Sales Analysis Report

September 15, 2023

0.1 Objective: Data Manipulation and Exploring/Visualize using Pandas, matplotlib, seaborn libs

0.1.1 Import libs

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

[ ]:
```
# Suppress all warnings
warnings.filterwarnings("ignore")
```

0.1.2 Import Sales Data Set

[ ]:
```
df = pd.read_excel('superstore_sales.xlsx',parse_dates=True,sheet_name='Orders')
df
```

[ ]:
```
              order_id order_date  ship_date       ship_mode  \
0         AG-2011-2040 2019-03-20 2019-03-25  Standard Class
1        IN-2011-47883 2019-03-20 2019-03-27  Standard Class
2         HU-2011-1220 2019-03-20 2019-03-24    Second Class
3      IT-2011-3647632 2019-03-20 2019-03-24    Second Class
4        IN-2011-47883 2019-03-20 2019-03-27  Standard Class
…                  …          …          …              …
51285   CA-2014-115427 2023-03-19 2023-03-23  Standard Class
51286     MO-2014-2560 2023-03-19 2023-03-24  Standard Class
51287   MX-2014-110527 2023-03-19 2023-03-21    Second Class
51288   MX-2014-114783 2023-03-19 2023-03-25  Standard Class
51289   CA-2014-156720 2023-03-19 2023-03-23  Standard Class

          customer_name      segment            state        country  market  \
0       Toby Braunhardt     Consumer       Constantine        Algeria  Africa
1           Joseph Holt     Consumer   New South Wales      Australia    APAC
2         Annie Thurman     Consumer          Budapest        Hungary    EMEA
3          Eugene Moren  Home Office         Stockholm         Sweden      EU
4           Joseph Holt     Consumer   New South Wales      Australia    APAC
…                   …          …                …            …     …
```

```
51285        Erica Bern    Corporate        California  United States      US
51286         Liz Preis     Consumer  Souss-Massa-Draâ        Morocco  Africa
51287  Charlotte Melton     Consumer           Managua      Nicaragua   LATAM
51288     Tamara Dahlen     Consumer         Chihuahua         Mexico   LATAM
51289     Jill Matthias     Consumer          Colorado  United States      US

        region  …         category sub_category  \
0       Africa  …  Office Supplies      Storage
1      Oceania  …  Office Supplies     Supplies
2         EMEA  …  Office Supplies      Storage
3        North  …  Office Supplies        Paper
4      Oceania  …        Furniture  Furnishings
…        …  …              …           …
51285     West  …  Office Supplies      Binders
51286   Africa  …  Office Supplies      Binders
51287  Central  …  Office Supplies       Labels
51288    North  …  Office Supplies       Labels
51289     West  …  Office Supplies    Fasteners

                                          product_name    sales  quantity  \
0                                Tenex Lockers, Blue  408.300         2
1                            Acme Trimmer, High Speed  120.366         3
2                             Tenex Box, Single Width   66.120         4
3                           Enermax Note Cards, Premium   44.865         3
4                            Eldon Light Bulb, Duo Pack  113.670         5
…                                                  …      …       …
51285  Cardinal Slant-D Ring Binder, Heavy Gauge Vinyl   13.904         2
51286          Wilson Jones Hole Reinforcements, Clear    3.990         1
51287           Hon Color Coded Labels, 5000 Label Set   26.400         3
51288           Hon Legal Exhibit Labels, Alphabetical    7.120         1
51289                               Bagged Rubber Bands    3.024         3

       discount    profit  shipping_cost  order_priority  year
0           0.0  106.1400         35.460          Medium  2019
1           0.1   36.0360          9.720          Medium  2019
2           0.0   29.6400          8.170            High  2019
3           0.5  -26.0550          4.820            High  2019
4           0.1   37.7700          4.700          Medium  2019
…         …       …             …              …   …
51285       0.2    4.5188          0.890          Medium  2023
51286       0.0    0.4200          0.490          Medium  2023
51287       0.0   12.3600          0.350          Medium  2023
51288       0.0    0.5600          0.199          Medium  2023
51289       0.2   -0.6048          0.170          Medium  2023

[51290 rows x 21 columns]
```

0.2 ETA

[ ]:
```
df.shape
```

[ ]:
```
(51290, 21)
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['order_id', 'order_date', 'ship_date', 'ship_mode', 'customer_name',
       'segment', 'state', 'country', 'market', 'region', 'product_id',
       'category', 'sub_category', 'product_name', 'sales', 'quantity',
       'discount', 'profit', 'shipping_cost', 'order_priority', 'year'],
      dtype='object')
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   order_id        51290 non-null  object
 1   order_date      51290 non-null  datetime64[ns]
 2   ship_date       51290 non-null  datetime64[ns]
 3   ship_mode       51290 non-null  object
 4   customer_name   51290 non-null  object
 5   segment         51290 non-null  object
 6   state           51290 non-null  object
 7   country         51290 non-null  object
 8   market          51290 non-null  object
 9   region          51290 non-null  object
 10  product_id      51290 non-null  object
 11  category        51290 non-null  object
 12  sub_category    51290 non-null  object
 13  product_name    51290 non-null  object
 14  sales           51290 non-null  float64
 15  quantity        51290 non-null  int64
 16  discount        51290 non-null  float64
 17  profit          51290 non-null  float64
 18  shipping_cost   51290 non-null  float64
 19  order_priority  51290 non-null  object
 20  year            51290 non-null  int64
dtypes: datetime64[ns](2), float64(4), int64(2), object(13)
memory usage: 8.2+ MB
```

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

[ ]:
```
                          order_date                       ship_date  \
count                          51290                           51290
mean   2021-07-28 21:26:49.155780352  2021-08-01 20:42:42.745174528
min              2019-03-20 00:00:00            2019-03-22 00:00:00
25%              2020-09-05 00:00:00            2020-09-09 00:00:00
50%              2021-09-24 00:00:00            2021-09-28 00:00:00
75%              2022-08-08 00:00:00            2022-08-12 00:00:00
max              2023-03-19 00:00:00            2023-03-26 00:00:00
std                              NaN                            NaN

               sales      quantity      discount        profit  shipping_cost  \
count   51290.000000  51290.000000  51290.000000  51290.000000   51290.000000
mean      246.490581      3.476545      0.142908     28.641740      26.375818
min         0.444000      1.000000      0.000000  -6599.978000       0.002000
25%        30.758625      2.000000      0.000000      0.000000       2.610000
50%        85.053000      3.000000      0.000000      9.240000       7.790000
75%       251.053200      5.000000      0.200000     36.810000      24.450000
max     22638.480000     14.000000      0.850000   8399.976000     933.570000
std       487.565361      2.278766      0.212280    174.424113      57.296810

                year
count   51290.000000
mean     2021.072821
min      2019.000000
25%      2020.000000
50%      2021.000000
75%      2022.000000
max      2023.000000
std         1.185047
```

0.2.1 Checking Null values

[ ]:
```
df.isnull().sum()
```

[ ]:
```
order_id          0
order_date        0
ship_date         0
ship_mode         0
customer_name     0
segment           0
state             0
country           0
market            0
region            0
product_id        0
category          0
sub_category      0
```

```
product_name      0
sales             0
quantity          0
discount          0
profit            0
shipping_cost     0
order_priority    0
year              0
dtype: int64
```

0.2.2 1. Which are the most selling products?

[ ]:
```
most_selling_product = df.groupby('product_name',as_index=False).
 ↪aggregate({'quantity':'sum'}).sort_values(by='quantity',ascending=False).
 ↪head(10)
most_selling_product.reset_index(drop=True,inplace=True)
most_selling_product
```

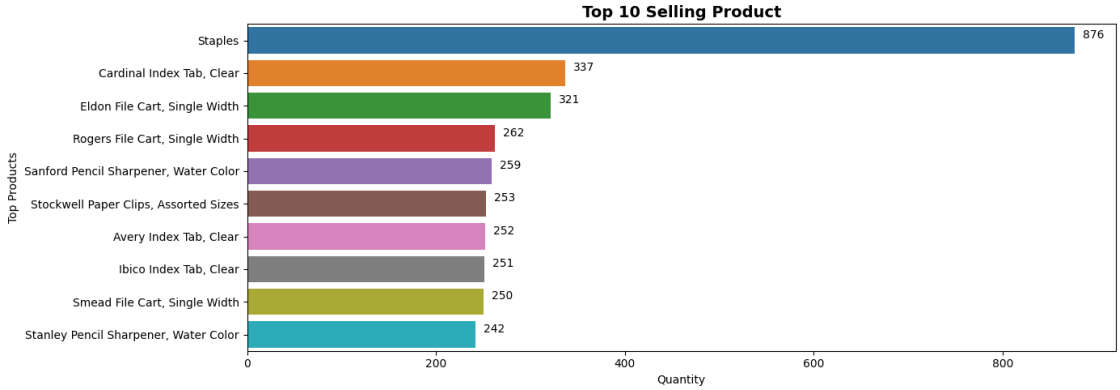
[ ]:

	product_name	quantity
0	Staples	876
1	Cardinal Index Tab, Clear	337
2	Eldon File Cart, Single Width	321
3	Rogers File Cart, Single Width	262
4	Sanford Pencil Sharpener, Water Color	259
5	Stockwell Paper Clips, Assorted Sizes	253
6	Avery Index Tab, Clear	252
7	Ibico Index Tab, Clear	251
8	Smead File Cart, Single Width	250
9	Stanley Pencil Sharpener, Water Color	242

[ ]:
```
fig,axs = plt.subplots(figsize=(14,5),dpi=100)
sns.
 ↪barplot(most_selling_product,y='product_name',x='quantity',estimator='sum',label='quantity'
axs.set(ylabel='Top Products',xlabel='Quantity')
axs.set_title(label='Top 10 Selling Product',fontsize=14,fontweight='bold')


# Add data labels
for p in axs.patches:
    axs.annotate(f'{p.get_width():.0f}', (p.get_width()+20, p.get_y() + p.
 ↪get_height() / 2),
                 ha='center', va='center', fontsize=10, color='black',␣
 ↪xytext=(0, 5),
                 textcoords='offset points')
```

```
plt.tight_layout(w_pad=200,h_pad=200)
plt.show()
```



0.2.3 2. Which are the Top 10 products by sales?

[ ]:
```
top_sold = df.groupby('product_name',as_index=False).aggregate({'sales':'sum'}).
 ↪sort_values(by='sales',ascending=False).head(10)
top_sold.reset_index(drop=True,inplace=True)
top_sold
```

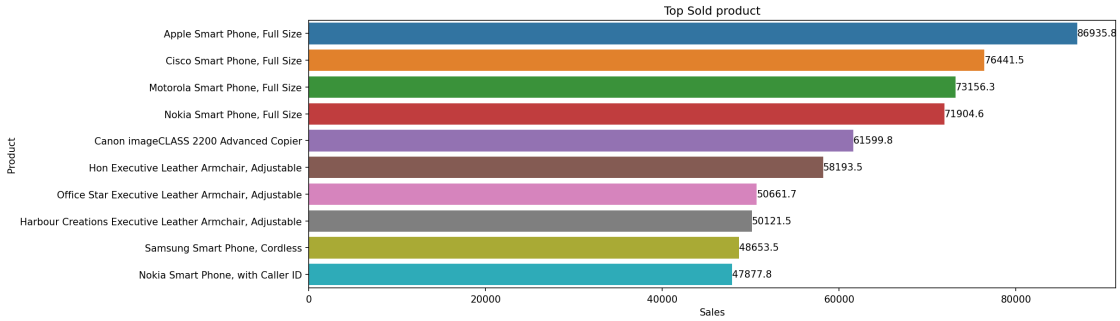
[ ]:

	product_name	sales
0	Apple Smart Phone, Full Size	86935.7786
1	Cisco Smart Phone, Full Size	76441.5306
2	Motorola Smart Phone, Full Size	73156.3030
3	Nokia Smart Phone, Full Size	71904.5555
4	Canon imageCLASS 2200 Advanced Copier	61599.8240
5	Hon Executive Leather Armchair, Adjustable	58193.4841
6	Office Star Executive Leather Armchair, Adjust…	50661.6840
7	Harbour Creations Executive Leather Armchair, …	50121.5160
8	Samsung Smart Phone, Cordless	48653.4600
9	Nokia Smart Phone, with Caller ID	47877.7857

[ ]:
```
fig,axs = plt.subplots(figsize=(15,5),dpi=150)
sns.barplot(top_sold,y='product_name',x='sales')
axs.set(xlabel='Sales',ylabel='Product',title='Top Sold product')

# Add data labels
for p in axs.patches:
    axs.annotate(f'{p.get_width():.1f}', (p.get_width(), p.get_y() + p.
 ↪get_height() / 2),
                 ha='left', va='center', fontsize=10, color='black', xytext=(0,␣
 ↪0),
```

```
                textcoords='offset points')
plt.tight_layout(pad=500)
plt.show()
```



0.2.4 3. Which are the most profitable products?

[ ]:
```
most_profiable_product = df.groupby('product_name',as_index=False).
 ↪aggregate({'profit':'sum'}).sort_values(by='profit',ascending=False).head(10)
most_profiable_product.reset_index(drop=True,inplace=True)
most_profiable_product
```

