```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


import plotly.express as px



from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Adidas US Sales Datasets_1.csv')
```

```
df.head(10)
```

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnamed: 11	Unnam
0	NaN	Adidas Sales Database	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I
3	Retailer	Retailer ID	Invoice Date	Region	State	City	Product	Price per Unit	Units Sold	Total Sales	Operating Profit	Operating Margin	S Met
4	Foot Locker	1185732	01/01/2020	Northeast	New York	New York	Men's Street Footwear	$50.00	1,200	$600,000	$300,000	50%	In-s
5	Foot Locker	1185732	02/01/2020	Northeast	New York	New York	Men's Athletic Footwear	$50.00	1,000	$500,000	$150,000	30%	In-s
6	Foot Locker	1185732	03/01/2020	Northeast	New York	New York	Women's Street Footwear	$40.00	1,000	$400,000	$140,000	35%	In-s

```
df=df.drop(index=[0,1,2])
df=pd.DataFrame(df.values)
new_header=df.iloc[0].tolist()

df=df[1:]

df.columns=new_header


df.head()
```

	Retailer	Retailer ID	Invoice Date	Region	State	City	Product	Price per Unit	Units Sold	Total Sales	Operating Profit	Operating Margin	Sales Method
1	Foot Locker	1185732	01/01/2020	Northeast	New York	New York	Men's Street Footwear	$50.00	1,200	$600,000	$300,000	50%	In-store
2	Foot Locker	1185732	02/01/2020	Northeast	New York	New York	Men's Athletic Footwear	$50.00	1,000	$500,000	$150,000	30%	In-store
3	Foot Locker	1185732	03/01/2020	Northeast	New York	New York	Women's Street Footwear	$40.00	1,000	$400,000	$140,000	35%	In-store
4	Foot Locker	1185732	04/01/2020	Northeast	New York	New York	Women's Athletic	$45.00	850	$382.500	$133.875	35%	In-store

```
for col in df.columns:
  print(f"{col}:{df[col].unique()}")
```

```
Retailer:['Foot Locker' 'Walmart' 'Sports Direct' 'West Gear' "Kohl's" 'Amazon']
Retailer ID:['1185732' '1197831' '1128299' '1189833']
Invoice Date:['01/01/2020' '02/01/2020' '03/01/2020' '04/01/2020' '05/01/2020'
 '06/01/2020' '07/01/2020' '08/01/2020' '21/01/2020' '22/01/2020'
 '23/01/2020' '24/01/2020' '25/01/2020' '26/01/2020' '27/01/2020'
 '28/01/2020' '29/01/2020' '30/01/2020' '31/01/2020' '01/02/2020'
 '02/02/2020' '03/02/2020' '04/02/2020' '05/02/2020' '06/02/2020'
 '07/02/2020' '08/02/2020' '09/02/2020' '10/02/2020' '03/03/2020'
 '04/03/2020' '05/03/2020' '06/03/2020' '07/03/2020' '08/03/2020'
 '09/03/2020' '10/03/2020' '11/03/2020' '12/03/2020' '13/03/2020'
 '14/03/2020' '15/03/2020' '16/03/2020' '17/03/2020' '18/03/2020'
 '31/03/2020' '17/04/2020' '18/04/2020' '19/04/2020' '20/04/2020'
 '21/04/2020' '22/04/2020' '23/04/2020' '24/04/2020' '25/04/2020'
 '26/04/2020' '27/04/2020' '28/04/2020' '29/04/2020' '30/04/2020'
 '01/05/2020' '02/05/2020' '03/05/2020' '04/05/2020' '05/05/2020'
 '06/05/2020' '07/05/2020' '18/07/2020' '19/07/2020' '20/07/2020'
 '21/07/2020' '22/07/2020' '23/07/2020' '24/07/2020' '25/07/2020'
 '26/07/2020' '27/07/2020' '28/07/2020' '29/07/2020' '30/07/2020'
 '31/07/2020' '01/08/2020' '02/08/2020' '03/08/2020' '04/08/2020'
 '05/08/2020' '06/08/2020' '07/08/2020' '08/08/2020' '09/08/2020'
 '10/08/2020' '11/08/2020' '12/08/2020' '13/08/2020' '14/08/2020'
 '15/08/2020' '16/08/2020' '17/08/2020' '18/08/2020' '19/08/2020'
 '20/08/2020' '21/08/2020' '22/08/2020' '23/08/2020' '24/08/2020'
 '25/08/2020' '26/08/2020' '27/08/2020' '28/08/2020' '29/08/2020'
 '30/08/2020' '31/08/2020' '01/09/2020' '02/09/2020' '03/09/2020'
 '04/09/2020' '05/09/2020' '06/09/2020' '07/09/2020' '08/09/2020'
 '09/09/2020' '10/09/2020' '11/09/2020' '12/09/2020' '13/09/2020'
 '14/09/2020' '15/09/2020' '16/09/2020' '20/10/2020' '21/10/2020'
 '22/10/2020' '23/10/2020' '24/10/2020' '25/10/2020' '26/10/2020'
 '27/10/2020' '28/10/2020' '29/10/2020' '30/10/2020' '31/10/2020'
 '01/11/2020' '02/11/2020' '03/11/2020' '04/11/2020' '05/11/2020'
```

```
'06/11/2020' '07/11/2020' '08/11/2020' '09/11/2020' '10/11/2020'
'11/11/2020' '12/11/2020' '13/11/2020' '14/11/2020' '15/11/2020'
'22/12/2020' '23/12/2020' '24/12/2020' '25/12/2020' '26/12/2020'
'27/12/2020' '28/12/2020' '29/12/2020' '30/12/2020' '31/12/2020'
'01/01/2021' '02/01/2021' '03/01/2021' '04/01/2021' '05/01/2021'
'06/01/2021' '07/01/2021' '08/01/2021' '09/01/2021' '10/01/2021'
'11/01/2021' '12/01/2021' '13/01/2021' '14/01/2021' '15/01/2021'
'16/01/2021' '17/01/2021' '18/01/2021' '19/01/2021' '20/01/2021'
'21/01/2021' '22/01/2021' '23/01/2021' '24/01/2021' '25/01/2021'
'26/01/2021' '27/01/2021' '28/01/2021' '29/01/2021' '30/01/2021'
'31/01/2021' '01/02/2021' '02/02/2021' '03/02/2021' '04/02/2021'
'05/02/2021' '06/02/2021' '07/02/2021' '08/02/2021' '09/02/2021'
'10/02/2021' '11/02/2021' '12/02/2021' '13/02/2021' '14/02/2021'
'15/02/2021' '16/02/2021' '17/02/2021' '18/02/2021' '09/03/2021'
'10/03/2021' '11/03/2021' '12/03/2021' '13/03/2021' '14/03/2021'
'15/03/2021' '16/03/2021' '17/03/2021' '18/03/2021' '19/03/2021'
'20/03/2021' '21/03/2021' '22/03/2021' '23/03/2021' '06/04/2021'
'07/04/2021' '08/04/2021' '09/04/2021' '10/04/2021' '11/04/2021'
'12/04/2021' '13/04/2021' '14/04/2021' '15/04/2021' '16/04/2021'
'17/04/2021' '18/04/2021' '19/04/2021' '20/04/2021' '21/04/2021'
'22/04/2021' '23/04/2021' '24/04/2021' '25/04/2021' '26/04/2021'
'27/04/2021' '28/04/2021' '29/04/2021' '30/04/2021' '01/05/2021'
'02/05/2021' '03/05/2021' '04/05/2021' '05/05/2021' '06/05/2021'
'07/05/2021' '08/05/2021' '09/05/2021' '10/05/2021' '11/05/2021'
'12/05/2021' '13/05/2021' '14/05/2021' '15/05/2021' '16/05/2021'
'17/05/2021' '18/05/2021' '19/05/2021' '20/05/2021' '21/05/2021'
'22/05/2021' '23/05/2021' '24/05/2021' '25/05/2021' '26/05/2021'
```

```
df.isna().sum()
```

	0
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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9648 entries, 1 to 9648
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Retailer          9648 non-null   object
 1   Retailer ID       9648 non-null   object
 2   Invoice Date      9648 non-null   object
 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   object
 8   Units Sold        9648 non-null   object
 9   Total Sales       9648 non-null   object
 10  Operating Profit  9648 non-null   object
 11  Operating Margin  9648 non-null   object
 12  Sales Method      9648 non-null   object
dtypes: object(13)
memory usage: 980.0+ KB
```

```
columns_to_clean=['Price per Unit','Total Sales','Operating Profit']
for cols in columns_to_clean:
  df[cols]=df[cols].str.replace('$','', regex=False)

df['Operating Margin']=df['Operating Margin'].str.replace('%','', regex=False)



cols_with_comma=['Units Sold','Total Sales','Operating Profit']
for col in cols_with_comma:
  df[col]=df[col].str.replace(',','', regex=False)


columns_to_convert=['Price per Unit','Units Sold','Total Sales','Operating Profit','Operating Margin']
df[columns_to_convert]=df[columns_to_convert].apply(pd.to_numeric, errors='coerce')


df['Invoice Date']=pd.to_datetime(df['Invoice Date'], format='%d/%m/%Y', errors='coerce')


df.head()
```

	Retailer	Retailer ID	Invoice Date	Region	State	City	Product	Price per Unit	Units Sold	Total Sales	Operating Profit	Operating Margin	Sales Method
1	Foot Locker	1185732	2020-01-01	Northeast	New York	New York	Men's Street Footwear	50.0	1200	600000	300000	50	In-store
2	Foot Locker	1185732	2020-01-02	Northeast	New York	New York	Men's Athletic Footwear	50.0	1000	500000	150000	30	In-store
3	Foot Locker	1185732	2020-01-03	Northeast	New York	New York	Women's Street Footwear	40.0	1000	400000	140000	35	In-store
4	Foot	1185732	2020-01-	Northeast	New	New	Women's Athletic	45.0	850	382500	133875	35	In-store

Start coding or generate with AI.

```
df['Month']=df['Invoice Date'].dt.month
df['Year']=df['Invoice Date'].dt.year
df['Quarter']=df['Invoice Date'].dt.quarter
```

```
df.head()
```

	Retailer	Retailer ID	Invoice Date	Region	State	City	Product	Price per Unit	Units Sold	Total Sales	Operating Profit	Operating Margin	Sales Method	Month	Year
1	Foot Locker	1185732	2020-01-01	Northeast	New York	New York	Men's Street Footwear	50.0	1200	600000	300000	50	In-store	1	2020
2	Foot Locker	1185732	2020-01-02	Northeast	New York	New York	Men's Athletic Footwear	50.0	1000	500000	150000	30	In-store	1	2020
3	Foot Locker	1185732	2020-01-03	Northeast	New York	New York	Women's Street Footwear	40.0	1000	400000	140000	35	In-store	1	2020
							Women's								

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9648 entries, 1 to 9648
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Retailer          9648 non-null   object
 1   Retailer ID       9648 non-null   object
 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   int64
 10  Operating Profit  9648 non-null   int64
 11  Operating Margin  9648 non-null   int64
 12  Sales Method      9648 non-null   object
 13  Month             9648 non-null   int32
 14  Year              9648 non-null   int32
 15  Quarter           9648 non-null   int32
dtypes: datetime64[ns](1), float64(1), int32(3), int64(4), object(7)
memory usage: 1.1+ MB
```

```
df.describe()
```

	Invoice Date	Price per Unit	Units Sold	Total Sales	Operating Profit	Operating Margin	Month	Year	Quarter
count	9648	9648.000000	9648.000000	9648.000000	9648.000000	9648.000000	9648.000000	9648.000000	9648.00000(
mean	2021-05-10 15:20:44.776119296	45.216625	256.930037	93273.437500	34425.282131	42.299129	6.458126	2020.865050	2.49212:
min	2020-01-01 00:00:00	7.000000	0.000000	0.000000	0.000000	10.000000	1.000000	2020.000000	1.00000(
25%	2021-02-17 00:00:00	35.000000	106.000000	4254.500000	1922.000000	35.000000	3.000000	2021.000000	1.00000(
50%	2021-06-04 00:00:00	45.000000	176.000000	9576.000000	4371.500000	41.000000	6.000000	2021.000000	2.00000(

```
df['Retailer'].unique()
```

```
array(['Foot Locker', 'Walmart', 'Sports Direct', 'West Gear', "Kohl's",
       'Amazon'], dtype=object)
```

```
df['Product'].unique()
```

```
array(["Men's Street Footwear", "Men's Athletic Footwear",
       "Women's Street Footwear", "Women's Athletic Footwear",
       "Men's Apparel", "Women's Apparel"], dtype=object)
```

```
top_performing_retailers=df.groupby('Retailer')['Total Sales'].sum().sort_values(ascending=False)
top_performing_retailers
```

Retailer	Total Sales
West Gear	242964333
Foot Locker	220094720
Sports Direct	182470997
Kohl's	102114753
Amazon	77698912
Walmart	74558410

dtype: int64

```
top_performing_products=df.groupby('Product')['Total Sales'].sum().sort_values(ascending=False)
top_performing_products
```

Product	Total Sales
Men's Street Footwear	208826244
Women's Apparel	179038860
Men's Athletic Footwear	153673680
Women's Street Footwear	128002813
Men's Apparel	123728632
Women's Athletic Footwear	106631896

dtype: int64

```
top_performing_region=df.groupby('Region')['Total Sales'].sum().sort_values(ascending=False)
top_performing_region
```

