jupyter e_commerce_project Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

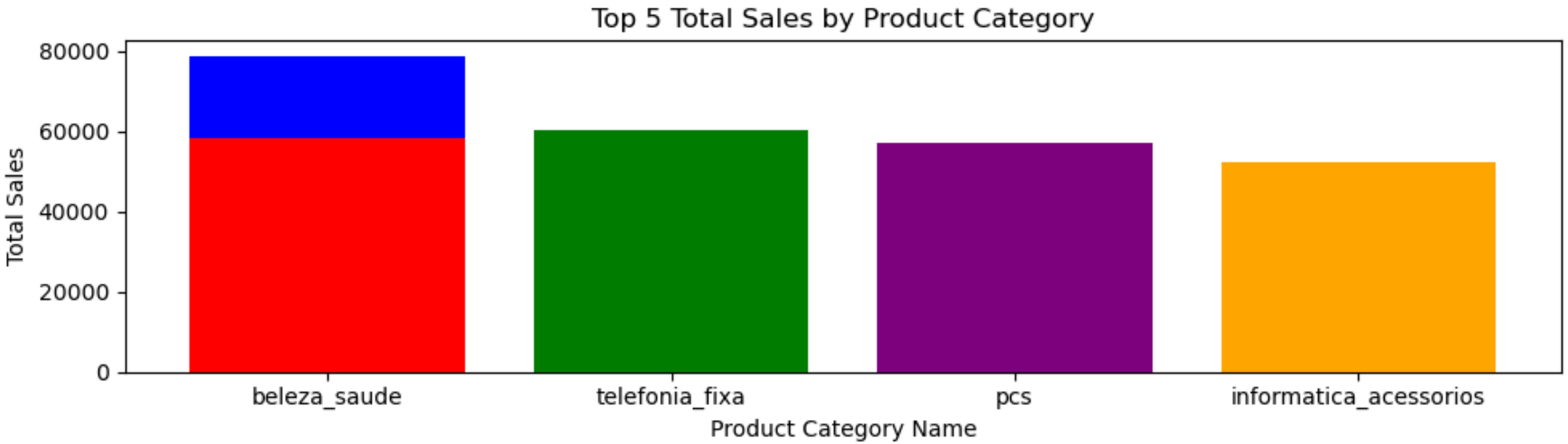
fact 1/1

DATA VUSUALIZATION

line plot ¶

[156]:

```
import matplotlib.pyplot as plt
import os  # Add this import
fact_orders_info['total_sales'] = fact_orders_info['price'] * fact_orders_info['order_item_id']
total_sales_by_product = fact_orders_info.groupby('product_id')['total_sales'].sum().reset_index()
print(dim_products.columns)  # Ensure 'product_category_name' is a valid column
total_sales_by_product = pd.merge(total_sales_by_product,
                                  dim_products[['product_id', 'product_category_name']],
                                  on='product_id',
                                  how='left')
total_sales_by_product['product_category_name'] = total_sales_by_product['product_category_name'].fillna('Unknown')
total_sales_by_product['product_category_name'] = total_sales_by_product['product_category_name'].astype(str)
top_5_sales = total_sales_by_product.sort_values(by='total_sales', ascending=False).head(5)
colors = ['blue', 'green', 'red', 'purple', 'orange']
plt.figure(figsize=(10, 3))
plt.bar(top_5_sales['product_category_name'], top_5_sales['total_sales'], color=colors)
plt.title('Top 5 Total Sales by Product Category')
plt.ylabel('Total Sales')
plt.xlabel('Product Category Name')
plt.tight_layout()
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop", "top_5_sales.png")
plt.savefig(desktop_path)
plt.show()
```

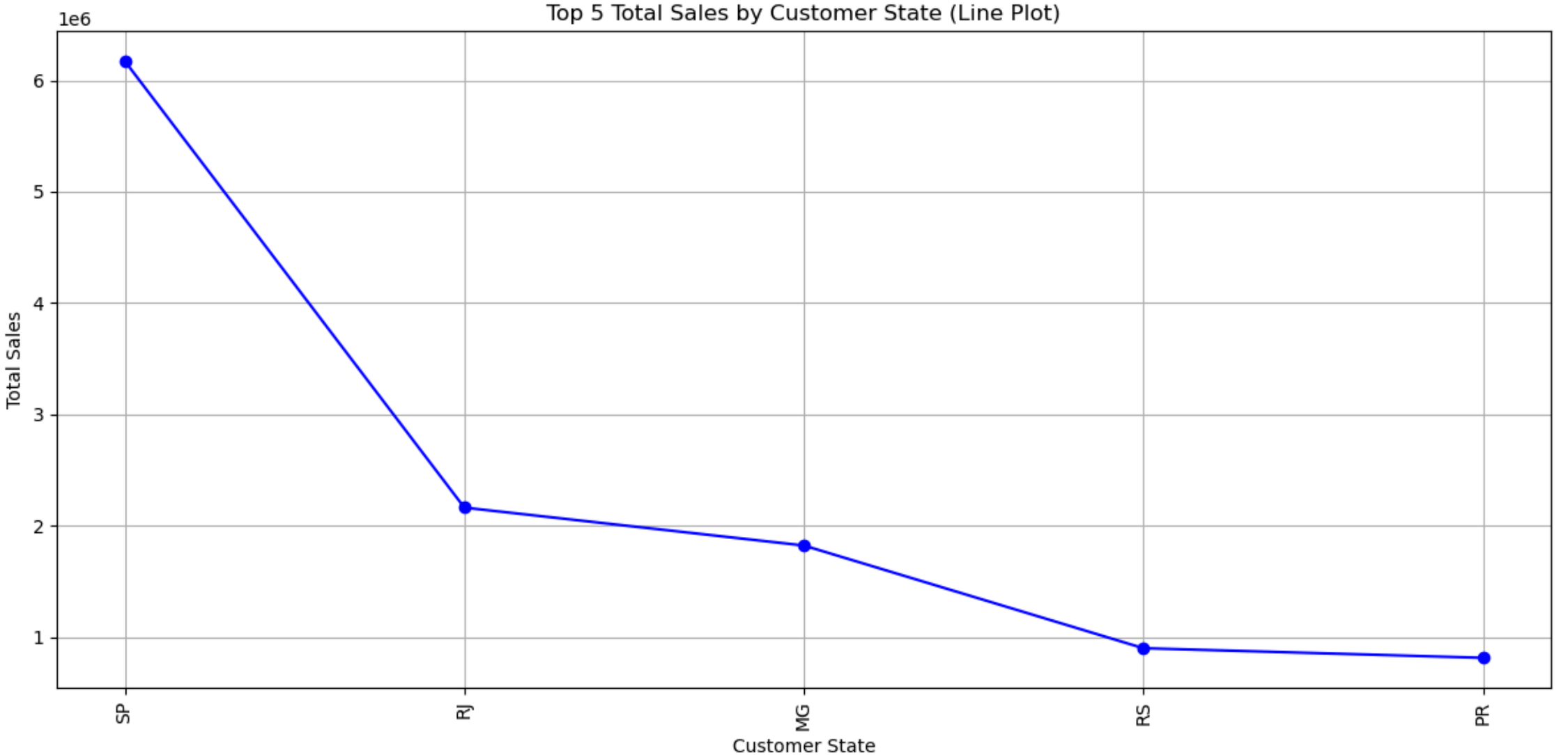

Top 5 Total Sales by Product Category
Total Sales
80000
60000
40000
20000
0
beleza_saude
telefonia_fixa
pcs
informatica_acessorios
Product Category Name

Product Category Name

fact 1/

[157]:

```
import pandas as pd
import matplotlib.pyplot as plt
import os
fact_orders_info['total_sales'] = fact_orders_info['price'] * fact_orders_info['order_item_id']
merged_data = pd.merge(
    fact_orders_info,
    dim_customers[['customer_id', 'customer_state']],
    on='customer_id',
    how='left',
    suffixes=('', '_dup')
)
for col in merged_data.columns:
    if '_dup' in col:
        merged_data.drop(col, axis=1, inplace=True)
total_sales_by_state = merged_data.groupby('customer_state')['total_sales'].sum().reset_index()
top_5_states = total_sales_by_state.sort_values(by='total_sales', ascending=False).head(5)
plt.figure(figsize=(12, 6))
plt.plot(top_5_states['customer_state'], top_5_states['total_sales'], marker='o', color='blue')
plt.xticks(rotation=90)
plt.title('Top 5 Total Sales by Customer State (Line Plot)')
plt.ylabel('Total Sales')
plt.xlabel('Customer State')
plt.grid(True)
plt.tight_layout()
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop", "top_5_sales_by_state.png")
plt.savefig(desktop_path)
plt.show()
```


Top 5 Total Sales by Customer State (Line Plot)
1e6
6
5
4
3
2
1
Total Sales
SP
RJ
MG
RS
PR
Customer State

