Business problems that can be solved using this dataset:

1. Sales Performance Analysis: Which products are excelling in sales, and which are underperforming?
2. Regional Market Analysis: Which stores are experiencing strong sales, and which ones are lagging?
3. Profit Margin Analysis: Does the profit margin significantly impact sales?
4. Efficiency of Sales Methods: Which sales method is more effective - in-store or online?
5. Price Optimization: Is there a specific price range that achieves better sales than others?
6. Product Portfolio Optimization I: Determine which products are most profitable, segmented by location.
7. Market Expansion Opportunities: Assess the best and worst performing stores based on their locations.
8. Time Series Analysis: Investigate whether there has been a consistent sales trend over time or any noticeable monthly trends.
9. Predictive Sales Analysis: Develop a forecast for monthly sales.

In [1]:
```
#Importing the basic librarires fot analysis

import numpy as np
import pandas as pd
import seaborn as sns
import os
import opendatasets as od
import matplotlib.pyplot as plt
plt.style.use("ggplot")  #using style ggplot

%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
import datetime as dt
import plotly.graph_objects as go
import plotly.express as px

from wordcloud import WordCloud, STOPWORDS
import re
import folium
```

In [2]:
```
# Download the data set files

# Assign the Kaggle data set URL into variable
dataset = 'https://www.kaggle.com/datasets/heemalichaudhari/adidas-sales-dat
# Using opendatasets let's download the data sets
od.download(dataset)
```

```
Skipping, found downloaded files in "./adidas-sales-dataset" (use force=True
to force download)
```

In [3]:
```
data_dir ='./adidas-sales-dataset'
```

In [4]:
```
os.listdir(data_dir)
```

Out[4]:
```
['~$Adidas US Sales Datasets.xlsx', 'Adidas US Sales Datasets.xlsx']
```

In [5]:
```
#Importing the dataset

data = pd.read_excel(r'./adidas-sales-dataset/Adidas US Sales Datasets.xlsx'
```

In [6]:
```
data.head(5)
```

Out[6]:

	Retailer	Retailer ID	Invoice Date	Region	State	City	Product	Price per Unit	Units Sold	Total Sales
0	Foot Locker	1185732	2020-01-01	Northeast	New York	New York	Men's Street Footwear	50.0	1200	600000.0
1	Foot Locker	1185732	2020-01-02	Northeast	New York	New York	Men's Athletic Footwear	50.0	1000	500000.0
2	Foot Locker	1185732	2020-01-03	Northeast	New York	New York	Women's Street Footwear	40.0	1000	400000.0
3	Foot Locker	1185732	2020-01-04	Northeast	New York	New York	Women's Athletic Footwear	45.0	850	382500.0
4	Foot Locker	1185732	2020-01-05	Northeast	New York	New York	Men's Apparel	60.0	900	540000.0

In [11]:
```
# Overview of the data tail
data.tail()
```

Out[11]:

	Retailer	Retailer ID	Invoice Date	Region	State	City	Product	Price per Unit	U
9643	Foot Locker	1185732	2021-01-24	Northeast	New Hampshire	Manchester	Men's Apparel	50.0	
9644	Foot Locker	1185732	2021-01-24	Northeast	New Hampshire	Manchester	Women's Apparel	41.0	
9645	Foot Locker	1185732	2021-02-22	Northeast	New Hampshire	Manchester	Men's Street Footwear	41.0	
9646	Foot Locker	1185732	2021-02-22	Northeast	New Hampshire	Manchester	Men's Athletic Footwear	42.0	
9647	Foot Locker	1185732	2021-02-22	Northeast	New Hampshire	Manchester	Women's Street Footwear	29.0	

- `Retailer` : Names such as Foot Locker, Walmart, Sports Direct, and West Gear.
- `Retailer ID` : A unique identifier for each retailer.
- `Invoice Date` : The date the invoice was generated.
- `Region` : Geographic areas like West, Northeast, Southeast, South, and Mid-west.
- `Product` : Categories like Men's and Women's Street and Athletic Footwear, and Apparel.
- `Price per Unit` : The cost of a single item of a specific product.
- `Units Sold` : The quantity of a product sold in a specific timeframe.
- `Total Sales` : The total revenue from sales over a certain period.
- `Operating Profit` : A measure of the profitability from the main business operations.
- `Operating Margin` : A ratio indicating the profitability of the business operations.
- `Sales Method` : The channels through which sales were made, including In-store, Outlet, and Online.

In [7]:
```
# Show shape dataset
data.shape
```

Out[7]:
```
(9648, 13)
```

In [8]:
```
# Review duplicate datset
data.duplicated().sum()
```

Out[8]:
```
0
```

In [9]:
```
# Info of the dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9648 entries, 0 to 9647
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Retailer          9648 non-null   object
 1   Retailer ID       9648 non-null   int64
 2   Invoice Date      9648 non-null   datetime64[ns]
 3   Region            9648 non-null   object
 4   State             9648 non-null   object
 5   City              9648 non-null   object
 6   Product           9648 non-null   object
 7   Price per Unit    9648 non-null   float64
 8   Units Sold        9648 non-null   int64
 9   Total Sales       9648 non-null   float64
 10  Operating Profit  9648 non-null   float64
 11  Operating Margin  9648 non-null   float64
 12  Sales Method      9648 non-null   object
dtypes: datetime64[ns](1), float64(4), int64(2), object(6)
memory usage: 980.0+ KB
```

