In [1]:
```python
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go

from PIL import Image
import IPython.display as display

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
```

In [2]:
```python
df = pd.read_csv("Sports_H_and_M.csv")
```

In [3]:
```python
df.head()
```

Out[3]:

	Name_of_product	category	price_of_product(in dollar)	brand_name	Samples in total	product_lir
0	Sports Shorts	sportswear_men_clothing_shorts	12.99	H&M	3	/en_us/productpage.0995634001.htr
1	DryMove Sports Joggers	sportswear_men_clothing_trousersjoggers	39.99	H&M	6	/en_us/productpage.0862104002.htr
2	DryMove Sports Joggers	sportswear_men_clothing_trousersjoggers	39.99	H&M	6	/en_us/productpage.0862104009.htr
3	DryMove Sports Hoodie	men_sport_tops	64.99	H&M	3	/en_us/productpage.1113571001.htr
4	Sports Shorts	sportswear_men_clothing_shorts	12.99	H&M	3	/en_us/productpage.0995634007.htr

Converted price_of_product(in dollar) column ----> prices in dollars

In [4]:
```python
df = df.rename(columns={'price_of_product(in dollar)': 'prices in dollars'})
```

In [5]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9146 entries, 0 to 9145
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name_of_product    9146 non-null   object 
 1   category           9146 non-null   object 
 2   prices in dollars  9146 non-null   float64
 3   brand_name         9146 non-null   object 
 4   Samples in total   9146 non-null   int64  
 5   product_link       9146 non-null   object 
 6   swatches_color     9146 non-null   object 
dtypes: float64(1), int64(1), object(5)
memory usage: 500.3+ KB
```

Checking for null values, (no null values encountered)

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

Out[6]:
```
Name_of_product      0
category             0
prices in dollars    0
brand_name           0
Samples in total     0
product_link         0
swatches_color       0
dtype: int64
```

Calculating total sales revenue

In [7]:
```python
total_sales = round(df['prices in dollars'].sum(), 2)
print("Total Sales Revenue: $", total_sales)
```

```
Total Sales Revenue: $ 401799.92
```

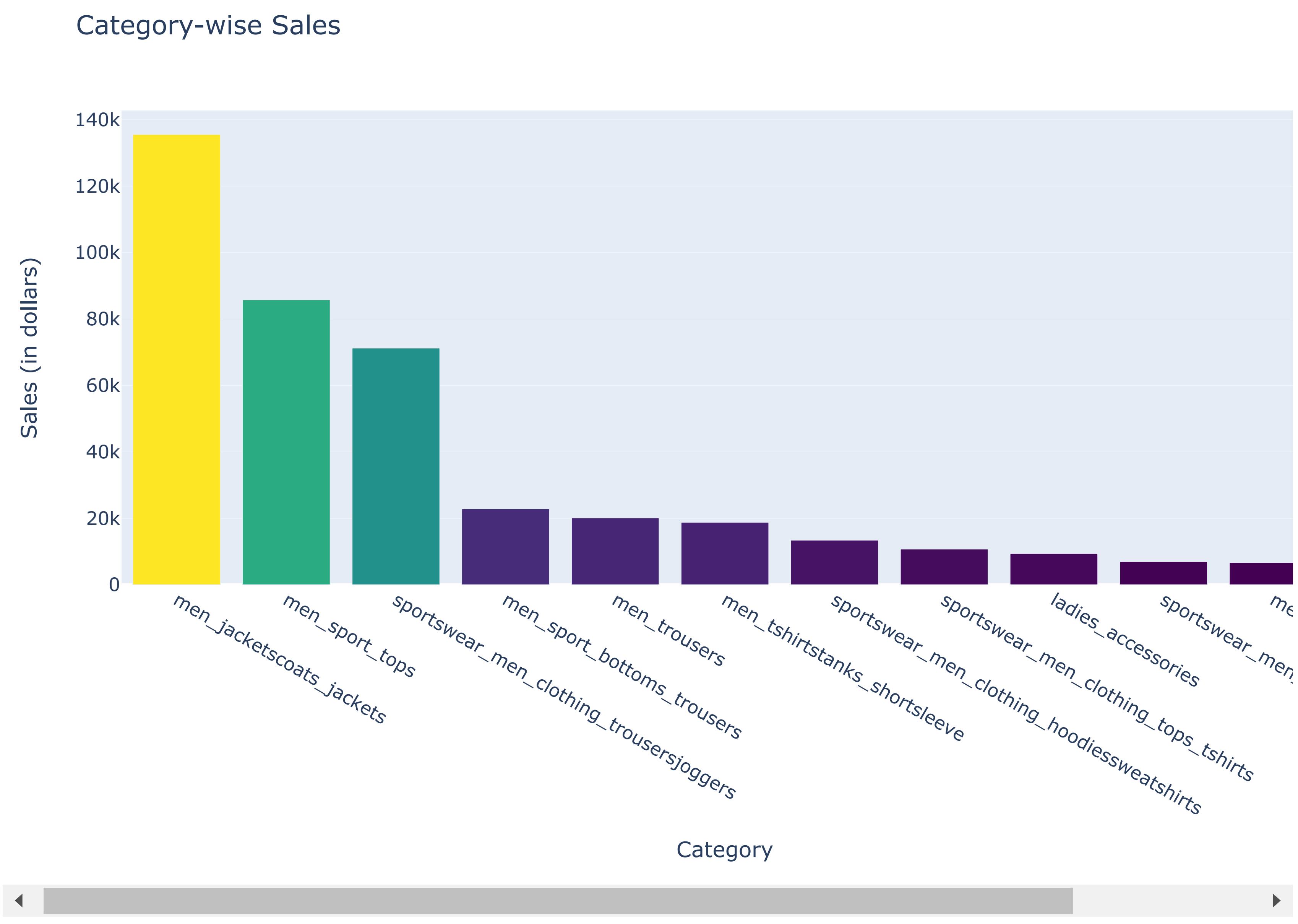
Analyzing category-wise sales

In [8]:
```python
category_sales = df.groupby('category')['prices in dollars'].sum()
category_sales = category_sales.sort_values(ascending=False)  # Sort the categories by sales in descendir

category_sales
```

Out[8]:
```
category
men_jacketscoats_jackets                     135557.17
men_sport_tops                                85800.24
sportswear_men_clothing_trousersjoggers       71266.17
men_sport_bottoms_trousers                    22862.31
men_trousers                                  20172.31
men_tshirtstanks_shortsleeve                  18816.55
sportswear_men_clothing_hoodiessweatshirts    13447.31
sportswear_men_clothing_tops_tshirts          10754.62
ladies_accessories                             9412.31
sportswear_men_clothing_shorts                 6988.62
men_tshirtstanks_longsleeve                    6722.31
Name: prices in dollars, dtype: float64
```

Creating a bar chart for category-wise sales

In [34]:
```python
# Calculate the sales for each category
category_sales = df.groupby('category')['prices in dollars'].sum().sort_values(ascending=False)

# Create the bar chart
fig = go.Figure(data=[go.Bar(
    x=category_sales.index,
    y=category_sales.values,
    marker=dict(
        color=category_sales.values,
        colorscale='Viridis',
        colorbar=dict(title='Sales'),
    )
)])

# Set chart labels and title
fig.update_layout(
    title="Category-wise Sales",
    xaxis_title="Category",
    yaxis_title="Sales (in dollars)",
    height=600,  # Adjust the height as desired
    width=1000  # Adjust the width as desired
)

