

eda-analysis

February 22, 2025

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[2]: data = pd.read_csv(r'Location', encoding='latin1')
```

```
[3]: df= data.copy()
```

```
[4]: df.head(3)
```

```
[4]:   Row ID      Order ID  Order Date  Ship Date  Ship Mode  Customer ID \
0    32298  CA-2012-124891  31-07-2012  31-07-2012    Same Day    RH-19495
1    26341  IN-2013-77878    2/5/2013    5/2/2013  Second Class    JR-16210
2    25330  IN-2013-71249  17-10-2013  17-10-2013    First Class    CR-12730
```

```
      Customer Name  Segment      City      State ... \
0    Rick Hansen   Consumer  New York City    New York ...
1  Justin Ritter   Corporate  Wollongong  New South Wales ...
2   Craig Reiter   Consumer    Brisbane    Queensland ...
```

```
      Product ID  Category Sub-Category \
0  TEC-AC-10003033  Technology  Accessories
1  FUR-CH-10003950  Furniture    Chairs
2  TEC-PH-10004664  Technology    Phones
```

```
      Product Name      Sales Quantity \
0  Plantronics CS510 - Over-the-Head monaural Wir...  2309.650      7
1      Novimex Executive Leather Armchair, Black  3709.395      9
2      Nokia Smart Phone, with Caller ID  5175.171      9
```

```
      Discount  Profit  Shipping Cost  Order Priority
0         0.0  762.1845         933.57      Critical
1         0.1 -288.7650         923.63      Critical
2         0.1  919.9710         915.49      Medium
```

```
[3 rows x 24 columns]
```

1 Data Integrity:

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Row ID                51290 non-null  int64
1   Order ID              51290 non-null  object
2   Order Date            51290 non-null  object
3   Ship Date             51290 non-null  object
4   Ship Mode             51290 non-null  object
5   Customer ID           51290 non-null  object
6   Customer Name         51290 non-null  object
7   Segment               51290 non-null  object
8   City                  51290 non-null  object
9   State                 51290 non-null  object
10  Country               51290 non-null  object
11  Postal Code           9994 non-null   float64
12  Market                51290 non-null  object
13  Region                51290 non-null  object
14  Product ID            51290 non-null  object
15  Category              51290 non-null  object
16  Sub-Category          51290 non-null  object
17  Product Name          51290 non-null  object
18  Sales                 51290 non-null  float64
19  Quantity              51290 non-null  int64
20  Discount              51290 non-null  float64
21  Profit                51290 non-null  float64
22  Shipping Cost         51290 non-null  float64
23  Order Priority         51290 non-null  object
dtypes: float64(5), int64(2), object(17)
memory usage: 9.4+ MB
```

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

```
[6]: (51290, 24)
```

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

```
[7]:
```

	Row ID	Postal Code	Sales	Quantity	Discount	\
count	51290.00000	9994.000000	51290.000000	51290.000000	51290.000000	
mean	25645.50000	55190.379428	246.490581	3.476545	0.142908	
std	14806.29199	32063.693350	487.565361	2.278766	0.212280	
min	1.00000	1040.000000	0.444000	1.000000	0.000000	

25%	12823.25000	23223.000000	30.758625	2.000000	0.000000
50%	25645.50000	56430.500000	85.053000	3.000000	0.000000
75%	38467.75000	90008.000000	251.053200	5.000000	0.200000
max	51290.00000	99301.000000	22638.480000	14.000000	0.850000

	Profit	Shipping Cost
count	51290.000000	51290.000000
mean	28.610982	26.375915
std	174.340972	57.296804
min	-6599.978000	0.000000
25%	0.000000	2.610000
50%	9.240000	7.790000
75%	36.810000	24.450000
max	8399.976000	933.570000

```
[8]: df.describe(include = 'all')
```

```
[8]:
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	\
count	51290.00000	51290	51290	51290	51290	
unique	NaN	25035	1430	1430	4	
top	NaN	CA-2014-100111	18-06-2014	18-06-2014	Standard Class	
freq	NaN	14	135	135	30775	
mean	25645.50000	NaN	NaN	NaN	NaN	
std	14806.29199	NaN	NaN	NaN	NaN	
min	1.00000	NaN	NaN	NaN	NaN	
25%	12823.25000	NaN	NaN	NaN	NaN	
50%	25645.50000	NaN	NaN	NaN	NaN	
75%	38467.75000	NaN	NaN	NaN	NaN	
max	51290.00000	NaN	NaN	NaN	NaN	

	Customer ID	Customer Name	Segment	City	State	...	\
count	51290	51290	51290	51290	51290	...	
unique	1590	795	3	3636	1094	...	
top	PO-18850	Muhammed Yedwab	Consumer	New York City	California	...	
freq	97	108	26518	915	2001	...	
mean	NaN	NaN	NaN	NaN	NaN	...	
std	NaN	NaN	NaN	NaN	NaN	...	
min	NaN	NaN	NaN	NaN	NaN	...	
25%	NaN	NaN	NaN	NaN	NaN	...	
50%	NaN	NaN	NaN	NaN	NaN	...	
75%	NaN	NaN	NaN	NaN	NaN	...	
max	NaN	NaN	NaN	NaN	NaN	...	

	Product ID	Category	Sub-Category	Product Name	\
count	51290	51290	51290	51290	
unique	10292	3	17	3788	
top	OFF-AR-10003651	Office Supplies	Binders	Staples	

freq	35	31273	6152	227
mean	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN

	Sales	Quantity	Discount	Profit	Shipping Cost \
count	51290.000000	51290.000000	51290.000000	51290.000000	51290.000000
unique	NaN	NaN	NaN	NaN	NaN
top	NaN	NaN	NaN	NaN	NaN
freq	NaN	NaN	NaN	NaN	NaN
mean	246.490581	3.476545	0.142908	28.610982	26.375915
std	487.565361	2.278766	0.212280	174.340972	57.296804
min	0.444000	1.000000	0.000000	-6599.978000	0.000000
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

	Order Priority
count	51290
unique	4
top	Medium
freq	29433
mean	NaN
std	NaN
min	NaN
25%	NaN
50%	NaN
75%	NaN
max	NaN

[11 rows x 24 columns]

```
[9]: df.dtypes
```

```
[9]: Row ID          int64
Order ID          object
Order Date        object
Ship Date         object
Ship Mode         object
Customer ID       object
Customer Name     object
Segment          object
```

```

City                object
State               object
Country             object
Postal Code         float64
Market              object
Region              object
Product ID          object
Category            object
Sub-Category        object
Product Name        object
Sales               float64
Quantity            int64
Discount             float64
Profit              float64
Shipping Cost        float64
Order Priority       object
dtype: object

```

```

[10]: #Columns that contain Float Values
float_type = df[['Postal Code', 'Sales', 'Discount', 'Profit', 'Shipping Cost']]
float_type.head()

```

```

[10]:   Postal Code    Sales  Discount    Profit  Shipping Cost
0      10024.0  2309.650         0.0  762.1845          933.57
1         NaN  3709.395         0.1 -288.7650          923.63
2         NaN  5175.171         0.1  919.9710          915.49
3         NaN  2892.510         0.1  -96.5400          910.16
4         NaN  2832.960         0.0  311.5200          903.04

```

```

[11]: else_type = df[['Row ID', 'Order ID', 'Ship Mode', 'Customer ID', 'Customer Name',
                    'Segment', 'City', 'State', 'Country']]
else_type.head()

```

```

[11]:   Row ID    Order ID    Ship Mode  Customer ID  Customer Name \
0    32298  CA-2012-124891    Same Day    RH-19495    Rick Hansen
1    26341  IN-2013-77878  Second Class    JR-16210    Justin Ritter
2    25330  IN-2013-71249    First Class    CR-12730    Craig Reiter
3    13524  ES-2013-1579342  First Class    KM-16375  Katherine Murray
4    47221  SG-2013-4320    Same Day    RH-9495    Rick Hansen

```

```

      Segment      City      State      Country
0    Consumer  New York City    New York  United States
1  Corporate  Wollongong  New South Wales  Australia
2    Consumer    Brisbane  Queensland  Australia
3  Home Office    Berlin    Berlin    Germany
4    Consumer    Dakar    Dakar    Senegal

```

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

```
[12]: Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',  
         'Customer ID', 'Customer Name', 'Segment', 'City', 'State', 'Country',  
         'Postal Code', 'Market', 'Region', 'Product ID', 'Category',  
         'Sub-Category', 'Product Name', 'Sales', 'Quantity', 'Discount',  
         'Profit', 'Shipping Cost', 'Order Priority'],  
         dtype='object')
```

```
[13]: date_type = df[['Order Date', 'Ship Date']]  
      date_type.head()
```

```
[13]:   Order Date  Ship Date  
0  31-07-2012  31-07-2012  
1    2/5/2013    5/2/2013  
2  17-10-2013  17-10-2013  
3  28-01-2013  28-01-2013  
4   11/5/2013   5/11/2013
```

```
[14]: df['Order Date'] = pd.to_datetime(df['Order Date'], format='mixed',  
    ↪errors='coerce')  
df['Ship Date'] = pd.to_datetime(df['Ship Date'], format='mixed',  
    ↪errors='coerce')  
  
# Verify the conversion  
print(df[['Order Date', 'Ship Date']].dtypes)
```

```
Order Date    datetime64[ns]  
Ship Date     datetime64[ns]  
dtype: object
```

```
[15]: df.dtypes
```

```
[15]: Row ID                int64  
      Order ID            object  
      Order Date          datetime64[ns]  
      Ship Date           datetime64[ns]  
      Ship Mode           object  
      Customer ID         object  
      Customer Name       object  
      Segment            object  
      City               object  
      State              object  
      Country            object  
      Postal Code         float64  
      Market             object  
      Region             object
```

```

Product ID          object
Category            object
Sub-Category        object
Product Name        object
Sales               float64
Quantity            int64
Discount            float64
Profit              float64
Shipping Cost       float64
Order Priority       object
dtype: object

```

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

```
[16]: Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
            'Customer ID', 'Customer Name', 'Segment', 'City', 'State', 'Country',
            'Postal Code', 'Market', 'Region', 'Product ID', 'Category',
            'Sub-Category', 'Product Name', 'Sales', 'Quantity', 'Discount',
            'Profit', 'Shipping Cost', 'Order Priority'],
           dtype='object')
```

```
[17]: df['Ship Mode'] = data['Ship Mode'].astype('category')
df['Segment'] = data['Segment'].astype('category')
df['Country'] = data['Country'].astype('category')
df['Market'] = data['Market'].astype('category')
df['Region'] = data['Region'].astype('category')
df['Category'] = data['Category'].astype('category')
df['Sub-Category'] = data['Sub-Category'].astype('category')
df['Order Priority'] = data['Order Priority'].astype('category')
```

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

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   Row ID                51290 non-null  int64
 1   Order ID              51290 non-null  object
 2   Order Date            51290 non-null  datetime64[ns]
 3   Ship Date             51290 non-null  datetime64[ns]
 4   Ship Mode             51290 non-null  category
 5   Customer ID           51290 non-null  object
 6   Customer Name         51290 non-null  object
 7   Segment              51290 non-null  category
 8   City                 51290 non-null  object
 9   State                51290 non-null  object

```

```

10 Country          51290 non-null  category
11 Postal Code      9994 non-null   float64
12 Market          51290 non-null  category
13 Region          51290 non-null  category
14 Product ID      51290 non-null  object
15 Category        51290 non-null  category
16 Sub-Category    51290 non-null  category
17 Product Name    51290 non-null  object
18 Sales           51290 non-null  float64
19 Quantity        51290 non-null  int64
20 Discount        51290 non-null  float64
21 Profit          51290 non-null  float64
22 Shipping Cost   51290 non-null  float64
23 Order Priority   51290 non-null  category
dtypes: category(8), datetime64[ns](2), float64(5), int64(2), object(7)
memory usage: 6.7+ MB

```

```
[19]: df.duplicated().sum()
```

```
[19]: 0
```

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

```

[20]: Row ID          0
      Order ID       0
      Order Date     0
      Ship Date      0
      Ship Mode      0
      Customer ID    0
      Customer Name  0
      Segment       0
      City          0
      State         0
      Country       0
      Postal Code   41296
      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

```


dtype: int64

```
[21]: df.nunique()
```

```
[21]: Row ID          51290
      Order ID       25035
      Order Date     1430
      Ship Date      1430
      Ship Mode       4
      Customer ID    1590
      Customer Name   795
      Segment        3
      City           3636
      State          1094
      Country        147
      Postal Code     631
      Market         7
      Region         13
      Product ID     10292
      Category       3
      Sub-Category   17
      Product Name   3788
      Sales          22995
      Quantity       14
      Discount       27
      Profit         24575
      Shipping Cost  10037
      Order Priority   4
      dtype: int64
```

```
[22]: postal_code_mode = df['Postal Code'].mode()[0]
      #postal_code_mode
      df['Postal Code'] = df['Postal Code'].fillna(postal_code_mode)
```

```
[23]: df['Postal Code'].isnull().sum()
```

```
[23]: 0
```

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

```
[24]: Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
        'Customer ID', 'Customer Name', 'Segment', 'City', 'State', 'Country',
        'Postal Code', 'Market', 'Region', 'Product ID', 'Category',
        'Sub-Category', 'Product Name', 'Sales', 'Quantity', 'Discount',
        'Profit', 'Shipping Cost', 'Order Priority'],
        dtype='object')
```

```
[25]: Sales_negative = df['Sales'] >= 0
Sales_negative
```

```
[25]: 0      True
      1      True
      2      True
      3      True
      4      True
      ...
     51285   True
     51286   True
     51287   True
     51288   True
     51289   True
      Name: Sales, Length: 51290, dtype: bool
```

```
[26]: negative_sales = df[df['Sales'] < 0]
      if not negative_sales.empty:
          print("Negative values in 'Sales':")
          print(negative_sales)
      negative_sales
```

```
[26]: Empty DataFrame
      Columns: [Row ID, Order ID, Order Date, Ship Date, Ship Mode, Customer ID,
      Customer Name, Segment, City, State, Country, Postal Code, Market, Region,
      Product ID, Category, Sub-Category, Product Name, Sales, Quantity, Discount,
      Profit, Shipping Cost, Order Priority]
      Index: []

      [0 rows x 24 columns]
```

```
[27]: # Check for negative values in 'Profit' column
      negative_profit = df[df['Profit'] < 0]
      if not negative_profit.empty:
          print("Negative values in 'Profit':")
          display(negative_profit.head(3))
```

Negative values in 'Profit':

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	\
1	26341	IN-2013-77878	2013-02-05	2013-05-02	Second Class	JR-16210	
3	13524	ES-2013-1579342	2013-01-28	2013-01-28	First Class	KM-16375	
9	40936	CA-2012-116638	2012-01-28	2012-01-28	Second Class	JH-15985	

	Customer Name	Segment	City	State	...	\
1	Justin Ritter	Corporate	Wollongong	New South Wales	...	
3	Katherine Murray	Home Office	Berlin	Berlin	...	
9	Joseph Holt	Consumer	Concord	North Carolina	...	

	Product ID	Category	Sub-Category	\
1	FUR-CH-10003950	Furniture	Chairs	
3	TEC-PH-10004583	Technology	Phones	
9	FUR-TA-10000198	Furniture	Tables	

	Product Name	Sales	Quantity	\
1	Novimex Executive Leather Armchair, Black	3709.395	9	
3	Motorola Smart Phone, Cordless	2892.510	5	
9	Chromcraft Bull-Nose Wood Oval Conference Tabl...	4297.644	13	

	Discount	Profit	Shipping Cost	Order Priority
1	0.1	-288.7650	923.63	Critical
3	0.1	-96.5400	910.16	Medium
9	0.4	-1862.3124	865.74	Critical

[3 rows x 24 columns]

```
[28]: negative_profit_count = (df['Profit'] < 0).sum()
negative_profit_count
```

[28]: 12544

```
[29]: mean_profit = df[df['Profit'] >= 0]['Profit'].mean()
data.loc[data['Profit'] < 0, 'Profit'] = mean_profit
mean_profit
```

[29]: 61.634838357507874

```
[30]: # Replace negative 'Profit' values with the calculated mean
df['Profit'] = df['Profit'].apply(lambda x: mean_profit if x < 0 else x)
```

```
[31]: print("Negative values in 'Profit' have been replaced with the mean of
↪non-negative profits.")
print(df['Profit'].head(5))
```

Negative values in 'Profit' have been replaced with the mean of non-negative profits.

```
0    762.184500
1     61.634838
2    919.971000
3     61.634838
4    311.520000
```

Name: Profit, dtype: float64

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

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 51290 entries, 0 to 51289

Data columns (total 24 columns):

#	Column	Non-Null Count	Dtype
0	Row ID	51290 non-null	int64
1	Order ID	51290 non-null	object
2	Order Date	51290 non-null	datetime64[ns]
3	Ship Date	51290 non-null	datetime64[ns]
4	Ship Mode	51290 non-null	category
5	Customer ID	51290 non-null	object
6	Customer Name	51290 non-null	object
7	Segment	51290 non-null	category
8	City	51290 non-null	object
9	State	51290 non-null	object
10	Country	51290 non-null	category
11	Postal Code	51290 non-null	float64
12	Market	51290 non-null	category
13	Region	51290 non-null	category
14	Product ID	51290 non-null	object
15	Category	51290 non-null	category
16	Sub-Category	51290 non-null	category
17	Product Name	51290 non-null	object
18	Sales	51290 non-null	float64
19	Quantity	51290 non-null	int64
20	Discount	51290 non-null	float64
21	Profit	51290 non-null	float64
22	Shipping Cost	51290 non-null	float64
23	Order Priority	51290 non-null	category

dtypes: category(8), datetime64[ns](2), float64(5), int64(2), object(7)
memory usage: 6.7+ MB

2 Segment the Data:

```
[33]: segment_data = df[['Customer ID', 'Customer Name', 'Category', 'Product_
    ↳ Name', 'Sub-Category',
    ↳ 'Sales', 'Profit', 'Quantity', 'Discount', 'Order Date', 'City',
    ↳ 'State', 'Region',
    ↳ 'Market', 'Order Priority']]
segment_data.head()
```

```
[33]: Customer ID      Customer Name      Category \
0      RH-19495      Rick Hansen      Technology
1      JR-16210      Justin Ritter      Furniture
2      CR-12730      Craig Reiter      Technology
3      KM-16375      Katherine Murray      Technology
4      RH-9495      Rick Hansen      Technology
```

	Product Name	Sub-Category	Sales	\
0	Plantronics CS510 - Over-the-Head monaural Wir...	Accessories	2309.650	
1	Novimex Executive Leather Armchair, Black	Chairs	3709.395	
2	Nokia Smart Phone, with Caller ID	Phones	5175.171	
3	Motorola Smart Phone, Cordless	Phones	2892.510	
4	Sharp Wireless Fax, High-Speed	Copiers	2832.960	

	Profit	Quantity	Discount	Order Date	City	State	\
0	762.184500	7	0.0	2012-07-31	New York City	New York	
1	61.634838	9	0.1	2013-02-05	Wollongong	New South Wales	
2	919.971000	9	0.1	2013-10-17	Brisbane	Queensland	
3	61.634838	5	0.1	2013-01-28	Berlin	Berlin	
4	311.520000	8	0.0	2013-11-05	Dakar	Dakar	

	Region	Market	Order	Priority
0	East	US		Critical
1	Oceania	APAC		Critical
2	Oceania	APAC		Medium
3	Central	EU		Medium
4	Africa	Africa		Critical

3 Sales and Profit Analysis

1. What is the trend of sales and profit over time?
2. Which regions and markets contribute the most to sales and profit?
3. Which cities and states are the most profitable?
4. Which products are the most popular (highest quantity sold)?
5. What is the sales and profit distribution across different markets?

4 What is the trend of sales and profit over time?

```
[34]: segment_data.loc[:, 'Order Date'] = pd.to_datetime(segment_data['Order Date'])
segment_data.loc[:, 'Year'] = segment_data['Order Date'].dt.year
sales_profit_trend = segment_data.groupby('Year')[['Sales', 'Profit']].sum()
print(sales_profit_trend)
```

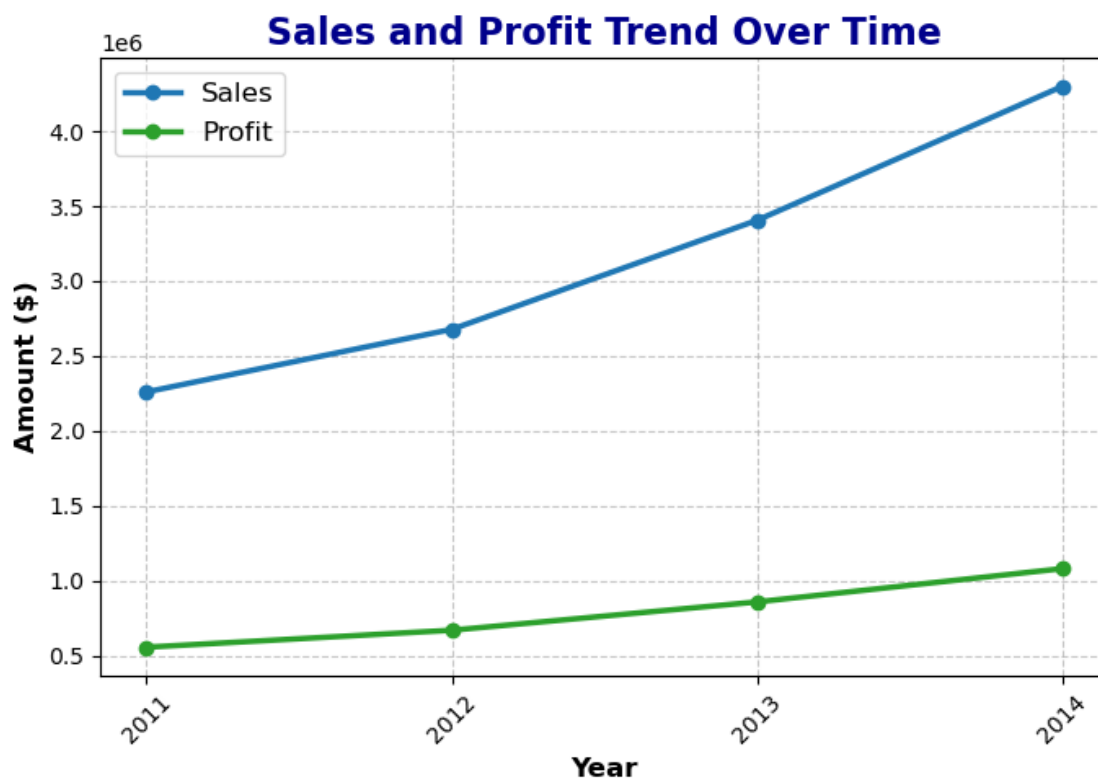
	Sales	Profit
Year		
2011	2.259451e+06	5.552016e+05
2012	2.677439e+06	6.684937e+05
2013	3.405746e+06	8.575934e+05
2014	4.299866e+06	1.079962e+06

C:\Users\Qammer Mehmood\AppData\Local\Temp\ipykernel_7208\2864655434.py:2:
SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using `.loc[row_indexer,col_indexer] = value` instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
`segment_data.loc[:, 'Year'] = segment_data['Order Date'].dt.year`

```
[35]: plt.figure(figsize=(7, 5))
plt.plot(sales_profit_trend.index, sales_profit_trend['Sales'], label='Sales',
        color='#1f77b4', linewidth=2.5, marker='o')
plt.plot(sales_profit_trend.index, sales_profit_trend['Profit'],
        label='Profit', color='#2ca02c', linewidth=2.5, marker='o')
plt.title('Sales and Profit Trend Over Time', fontsize=16, fontweight='bold',
        color='darkblue')
plt.xlabel('Year', fontsize=12, fontweight='bold')
plt.ylabel('Amount ($)', fontsize=12, fontweight='bold')
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(loc='upper left', fontsize=12)
plt.xticks(sales_profit_trend.index, rotation=45)
plt.tight_layout()
plt.show()
```



Sales and Profit Trend Report

The line chart shows a consistent upward trend in both Sales (represented by the blue line) and Profit (represented by the green line) over the years. Sales growth indicates increasing revenue, while the profit trend highlights improved profitability. Year-to-year fluctuations reflect market and operational impacts, but the overall trajectory suggests strong business performance.

5 Which regions and markets contribute the most to sales and profit?

```
[36]: region_market_sales_profit = segment_data.groupby(['Region', 'Market'],  
        ↪observed=False)[['Sales', 'Profit']].sum()  
top_region_sales = region_market_sales_profit.sort_values(by='Sales',  
        ↪ascending=False)  
top_market_sales = region_market_sales_profit.sort_values(by='Profit',  
        ↪ascending=False)
```

```
[37]: print("Top Regions by Sales:")  
top_region_sales[['Sales']].head()
```

Top Regions by Sales:

```
[37]:
```

Region	Market	Sales
Central	EU	1.720553e+06
Oceania	APAC	1.100185e+06
Southeast Asia	APAC	8.844232e+05
North Asia	APAC	8.483098e+05
EMEA	EMEA	8.061613e+05

```
[38]: print("\nTop Markets by Profit:")  
top_market_sales[['Profit']].head()
```

Top Markets by Profit:

```
[38]:
```

Region	Market	Profit
Central	EU	380670.456092
EMEA	EMEA	265949.434402
Oceania	APAC	241333.589821
Africa	Africa	241230.547508
Southeast Asia	APAC	210130.967780

```
[39]: # Sort the data for regions and markets by Sales and Profit  
top_regions_sales = region_market_sales_profit.sort_values(by='Sales',  
        ↪ascending=False).head()
```

```

top_markets_profit = region_market_sales_profit.sort_values(by='Profit',
    ↪ascending=False).head()

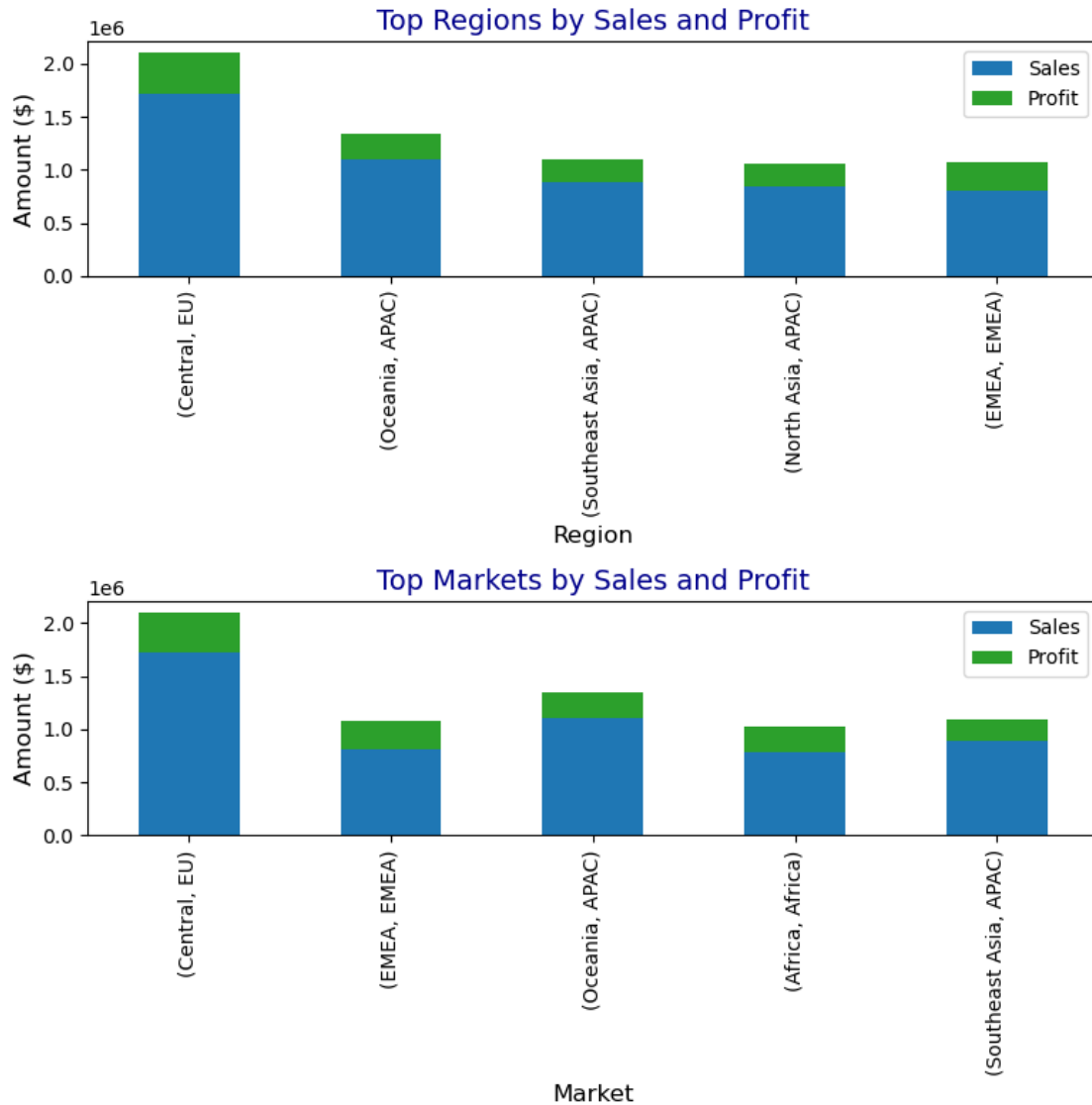
# Create subplots
fig, ax = plt.subplots(2, 1, figsize=(8, 8))

# Plotting Sales by Region (Stacked Bar Chart)
top_regions_sales[['Sales', 'Profit']].plot(kind='bar', stacked=True, ax=ax[0],
    ↪color=['#1f77b4', '#2ca02c'])
ax[0].set_title('Top Regions by Sales and Profit', fontsize=14,
    ↪color='darkblue')
ax[0].set_ylabel('Amount ($)', fontsize=12)
ax[0].set_xlabel('Region', fontsize=12)

# Plotting Profit by Market (Stacked Bar Chart)
top_markets_profit[['Sales', 'Profit']].plot(kind='bar', stacked=True,
    ↪ax=ax[1], color=['#1f77b4', '#2ca02c'])
ax[1].set_title('Top Markets by Sales and Profit', fontsize=14,
    ↪color='darkblue')
ax[1].set_ylabel('Amount ($)', fontsize=12)
ax[1].set_xlabel('Market', fontsize=12)

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

```

Most Profitable Regions and Markets

The analysis highlights the regions and markets with the highest contributions to both sales and profit. A stacked bar chart was used to compare sales and profit across different regions and markets. The blue bars represent sales, and the green bars represent profit, allowing easy visualization of the most profitable regions and markets. The chart clearly identifies the top-performing regions and markets, offering insights into key areas driving both sales and profit growth.

6 Which cities and states are the most profitable?

```
[40]: city_state_profit = segment_data.groupby(['City', 'State'])['Profit'].sum().
      ↪reset_index()

top_city_state_profit = city_state_profit.sort_values(by='Profit',
      ↪ascending=False).head()

top_city_state_profit
```

```
[40]:
```

	City	State	Profit
2402	New York City	New York	68468.399834
2006	Los Angeles	California	33891.576919
3085	Seattle	Washington	30527.425014
2096	Manila	National Capital	26143.423492
1580	Jakarta	Jakarta	25412.669240

```
[41]: # Sort the data for top cities and states by Profit
top_city_state_profit = city_state_profit.sort_values(by='Profit',
      ↪ascending=False).head()

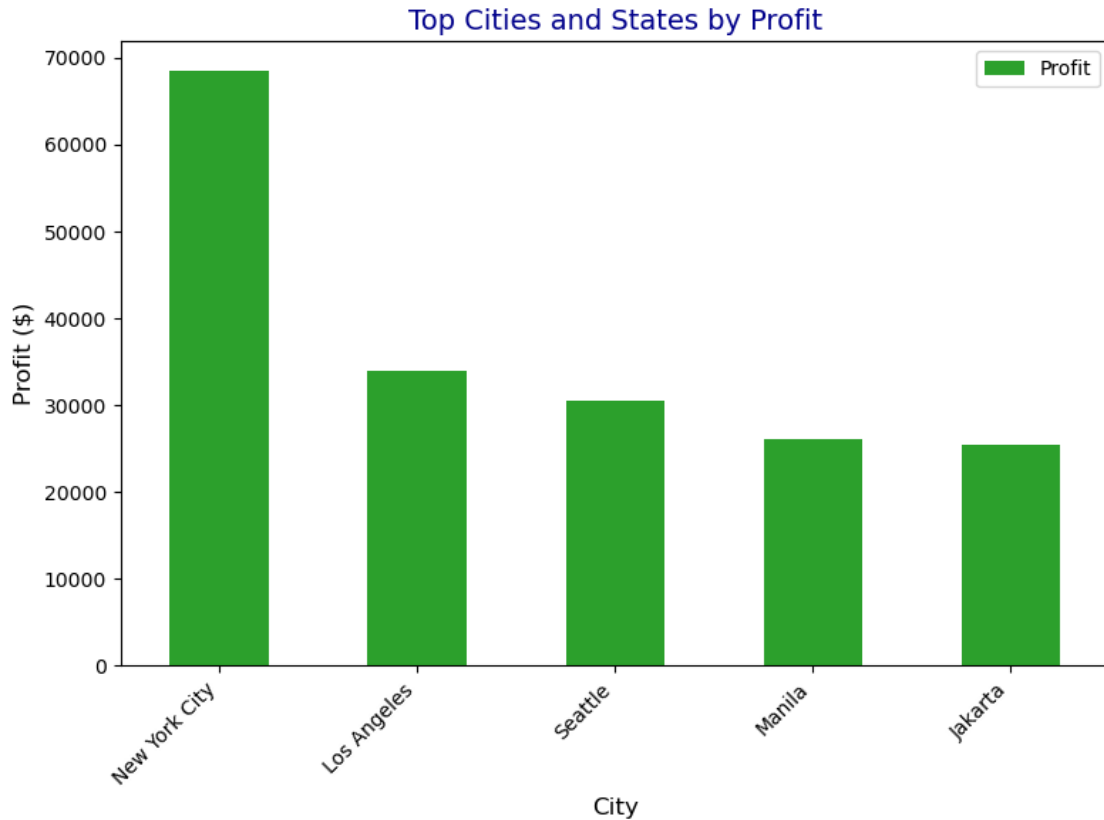
# Create the bar chart
fig, ax = plt.subplots(figsize=(8, 6))

# Plotting Profit by City and State (Bar Chart)
top_city_state_profit.plot(kind='bar', x='City', y='Profit', ax=ax,
      ↪color='#2ca02c')

# Set the title and labels
ax.set_title('Top Cities and States by Profit', fontsize=14, color='darkblue')
ax.set_ylabel('Profit ($)', fontsize=12)
ax.set_xlabel('City', fontsize=12)

# Rotate the x-axis labels for better readability
plt.xticks(rotation=45, ha='right')

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



Report: Most Profitable Cities and States

The analysis identifies the top cities and states contributing the most to profit. The horizontal bar chart showcases the most profitable city-state pairs, with higher profit values clearly visible. Cities with the highest profits stand out, providing a quick comparison of profitability. The green bars represent total profit, offering insights into key areas of business performance.

7 Which products are the most popular (highest quantity sold)?

```
[42]: # Grouping the data by Product Name and summing the Quantity
product_quantity = segment_data.groupby('Product Name')['Quantity'].sum().
    ↪reset_index()

# Sorting the data by Quantity in descending order to get the most popular
    ↪products
top_products = product_quantity.sort_values(by='Quantity', ascending=False).
    ↪head()

# Display the result
top_products
```

```
[42]:
```

	Product Name	Quantity
3275	Staples	876
894	Cardinal Index Tab, Clear	337
1210	Eldon File Cart, Single Width	321
2840	Rogers File Cart, Single Width	262
3070	Sanford Pencil Sharpener, Water Color	259

```
[43]: top_products = product_quantity.sort_values(by='Quantity', ascending=False).
      ↪head()

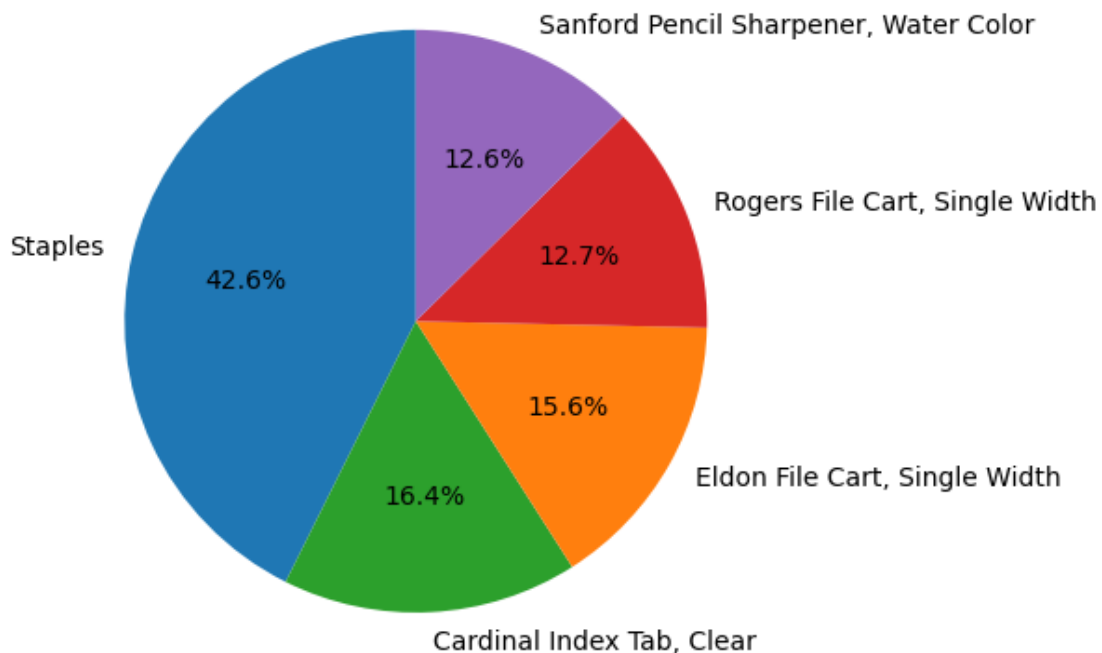
fig, ax = plt.subplots(figsize=(6, 6))

ax.pie(top_products['Quantity'], labels=top_products['Product Name'],
      ↪autopct='%1.1f%%', colors=['#1f77b4', '#2ca02c', '#ff7f0e', '#d62728',
      ↪'#9467bd'], startangle=90)

ax.set_title('Top Products by Quantity Sold', fontsize=14, color='darkblue')

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

Top Products by Quantity Sold



Report: Most Popular Products by Quantity Sold

The analysis highlights the top products based on the quantity sold. A pie chart was used to display the proportion of each product's sales, with each product represented by a different color. The blue segment represents the most popular product, while the green and other colors represent the subsequent top products.

This chart effectively showcases the share of each product in the total quantity sold, providing a clear view of the most popular products in the dataset. The visual representation allows easy comparison of product popularity in terms of volume.

8 What is the sales and profit distribution across different markets?

```
[44]: # Grouping the data by Market and summing Sales and Profit with observed=False
      ↪to avoid the warning
market_sales_profit = segment_data.groupby('Market', observed=False)[['Sales',
      ↪'Profit']].sum().reset_index()

# Display the result
market_sales_profit
```

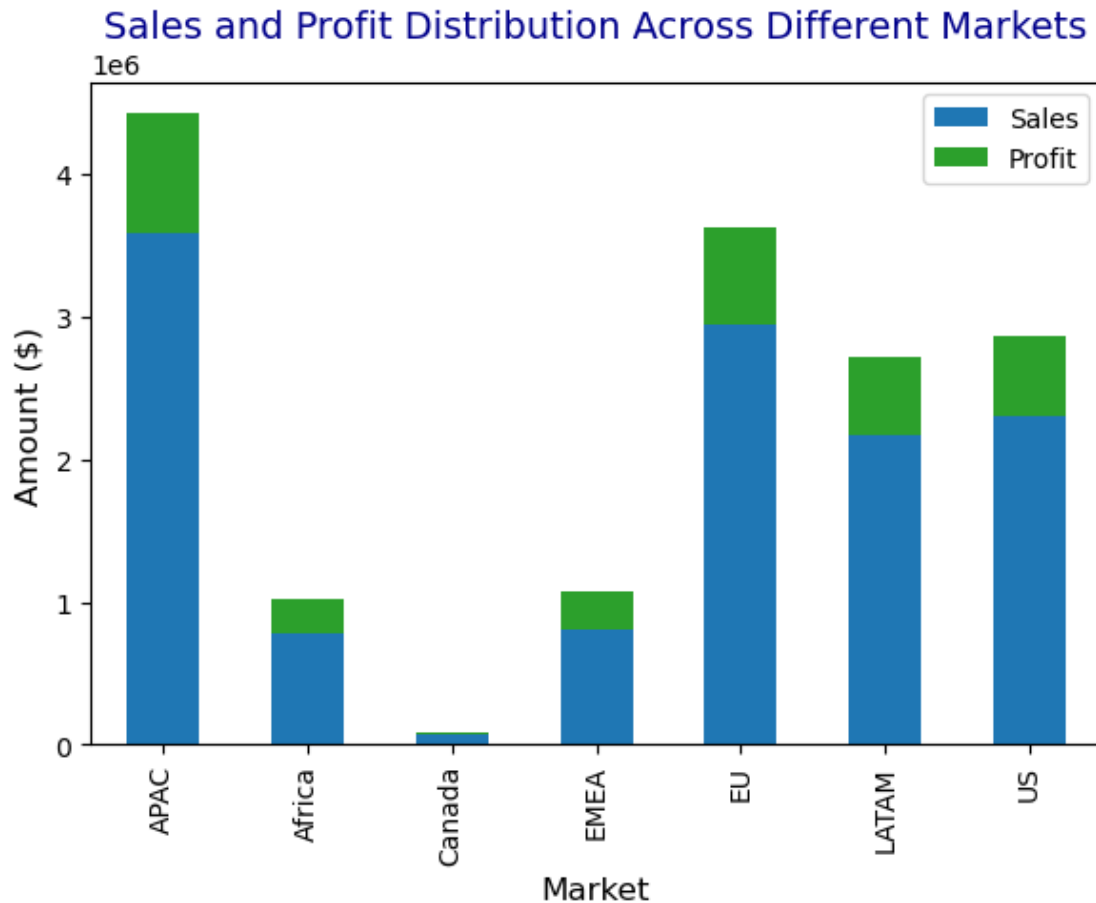
```
[44]:   Market      Sales      Profit
0   APAC  3.585744e+06  835093.627750
1  Africa  7.837732e+05  241230.547508
2  Canada  6.692817e+04   17817.390000
3   EMEA  8.061613e+05  265949.434402
4    EU   2.938089e+06  689561.324641
5  LATAM  2.164605e+06  553751.445089
6    US   2.297201e+06  557847.089967
```

```
[45]: # Create the bar chart for Sales and Profit by Market
fig, ax = plt.subplots(figsize=(6, 5))

# Plotting Sales and Profit for each Market as a stacked bar chart
market_sales_profit.set_index('Market')[['Sales', 'Profit']].plot(kind='bar',
      ↪stacked=True, ax=ax, color=['#1f77b4', '#2ca02c'])

# Set the title and labels
ax.set_title('Sales and Profit Distribution Across Different Markets',
      ↪fontsize=14, color='darkblue')
ax.set_ylabel('Amount ($)', fontsize=12)
ax.set_xlabel('Market', fontsize=12)

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



Report: Sales and Profit Distribution Across Different Markets The analysis of sales and profit distribution across different markets was visualized using a stacked bar chart. The chart displays Sales in blue and Profit in green for each market. By stacking these values, it becomes easy to compare the total contribution of sales and profit within each market.

The chart highlights which markets are the most profitable and which generate the highest sales. Markets with larger bars indicate higher overall sales and profit, while smaller bars suggest lower contributions in these areas. This visualization allows for a clear, comparative understanding of market performance in terms of both sales and profitability.

[]: