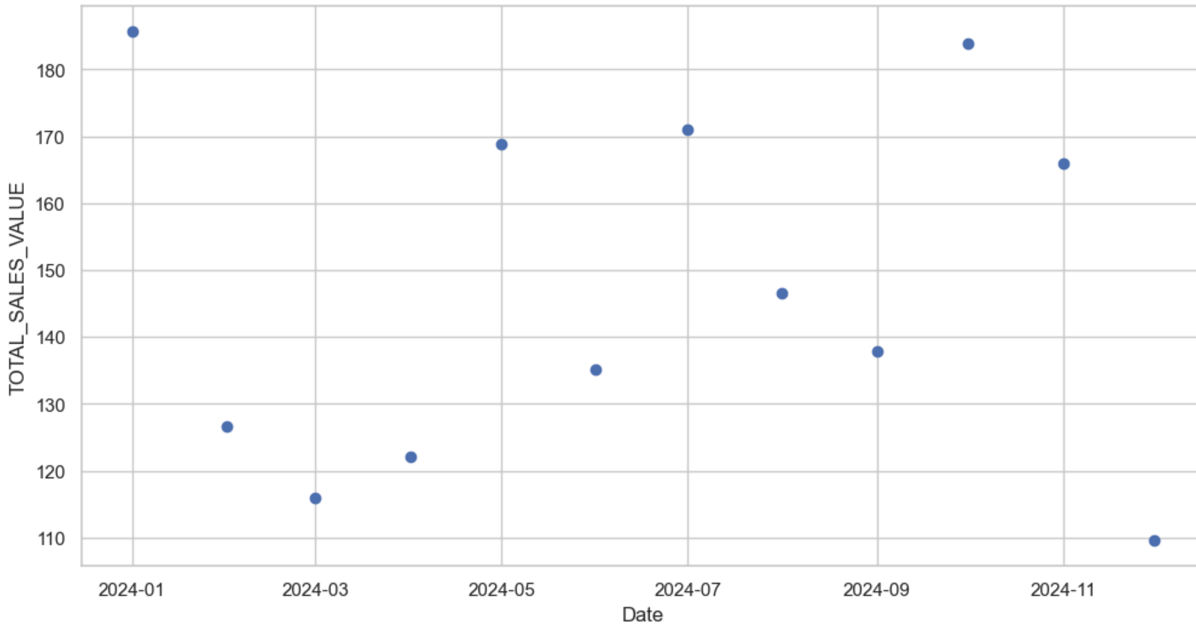
```
monthly_sales_df = monthly_sales[['MONTH_YEAR', 'TOTAL_SALES_VALUE']]
monthly_sales_df = monthly_sales_df.set_index('MONTH_YEAR')
monthly_sales_df.head()
```

Out[60]:

MONTH_YEAR	TOTAL_SALES_VALUE
2024-01-01	185.626186
2024-02-01	126.579702
2024-03-01	116.000676
2024-04-01	122.110750
2024-05-01	168.781502

In [61]:

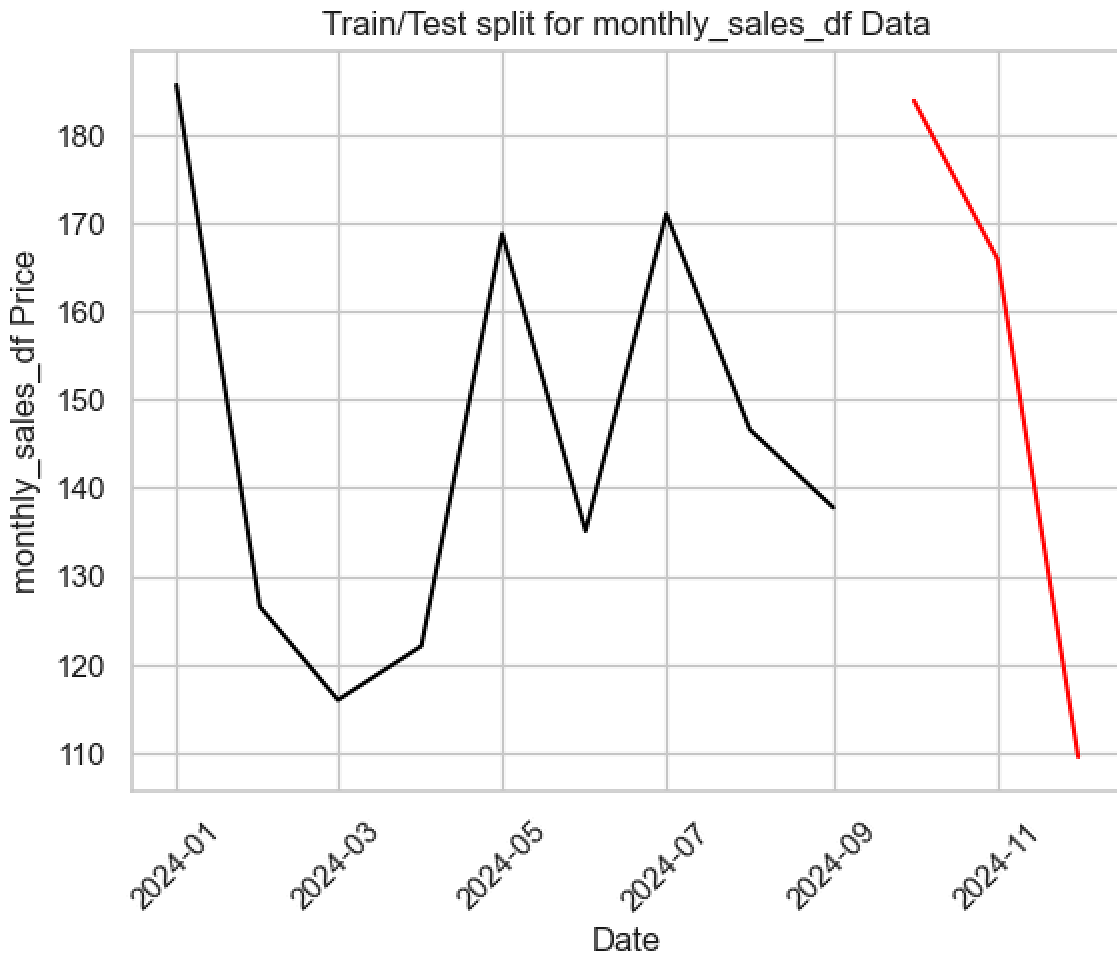
```
fig, ax = plt.subplots(figsize = (12,6))
ax.scatter(monthly_sales['MONTH_YEAR'], monthly_sales["TOTAL_SALES_VALUE"])
ax.set_xlabel('Date')
ax.set_ylabel('TOTAL_SALES_VALUE')
plt.show()
```



In [62]:
```
monthly_sales.to_csv('monthly_sales.csv', index=False)
```

- Split the data to train and test

In [63]:
```
test_start = "2024-9-01"
test_stop = "2024-12-01"

# Ensure the DataFrame index is a datetime index
monthly_sales_df.index = pd.to_datetime(monthly_sales_df.index)

# Define the training and testing datasets
train = monthly_sales_df[monthly_sales_df.index <= pd.to_datetime(test_start
test = monthly_sales_df[(monthly_sales_df.index > pd.to_datetime(test_start,

# Plotting the train and test data
plt.plot(train, color="black")
plt.plot(test, color="red")
plt.ylabel('monthly_sales_df Price')
plt.xlabel('Date')
plt.xticks(rotation=45)
plt.title("Train/Test split for monthly_sales_df Data")
plt.show()
```



In [64]:
```
train.tail()
```

Out[64]:

	TOTAL_SALES_VALUE
MONTH_YEAR	
2024-05-01	168.781502
2024-06-01	135.140164
2024-07-01	171.042631
2024-08-01	146.618918
2024-09-01	137.791455

In [65]:
```
test.head()
```

Out[65]:

	TOTAL_SALES_VALUE
MONTH_YEAR	
2024-10-01	183.840551
2024-11-01	165.933104
2024-12-01	109.557214

In [66]:
```
order = (1, 0, 1)  # (p, d, q) - you may need to adjust these values based c

model = ARIMA(train['TOTAL_SALES_VALUE'], order = order)
```

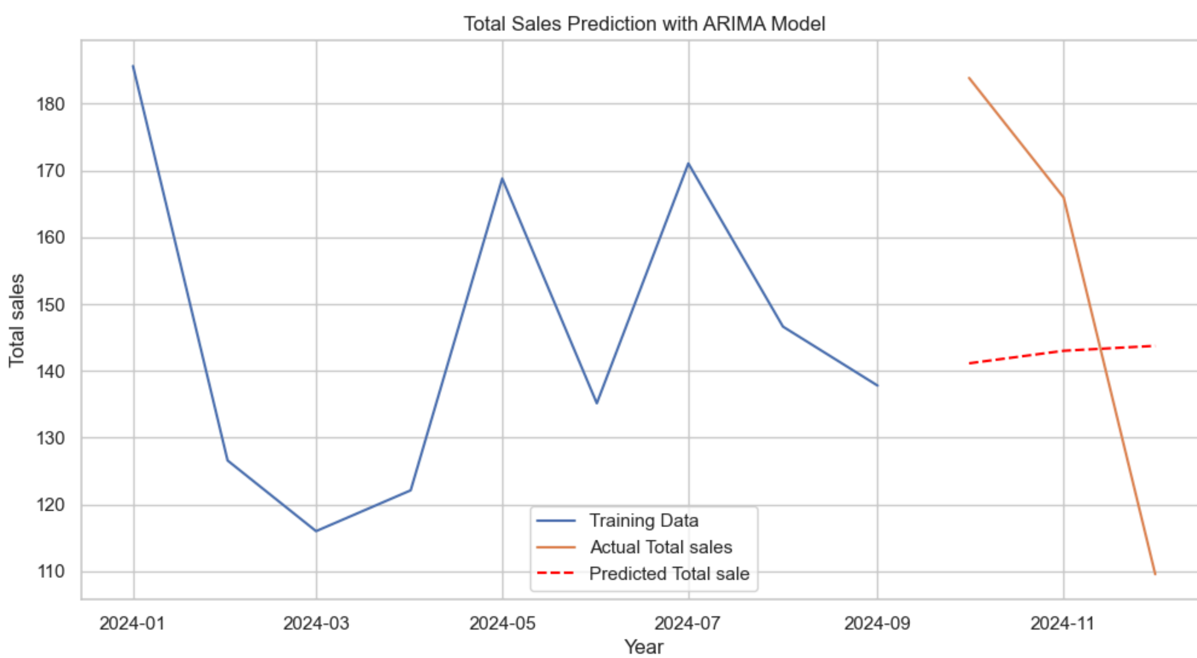
In [67]:
```
arima_model = model.fit()
```