# Display the chart
fig.show()

```



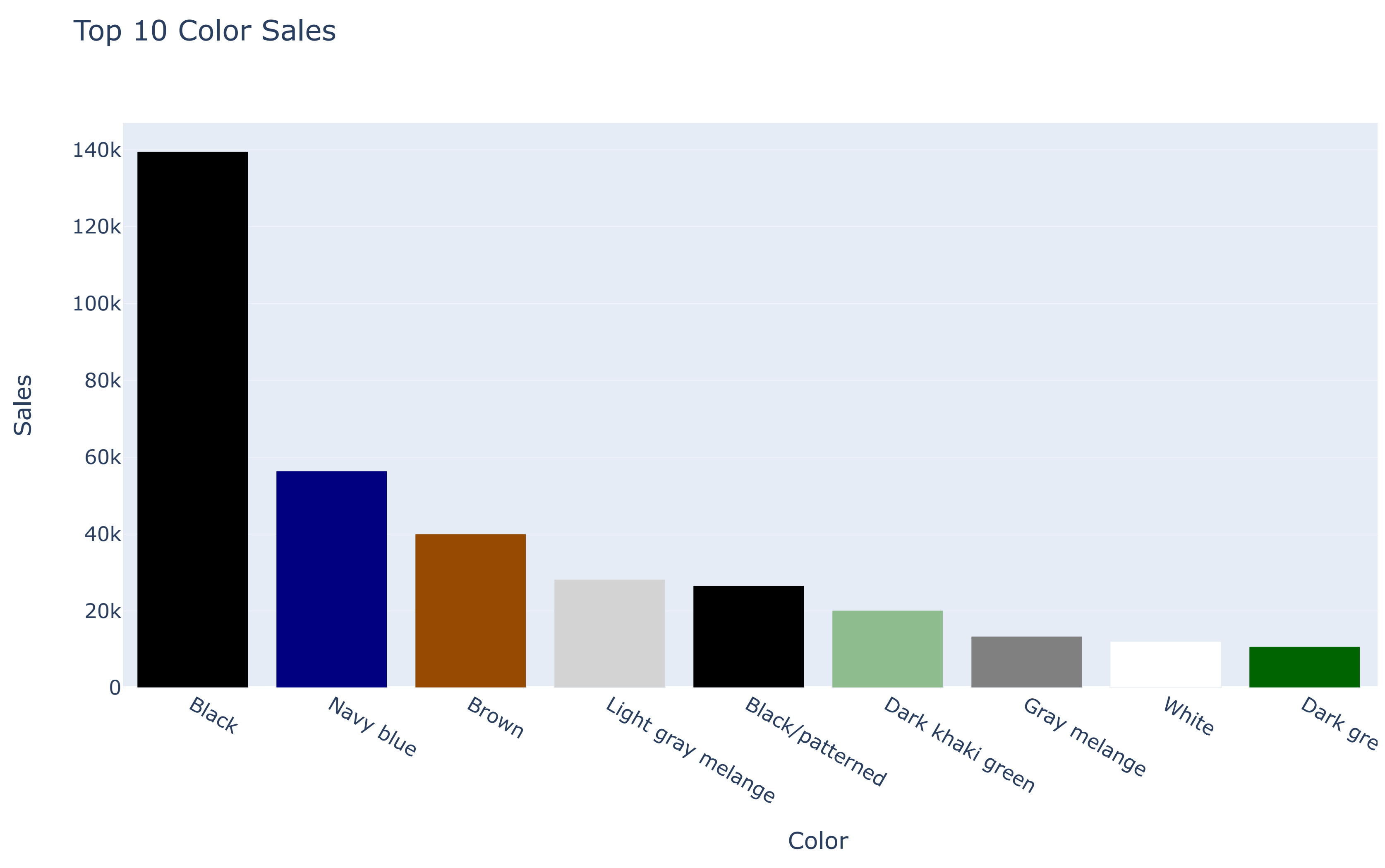
In [29]:
```python
# Calculating the sales for each swatch color
color_sales = df.groupby('swatches_color')['prices in dollars'].sum().sort_values(ascending=False)

# Selecting the top 10 colors
top_10_colors = color_sales.head(10)

# Filtering the dataframe to include only the top 10 colors
df_top_colors = df[df['swatches_color'].isin(top_10_colors.index)]

# Creating a cross-tab of color and product
color_product_cross_tab = pd.crosstab(df_top_colors['swatches_color'], df_top_colors['Name_of_product'])

color_product_cross_tab
```

Out[29]:

Name_of_product	DryMove Reversible Track Jacket	DryMove Running Shirt	DryMove Sports Hoodie	DryMove Sports Joggers	DryMove Sports Shirt	Fast-drying Sports Shirt	Padded Leg Gaiters	Padded Shell Pants	Puffer Pants with Belt	Regular Fit Fast-drying Track Jacket	Regular Fit Lightweight Outdoor Jacket	Regular Fit Padded Vest
swatches_color												
Black	0	269	269	0	538	269	269	0	269	0	269	269
Black/patterned	269	0	0	0	0	0	0	0	0	0	0	0
Brown	0	0	0	0	0	0	0	0	0	0	0	0
Dark gray melange	0	0	0	269	0	0	0	0	0	0	0	0
Dark green	0	0	0	269	0	0	0	0	0	0	0	0
Dark khaki green	0	0	0	0	0	0	0	269	0	0	0	0
Gray melange	0	0	0	0	0	0	0	0	0	269	0	0
Light gray melange	0	0	269	269	0	0	0	0	0	0	0	0
Navy blue	0	0	269	269	0	0	0	0	0	0	269	0
White	0	0	0	0	269	0	0	0	0	0	0	0

Below graph describes which color has the more sales ⬇

In [32]:
```python
# Calculate the sales for each swatch color
color_sales = df.groupby('swatches_color')['prices in dollars'].sum().sort_values(ascending=False)

# Selecting the top 10 colors
top_10_colors = color_sales.head(10)

# Defining the custom colors for each bar
custom_colors = ['#000000', '#000080', '#964B00', '#D3D3D3', '#000000', '#8FBC8F', '#808080',
                 '#FFFFFF', '#006400', '#A9A9A9']

# Creating a bar chart for the top 10 color sales
bar = go.Bar(x=top_10_colors.index,
             y=top_10_colors.values,
             marker=dict(color=custom_colors)
             )

layout_bar = go.Layout(title='Top 10 Color Sales',
                       xaxis=dict(title='Color'),
                       yaxis=dict(title='Sales'),
                       height=550,  # Adjust the height as desired
                       width=1000  # Adjust the width as desired
                       )