Region	Total Sales
West	269943182
Northeast	186324067
Southeast	163171236
South	144663181
Midwest	135800459

dtype: int64

```
top_performing_state=df.groupby('State')['Total Sales'].sum().sort_values(ascending=False)
top_performing_state
```

State	Total Sales
New York	64229039
California	60174133
Florida	59283714
Texas	46359746
South Carolina	29285637
Washington	26330718
North Carolina	23956531
Louisiana	23750781
Hawaii	22282457
Virginia	21575040
Oregon	21349674
Colorado	20996536
Nevada	20858509
New Mexico	19865016
Idaho	19276878
Georgia	18997466
Michigan	18625433
Wyoming	18577517
Ohio	18484583
Tennessee	18067440
Alabama	17633424
New Hampshire	16411667
Arizona	15782221
Montana	15710886
Mississippi	15591709
Alaska	14753103
Vermont	14352923
Arkansas	12639347
Delaware	12298412
Connecticut	11573448

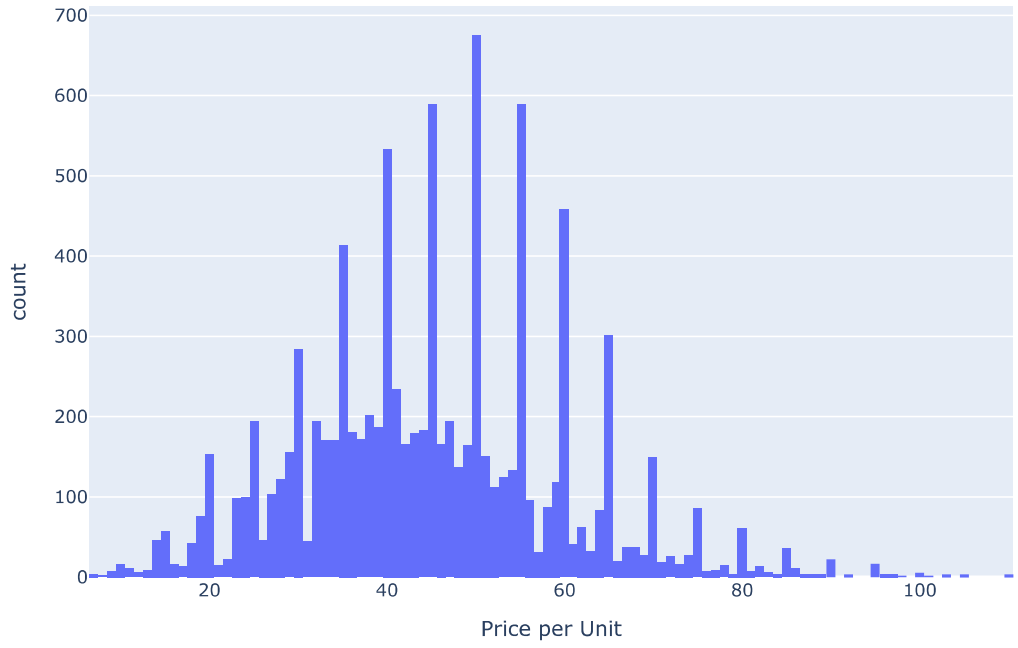
```
top_performing_city=df.groupby('City')['Total Sales'].sum().sort_values(ascending=False)
top_performing_city
```

City	Total Sales
Charleston	39974797
New York	39801235
San Francisco	34539220
Miami	31600863
Portland	30545652
Orlando	27682851
Seattle	26330718
Los Angeles	25634913
Houston	25456882
Albany	24427804
Charlotte	23956531
New Orleans	23750781
Honolulu	22282457
Richmond	21575040
Denver	20996536
Dallas	20902864
Las Vegas	20858509
Albuquerque	19865016
Boise	19276878
Atlanta	18997466
Detroit	18625433
Cheyenne	18577517
Columbus	18484583
Knoxville	18067440
Birmingham	17633424
Manchester	16411667
Phoenix	15782221
Billings	15710886
Jackson	15591709
Anchorage	14753103

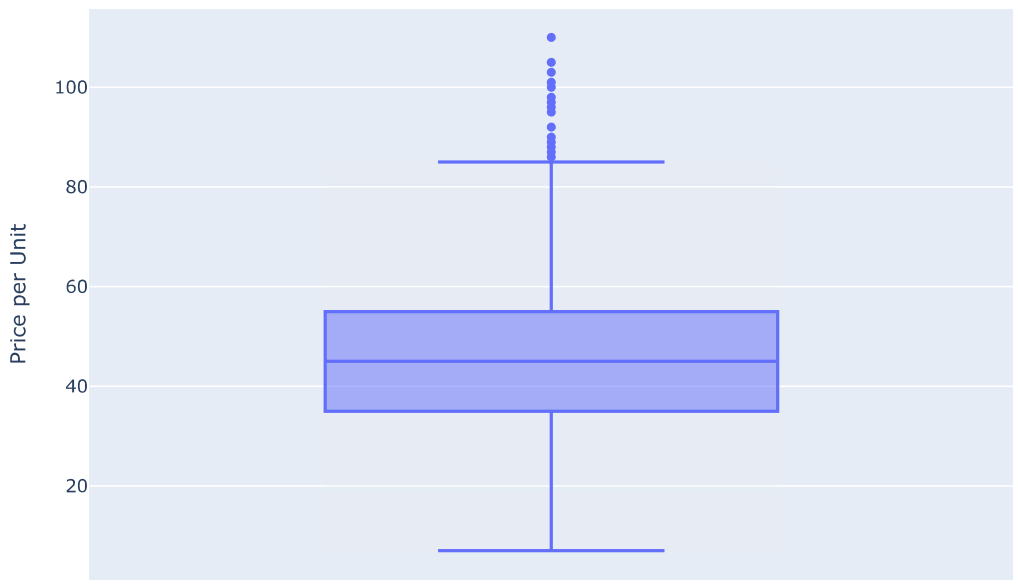
```
top_quarters=df.groupby(['Year','Quarter'])['Total Sales'].sum().sort_values(ascending=False)
top_quarters.reset_index()
tp=top_quarters.to_frame().reset_index()
tp
```

	Year	Quarter	Total Sales
0	2021	3	209979925
1	2021	4	190125234
2	2021	2	177240198
3	2021	1	140476093
4	2020	3	55328429
5	2020	2	50354839
6	2020	1	48912311
7	2020	4	27485096

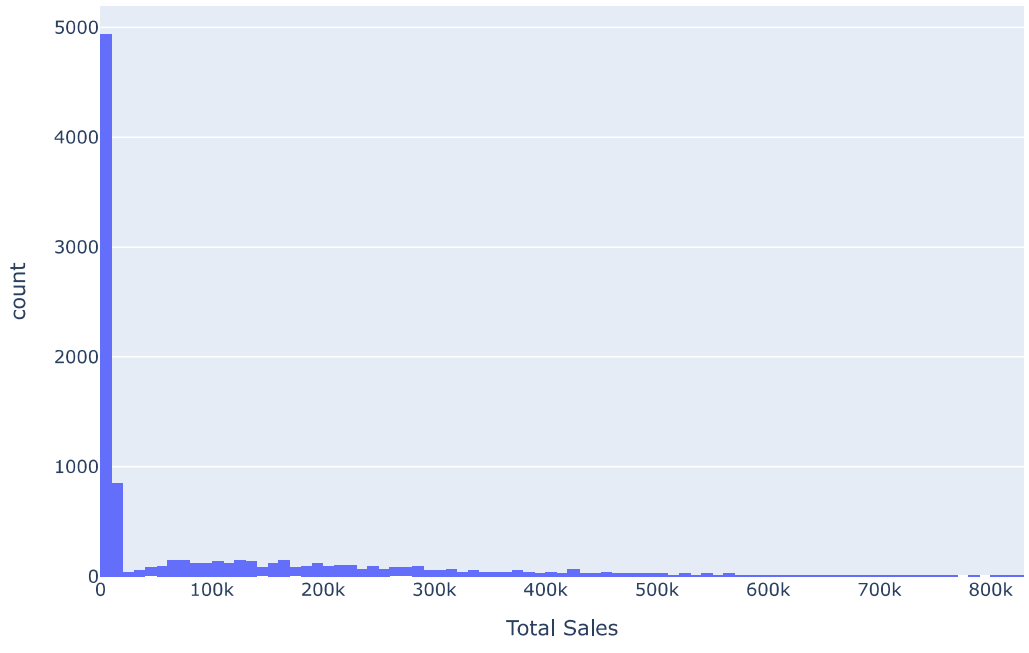
```
fig=px.histogram(df, x='Price per Unit')
fig.show()
```



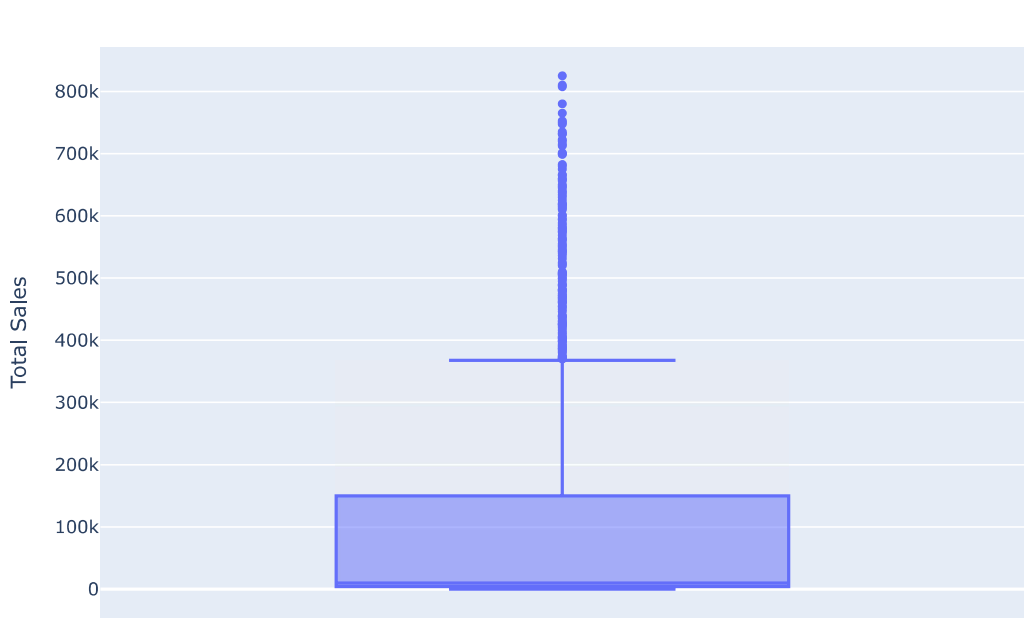

```
fig=px.box(df, y='Price per Unit')
fig.show()
```



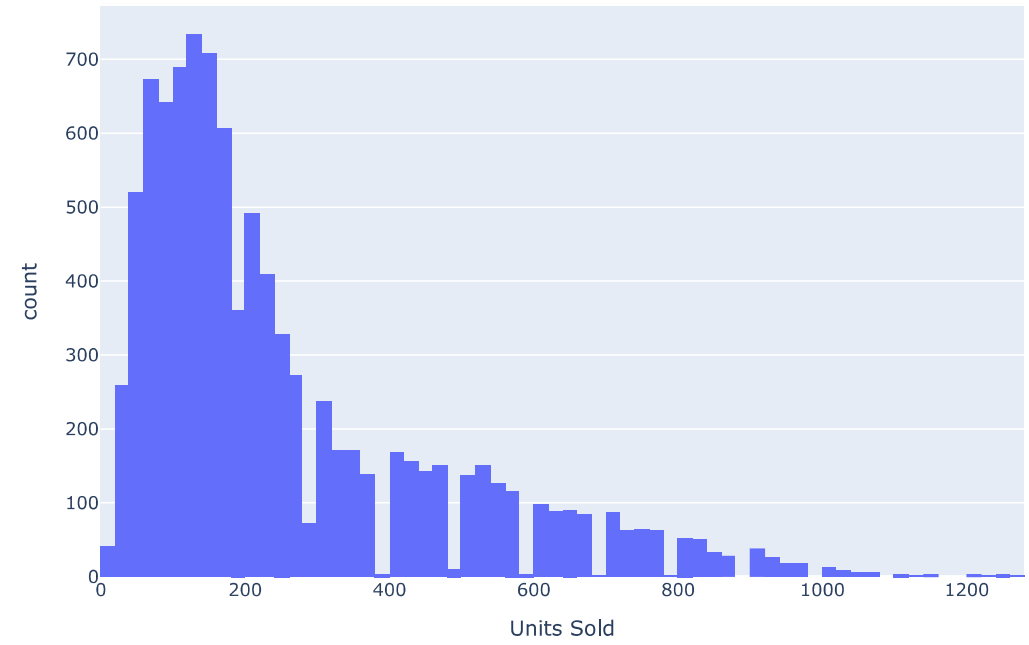

```
fig=px.histogram(df, x='Total Sales')
fig.show()
```