In [10]:
```
# Null values
data.isnull().sum()
```

Out[10]:
```
Retailer            0
Retailer ID         0
Invoice Date        0
Region              0
State               0
City                0
Product             0
Price per Unit      0
Units Sold          0
Total Sales         0
Operating Profit    0
Operating Margin    0
Sales Method        0
dtype: int64
```

In [12]:
```
# Remove non-numeric characters from the columns
data['Total Sales'] = data['Total Sales'].astype(str).str.replace(r'[$, ]',
data['Units Sold'] = data['Units Sold'].astype(str).str.replace(r'[$, ]', ''
data['Operating Profit'] = data['Operating Profit'].astype(str).str.replace(
data['Operating Margin'] = data['Operating Margin'].astype(str).str.replace(
```

In [13]:
```
# Convert Invoice Date to DateTime Object.
data['Invoice Date'] = pd.to_datetime(data['Invoice Date'])
```

1 . Analisis Kinerja Penjualan: Produk mana yang unggul dalam penjualan, dan mana yang berkinerja buruk?

In [14]:
```
# First lets find out the total revenue
total_revenue = data['Total Sales'].sum()
total_revenue
```

Out[14]:
```
899902125.0
```

In [15]:
```
# Aggregating total sales and units sold for each product
product_sales = data.groupby('Product').agg({'Total Sales': 'sum', 'Units Sc

# Sorting products by total sales in descending order to identify top-perfor
top_performing_products = product_sales.sort_values(by='Total Sales', ascend

# Displaying the results
print("Top-Performing Products:")
top_performing_products
```

```
Top-Performing Products:
```

Out[15]:

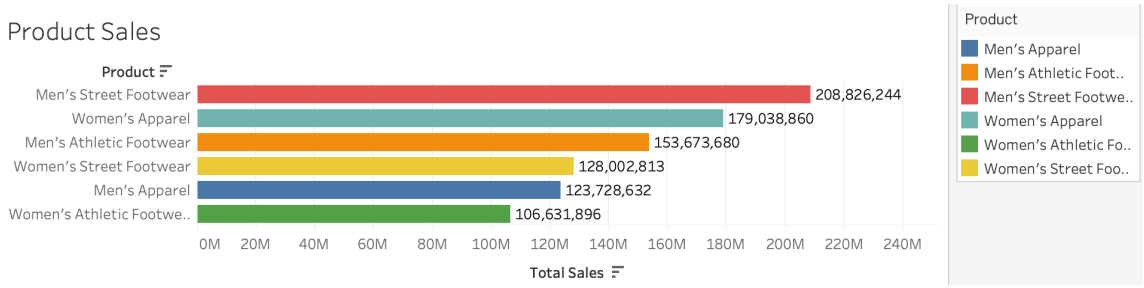
	Product	Total Sales	Units Sold
2	Men's Street Footwear	208826244.0	593320.0
3	Women's Apparel	179038860.0	433827.0
1	Men's Athletic Footwear	153673680.0	435526.0
5	Women's Street Footwear	128002813.0	392269.0
0	Men's Apparel	123728632.0	306683.0

In [16]:
```
# Sorting products by total sales in ascending order to identify underperfor
underperforming_products = product_sales.sort_values(by='Total Sales', ascer

print("\nUnderperforming Products:")
underperforming_products
```

```
Underperforming Products:
```

Out[16]:

	Product	Total Sales	Units Sold
4	Women's Athletic Footwear	106631896.0	317236.0
0	Men's Apparel	123728632.0	306683.0
5	Women's Street Footwear	128002813.0	392269.0
1	Men's Athletic Footwear	153673680.0	435526.0
3	Women's Apparel	179038860.0	433827.0



`Key Insight`:

- `Men's Street Footwear` emerges as the top-performing category with robust sales amounting to $208,826,244, highlighting a strong market preference.
- In contrast `Women's Athletic Footwear` shows a relatively lower performance, recording sales of $106,631,896.

This disparity suggests potential growth areas and indicates a need for targeted strategies to enhance the appeal of underperforming categories like Women's Athletic Footwear.

2, Analisis Pasar Regional: Toko mana yang mengalami penjualan yang kuat, dan toko mana yang tertinggal?

In [17]:
```
# Group by 'City' and 'Retailer', and sum the 'Units Sold'
three_columns_grouped = data.groupby(['City', 'Retailer'])['Units Sold'].sum

# Sorting the results within each city to find the top and worst performing
three_columns_sorted = three_columns_grouped.sort_values(by=['City', 'Units

# Getting the top performing retailer in each city
top_performers = three_columns_sorted.groupby('City').head(1)

# Getting the worst performing retailer in each city
worst_performers = three_columns_sorted.groupby('City').tail(1)
```

In [18]:
```
# Display the results
print("Top Performing Retailers in Each City:")
top_performers.head()
```

```
Top Performing Retailers in Each City:
```

Out[18]:

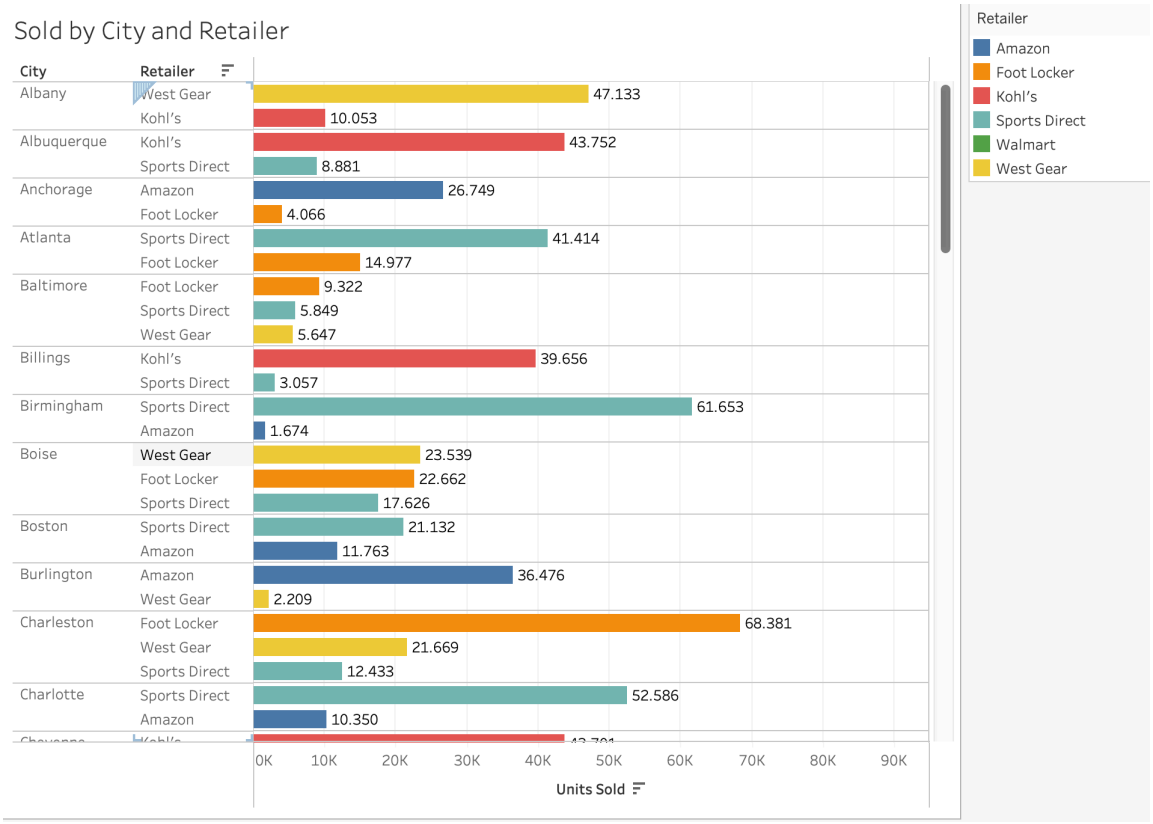
	City	Retailer	Units Sold
1	Albany	West Gear	47133.0
2	Albuquerque	Kohl's	43752.0
4	Anchorage	Amazon	26749.0
7	Atlanta	Sports Direct	41414.0
8	Baltimore	Foot Locker	9322.0

In [19]:
```
print("\nWorst Performing Retailers in Each City:")
worst_performers.head()
```

```
Worst Performing Retailers in Each City:
```

Out[19]:

	City	Retailer	Units Sold
0	Albany	Kohl's	10053.0
3	Albuquerque	Sports Direct	8881.0
5	Anchorage	Foot Locker	4066.0
6	Atlanta	Foot Locker	14977.0
10	Baltimore	West Gear	5647.0




```
Wawasan Utama:
```

- West Gear di Albany dan Kohl's di Albuquerque muncul sebagai yang berkinerja terbaik di kota masing-masing, dengan West Gear menjual 47.133 unit dan Kohl's menjual 43.752 unit .
- Sebaliknya, yang berkinerja terendah adalah Kohl's di Albany dan Sports Direct di Albuquerque, dengan penjualan masing-masing 10.053 dan 8.881 unit.

3. Analisis Margin Laba: Apakah laba Operasional berdampak signifikan terhadap penjualan?

In [20]:
```
# We can answer the above question in the traditional long way using tables

# Correlation bewtween Operating Profit and Total Sales
correlation = data['Operating Profit'].corr(data['Total Sales'])
correlation
```

Out[20]:
```
0.9563074349716087
```

In [21]:
```
# Correlation bewtween Operating Profit and Total Sales
correlation = data['Operating Profit'].corr(data['Units Sold'])
correlation
```

Out[21]:
```
0.8923793765537954
```

Selain itu, ada juga korelasi positif antara Unit Terjual dan Laba Operasional.