figure_bar = go.Figure(data=[bar], layout=layout_bar)
figure_bar.show()

```



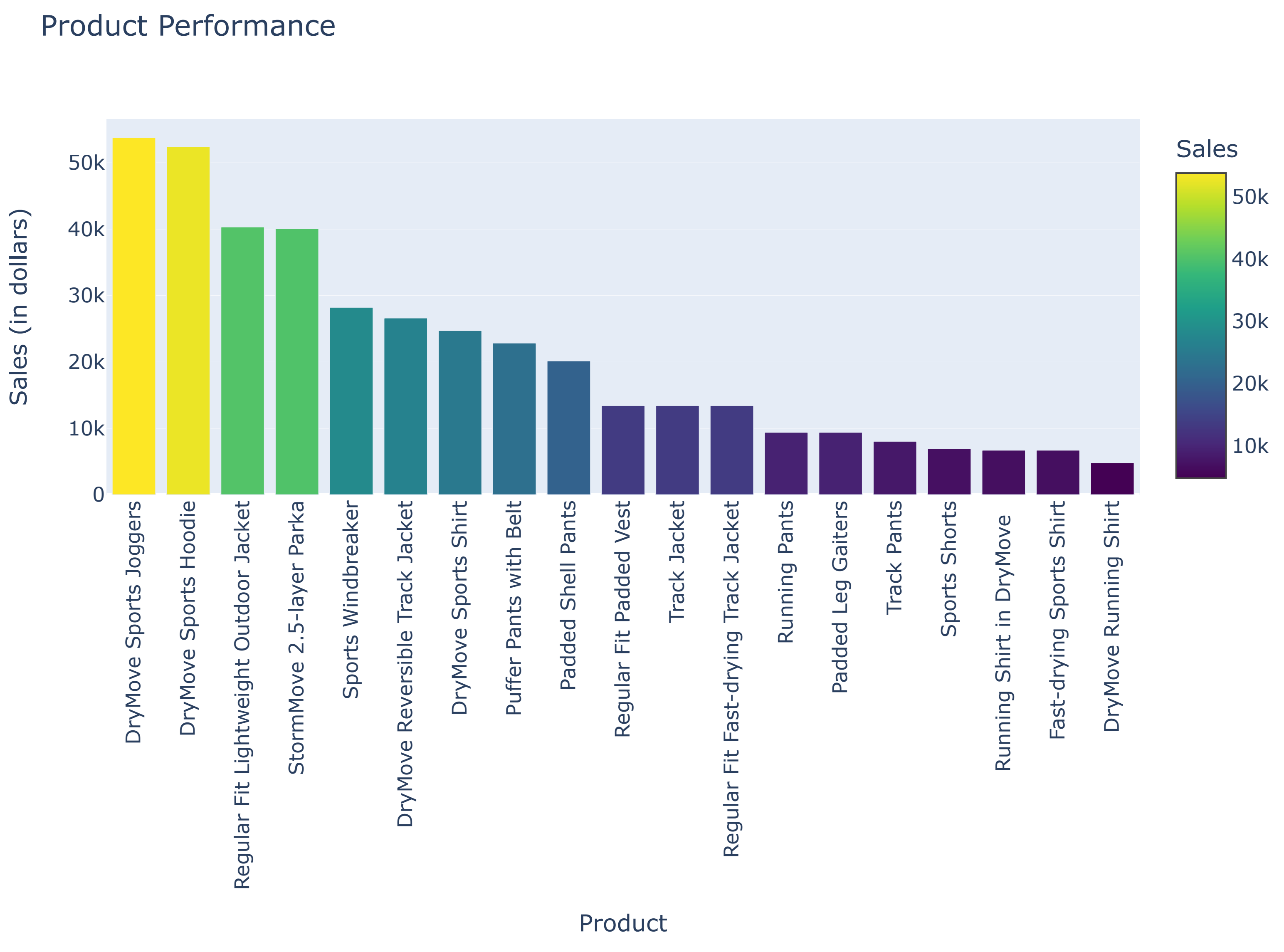
Analyzing product performance

In [11]:
```
hm_df = df[df['brand_name'] == 'H&M']


product_sales = hm_df.groupby('Name_of_product')['prices in dollars'].sum()
product_sales = product_sales.sort_values(ascending=False)

print("\nProduct Performance:")
print(product_sales)
```

```

Product Performance:
Name_of_product
DryMove Sports Joggers                    53786.55
DryMove Sports Hoodie                     52446.93
Regular Fit Lightweight Outdoor Jacket    40344.62
StormMove 2.5-layer Parka                 40081.00
Sports Windbreaker                        28236.93
DryMove Reversible Track Jacket           26631.00
DryMove Sports Shirt                      24731.86
Puffer Pants with Belt                    22862.31
Padded Shell Pants                        20172.31
Regular Fit Padded Vest                   13447.31
Track Jacket                              13447.31
Regular Fit Fast-drying Track Jacket      13447.31
Running Pants                              9412.31
Padded Leg Gaiters                         9412.31
Track Pants                                8067.31
Sports Shorts                              6988.62
Running Shirt in DryMove                   6722.31
Fast-drying Sports Shirt                   6722.31
DryMove Running Shirt                      4839.31
Name: prices in dollars, dtype: float64
```

Product Performance Bar Chart ⬇

In [12]:
```python
fig_sales = go.Figure(data=[go.Bar(
    x=product_sales.index,
    y=product_sales.values,
    marker=dict(
        color=product_sales.values,
        colorscale='Viridis',
        colorbar=dict(title='Sales'),
    )
)])

# Setting up the chart labels and title for product performance
fig_sales.update_layout(
    title="Product Performance",
    xaxis_title="Product",
    yaxis_title="Sales (in dollars)"
)


fig_sales.update_layout(xaxis_tickangle=-90)
fig_sales.update_layout(height=600, width=800)

fig_sales.show()
```



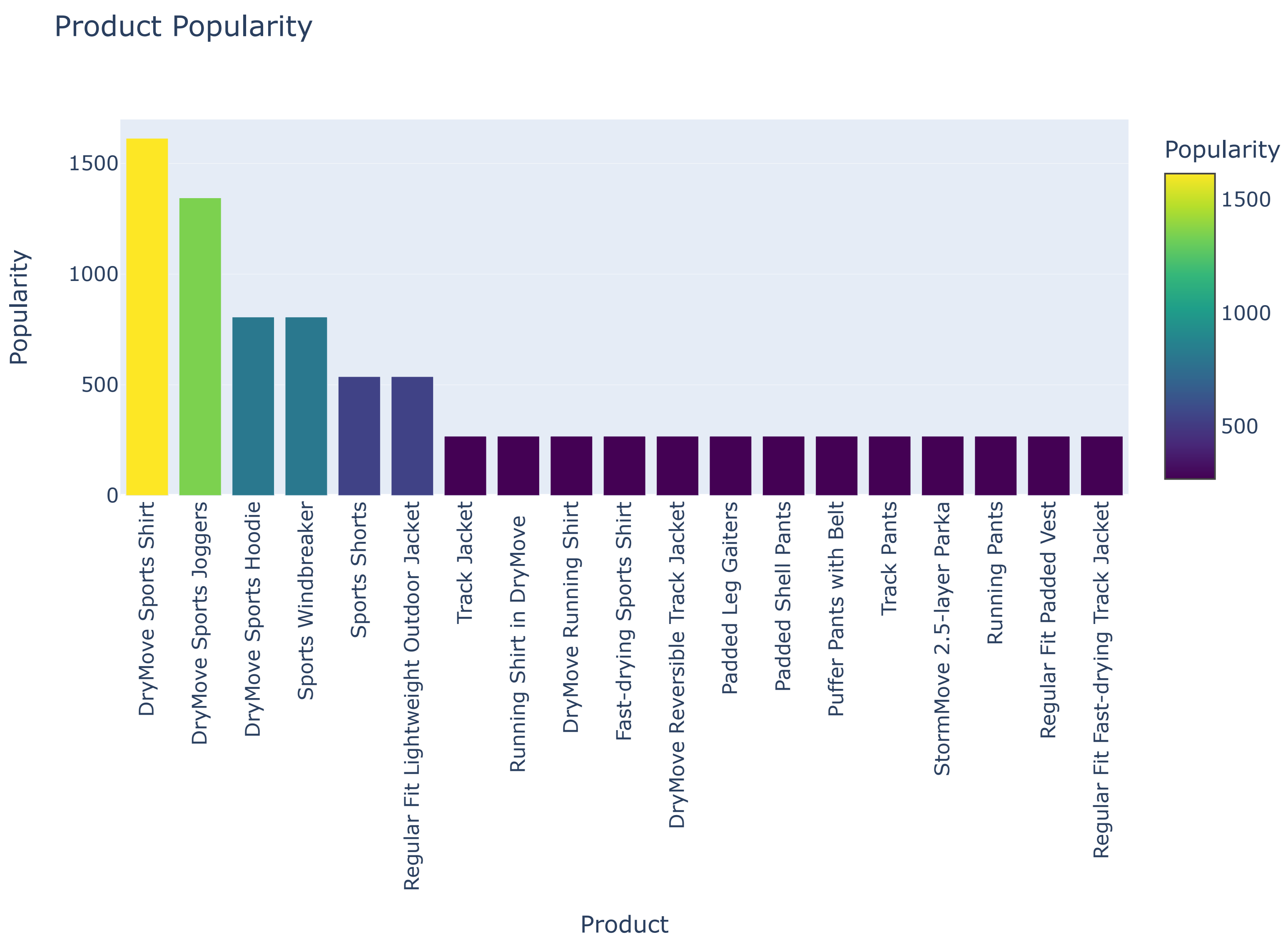
Analyzing product popularity

In [13]:
```
product_popularity = hm_df['Name_of_product'].value_counts()
print("\nProduct Popularity:")
print(product_popularity)
```

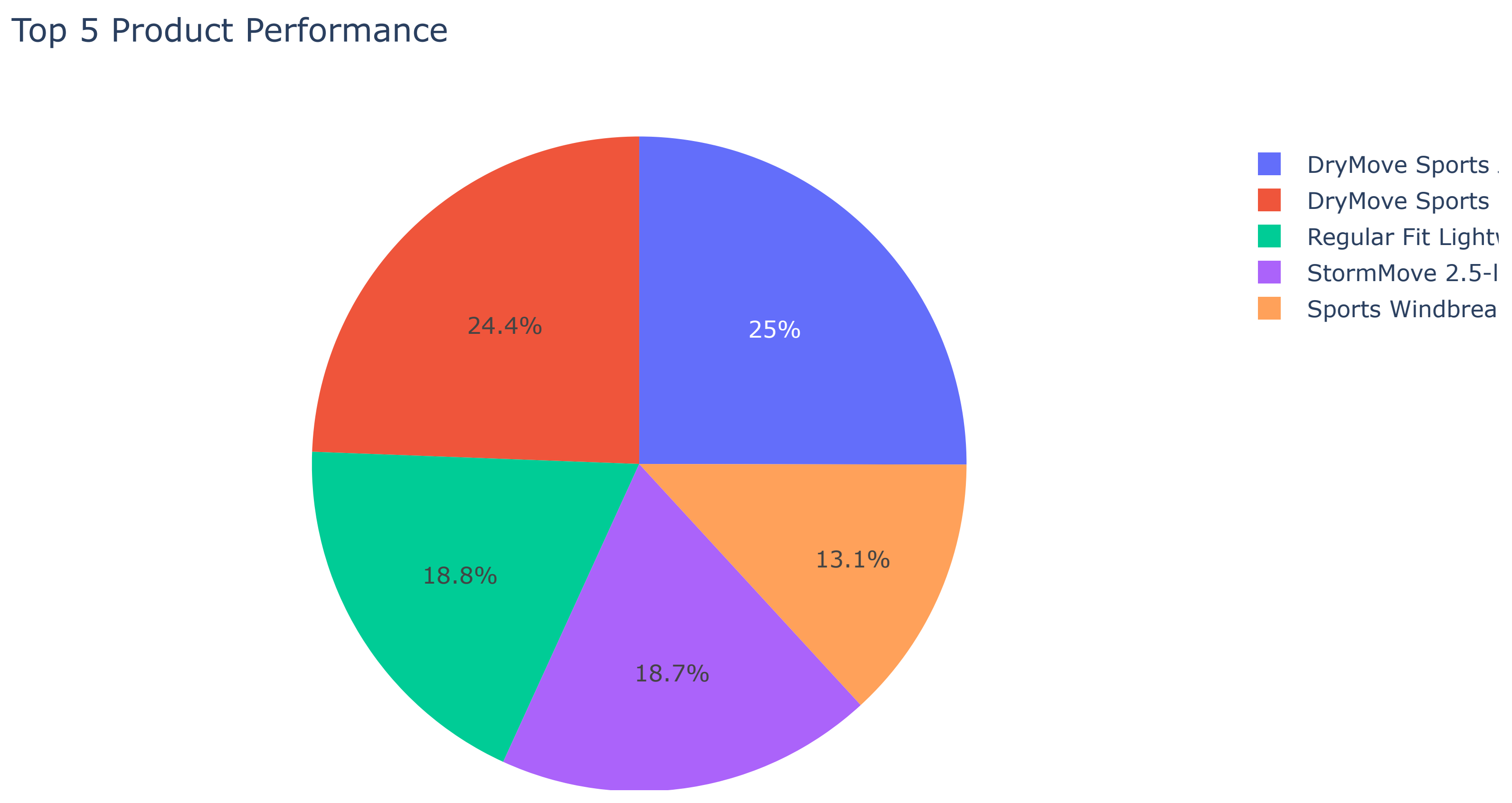
```
Product Popularity:
DryMove Sports Shirt                        1614
DryMove Sports Joggers                      1345
DryMove Sports Hoodie                        807
Sports Windbreaker                           807
Sports Shorts                                538
Regular Fit Lightweight Outdoor Jacket       538
Track Jacket                                 269
Running Shirt in DryMove                     269
DryMove Running Shirt                        269
Fast-drying Sports Shirt                     269
DryMove Reversible Track Jacket              269
Padded Leg Gaiters                           269
Padded Shell Pants                           269
Puffer Pants with Belt                       269
Track Pants                                  269
StormMove 2.5-layer Parka                    269
Running Pants                                269
Regular Fit Padded Vest                      269
Regular Fit Fast-drying Track Jacket         269
Name: Name_of_product, dtype: int64
```

Product Popoularity bar chart ⬇

In [14]:
```
fig_popularity = go.Figure(data=[go.Bar(
    x=product_popularity.index,
    y=product_popularity.values,
    marker=dict(
        color=product_popularity.values,
        colorscale='Viridis',
        colorbar=dict(title='Popularity'),
    )
)])

# Setting up the chart labels and title for product popularity
fig_popularity.update_layout(
    title="Product Popularity",
    xaxis_title="Product",
    yaxis_title="Popularity"
)

fig_popularity.update_layout(xaxis_tickangle=-90)
fig_popularity.update_layout(height=600, width=800)

fig_popularity.show()
```

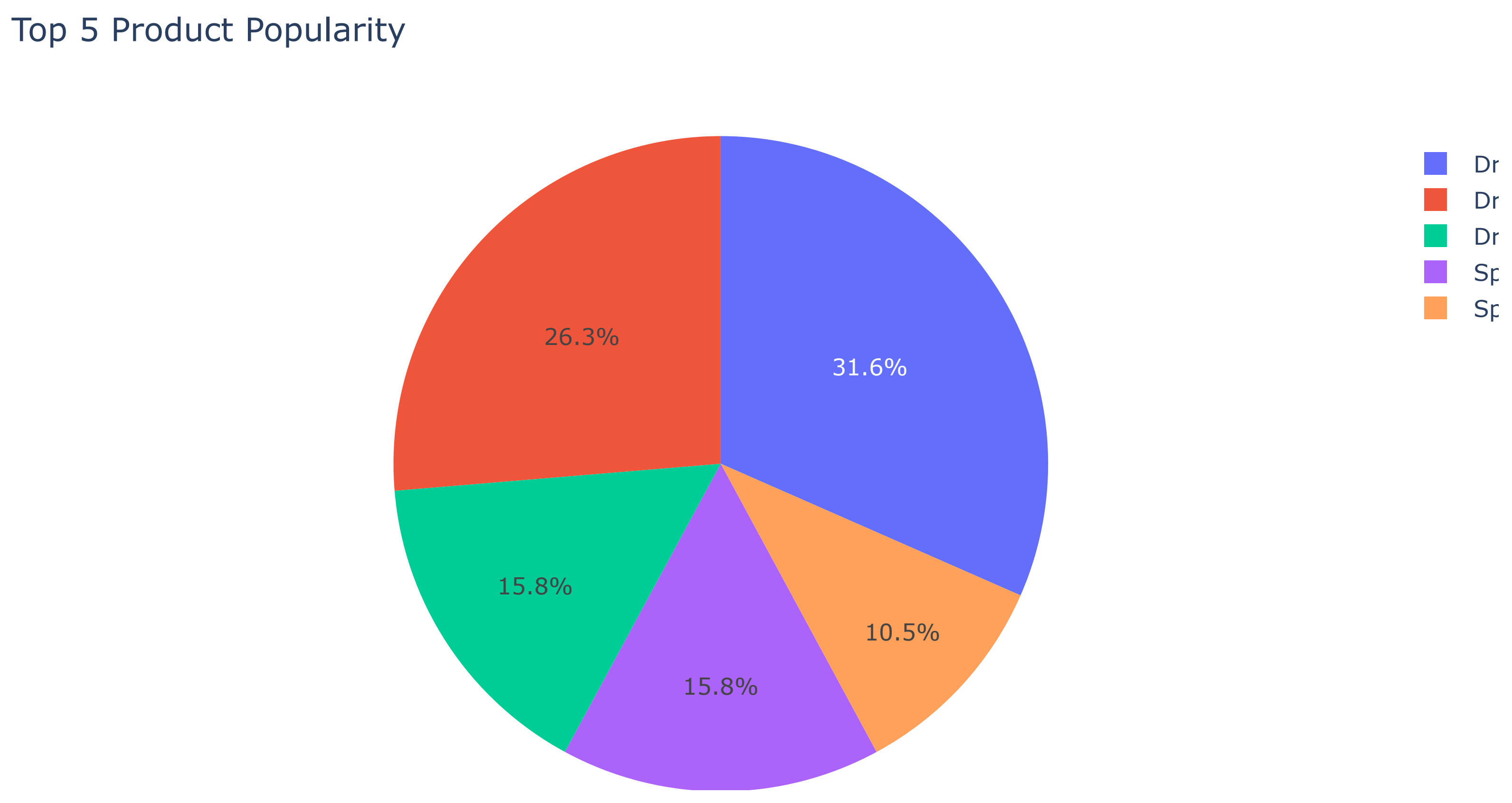


In [15]:
```python
top_5_products = product_sales.head(5)

# Create pie chart for product performance
fig_sales = px.pie(names=top_5_products.index, values=top_5_products.values, title="Top 5 Product Perforn

# Display the chart
fig_sales.show()
```



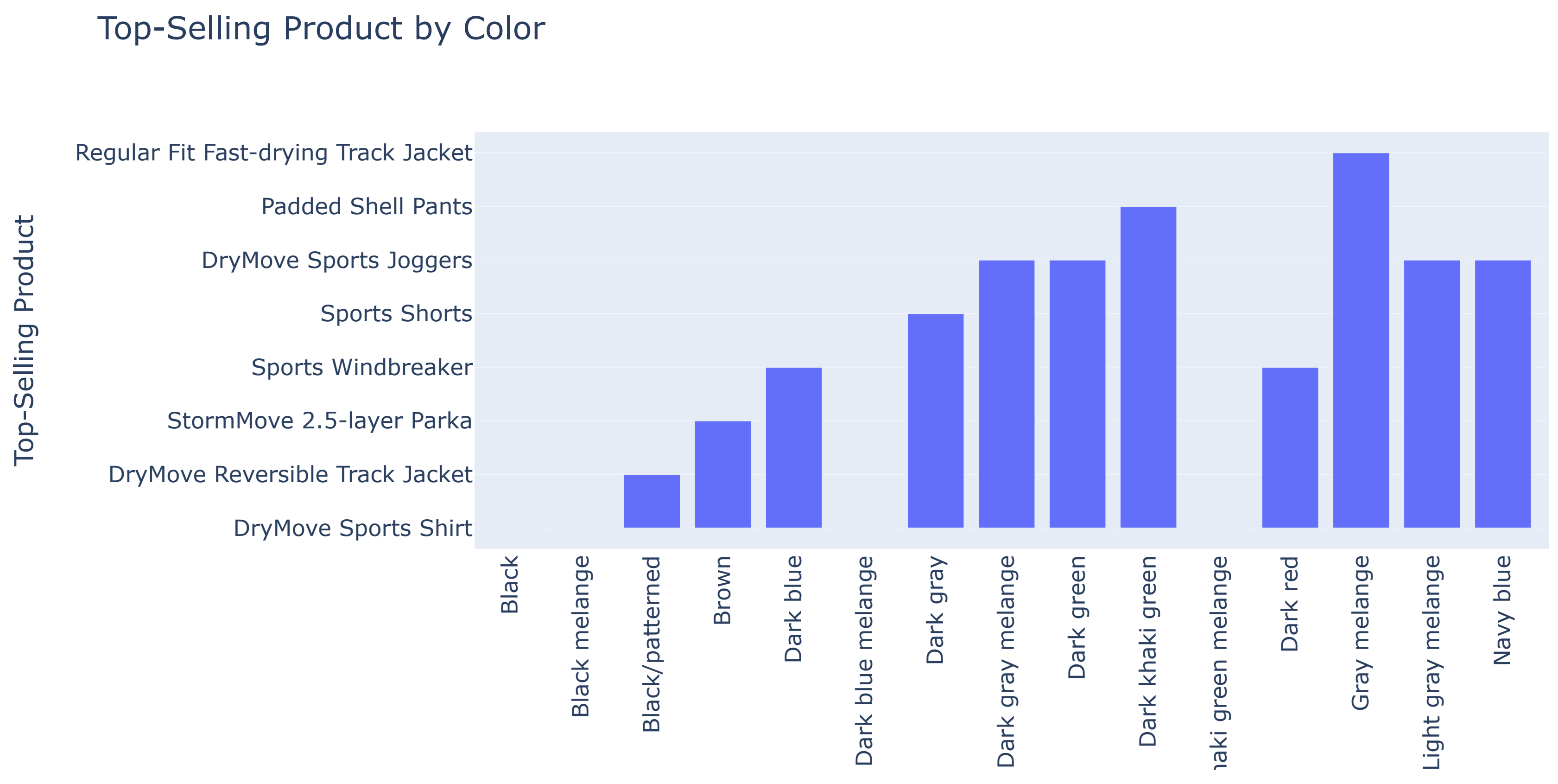
Above pie charts shows the top5 best performing products ⬆

In [16]:
```python
# Select top 5 products with highest popularity
top_5_products = product_popularity.head(5)

# Create pie chart for product popularity
fig_popularity = px.pie(names=top_5_products.index, values=top_5_products.values, title="Top 5 Product Pc

# Display the chart
fig_popularity.show()
```



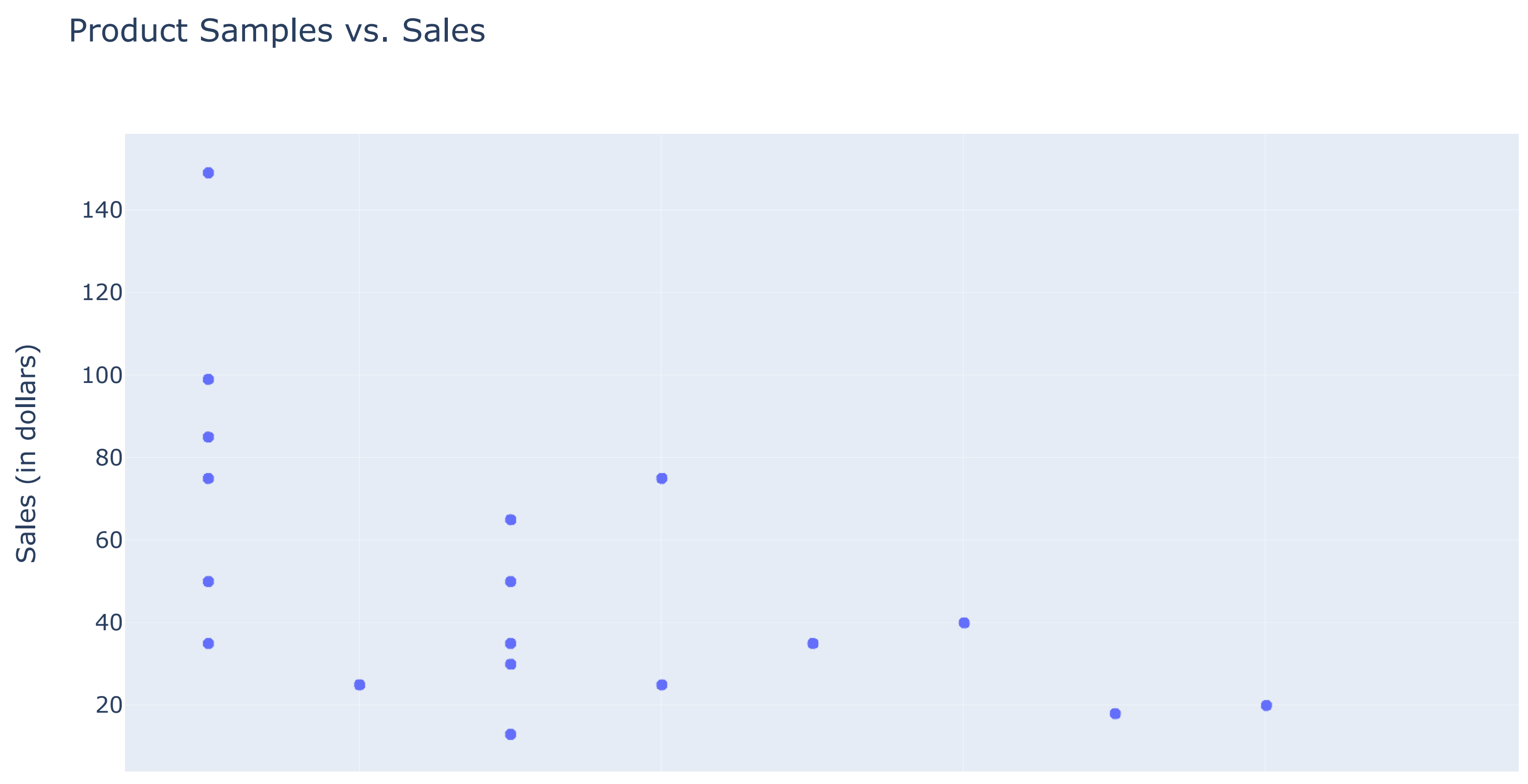
Above pie charts shows the top5 popular products ⬆

In [17]:
```
top_products = df.groupby('swatches_color')['Name_of_product'].agg(lambda x: x.value_counts().index[0]).r

# Plotting the top-selling products by color
fig = px.bar(top_products, x='swatches_color', y='Name_of_product',
             labels={'swatches_color': 'Color', 'Name_of_product': 'Top-Selling Product'},
             title='Top-Selling Product by Color')

fig.update_layout(xaxis_tickangle=-90)

fig.show()
```



Above graph shows the top selling product by color⬆

In [18]:
```python
# Filter data for H&M brand
hm_df = df[df['brand_name'] == 'H&M']

# Extract the relevant columns for analysis
samples_sales_df = hm_df[['Samples in total', 'prices in dollars']]

# Create scatter plot for product samples vs. sales
fig_samples_sales = px.scatter(samples_sales_df, x='Samples in total', y='prices in dollars',
                              title='Product Samples vs. Sales')

# Set chart labels
fig_samples_sales.update_layout(xaxis_title="Product Samples", yaxis_title="Sales (in dollars)")

# Display the chart
fig_samples_sales.show()
```



Above graph shows there are some outliers in the data where products with a relatively low number of samples have high sales.

These outliers indicate that factors other than the number of samples might influence sales performance.

Lets check what is that outlier👁

In [19]:
```python
# Filter data for H&M brand
hm_df = df[df['brand_name'] == 'H&M']

# Calculate the z-scores for sales and samples
hm_df['sales_zscore'] = (hm_df['prices in dollars'] - hm_df['prices in dollars'].mean()) / hm_df['prices
hm_df['samples_zscore'] = (hm_df['Samples in total'] - hm_df['Samples in total'].mean()) / hm_df['Samples

# Identify outliers based on z-scores
outliers = hm_df[(hm_df['sales_zscore'] > 2) | (hm_df['sales_zscore'] < -2) | (hm_df['samples_zscore'] >

# Display the products with outliers
outliers_products = outliers['Name_of_product'].unique()
print("Products with outliers:")
print(outliers_products)
```

```
Products with outliers:
['StormMove 2.5-layer Parka']
```

This is the product which has low samples but high sales

In [20]:
```
# Specify the path to the image file on your local device
image_path = "stormmove.jpg"  # Replace with the actual path to your image file

# Open the image file
image = Image.open(image_path)

# Display the image
display.display(image)
```

In [21]:
```
outliers
```

931	StormMove 2.5-layer Parka	men_jacketscoats_jackets	149.0	H&M	1	/en_us/productpage.1067832001.html	Brown
965	StormMove 2.5-layer Parka	men_jacketscoats_jackets	149.0	H&M	1	/en_us/productpage.1067832001.html	Brown
999	StormMove 2.5-layer Parka	men_jacketscoats_jackets	149.0	H&M	1	/en_us/productpage.1067832001.html	Brown
1033	StormMove 2.5-layer Parka	men_jacketscoats_jackets	149.0	H&M	1	/en_us/productpage.1067832001.html	Brown
1067	StormMove 2.5-layer Parka	men_jacketscoats_jackets	149.0	H&M	1	/en_us/productpage.1067832001.html	Brown
1101	StormMove 2.5-layer Parka	men_jacketscoats_jackets	149.0	H&M	1	/en_us/productpage.1067832001.html	Brown
1135	StormMove 2.5-layer Parka	men_jacketscoats_jackets	149.0	H&M	1	/en_us/productpage.1067832001.html	Brown
1169	StormMove 2.5-	men_jacketscoats_jackets	149.0	H&M	1	/en_us/productpage.1067832001.html	Brown

Now let's check some Stats

In [22]:
```
stats = df.describe()
stats
```

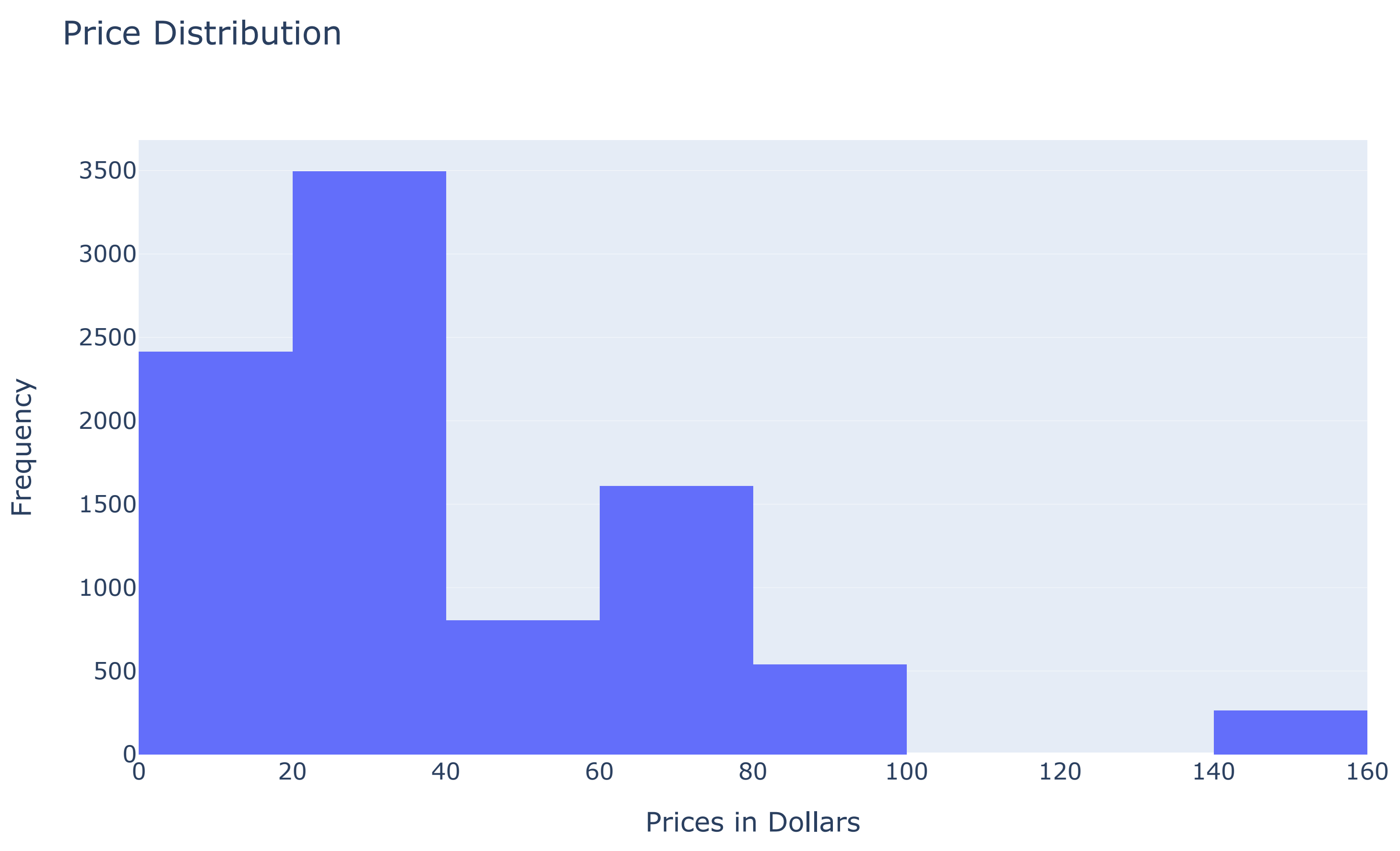
Out[22]:

	prices in dollars	Samples in total
count	9146.000000	9146.000000
mean	43.931765	4.558824
std	29.229448	2.788692
min	12.990000	1.000000
25%	19.990000	3.000000
50%	37.490000	4.000000
75%	64.990000	6.000000
max	149.000000	10.000000

>The prices of the products range from 12.99 to 149.00 dollars.

>The 25th percentile of prices is $19.99, meaning 25% of the products have prices below this value.

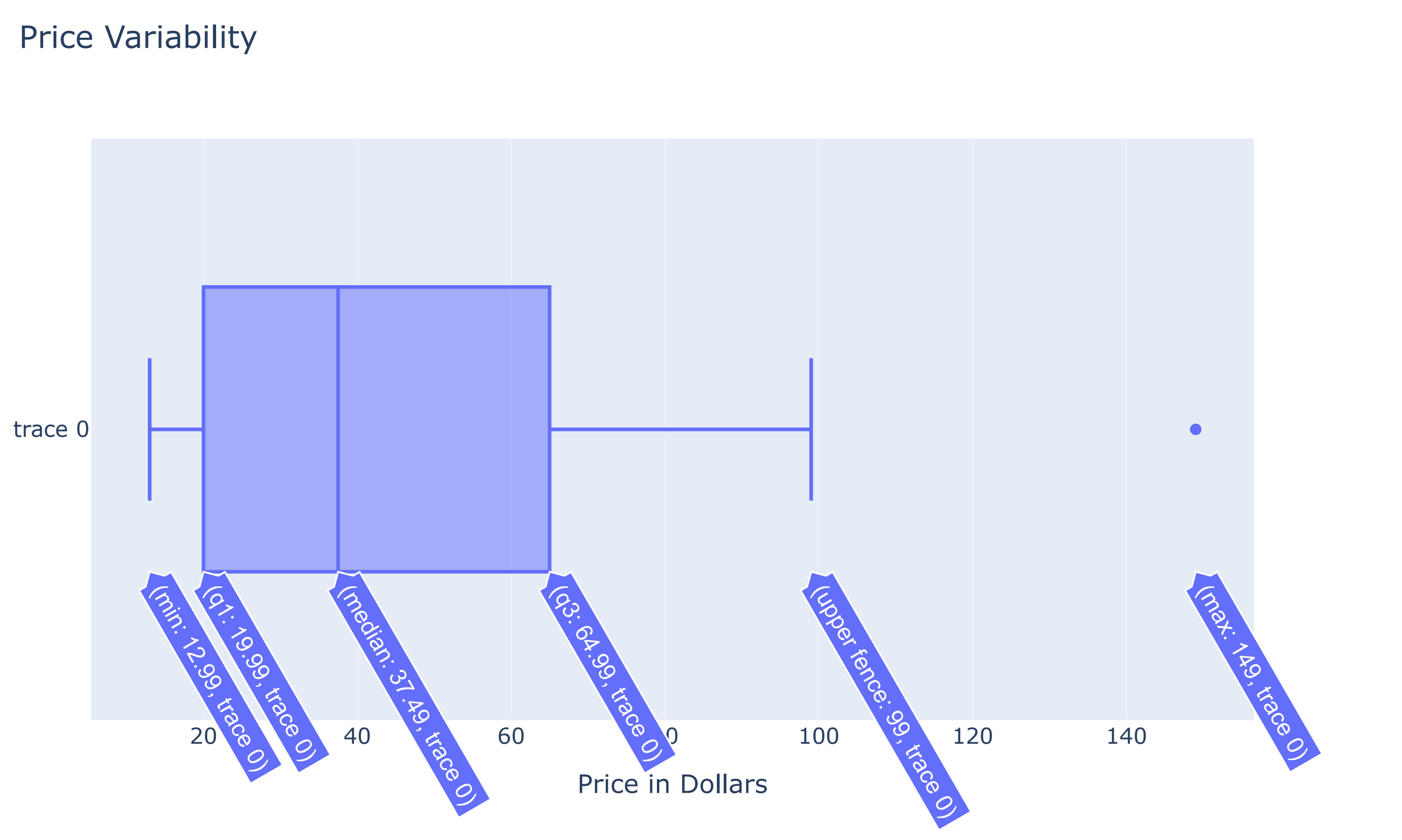
In [42]:
```python
# Create the histogram to visualize the distribution
histogram = go.Histogram(x=df['prices in dollars'], nbinsx=10)

layout_histogram = go.Layout(
    title='Price Distribution',
    xaxis=dict(title='Prices in Dollars'),  # Add x-axis label
    yaxis=dict(title='Frequency'),  # Add y-axis label
    height=500,  # Adjust the height as desired
    width=800  # Adjust the width as desired
)

figure_histogram = go.Figure(data=[histogram], layout=layout_histogram)
figure_histogram.show()
```



The majority of products fall in the price range of around 15 to 80 dollars.

There is a right-skew in the distribution, indicating that there are relatively fewer products with higher prices.

In [39]:
```
# Create the box plot to visualize the data variability
box_plot = go.Box(x=df['prices in dollars'], orientation='h')

layout_box_plot = go.Layout(
    title='Price Variability',
    xaxis=dict(title='Price in Dollars'),  # Add x-axis label
    height=500,  # Adjust the height as desired
    width=800  # Adjust the width as desired
)

figure_box_plot = go.Figure(data=[box_plot], layout=layout_box_plot)
figure_box_plot.show()

```



In [25]:
```
# Calculate descriptive statistics
price_stats = df['prices in dollars'].describe()
sales_stats = df['Samples in total'].describe()

price_stats
```

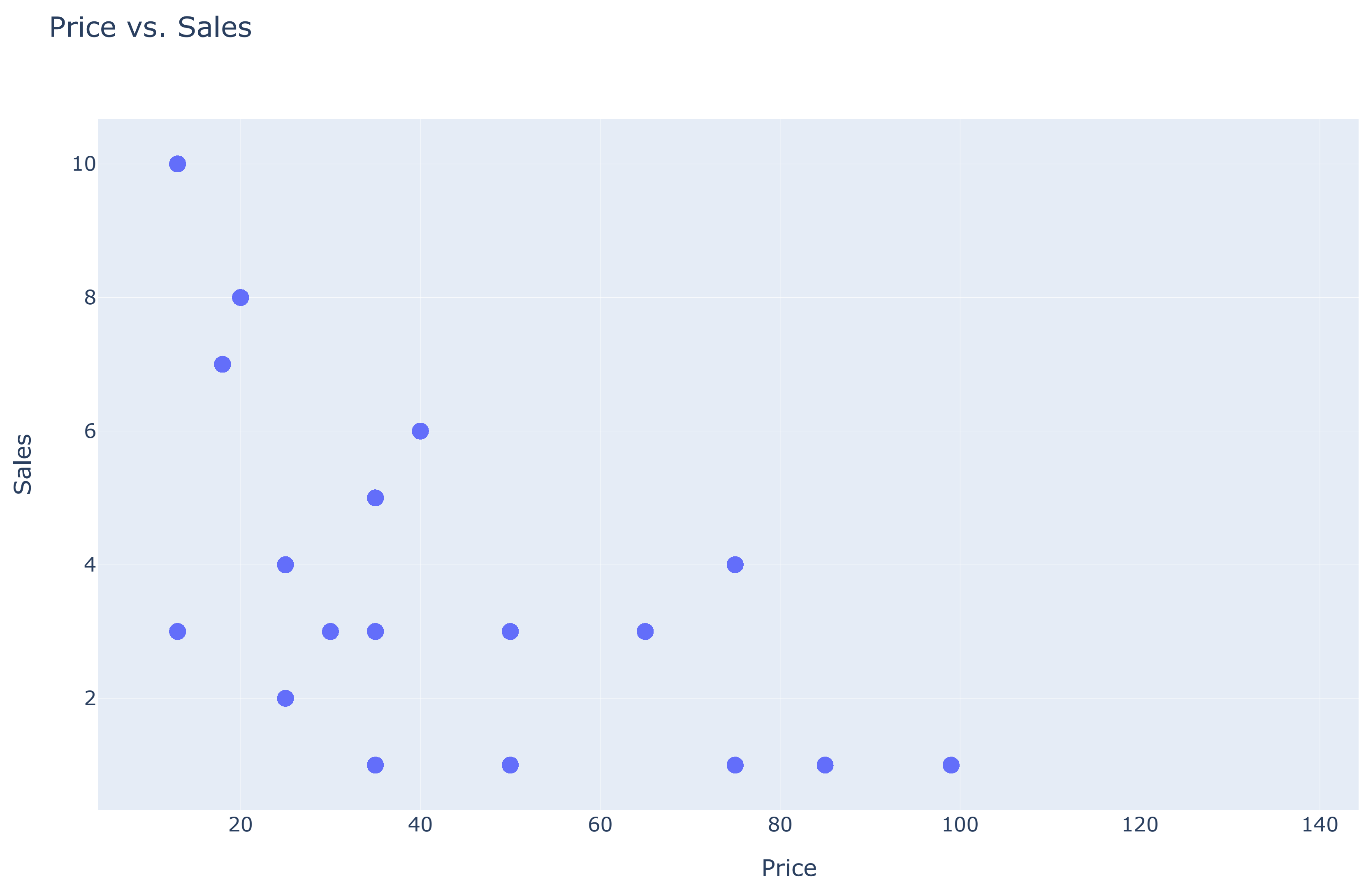
Out[25]:
```
count    9146.000000
mean       43.931765
std        29.229448
min        12.990000
25%        19.990000
50%        37.490000
75%        64.990000
max       149.000000
Name: prices in dollars, dtype: float64
```

In [26]:
```
sales_stats
```

Out[26]:
```
count    9146.000000
mean        4.558824
std         2.788692
min         1.000000
25%         3.000000
50%         4.000000
75%         6.000000
max        10.000000
Name: Samples in total, dtype: float64
```

In [36]:

```python
# Create the scatter plot of price vs. sales
scatter = go.Scatter(
    x=df['prices in dollars'],
    y=df['Samples in total'],
    mode='markers',
    marker=dict(size=10),
)

layout_scatter = go.Layout(
    title='Price vs. Sales',
    xaxis=dict(title='Price'),
    yaxis=dict(title='Sales'),
    height=600,  # Adjust the height as desired
    width=990  # Adjust the width as desired
)

figure_scatter = go.Figure(data=[scatter], layout=layout_scatter)
figure_scatter.show()
```



THANK YOU