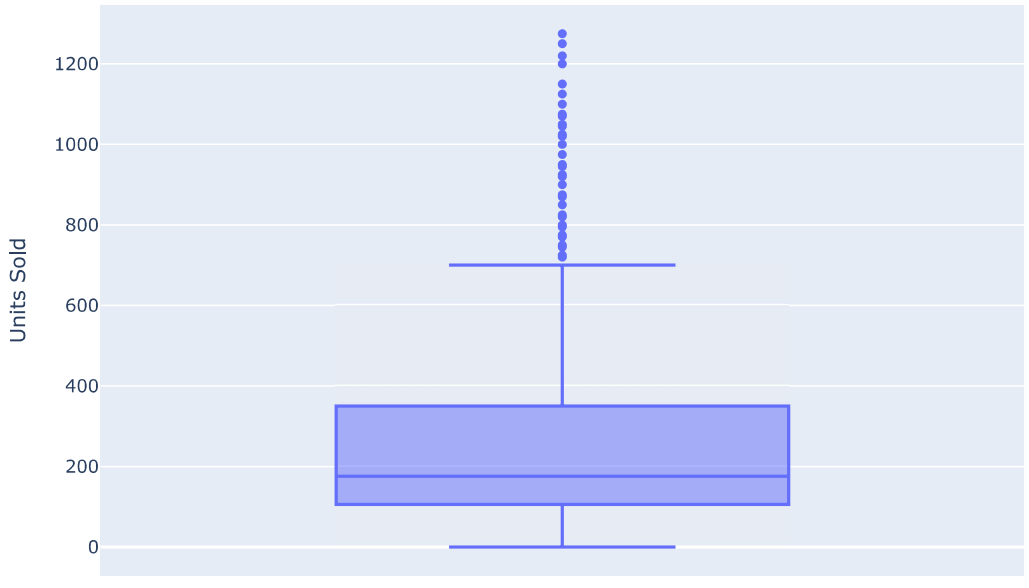

```
fig=px.box(df, y='Total Sales')
fig.show()
```



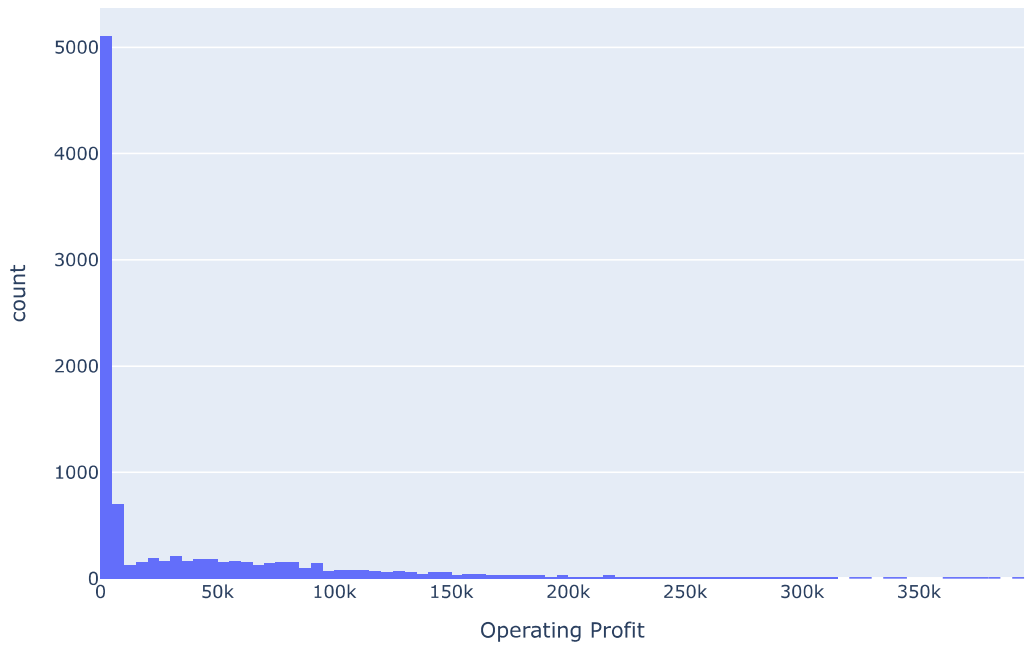

```
fig=px.histogram(df, x='Units Sold')
fig.show()
```



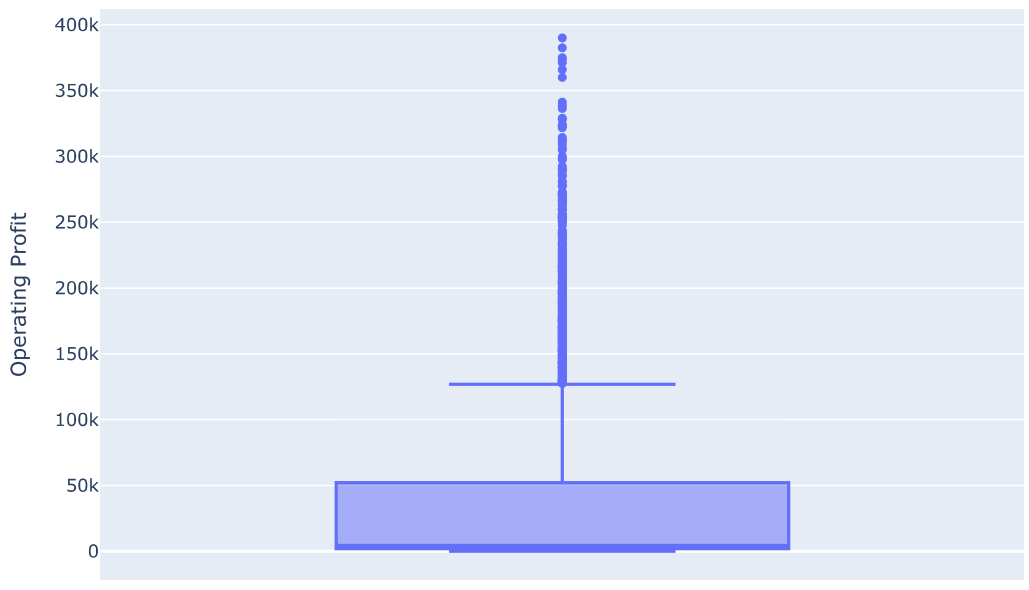

```
fig=px.box(df, y='Units Sold')
fig.show()
```



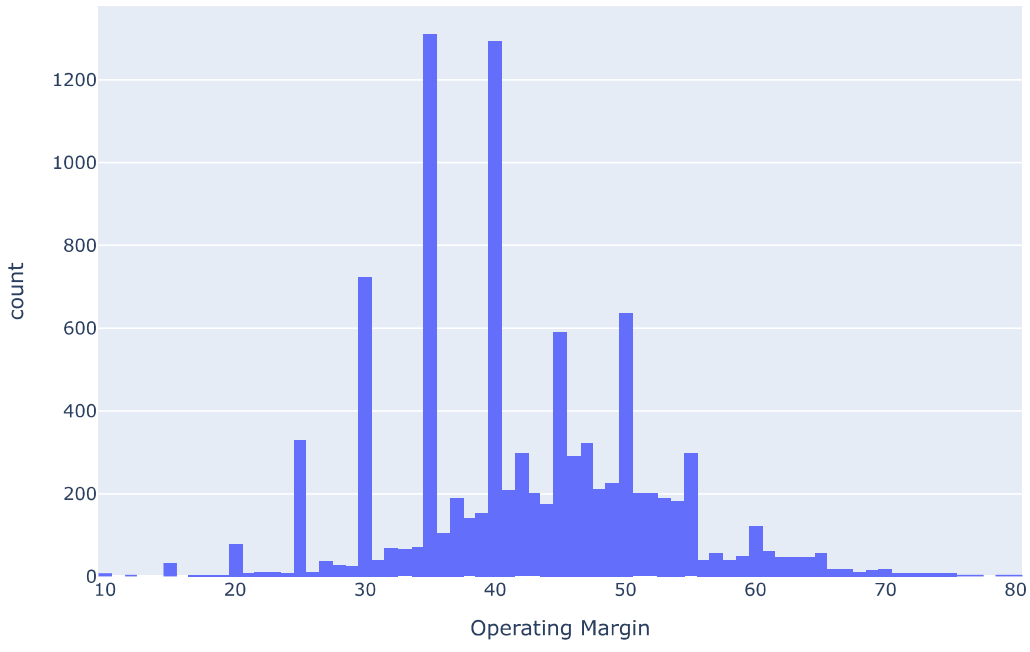

```
fig=px.histogram(df, x='Operating Profit')
fig.show()
```



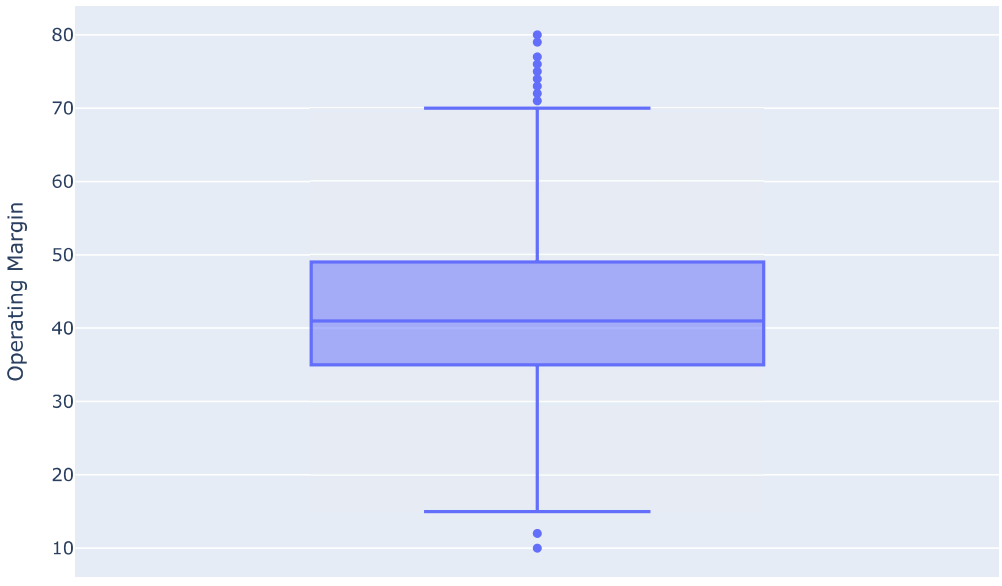

```
fig=px.box(df, y='Operating Profit')
fig.show()
```




```
fig=px.histogram(df, x='Operating Margin')
fig.show()
```




```
fig=px.box(df, y='Operating Margin')
fig.show()
```

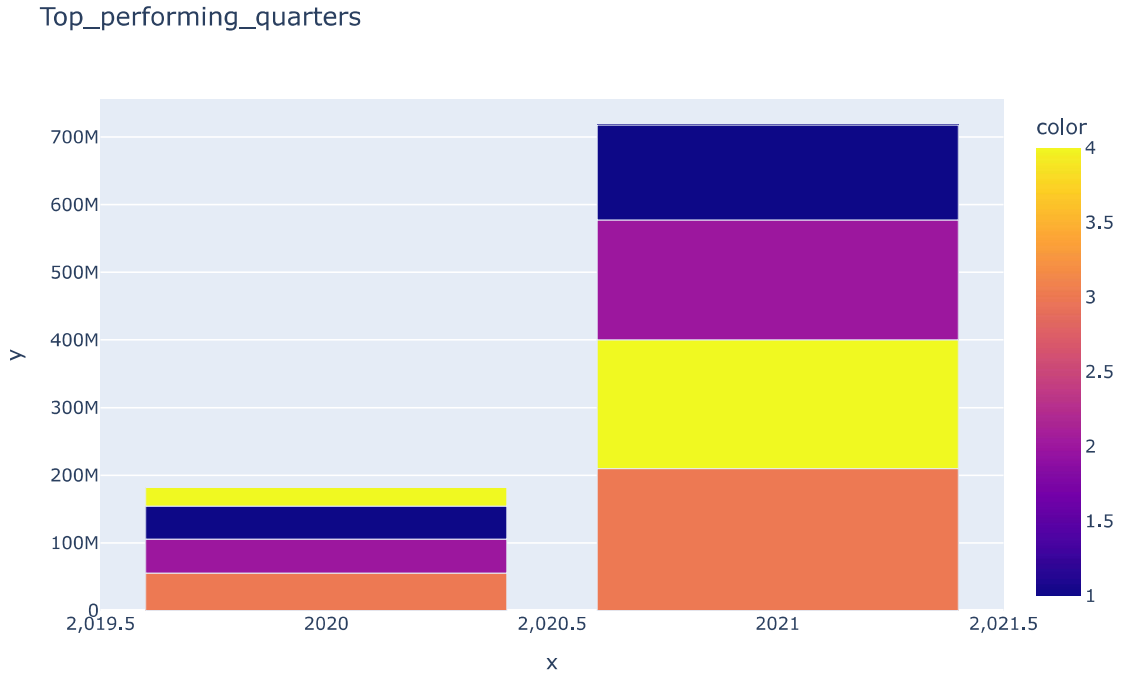



```
df.head()
```

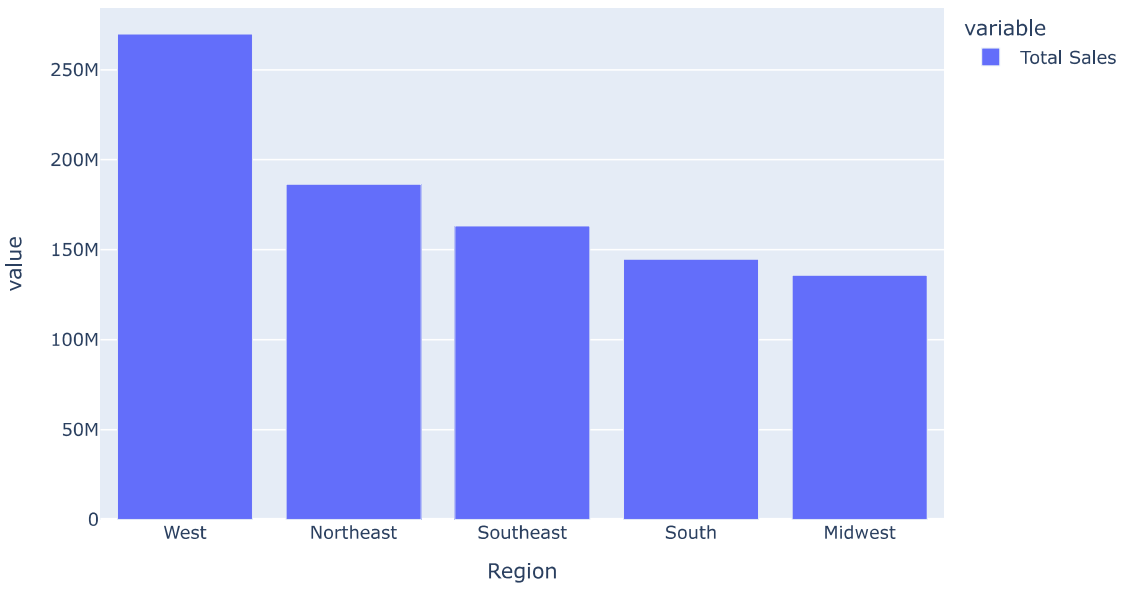
	Retailer	Retailer ID	Invoice Date	Region	State	City	Product	Price per Unit	Units Sold	Total Sales	Operating Profit	Operating Margin	Sales Method	Month	Year
1	Foot Locker	1185732	2020-01-01	Northeast	New York	New York	Men's Street Footwear	50.0	1200	600000	300000	50	In-store	1	2020
2	Foot Locker	1185732	2020-01-02	Northeast	New York	New York	Men's Athletic Footwear	50.0	1000	500000	150000	30	In-store	1	2020
3	Foot Locker	1185732	2020-01-03	Northeast	New York	New York	Women's Street Footwear	40.0	1000	400000	140000	35	In-store	1	2020

Data Visualizations

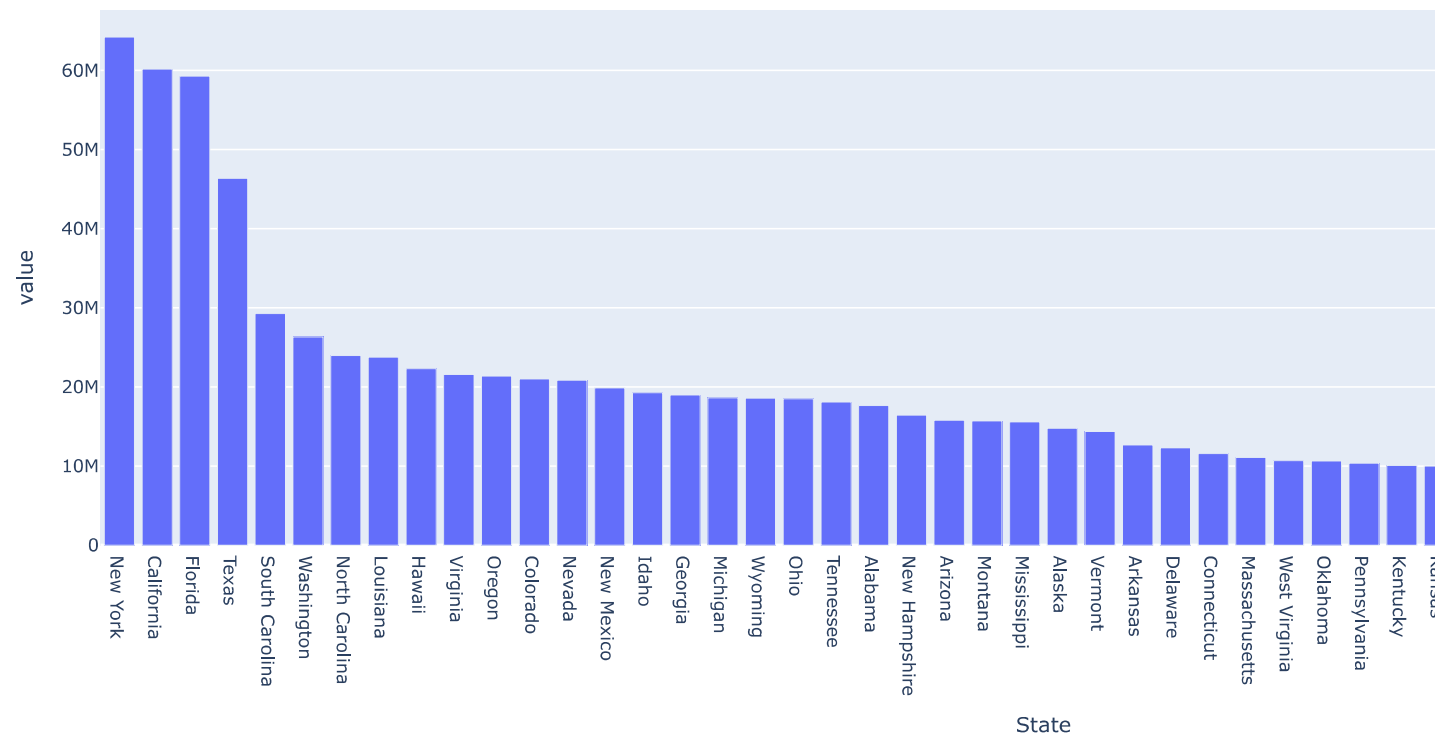
```
fig=px.bar(top_quarters, x=tp['Year'], y=tp['Total Sales'],color=tp['Quarter'], title='Top_performing_quarters')
fig.show()
```



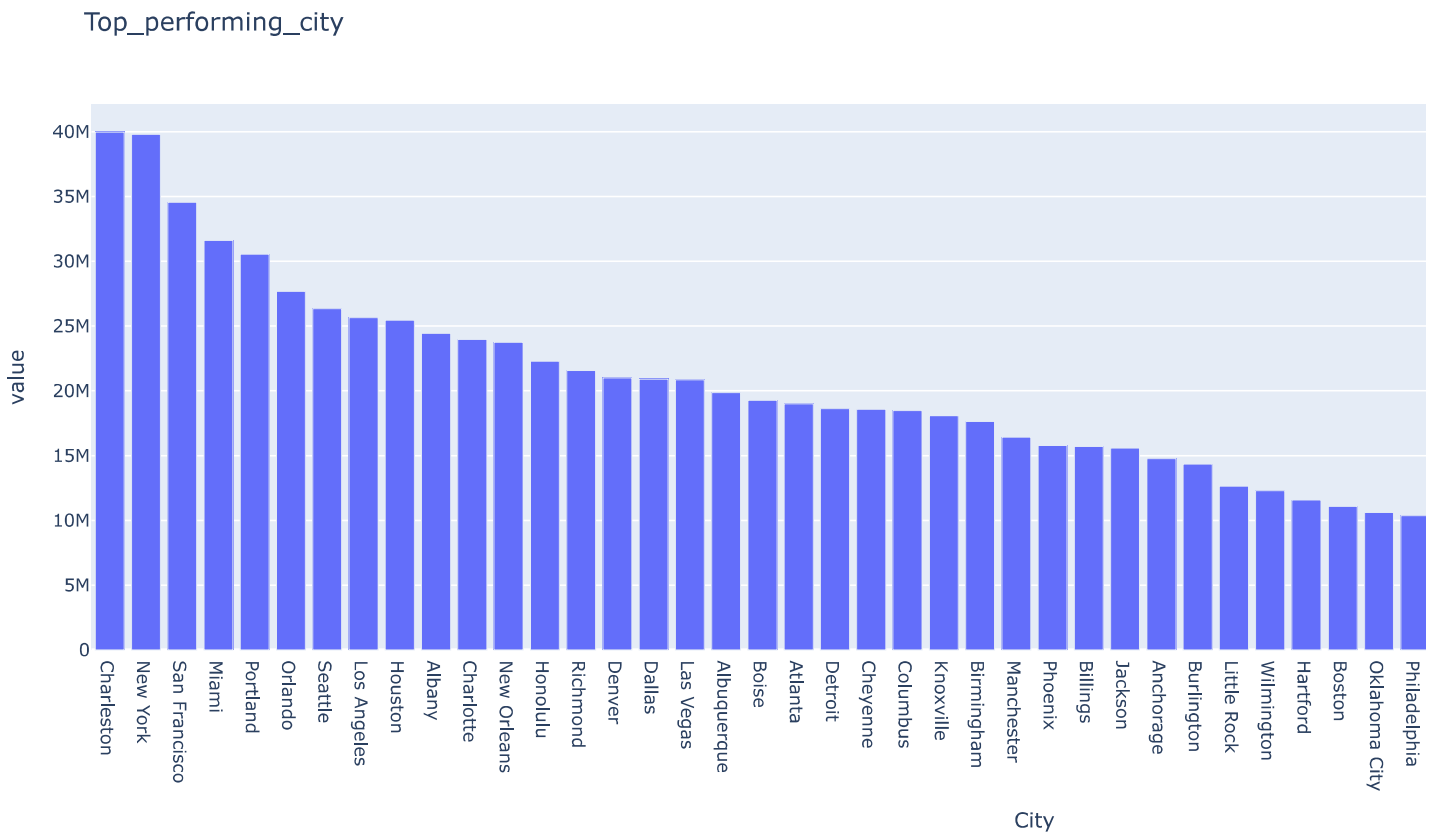

```
px.bar(top_performing_region, title='Top Performing Region')
```



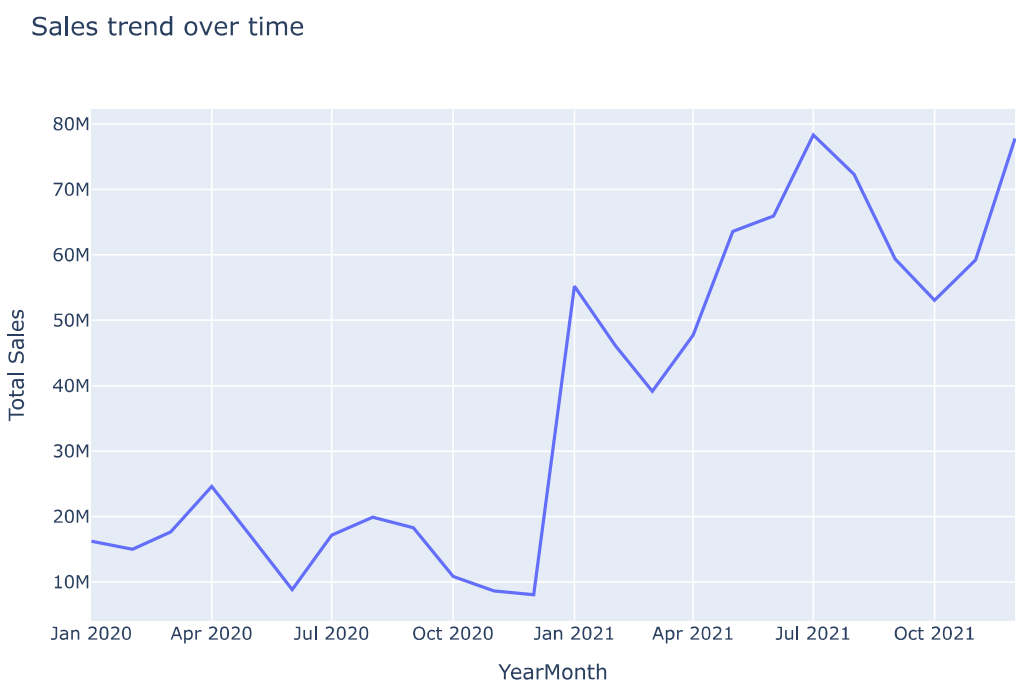

```
fig=px.bar(top_performing_state, title='Top_performing_state')
fig.update_layout(
    autosize=False,
    width=1500,
    height=600
)
fig.show()
```



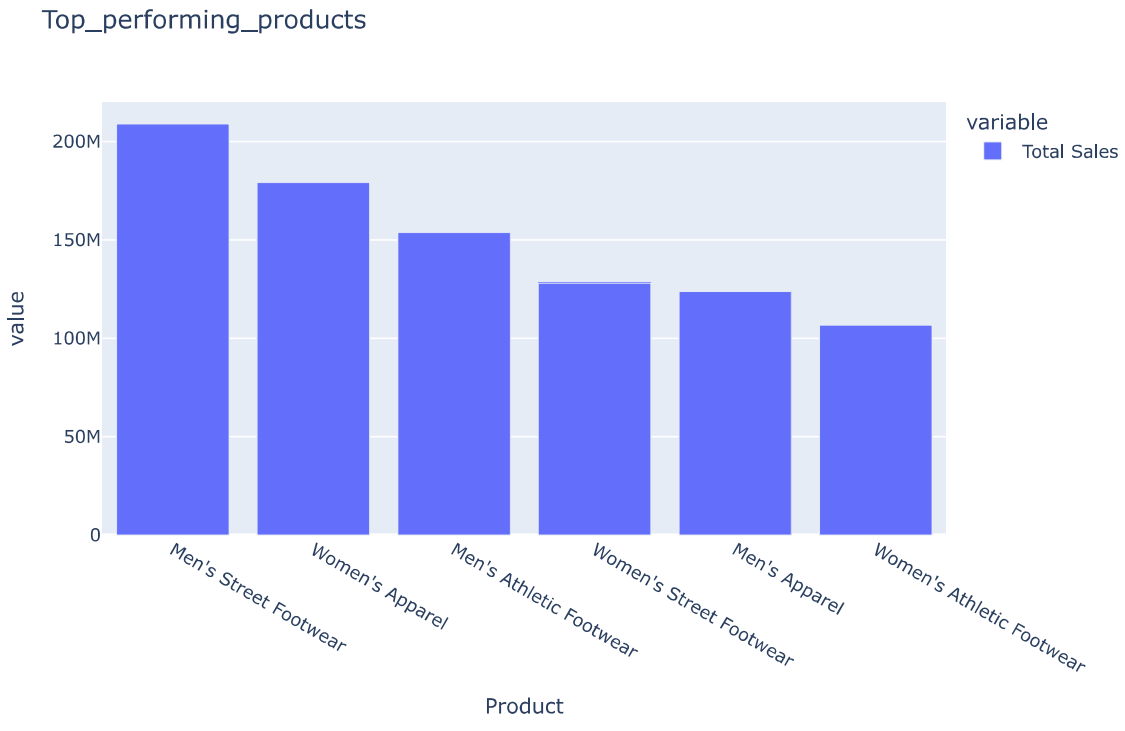
```
fig=px.bar(top_performing_city, title='Top_performing_city')
fig.update_layout(
    autosize=False,
    width=1500,
    height=600
)
fig.show()
```



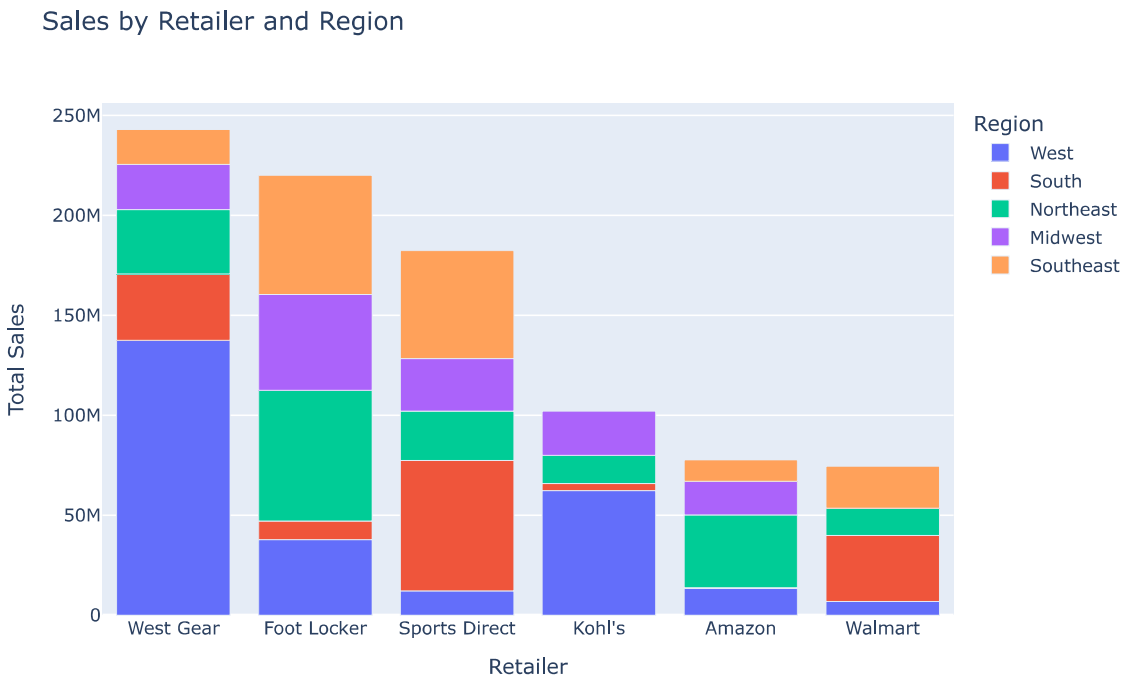

```
df['YearMonth']=df['Invoice Date'].dt.to_period('M')
sales_by_month=df.groupby('YearMonth')['Total Sales'].sum().reset_index()

sales_by_month['YearMonth']=sales_by_month['YearMonth'].astype(str)

fig=px.line(sales_by_month, x='YearMonth', y='Total Sales', title='Sales trend over time')
fig.show()
```



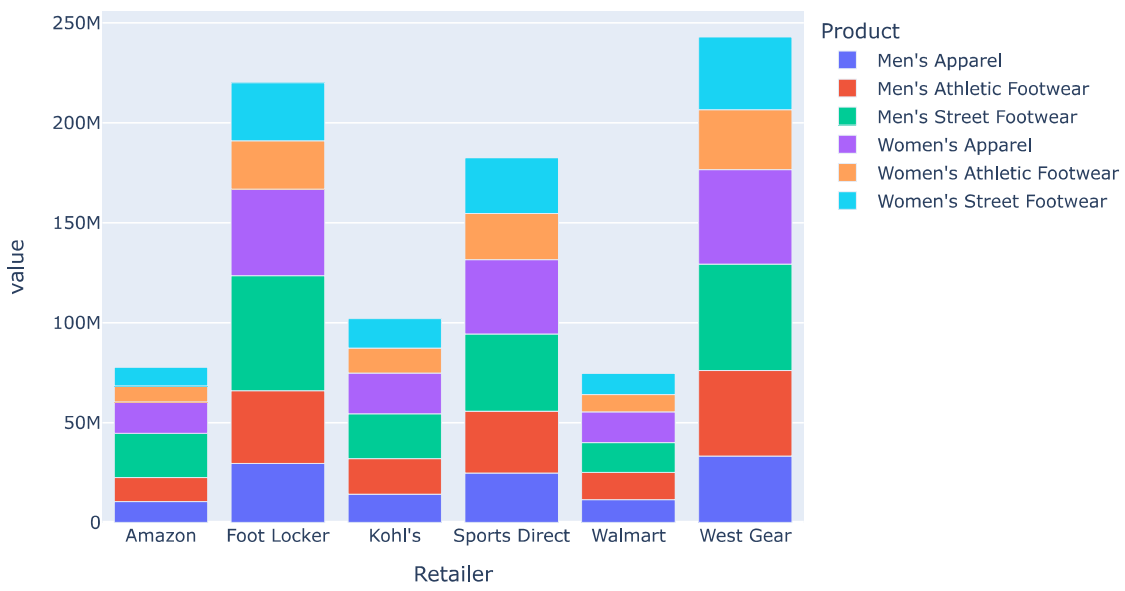

```
px.bar(top_performing_products, title='Top_performing_products')
```



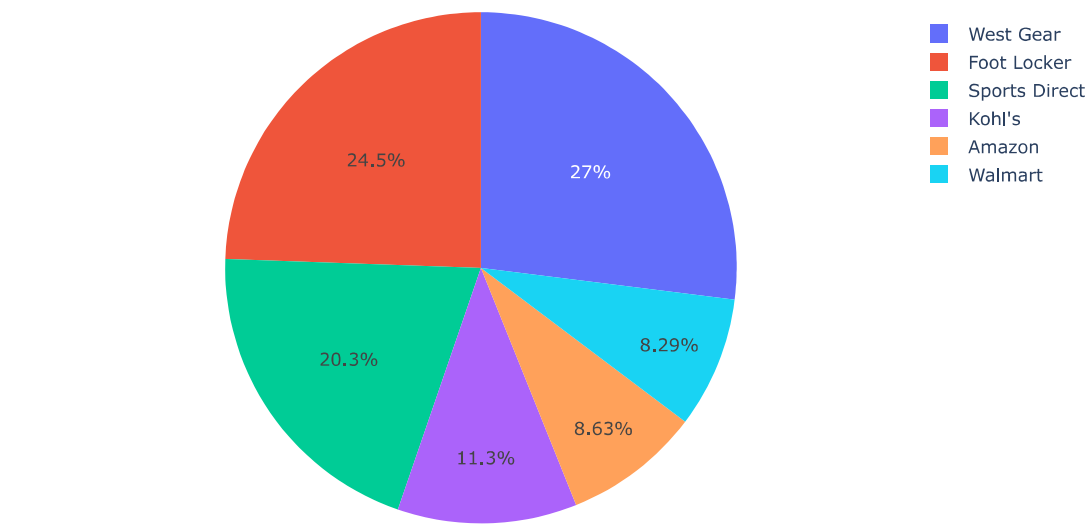

```
sales_by_retailer_region=df.groupby(['Retailer','Region'])['Total Sales'].sum().sort_values(ascending=False).reset_index()
sales_by_retailer_region=sales_by_retailer_region.sort_values(['Retailer','Total Sales'], ascending=[False,False])
sorted_retailers=sales_by_retailer_region.groupby('Retailer')['Total Sales'].sum().sort_values(ascending=False).index.tolist()
fig=px.bar(sales_by_retailer_region, x='Retailer', y='Total Sales', color='Region', title='Sales by Retailer and Region', category_orders={'Retailer': sorted_retailers})
fig.show()
```



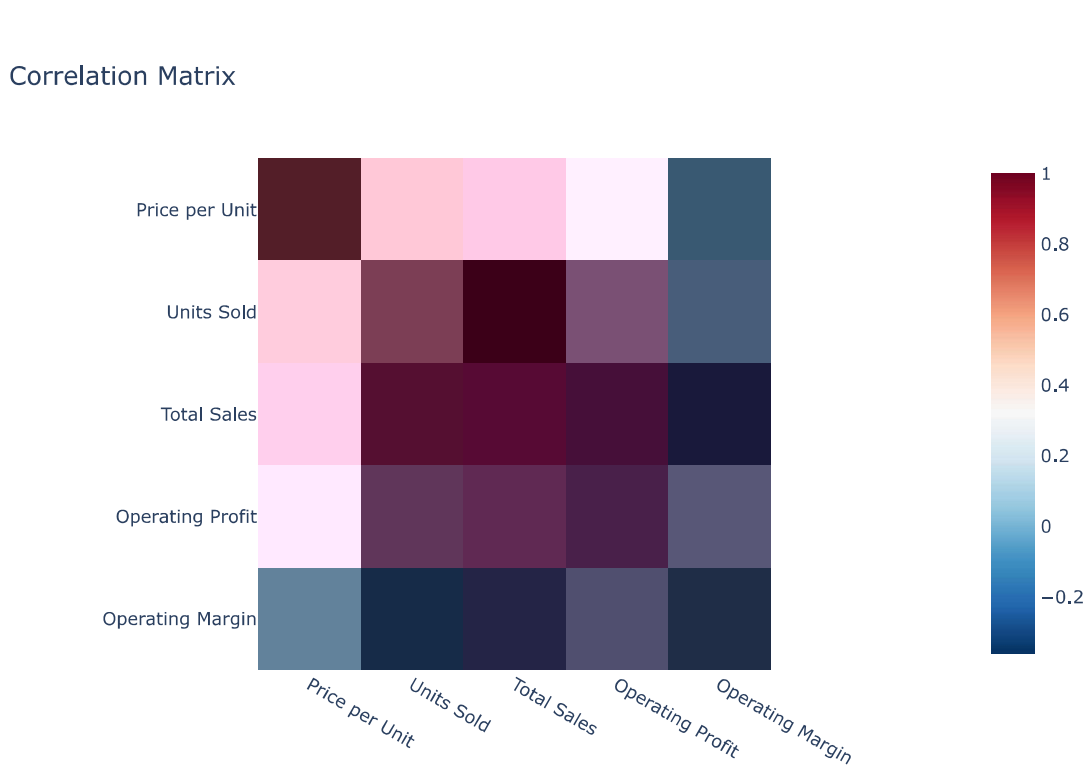

```
product_sales = df.groupby(['Retailer', 'Product'])['Total Sales'].sum().sort_values(ascending=False)
product_sales_df = product_sales.unstack(level=1)
fig=px.bar(product_sales_df, title='Total Sales by Product and Retailer')
fig.show()
```



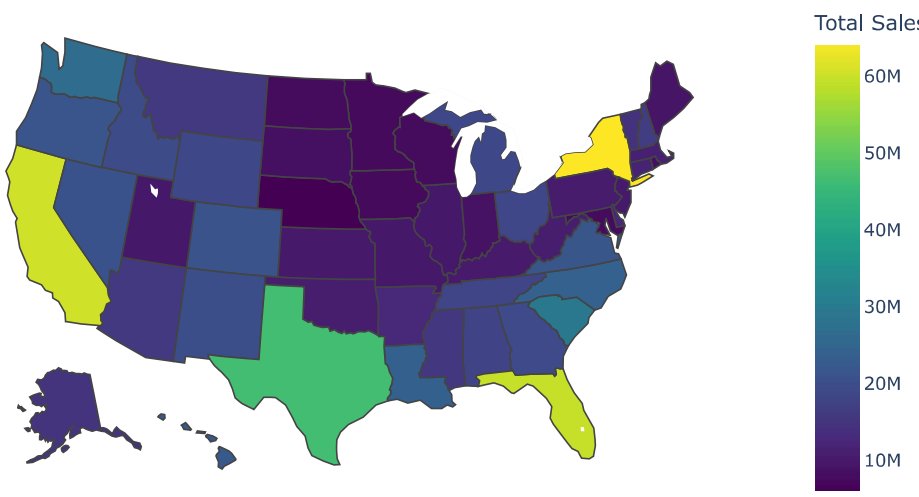
```
import plotly.express as px

# Group the data by retailer and sum the total sales for each retailer
sales_retailer = df.groupby('Retailer')['Total Sales'].sum()

# Calculate the total sales of all retailers
sales_total= sales_retailer.sum()

# Calculate the market share of each retailer by dividing their total sales by the total sales of all retailers
market_share = sales_retailer / sales_total

# Create a pie chart using plotly
fig = px.pie(market_share, values=market_share, names=market_share.index, title='Market Share of Retailers')

# Show the plot
fig.show()
```



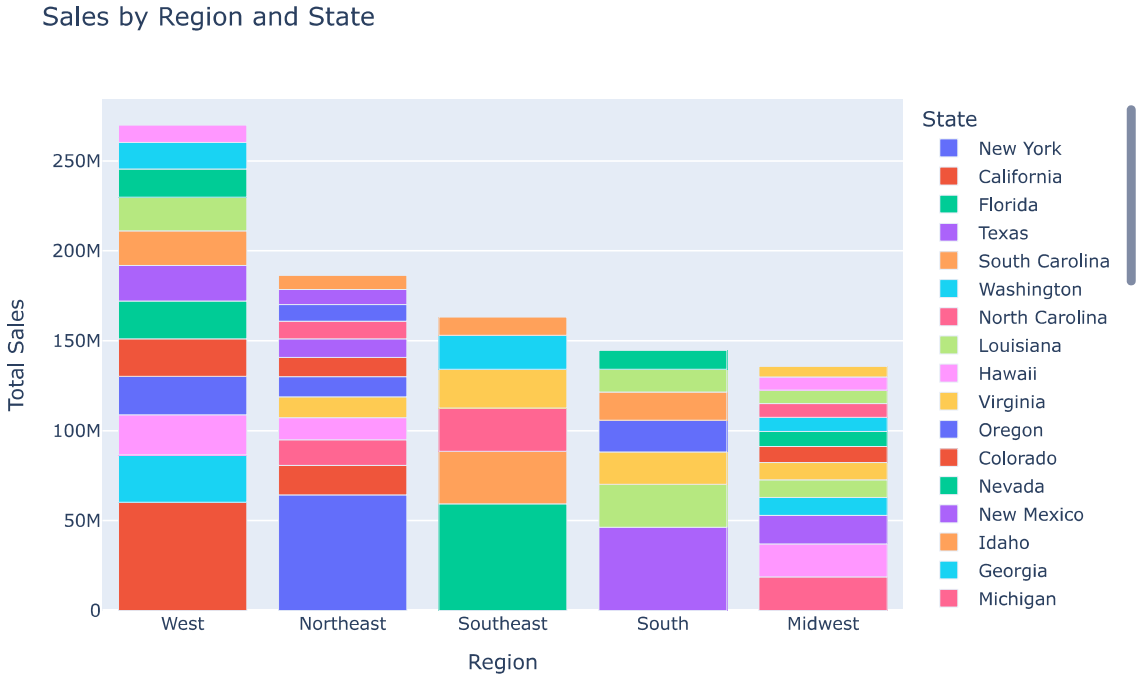

```
numerical_cols=['Price per Unit', 'Units Sold', 'Total Sales', 'Operating Profit', 'Operating Margin']
correlation_matrix=df[numerical_cols].corr()

fig=px.imshow(correlation_matrix, title='Correlation Matrix', color_continuous_scale='RdBu_r')
fig.show()
```



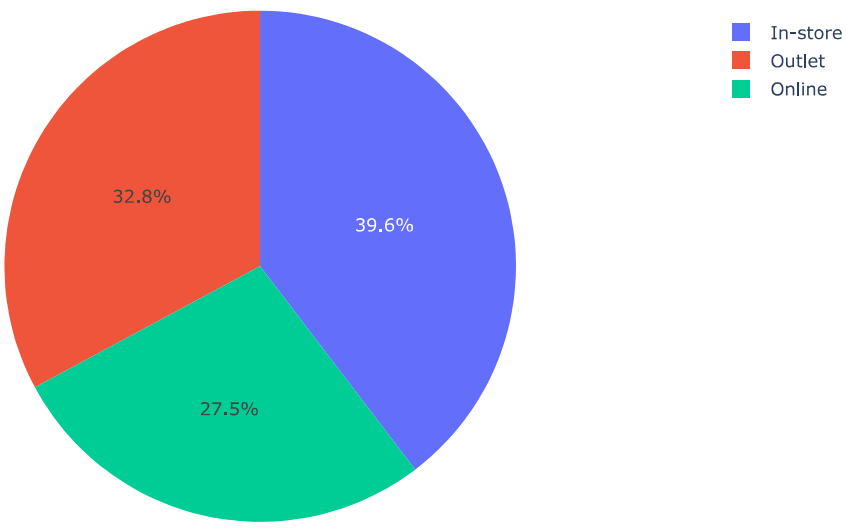
```
    us_state_abbrev = {
        'Alabama': 'AL',
        'Alaska': 'AK',
        'Arizona': 'AZ',
        'Arkansas': 'AR',
        'California': 'CA',
        'Colorado': 'CO',
        'Connecticut': 'CT',
        'Delaware': 'DE',
        'Florida': 'FL',
        'Georgia': 'GA',
        'Hawaii': 'HI',
        'Idaho': 'ID',
        'Illinois': 'IL',
        'Indiana': 'IN',
        'Iowa': 'IA',
        'Kansas': 'KS',
        'Kentucky': 'KY',
        'Louisiana': 'LA',
        'Maine': 'ME',
        'Maryland': 'MD',
        'Massachusetts': 'MA',
        'Michigan': 'MI',
        'Minnesota': 'MN',
        'Mississippi': 'MS',
        'Missouri': 'MO',
        'Montana': 'MT',
        'Nebraska': 'NE',
        'Nevada': 'NV',
        'New Hampshire': 'NH',
        'New Jersey': 'NJ',
        'New Mexico': 'NM',
        'New York': 'NY',
        'North Carolina': 'NC',
        'North Dakota': 'ND',
        'Ohio': 'OH',
        'Oklahoma': 'OK',
        'Oregon': 'OR',
        'Pennsylvania': 'PA',
        'Rhode Island': 'RI',
        'South Carolina': 'SC',
        'South Dakota': 'SD',
        'Tennessee': 'TN',
        'Texas': 'TX',
        'Utah': 'UT',
        'Vermont': 'VT',
        'Virginia': 'VA',
        'Washington': 'WA',
        'West Virginia': 'WV',
        'Wisconsin': 'WI',
        'Wyoming': 'WY'
    }
    # state_sales['State_abbrev'] = state_sales['State'].map(us_state_abbrev)


    state_sales= df.groupby('State')['Total Sales'].sum().reset_index()
    state_sales['State_abbrev'] = state_sales['State'].map(us_state_abbrev)
    fig=px.choropleth(state_sales,
                      locations='State_abbrev',
                      locationmode='USA-states',
                      color='Total Sales',
                      scope='usa',
                      title='Geographic Distribution of Sales',
                      color_continuous_scale='Viridis')
    fig.show()
```



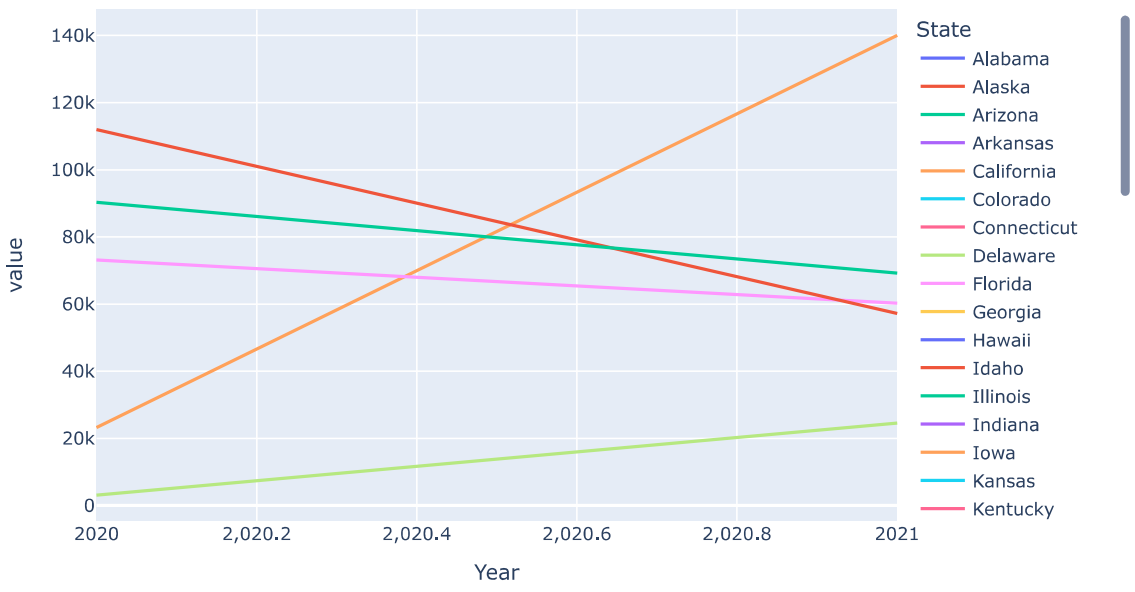

```
sales_by_region_state=df.groupby(['Region', 'State'])['Total Sales'].sum().sort_values(ascending=False).reset_index()

sorted_regions=sales_by_region_state.groupby('Region')['Total Sales'].sum().sort_values(ascending=False).index.tolist()
fig=px.bar(sales_by_region_state, x='Region', y='Total Sales', color='State', title='Sales by Region and State', category_orders={'Region':sorted_regions})
fig.show()
```



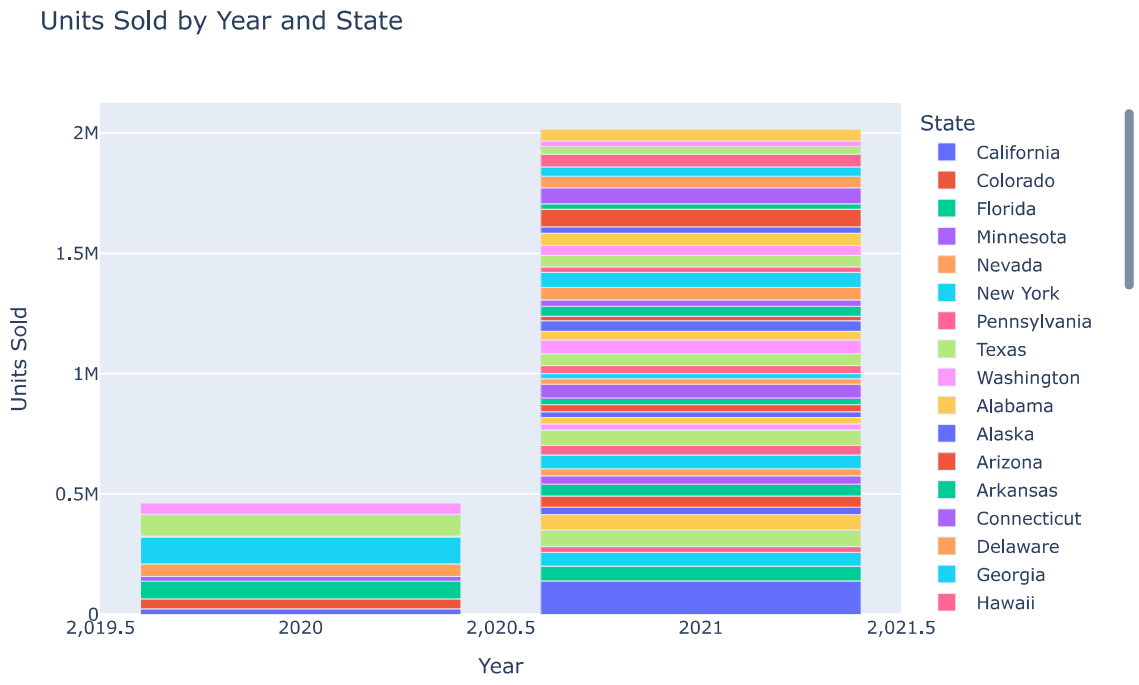

```
sales_by_method=df.groupby('Sales Method')['Total Sales'].sum().reset_index()
fig=px.pie(sales_by_method, values='Total Sales', names='Sales Method', title='Sales distribution by Method')
fig.show()
```




```
units_by_year_and_state = df.groupby(['Year', 'State'])['Units Sold'].sum()

units_by_year_and_state = units_by_year_and_state.reset_index()

units_by_year = units_by_year_and_state.pivot(index='Year', columns='State', values='Units Sold')

px.line(units_by_year, title='Units Sold by Year and State')
```



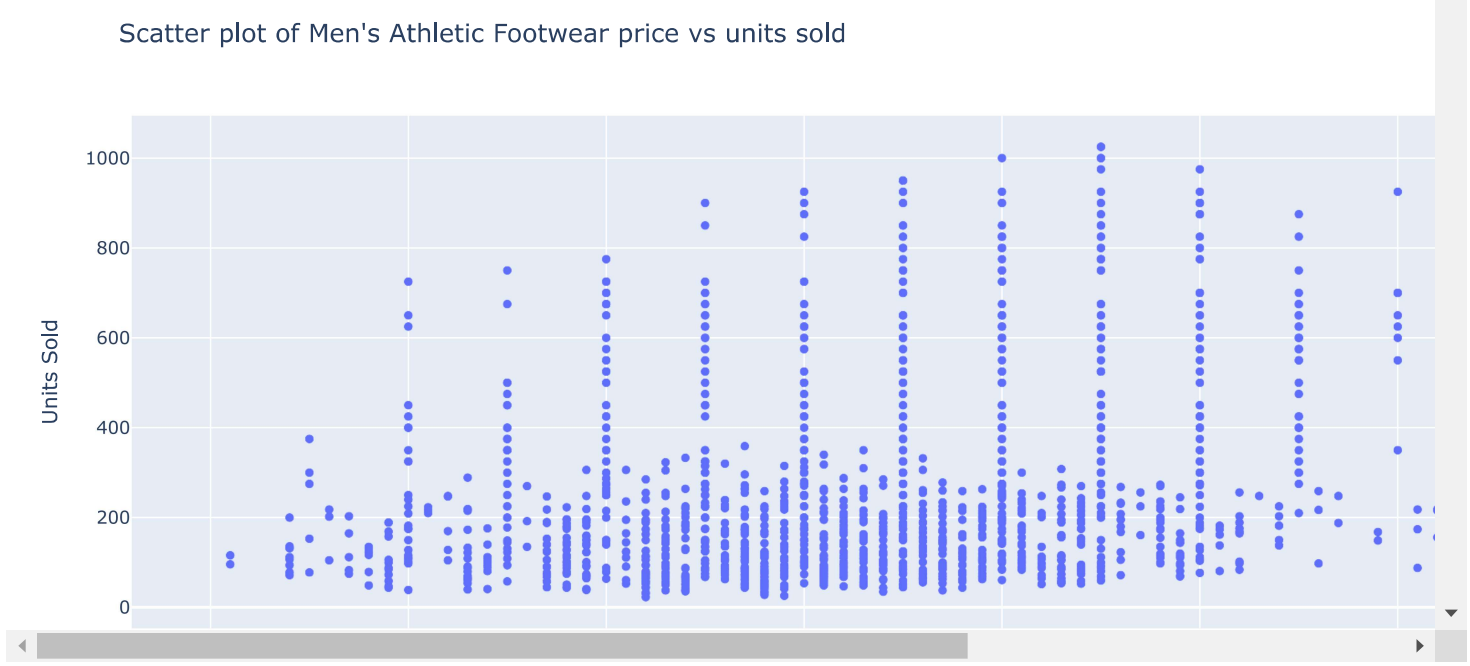

```
units_by_year_and_state = units_by_year_and_state.reset_index()
px.bar(units_by_year_and_state, x='Year', y='Units Sold', color='State', title='Units Sold by Year and State')
```



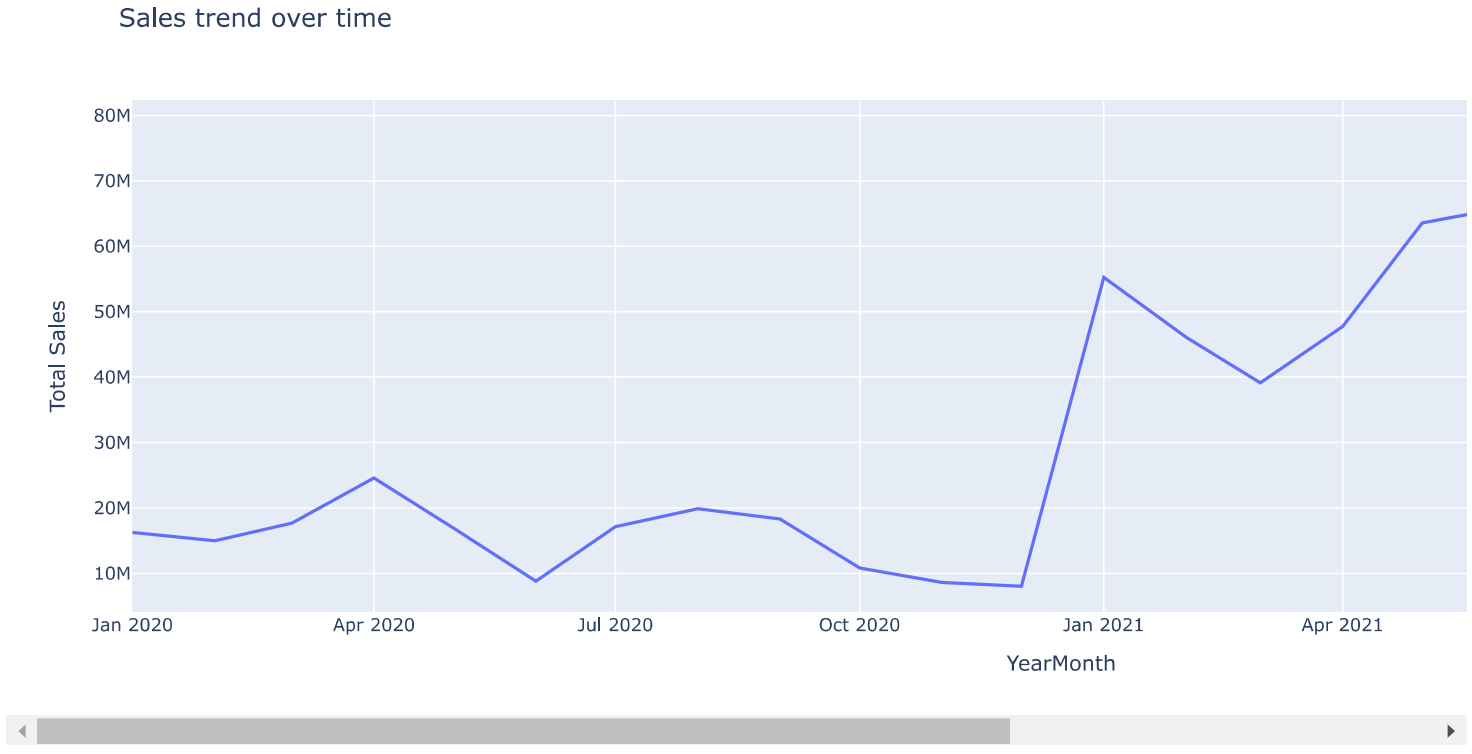

```
fig=px.scatter( df,
                x='Price per Unit',
                y='Units Sold',
                color='Product',
                title='Scatter plot of price vs units sold'
                )
fig.show()
```


```
grouped_data=df.groupby('Price per Unit')['Units Sold'].mean().reset_index()

for product in df['Product'].unique():

  fig=px.scatter(df[df['Product']==product],
                 x='Price per Unit',
                 y='Units Sold',
                 title=f'Scatter plot of {product} price vs units sold'
                 )
  fig.show()
```


Scatter plot of Men's Street Footwear price vs units sold
Units Sold
Price per Unit
200
400
600
800
1000
1200
10
20
30
40
50
60
70


Scatter plot of Men's Athletic Footwear price vs units sold
Units Sold
0
200
400
600
800
1000

```
fig=px.line(sales_by_month, x='YearMonth', y='Total Sales', title='Sales trend over time')
fig.show()
```

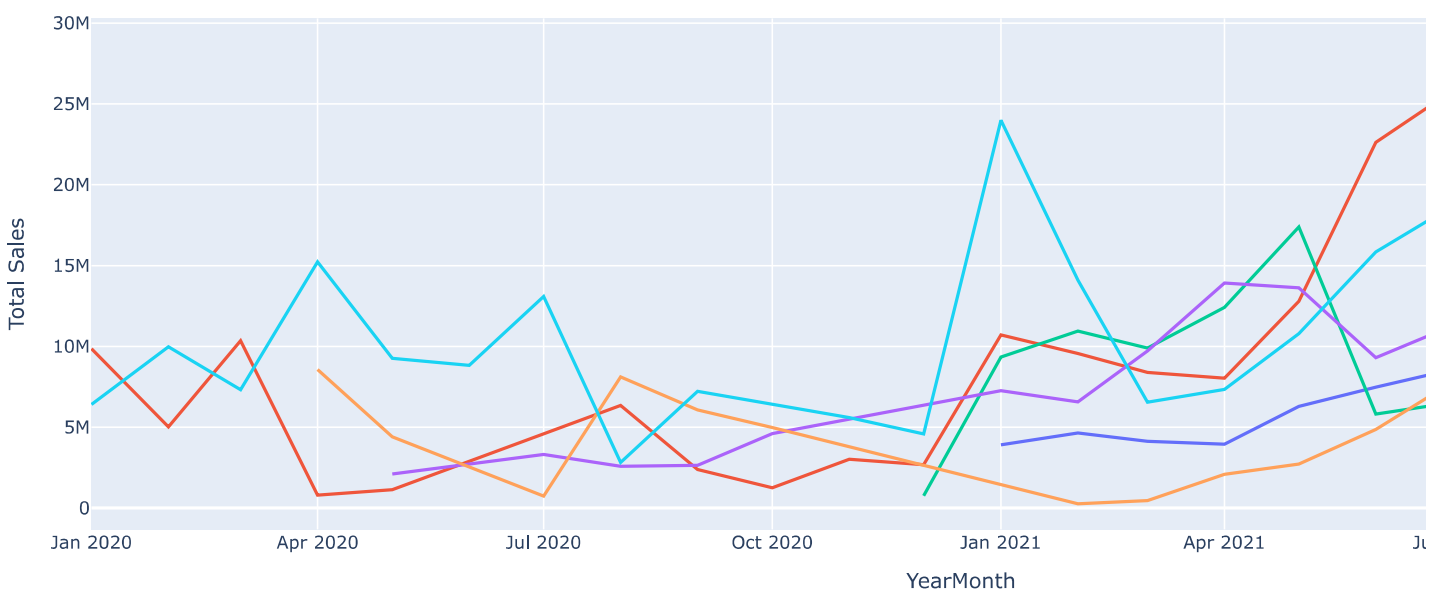


Start coding or generate with AI.

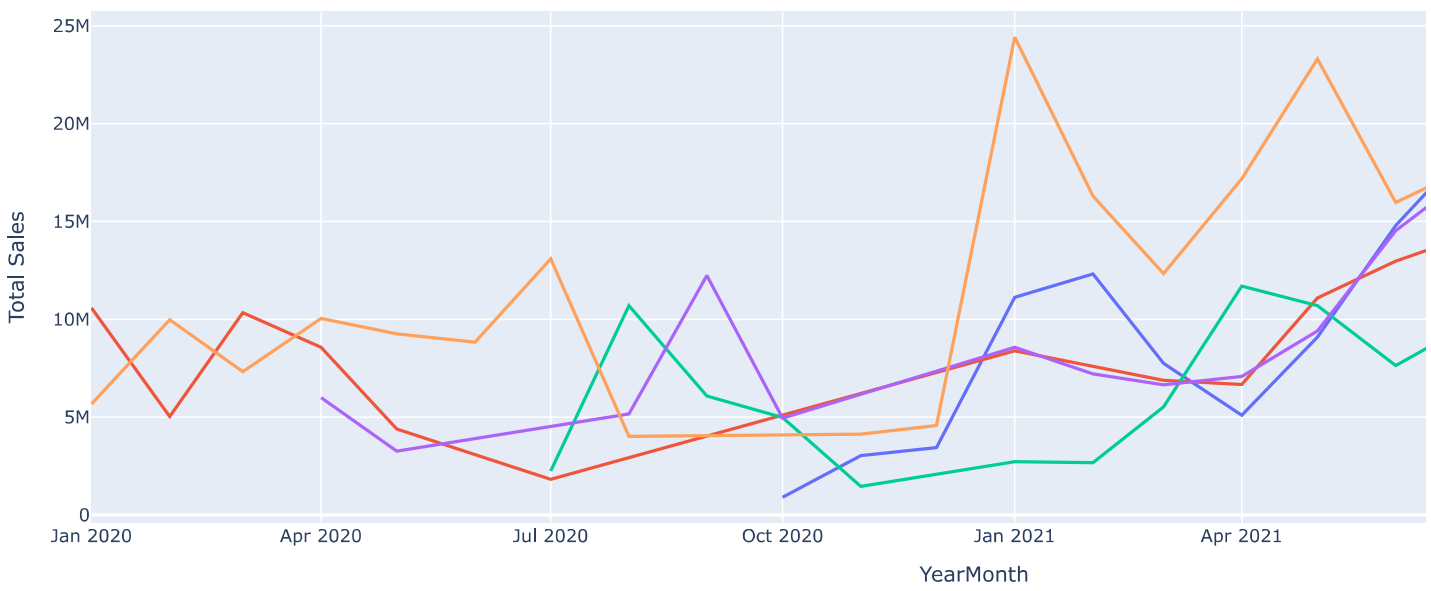
```
df.head()
```

	Retailer	Retailer ID	Invoice Date	Region	State	City	Product	Price per Unit	Units Sold	Total Sales	Operating Profit	Operating Margin	Sales Method	Month	Year
1	Foot Locker	1185732	2020-01-01	Northeast	New York	New York	Men's Street Footwear	50.0	1200	600000	300000	50	In-store	1	2020
2	Foot Locker	1185732	2020-01-02	Northeast	New York	New York	Men's Athletic Footwear	50.0	1000	500000	150000	30	In-store	1	2020
3	Foot Locker	1185732	2020-01-03	Northeast	New York	New York	Women's Street Footwear	40.0	1000	400000	140000	35	In-store	1	2020

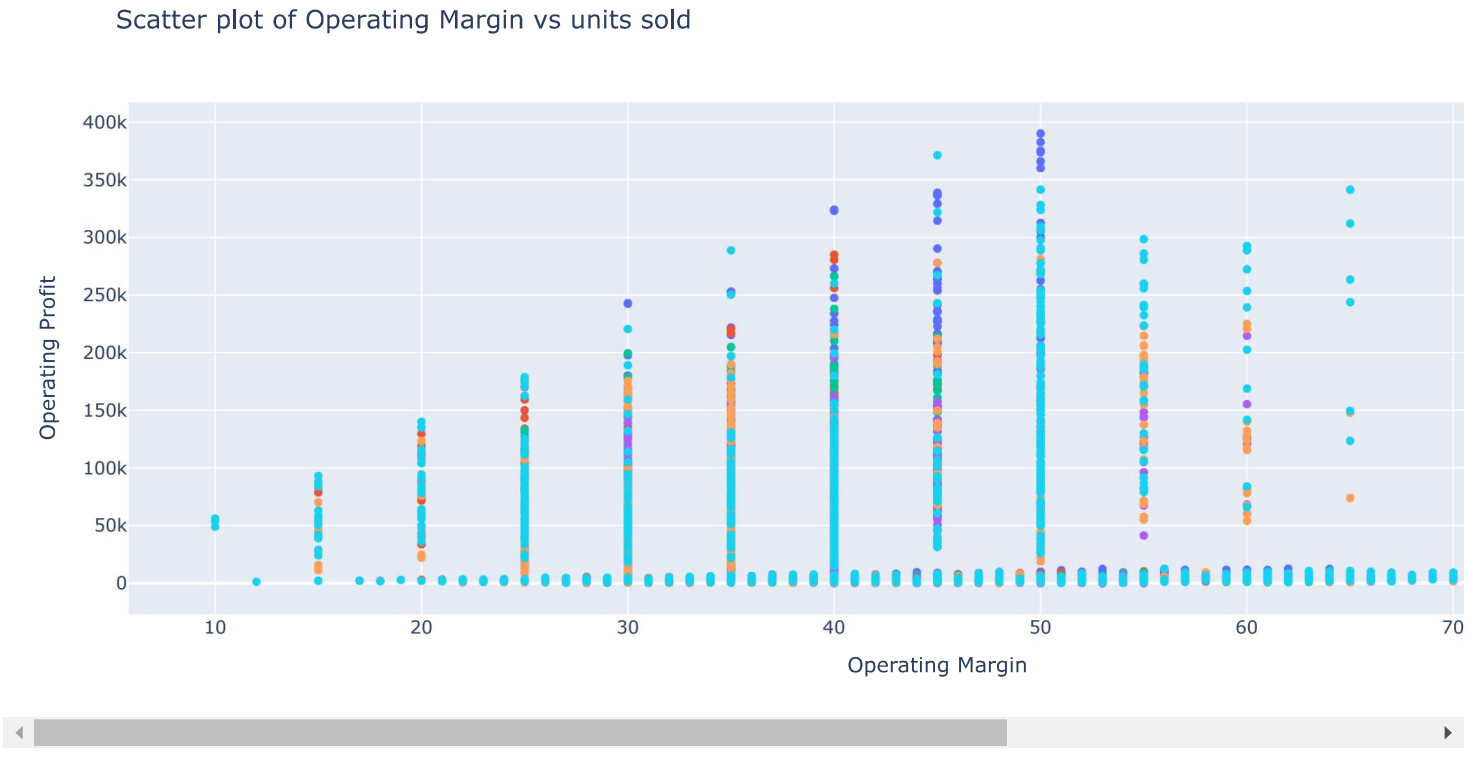
```
sales_by_retailer_month=df.groupby(['Retailer','YearMonth'])['Total Sales'].sum().reset_index()

sales_by_retailer_month['YearMonth']=sales_by_retailer_month['YearMonth'].astype(str)
fig=px.line(sales_by_retailer_month, x='YearMonth', y='Total Sales', color='Retailer', title='Sales trend over time')
fig.show()
```




```
sales_by_region=df.groupby(['Region','YearMonth'])['Total Sales'].sum().reset_index()
sales_by_region['YearMonth']=sales_by_region['YearMonth'].astype(str)
fig=px.line(sales_by_region, x='YearMonth', y='Total Sales', color='Region', title='Sales trend over time')
fig.show()
```




```
fig=px.scatter( df,
                x='Operating Margin',
                y='Operating Profit',
                color='Product',
                title='Scatter plot of Operating Margin vs units sold'
                )
fig.show()
```



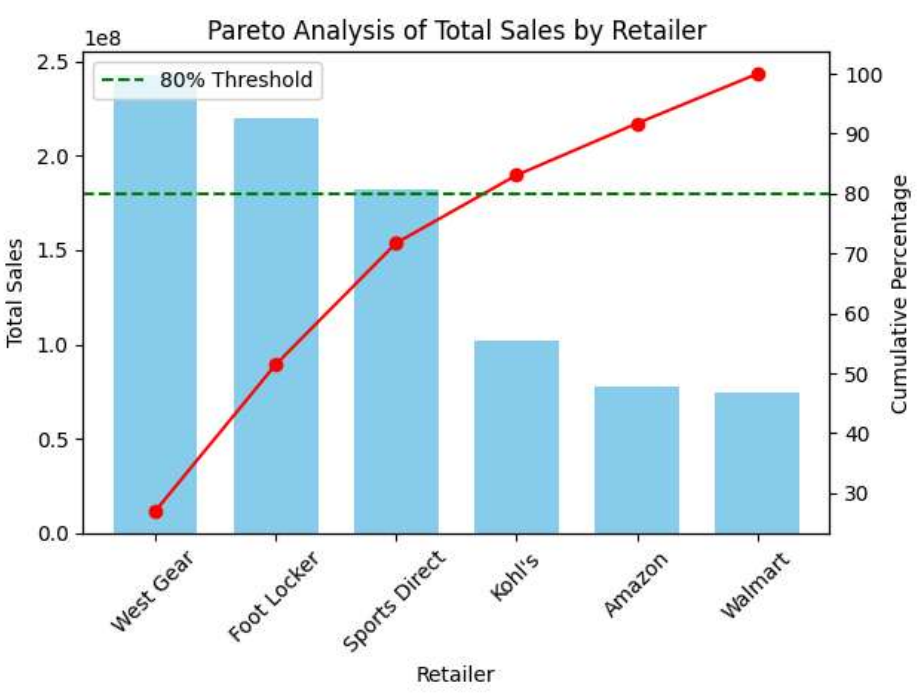

```
df.head()
```

	Retailer	Retailer ID	Invoice Date	Region	State	City	Product	Price per Unit	Units Sold	Total Sales	Operating Profit	Operating Margin	Sales Method	Month	Year
1	Foot Locker	1185732	2020-01-01	Northeast	New York	New York	Men's Street Footwear	50.0	1200	600000	300000	50	In-store	1	2020
2	Foot Locker	1185732	2020-01-02	Northeast	New York	New York	Men's Athletic Footwear	50.0	1000	500000	150000	30	In-store	1	2020
3	Foot Locker	1185732	2020-01-03	Northeast	New York	New York	Women's Street Footwear	40.0	1000	400000	140000	35	In-store	1	2020

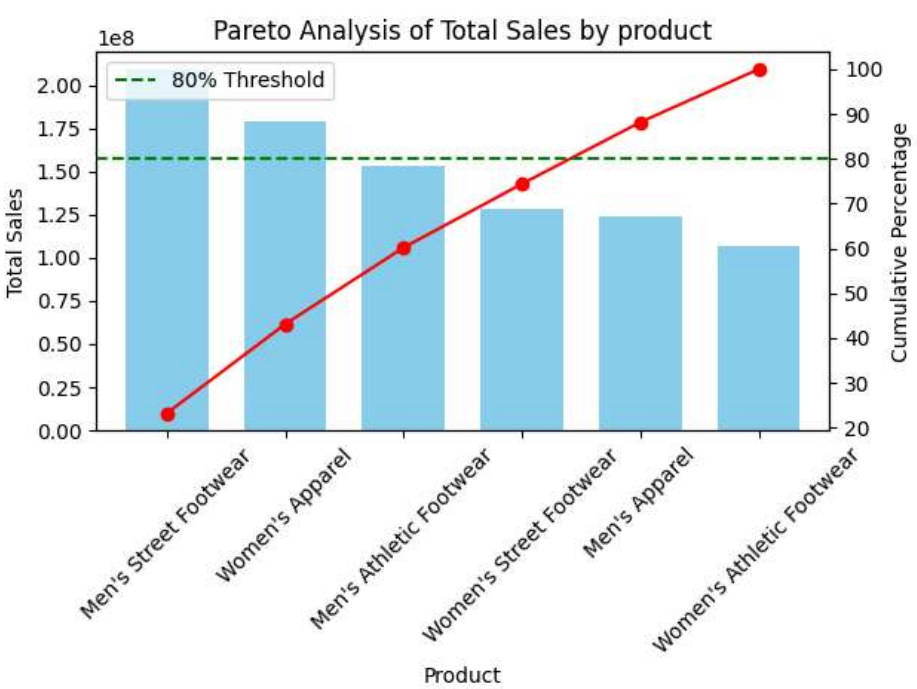
```
!pip install paretochart
```

```
Collecting paretochart
  Downloading paretochart-1.0.tar.gz (5.7 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from paretochart) (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->paretochart) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->paretochart) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->paretochart) (4.55.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->paretochart) (1.4.7)
Requirement already satisfied: numpy<2,>=1.21 in /usr/local/lib/python3.10/dist-packages (from matplotlib->paretochart) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->paretochart) (24.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->paretochart) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->paretochart) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->paretochart) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->paretochart) (1.16.0)
Building wheels for collected packages: paretochart
  Building wheel for paretochart (setup.py) ... done
  Created wheel for paretochart: filename=paretochart-1.0-py3-none-any.whl size=6025 sha256=2430b5b870bf8e561695f1bed062d06faf08abde1c4f1f1ae68a8fb20445b891
  Stored in directory: /root/.cache/pip/wheels/2c/1b/c5/f800a2d0620bdc1e240a98200ac97c997c63272a0963278160
Successfully built paretochart
Installing collected packages: paretochart
Successfully installed paretochart-1.0
```

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

# Assuming 'df' is your DataFrame and has columns 'Retailer' and 'Total Sales'
# Calculate total sales for each retailer
retailer_sales = df.groupby('Retailer')['Total Sales'].sum().sort_values(ascending=False)

# Calculate cumulative percentage
cumulative_percentage = (retailer_sales.cumsum() / retailer_sales.sum()) * 100

# Plot bar chart for total sales
fig, ax1 = plt.subplots()

# Bar chart
retailer_sales.plot(kind='bar', color='skyblue', ax=ax1, width=0.7)
ax1.set_ylabel('Total Sales')
ax1.set_xlabel('Retailer')
ax1.tick_params(axis='x', rotation=45)

# Add secondary axis for cumulative percentage
ax2 = ax1.twinx()
ax2.plot(cumulative_percentage.index, cumulative_percentage, color='red', marker='o', linestyle='-')
ax2.set_ylabel('Cumulative Percentage')
ax2.axhline(80, color='green', linestyle='--', label='80% Threshold')
ax2.legend(loc='best')

# Title and layout adjustments
plt.title('Pareto Analysis of Total Sales by Retailer')
plt.tight_layout()
plt.show()
```




```
product_sales = df.groupby('Product')['Total Sales'].sum().sort_values(ascending=False)
cumulative_percentage = (product_sales.cumsum() / product_sales.sum()) * 100

fig, ax1 = plt.subplots()

product_sales.plot(kind='bar', color='skyblue', ax=ax1, width=0.7)
ax1.set_ylabel('Total Sales')
ax1.set_xlabel('Product')
ax1.tick_params(axis='x', rotation=45)

ax2 = ax1.twinx()
ax2.plot(cumulative_percentage.index, cumulative_percentage, color='red', marker='o', linestyle='-')
ax2.set_ylabel('Cumulative Percentage')
ax2.axhline(80, color='green', linestyle='--', label='80% Threshold')
ax2.legend(loc='best')

plt.title('Pareto Analysis of Total Sales by product')
plt.tight_layout()
plt.show()
```



Models

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9648 entries, 1 to 9648
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Retailer          9648 non-null   object
 1   Retailer ID       9648 non-null   object
 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   int64
 10  Operating Profit  9648 non-null   int64
 11  Operating Margin  9648 non-null   int64
 12  Sales Method      9648 non-null   object
 13  Month             9648 non-null   int32
 14  Year              9648 non-null   int32
 15  Quarter           9648 non-null   int32
 16  YearMonth         9648 non-null   period[M]
dtypes: datetime64[ns](1), float64(1), int32(3), int64(4), object(7), period[M](1)
memory usage: 1.1+ MB
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```