Edit View Run Kernel Settings Help

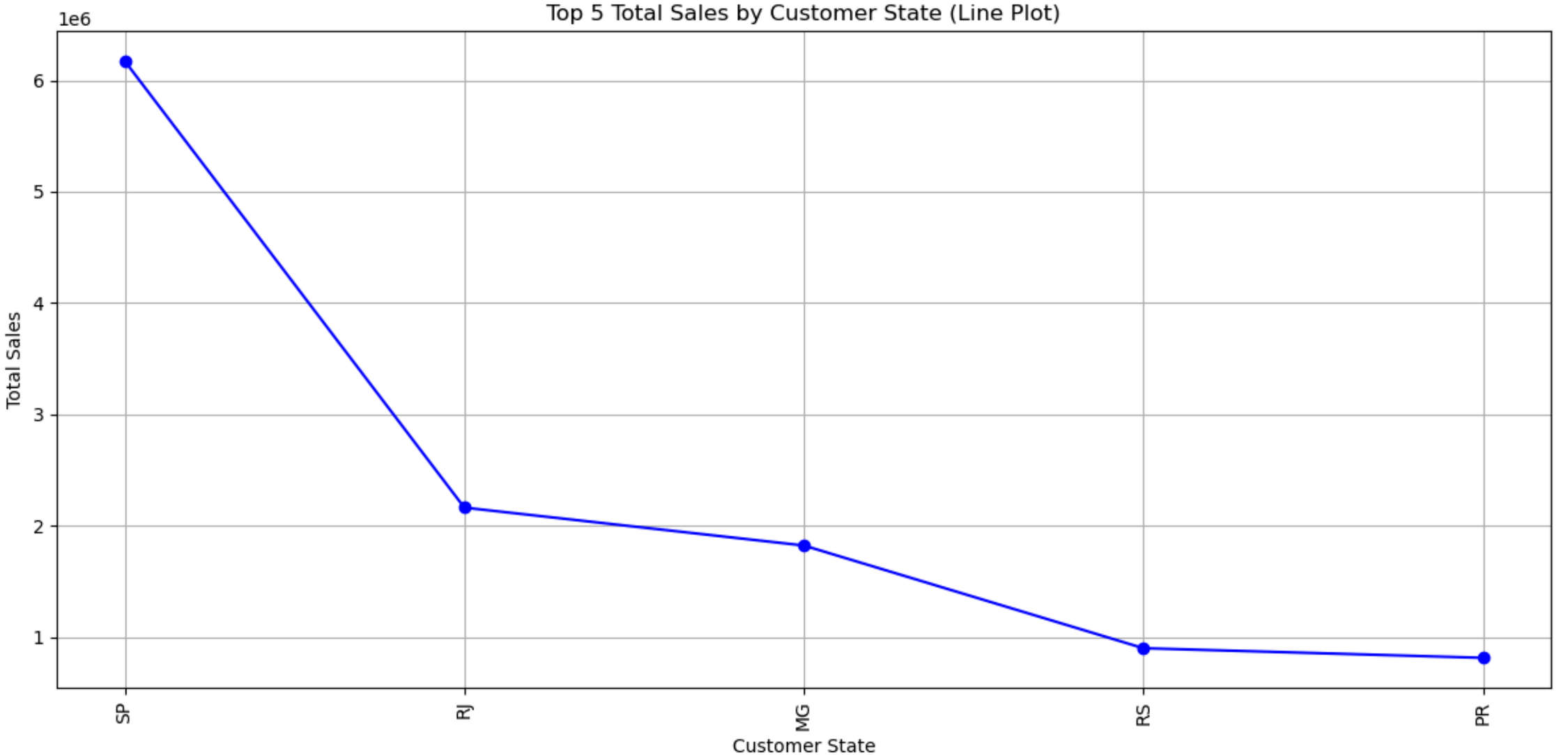
Code

JupyterLab Python 3 (ipy

fact

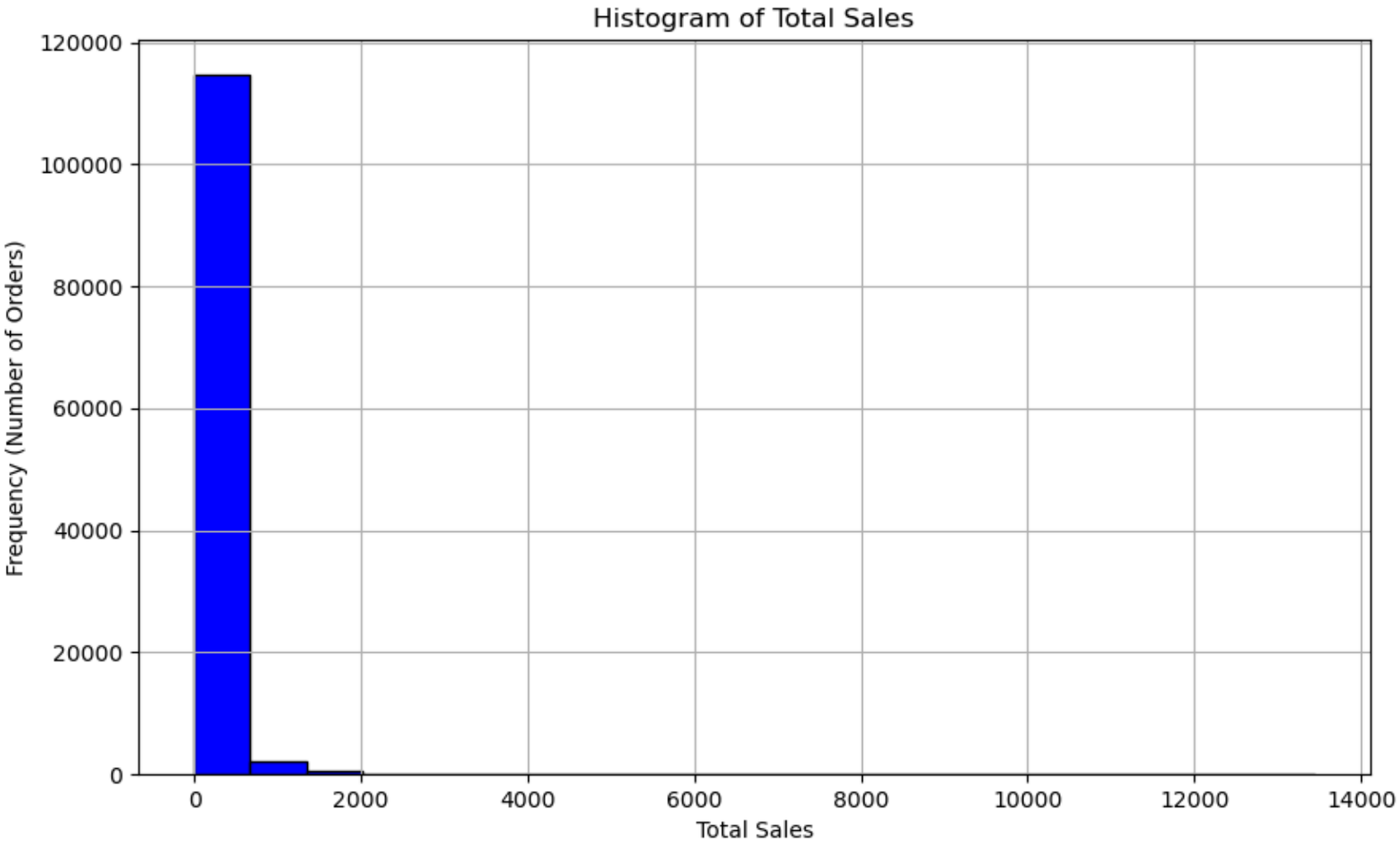
[ ]:

[157]:
```
import pandas as pd
import matplotlib.pyplot as plt
import os
fact_orders_info['total_sales'] = fact_orders_info['price'] * fact_orders_info['order_item_id']
merged_data = pd.merge(
    fact_orders_info,
    dim_customers[['customer_id', 'customer_state']],
    on='customer_id',
    how='left',
    suffixes=('', '_dup')
)
for col in merged_data.columns:
    if '_dup' in col:
        merged_data.drop(col, axis=1, inplace=True)
total_sales_by_state = merged_data.groupby('customer_state')['total_sales'].sum().reset_index()
top_5_states = total_sales_by_state.sort_values(by='total_sales', ascending=False).head(5)
plt.figure(figsize=(12, 6))
plt.plot(top_5_states['customer_state'], top_5_states['total_sales'], marker='o', color='blue')
plt.xticks(rotation=90)
plt.title('Top 5 Total Sales by Customer State (Line Plot)')
plt.ylabel('Total Sales')
plt.xlabel('Customer State')
plt.grid(True)
plt.tight_layout()
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop", "top_5_sales_by_state.png")
plt.savefig(desktop_path)
plt.show()
```


Top 5 Total Sales by Customer State (Line Plot)
1e6
6
5
4
3
2
1
Total Sales
SP
RJ
MG
RS
PR
Customer State

[ ]:

[158]:
```
import pandas as pd
import matplotlib.pyplot as plt
import os
fact_orders_info['total_sales'] = fact_orders_info['price'] * fact_orders_info['order_item_id']
total_sales_data = fact_orders_info['total_sales']
plt.figure(figsize=(10, 6))
plt.hist(total_sales_data, bins=20, color='blue', edgecolor='black')
plt.title('Histogram of Total Sales')
plt.xlabel('Total Sales')
plt.ylabel('Frequency (Number of Orders)')
plt.grid(True)
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop", "total_sales_histogram.png")
plt.savefig(desktop_path)  # Save the histogram to the desktop
plt.tight_layout()
plt.show()
```


Histogram of Total Sales
Frequency (Number of Orders)
120000
100000
80000
60000
40000
20000
0
0
2000
4000
6000
8000
10000
12000
14000
Total Sales

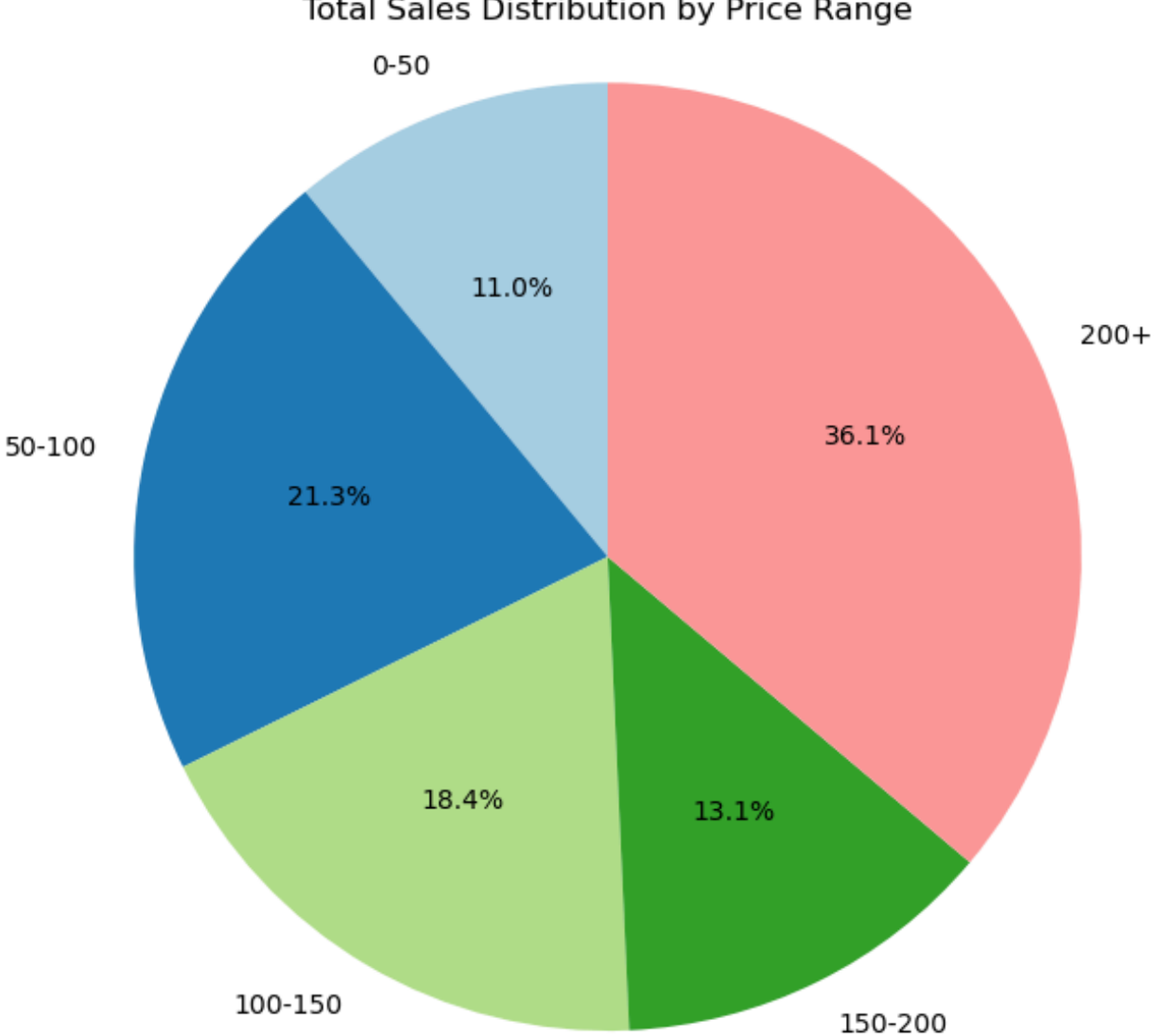
bs Settings Help

Markdown Notebook Python 3 (ipykernel)

0 2000 4000 6000 8000 10000 12000 14000

Total Sales

[159]:
```
import pandas as pd
import matplotlib.pyplot as plt
import os
fact_orders_info['total_sales'] = fact_orders_info['price'] * fact_orders_info['order_item_id']
bins = [0, 50, 100, 150, 200, 1000]
labels = ['0-50', '50-100', '100-150', '150-200', '200+']
fact_orders_info['price_range'] = pd.cut(fact_orders_info['price'], bins=bins, labels=labels, include_lowest=Tru
sales_by_price_range = fact_orders_info.groupby('price_range', observed=False)['total_sales'].sum()
plt.figure(figsize=(10, 6))
plt.pie(sales_by_price_range, labels=sales_by_price_range.index, autopct='%1.1f%%', startangle=90, colors=plt.cm
plt.title('Total Sales Distribution by Price Range')
plt.axis('equal')
plt.tight_layout()
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop", "total_sales_by_price_range.png")
plt.savefig(desktop_path)
plt.show()
```

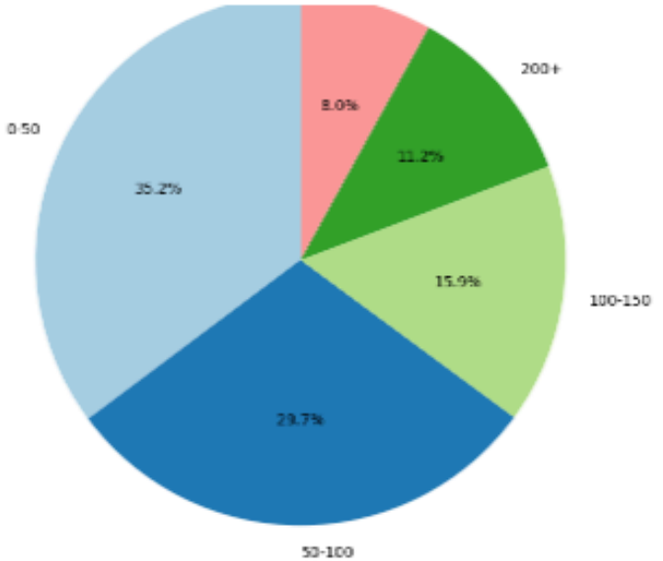

Total Sales Distribution by Price Range
0-50
11.0%
200+
36.1%
50-100
21.3%
18.4%
13.1%
100-150
150-200


```
import pandas as pd
import matplotlib.pyplot as plt
import os
def plot_pie_chart(data, column, title, bins=None, labels=None, save_path=None):
    if bins and labels:
        data['binned'] = pd.cut(data[column], bins=bins, labels=labels, include_lowest=True)
        data_to_plot = data['binned'].value_counts()
    else:
        data_to_plot = data[column].value_counts()
    plt.figure(figsize=(10, 6))
    plt.pie(data_to_plot, labels=data_to_plot.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Paired.colors)
    plt.title(title)
    plt.axis('equal')
    plt.tight_layout()
    if save_path:
        plt.savefig(save_path)

    plt.show()
desktop_path = os.path.expanduser("~") + "/Desktop"
if 'product_category_name' in fact_orders_info.columns:
    plot_pie_chart(
        fact_orders_info,
        'product_category_name',
        'Total Orders by Product Category',
        save_path=os.path.join(desktop_path, "total_orders_by_product_category.png")
    )
bins = [0, 50, 100, 150, 200, 1000]
labels = ['0-50', '50-100', '100-150', '150-200', '200+']
if 'price' in fact_orders_info.columns:
    plot_pie_chart(
        fact_orders_info,
        'price',
        'Price Distribution',
        bins=bins,
        labels=labels,
        save_path=os.path.join(desktop_path, "price_distribution.png")
    )
bins_sales = [0, 100, 500, 1000, 5000, 10000]
labels_sales = ['0-100', '100-500', '500-1000', '1000-5000', '5000+']
if 'total_sales' in fact_orders_info.columns:
    plot_pie_chart(
        fact_orders_info,
        'total_sales',
        'Total Sales Distribution',
        bins=bins_sales,
        labels=labels_sales,
        save_path=os.path.join(desktop_path, "total_sales_distribution.png")
    )
```

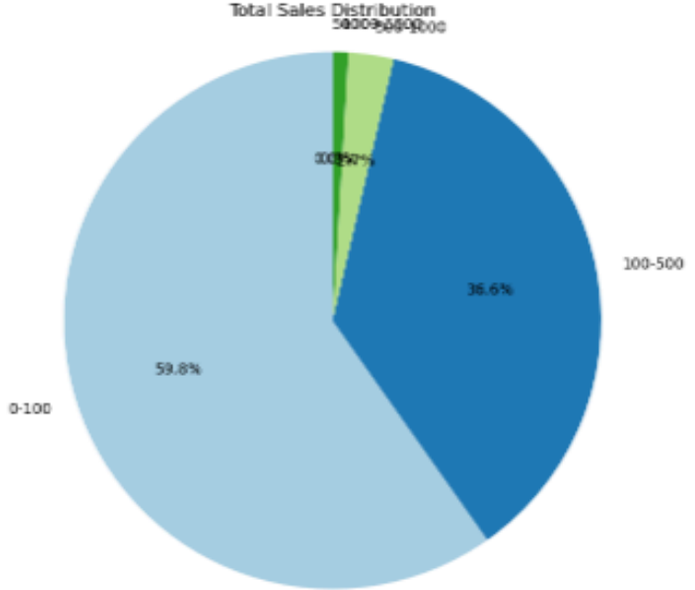
Price Distribution

tings Help

Markdown

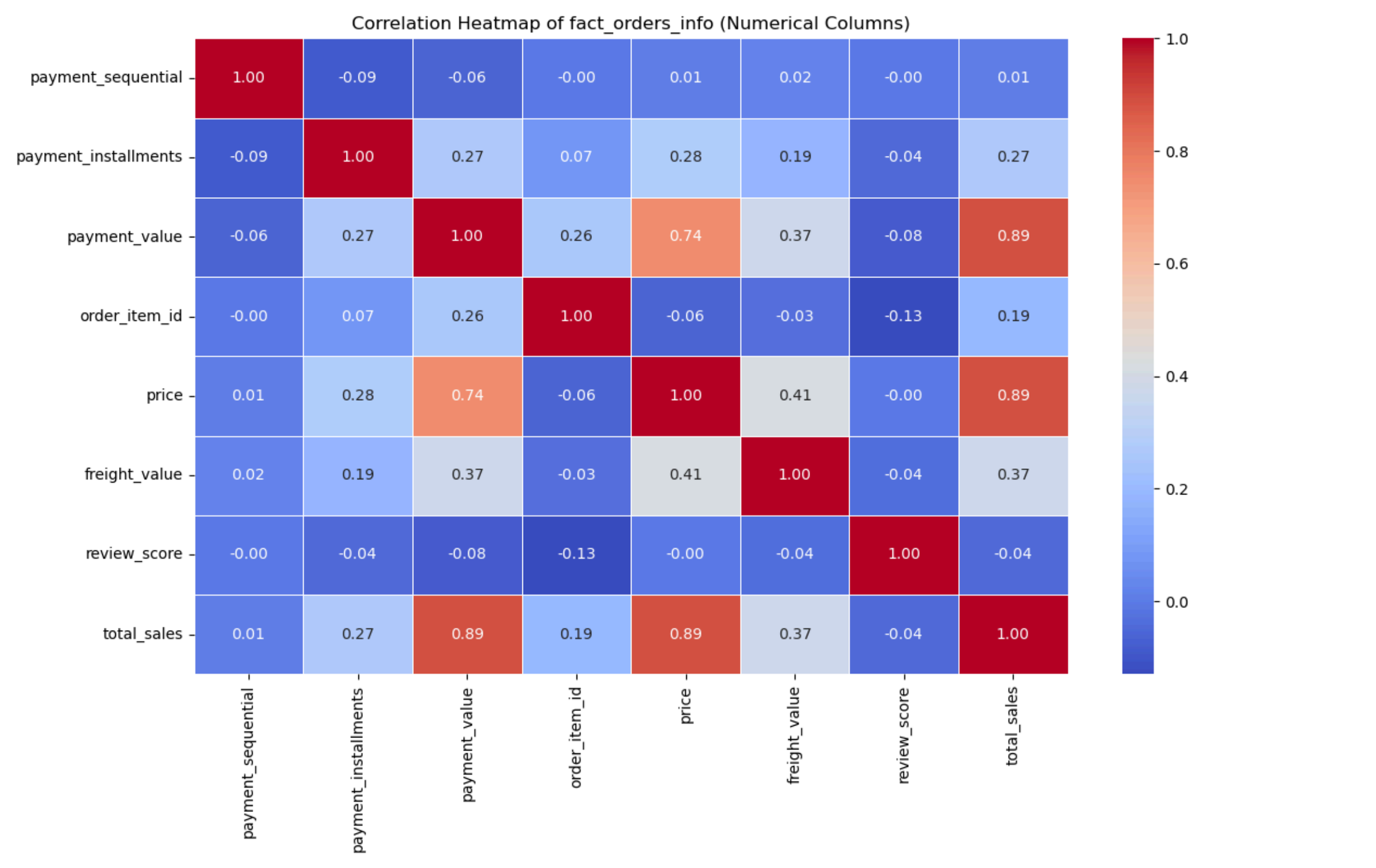


Total Sales Distribution



[161]:

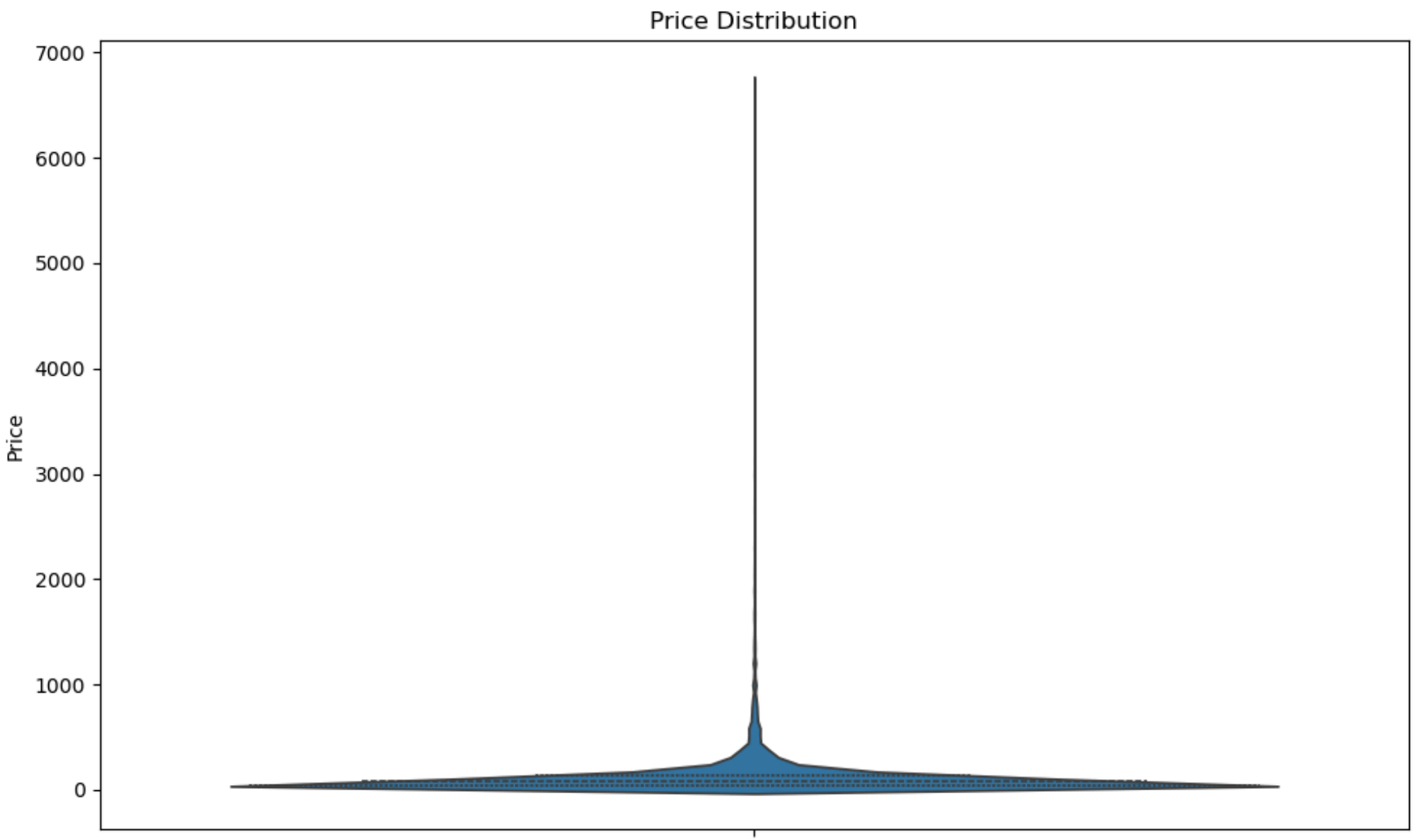
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Select numerical columns from fact_orders_info
fact_orders_numerical = fact_orders_info.select_dtypes(include=['float64', 'int64'])

# Calculate the correlation matrix
correlation_matrix = fact_orders_numerical.corr()

# Plot the correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap of fact_orders_info (Numerical Columns)')
plt.tight_layout()

# Define path to save plot to the desktop
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop", "fact_orders_correlation_heatmap.png")
plt.savefig(desktop_path)  # Save the heatmap to the desktop

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

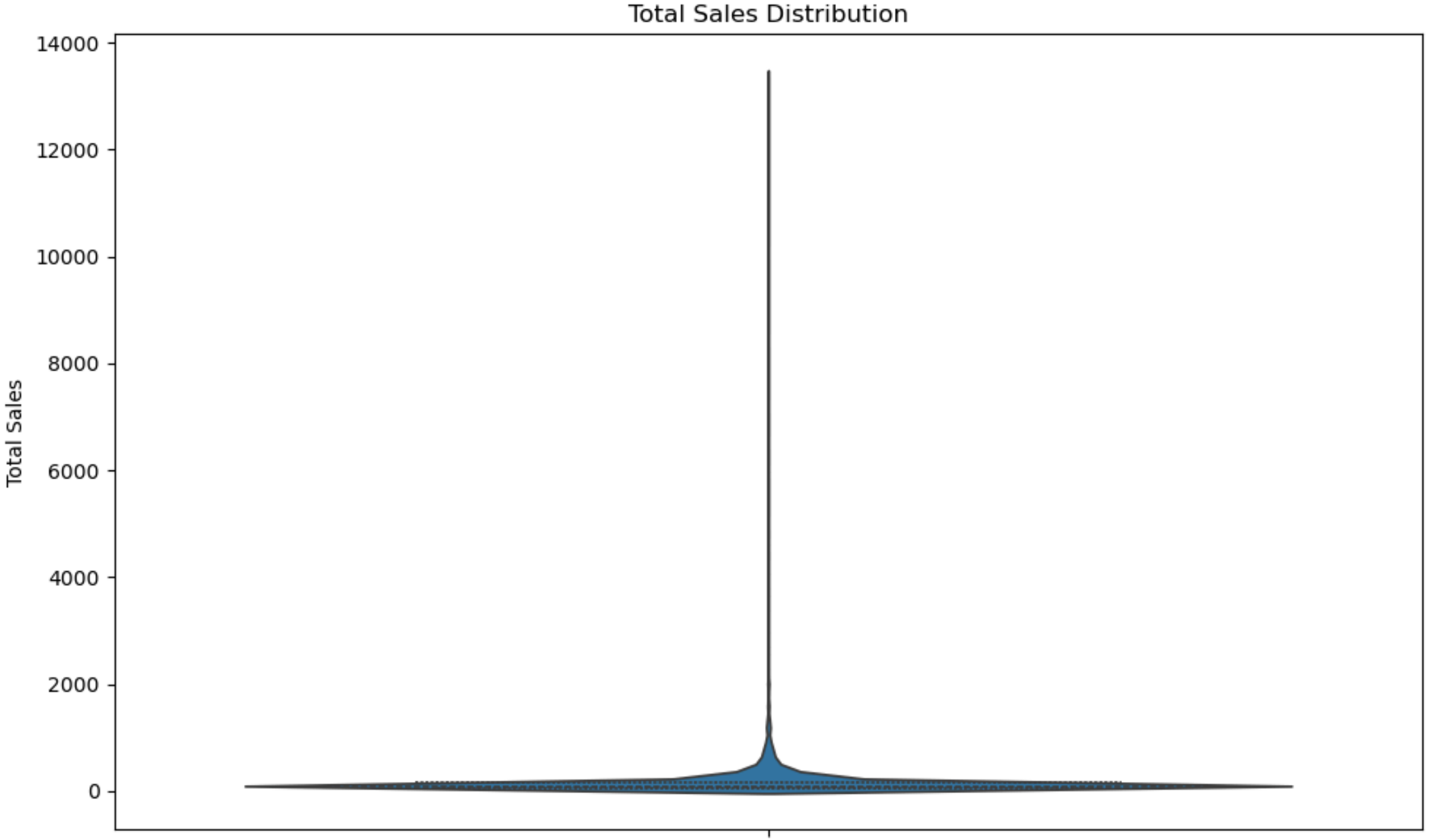

Correlation Heatmap of fact_orders_info (Numerical Columns)
payment_sequential
payment_installments
payment_value
order_item_id
price
freight_value
review_score
total_sales
1.00 -0.09 -0.06 -0.00 0.01 0.02 -0.00 0.01
-0.09 1.00 0.27 0.07 0.28 0.19 -0.04 0.27
-0.06 0.27 1.00 0.26 0.74 0.37 -0.08 0.89
-0.00 0.07 0.26 1.00 -0.06 -0.03 -0.13 0.19
0.01 0.28 0.74 -0.06 1.00 0.41 -0.00 0.89
0.02 0.19 0.37 -0.03 0.41 1.00 -0.04 0.37
-0.00 -0.04 -0.08 -0.13 -0.00 -0.04 1.00 -0.04
0.01 0.27 0.89 0.19 0.89 0.37 -0.04 1.00
1.0
0.8
0.6
0.4
0.2
0.0

Code

[163]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

def plot_violin_chart(data, column, title, x_label=None, y_label=None, save_path=None):
    plt.figure(figsize=(10, 6))
    sns.violinplot(data=data, y=column, inner="quartile")
    plt.title(title)
    plt.xlabel(x_label if x_label else '')
    plt.ylabel(y_label if y_label else column)
    plt.tight_layout()

    if save_path:
        plt.savefig(save_path)
    plt.show()

# Define the desktop path
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")

# Violin plot for product price
if 'price' in fact_orders_info.columns:
    plot_violin_chart(
        fact_orders_info, 'price', 'Price Distribution', y_label='Price',
        save_path=os.path.join(desktop_path, "price_distribution.png")
    )

# Violin plot for total sales
if 'total_sales' in fact_orders_info.columns:
    plot_violin_chart(
        fact_orders_info, 'total_sales', 'Total Sales Distribution', y_label='Total Sales',
        save_path=os.path.join(desktop_path, "total_sales_distribution.png")
    )
```

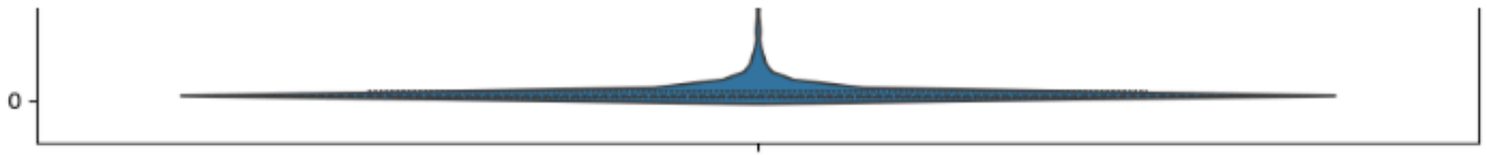

Price Distribution
Price
7000
6000
5000
4000
3000
2000
1000
0


Total Sales Distribution
Total Sales
14000
12000
10000
8000
6000
4000
2000
0

Settings Help

Code

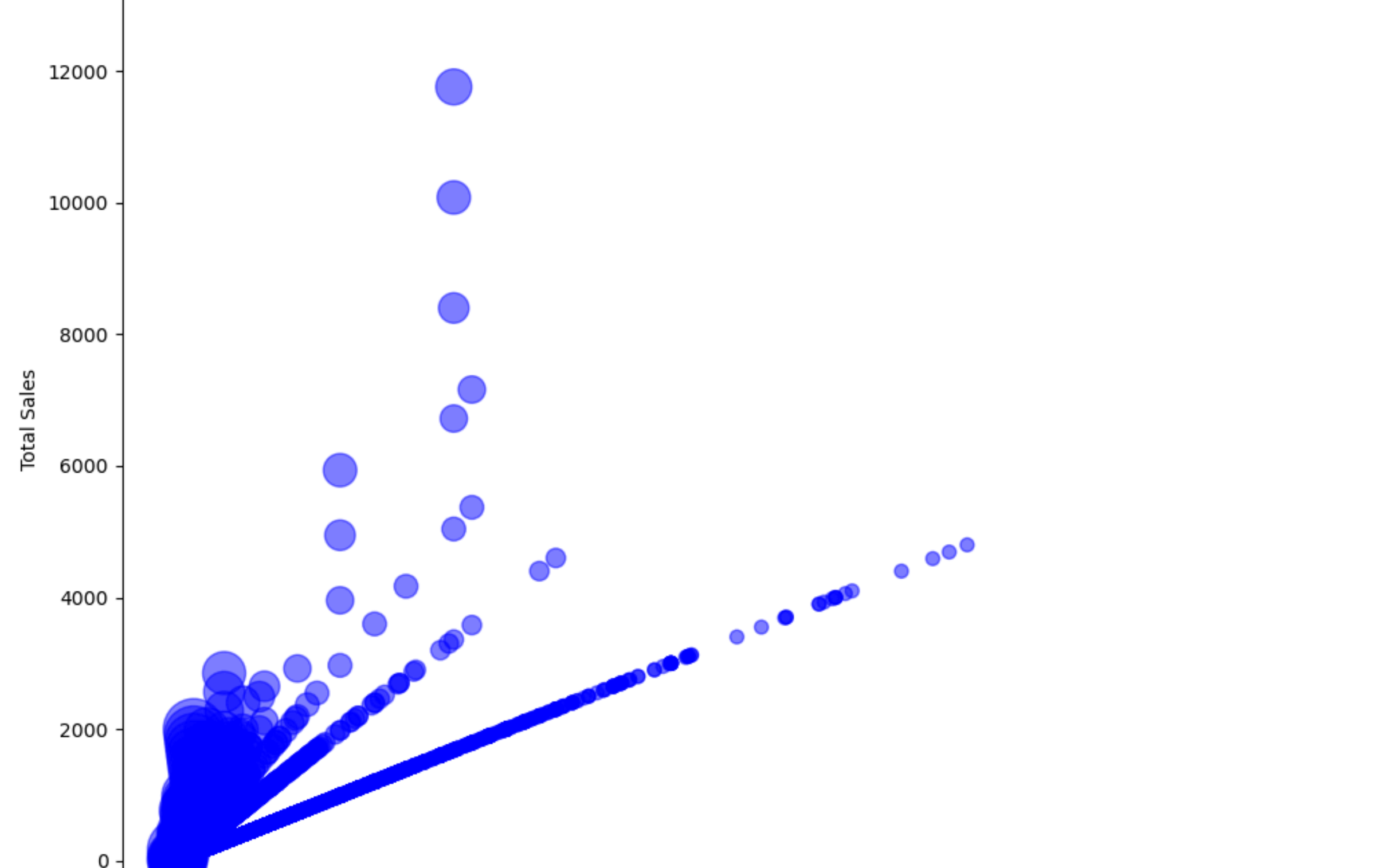
Notebook Python 3 (ipykernel)



[164]:

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

def plot_bubble_chart(data, x_column, y_column, size_column, title, x_label=None, y_label=None, size_factor=100, save_path=None):
    plt.figure(figsize=(10, 8))

    plt.scatter(data[x_column], data[y_column],
                s=data[size_column] * size_factor,
                alpha=0.5, color='blue')
        plt.title(title)
    plt.xlabel(x_label if x_label else x_column)
    plt.ylabel(y_label if y_label else y_column)

    plt.tight_layout()

    if save_path:
        plt.savefig(save_path)
    plt.show()

desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")

if 'price' in fact_orders_info.columns and 'total_sales' in fact_orders_info.columns:
    plot_bubble_chart(fact_orders_info,
                      x_column='price',
                      y_column='total_sales',
                      size_column='order_item_id',
                      title='Bubble Chart: Price vs Total Sales',
                      x_label='Price',
                      y_label='Total Sales',
                      size_factor=50,
                      save_path=os.path.join(desktop_path, "price_vs_total_sales_bubble_chart.png"))
```

Bubble Chart: Price vs Total Sales


Total Sales
12000
10000
8000
6000
4000
2000
0

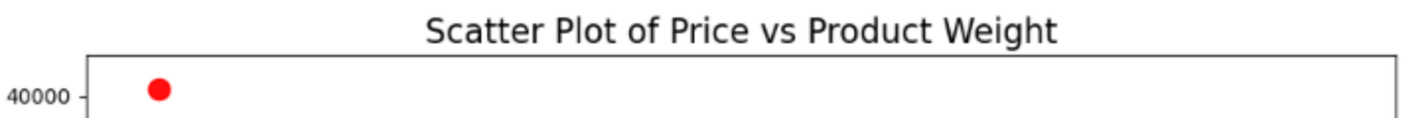
Tabs Settings Help

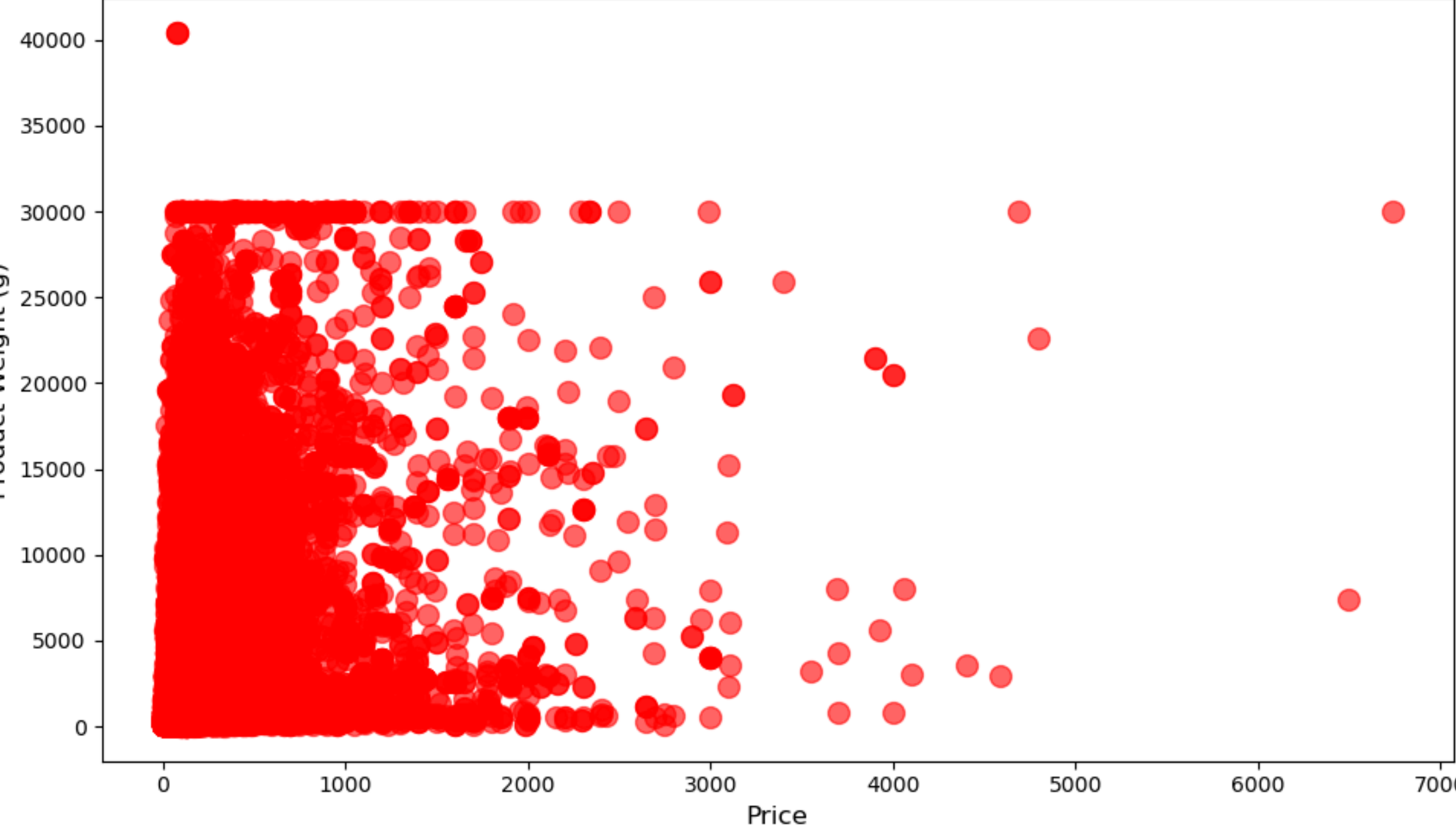
Code

Notebook Python 3 (ipykernel)

[165]:

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

# Merge data
merged_data = pd.merge(fact_orders_info, dim_products, on='product_id')

# Plot settings
plt.figure(figsize=(10, 6))
plt.scatter(merged_data['price'], merged_data['product_weight_g'],
            color='red', alpha=0.6, s=100)
plt.title('Scatter Plot of Price vs Product Weight', fontsize=16)
plt.xlabel('Price', fontsize=12)
plt.ylabel('Product Weight (g)', fontsize=12)
plt.tight_layout()

# Define the desktop path and save the plot
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")
plt.savefig(os.path.join(desktop_path, "price_vs_product_weight_scatter.png"))
plt.show()
```




40000
35000
30000
25000
20000
15000
10000
5000
0
0
1000
2000
3000
4000
5000
6000
Price

166]:
```
import pandas as pd
import matplotlib.pyplot as plt
import os
merged_data = pd.merge(fact_orders_info, dim_products, on='product_id')

plt.figure(figsize=(10, 6))
plt.scatter(merged_data['price'], merged_data['product_weight_g'],
            color='red', alpha=0.6, s=100)
plt.title('Scatter Plot of Price vs Product Weight', fontsize=16)
plt.xlabel('Price', fontsize=12)
plt.ylabel('Product Weight (g)', fontsize=12)

for i, txt in enumerate(merged_data['product_id']):  # Assuming product_id for notations
    plt.annotate('*', (merged_data['price'].iloc[i], merged_data['product_weight_g'].iloc[i]),
                 fontsize=12, color='blue')
plt.tight_layout()
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")
plt.savefig(os.path.join(desktop_path, "price_vs_product_weight_scatter_annotated.png"))
```


Scatter Plot of Price vs Product Weight
000
000
000
000
000
000
000
0
0
1000
2000
3000
4000
5000
6000
Price