1. Korelasinya positif, 0,9563
2. Hal ini menunjukkan bahwa jika Total Penjualan naik maka Laba Operasional juga naik.
3. Jadi jawaban pertanyaan nomor 3 adalah YA. Laba Operasional memang mempengaruhi Unit Terjual secara positif

4. Efisiensi Metode Penjualan: Metode penjualan mana yang lebih efektif – di dalam toko atau online?

In [24]:
```
# Finding the unique column in the df

sales_method_col = data['Sales Method']
unique_sales_method_col = sales_method_col.unique()
unique_sales_method_col
```

Out[24]:
```
array(['In-store', 'Outlet', 'Online'], dtype=object)
```

In [26]:
```
pip install us
```

```
Collecting us
  Downloading us-3.1.1.tar.gz (14 kB)
  Preparing metadata (setup.py) ... done
Collecting jellyfish==0.11.2 (from us)
  Downloading jellyfish-0.11.2-cp311-cp311-macosx_10_7_x86_64.whl.metadata
(2.5 kB)
Downloading jellyfish-0.11.2-cp311-cp311-macosx_10_7_x86_64.whl (328 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 328.9/328.9 kB 914.1 kB/s eta 0:0
0:00a 0:00:01
Building wheels for collected packages: us
  Building wheel for us (setup.py) ... done
  Created wheel for us: filename=us-3.1.1-py3-none-any.whl size=12546 sha256
=7e4d519dda4eb24084a9e74e40f7b22d024e8dc947a42c7ff2ab738017b60b26
  Stored in directory: /Users/burhanudin/Library/Caches/pip/wheels/88/1d/36/
4c39b41019d5c316628db111f85de00af4476af5b9c5ddccc3
Successfully built us
Installing collected packages: jellyfish, us
  Attempting uninstall: jellyfish
    Found existing installation: jellyfish 1.0.1
    Uninstalling jellyfish-1.0.1:
      Successfully uninstalled jellyfish-1.0.1
Successfully installed jellyfish-0.11.2 us-3.1.1
Note: you may need to restart the kernel to use updated packages.
```

In [27]:
```
from us import states

state_column = data['State']

# Get the two-letter abbreviations of the state names
state_abbreviations = []
for state in state_column:
    try:
        # Lookup state by name or existing abbreviation
        state_abbreviation = states.lookup(state).abbr
    except AttributeError:
        # In case the state is not found, set the abbreviation to None
        state_abbreviation = None
    state_abbreviations.append(state_abbreviation)

# Create a new column in the dataframe with the two-letter abbreviations
data['State Abbreviation'] = state_abbreviations
```

In [28]:
```
# Checking the contents of the new column
data['State Abbreviation'].head()
```

Out[28]:
```
0    NY
1    NY
2    NY
3    NY
4    NY
Name: State Abbreviation, dtype: object
```

4.1 Demonstrating Total Sales by State, Product, and Sales Method in the US using Choropleth Map

In [29]:

```
# Removing currency symbols and converting 'Total Sales' to numeric
data['Total Sales'] = data['Total Sales'].replace('[\$,]', '', regex=True).a

# Initialize the figure
fig = go.Figure()

# List of unique products and sales methods
products = data['Product'].unique()
sales_methods = data['Sales Method'].unique()

# Add traces for each combination of product and sales method
for product in products:
    for method in sales_methods:
        filtered_df = data[(data['Product'] == product) & (data['Sales Metho
        state_sales = filtered_df.groupby('State Abbreviation')['Total Sales

        fig.add_trace(
            go.Choropleth(
                locations=state_sales['State Abbreviation'],
                z=state_sales['Total Sales'],
                locationmode='USA-states',
                colorscale='Viridis',
                name=f"{product} - {method}",
                showscale=True,
                visible=False  # Initially, all traces are hidden
            )
        )

# Update layout with dropdown menus
product_buttons = [
    {'label': product, 'method': 'update', 'args': [{'visible': [trace.name.
    for product in products
]

sales_method_buttons = [
    {'label': method, 'method': 'update', 'args': [{'visible': [method in tr
    for method in sales_methods
]

fig.update_layout(
    updatemenus=[
        {'buttons': product_buttons, 'direction': 'down', 'showactive': True
        {'buttons': sales_method_buttons, 'direction': 'down', 'showactive':
    ],
    geo=dict(scope='usa'),
    title="Total Sales by State, Product, and Sales Method"
)

# Initially displaying the first product and first sales method
if fig.data:
```

```
    fig.data[0].visible = True

fig.show()
```

In [30]:
```
#Which method is best
# Segmenting and aggregating data by 'Sales Method'
sales_method_grouped = data.groupby('Sales Method').agg({'Total Sales': 'sum

# Calculating the Operating Margin for each sales method
sales_method_grouped['Operating Margin'] = sales_method_grouped['Operating P

sales_method_grouped
```

Out[30]:

Sales Method	Total Sales	Operating Profit	Operating Margin
In-store	356643750.0	1.275913e+08	0.357756
Online	247672882.0	9.655518e+07	0.389850
Outlet	295585493.0	1.079883e+08	0.365337

Tafsirkan tabel di atas dengan kata-kata agar lebih mudah dipahami.