In [68]:
```
predictions = arima_model.get_forecast(steps = len(test))
```

In [69]:
```
predicted_values = predictions.predicted_mean
predicted_values
```

Out[69]:
```
2024-10-01    141.124809
2024-11-01    143.001362
2024-12-01    143.741772
Freq: MS, Name: predicted_mean, dtype: float64
```

In [70]:
```
plt.figure(figsize = (12, 6))
plt.plot(train['TOTAL_SALES_VALUE'], label = 'Training Data')
plt.plot(test['TOTAL_SALES_VALUE'], label = f'Actual Total sales')
plt.plot(test.index, predicted_values, label = f'Predicted Total sale', line
plt.title(f'Total Sales Prediction with ARIMA Model')
plt.xlabel('Year')
plt.ylabel('Total sales')
plt.legend()
plt.show()
```



In [71]:
```
def calculate_mape(actual, predicted):
    return np.mean(np.abs((actual - predicted) / actual)) * 100

mape = calculate_mape(test['TOTAL_SALES_VALUE'], predicted_values)
print(f'Mean Absolute Percentage Error (MAPE): {mape:.2f}%')
```

```
Mean Absolute Percentage Error (MAPE): 22.75%
```

- sarimax model

In [72]:
```
order = (1, 0, 1)  # (p, d, q) - you may need to adjust these values based o
seasonal_order = (1, 1, 1, 3)  # (P, D, Q, S) - seasonal components, since m
```

In [73]:
```
model_s = SARIMAX(monthly_sales_df['TOTAL_SALES_VALUE'], order = order, seas
```

In [74]:
```
sarima_model = model_s.fit(disp = False)
```

In [75]:
```
start = len(train)
end = len(train) + len(test) - 1
predictions = sarima_model.get_prediction(start = start, end = end, dynamic
```

In [76]:
```
start, end
```

Out[76]:
```
(9, 11)
```

In [77]:
```
predicted_values = predictions.predicted_mean
predicted_values
```

Out[77]:
```
2024-10-01    150.042095
2024-11-01    157.220114
2024-12-01    130.605756
Freq: MS, Name: predicted_mean, dtype: float64
```

In [78]:
```
monthly_sales_df.tail(3)
```

Out[78]:

MONTH_YEAR	TOTAL_SALES_VALUE
2024-10-01	183.840551
2024-11-01	165.933104
2024-12-01	109.557214

In [79]:
```
mse = ((predicted_values - test['TOTAL_SALES_VALUE']) ** 2).mean()
rmse = np.sqrt(mse)

print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
```

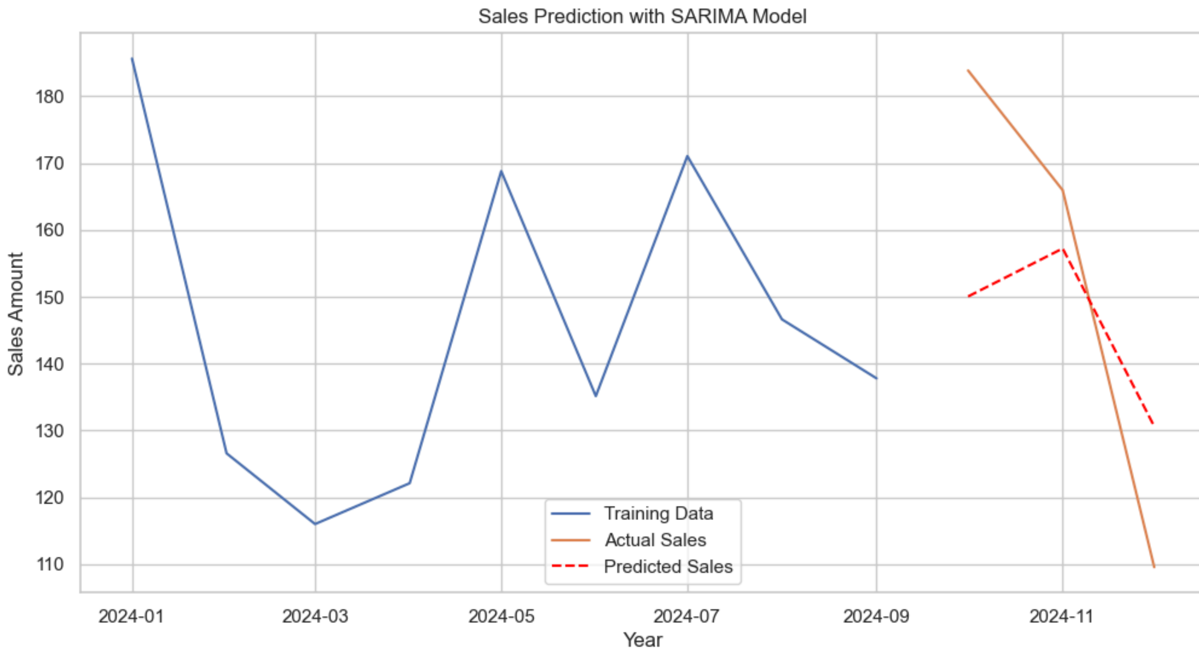
```
Mean Squared Error (MSE): 553.7643192324339
Root Mean Squared Error (RMSE): 23.5321975011352
```

In [80]:
```
mape = calculate_mape(test['TOTAL_SALES_VALUE'], predicted_values)
print(f'Mean Absolute Percentage Error (MAPE): {mape:.2f}%')
```

```
Mean Absolute Percentage Error (MAPE): 14.28%
```

In [81]:
```
plt.figure(figsize = (12, 6))
plt.plot(train['TOTAL_SALES_VALUE'], label = 'Training Data')
plt.plot(test['TOTAL_SALES_VALUE'], label = f'Actual Sales')
```

```
plt.plot(predicted_values, label = f'Predicted Sales', linestyle ='--', colo
plt.title(f'Sales Prediction with SARIMA Model')
plt.xlabel('Year')
plt.ylabel(f'Sales Amount')
plt.legend()
plt.show()
```



- On sarima model try a seasonal of 1 year(like 1 fiscal year)

In [82]:
```
order = (1, 0, 1)  # (p, d, q) - you may need to adjust these values based o
seasonal_order = (1, 1, 1, 12)
```

In [83]:
```
start = len(monthly_sales_df)
end = len(monthly_sales_df) + 2
predictions = sarima_model.get_prediction(start = start, end = end, dynamic

start, end
```

Out[83]:
```
(12, 14)
```

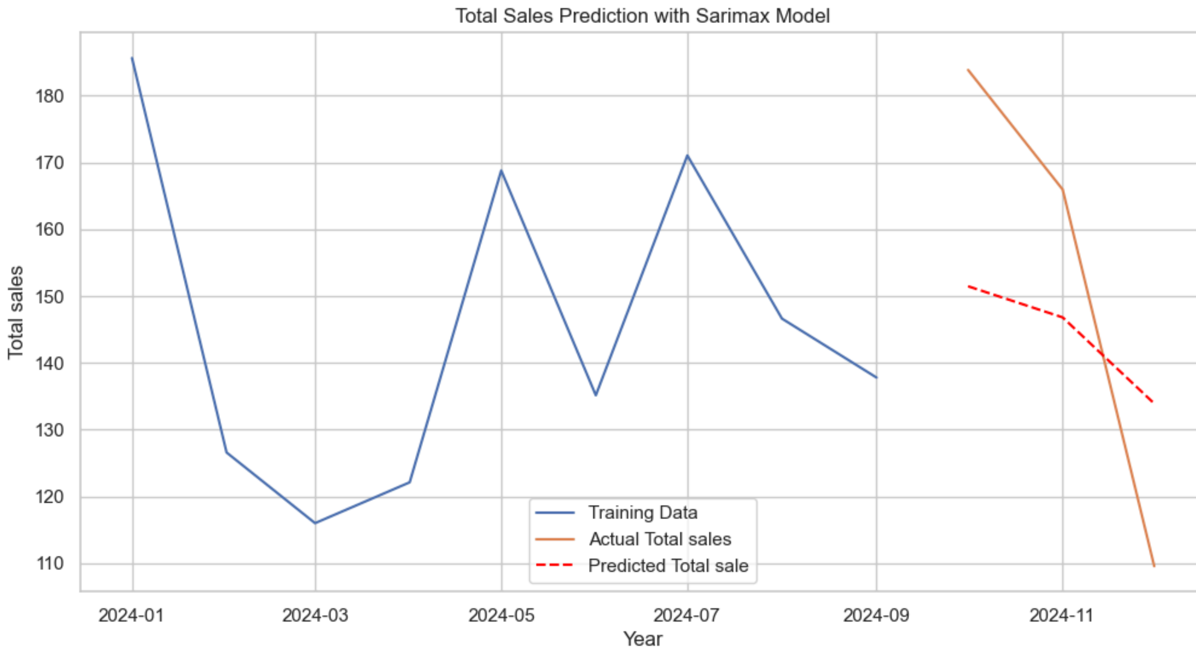
In [84]:
```
predicted_values = predictions.predicted_mean
predicted_values
```

Out[84]:
```
2025-01-01    151.458628
2025-02-01    146.797282
2025-03-01    133.905230
Freq: MS, Name: predicted_mean, dtype: float64
```

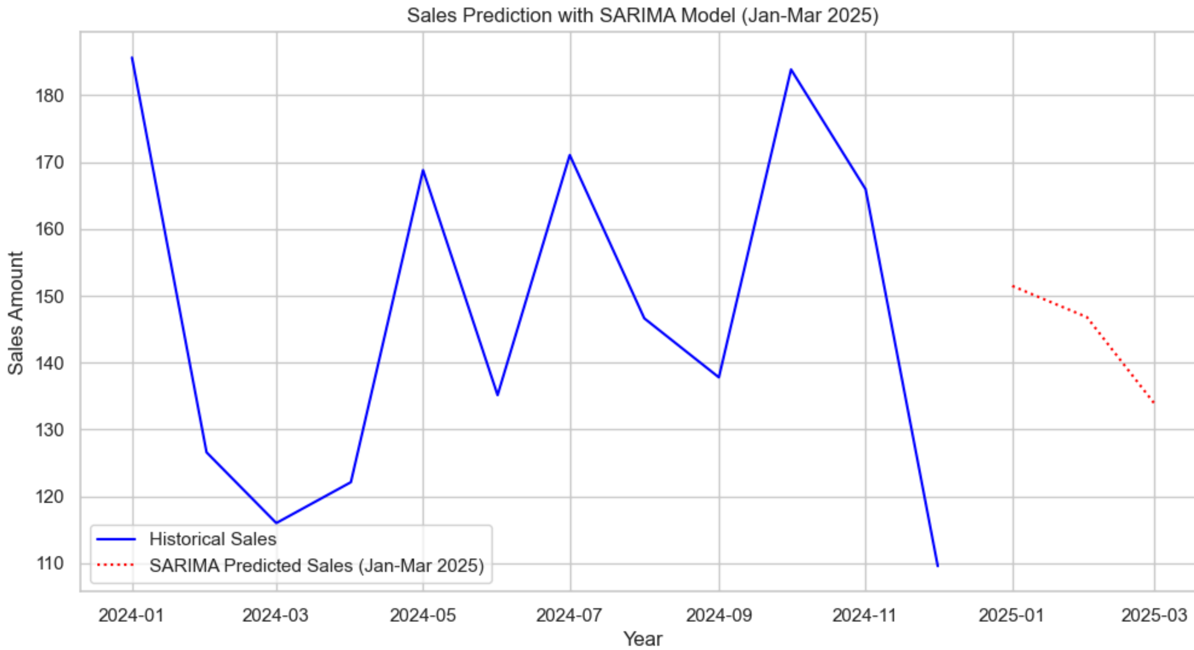
In [ ]:

In [85]:
```
plt.figure(figsize = (12, 6))
plt.plot(train['TOTAL_SALES_VALUE'], label = 'Training Data')
plt.plot(test['TOTAL_SALES_VALUE'], label = f'Actual Total sales')
plt.plot(test.index, predicted_values, label = f'Predicted Total sale', line
```

```
plt.title(f'Total Sales Prediction with Sarimax Model')
plt.xlabel('Year')
plt.ylabel('Total sales')
plt.legend()
plt.show()
```

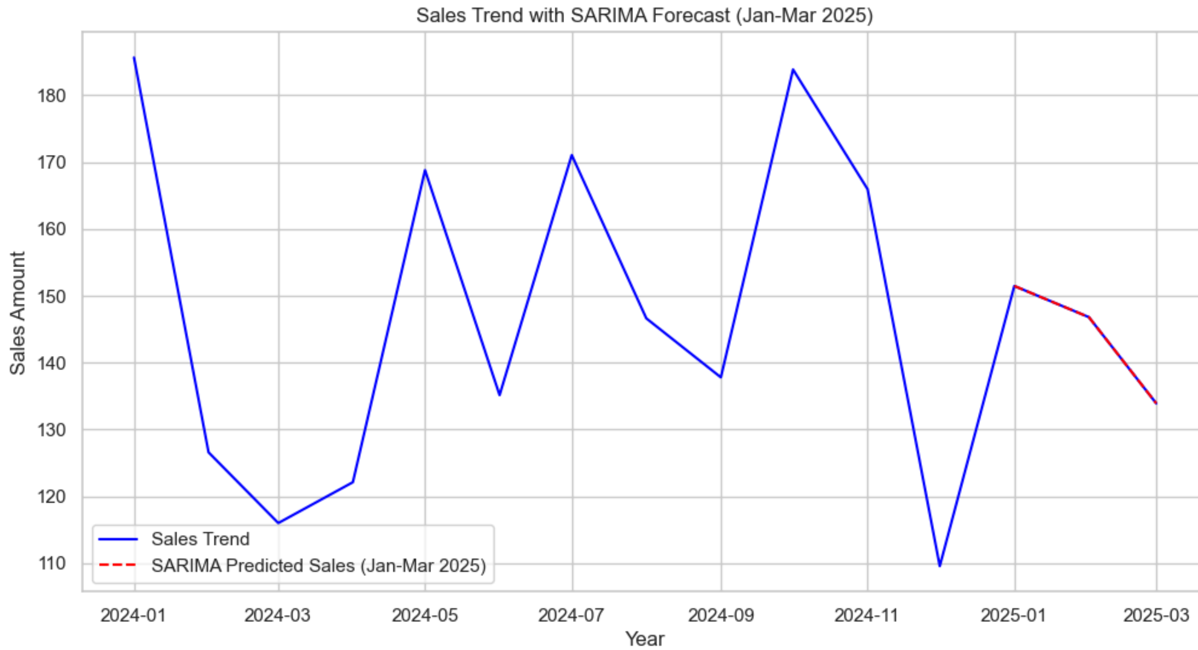


In [86]:
```
import matplotlib.pyplot as plt

# Plot the original monthly sales data (up to the last month)
plt.figure(figsize=(12, 6))
plt.plot(monthly_sales_df.index, monthly_sales_df['TOTAL_SALES_VALUE'], labe

# Extend the plot with the SARIMA forecasted values for Jan-Mar 2025 (dottec
forecast_dates = pd.date_range(start='2025-01-01', end='2025-03-31', freq='M
plt.plot(forecast_dates, predictions.predicted_mean.values, label='SARIMA Pr

# Adding titles and labels
plt.title(f'Sales Prediction with SARIMA Model (Jan-Mar 2025)')
plt.xlabel('Year')
plt.ylabel('Sales Amount')
plt.legend()

# Display the plot
plt.show()
```



In [87]:

```
import matplotlib.pyplot as plt
import pandas as pd

# Create a new dataframe that concatenates the historical data with the fore
forecast_dates = pd.date_range(start='2025-01-01', end='2025-03-31', freq='M
forecast_df = pd.DataFrame({
    'MONTH_YEAR': forecast_dates,
    'TOTAL_SALES_VALUE': predictions.predicted_mean.values
})
monthly_sales_df_t = monthly_sales_df.reset_index()
# Concatenate historical data with the forecasted data
combined_df = pd.concat([monthly_sales_df_t[['MONTH_YEAR', 'TOTAL_SALES_VALU

# Plotting the combined data
plt.figure(figsize=(12, 6))
plt.plot(combined_df['MONTH_YEAR'], combined_df['TOTAL_SALES_VALUE'], label=

# Highlight the forecasted (predicted) values
plt.plot(forecast_df['MONTH_YEAR'], forecast_df['TOTAL_SALES_VALUE'], label=

# Adding titles and labels
plt.title(f'Sales Trend with SARIMA Forecast (Jan-Mar 2025)')
plt.xlabel('Year')
plt.ylabel('Sales Amount')
plt.legend()

# Display the plot
plt.show()
```



-- Trying facebook model (prophet for forecasting)

In [88]:
```
from prophet import Prophet
```

In [89]:
```
tr = train[['TOTAL_SALES_VALUE']]
df_p = tr.reset_index()[["MONTH_YEAR", 'TOTAL_SALES_VALUE']].rename(
    columns={"MONTH_YEAR": "ds", 'TOTAL_SALES_VALUE': "y"}
)
df_p.head()
```

Out[89]:

	ds	y
0	2024-01-01	185.626186
1	2024-02-01	126.579702
2	2024-03-01	116.000676
3	2024-04-01	122.110750
4	2024-05-01	168.781502

In [90]:
```
ts = test.reset_index()[["MONTH_YEAR"]]
ts = ts.rename(columns={"MONTH_YEAR": "ds"})
ts
```

Out[90]:

	ds
0	2024-10-01
1	2024-11-01
2	2024-12-01

- Initialize a prophet model and train it

In [91]:
```
# prophet model initialization
prophet_m = Prophet(seasonality_mode='additive')
prophet_m.fit(df_p)
```

```
05:07:39 - cmdstanpy - INFO - Chain [1] start processing
05:07:41 - cmdstanpy - INFO - Chain [1] done processing
```

Out[91]: `<prophet.forecaster.Prophet at 0x1657506e990>`

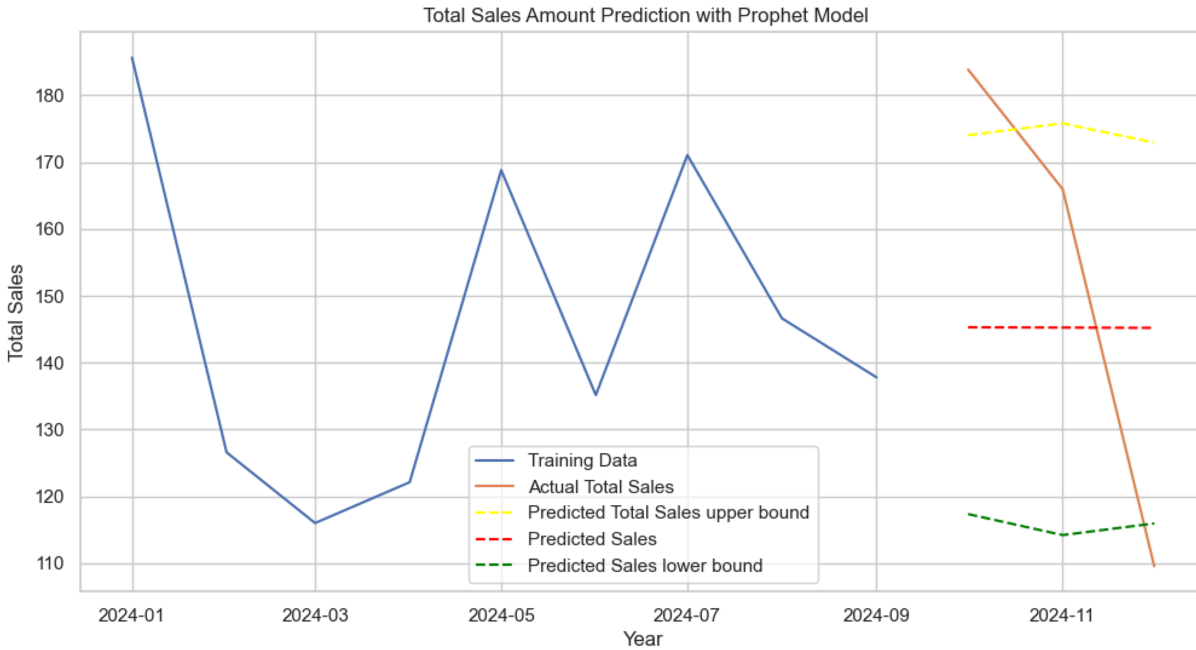
In [92]:
```
# Make predictions
predictions = prophet_m.predict(ts)

predictions.head()
```

Out[92]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	addit
0	2024-10-01	145.280456	117.343479	174.013175	145.280456	145.280456	
1	2024-11-01	145.236724	114.202514	175.774623	145.236723	145.236724	
2	2024-12-01	145.194402	115.942294	172.905135	145.194401	145.194403	

In [93]:
```
predictions = predictions.set_index('ds')
```

In [94]:
```
plt.figure(figsize = (12, 6))
plt.plot(train['TOTAL_SALES_VALUE'], label = 'Training Data')
plt.plot(test[['TOTAL_SALES_VALUE']], label = f'Actual Total Sales')
plt.plot(test.index, predictions['yhat_upper'], label = f'Predicted Total Sa
plt.plot(test.index, predictions['yhat'], label = f'Predicted Sales', linest
plt.plot(test.index, predictions['yhat_lower'], label = f'Predicted Sales lo
plt.title(' Total Sales Amount Prediction with Prophet Model')
plt.xlabel('Year')
plt.ylabel(f'Total Sales')
plt.legend()
plt.show()
```



In [95]:
```
mse = ((predictions['yhat'] - test['TOTAL_SALES_VALUE']) ** 2).mean()
rmse = np.sqrt(mse)

print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
```

```
Mean Squared Error (MSE): 1061.7434158892108
Root Mean Squared Error (RMSE): 32.58440448879204
```

In [96]:
```
mape = calculate_mape(test['TOTAL_SALES_VALUE'], predictions['yhat_lower'])
print(f'MAPE: {mape:.2f}%')
```

```
MAPE: 24.39%
```

In [97]:
```
mape = calculate_mape(test['TOTAL_SALES_VALUE'], predictions['yhat'])
print(f'MAPE: {mape:.2f}%')
```

```
MAPE: 21.99%
```

In [98]:
```
mape = calculate_mape(test['TOTAL_SALES_VALUE'], predictions['yhat_upper'])
print(f'MAPE: {mape:.2f}%')
```

```
MAPE: 23.03%
```

Now train with all the 12 months data and predict 3 comming months

In [99]:
```
prophet_data = monthly_sales_df.reset_index()[["MONTH_YEAR", 'TOTAL_SALES_VA
    columns={"MONTH_YEAR": "ds", 'TOTAL_SALES_VALUE': "y"}
)
prophet_data
```

Out[99]:

	ds	y
0	2024-01-01	185.626186
1	2024-02-01	126.579702
2	2024-03-01	116.000676
3	2024-04-01	122.110750
4	2024-05-01	168.781502
5	2024-06-01	135.140164
6	2024-07-01	171.042631
7	2024-08-01	146.618918
8	2024-09-01	137.791455
9	2024-10-01	183.840551
10	2024-11-01	165.933104
11	2024-12-01	109.557214

In [103…

```
# fit the data into a prophet model
# prophet_m.predict(prophet_data)['yhat']
```

- Making prediction on the 3 new months of 2025 on the dataset using prophet.

In [247…

```
# Create a dataframe for the prediction period (Jan to March 2025)
future_dates = pd.date_range(start='2025-01-01', end='2025-03-31', freq='MS'
future_df = pd.DataFrame({'ds': future_dates})
future_df
```

Out[247…

	ds
0	2025-01-01
1	2025-02-01
2	2025-03-01

In [248…

```
# Make predictions
forecast = prophet_m.predict(future_df)

# Display the predicted values
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
```

Out[248…

	ds	yhat	yhat_lower	yhat_upper
0	2025-01-01	145.150670	116.196032	173.346166
1	2025-02-01	145.106937	115.216542	174.538499
2	2025-03-01	145.067437	114.995680	172.943397

- Sarima model is better for the forecasting here, I think if more data is provided this will give a better prediction, looking forward to it.

Anomaly Detection:

- Identify any unusual spikes or drops in sales performance (Quantity or Value) and explain possible reasons based on the data.

In [124…

```
df_outlier_check = df.copy()
df_outlier_check.columns
```

Out[124…

```
Index(['DATE', 'ANONYMIZED_CATEGORY', 'ANONYMIZED_PRODUCT',
       'ANONYMIZED_BUSINESS', 'ANONYMIZED_LOCATION', 'QUANTITY', 'UNIT_PRIC
E',
       'Month-Year', 'SALES_VALUE', 'TOTAL_SALES_VALUE', 'MONTH_YEAR'],
      dtype='object')
```

In [127…

```
# 1️⃣ **IQR Method for Outlier Detection**
Q1_qty = df_outlier_check['QUANTITY'].quantile(0.25)
Q3_qty = df_outlier_check['QUANTITY'].quantile(0.75)
IQR_qty = Q3_qty - Q1_qty

Q1_val = df_outlier_check['SALES_VALUE'].quantile(0.25)
Q3_val = df_outlier_check['SALES_VALUE'].quantile(0.75)
IQR_val = Q3_val - Q1_val

# Define bounds for anomalies
lower_bound_qty = Q1_qty - 1.5 * IQR_qty
upper_bound_qty = Q3_qty + 1.5 * IQR_qty

lower_bound_val = Q1_val - 1.5 * IQR_val
upper_bound_val = Q3_val + 1.5 * IQR_val

# Mark anomalies
df_outlier_check['IQR_Anomaly_Qty'] = (df_outlier_check['QUANTITY'] < lower_
df_outlier_check['IQR_Anomaly_Val'] = (df_outlier_check['SALES_VALUE'] < low
```

In [128…

```
df_outlier_check[df_outlier_check['IQR_Anomaly_Qty']]
```

Out[128…

	DATE	ANONYMIZED_CATEGORY	ANONYMIZED_PRODUCT	ANONYMI
0	2024-01-01 05:54:00	Category-75	Product-086d	
11	2024-01-01 13:24:00	Category-100	Product-b9c9	
15	2024-01-01 13:30:00	Category-75	Product-086d	
16	2024-01-01 13:40:00	Category-75	Product-6aa1	
17	2024-01-01 13:40:00	Category-100	Product-b9c9	
...	...	...	...	
329841	2024-12-31 18:01:00	Category-99	Product-4646	
329844	2024-12-31 18:01:00	Category-120	Product-0451	
329846	2024-12-31 18:02:00	Category-85	Product-0c64	
329869	2024-12-31 18:19:00	Category-108	Product-ee77	
329874	2024-12-31 18:20:00	Category-120	Product-83fd	

48115 rows × 13 columns

In [129…

```
df_outlier_check[df_outlier_check['IQR_Anomaly_Val']]
```

Out[129…

	DATE	ANONYMIZED_CATEGORY	ANONYMIZED_PRODUCT	ANONYMI
0	2024-01-01 05:54:00	Category-75	Product-086d	
15	2024-01-01 13:30:00	Category-75	Product-086d	
16	2024-01-01 13:40:00	Category-75	Product-6aa1	
17	2024-01-01 13:40:00	Category-100	Product-b9c9	
22	2024-01-01 14:28:00	Category-75	Product-2175	
...	...	...	...	
329747	2024-12-31 17:29:00	Category-120	Product-4156	
329783	2024-12-31 17:46:00	Category-119	Product-66e0	
329818	2024-12-31 17:54:00	Category-119	Product-5ec7	
329838	2024-12-31 18:01:00	Category-121	Product-afb7	
329845	2024-12-31 18:01:00	Category-76	Product-e805	