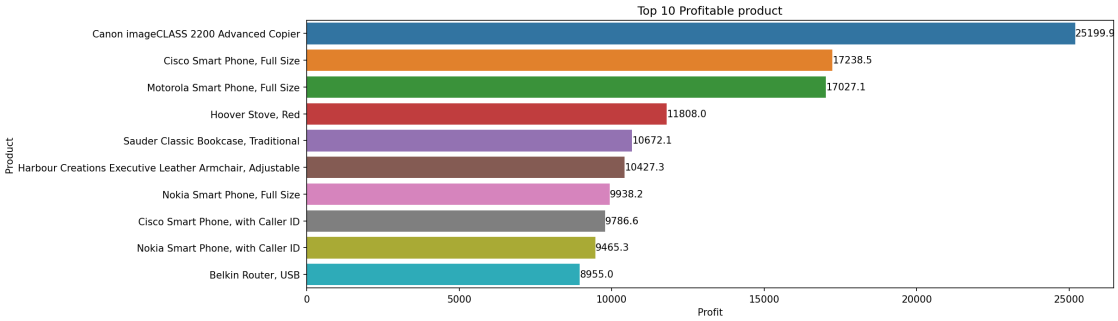
[ ]:

	product_name	profit
0	Canon imageCLASS 2200 Advanced Copier	25199.9280
1	Cisco Smart Phone, Full Size	17238.5206
2	Motorola Smart Phone, Full Size	17027.1130
3	Hoover Stove, Red	11807.9690
4	Sauder Classic Bookcase, Traditional	10672.0730
5	Harbour Creations Executive Leather Armchair, …	10427.3260
6	Nokia Smart Phone, Full Size	9938.1955
7	Cisco Smart Phone, with Caller ID	9786.6408
8	Nokia Smart Phone, with Caller ID	9465.3257
9	Belkin Router, USB	8955.0180

[ ]:
```
fig,axs = plt.subplots(figsize=(15,5),dpi=150)
sns.
 ↪barplot(most_profiable_product,x='profit',y='product_name',estimator='sum',label='profit')
axs.set(xlabel='Profit',ylabel='Product',title='Top 10 Profitable product')

# Add data labels
for p in axs.patches:
    axs.annotate(f'{p.get_width():.1f}', (p.get_width(), p.get_y() + p.
 ↪get_height() / 2),
                 ha='left', va='center', fontsize=10, color='black', xytext=(0,␣
 ↪0),
```

```
                textcoords='offset points')
plt.tight_layout(pad=500)
plt.show()
```



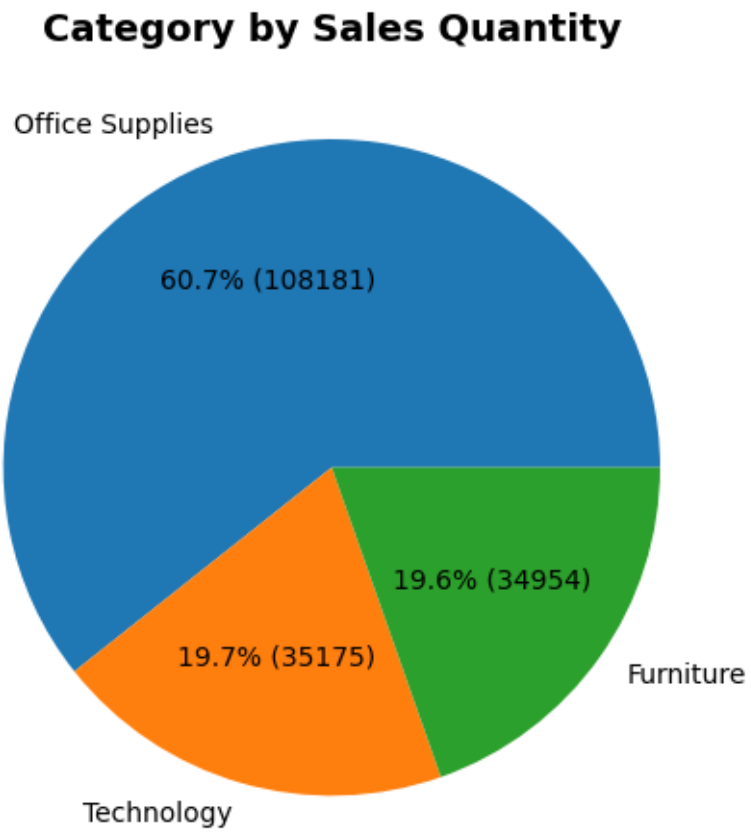
0.2.5 4. What category sold the most?

[ ]:
```
most_sold_cat = df.groupby('category').aggregate({'quantity':'sum'}).
 ↪sort_values(by='quantity',ascending=False).reset_index()
most_sold_cat
```

[ ]:
```
          category  quantity
0  Office Supplies    108182
1       Technology     35176
2        Furniture     34954
```

[ ]:
```
fig, axs = plt.subplots(dpi=100)
axs.pie(most_sold_cat['quantity'], labels=most_sold_cat['category'],␣
 ↪autopct=lambda p: f'{p:.1f}% ({int(p * sum(most_sold_cat["quantity"])/
 ↪100)})')
plt.title('Category by Sales Quantity', fontweight='bold', fontsize=14)

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



[ ]:
```
print(f'We can clearly see that most profitable category is {most_sold_cat.
  ↪iloc[0,0]} and its count is {most_sold_cat.iloc[0,1]:.0f}')
```

```
We can clearly see that most profitable category is Office Supplies and its
count is 108182
```

0.2.6 5. Which are the most profitable category?

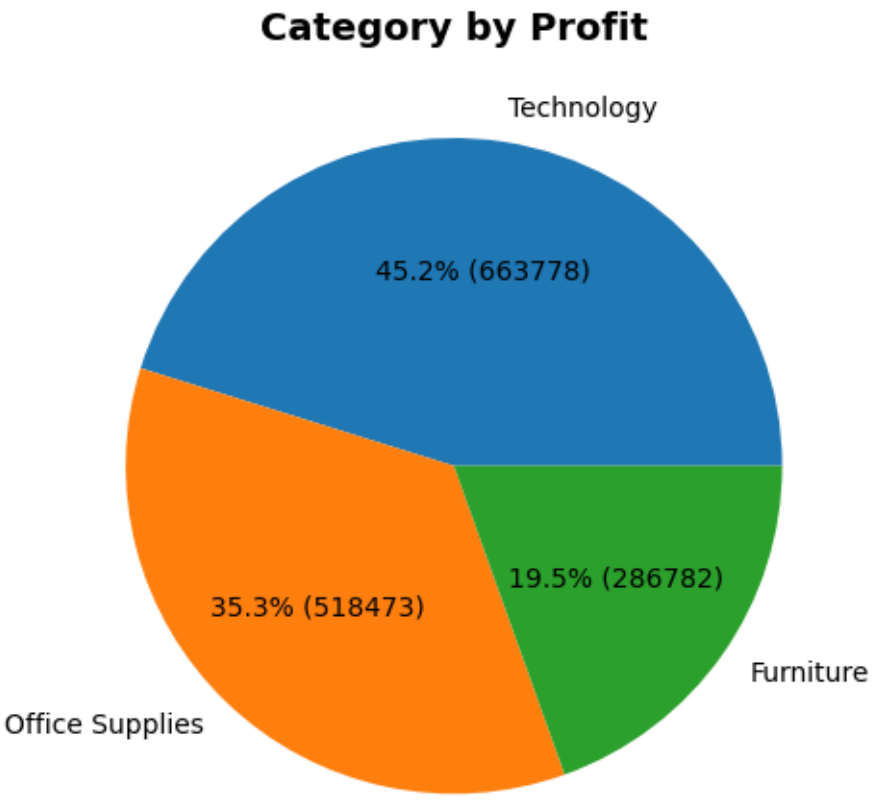
[ ]:
```
top_profitable_catagory = df.groupby(by='category',as_index=False).
  ↪agg({'profit':'sum'}).sort_values('profit',ascending=False)
top_profitable_catagory.reset_index(drop=True,inplace=True)
top_profitable_catagory
```

[ ]:

	category	profit
0	Technology	663778.73318
1	Office Supplies	518473.83430
2	Furniture	286782.25380

[ ]:
```
fig, axs = plt.subplots(dpi=100)
axs.pie(x=top_profitable_catagory['profit'],␣
 ↪labels=top_profitable_catagory['category'], autopct=lambda p: f'{p:.1f}%␣
 ↪({int(p * sum(top_profitable_catagory["profit"])/100)})')
plt.title('Category by Profit', fontweight='bold', fontsize=14)