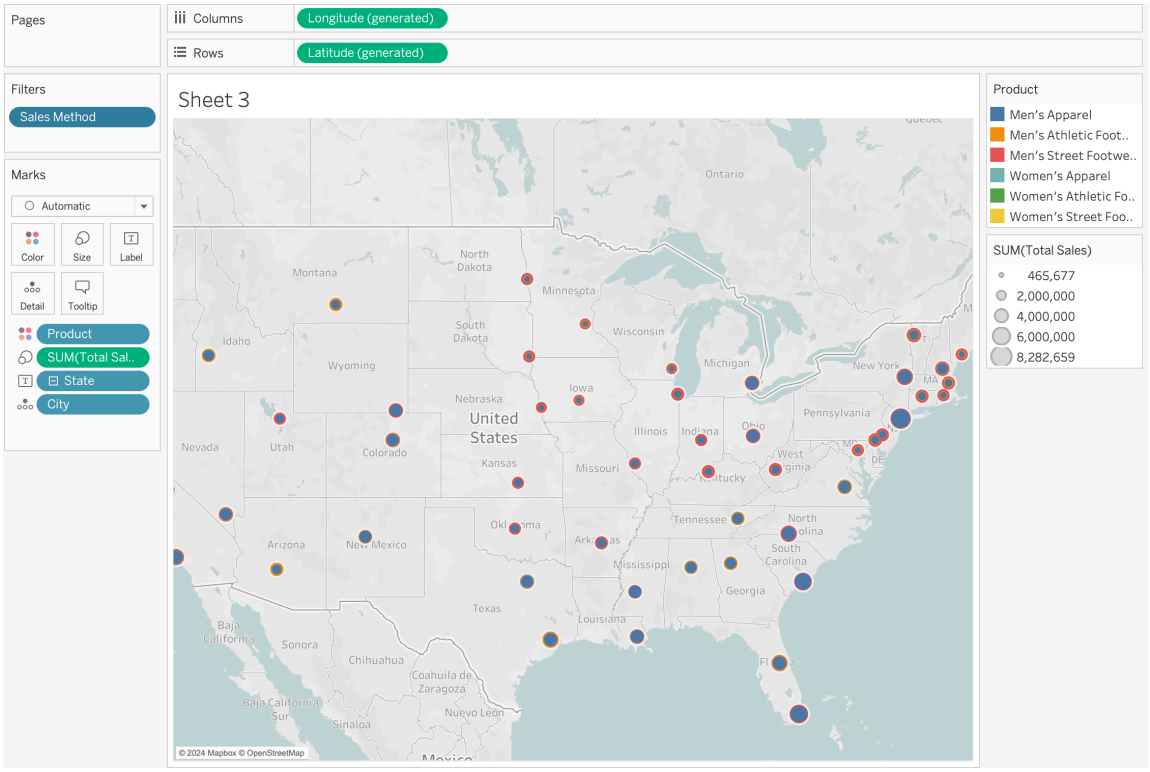
Total Penjualan: • Di dalam toko: 247,672,882 • Outlet: $295,585,493 Penjualan di dalam toko memiliki total penjualan tertinggi, diikuti oleh outlet dan kemudian penjualan online.

Laba Operasional: • Di dalam toko: 96.555.180 (kurang-lebih) • Outlet: $107.988.300 (kurang-lebih) Mirip dengan total penjualan, penjualan di dalam toko memimpin dalam laba operasional, diikuti oleh outlet dan kemudian online.

Margin Operasional (Laba Operasional dibagi Total Penjualan): • Di dalam toko: 0,357756 (atau 35,78%) • Online: 0,389850 (atau 38,99%) • Outlet: 0,365337 (atau 36,53%)

Di sini, meskipun penjualan di dalam toko memiliki total penjualan dan keuntungan tertinggi, penjualan online memiliki margin operasi tertinggi, yang menunjukkan profitabilitas yang lebih tinggi dibandingkan dengan penjualan yang dihasilkan. Wawasan: • Dalam hal Total Penjualan dan Laba Operasional: Metode penjualan di dalam toko adalah yang paling efektif, menghasilkan total penjualan dan laba operasional tertinggi. • Dalam hal Margin Operasional: Penjualan online adalah yang paling efektif, yang menunjukkan bahwa meskipun total penjualan dan laba operasional lebih rendah dibandingkan penjualan di toko, namun profitabilitas relatif terhadap penjualan yang dihasilkan lebih tinggi.

Saat memutuskan metode penjualan mana yang lebih efektif, itu tergantung pada apa yang diprioritaskan oleh bisnis. Jika fokusnya adalah memaksimalkan total pendapatan dan keuntungan, maka penjualan di dalam toko akan lebih efektif. Namun jika fokusnya pada efisiensi dalam hal keuntungan yang dihasilkan per dolar penjualan, maka penjualan online lebih efektif.



5.Optimasi Harga: Apakah ada kisaran harga tertentu yang menghasilkan penjualan lebih baik daripada yang lain?

In [31]:
```
# Define the price bins
bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130]

# Create a new column for the price range
data['Price Range'] = pd.cut(data['Price per Unit'], bins)

# Group by the price range and sum the total sales
sales_by_price_range = data.groupby('Price Range')['Total Sales'].sum().rese

# Sorting the results to see which price range has the highest sales
sorted_sales_by_price_range = sales_by_price_range.sort_values(by='Total Sal

# Results are sorted based on Total Sales
sorted_sales_by_price_range
```

```
/var/folders/f8/1kgfq8w13mqfvx1917j48k1c0000gp/T/ipykernel_14714/623158052.p
y:8: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a
future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
```

Out[31]:

	Price Range	Total Sales
4	(40, 50]	220666307.0
5	(50, 60]	210865002.0
6	(60, 70]	190679285.0
3	(30, 40]	134726187.0
7	(70, 80]	52787579.0
8	(80, 90]	33547420.0
2	(20, 30]	29636023.0
9	(90, 100]	14468685.0
1	(10, 20]	6574478.0
11	(110, 120]	3080000.0
10	(100, 110]	2785706.0
0	(0, 10]	85453.0
12	(120, 130]	0.0