31306 rows × 13 columns

- Z-Score Method for Outlier Detection**

In [131…

```
df_outlier_check['Z_Score_Qty'] = zscore(df_outlier_check['QUANTITY'])
df_outlier_check['Z_Score_Val'] = zscore(df_outlier_check['SALES_VALUE'])

# Define threshold for anomalies
threshold = 2.5
df_outlier_check['Z_Anomaly_Qty'] = (np.abs(df_outlier_check['Z_Score_Qty'])
df_outlier_check['Z_Anomaly_Val'] = (np.abs(df_outlier_check['Z_Score_Val'])
```

In [133…

```
df_outlier_check[df_outlier_check['Z_Anomaly_Qty']]
```

Out[133…

	DATE	ANONYMIZED_CATEGORY	ANONYMIZED_PRODUCT	ANONYMI
17	2024-01-01 13:40:00	Category-100	Product-b9c9	
51	2024-01-01 15:30:00	Category-120	Product-29ee	
54	2024-01-01 15:30:00	Category-100	Product-392e	
141	2024-01-01 16:02:00	Category-75	Product-086d	
457	2024-01-01 17:47:00	Category-75	Product-2175	
...	...	...	...	
329718	2024-12-31 17:19:00	Category-100	Product-94a8	
329727	2024-12-31 17:19:00	Category-94	Product-f5d3	
329730	2024-12-31 17:19:00	Category-100	Product-f3ee	
329747	2024-12-31 17:29:00	Category-120	Product-4156	
329869	2024-12-31 18:19:00	Category-108	Product-ee77	

5427 rows × 17 columns

In [132…

```
df_outlier_check[df_outlier_check['Z_Anomaly_Val']]
```

Out[132…

	DATE	ANONYMIZED_CATEGORY	ANONYMIZED_PRODUCT	ANONYMI
17	2024-01-01 13:40:00	Category-100	Product-b9c9	
51	2024-01-01 15:30:00	Category-120	Product-29ee	
54	2024-01-01 15:30:00	Category-100	Product-392e	
141	2024-01-01 16:02:00	Category-75	Product-086d	
196	2024-01-01 16:19:00	Category-76	Product-e805	
...	...	...	...	
329281	2024-12-31 08:49:00	Category-100	Product-68e7	
329282	2024-12-31 08:50:00	Category-75	Product-1196	
329316	2024-12-31 09:19:00	Category-75	Product-1196	
329435	2024-12-31 12:52:00	Category-91	Product-3f76	
329718	2024-12-31 17:19:00	Category-100	Product-94a8	

5639 rows × 17 columns

Isolation Forest (Machine Learning-Based Anomaly Detection)

In [135…

```
model = IsolationForest(contamination=0.02, random_state=42)
df_outlier_check['Iso_Anomaly'] = model.fit_predict(df_outlier_check[['QUANT
df_outlier_check['Iso_Anomaly'] = df_outlier_check['Iso_Anomaly'] == -1  # C
```

In [137…

```
df_outlier_check[df_outlier_check['Iso_Anomaly']]
```

Out[137…

	DATE	ANONYMIZED_CATEGORY	ANONYMIZED_PRODUCT	ANONYMI
17	2024-01-01 13:40:00	Category-100	Product-b9c9	
51	2024-01-01 15:30:00	Category-120	Product-29ee	
54	2024-01-01 15:30:00	Category-100	Product-392e	
141	2024-01-01 16:02:00	Category-75	Product-086d	
196	2024-01-01 16:19:00	Category-76	Product-e805	
...	...	...	...	
329718	2024-12-31 17:19:00	Category-100	Product-94a8	
329727	2024-12-31 17:19:00	Category-94	Product-f5d3	
329730	2024-12-31 17:19:00	Category-100	Product-f3ee	
329747	2024-12-31 17:29:00	Category-120	Product-4156	
329869	2024-12-31 18:19:00	Category-108	Product-ee77	

6593 rows × 18 columns

Filter Only Anomalous Sales

In [144…

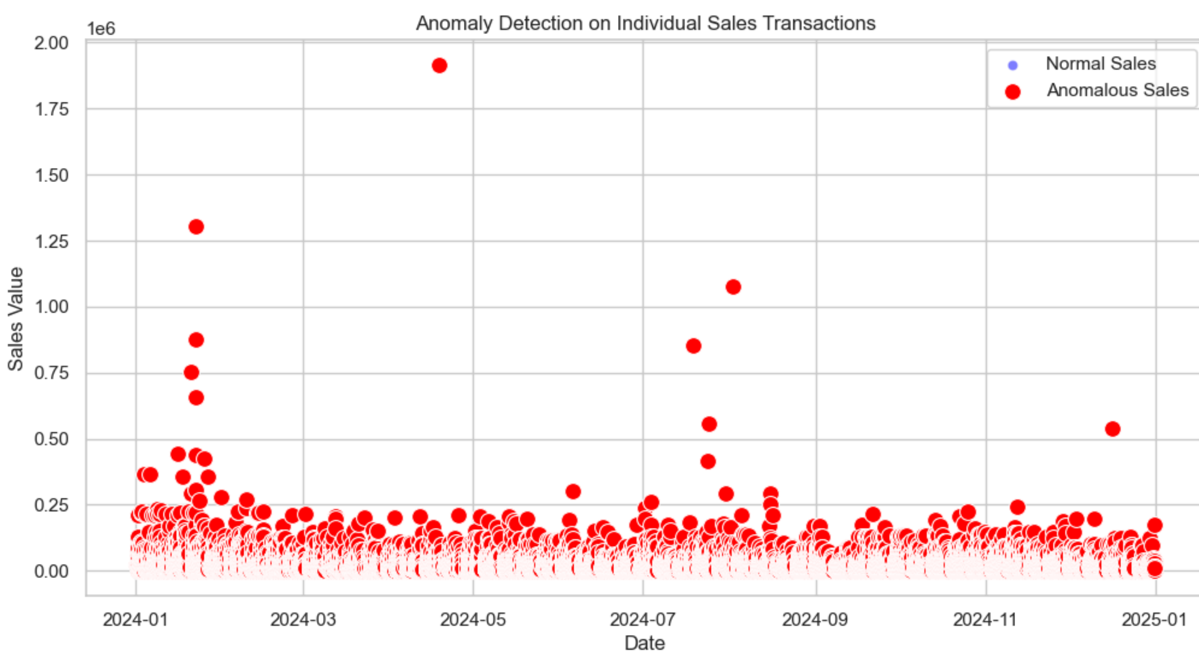
```
anomalies = df_outlier_check[df_outlier_check[['IQR_Anomaly_Qty', 'IQR_Anoma

# Display anomalies
anomalies[['DATE', 'ANONYMIZED_PRODUCT', 'QUANTITY', 'SALES_VALUE',
                 'IQR_Anomaly_Qty', 'IQR_Anomaly_Val',
                 'Z_Anomaly_Qty', 'Z_Anomaly_Val', 'Iso_Anomaly']].head()
```

Out[144…

	DATE	ANONYMIZED_PRODUCT	QUANTITY	SALES_VALUE	IQR_Anomaly_
0	2024-01-01 05:54:00	Product-086d	10	21850.0	
11	2024-01-01 13:24:00	Product-b9c9	5	10800.0	
15	2024-01-01 13:30:00	Product-086d	10	21850.0	
16	2024-01-01 13:40:00	Product-6aa1	10	21990.0	
17	2024-01-01 13:40:00	Product-b9c9	30	64800.0	

In [142…

```
# Visualization - Anomalies in Sales
plt.figure(figsize=(12, 6))
sns.scatterplot(data=df_outlier_check, x='DATE', y='SALES_VALUE', label='Nor
sns.scatterplot(data=anomalies, x='DATE', y='SALES_VALUE', color='red', labe
plt.title('Anomaly Detection on Individual Sales Transactions')
plt.xlabel('Date')
plt.ylabel('Sales Value')
plt.legend()
plt.show()
```



- Correlation Analysis:Examine relationships between Quantity and Value. Provide insights into which factors drive sales performance.

In [104…
```
monthly_sales.head()
```

Out[104…

	MONTH_YEAR	TOTAL_SALES_VALUE	QUANTITY
0	2024-01-01	185.626186	67.526
1	2024-02-01	126.579702	44.063
2	2024-03-01	116.000676	45.381
3	2024-04-01	122.110750	50.554
4	2024-05-01	168.781502	69.551

In [106…
```
monthly_sales[['TOTAL_SALES_VALUE', 'QUANTITY']].corr()
```

Out[106…

	TOTAL_SALES_VALUE	QUANTITY
TOTAL_SALES_VALUE	1.000000	0.831049
QUANTITY	0.831049	1.000000

> Correlation between Quantity and Sales show a Strong Positive corellation of 0.83 .This show that the more quantity more value is derived. (the more the qauntiy the more the sales value.)

- Section 4: Strategic Insights and Recommendations (20 points)

- Product Strategy: Based on your analysis, recommend one product category to prioritize for marketing campaigns. Justify your choice using the data

In [119…
```
# Identify top 5 most valuable products(sales value)
top_10_valuable_products = product_summary.nlargest(20, 'Total_Sales_Value')
top_10_valuable_products
```

Out[119…

	ANONYMIZED_PRODUCT	Total_Quantity	Total_Sales_Value	Sales_Contribut
0	Product-e805	42602	262787281.0	14.854
1	Product-8f75	37566	158797460.0	8.976
2	Product-66e0	46957	70704225.0	3.996
3	Product-29ee	35940	68248274.0	3.857
4	Product-4156	28487	56956007.0	3.219
5	Product-faa5	12764	55751850.0	3.151
6	Product-2175	11985	52581105.0	2.972
7	Product-d09c	18081	50089697.0	2.831
8	Product-3050	23751	44690235.0	2.526
9	Product-b31e	8902	39149239.0	2.213
10	Product-086d	18473	39140645.0	2.212
11	Product-6e9c	13529	36064415.0	2.038
12	Product-14f3	17061	31815441.0	1.798
13	Product-83fd	17053	29238045.0	1.652
14	Product-9a3e	14639	26187371.0	1.480
15	Product-68e7	13004	23855545.0	1.348
16	Product-61dd	8440	22517921.0	1.272
17	Product-1196	4935	21734130.0	1.228
18	Product-1609	5097	21461850.0	1.213
19	Product-4832	5077	21347430.0	1.206

In [116…

```
# Calculate total sales value and total quantity
total_sales_value = product_summary['Total_Sales_Value'].sum()
total_quantity = product_summary['Total_Quantity'].sum()

# Calculate Sales Contribution (% of total sales)
product_summary['Sales_Contribution'] = (product_summary['Total_Sales_Value'

# Calculate Quantity Contribution (% of total quantity)
product_summary['Quantity_Contribution'] = (product_summary['Total_Quantity'

# Calculate Average Selling Price (ASP)
product_summary['ASP'] = product_summary['Total_Sales_Value'] / product_summ
product_summary.head()
```

Out[116…

	ANONYMIZED_PRODUCT	Total_Quantity	Total_Sales_Value	Sales_Contributi
0	Product-0001	286	730730.0	0.0413
1	Product-0031	49	85554.0	0.0048
2	Product-004f	8	39040.0	0.0022
3	Product-02e4	225	58620.0	0.0033
4	Product-031c	1237	2404010.0	0.1358

In [118…

```
product_summary.sort_values(by='Sales_Contribution', ascending=False).head()
```

Out[118…

	ANONYMIZED_PRODUCT	Total_Quantity	Total_Sales_Value	Sales_Contribu
750	Product-e805	42602	262787281.0	14.85
476	Product-8f75	37566	158797460.0	8.97
338	Product-66e0	46957	70704225.0	3.99
127	Product-29ee	35940	68248274.0	3.85
213	Product-4156	28487	56956007.0	3.21

- For a product to prioritise, I choose **PProduct-e805**
- **Why** This product to me seems a potential product to generate more Sales.
  - Currently the product contributes 14% of total sales, making it the most significant revenue Driver.
  - It demand also is high making a contribution of 5.56% so this product seem to prioritised by customers or it sove a particular available niche in the market currently.
- *NOTE*: There also other products that seems to generate great sale thou less sold currently myabe a look into the product also needs to highlight like for product **Product-b31e**

Customer Retention: Identify businesses that have reduced their purchase frequency over time. Suggest strategies to re-engage these customers.

In [127…

```
df.columns
```

Out[127…

```
Index(['DATE', 'ANONYMIZED_CATEGORY', 'ANONYMIZED_PRODUCT',
       'ANONYMIZED_BUSINESS', 'ANONYMIZED_LOCATION', 'QUANTITY', 'UNIT_PRIC
E',
       'Month-Year', 'SALES_VALUE', 'TOTAL_SALES_VALUE', 'MONTH_YEAR'],
      dtype='object')
```

In [129…

```
# group product, thn to get total sales value and quantity of each product
product_summary = df.groupby(['ANONYMIZED_BUSINESS', 'MONTH_YEAR']).agg(
    Total_Quantity=('QUANTITY', 'sum'),
    Total_Sales_Value=('TOTAL_SALES_VALUE', 'sum')
).reset_index()

product_summary.sort_values(by=['ANONYMIZED_BUSINESS', 'MONTH_YEAR']).head()
```

Out[129…

	ANONYMIZED_BUSINESS	MONTH_YEAR	Total_Quantity	Total_Sales_Value
0	Business-0000	2024-06	2	3140.0
1	Business-0000	2024-08	1	185.0
2	Business-0000	2024-10	3	4240.0
3	Business-0000	2024-11	1	2700.0
4	Business-0000	2024-12	1	180.0

In [130…

```
product_summary.sort_values(by=['ANONYMIZED_BUSINESS', 'MONTH_YEAR'])
```

Out[130…

	ANONYMIZED_BUSINESS	MONTH_YEAR	Total_Quantity	Total_Sales_Val
0	Business-0000	2024-06	2	314(
1	Business-0000	2024-08	1	18!
2	Business-0000	2024-10	3	424(
3	Business-0000	2024-11	1	270(
4	Business-0000	2024-12	1	18(
...	...	...	...	
20678	Business-ffd2	2024-09	8	1244{
20679	Business-ffff	2024-09	18	1831(
20680	Business-ffff	2024-10	25	2527!
20681	Business-ffff	2024-11	50	5564!
20682	Business-ffff	2024-12	17	1105!

20683 rows × 4 columns

In [134…

```
# Sort data by business and time
product_summary = product_summary.sort_values(by=['ANONYMIZED_BUSINESS', 'MO

# Calculate month-over-month percentage change
product_summary['Quantity_Change'] = product_summary.groupby('ANONYMIZED_BUS
product_summary['Sales_Value_Change'] = product_summary.groupby('ANONYMIZED_

# Identify businesses with consistent decline over multiple months
declining_businesses = product_summary.groupby('ANONYMIZED_BUSINESS').filter
    lambda x: (x['Quantity_Change'] < 0).sum() >= 3
)
```

```python
# Businesses with declining sales
list(declining_businesses['ANONYMIZED_BUSINESS'].unique())
```

```
Out[134… ['Business-0072',
 'Business-0078',
 'Business-016c',
 'Business-0204',
 'Business-0287',
 'Business-0354',
 'Business-0357',
 'Business-0384',
 'Business-03a3',
 'Business-03ac',
 'Business-03e1',
 'Business-040a',
 'Business-044c',
 'Business-045b',
 'Business-046d',
 'Business-04b6',
 'Business-0523',
 'Business-053b',
 'Business-05d3',
 'Business-0610',
 'Business-069f',
 'Business-06ba',
 'Business-06ed',
 'Business-076a',
 'Business-07de',
 'Business-07ef',
 'Business-0825',
 'Business-0855',
 'Business-0872',
 'Business-089e',
 'Business-08e8',
 'Business-09a0',
 'Business-09ae',
 'Business-09c8',
 'Business-0a38',
 'Business-0ad5',
 'Business-0ad6',
 'Business-0af3',
 'Business-0b68',
 'Business-0b84',
 'Business-0c75',
 'Business-0c77',
 'Business-0c88',
 'Business-0c98',
 'Business-0ca8',
 'Business-0cbc',
 'Business-0cd5',
 'Business-0d26',
 'Business-0d27',
 'Business-0d61',
 'Business-0d9c',
 'Business-0daa',
 'Business-0dbe',
 'Business-0e22',
 'Business-0e56',
 'Business-0e5b',
```

```
'Business-0e75',
'Business-0e77',
'Business-0e7c',
'Business-0e99',
'Business-0ea3',
'Business-0f38',
'Business-0f72',
'Business-101d',
'Business-102d',
'Business-1065',
'Business-108f',
'Business-10e2',
'Business-11da',
'Business-1239',
'Business-126a',
'Business-1274',
'Business-12ce',
'Business-12e1',
'Business-135d',
'Business-136e',
'Business-1371',
'Business-13a7',
'Business-13b0',
'Business-13f9',
'Business-1488',
'Business-149d',
'Business-14b6',
'Business-14ca',
'Business-14dd',
'Business-1510',
'Business-152a',
'Business-1548',
'Business-155f',
'Business-15bc',
'Business-1692',
'Business-1696',
'Business-16e7',
'Business-1710',
'Business-171a',
'Business-17a7',
'Business-1866',
'Business-1886',
'Business-188a',
'Business-18e4',
'Business-18f1',
'Business-18fe',
'Business-194c',
'Business-195b',
'Business-198b',
'Business-1993',
'Business-19c9',
'Business-19dc',
'Business-19e8',
'Business-19f9',
'Business-1a3a',
'Business-1a74',
```

```
'Business-1a76',
'Business-1a8b',
'Business-1a8c',
'Business-1ae8',
'Business-1afb',
'Business-1b52',
'Business-1b81',
'Business-1c6c',
'Business-1c70',
'Business-1c98',
'Business-1cb1',
'Business-1cef',
'Business-1d1c',
'Business-1d4a',
'Business-1d55',
'Business-1d95',
'Business-1dc2',
'Business-1ddb',
'Business-1ddc',
'Business-1e10',
'Business-1e3e',
'Business-1e57',
'Business-1e65',
'Business-1e98',
'Business-1ed2',
'Business-1ede',
'Business-1f08',
'Business-1f42',
'Business-1f50',
'Business-1f55',
'Business-1f6d',
'Business-1f97',
'Business-2019',
'Business-2078',
'Business-20e4',
'Business-20fc',
'Business-211b',
'Business-2130',
'Business-2153',
'Business-2159',
'Business-2163',
'Business-2177',
'Business-218a',
'Business-2229',
'Business-223a',
'Business-2249',
'Business-2282',
'Business-2297',
'Business-22e2',
'Business-22fd',
'Business-2321',
'Business-2326',
'Business-234d',
'Business-2366',
'Business-2368',
'Business-238b',
```

```
'Business-23ac',
'Business-23f0',
'Business-240b',
'Business-242c',
'Business-242f',
'Business-245e',
'Business-2482',
'Business-2485',
'Business-24bc',
'Business-2574',
'Business-258b',
'Business-2590',
'Business-25ce',
'Business-25df',
'Business-25f7',
'Business-25f9',
'Business-260e',
'Business-263c',
'Business-2672',
'Business-268b',
'Business-2693',
'Business-26de',
'Business-270a',
'Business-274a',
'Business-27f9',
'Business-281c',
'Business-2843',
'Business-2848',
'Business-284b',
'Business-288e',
'Business-28b6',
'Business-28e1',
'Business-28e4',
'Business-2943',
'Business-294a',
'Business-29b9',
'Business-29bc',
'Business-29ce',
'Business-2a7a',
'Business-2a83',
'Business-2ab1',
'Business-2afb',
'Business-2b21',
'Business-2b24',
'Business-2b63',
'Business-2b76',
'Business-2b82',
'Business-2b91',
'Business-2ba7',
'Business-2be7',
'Business-2ca8',
'Business-2d82',
'Business-2d89',
'Business-2d8d',
'Business-2db8',
'Business-2e04',
```

```
'Business-2e05',
'Business-2e1a',
'Business-2e23',
'Business-2e37',
'Business-2e4a',
'Business-2e8c',
'Business-2e92',
'Business-2eba',
'Business-2ec3',
'Business-2ed4',
'Business-2ef4',
'Business-2efb',
'Business-2f66',
'Business-2f6a',
'Business-2fd2',
'Business-300a',
'Business-3090',
'Business-3099',
'Business-309b',
'Business-30e8',
'Business-30fc',
'Business-3112',
'Business-3194',
'Business-319b',
'Business-31b8',
'Business-31d6',
'Business-31f8',
'Business-3215',
'Business-323e',
'Business-3280',
'Business-328d',
'Business-32f2',
'Business-3326',
'Business-332c',
'Business-332e',
'Business-3370',
'Business-33ac',
'Business-3410',
'Business-3475',
'Business-348a',
'Business-34bc',
'Business-34f4',
'Business-3515',
'Business-3539',
'Business-35a4',
'Business-35c0',
'Business-3629',
'Business-3683',
'Business-3709',
'Business-3713',
'Business-3731',
'Business-3772',
'Business-37bc',
'Business-37cb',
'Business-37ed',
'Business-37ee',
```

```
'Business-3830',
'Business-3852',
'Business-3880',
'Business-3955',
'Business-3970',
'Business-3a00',
'Business-3a97',
'Business-3b03',
'Business-3b12',
'Business-3ba8',
'Business-3bbc',
'Business-3bce',
'Business-3bd4',
'Business-3bf0',
'Business-3c65',
'Business-3c8e',
'Business-3c92',
'Business-3cac',
'Business-3ccf',
'Business-3cdf',
'Business-3cef',
'Business-3cf8',
'Business-3d20',
'Business-3d23',
'Business-3d5b',
'Business-3d8c',
'Business-3d9c',
'Business-3d9d',
'Business-3db1',
'Business-3dbf',
'Business-3dfc',
'Business-3e2d',
'Business-3e63',
'Business-3e70',
'Business-3ead',
'Business-3eda',
'Business-3edd',
'Business-3ee2',
'Business-3ef6',
'Business-3f6c',
'Business-3f85',
'Business-3fa3',
'Business-3fd8',
'Business-404e',
'Business-4051',
'Business-4087',
'Business-40c2',
'Business-4133',
'Business-414c',
'Business-41f1',
'Business-41f3',
'Business-4219',
'Business-422a',
'Business-4248',
'Business-4293',
'Business-42bf',
```

```
'Business-42df',
'Business-4305',
'Business-4337',
'Business-4342',
'Business-4349',
'Business-434a',
'Business-4372',
'Business-43c3',
'Business-43f5',
'Business-43f8',
'Business-447e',
'Business-44f1',
'Business-450d',
'Business-457d',
'Business-45c0',
'Business-4601',
'Business-4608',
'Business-4651',
'Business-4661',
'Business-468e',
'Business-46f4',
'Business-470a',
'Business-4714',
'Business-476c',
'Business-47fa',
'Business-4804',
'Business-483f',
'Business-4846',
'Business-48f6',
'Business-4919',
'Business-4925',
'Business-4952',
'Business-49c2',
'Business-4a11',
'Business-4a5d',
'Business-4ac4',
'Business-4af0',
'Business-4b2d',
'Business-4b76',
'Business-4b8d',
'Business-4b91',
'Business-4bbb',
'Business-4bc8',
'Business-4c26',
'Business-4c5e',
'Business-4c9e',
'Business-4cad',
'Business-4d1a',
'Business-4d3c',
'Business-4d5c',
'Business-4d69',
'Business-4d8e',
'Business-4d9c',
'Business-4e24',
'Business-4e69',
'Business-4e94',
```

```
'Business-4eea',
'Business-4f4c',
'Business-4fbd',
'Business-4fce',
'Business-4fee',
'Business-4ff2',
'Business-4ffe',
'Business-5088',
'Business-50b5',
'Business-50b6',
'Business-50ba',
'Business-5162',
'Business-5172',
'Business-51e9',
'Business-5207',
'Business-522f',
'Business-5293',
'Business-52a1',
'Business-52d7',
'Business-52fa',
'Business-5319',
'Business-5327',
'Business-5378',
'Business-5407',
'Business-5415',
'Business-5430',
'Business-5435',
'Business-545d',
'Business-5477',
'Business-5494',
'Business-5499',
'Business-54ac',
'Business-54ad',
'Business-54c5',
'Business-5525',
'Business-553e',
'Business-557b',
'Business-557e',
'Business-55d4',
'Business-5600',
'Business-5613',
'Business-5661',
'Business-567a',
'Business-5685',
'Business-5700',
'Business-5760',
'Business-5780',
'Business-57f0',
'Business-5809',
'Business-5829',
'Business-585d',
'Business-586b',
'Business-58a8',
'Business-58fe',
'Business-5953',
'Business-5999',
```

```
'Business-59a0',
'Business-59c1',
'Business-59ed',
'Business-5a62',
'Business-5aa4',
'Business-5ace',
'Business-5b02',
'Business-5b07',
'Business-5b6f',
'Business-5ba6',
'Business-5bac',
'Business-5bb6',
'Business-5bf6',
'Business-5c2e',
'Business-5c56',
'Business-5c6b',
'Business-5c6c',
'Business-5cac',
'Business-5cc9',
'Business-5d02',
'Business-5d3e',
'Business-5d41',
'Business-5d7e',
'Business-5da0',
'Business-5de0',
'Business-5de9',
'Business-5e1d',
'Business-5ebd',
'Business-5ef1',
'Business-5efe',
'Business-5f36',
'Business-5f65',
'Business-5fa7',
'Business-5faf',
'Business-5fc7',
'Business-5fcd',
'Business-5feb',
'Business-5fec',
'Business-5fef',
'Business-5ff6',
'Business-6001',
'Business-6011',
'Business-603c',
'Business-6049',
'Business-6068',
'Business-6087',
'Business-60b8',
'Business-60bd',
'Business-60f4',
'Business-60fb',
'Business-610c',
'Business-6162',
'Business-61e9',
'Business-6326',
'Business-636c',
'Business-639f',
```

```
'Business-63d9',
'Business-63e2',
'Business-643c',
'Business-6468',
'Business-648f',
'Business-64a7',
'Business-64b6',
'Business-64d2',
'Business-6510',
'Business-6541',
'Business-655d',
'Business-658e',
'Business-660d',
'Business-6640',
'Business-6652',
'Business-6685',
'Business-66d6',
'Business-66db',
'Business-66e8',
'Business-6700',
'Business-6720',
'Business-674d',
'Business-6782',
'Business-67a9',
'Business-67b6',
'Business-67ef',
'Business-6823',
'Business-6855',
'Business-6877',
'Business-68a7',
'Business-68a9',
'Business-68ef',
'Business-6950',
'Business-696b',
'Business-69b4',
'Business-69b8',
'Business-69e4',
'Business-6a2b',
'Business-6a8e',
'Business-6a90',
'Business-6a96',
'Business-6ace',
'Business-6ae1',
'Business-6b30',
'Business-6b65',
'Business-6b6b',
'Business-6ba6',
'Business-6bdf',
'Business-6bf6',
'Business-6c27',
'Business-6c8d',
'Business-6ca4',
'Business-6cc2',
'Business-6cd7',
'Business-6d60',
'Business-6daf',
```

```
'Business-6de8',
'Business-6e74',
'Business-6ecc',
'Business-6eda',
'Business-6eff',
'Business-6f0c',
'Business-6f3b',
'Business-6f4d',
'Business-6f76',
'Business-6f99',
'Business-6fd0',
'Business-7002',
'Business-700f',
'Business-7016',
'Business-709d',
'Business-70c0',
'Business-70c4',
'Business-70f8',
'Business-719c',
'Business-71ac',
'Business-727d',
'Business-729e',
'Business-72bc',
'Business-7341',
'Business-736c',
'Business-7372',
'Business-7389',
'Business-7403',
'Business-7456',
'Business-746a',
'Business-7488',
'Business-74b1',
'Business-74dc',
'Business-74f4',
'Business-750a',
'Business-7515',
'Business-755c',
'Business-7580',
'Business-75a6',
'Business-75b3',
'Business-75df',
'Business-75fe',
'Business-7678',
'Business-76d2',
'Business-7745',
'Business-7749',
'Business-7758',
'Business-776d',
'Business-777b',
'Business-77af',
'Business-77b0',
'Business-77bb',
'Business-77fa',
'Business-786c',
'Business-789e',
'Business-78a8',
```

```
'Business-78dd',
'Business-78e2',
'Business-794b',
'Business-7964',
'Business-7982',
'Business-7987',
'Business-79a4',
'Business-7a03',
'Business-7a60',
'Business-7a9b',
'Business-7ab4',
'Business-7af0',
'Business-7af1',
'Business-7b0e',
'Business-7baa',
'Business-7bbc',
'Business-7c25',
'Business-7c3e',
'Business-7c6a',
'Business-7c8a',
'Business-7c9e',
'Business-7cb0',
'Business-7cf4',
'Business-7d23',
'Business-7d24',
'Business-7d69',
'Business-7d7e',
'Business-7d81',
'Business-7dd6',
'Business-7e69',
'Business-7e71',
'Business-7e9d',
'Business-7eb1',
'Business-7eff',
'Business-7f0d',
'Business-7f36',
'Business-7f41',
'Business-7f6a',
'Business-7fa5',
'Business-7fda',
'Business-800e',
'Business-8077',
'Business-80b3',
'Business-8119',
'Business-8157',
'Business-8159',
'Business-8177',
'Business-81b7',
'Business-821e',
'Business-8264',
'Business-826f',
'Business-827a',
'Business-82c7',
'Business-82c9',
'Business-8337',
'Business-838f',
```

```
'Business-8395',
'Business-83a0',
'Business-83cd',
'Business-83dd',
'Business-8401',
'Business-843b',
'Business-8448',
'Business-845f',
'Business-84bc',
'Business-8504',
'Business-8513',
'Business-8535',
'Business-8561',
'Business-856e',
'Business-85ca',
'Business-8603',
'Business-863b',
'Business-8647',
'Business-8680',
'Business-8697',
'Business-86a4',
'Business-86a8',
'Business-86b1',
'Business-8701',
'Business-8754',
'Business-8765',
'Business-876f',
'Business-883a',
'Business-884f',
'Business-8854',
'Business-8870',
'Business-88e2',
'Business-88f5',
'Business-8909',
'Business-890d',
'Business-8914',
'Business-891a',
'Business-898c',
'Business-89ac',
'Business-89da',
'Business-8a31',
'Business-8abb',
'Business-8acb',
'Business-8b36',
'Business-8b4f',
'Business-8b77',
'Business-8b9b',
'Business-8bb0',
'Business-8bb4',
'Business-8bbf',
'Business-8be0',
'Business-8bf5',
'Business-8c49',
'Business-8c70',
'Business-8c9c',
'Business-8d06',
```

```
'Business-8d3c',
'Business-8d52',
'Business-8dfa',
'Business-8e52',
'Business-8e85',
'Business-8ef0',
'Business-8f4e',
'Business-8f6d',
'Business-8f86',
'Business-8f8d',
'Business-8fae',
'Business-905b',
'Business-9066',
'Business-90ad',
'Business-90af',
'Business-912c',
'Business-9140',
'Business-9147',
'Business-9159',
'Business-9166',
'Business-9179',
'Business-9184',
'Business-9193',
'Business-91a4',
'Business-91b3',
'Business-91b5',
'Business-9226',
'Business-9239',
'Business-925d',
'Business-926b',
'Business-9280',
'Business-930d',
'Business-9343',
'Business-93eb',
'Business-9452',
'Business-949f',
'Business-9540',
'Business-95a5',
'Business-95db',
'Business-9632',
'Business-9657',
'Business-965c',
'Business-966e',
'Business-9695',
'Business-96d3',
'Business-96eb',
'Business-9768',
'Business-978e',
'Business-97d2',
'Business-9872',
'Business-98b5',
'Business-98d1',
'Business-98e4',
'Business-9909',
'Business-9911',
'Business-991a',
```

```
'Business-996a',
'Business-9981',
'Business-998c',
'Business-9a67',
'Business-9ab5',
'Business-9add',
'Business-9b3a',
'Business-9baf',
'Business-9bb3',
'Business-9c5a',
'Business-9cb1',
'Business-9cdf',
'Business-9daf',
'Business-9e08',
'Business-9ea0',
'Business-9ec4',
'Business-9edf',
'Business-9faf',
'Business-9fd0',
'Business-a02b',
'Business-a062',
'Business-a078',
'Business-a085',
'Business-a171',
'Business-a196',
'Business-a19a',
'Business-a1a7',
'Business-a1f1',
'Business-a221',
'Business-a223',
'Business-a22f',
'Business-a24a',
'Business-a264',
'Business-a282',
'Business-a2a1',
'Business-a334',
'Business-a3da',
'Business-a3fe',
'Business-a417',
'Business-a427',
'Business-a43d',
'Business-a44a',
'Business-a4ea',
'Business-a4fe',
'Business-a50a',
'Business-a523',
'Business-a58f',
'Business-a5ac',
'Business-a641',
'Business-a658',
'Business-a68c',
'Business-a6a1',
'Business-a6d9',
'Business-a703',
'Business-a712',
'Business-a728',
```

```
'Business-a76c',
'Business-a7aa',
'Business-a7d6',
'Business-a850',
'Business-a85f',
'Business-a8b3',
'Business-a8bd',
'Business-a8d7',
'Business-a8da',
'Business-a8db',
'Business-a91c',
'Business-a999',
'Business-aa2b',
'Business-aa71',
'Business-aa74',
'Business-aab8',
'Business-aabd',
'Business-aad4',
'Business-aaef',
'Business-ab40',
'Business-ab77',
'Business-aba8',
'Business-abca',
'Business-abcb',
'Business-abf9',
'Business-ac17',
'Business-ac7e',
'Business-ad5a',
'Business-ad63',
'Business-adc1',
'Business-adc7',
'Business-ae03',
'Business-ae31',
'Business-ae6f',
'Business-ae7c',
'Business-ae94',
'Business-aec5',
'Business-aef2',
'Business-af19',
'Business-af24',
'Business-afce',
'Business-afe2',
'Business-b019',
'Business-b02f',
'Business-b04e',
'Business-b093',
'Business-b094',
'Business-b104',
'Business-b188',
'Business-b197',
'Business-b1c4',
'Business-b1c6',
'Business-b22c',
'Business-b256',
'Business-b25b',
'Business-b295',
```

```
'Business-b299',
'Business-b303',
'Business-b325',
'Business-b38f',
'Business-b395',
'Business-b412',
'Business-b43f',
'Business-b444',
'Business-b453',
'Business-b473',
'Business-b477',
'Business-b48e',
'Business-b4b9',
'Business-b4f8',
'Business-b51c',
'Business-b587',
'Business-b5a6',
'Business-b5f0',
'Business-b603',
'Business-b649',
'Business-b69f',
'Business-b6eb',
'Business-b710',
'Business-b75e',
'Business-b7a6',
'Business-b7d1',
'Business-b7d7',
'Business-b7e7',
'Business-b816',
'Business-b844',
'Business-b882',
'Business-b898',
'Business-b8e8',
'Business-b903',
'Business-b980',
'Business-b9a6',
'Business-b9b8',
'Business-b9f6',
'Business-ba13',
'Business-ba25',
'Business-ba52',
'Business-ba5f',
'Business-ba97',
'Business-bab3',
'Business-bb3f',
'Business-bb8a',
'Business-bb8c',
'Business-bbc0',
'Business-bc3d',
'Business-bc6a',
'Business-bc7f',
'Business-bceb',
'Business-bcff',
'Business-bd0a',
'Business-bd15',
'Business-bd6d',
```

```
'Business-bd75',
'Business-bd8a',
'Business-bd94',
'Business-be24',
'Business-be73',
'Business-be84',
'Business-bf11',
'Business-bf42',
'Business-bf85',
'Business-bf95',
'Business-bfa2',
'Business-bfc3',
'Business-bfcd',
'Business-bfd5',
'Business-bfdc',
'Business-bff1',
'Business-c024',
'Business-c030',
'Business-c0a9',
'Business-c0ce',
'Business-c105',
'Business-c13c',
'Business-c192',
'Business-c1d8',
'Business-c1e3',
'Business-c206',
'Business-c22f',
'Business-c244',
'Business-c321',
'Business-c38e',
'Business-c399',
'Business-c3a2',
'Business-c3bc',
'Business-c3cb',
'Business-c428',
'Business-c431',
'Business-c440',
'Business-c461',
'Business-c472',
'Business-c483',
'Business-c4c7',
'Business-c4e6',
'Business-c4f8',
'Business-c569',
'Business-c574',
'Business-c590',
'Business-c5e8',
'Business-c5ff',
...]
```

- Strategies to re-engage this Customers:
    - Making Sure this bussines stock the most favaourite Products by customers.
    - Run A/B Testing to try engage on there preferences

- Maybe Introduction of discount traffis on these business.
    - Looking into competitors around to maybe thet have drive our customers
    - Doing Campaign awareness on things being introduced in these busines around there vicinities.

Operational Efficiency: Suggest improvements to inventory management or supply chain processes based on trends in product performance and seasonal demand.

- using The forecasting Output(sarima model seemed to capture the seasonality) we can now how to restocked, more data thou is need to reaaly capture the seasonalty issue here.
- Maybe imporving Invertory management by classifying products based on demand, so we can make sure high-demand rpoducts always need to available.

Bonus Section: Open-Ended Problem (Optional, 10 points)

- Predictive Analysis: Identify external factors that could influence sales (e.g., economic conditions, competitor actions). Propose a methodology to incorporate such factors into future analyses.

    - I have worked previous on predictive model @ KRa to predict revenue collection, And I want to identity such factors, mostly is trial and error on all the features we think might be affecting the sales. For example we might need to look into the country's GDP, Politics effects, look into competitors effect on the market also, maybe have a marco and micro features listed and maybe we track them so we utilized them or model training.
- Scalability: If the dataset were 10 times larger, what optimizations would you implement for data storage, processing, and analysis?

    - This is where Data Modelling and Warehousing skill come in. Modelling has several Design from common One **star**, **snowflake** schemas, Data vault. Since our data is currently sales record, I think developing a warehouse using star schema approach for a start will be great, maybe with time where need come in or more dimension or slicing Anaysis Needs arise we might consider a Snoflake approach.