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



[ ]:
```
print(f'We can clearly see that most profitable category is␣
 ↪{top_profitable_catagory.iloc[0,0]} and its value is␣
 ↪ {top_profitable_catagory.iloc[0,1]:.0f}')
```

```
We can clearly see that most profitable category is Technology and its value is
 663779
```

0.2.7 6. Total sales values by category and subcategory

[ ]:
```
sales_by_cat_subcat = df.groupby(by=['category','sub_category']).
↪aggregate({'sales':'sum'})
sales_by_cat_subcat
```

[ ]:
```
                                    sales
category        sub_category
Furniture       Bookcases     1.466572e+06
                Chairs        1.501682e+06
                Furnishings   3.855783e+05
                Tables        7.570419e+05
Office Supplies Appliances    1.011064e+06
                Art           3.720920e+05
                Binders       4.619115e+05
                Envelopes     1.709043e+05
                Fasteners     8.324232e+04
                Labels        7.340403e+04
                Paper         2.442917e+05
                Storage       1.127086e+06
                Supplies      2.430742e+05
Technology      Accessories   7.492370e+05
                Copiers       1.509436e+06
                Machines      7.790601e+05
                Phones        1.706824e+06
```

0.2.8 7. Which are the most selling products in subcategory?

[ ]:
```
top_selling_subcat = df.groupby(by='sub_category').aggregate({'sales':'sum'}).
↪sort_values('sales',ascending=False).reset_index().head(10)
top_selling_subcat
```

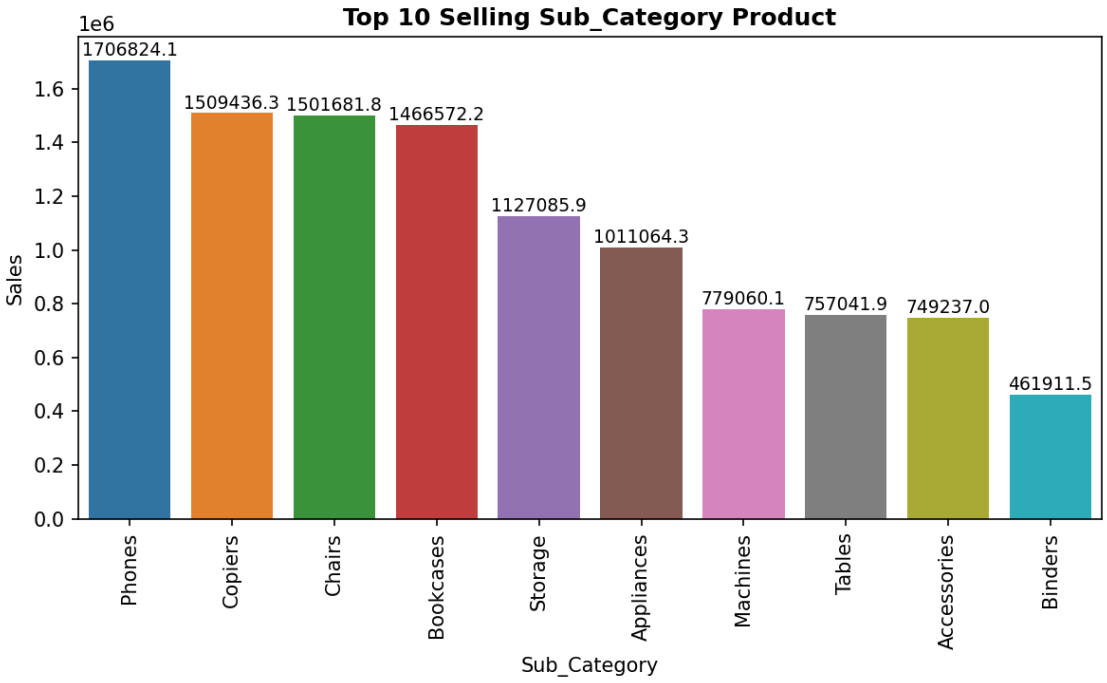
[ ]:
```
  sub_category         sales
0       Phones  1.706824e+06
1      Copiers  1.509436e+06
2       Chairs  1.501682e+06
3    Bookcases  1.466572e+06
4      Storage  1.127086e+06
5   Appliances  1.011064e+06
6     Machines  7.790601e+05
7       Tables  7.570419e+05
8  Accessories  7.492370e+05
9      Binders  4.619115e+05
```

[ ]:
```
fig,axs = plt.subplots(figsize=(8,5),dpi=150)
sns.barplot(top_selling_subcat,x='sub_category',y='sales',estimator='sum')
axs.set(ylabel='Sales',xlabel='Sub_Category')
```

```
axs.set_title(label='Top 10 Selling Sub_Category␣
 ↪Product',fontsize=12,fontweight='bold')
axs.set_xticklabels(axs.get_xticklabels(),rotation=90)

# Add data labels
for p in axs.patches:
    axs.annotate(f'{p.get_height():.1f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                 ha='center', va='center', fontsize=9, color='black', xytext=(0,␣
 ↪5),
                 textcoords='offset points')


fig.tight_layout()
fig.show()
```



0.2.9 8. Which customer segments are the most profitable ?

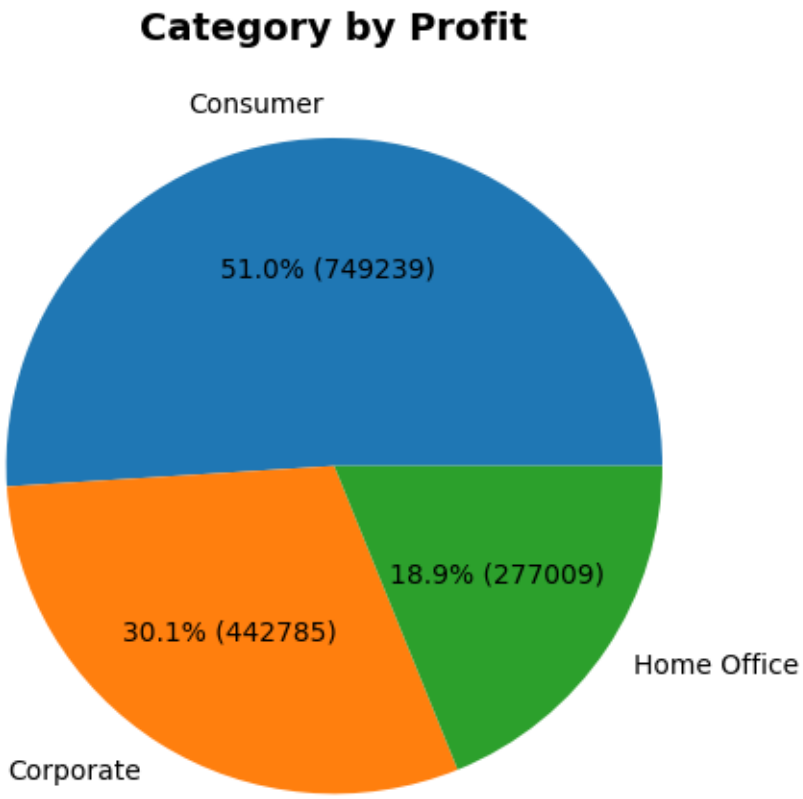
[ ]:
```
profitable_segments = df.groupby(by='segment',as_index=False).
 ↪aggregate({'profit':'sum'}).sort_values('profit',ascending=False)
profitable_segments.reset_index(drop=True,inplace=True)
profitable_segments
```

[ ]:

	segment	profit
0	Consumer	749239.78206
1	Corporate	442785.85866
2	Home Office	277009.18056

[ ]:
```
fig, axs = plt.subplots(dpi=100)
axs.pie(x=profitable_segments['profit'], labels=profitable_segments['segment'],␣
 ↪autopct=lambda p: f'{p:.1f}% ({int(p *␣
 ↪sum(top_profitable_catagory["profit"])/100)})')
plt.title('Category by Profit', fontweight='bold', fontsize=14)

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



0.2.10 9. What shipping modes sold the most products?

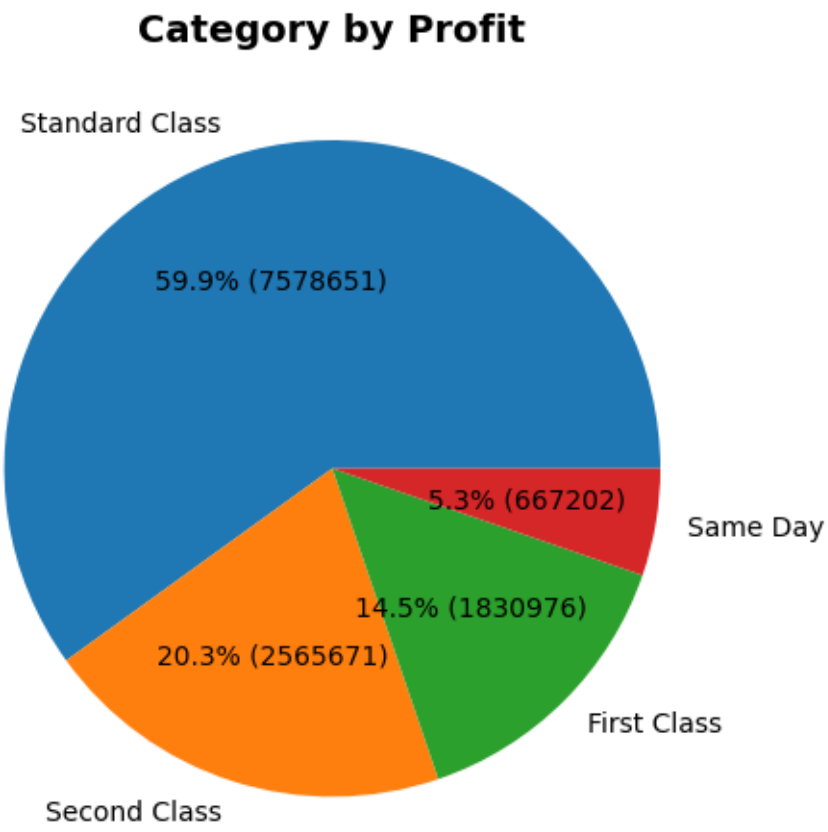
[ ]:
```
shipingmode_by_sales = df.groupby(by='ship_mode',as_index=False).
↪aggregate({'sales':'sum'}).sort_values('sales',ascending=False)
shipingmode_by_sales.reset_index(drop=True,inplace=True)
shipingmode_by_sales
```

[ ]:

	ship_mode	sales
0	Standard Class	7.578652e+06
1	Second Class	2.565672e+06
2	First Class	1.830976e+06
3	Same Day	6.672020e+05

[ ]:
```
fig, axs = plt.subplots(dpi=100)
axs.pie(x= shipingmode_by_sales['sales'], labels=␣
↪shipingmode_by_sales['ship_mode'], autopct=lambda p: f'{p:.1f}% ({int(p *␣
↪sum(shipingmode_by_sales["sales"])/100)})')
plt.title('Category by Profit', fontweight='bold', fontsize=14)

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

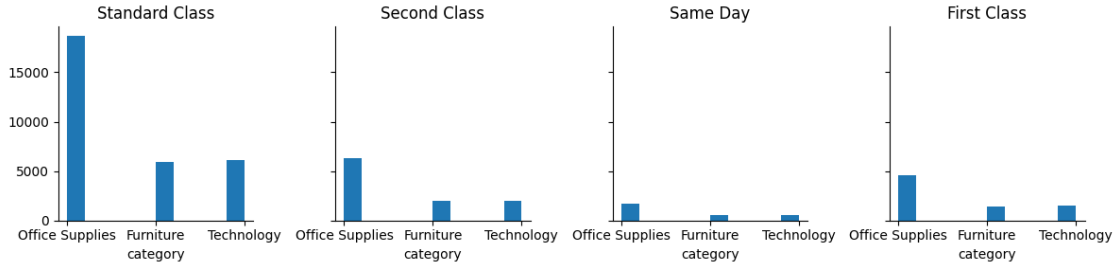


0.2.11 10. Visualize the 'Category' column from the Shipmode column dataset standpoints.

[ ]:
```
# Create a FacetGrid
cat_hist = sns.FacetGrid(df, col='ship_mode')
cat_hist.map(plt.hist, 'category')

# Get titles from each subplot and modify them
for ax in cat_hist.axes.flat:
    title = ax.get_title()
    modified_title = title.split('=')[-1].strip()
    ax.set_title(modified_title)

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



0.2.12 11. What market sold the most products?

[ ]:
```
sold_by_market = df.groupby('market').aggregate({'market':'count'})
sold_by_market.columns=['Martket Count']
sold_by_market = sold_by_market.reset_index().sort_values(by='Martket␣
 ↪Count',ascending=False)
sold_by_market.reset_index(drop=True,inplace=True)
sold_by_market
```

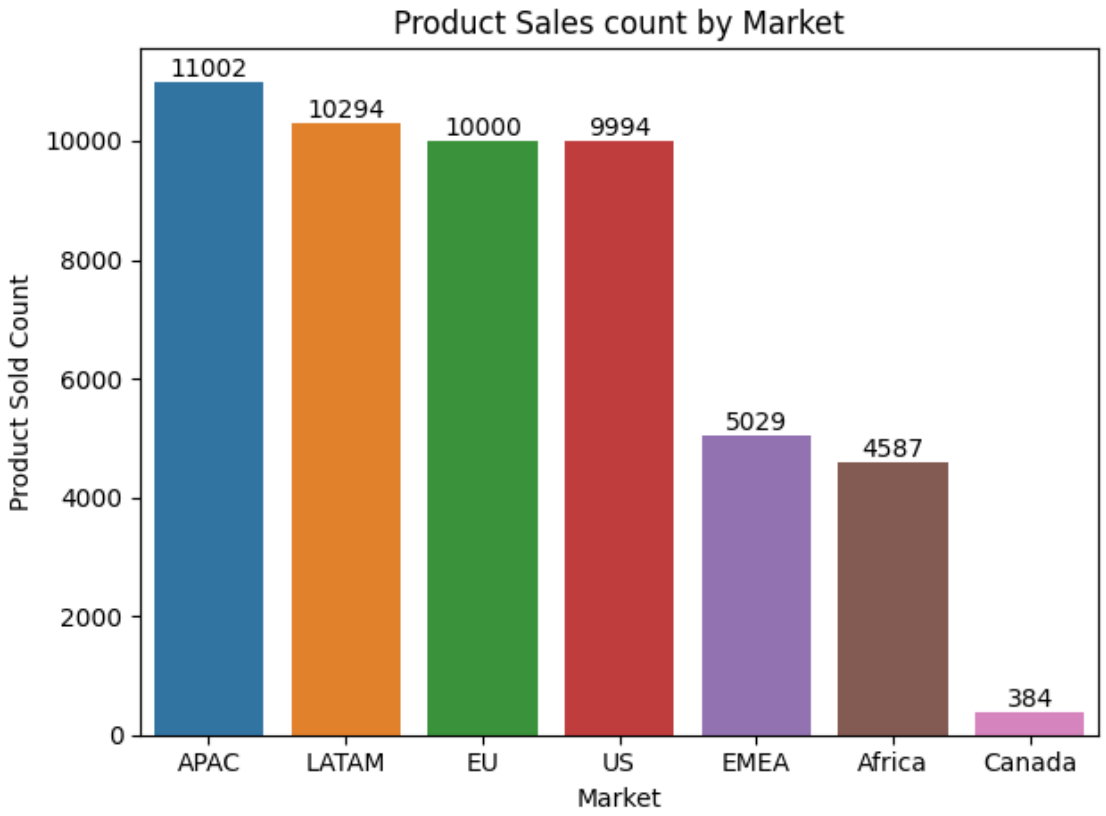
[ ]:

	market	Martket Count
0	APAC	11002
1	LATAM	10294
2	EU	10000
3	US	9994
4	EMEA	5029
5	Africa	4587
6	Canada	384

[ ]:
```
fig,axs = plt.subplots()
sns.barplot(sold_by_market,x='market',y='Martket Count',ax=axs)

# Add data labels
for p in axs.patches:
    axs.annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                 ha='center', va='center', fontsize=10, color='black',␣
 ↪xytext=(0, 5),
                 textcoords='offset points')

axs.set(xlabel='Market',ylabel='Product Sold Count',title='Product Sales count␣
 ↪by Market')
plt.tight_layout()
plt.show()
```



0.2.13 12. Which are the Top 10 country by sales?

[ ]:
```
top_sales_country = df.groupby('country',as_index=False).aggregate({'sales':
 ↪'sum'}).sort_values(by='sales',ascending=False).head(10)
top_sales_country.reset_index(drop=True,inplace=True)
top_sales_country
```

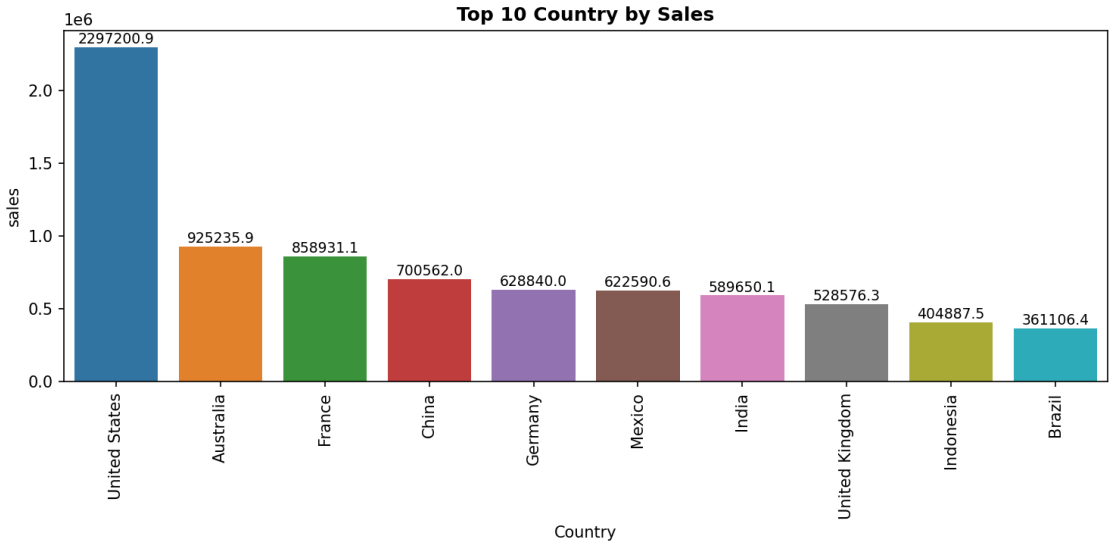
[ ]:

	country	sales
0	United States	2.297201e+06
1	Australia	9.252359e+05
2	France	8.589311e+05
3	China	7.005620e+05
4	Germany	6.288400e+05
5	Mexico	6.225906e+05
6	India	5.896501e+05
7	United Kingdom	5.285763e+05
8	Indonesia	4.048875e+05
9	Brazil	3.611064e+05

[ ]:
```
fig,axs = plt.subplots(figsize=(10,5),dpi=150)
sns.barplot(top_sales_country,x='country',y='sales',estimator='sum')
axs.set(ylabel='sales',xlabel='Country')
axs.set_title(label='Top 10 Country by Sales',fontsize=12,fontweight='bold')
axs.set_xticklabels(axs.get_xticklabels(),rotation=90)

# Add data labels
for p in axs.patches:
    axs.annotate(f'{p.get_height():.1f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                 ha='center', va='center', fontsize=9, color='black', xytext=(0,␣
 ↪5),
                 textcoords='offset points')


fig.tight_layout()
fig.show()
```



0.2.14 13. Which are the average shipping cost for top 10 different countries?

[ ]:
```
avg_shipcost_by_country = df.groupby('country').aggregate({'shipping_cost':
 ↪'mean'}).sort_values('shipping_cost',ascending=False).reset_index().head(10)
avg_shipcost_by_country
```

[ ]:

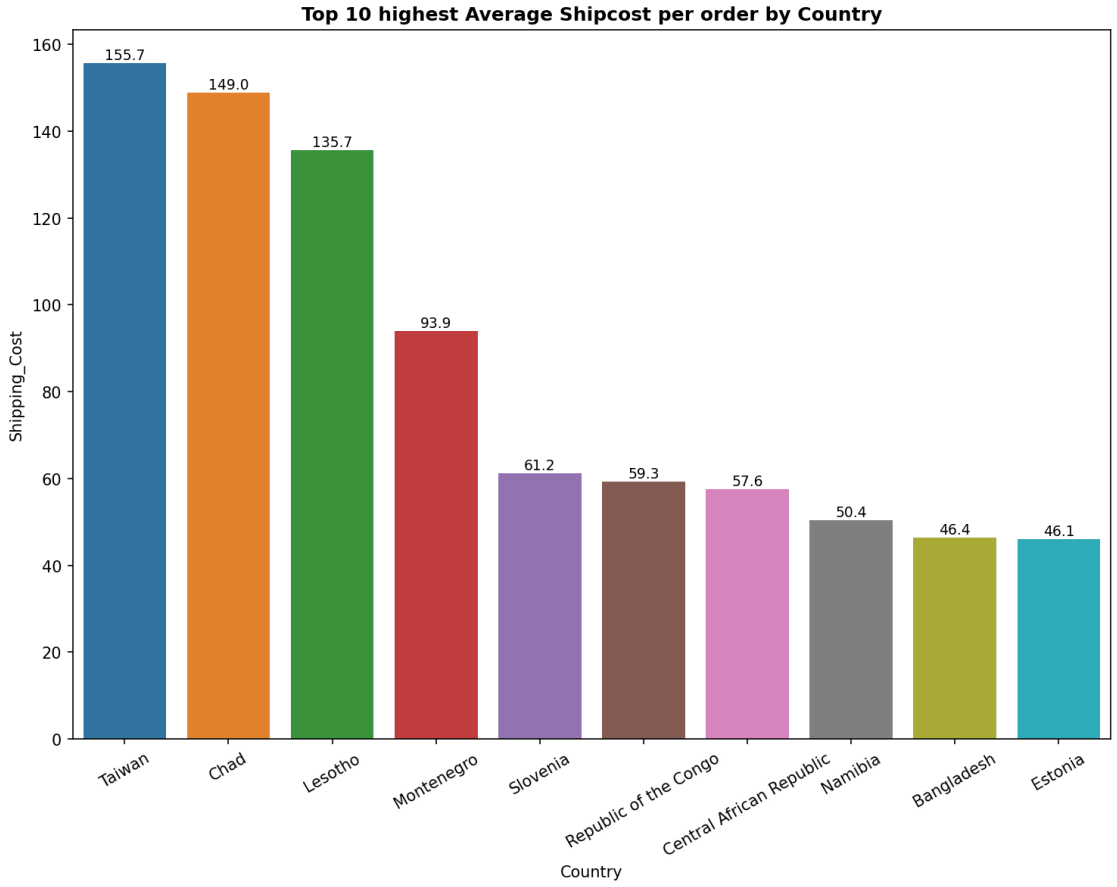
	country	shipping_cost
0	Taiwan	155.660714
1	Chad	148.970000
2	Lesotho	135.650000
3	Montenegro	93.937500
4	Slovenia	61.220000
5	Republic of the Congo	59.303333
6	Central African Republic	57.625714
7	Namibia	50.370000
8	Bangladesh	46.402883
9	Estonia	46.070000

[ ]:
```
fig,axs = plt.subplots(figsize=(10,8),dpi=150)
sns.
 ↪barplot(avg_shipcost_by_country,x='country',y='shipping_cost',estimator='sum')
axs.set(ylabel='Shipping_Cost',xlabel='Country')
axs.set_title(label='Top 10 highest Average Shipcost per order by␣
 ↪Country',fontsize=12,fontweight='bold')
axs.set_xticklabels(axs.get_xticklabels(),rotation=30)

# Add data labels
for p in axs.patches:
```

```
    axs.annotate(f'{p.get_height():.1f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                 ha='center', va='center', fontsize=9, color='black', xytext=(0,␣
 ↪5),
                 textcoords='offset points')


fig.tight_layout()
fig.show()
```



0.2.15 14. Who are the top-10 most profitable customers?

[ ]:
```
top10_profitable_cust = df.groupby('customer_name').aggregate({'profit':'sum'}).
 ↪sort_values('profit',ascending=False).reset_index().head(10)
top10_profitable_cust
```

[ ]:

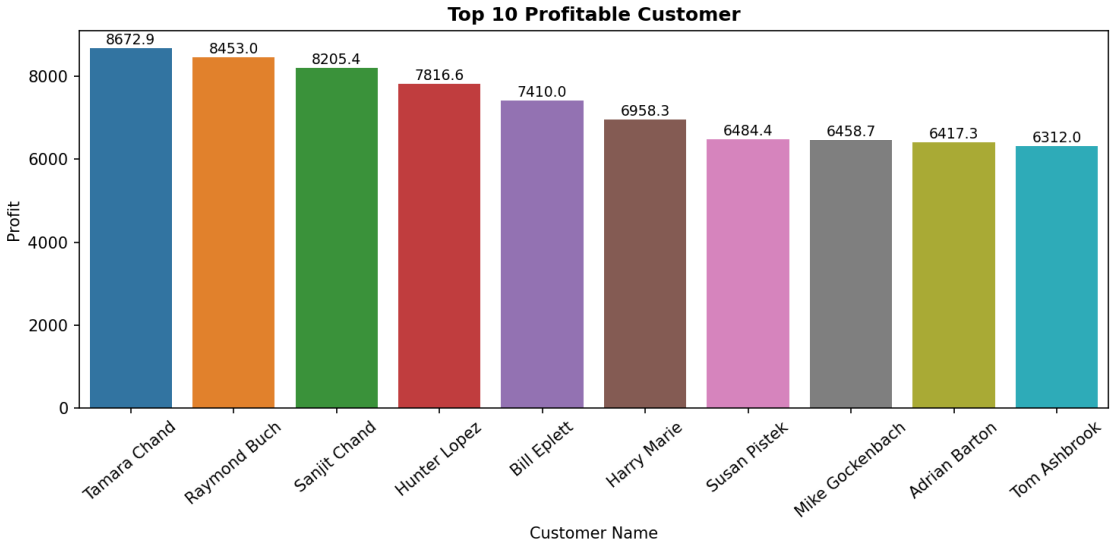
	customer_name	profit
0	Tamara Chand	8672.89890
1	Raymond Buch	8453.04950

```
2     Sanjit Chand  8205.37990
3     Hunter Lopez  7816.56778
4      Bill Eplett  7410.00530
5      Harry Marie  6958.28640
6     Susan Pistek  6484.40726
7  Mike Gockenbach  6458.67620
8    Adrian Barton  6417.28450
9     Tom Ashbrook  6311.97910
```

[ ]:
```
fig,axs = plt.subplots(figsize=(10,5),dpi=150)
sns.barplot(top10_profitable_cust,x='customer_name',y='profit',estimator='sum')
axs.set(ylabel='Profit',xlabel='Customer Name')
axs.set_title(label='Top 10 Profitable Customer',fontsize=12,fontweight='bold')
axs.set_xticklabels(axs.get_xticklabels(),rotation=40)

# Add data labels
for p in axs.patches:
    axs.annotate(f'{p.get_height():.1f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                 ha='center', va='center', fontsize=9, color='black', xytext=(0,␣
 ↪5),
                 textcoords='offset points')


fig.tight_layout()
fig.show()
```



0.2.16 15. Total sales values by year and month.

[ ]:
```
df['order_month'] = df['order_date'].dt.month_name()
df['order_monthno'] = df['order_date'].dt.month
df['mon_year'] = df['year'].astype(str) +" "+ df['order_month']
```

[ ]:
```
sales_by_year_month = df.
 ↪groupby(['year','order_month','order_monthno','mon_year'],as_index=False).
 ↪aggregate({'sales':'sum'}).sort_values(by=['year','order_monthno'])
sales_by_year_month.reset_index(drop=True,inplace=True)
sales_by_year_month
```

[ ]:

	year	order_month	order_monthno	mon_year	sales
0	2019	March	3	2019 March	42825.21582
1	2019	April	4	2019 April	88954.75486
2	2019	May	5	2019 May	119442.15558
3	2019	June	6	2019 June	139826.74052
4	2019	July	7	2019 July	127641.66712
5	2019	August	8	2019 August	186148.99120
6	2019	September	9	2019 September	149219.55656
7	2019	October	10	2019 October	172848.24536
8	2019	November	11	2019 November	227554.79346
9	2019	December	12	2019 December	266511.22072
10	2020	January	1	2020 January	264655.56066
11	2020	February	2	2020 February	262912.29768
12	2020	March	3	2020 March	272717.77890
13	2020	April	4	2020 April	117231.40512
14	2020	May	5	2020 May	119338.42248
15	2020	June	6	2020 June	160935.84766
16	2020	July	7	2020 July	184981.66746
17	2020	August	8	2020 August	232822.90736
18	2020	September	9	2020 September	221491.22934
19	2020	October	10	2020 October	201818.10992
20	2020	November	11	2020 November	289392.91042
21	2020	December	12	2020 December	286677.60594
22	2021	January	1	2021 January	309593.88214
23	2021	February	2	2021 February	273932.56738
24	2021	March	3	2021 March	310120.17082
25	2021	April	4	2021 April	183370.69674
26	2021	May	5	2021 May	198970.28066
27	2021	June	6	2021 June	169011.30968
28	2021	July	7	2021 July	194334.92118
29	2021	August	8	2021 August	362186.13000
30	2021	September	9	2021 September	305007.63958
31	2021	October	10	2021 October	253383.70574
32	2021	November	11	2021 November	356959.55586
33	2021	December	12	2021 December	349611.07536

34	2022	January	1	2022 January	321874.93608
35	2022	February	2	2022 February	367667.01870
36	2022	March	3	2022 March	323807.94670
37	2022	April	4	2022 April	232856.12852
38	2022	May	5	2022 May	210440.70844
39	2022	June	6	2022 June	273854.53258
40	2022	July	7	2022 July	258454.24262
41	2022	August	8	2022 August	344486.91522
42	2022	September	9	2022 September	349408.25830
43	2022	October	10	2022 October	352211.88060
44	2022	November	11	2022 November	457983.86840
45	2022	December	12	2022 December	461606.25356
46	2023	January	1	2023 January	471599.18394
47	2023	February	2	2023 February	517965.33796
48	2023	March	3	2023 March	295853.67898

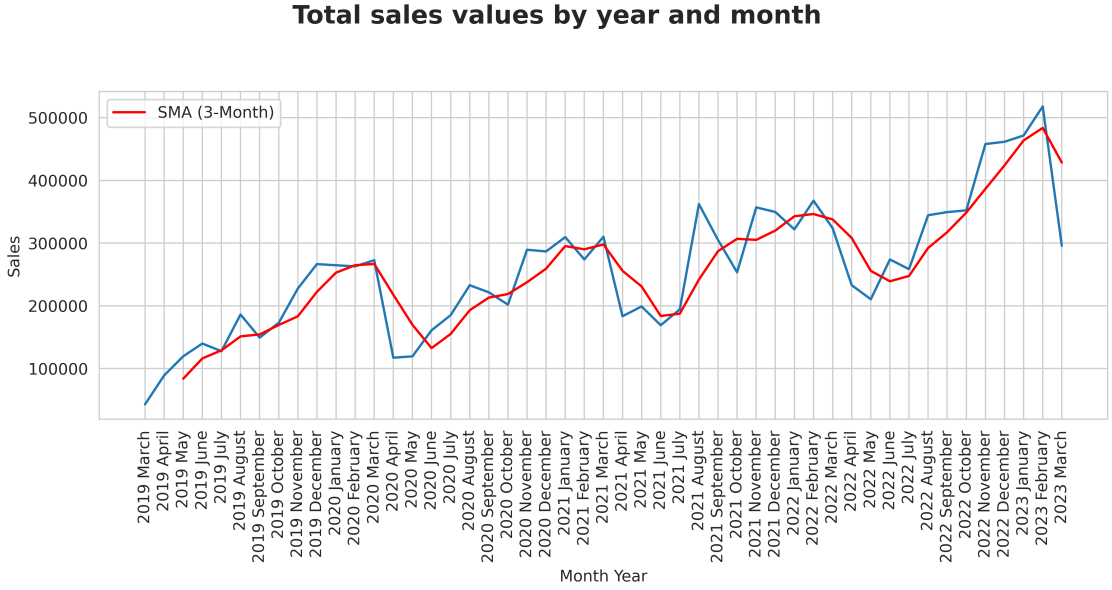
[ ]:
```
sns.set_style('whitegrid')
fig, axs = plt.subplots(figsize=(10, 5), dpi=500)

# Create the lineplot on axs
sns.lineplot(data=sales_by_year_month, x='mon_year', y='sales', ax=axs)
axs.set_xticklabels(axs.get_xticklabels(), rotation=90)
axs.set(xlabel='Month Year', ylabel='Sales')

# Calculate and plot the SMA line
sales_by_year_month['sma'] = sales_by_year_month['sales'].rolling(window=3).
 ↪mean()
sns.lineplot(data=sales_by_year_month, x='mon_year', y='sma', color='red',␣
 ↪ax=axs, label='SMA (3-Month)')

# Set the figure title
fig.suptitle('Total sales values by year and month', fontsize=16, y=1.05,␣
 ↪fontweight='bold')

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


Total sales values by year and month
SMA (3-Month)
Sales
500000
400000
300000
200000
100000
2019 March
2019 April
2019 May
2019 June
2019 July
2019 August
2019 September
2019 October
2019 November
2019 December
2020 January
2020 February
2020 March
2020 April
2020 May
2020 June
2020 July
2020 August
2020 September
2020 October
2020 November
2020 December
2021 January
2021 February
2021 March
2021 April
2021 May
2021 June
2021 July
2021 August
2021 September
2021 October
2021 November
2021 December
2022 January
2022 February
2022 March
2022 April
2022 May
2022 June
2022 July
2022 August
2022 September
2022 October
2022 November
2022 December
2023 January
2023 February
2023 March
Month Year

1 END