6. Optimasi Portofolio Produk: Tentukan produk mana yang paling menguntungkan, tersegmentasi berdasarkan lokasi.

In [33]:
```
# Grouping data by location and product
grouped_data = data.groupby(['City', 'Product']).agg({'Operating Profit': 's

# Calculating Profit Margin
grouped_data['Profit Margin'] = grouped_data['Operating Profit'] / grouped_d

# Sorting within each location to find the most profitable products
grouped_data.sort_values(by=['City', 'Profit Margin'], ascending=[True, Fals

# Optionally, display the top 5 most profitable products for each city
top_products_by_city = grouped_data.groupby('City').head(5)

top_products_by_city
```

Out[33]:

	City	Product	Operating Profit	Total Sales	Profit Margin
3	Albany	Women's Apparel	2343150.19	4779109.0	0.490290
2	Albany	Men's Street Footwear	2649927.25	5773987.0	0.458942
0	Albany	Men's Apparel	1354932.08	3873553.0	0.349791
1	Albany	Men's Athletic Footwear	1329200.59	3829596.0	0.347086
4	Albany	Women's Athletic Footwear	908094.68	2973011.0	0.305446
...	...	...	...	...	...
309	Wilmington	Women's Apparel	1016327.95	2520405.0	0.403240
310	Wilmington	Women's Athletic Footwear	500967.12	1242580.0	0.403167
311	Wilmington	Women's Street Footwear	605627.89	1502297.0	0.403135
308	Wilmington	Men's Street Footwear	1244055.88	3522197.0	0.353205
307	Wilmington	Men's Athletic Footwear	664729.64	1884159.0	0.352799

260 rows × 5 columns

In [34]:

```
# Grouping data by product and city and summing total sales
grouped_data = data.groupby(['Product', 'City'])['Total Sales'].sum().reset_

# For each product, find the city where it performs the best and the worst
best_worst_performing_cities = []

for product in grouped_data['Product'].unique():
    product_data = grouped_data[grouped_data['Product'] == product]
    best_city = product_data[product_data['Total Sales'] == product_data['To
    worst_city = product_data[product_data['Total Sales'] == product_data['T
    best_worst_performing_cities.extend([best_city, worst_city])

# Concatenating the results into a single DataFrame
best_worst_performing_cities_df = pd.concat(best_worst_performing_cities).re

# Display the results
print("Best and Worst Performing Cities for Each Product:")
best_worst_performing_cities_df
```

```
Best and Worst Performing Cities for Each Product:
```

Out[34]:

	Product	City	Total Sales
0	Men's Apparel	New York	6835166.0
1	Men's Apparel	Omaha	530197.0
2	Men's Athletic Footwear	New York	6301528.0
3	Men's Athletic Footwear	Omaha	942983.0
4	Men's Street Footwear	Charleston	9479502.0
5	Men's Street Footwear	Omaha	2131074.0
6	Women's Apparel	Charleston	8147789.0
7	Women's Apparel	Omaha	1202661.0
8	Women's Athletic Footwear	New York	5201048.0
9	Women's Athletic Footwear	Omaha	465677.0
10	Women's Street Footwear	San Francisco	5549840.0
11	Women's Street Footwear	Omaha	656446.0

7. Peluang Perluasan Pasar: Menilai toko dengan kinerja terbaik dan terburuk berdasarkan lokasinya.

In [35]:
```
# Convert the dictionary to a DataFrame, if not already done
df = pd.DataFrame(data)

# Grouping Data by both City and Store Name
grouped_data = df.groupby(['City', 'Retailer']).agg({
    'Total Sales': 'sum',
    'Operating Profit': 'sum'
}).reset_index()

# Calculating Performance Metrics
grouped_data['Profit Margin'] = grouped_data['Operating Profit'] / grouped_d

# Sorting by Total Sales and Profit Margin
grouped_data = grouped_data.sort_values(by=['City', 'Total Sales', 'Profit M

# Displaying the grouped data
print("Store Performance by City:")
grouped_data
```

```
Store Performance by City:
```

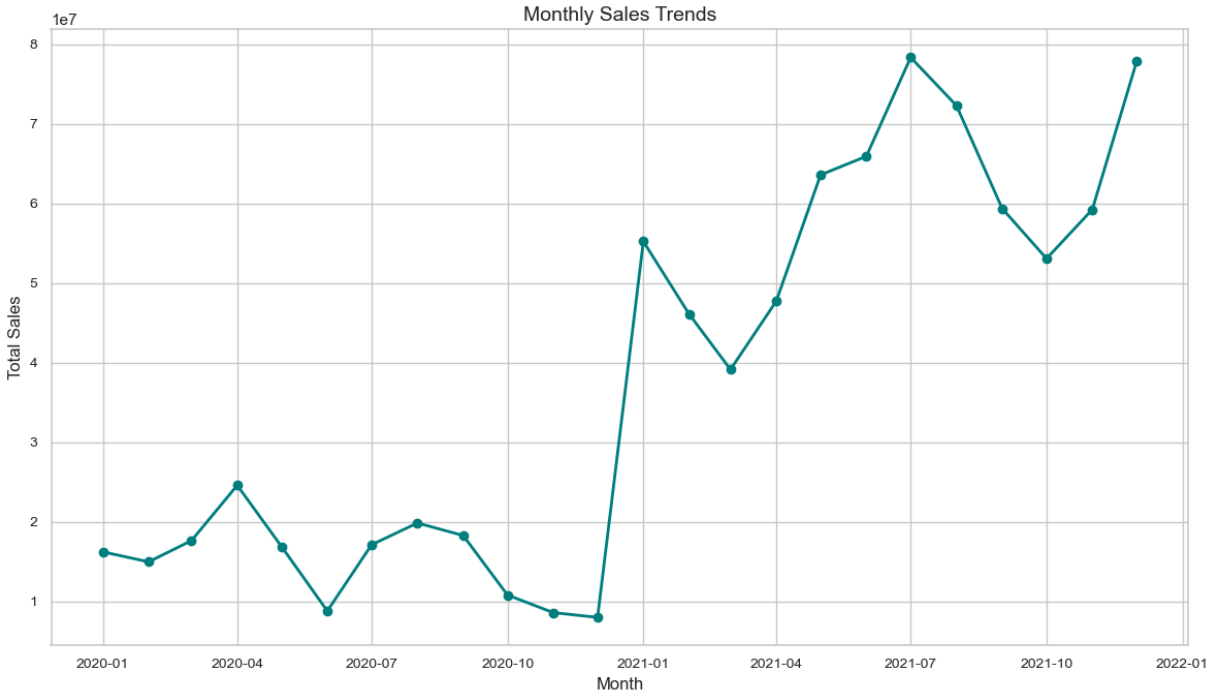
Out[35]:

	City	Retailer	Total Sales	Operating Profit	Profit Margin
1	Albany	West Gear	20735165.0	8062399.80	0.388827
0	Albany	Kohl's	3692639.0	1367451.11	0.370318
2	Albuquerque	Kohl's	17065965.0	5783668.15	0.338901
3	Albuquerque	Sports Direct	2799051.0	954392.26	0.340970
4	Anchorage	Amazon	13365025.0	4143804.75	0.310048
...	...	...	...	...	...
103	St. Louis	West Gear	1701133.0	681457.25	0.400590
105	Wichita	Kohl's	6451914.0	2279774.62	0.353349
104	Wichita	Foot Locker	3520950.0	1230372.67	0.349443
106	Wilmington	Foot Locker	8387568.0	3077352.68	0.366895
107	Wilmington	Kohl's	3910844.0	1446997.38	0.369996

108 rows × 5 columns

8. Analisis Rangkaian Waktu: Selidiki apakah ada tren penjualan yang konsisten dari waktu ke waktu atau tren bulanan yang nyata.

In [36]:
```
# Group by month and calculate total sales for each month
monthly_sales = data.groupby(data['Invoice Date'].dt.to_period('M'))['Total

# Convert the index (which is of type 'Period') to DateTime objects
monthly_sales.index = monthly_sales.index.to_timestamp()

# Setting the style for a more aesthetic plot
sns.set(style="whitegrid")

# Plotting the sales trends over time
plt.figure(figsize=(12, 7))
plt.plot(monthly_sales.index, monthly_sales.values, marker='o', color='teal'
plt.xlabel('Month', fontsize=12)
plt.ylabel('Total Sales', fontsize=12)
plt.title('Monthly Sales Trends', fontsize=14)
plt.xticks(rotation=0)
plt.tick_params(axis='both', which='major', labelsize=10)
plt.tight_layout()
plt.show()
```



In [37]:
```
# Calculate the difference between consecutive months for trend analysis
diff = monthly_sales.diff()

# Define the threshold for significant change (seasonality)
seasonality_threshold = 10000

# Check for seasonality
if (diff.abs() > seasonality_threshold).any():
    print("These sales follow a seasonality.")
else:
    print("These sales do not follow a seasonality.")

# Check for trend
if (diff > 0).all():
    print("There is an increasing trend in sales.")
elif (diff < 0).all():
    print("There is a decreasing trend in sales.")
else:
    print("There is no consistent trend in sales.")
```

```
These sales follow a seasonality.
There is no consistent trend in sales.
```

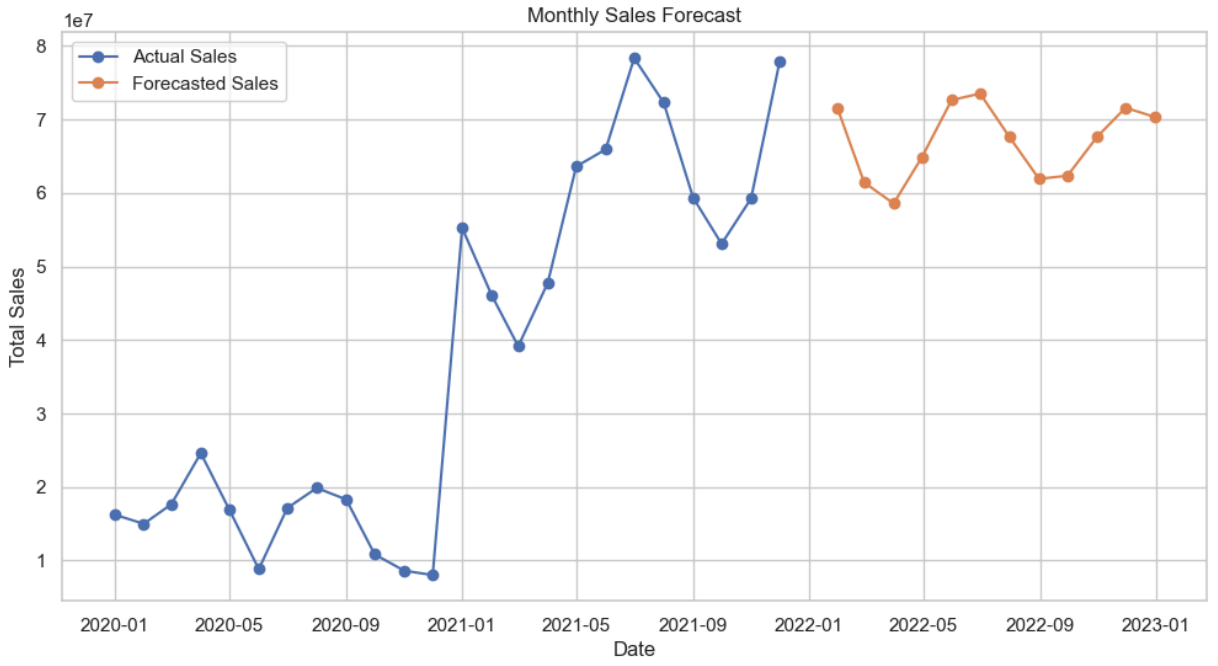
9. Analisis Penjualan Prediktif: Kembangkan perkiraan penjualan bulanan.

In [38]:
```
# add library requirement

import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
```

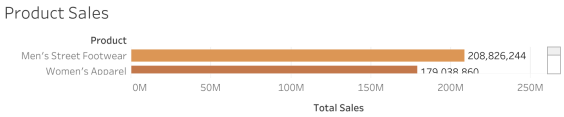
```
from statsmodels.tsa.arima.model import ARIMA
import plotly.graph_objs as go
```

In [39]:

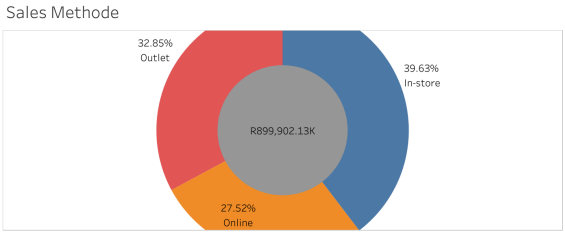
```
# Fit the ARIMA model to the data
model = ARIMA(monthly_sales, order=(2,1,2))  # Order: (p, d, q)
model_fit = model.fit()

# Make predictions for the next 12 months
forecast_steps = 12
forecast = model_fit.forecast(steps=forecast_steps)

#Visualize the actual sales data and the forecasted values for the next mont
plt.figure(figsize=(12, 6))
plt.plot(monthly_sales.index, monthly_sales.values, marker='o', label='Actua
plt.plot(pd.date_range(start=monthly_sales.index[-1], periods=forecast_steps
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.title('Monthly Sales Forecast')
plt.legend()
plt.xticks(rotation=0)
plt.grid(True)
plt.show()
```

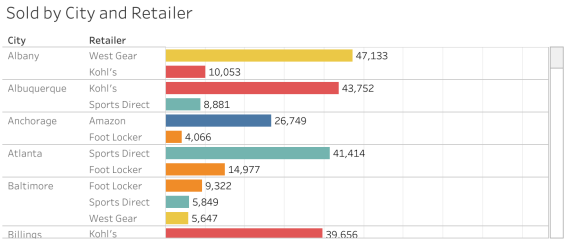


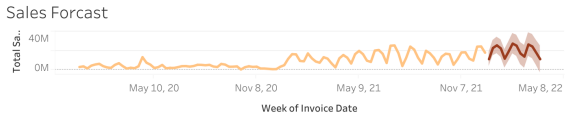
Dashboard

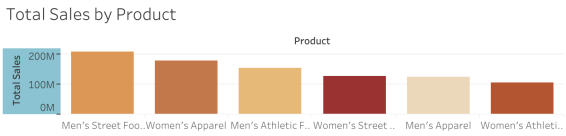

Product Sales
Product
Men's Street Footwear
Women's Apparel
208,826,244
0M
50M
100M
150M
200M
250M
Total Sales


Monthly Trend of Sales
Invoice Date
Total Sales
100M
50M
0M
71,479,142
61,100,153
72,339,970
56,809,109
80,507,695
74,747,372
85,480,694
77,661,459
82,166,201
63,911,033
67,857,340
January
February
March
April
May
June
July
August
Septem..
October
Novem..
Decem..


Sales Methode
32.85%
Outlet
39.63%
In-store
R899,902.13K
27.52%
Online


Sold by City and Retailer
City
Retailer
Albany
West Gear
47,133
Kohl's
10,053
Albuquerque
Kohl's
43,752
Sports Direct
8,881
Anchorage
Amazon
26,749
Foot Locker
4,066
Atlanta
Sports Direct
41,414
Foot Locker
14,977
Baltimore
Foot Locker
9,322
Sports Direct
5,849
West Gear
5,647
Billings
Kohl's
39,656


Sales Forcast
Total Sa...
40M
0M
May 10, 20
Nov 8, 20
May 9, 21
Nov 7, 21
May 8, 22
Week of Invoice Date


Total Sales by Product
Product
Total Sales
200M
100M
0M
Men's Street Foo..
Women's Apparel
Men's Athletic F..
Women's Street ..
Men's Apparel
Women's Athleti..

In [ ]: