hvdwoxd37

July 26, 2024

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

I possess XYZ Company's sales data. The required libraries must be imported in order to perform data analysis.

[2]:
```
df = pd.read_csv("Sales Data.csv",encoding = "latin-1")
```

importing our csv file into our jupyter environment

[3]:
```
df
```

[3]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	\
0	1	US-2019-103800	1/3/2019	1/7/2019	Standard Class	
1	2	US-2019-112326	1/4/2019	1/8/2019	Standard Class	
2	3	US-2019-112326	1/4/2019	1/8/2019	Standard Class	
3	4	US-2019-112326	1/4/2019	1/8/2019	Standard Class	
4	5	US-2019-141817	1/5/2019	1/12/2019	Standard Class	
…	…	…	…	…	…	
10189	10190	US-2022-143259	12/30/2022	1/3/2023	Standard Class	
10190	10191	US-2022-115427	12/30/2022	1/3/2023	Standard Class	
10191	10192	US-2022-156720	12/30/2022	1/3/2023	Standard Class	
10192	10193	US-2022-143259	12/30/2022	1/3/2023	Standard Class	
10193	10194	CA-2022-143500	12/30/2022	1/3/2023	Standard Class	

	Customer ID	Customer Name	Segment	Country/Region	\
0	DP-13000	Darren Powers	Consumer	United States	
1	PO-19195	Phillina Ober	Home Office	United States	
2	PO-19195	Phillina Ober	Home Office	United States	
3	PO-19195	Phillina Ober	Home Office	United States	
4	MB-18085	Mick Brown	Consumer	United States	
…	…	…	…	…	
10189	PO-18865	Patrick O'Donnell	Consumer	United States	
10190	EB-13975	Erica Bern	Corporate	United States	
10191	JM-15580	Jill Matthias	Consumer	United States	
10192	PO-18865	Patrick O'Donnell	Consumer	United States	

```
10193    HO-15230        Harry Olson     Consumer         Canada

                City  … Postal Code   Region       Product ID  \
0            Houston  …       77095  Central  OFF-PA-10000174
1         Naperville  …       60540  Central  OFF-BI-10004094
2         Naperville  …       60540  Central  OFF-LA-10003223
3         Naperville  …       60540  Central  OFF-ST-10002743
4       Philadelphia  …       19143     East  OFF-AR-10003478
…                  …  …           …        …              …
10189  New York City  …       10009     East  OFF-BI-10003684
10190      Fairfield  …       94533     West  OFF-BI-10004632
10191       Loveland  …       80538     West  OFF-FA-10003472
10192  New York City  …       10009     East  TEC-PH-10004774
10193  Charlottetown  …         C0A     East  OFF-BI-10004040

              Category Sub-Category  \
0      Office Supplies        Paper
1      Office Supplies      Binders
2      Office Supplies       Labels
3      Office Supplies      Storage
4      Office Supplies          Art
…                  …           …
10189  Office Supplies      Binders
10190  Office Supplies      Binders
10191  Office Supplies    Fasteners
10192       Technology       Phones
10193  Office Supplies      Binders

                                          Product Name    Sales  Quantity  \
0      Message Book, Wirebound, Four 5 1/2" X 4" Form…   16.448         2
1             GBC Standard Plastic Binding Systems Combs    3.540         2
2                                           Avery 508   11.784         3
3                       SAFCO Boltless Steel Shelving  272.736         3
4      Avery Hi-Liter EverBold Pen Style Fluorescent …   19.536         3
…                                                   …       …       …
10189              Wilson Jones Legal Size Ring Binders   52.776         3
10190                                GBC Binding covers   20.720         2
10191                               Bagged Rubber Bands    3.024         3
10192                          Gear Head AU3700S Headset   90.930         7
10193                        Wilson Jones Impact Binders    3.024         3

       Discount   Profit
0           0.2   5.5512
1           0.8  -5.4870
2           0.2   4.2717
3           0.2 -64.7748
4           0.2   4.8840
```

```
…          …       …
10189       0.2  19.7910
10190       0.2   6.4750
10191       0.2  -0.6048
10192       0.0   2.7279
10193       0.2  -0.6048

[10194 rows x 21 columns]
```

carrying out the required checks in order to properly analyze data

[4]:
```
df.columns
```

[4]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country/Region', 'City',
       'State/Province', 'Postal Code', 'Region', 'Product ID', 'Category',
       'Sub-Category', 'Product Name', 'Sales', 'Quantity', 'Discount',
       'Profit'],
      dtype='object')
```

[5]:
```
df.head
```

[5]:
```
<bound method NDFrame.head of        Row ID        Order ID  Order Date  Ship
Date       Ship Mode  \
0           1  US-2019-103800    1/3/2019   1/7/2019  Standard Class
1           2  US-2019-112326    1/4/2019   1/8/2019  Standard Class
2           3  US-2019-112326    1/4/2019   1/8/2019  Standard Class
3           4  US-2019-112326    1/4/2019   1/8/2019  Standard Class
4           5  US-2019-141817    1/5/2019  1/12/2019  Standard Class
…       …              …         …        …               …
10189   10190  US-2022-143259  12/30/2022   1/3/2023  Standard Class
10190   10191  US-2022-115427  12/30/2022   1/3/2023  Standard Class
10191   10192  US-2022-156720  12/30/2022   1/3/2023  Standard Class
10192   10193  US-2022-143259  12/30/2022   1/3/2023  Standard Class
10193   10194  CA-2022-143500  12/30/2022   1/3/2023  Standard Class

      Customer ID      Customer Name      Segment Country/Region  \
0        DP-13000      Darren Powers     Consumer  United States
1        PO-19195      Phillina Ober  Home Office  United States
2        PO-19195      Phillina Ober  Home Office  United States
3        PO-19195      Phillina Ober  Home Office  United States
4        MB-18085         Mick Brown     Consumer  United States
…           …                …           …             …
10189    PO-18865  Patrick O'Donnell     Consumer  United States
10190    EB-13975         Erica Bern    Corporate  United States
10191    JM-15580      Jill Matthias     Consumer  United States
10192    PO-18865  Patrick O'Donnell     Consumer  United States
```

```
10193    HO-15230        Harry Olson     Consumer         Canada

                City  … Postal Code   Region       Product ID  \
0            Houston  …       77095  Central  OFF-PA-10000174
1         Naperville  …       60540  Central  OFF-BI-10004094
2         Naperville  …       60540  Central  OFF-LA-10003223
3         Naperville  …       60540  Central  OFF-ST-10002743
4       Philadelphia  …       19143     East  OFF-AR-10003478
…                  …  …           …        …              …
10189  New York City  …       10009     East  OFF-BI-10003684
10190      Fairfield  …       94533     West  OFF-BI-10004632
10191       Loveland  …       80538     West  OFF-FA-10003472
10192  New York City  …       10009     East  TEC-PH-10004774
10193  Charlottetown  …         C0A     East  OFF-BI-10004040

              Category Sub-Category  \
0      Office Supplies        Paper
1      Office Supplies      Binders
2      Office Supplies       Labels
3      Office Supplies      Storage
4      Office Supplies          Art
…                  …           …
10189  Office Supplies      Binders
10190  Office Supplies      Binders
10191  Office Supplies    Fasteners
10192       Technology       Phones
10193  Office Supplies      Binders

                                          Product Name    Sales  Quantity  \
0      Message Book, Wirebound, Four 5 1/2" X 4" Form…   16.448         2
1             GBC Standard Plastic Binding Systems Combs    3.540         2
2                                           Avery 508   11.784         3
3                       SAFCO Boltless Steel Shelving  272.736         3
4      Avery Hi-Liter EverBold Pen Style Fluorescent …   19.536         3
…                                                   …       …       …
10189              Wilson Jones Legal Size Ring Binders   52.776         3
10190                                GBC Binding covers   20.720         2
10191                               Bagged Rubber Bands    3.024         3
10192                          Gear Head AU3700S Headset   90.930         7
10193                        Wilson Jones Impact Binders    3.024         3

       Discount   Profit
0           0.2   5.5512
1           0.8  -5.4870
2           0.2   4.2717
3           0.2 -64.7748
4           0.2   4.8840
```

```
…          …       …
10189       0.2  19.7910
10190       0.2   6.4750
10191       0.2  -0.6048
10192       0.0   2.7279
10193       0.2  -0.6048

[10194 rows x 21 columns]>
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10194 entries, 0 to 10193
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          10194 non-null  int64
 1   Order ID        10194 non-null  object
 2   Order Date      10194 non-null  object
 3   Ship Date       10194 non-null  object
 4   Ship Mode       10194 non-null  object
 5   Customer ID     10194 non-null  object
 6   Customer Name   10194 non-null  object
 7   Segment         10194 non-null  object
 8   Country/Region  10194 non-null  object
 9   City            10194 non-null  object
 10  State/Province  10194 non-null  object
 11  Postal Code     10194 non-null  object
 12  Region          10194 non-null  object
 13  Product ID      10194 non-null  object
 14  Category        10194 non-null  object
 15  Sub-Category    10194 non-null  object
 16  Product Name    10194 non-null  object
 17  Sales           10194 non-null  float64
 18  Quantity        10194 non-null  int64
 19  Discount        10194 non-null  float64
 20  Profit          10194 non-null  float64
dtypes: float64(3), int64(2), object(16)
memory usage: 1.6+ MB
```

In order to check for outliers, we describe the data

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

[7]:

	Row ID	Sales	Quantity	Discount	Profit
count	10194.000000	10194.000000	10194.000000	10194.000000	10194.000000
mean	5097.500000	228.225854	3.791838	0.155385	28.673417
std	2942.898656	619.906839	2.228317	0.206249	232.465115

```
min        1.000000      0.444000      1.000000      0.000000  -6599.978000
25%     2549.250000     17.220000      2.000000      0.000000      1.760800
50%     5097.500000     53.910000      3.000000      0.200000      8.690000
75%     7645.750000    209.500000      5.000000      0.200000     29.297925
max    10194.000000  22638.480000     14.000000      0.800000   8399.976000
```

a few sales quantity, discount, and profit anomalies

[8]:
```
df.shape
```

[8]:
```
(10194, 21)
```

[9]:
```
df.describe(include = "object")
```

[9]:
```
              Order ID Order Date   Ship Date       Ship Mode Customer ID  \
count            10194      10194       10194           10194       10194
unique            5111       1242        1338               4         804
top     US-2022-100111   9/5/2021  12/16/2020  Standard Class    WB-21850
freq                14         38          38            6120          41

        Customer Name   Segment Country/Region           City State/Province  \
count           10194     10194          10194          10194          10194
unique            800         3              2            542             59
top     William Brown  Consumer  United States  New York City     California
freq               41      5281           9994            915           2001

       Postal Code Region       Product ID         Category Sub-Category  \
count        10194  10194            10194            10194        10194
unique         654      4             1862                3           17
top          10035   West  FUR-FU-10004270  Office Supplies      Binders
freq           263   3253               20             6128         1548

       Product Name
count         10194
unique         1849
top         Staples
freq             50
```

conward the Order_Date & Ship_Date to datetime formate

[10]:
```
df['Order Date']= pd.to_datetime(df['Order Date'])
```

[11]:
```
df['Ship Date']= pd.to_datetime(df['Ship Date'])
```

[12]:
```
df.info()
```

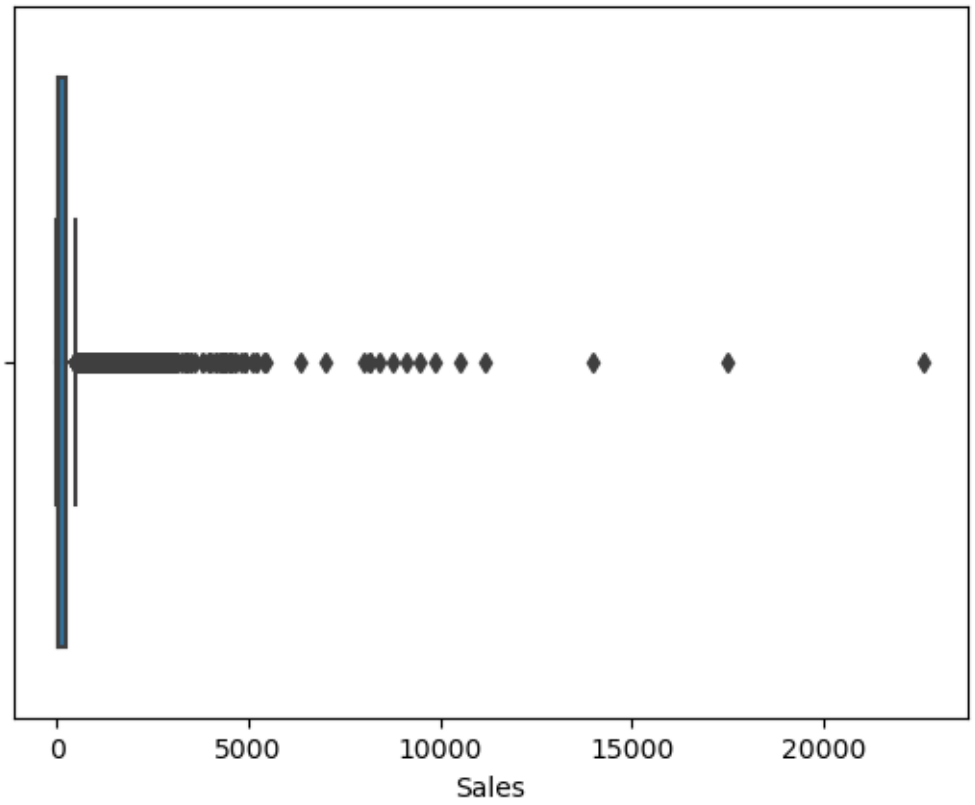
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10194 entries, 0 to 10193
```

```
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          10194 non-null  int64
 1   Order ID        10194 non-null  object
 2   Order Date      10194 non-null  datetime64[ns]
 3   Ship Date       10194 non-null  datetime64[ns]
 4   Ship Mode       10194 non-null  object
 5   Customer ID     10194 non-null  object
 6   Customer Name   10194 non-null  object
 7   Segment         10194 non-null  object
 8   Country/Region  10194 non-null  object
 9   City            10194 non-null  object
 10  State/Province  10194 non-null  object
 11  Postal Code     10194 non-null  object
 12  Region          10194 non-null  object
 13  Product ID      10194 non-null  object
 14  Category        10194 non-null  object
 15  Sub-Category    10194 non-null  object
 16  Product Name    10194 non-null  object
 17  Sales           10194 non-null  float64
 18  Quantity        10194 non-null  int64
 19  Discount        10194 non-null  float64
 20  Profit          10194 non-null  float64
dtypes: datetime64[ns](2), float64(3), int64(2), object(14)
memory usage: 1.6+ MB
```

creat a Box plot of Sales for look outliers

[13]:
```
sns.boxplot(x=df['Sales'])
plt.figsize= ((8,6))
xlabel = "sales"
plt.title = "box plot of sales"
plt.show
```

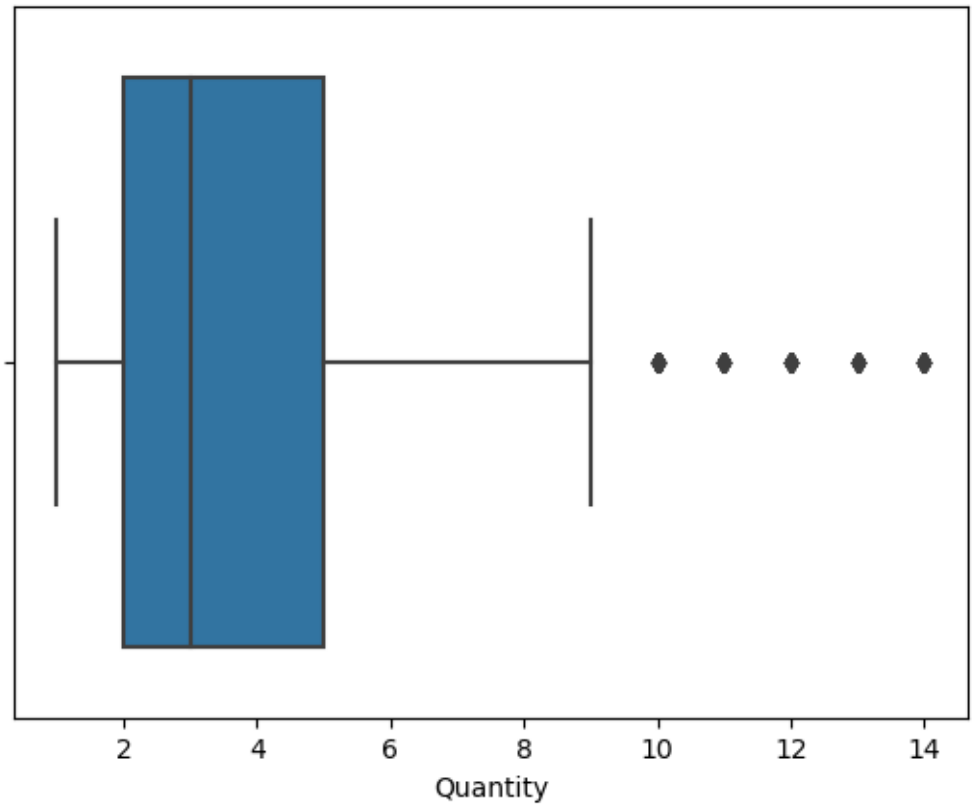
[13]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



#the graph shows that most of the sales data are concentrated at lower values, with a significant number of outliers at much higher sales values, indicating a highly skewed distribution with some very high sales figures.

[14]:
```
plt.figsize= ((8,6))
sns.boxplot(x=df['Quantity'])
plt.title = "box plot of Quantity"
xlabel = "Quantity"
plt.show
```

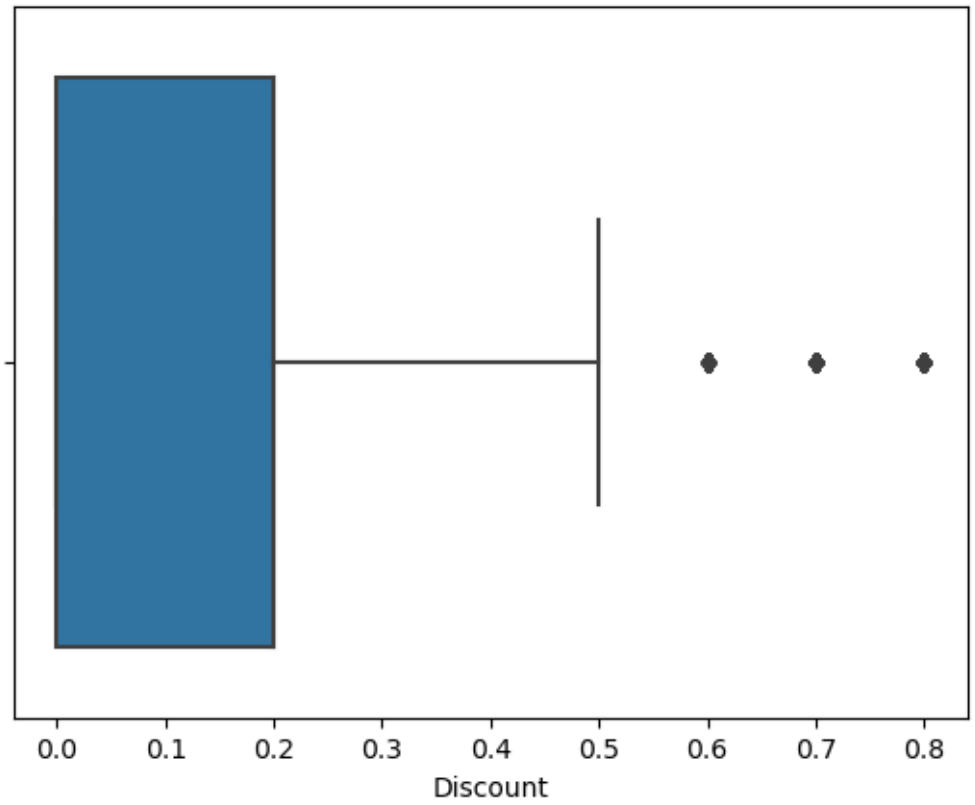
[14]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



#Most of the quantity data falls within the whiskers, which show the typical range. The median (middle value) is near the center of the box. Outliers are the points beyond the upper whisker, indicating unusually high quantities. This box plot shows the central tendency, spread, and outliers in the quantity data.

[15]:
```
plt.figsize= ((8,6))
sns.boxplot(x=df['Discount'])
plt.title = "box plot of Discount"
xlabel = "Discount"
plt.show
```

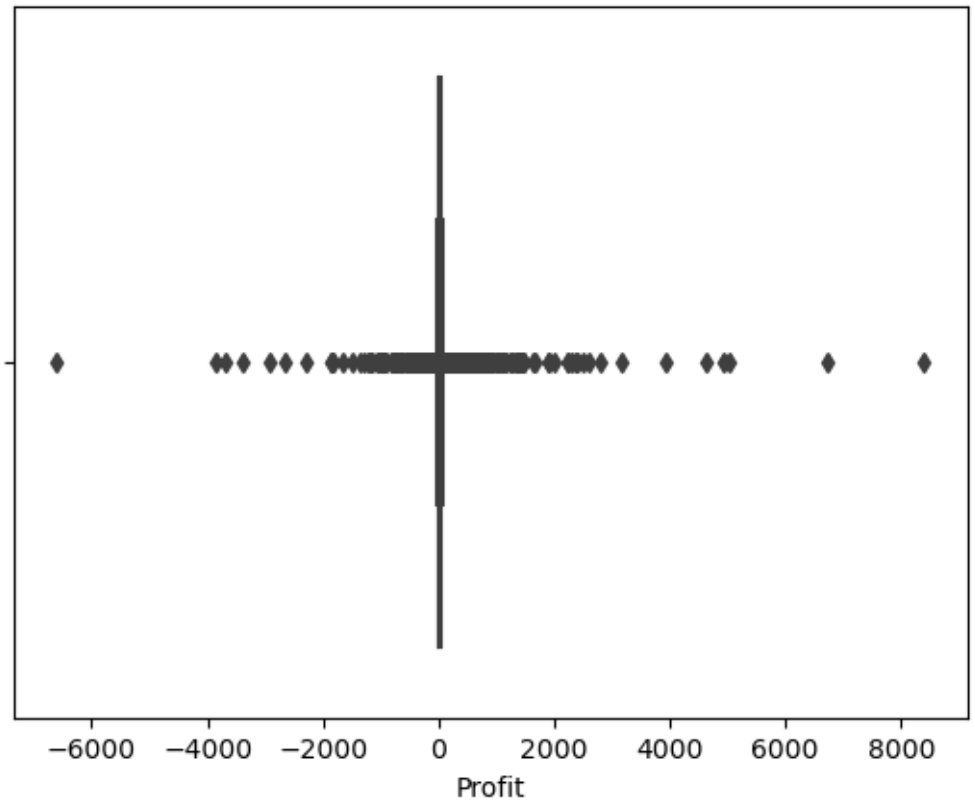
[15]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



#Most discount values are between 0.05 and 0.2. The median discount is around 0.1. A few higher discounts (outliers) extend up to 0.8. This box plot highlights the typical discount range, the central discount value, and the presence of a few high discounts.

[16]:
```
plt.figsize= ((8,6))
sns.boxplot(x=df['Profit'])
plt.title = "box plot of Profit"
xlabel = "Profit"
plt.show
```

[16]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



#Scatter plots are like dot maps for numbers. Dots move right for bigger values on one measure (like profit), and up for bigger values on another (like time). By seeing where the dots cluster, we can see if the two things change together.

[17]:
```
df.describe()
```

[17]:

	Row ID	Order Date	\
count	10194.000000	10194	
mean	5097.500000	2021-04-29 11:48:25.002942976	
min	1.000000	2019-01-03 00:00:00	
25%	2549.250000	2020-05-14 00:00:00	
50%	5097.500000	2021-06-25 00:00:00	
75%	7645.750000	2022-05-14 00:00:00	
max	10194.000000	2022-12-30 00:00:00	
std	2942.898656	NaN	

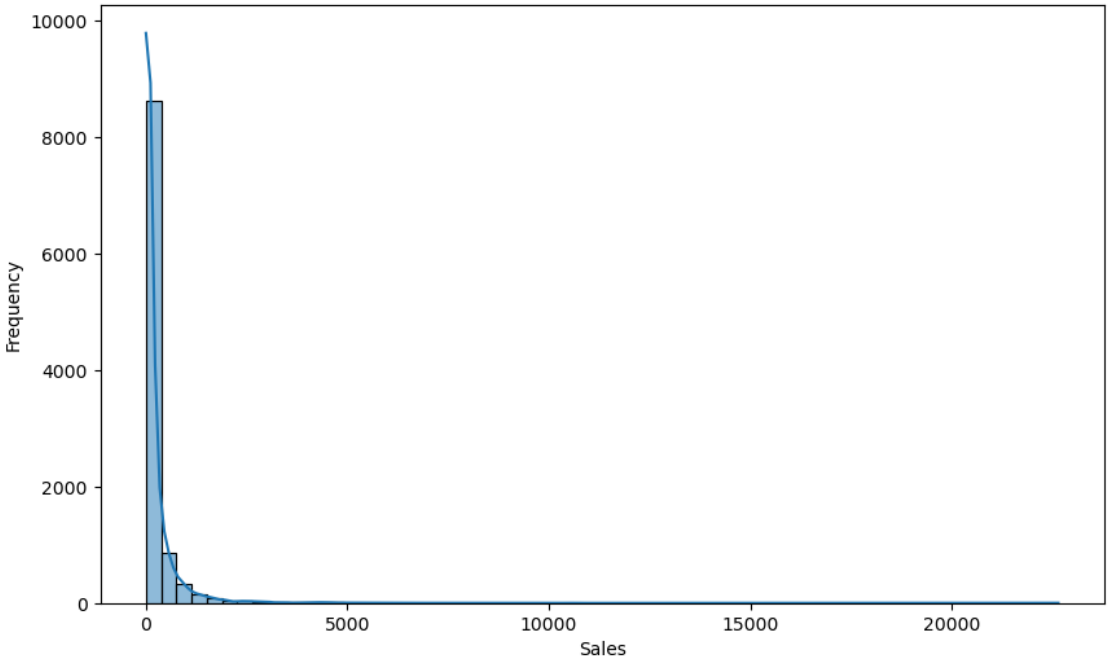
	Ship Date	Sales	Quantity	\
count	10194	10194.000000	10194.000000	
mean	2021-05-03 10:52:45.626839296	228.225854	3.791838	
min	2019-01-07 00:00:00	0.444000	1.000000	
25%	2020-05-19 00:00:00	17.220000	2.000000	

```
50%                  2021-06-28 00:00:00     53.910000      3.000000
75%                  2022-05-18 00:00:00    209.500000      5.000000
max                  2023-01-05 00:00:00  22638.480000     14.000000
std                                  NaN    619.906839      2.228317

           Discount        Profit
count  10194.000000  10194.000000
mean       0.155385     28.673417
min        0.000000  -6599.978000
25%        0.000000      1.760800
50%        0.200000      8.690000
75%        0.200000     29.297925
max        0.800000   8399.976000
std        0.206249    232.465115
```

[18]:
```
#df['Sales'] = df['sales'].astype(float)
```

[19]:
```
plt.figure(figsize=(10,6))
sns.histplot(df['Sales'], bins = 60, kde = True)
#plt.title('Distribution of Sales')
plt.title#('Distribution of sales')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.show()
```

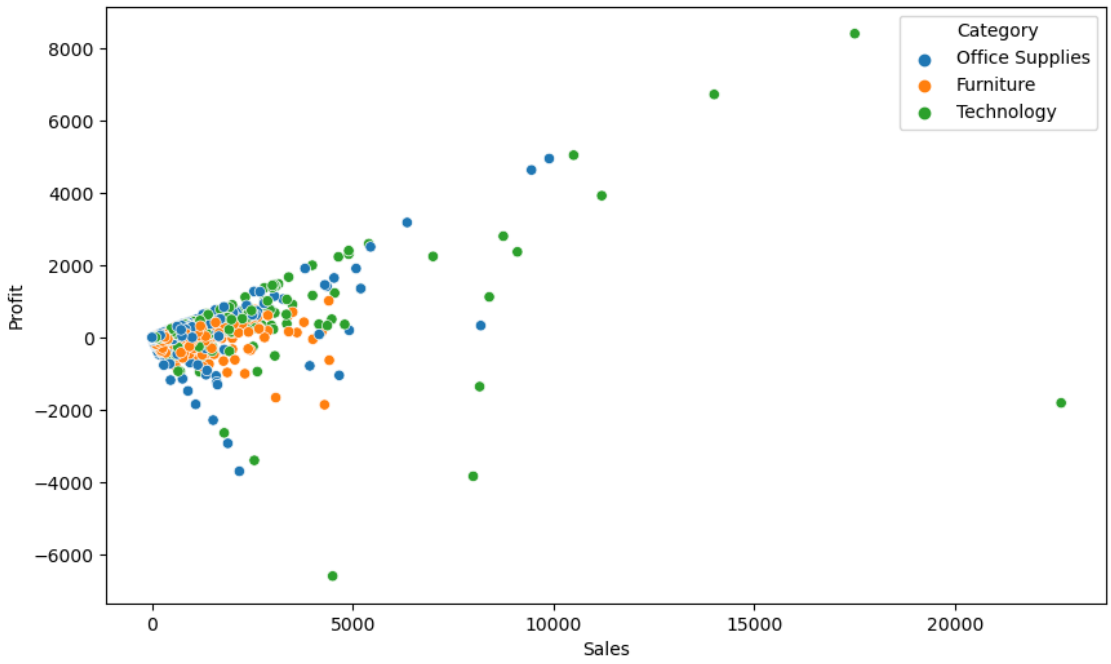


#The graph shows a breakdown of sales amounts. The horizontal axis shows the sales value (e.g., $2,000 to $22,000), and the vertical axis shows how many sales fall within each price range. The most frequent sales amounts are clustered around a specific range (likely $8,000 to $10,000). The smooth line helps visualize the overall distribution of the sales data.

[20]:
```
type('sales')
```

[20]:
```
str
```

[21]:
```
#relationship between sales and profit
plt.figure(figsize=(10,6))
sns.scatterplot(x= 'Sales', y ='Profit', data=df, hue ='Category')
plt.show()
```



[22]:
```
import plotly.express as px
#example interactive plot
fig = px.scatter(df, x ='Sales', y ='Profit', color='Category',␣
 ↪hover_data=['Product Name'])
fig.update_layout(title='interactive Sales vs Profit')
fig.show()
```

#The graph shows a breakdown of sales amounts. The horizontal axis shows the sales value (e.g., $2,000 to $22,000), and the vertical axis shows how many sales fall within each price range. The most sales are clustered around $8,000 to $10,000. The smooth line helps visualize the overall distribution of the sales data.

[23]:
```
df.isnull().any()
```

[23]:
```
Row ID           False
Order ID         False
Order Date       False
Ship Date        False
Ship Mode        False
Customer ID      False
Customer Name    False
Segment          False
Country/Region   False
City             False
State/Province   False
Postal Code      False
Region           False
Product ID       False
Category         False
Sub-Category     False
Product Name     False
Sales            False
Quantity         False
Discount         False
Profit           False
dtype: bool
```

[24]:
```
df.isnull().sum()
```

[24]:
```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country/Region   0
City             0
State/Province   0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
```

```
Profit            0
dtype: int64
```

[25]:
```
#list of categories columns (object)
categorical_columns = df.select_dtypes(include=['object']).columns

#display uniques values from categorical_columns
for col in categorical_columns:
    print(f"Unique values in '{col}':")
    print(df[col].unique())
    print("\n" + "-"*40 + "\n")
```

```
Unique values in 'Order ID':
['US-2019-103800' 'US-2019-112326' 'US-2019-141817' … 'US-2022-115427'
 'US-2022-156720' 'CA-2022-143500']

----------------------------------------

Unique values in 'Ship Mode':
['Standard Class' 'First Class' 'Second Class' 'Same Day']

----------------------------------------

Unique values in 'Customer ID':
['DP-13000' 'PO-19195' 'MB-18085' 'ME-17320' 'JO-15145' 'LS-17230'
 'VS-21820' 'MS-17830' 'AJ-10780' 'SV-20365' 'BD-11605' 'ND-18370'
 'MM-17920' 'CS-12250' 'BS-11590' 'EH-13990' 'DL-13315' 'DW-13195'
 'TS-21340' 'HL-15040' 'XP-21865' 'MV-17485' 'MM-18280' 'TB-21400'
 'JW-15220' 'SG-20605' 'IM-15055' 'CA-11965' 'SD-20485' 'EJ-13720'
 'JC-15340' 'MV-18190' 'LC-17050' 'BD-11500' 'EB-13930' 'CD-12790'
 'MH-17440' 'DB-13270' 'ND-18460' 'CK-12760' 'NM-18445' 'GA-14725'
 'NF-18385' 'SC-20095' 'ML-17395' 'ST-20530' 'BF-11020' 'MN-17935'
 'TB-21595' 'AB-10015' 'LC-16930' 'SR-20740' 'TM-21010' 'SA-20830'
 'MG-17875' 'JH-15430' 'JS-16030' 'VF-21715' 'SC-20380' 'DB-13060'
 'GW-14605' 'HR-14770' 'KN-16705' 'NH-18610' 'JS-15595' 'EJ-14155'
 'SC-20020' 'AJ-10945' 'AP-10720' 'CL-12565' 'AB-10150' 'CM-12715'
 'NP-18685' 'KT-16480' 'SW-20275' 'CK-12205' 'FM-14215' 'RB-19435'
 'KA-16525' 'NF-18595' 'CK-12325' 'DK-13225' 'JF-15295' 'NF-18475'
 'RB-19465' 'AS-10240' 'CP-12340' 'BN-11515' 'AZ-10750' 'QJ-19255'
 'EK-13795' 'SM-20320' 'SC-20050' 'MC-17605' 'GM-14440' 'KH-16690'
 'CB-12025' 'TS-21205' 'BF-11275' 'VM-21835' 'CC-12685' 'BG-11740'
 'PO-18865' 'MS-17710' 'CD-11920' 'CA-12265' 'DK-12835' 'CS-12505'
 'JD-15895' 'MP-18175' 'CV-12295' 'DR-12940' 'MG-18145' 'MP-17470'
 'SG-20890' 'KM-16720' 'Dl-13600' 'NC-18340' 'TG-21640' 'AA-10315'
 'KB-16240' 'DD-13570' 'JO-15280' 'KH-16330' 'AH-10690' 'RD-19585'
 'KE-16420' 'MZ-17515' 'PK-18910' 'KB-16585' 'GM-14695' 'RD-19900'
 'RP-19855' 'TH-21550' 'JG-15805' 'AY-10555' 'SC-20260' 'FH-14275'
```

'GP-14740' 'PJ-19015' 'RS-19765' 'VM-21685' 'RA-19885' 'JS-15880'
'JM-15655' 'PC-19000' 'SE-20110' 'JK-15370' 'MM-18055' 'GM-14455'
'KN-16450' 'EH-13765' 'SV-20785' 'BF-11170' 'PS-18970' 'SP-20650'
'JL-15835' 'AG-10390' 'BP-11230' 'AA-10375' 'DB-13555' 'AH-10030'
'LB-16795' 'TS-21160' 'BT-11305' 'DA-13450' 'LC-16885' 'AG-10300'
'MH-17290' 'MY-18295' 'GB-14530' 'DS-13180' 'AI-10855' 'LA-16780'
'HG-14845' 'LL-16840' 'KH-16360' 'JM-16195' 'AR-10510' 'CS-11860'
'NC-18415' 'JS-15940' 'AA-10480' 'BS-11365' 'PM-18940' 'EB-13870'
'PF-19120' 'GD-14590' 'SN-20710' 'HP-14815' 'CT-11995' 'SB-20290'
'PG-18820' 'AF-10885' 'GM-14680' 'VW-21775' 'PG-18895' 'BG-11695'
'HD-14785' 'RM-19375' 'EH-14005' 'AG-10270' 'NP-18325' 'AJ-10795'
'PW-19030' 'CC-12475' 'PA-19060' 'TT-21070' 'GA-14515' 'BS-11755'
'JH-15820' 'TD-20995' 'RF-19345' 'MC-17590' 'CC-12430' 'FG-14260'
'BT-11530' 'MW-18235' 'EH-14185' 'GT-14710' 'RP-19390' 'JL-15235'
'BW-11065' 'HF-14995' 'Dp-13240' 'MH-17785' 'LE-16810' 'MM-17260'
'PJ-18835' 'SS-20410' 'LW-16825' 'JG-15160' 'EB-13840' 'MC-17425'
'JK-15625' 'DW-13480' 'LT-17110' 'CR-12625' 'RE-19450' 'SC-20725'
'RB-19795' 'BT-11440' 'GT-14635' 'BM-11785' 'SB-20170' 'VT-21700'
'NR-18550' 'BP-11095' 'CS-11950' 'RD-19480' 'BH-11710' 'FO-14305'
'CW-11905' 'CM-12445' 'HK-14890' 'SR-20425' 'KL-16555' 'PS-18760'
'SC-20575' 'GH-14425' 'JF-15415' 'DB-13660' 'MC-18100' 'NC-19470'
'Co-12640' 'NW-18400' 'TB-21280' 'AA-10645' 'DV-13465' 'JE-15745'
'DL-13330' 'BS-11665' 'LP-17080' 'NS-18640' 'GK-14620' 'LM-17065'
'CA-12775' 'RW-19630' 'DL-12865' 'AR-10825' 'RA-19915' 'HA-14905'
'MS-17980' 'BD-11620' 'EH-13945' 'MS-17770' 'DM-13525' 'DP-13390'
'MC-17845' 'DC-12850' 'BC-11125' 'EP-13915' 'MH-18025' 'EB-13705'
'DS-13030' 'GH-14410' 'BM-11650' 'MH-17455' 'JK-15205' 'PV-18985'
'AC-10420' 'ME-17725' 'SS-20515' 'BF-10975' 'AW-10930' 'PB-19150'
'AH-10120' 'JL-15130' 'CA-12310' 'EM-14140' 'KM-16375' 'CC-12145'
'SW-20245' 'RD-19720' 'ME-18010' 'HM-14860' 'EM-14065' 'AB-10255'
'JL-15850' 'AG-10900' 'NC-18535' 'KL-16645' 'SG-20470' 'LF-17185'
'CL-11890' 'AS-10045' 'BB-10990' 'TS-21430' 'IL-15100' 'AS-10630'
'NZ-18565' 'BK-11260' 'VD-21670' 'RS-19420' 'CS-12130' 'GZ-14470'
'SS-20590' 'RF-19735' 'TB-21625' 'VG-21790' 'MB-17305' 'LR-16915'
'MA-17560' 'DB-13405' 'JH-15985' 'ED-13885' 'GM-14500' 'AG-10525'
'MG-17650' 'BW-11200' 'DH-13675' 'CS-12175' 'CS-12355' 'KH-16510'
'TS-21610' 'DJ-13510' 'LH-16900' 'JB-15400' 'DO-13645' 'CR-12730'
'DE-13255' 'JP-15520' 'AP-10915' 'RB-19645' 'RH-19495' 'JP-15460'
'DW-13585' 'JM-15265' 'JE-15715' 'RC-19960' 'MW-18220' 'KB-16315'
'PP-18955' 'FM-14290' 'CC-12610' 'ZD-21925' 'RA-19945' 'HH-15010'
'AT-10735' 'SJ-20500' 'LT-16765' 'KD-16345' 'MR-17545' 'PR-18880'
'NP-18700' 'VP-21760' 'SZ-20035' 'LR-17035' 'DK-13375' 'MC-17275'
'BM-11140' 'KT-16465' 'SP-20620' 'LH-17020' 'KC-16255' 'RR-19525'
'LC-17140' 'MS-17365' 'CC-12220' 'DL-12925' 'SF-20065' 'AR-10345'
'LW-17125' 'LW-17215' 'TA-21385' 'BN-11470' 'PT-19090' 'EA-14035'
'JM-15535' 'ON-18715' 'JC-15775' 'DM-13015' 'AJ-10960' 'MF-18250'
'JA-15970' 'AG-10765' 'JK-15640' 'FA-14230' 'MC-18130' 'DN-13690'
'SB-20185' 'PF-19165' 'MA-17995' 'ES-14020' 'HG-14965' 'SL-20155'

'LH-16750' 'JG-15115' 'AS-10090' 'VG-21805' 'TT-21220' 'JP-16135'
'TH-21115' 'JD-16060' 'MT-18070' 'BF-11005' 'KF-16285' 'NB-18580'
'CS-12400' 'TH-21100' 'KD-16495' 'GZ-14545' 'LD-17005' 'SM-20950'
'EB-14170' 'KW-16570' 'DM-12955' 'DB-13615' 'ES-14080' 'AR-10405'
'BP-11290' 'RD-19930' 'DO-13435' 'DF-13135' 'JD-16015' 'BD-11635'
'GG-14650' 'AM-10360' 'TN-21040' 'FC-14335' 'HA-14920' 'JF-15565'
'BP-11050' 'ZC-21910' 'SP-20545' 'TC-21295' 'HG-15025' 'JE-15610'
'AC-10660' 'LS-17245' 'TB-21520' 'AS-10225' 'LS-17200' 'IG-15085'
'RP-19270' 'JF-15490' 'EH-14125' 'MP-17965' 'FH-14365' 'LS-16975'
'SF-20965' 'MG-17680' 'DB-12910' 'AH-10585' 'RD-19810' 'TS-21370'
'BT-11395' 'NC-18625' 'AM-10705' 'AC-10450' 'NL-18310' 'DG-13300'
'KH-16630' 'PL-18925' 'SH-20395' 'SW-20755' 'JK-16120' 'PO-18850'
'TC-21535' 'DM-13345' 'MS-17530' 'WB-21850' 'TH-21235' 'BG-11035'
'MD-17350' 'TC-20980' 'BF-11215' 'TZ-21580' 'KD-16270' 'PK-19075'
'DB-12970' 'RF-19840' 'RM-19675' 'JL-15505' 'ML-17410' 'CM-11815'
'JF-15190' 'PB-19105' 'JD-16150' 'AB-10165' 'AH-10075' 'RH-19510'
'RB-19330' 'YC-21895' 'JK-15730' 'MK-17905' 'PW-19240' 'AB-10600'
'JC-16105' 'GR-14560' 'DB-13360' 'AD-10180' 'JM-15865' 'DH-13075'
'JK-15325' 'SC-20680' 'TC-21475' 'SP-20920' 'SS-20140' 'SM-20005'
'PB-19210' 'NG-18355' 'NG-18430' 'AT-10435' 'HR-14830' 'DK-12985'
'JP-11135' 'AR-10540' 'BF-11080' 'MO-17500' 'MC-17635' 'BT-11680'
'BS-11380' 'AH-10210' 'TR-21325' 'DP-13165' 'MD-17860' 'MC-17575'
'EM-13825' 'KN-16390' 'MK-19560' 'CP-12085' 'JB-15925' 'RL-19615'
'NB-18655' 'RC-19825' 'DK-13150' 'FP-14320' 'DJ-13420' 'TB-21250'
'TB-21175' 'SH-19975' 'LH-17155' 'CY-12745' 'KC-16675' 'RA-19285'
'SS-20875' 'CS-11845' 'SP-20860' 'SH-20635' 'CC-12100' 'SC-20305'
'SC-20695' 'EC-14050' 'IM-15070' 'MK-18160' 'TS-21505' 'PS-19045'
'AB-10105' 'KM-16660' 'TW-21025' 'CC-12670' 'SF-20200' 'CK-12595'
'MG-17695' 'TB-21190' 'MG-17890' 'RE-19405' 'DC-13285' 'BG-18435'
'KB-16600' 'MF-17665' 'HO-15230' 'PC-18745' 'JL-15175' 'CB-12535'
'AH-10465' 'AG-10495' 'CS-12490' 'YS-21880' 'KS-16300' 'CR-12820'
'RW-19540' 'TS-21655' 'LO-17170' 'SJ-20215' 'GT-14755' 'LC-16960'
'NP-18670' 'TP-21565' 'DK-13090' 'AH-10195' 'JJ-15445' 'DK-12895'
'TG-21310' 'EB-13750' 'JE-15475' 'CM-12385' 'CM-12655' 'JO-15550'
'BW-11110' 'JW-16075' 'FW-14395' 'DB-13210' 'BP-11185' 'BV-11245'
'CM-12115' 'MO-17950' 'JD-15790' 'RD-19660' 'EB-14110' 'HJ-14875'
'AW-10840' 'TP-21130' 'OT-18730' 'JK-16090' 'LS-16945' 'DB-13120'
'JB-16045' 'LC-16870' 'SV-20815' 'AG-10675' 'RO-19780' 'JH-16180'
'NK-18490' 'EN-13780' 'BD-11560' 'BD-11320' 'ML-18265' 'CG-12040'
'BP-11155' 'BB-11545' 'EM-13810' 'MJ-17740' 'GB-14575' 'JH-15910'
'FC-14245' 'SN-20560' 'SC-20770' 'SW-20455' 'CH-12070' 'RH-19600'
'BD-11725' 'VP-21730' 'KM-16225' 'FM-14380' 'VB-21745' 'TS-21085'
'RK-19300' 'BE-11455' 'PN-18775' 'EM-14095' 'PO-19180' 'TT-21265'
'JW-15955' 'SC-20440' 'RB-19705' 'DP-13105' 'FH-14350' 'KW-16435'
'CA-12055' 'NM-18520' 'BO-11425' 'KD-16615' 'BM-11575' 'AC-10615'
'MO-17800' 'AB-10060' 'CM-12190' 'JR-16210' 'SJ-20125' 'CM-11935'
'DR-12880' 'BS-11800' 'HW-14935' 'EM-13960' 'HM-14980' 'JJ-15760'
'GH-14485' 'SO-20335' 'CG-12520' 'JF-15355' 'MY-17380' 'SG-20080'

```
 'EG-13900' 'KC-16540' 'CB-12415' 'MT-17815' 'MZ-17335' 'GH-14665'
 'KB-16405' 'PB-18805' 'TP-21415' 'RH-19555' 'MH-17620' 'SU-20665'
 'AF-10870' 'TT-21460' 'SC-20800' 'DJ-13630' 'PM-19135' 'EM-14200'
 'ER-13855' 'JR-15670' 'DL-13495' 'BT-11485' 'BO-11350' 'EB-13975'
 'JB-16000' 'BE-11335' 'TB-21355' 'HE-14800' 'BE-11410' 'DW-13540'
 'JG-15310' 'CJ-12010' 'BD-11770' 'HZ-14950' 'SV-20935' 'JT-12540'
 'JM-15250' 'RB-19360' 'SW-20350' 'JM-15580' 'CR-12580' 'EL-13735'
 'CL-12700' 'RB-19570' 'CM-12235' 'SM-20905' 'LW-16990' 'CV-12805'
 'AS-10285' 'CC-12370' 'RW-19690' 'LP-17095' 'SC-20230' 'RS-19870'
 'DV-13045' 'HO-15234' 'HO-15233' 'HO-15232' 'HO-15231' 'JS-15685'
 'TB-21055' 'LD-16855' 'PF-19225' 'AR-10570' 'LB-16735' 'AG-10330'
 'ML-17755' 'AO-10810' 'MH-18115' 'JS-13215' 'CM-12160' 'SC-20845'
 'AS-10135' 'RR-19315' 'NS-18505' 'CD-11980' 'SK-19990' 'JE-16165'
 'CJ-11875' 'RM-19750' 'ML-18040' 'TM-21490' 'MG-18205' 'CC-12550'
 'CS-12460' 'JC-15385' 'TC-21145' 'PH-18790' 'JR-15700' 'CD-12280']

----------------------------------------

Unique values in 'Customer Name':
['Darren Powers' 'Phillina Ober' 'Mick Brown' 'Maria Etezadi'
 "Jack O'Briant" 'Lycoris Saunders' 'Vivek Sundaresam' 'Melanie Seite'
 'Anthony Jacobs' 'Seth Vernon' 'Brian Dahlen' 'Natalie DeCherney'
 'Michael Moore' 'Chris Selesnick' 'Brendan Sweed' 'Erica Hackney'
 'Delfina Latchford' 'David Wiener' 'Toby Swindell' 'Hunter Lopez'
 'Xylona Preis' 'Mark Van Huff' 'Muhammed MacIntyre' 'Tom Boeckenhauer'
 'Jane Waco' 'Speros Goranitis' 'Ionia McGrath' 'Carol Adams'
 'Shirley Daniels' 'Ed Jacobs' 'Jasper Cacioppo' 'Mike Vittorini'
 'Liz Carlisle' 'Bradley Drucker' 'Eric Barreto' 'Cynthia Delaney'
 'Mark Haberlin' 'Deborah Brumfield' 'Neil Ducich' 'Cyma Kinney'
 'Nathan Mautz' 'Guy Armstrong' 'Natalie Fritzler' 'Sanjit Chand'
 'Marina Lichtenstein' 'Shui Tom' 'Barry Französisch' 'Michael Nguyen'
 'Troy Blackwell' 'Aaron Bergman' 'Linda Cazamias' 'Steven Roelle'
 'Tamara Manning' 'Sue Ann Reed' 'Michael Grace' 'Jennifer Halladay'
 'Joy Smith' 'Vicky Freymann' 'Shahid Collister' 'Dave Brooks'
 'Giulietta Weimer' 'Hallie Redmond' 'Kristina Nunn' 'Nicole Hansen'
 'Jill Stevenson' 'Eva Jacobs' 'Sam Craven' 'Ashley Jarboe' 'Anne Pryor'
 'Clay Ludtke' 'Aimee Bixby' 'Craig Molinari' 'Nora Pelletier'
 'Kean Thornton' 'Scott Williamson' 'Chloris Kastensmidt' 'Filia McAdams'
 'Richard Bierner' 'Kelly Andreada' 'Nicole Fjeld' 'Christine Kargatis'
 'Dean Katz' 'Jason Fortune-' 'Neil Französisch' 'Rick Bensley'
 'Alan Shonely' 'Christine Phan' 'Bradley Nguyen' 'Annie Zypern'
 'Quincy Jones' 'Eileen Kiefer' 'Sean Miller' 'Sample Company A'
 'Matt Connell' 'Gary McGarr' 'Kristen Hastings' 'Cassandra Brandow'
 'Thomas Seio' 'Beth Fritzler' 'Vivian Mathis' 'Craig Carroll'
 'Bruce Geld' "Patrick O'Donnell" 'Maurice Satty' 'Carlos Daly'
 'Christina Anderson' 'Damala Kotsonis' 'Cindy Stewart' 'Jonathan Doherty'
 'Mike Pelletier' 'Christina VanderZanden' 'Daniel Raglin'
 'Mike Gockenbach' 'Mark Packer' 'Susan Gilcrest' 'Kunst Miller'
```

'Dorris liebe' 'Nat Carroll' 'Trudy Glocke' 'Alex Avila' 'Karen Bern'
'Dorothy Dickinson' "Jas O'Carroll" 'Katharine Harms' 'Anna Häberlin'
'Rob Dowd' 'Katrina Edelman' 'Mary Zewe' 'Paul Knutson' 'Ken Black'
'Greg Maxwell' 'Ruben Dartt' 'Roy Phan' 'Tracy Hopkins' 'John Grady'
'Andy Yotov' 'Scott Cohen' 'Frank Hawley' 'Guy Phonely' 'Pauline Johnson'
'Roland Schwarz' 'Valerie Mitchum' 'Ruben Ausman' 'John Stevenson'
'Jim Mitchum' 'Pauline Chand' 'Sanjit Engle' 'Jay Kimmel'
'Michelle Moray' 'Gary Mitchum' 'Kean Nguyen' 'Edward Hooks'
'Stewart Visinsky' 'Ben Ferrer' 'Paul Stevenson' 'Stephanie Phelps'
'John Lee' 'Allen Goldenen' 'Benjamin Patterson' 'Allen Armold'
'Dorothy Badders' 'Aaron Hawkins' 'Laurel Beltran' 'Theresa Swint'
'Beth Thompson' 'Dianna Arnett' 'Lena Creighton' 'Aleksandra Gannaway'
'Marc Harrigan' 'Muhammed Yedwab' 'George Bell' 'David Smith'
'Arianne Irving' 'Laura Armstrong' 'Harry Greene' 'Lauren Leatherbury'
'Katherine Hughes' 'Justin MacKendrick' 'Andrew Roberts'
'Cari Schnelling' 'Nathan Cano' 'Joni Sundaresam' 'Andrew Allen'
'Bill Shonely' 'Paul MacIntyre' 'Emily Burns' 'Peter Fuller'
'Giulietta Dortch' 'Steve Nguyen' 'Harold Pawlan' 'Carol Triggs'
'Sean Braxton' 'Patrick Gardner' 'Art Foster' 'Greg Matthias'
'Victoria Wilson' 'Paul Gonzalez' 'Brooke Gillingham' 'Harold Dahlen'
'Raymond Messe' 'Erica Hernandez' 'Alejandro Grove' 'Naresj Patel'
'Anthony Johnson' 'Pauline Webber' 'Cindy Chapman' 'Pete Armstrong'
'Ted Trevino' 'George Ashbrook' 'Bruce Stewart' 'John Huston'
'Tamara Dahlen' 'Randy Ferguson' 'Matt Collister' 'Chuck Clark'
'Frank Gastineau' 'Bradley Talbott' 'Mitch Willingham' 'Evan Henry'
'Greg Tran' 'Resi Pölking' 'Janet Lee' 'Barry Weirich' 'Herbert Flentye'
'Dean percer' 'Maya Herman' 'Laurel Elliston' 'Magdelene Morse'
'Patrick Jones' 'Shahid Shariari' 'Laurel Workman' 'James Galang'
'Ellis Ballard' 'Mark Cousins' 'Jim Karlsson' 'Dianna Wilson'
'Liz Thompson' 'Corey Roper' 'Richard Eichhorn' 'Steven Cartwright'
'Ross Baird' 'Bobby Trafton' 'Grant Thornton' 'Bryan Mills' 'Sarah Bern'
'Valerie Takahito' 'Nick Radford' 'Bart Pistole' 'Carlos Soltero'
'Rick Duston' 'Brosina Hoffman' 'Frank Olsen' 'Carl Weiss' 'Chuck Magee'
'Heather Kirkland' 'Sharelle Roach' 'Kelly Lampkin' 'Pamela Stobb'
'Sonia Cooley' 'Gary Hwang' 'Jennifer Ferguson' 'Duane Benoit'
'Mick Crebagga' 'Neil Cohen' 'Corey-Lock' 'Natalie Webber'
'Toby Braunhardt' 'Anna Andreadi' 'Dianna Vittorini' 'Joel Eaton'
'Denise Leinenbach' 'Brian Stugart' 'Liz Pelletier' 'Noel Staavos'
'Grace Kelly' 'Liz MacKendrick' 'Cynthia Arntzen' 'Rob Williams'
'Dan Lawera' 'Anthony Rawles' 'Russell Applegate' 'Helen Abelman'
'Michael Stewart' 'Brian DeCherney' 'Eric Hoffmann' 'Maxwell Schwartz'
'Don Miller' 'Dennis Pardue' 'Michael Chen' 'Dan Campbell'
'Becky Castell' 'Emily Phan' 'Michelle Huthwaite' 'Ed Braxton'
'Darrin Sayre' 'Gary Hansen' 'Brian Moss' 'Mark Hamilton' 'Jamie Kunitz'
'Paul Van Hugh' 'Alyssa Crouse' 'Max Engle' 'Shirley Schmidt'
'Barbara Fisher' 'Arthur Wiediger' 'Philip Brown' 'Adrian Hane'
'Jack Lebron' 'Christine Abelman' 'Eugene Moren' 'Katherine Murray'
'Charles Crestani' 'Scot Wooten' 'Roger Demir' 'Michelle Ellison'

'Harry Marie' 'Erin Mull' 'Alejandro Ballentine' 'John Lucas'
'Arthur Gainer' 'Nick Crebassa' 'Ken Lonsdale' 'Sheri Gordon'
'Luke Foster' 'Carl Ludwig' 'Aaron Smayling' 'Barry Blumstein'
'Tom Stivers' 'Ivan Liston' 'Ann Steele' 'Nick Zandusky' 'Berenike Kampe'
'Valerie Dominguez' 'Ricardo Sperren' 'Chad Sievert' 'Gary Zandusky'
'Sonia Sunley' 'Roland Fjeld' 'Trudy Brown' 'Vivek Gonzalez'
'Maria Bertelson' 'Lena Radford' 'Matt Abelman' 'Denny Blanton'
'Joseph Holt' 'Emily Ducich' 'Gene McClure' 'Andy Gerbode'
'Matthew Grinstein' 'Ben Wallace' 'Duane Huffman' 'Charles Sheldon'
'Christine Sundaresam' 'Keith Herrera' 'Troy Staebel' 'Don Jones'
'Lena Hernandez' 'Jennifer Braxton' "Doug O'Connell" 'Craig Reiter'
'Deanra Eno' 'Jeremy Pistek' 'Arthur Prichep' 'Robert Barroso'
'Rick Hansen' 'Jennifer Patt' 'Dorothy Wardle' 'Janet Molinari'
'Joe Elijah' 'Ryan Crowe' 'Mitch Webber' 'Karl Braun' 'Paul Prost'
'Frank Merwin' 'Corey Catlett' 'Zuschuss Donatelli' 'Ryan Akin'
'Hilary Holden' 'Annie Thurman' 'Shirley Jackson' 'Larry Tron'
'Katherine Ducich' 'Mathew Reese' 'Patrick Ryan' 'Nora Preis'
'Victoria Pisteka' 'Sam Zeldin' 'Lisa Ryan' 'Dennis Kane' 'Marc Crier'
'Becky Martin' 'Kean Takahito' 'Stefania Perrino' 'Lisa Hazard'
'Karen Carlisle' 'Rick Reed' 'Logan Currie' 'Maribeth Schnelling'
'Chris Cortes' 'Daniel Lacy' 'Sandra Flanagan' 'Alex Russell'
'Liz Willingham' 'Luke Weiss' 'Tom Ashbrook' 'Brad Norvell'
'Pete Takahito' 'Erin Ashbrook' 'Jessica Myrick' 'Odella Nelson'
'John Castell' 'Darrin Martin' 'Astrea Jones' 'Monica Federle'
'Joseph Airdo' 'Anthony Garverick' 'Jim Kriz' 'Frank Atkinson'
'Mike Caudle' 'Duane Noonan' 'Sarah Brown' 'Philip Fox' 'Michelle Arnett'
'Erica Smith' 'Henry Goldwyn' 'Sara Luxemburg' 'Larry Hughes'
'Jack Garza' 'Adam Shillingsburg' 'Vivek Grady' 'Thomas Thornton'
'Julie Prescott' 'Thea Hudgings' 'Julia Dunbar' 'Michelle Tran'
'Barry Franz' 'Karen Ferguson' 'Nicole Brennan' 'Christopher Schild'
'Thea Hendricks' 'Keith Dawkins' 'George Zrebassa' 'Lisa DeCherney'
'Suzanne McNair' 'Evan Bailliet' 'Kelly Williams' 'Dario Medina'
'Doug Bickford' 'Erin Smith' 'Allen Rosenblatt' 'Beth Paige'
"Russell D'Ascenzo" 'Denny Ordway' 'David Flashing' 'Joy Daniels'
'Brian Derr' 'Greg Guthrie' 'Alice McCarthy' 'Tanja Norvell' 'Fred Chung'
'Helen Andreada' 'Jill Fjeld' 'Barry Pond' 'Zuschuss Carroll'
'Sibella Parks' 'Toby Carlisle' 'Hunter Glantz' 'Jim Epp' 'Anna Chung'
'Lynn Smith' 'Tracy Blumstein' 'Alan Schoenberger' 'Luke Schmidt'
'Ivan Gibson' 'Rachel Payne' 'Jeremy Farry' 'Eugene Hildebrand'
'Michael Paige' 'Fred Hopkins' 'Lindsay Shagiari' 'Sylvia Foulston'
'Maureen Gastineau' 'Daniel Byrd' 'Angele Hood' 'Ross DeVincentis'
'Todd Sumrall' 'Bill Tyler' 'Noah Childs' 'Anne McFarland' 'Amy Cox'
'Nancy Lomonaco' 'Deirdre Greer' 'Ken Heidel' 'Paul Lucas'
'Shahid Hopkins' 'Steven Ward' 'Julie Kriz' "Patrick O'Brill"
'Tracy Collins' 'Denise Monton' 'MaryBeth Skach' 'William Brown'
'Tiffany House' 'Barry Gonzalez' 'Maribeth Dona' 'Tamara Chand'
'Benjamin Farhat' 'Tracy Zic' 'Karen Daniels' 'Pete Kriz' 'Darren Budd'
'Roy Französisch' 'Robert Marley' 'Jeremy Lonsdale' 'Maris LaWare'

'Candace McMahon' 'Jamie Frazer' 'Peter Bühler' 'Justin Deggeller'
'Alan Barnes' 'Adam Hart' 'Rick Huthwaite' 'Randy Bradley'
'Yoseph Carroll' 'Joe Kamberova' 'Michael Kennedy' 'Pierre Wener'
'Ann Blume' 'Julie Creighton' 'Georgia Rosenberg' 'Dennis Bolton'
'Alan Dominguez' 'John Murray' 'Dave Hallsten' 'Jason Klamczynski'
'Steve Carroll' 'Tony Chapman' 'Susan Pistek' 'Saphhira Shifley'
'Sally Matthias' 'Phillip Breyer' 'Nat Gilpin' 'Nathan Gelder'
'Alyssa Tate' 'Harold Ryan' 'Darren Koutras' 'James Peterman'
'Andy Reiter' 'Bart Folk' "Mary O'Rourke" 'Matthew Clasen'
'Brian Thompson' 'Bill Stewart' 'Alan Hwang' 'Toby Ritter'
'David Philippe' 'Michael Dominguez' 'Matt Collins' 'Elizabeth Moffitt'
'Katherine Nockton' 'Michael Knudson' 'Cathy Prescott' 'Joni Blumstein'
'Rob Lucas' 'Nona Balk' 'Roy Collins' 'David Kendrick' 'Frank Preis'
'Denny Joy' 'Tim Brockman' 'Thomas Boland' 'Sally Hughsby'
'Logan Haushalter' 'Craig Yedwab' 'Kimberly Carter' 'Ralph Arnett'
'Sung Shariari' 'Cari Sayre' 'Sung Pak' 'Stefanie Holloman'
'Chad Cunningham' 'Sean Christensen' 'Steve Chapman' 'Erin Creighton'
'Irene Maddox' 'Mike Kennedy' 'Tony Sayre' 'Penelope Sewall'
'Adrian Barton' 'Khloe Miller' 'Tamara Willingham' 'Craig Carreira'
'Sarah Foster' 'Clytie Kelty' 'Maureen Gnade' 'Thomas Brumley'
'Michael Granlund' 'Ricardo Emerson' 'Debra Catini' 'Bruce Galang'
'Ken Brennan' 'Maureen Fritzler' 'Harry Olson' 'Pamela Coakley'
'James Lanier' 'Claudia Bergmann' 'Amy Hunt' 'Andrew Gjertsen'
'Cindy Schnelling' 'Yana Sorensen' 'Karen Seio' 'Cyra Reiten'
'Rick Wilson' 'Trudy Schmidt' 'Lori Olson' 'Sarah Jordon' 'Guy Thornton'
'Lindsay Castell' 'Nora Paige' 'Tracy Poddar' 'Dave Kipp' 'Alan Haines'
'Jennifer Jackson' 'Dana Kaydos' 'Toby Gnade' 'Edward Becker'
'Jeremy Ellison' 'Christopher Martinez' 'Corinna Mitchell' 'Jesus Ocampo'
'Bart Watters' 'Julia West' 'Fred Wasserman' 'Dean Braden' 'Ben Peterman'
'Benjamin Venier' 'Chad McGuire' 'Michael Oakman' 'John Dryer'
'Robert Dilbeck' 'Eugene Barchas' 'Heather Jas' 'Anthony Witt'
'Theone Pippenger' 'Olvera Toch' 'Juliana Krohn' 'Linda Southworth'
'David Bremer' 'Julia Barnett' 'Lena Cacioppo' 'Stuart Van' 'Anna Gayman'
"Rose O'Brian" 'Justin Hirsh' 'Neil Knudson' 'Edward Nazzal'
'Brendan Dodson' 'Bill Donatelli' 'Muhammed Lee' 'Catherine Glotzbach'
'Becky Pak' 'Brenda Bowman' 'Eleni McCrary' 'Max Jones'
'Giulietta Baptist' 'Jonathan Howell' 'Frank Carlisle' 'Skye Norling'
'Stewart Carmichael' 'Shaun Weien' 'Cathy Hwang' 'Rob Haberlin'
'Bruce Degenhardt' 'Victor Preis' 'Kalyca Meade' 'Fred McMath'
'Victoria Brennan' 'Thais Sissman' 'Ralph Kennedy' 'Brad Eason'
'Parhena Norris' 'Eudokia Martin' 'Philisse Overcash' 'Tim Taslimi'
'Joni Wasserman' 'Shaun Chance' 'Roger Barcio' 'Dave Poirier'
'Fred Harton' 'Katrina Willman' 'Cathy Armstrong' 'Neoma Murray'
'Bobby Odegard' 'Ken Dana' 'Brendan Murry' 'Ann Chong' "Meg O'Connel"
'Adam Bellavance' 'Charlotte Melton' 'Justin Ritter' 'Sanjit Jacobs'
'Carlos Meador' 'Dan Reichenbach' 'Bryan Spruell' 'Helen Wasserman'
'Eric Murdock' 'Henry MacAllister' 'Joel Jenkins' 'Gene Hale'
"Sean O'Donnell" 'Claire Gute' 'Jay Fein' 'Maribeth Yedwab'

```
 'Sandra Glassco' 'Emily Grady' 'Kelly Collister' 'Christy Brittain'
 'Meg Tillman' 'Maria Zettner' 'Greg Hansen' 'Katrina Bavinger'
 'Patrick Bzostek' 'Tom Prescott' 'Ritsa Hightower' 'Matt Hagelstein'
 'Stephanie Ulpright' 'Art Ferguson' 'Tonja Turnell' 'Stuart Calhoun'
 'Doug Jacobs' 'Peter McVee' 'Evan Minnotte' 'Elpida Rittenbach'
 'Jim Radford' 'Dionis Lloyd' 'Brad Thomas' 'Bill Overfelt' 'Erica Bern'
 'Joy Bell-' 'Bill Eplett' 'Todd Boyes' 'Harold Engle' 'Bobby Elias'
 'Don Weiss' 'Jason Gross' 'Caroline Jumper' 'Bryan Davis' 'Henia Zydlo'
 'Susan Vittorini' 'Jill Trafton' 'Janet Martin' 'Raymond Buch'
 'Sean Wendt' 'Jill Matthias' 'Clay Rozendal' 'Ed Ludwig' 'Craig Leslie'
 'Rob Beeghly' 'Chris McAfee' 'Susan MacKendrick' 'Lindsay Williams'
 'Cynthia Voltz' 'Alejandro Savely' 'Christopher Conant' 'Robert Waldorf'
 'Liz Preis' 'Scot Coram' 'Roy Skaria' 'Darrin Van Huff' 'Jim Sink'
 'Ted Butterfield' 'Lela Donovan' 'Phillip Flathmann' 'Anemone Ratner'
 'Larry Blacks' 'Alex Grayson' 'Max Ludwig' "Anthony O'Donnell"
 'Mick Hernandez' 'Jennifer Sheldon' 'Charles McCrossin' 'Sung Chung'
 'Adrian Shami' 'Ralph Ritter' 'Neola Schneider' 'Carol Darley'
 'Sally Knutson' 'Justin Ellison' 'Carl Jackson' 'Roland Murray'
 'Michelle Lonsdale' 'Tony Molinari' 'Mitch Gastineau' 'Clay Cheatham'
 'Chuck Sachs' 'Jenna Caffey' 'Theresa Coyne' 'Patricia Hirasaki'
 'Jocasta Rupert' 'Christina DeMoss']

----------------------------------------

Unique values in 'Segment':
['Consumer' 'Home Office' 'Corporate']

----------------------------------------

Unique values in 'Country/Region':
['United States' 'Canada']

----------------------------------------

Unique values in 'City':
['Houston' 'Naperville' 'Philadelphia' 'Henderson' 'Athens' 'Los Angeles'
 'Huntsville' 'Laredo' 'Springfield' 'Dover' 'San Francisco'
 'Mount Pleasant' 'Newark' 'Bossier City' 'Roswell' 'Scottsdale'
 'Jonesboro' 'Westland' 'Smyrna' 'Miami' 'Toronto' 'Lafayette' 'Las Vegas'
 'Rapid City' 'Alexandria' 'San Diego' 'New York City' 'Detroit'
 'Mission Viejo' 'Green Bay' 'Saint Petersburg' 'Seattle' 'Escondido'
 'Romeoville' 'Chesapeake' 'Linden' 'North Las Vegas' 'Columbia' 'Concord'
 'Dallas' 'Chicago' 'Lubbock' 'Arlington' 'Richmond' 'Woodstock'
 'Moreno Valley' 'El Paso' 'Medford' 'Columbus' 'Elmhurst' 'Wilmington'
 'Margate' 'Yonkers' 'Des Moines' 'Denver' 'Royal Oak' 'Roseville'
 'Calgary' 'Huntington Beach' 'Logan' 'Jacksonville' 'Tampa' 'Raleigh'
 'Lakeville' 'Jackson' 'Burbank' 'Lakeland' 'Knoxville' 'Hamilton'
 'Asheville' 'Tucson' 'Portage' 'Greensboro' 'Delray Beach' 'Fresno'
```

'Pomona' 'Albuquerque' 'Plano' 'Brownsville' 'Long Beach' 'Apple Valley'
'Vallejo' 'Revere' 'Virginia Beach' 'Dearborn Heights' 'Decatur'
'Lancaster' 'Mobile' 'Marietta' 'Toledo' 'Glendale' 'Chandler' 'Lewiston'
'Great Falls' 'Austin' 'Redondo Beach' 'Lodi' 'Bloomington' 'Baltimore'
'San Jose' 'Troy' 'San Gabriel' 'Jamestown' 'Memphis' 'Lake Charles'
'Rochester' 'Louisville' 'Appleton' 'Middletown' 'San Antonio' 'Freeport'
'Lawrence' 'Kent' 'Fort Worth' 'Watertown' 'Milwaukee' 'Franklin'
'Hialeah' 'West Jordan' 'Oakland' 'Eau Claire' 'Akron' 'Cleveland'
'Midland' 'San Marcos' 'Vancouver' 'Montreal' 'Bellevue' 'Murray'
'Buffalo Grove' 'Little Rock' 'Lakewood' 'Orem' 'Aurora' 'Peoria'
'Bristol' 'Harrisonburg' 'Mishawaka' 'Hempstead' 'Halifax' 'Lawton'
'Waynesboro' 'Meriden' 'Pueblo' 'Chester' 'Phoenix' 'Minneapolis' 'Salem'
'Southaven' 'Cincinnati' 'Deltona' 'Plainfield' 'Palm Coast' 'El Cajon'
'Buffalo' 'Hackensack' 'Niagara Falls' 'League City' 'Sioux Falls'
'New Rochelle' 'Riverside' 'Omaha' 'Atlanta' 'Draper' 'Apopka'
'Charlotte' 'Bangor' 'Pleasant Grove' 'Texas City' 'Trenton' 'Vacaville'
'Hollywood' 'Fairfield' 'Hampton' 'Saint Charles' 'North Miami'
'Grand Rapids' 'Oceanside' 'Billings' 'Owensboro' 'Santa Fe'
'Fayetteville' 'Bowling Green' 'Tulsa' 'Oswego' 'Santa Clara' 'Pasco'
'Tyler' 'Macon' 'Lowell' 'Greenville' 'Gresham' 'Clifton' 'Oxnard'
'Olathe' 'Cary' 'Odessa' 'Tempe' 'Corpus Christi' 'Chula Vista' 'Garland'
'Boca Raton' 'Mesquite' 'Clarksville' 'Boynton Beach' 'Reno' 'Evanston'
'Durham' 'Edmonton' 'Indianapolis' 'Manteca' 'Pasadena' 'Edmonds'
'Mount Vernon' 'Everett' 'Parma' 'Beaumont' 'Montgomery' 'Texarkana'
'Newport News' 'Rancho Cucamonga' 'Rock Hill' 'Fort Lauderdale'
'Garden City' 'Cranston' 'Lorain' 'Avondale' 'Mason' 'Orange' 'Portland'
'Medina' 'Irving' 'Nashville' 'Wausau' 'Redding' 'Reading' 'Madison'
'Carrollton' 'Johnson City' 'Manhattan' 'Moorhead' 'Cedar Hill'
'Des Plaines' 'Provo' 'Salt Lake City' 'Coon Rapids' 'Monroe'
'Bolingbrook' 'Sacramento' 'Saint Louis' "St. John's" 'Woonsocket'
'Brentwood' 'Utica' 'Tigard' 'Skokie' 'Orlando' 'Clinton' 'Sandy Springs'
'Moncton' 'Oklahoma City' 'Gilbert' 'Olympia' 'Mesa' 'Caldwell' 'Marion'
'Florence' 'Grand Prairie' 'Thornton' 'Port Arthur' 'Colorado Springs'
'Beverly' 'Anaheim' 'Cottage Grove' 'Quebec City' 'Taylor'
'Charlottetown' 'Providence' 'Baytown' 'Woodbury' 'Park Ridge' 'Bartlett'
'Bozeman' 'West Palm Beach' 'Rome' 'Suffolk' 'Kenosha' 'Perth Amboy'
'Hot Springs' 'Las Cruces' 'Sterling Heights' 'Leominster' 'Altoona'
'Coppell' 'Bethlehem' 'New Castle' 'Plantation' 'Chico' 'Lehi' 'Auburn'
'San Bernardino' 'Thousand Oaks' 'Covington' 'Coral Springs' 'Normal'
'Lansing' 'Spokane' 'Norwich' 'Norfolk' 'Farmington' 'Mcallen'
'New Albany' 'Santa Maria' 'Daytona Beach' 'Washington' 'Tinley Park'
'Allen' 'Cuyahoga Falls' 'Camarillo' 'Wilson' 'Frankfort' 'Wichita'
'Haltom City' 'Manchester' 'Paterson' 'Pocatello' 'Layton' 'East Point'
'Carol Stream' 'Holyoke' 'Vineland' 'Amarillo' 'Bakersfield'
'Port Saint Lucie' 'Highland Park' 'South Bend' 'Hattiesburg' 'Kirkwood'
'Boise' 'Redmond' 'Eagan' 'Yucaipa' 'New Bedford' 'Allentown' 'Murrieta'
'Bedford' 'Holland' 'Charlottesville' 'Tamarac' 'La Quinta' 'Redlands'
'North Charleston' 'Lincoln Park' 'Quincy' 'Dubuque' 'Broken Arrow'

```
 'Rockford' 'Murfreesboro' 'Bayonne' 'Cambridge' 'Hillsboro' 'Rockville'
 'Warner Robins' 'Ann Arbor' 'Santa Barbara' 'Noblesville' 'Orland Park'
 'Sparks' 'Salinas' 'Conway' 'Burlington' 'Helena' 'Lebanon' 'Rio Rancho'
 'Frisco' 'Morristown' 'Lake Elsinore' 'Pembroke Pines' 'Champaign'
 'Dearborn' 'Santa Ana' 'Tallahassee' 'Temecula' 'Costa Mesa' 'Glenview'
 'Bullhead City' 'Lindenhurst' 'Superior' 'Dublin' 'Visalia' 'Missoula'
 'Gaithersburg' 'Longview' 'Westfield' 'Gulfport' 'Atlantic City'
 'Sierra Vista' 'Chattanooga' 'Belleville' 'La Crosse' 'Round Rock'
 'Andover' 'Milford' 'Harlingen' 'Redwood City' 'Bryan' 'Malden'
 'Littleton' 'Saint Peters' 'Norman' 'Grapevine' 'Gastonia'
 'Jefferson City' 'San Clemente' 'Hesperia' 'Encinitas' 'Yuma' 'Waterbury'
 'Warwick' 'Passaic' 'Parker' 'Longmont' 'York' 'Broomfield' 'Winnipeg'
 'Pensacola' 'Hendersonville' 'West Allis' 'Kenner' 'Davis' 'Edinburg'
 'Fort Collins' 'Pharr' 'Sheboygan' 'Englewood' 'Waco' 'Waukesha'
 'Georgetown' 'Cedar Rapids' 'Saint Paul' 'Thomasville' 'Wheeling'
 'Stockton' 'Arvada' 'Twin Falls' 'Laguna Niguel' 'Marlborough' 'Woodland'
 'Regina' 'Iowa City' 'La Porte' 'Lake Forest' 'Coral Gables' 'The Colony'
 'Marysville' 'Bridgeton' 'San Luis Obispo' 'Conroe' 'Urbandale' 'Eugene'
 'Cheyenne' 'Arlington Heights' 'Carlsbad' 'Edmond' 'Montebello' 'Shelton'
 'East Orange' 'Overland Park' 'Hickory' 'San Angelo' 'Morgan Hill'
 'Antioch' 'Richardson' 'Fremont' 'Greenwood' 'Homestead' 'Torrance'
 'Goldsboro' 'Nashua' 'Deer Park' 'Tuscaloosa' 'Ormond Beach' 'Keller'
 'Hoover' 'Muskogee' 'Rogers' 'Pompano Beach' 'Oak Park' 'New Brunswick'
 'Greeley' 'Kissimmee' 'Sanford' 'Danville' 'Westminster' 'Mansfield'
 'Fargo' 'Laurel' 'Bellingham' 'Missouri City' 'Pearland'
 'Rochester Hills' 'Maple Grove' 'Chapel Hill' 'Commerce City'
 'Citrus Heights' 'Pico Rivera' 'San Mateo' 'Waterloo' 'Elkhart'
 'Loveland' 'Inglewood' 'La Mesa' 'Modesto' 'Independence' 'Clovis'
 'Grand Island' 'Melbourne' 'Pine Bluff' 'Saint Cloud' 'Miramar' 'Mentor'
 'Meridian' 'Springdale' 'Sunnyvale' 'Coachella' 'Aberdeen' 'Jupiter'
 'Grove City' 'Elyria' 'Hagerstown' 'Saginaw' 'College Station' 'Ontario'
 'Renton' 'Canton' 'Summerville' 'Gladstone' 'Whittier' 'Abilene'
 'Palatine' 'Port Orange' 'Danbury']

----------------------------------------

Unique values in 'State/Province':
['Texas' 'Illinois' 'Pennsylvania' 'Kentucky' 'Georgia' 'California'
 'Virginia' 'Delaware' 'South Carolina' 'Ohio' 'Louisiana' 'Oregon'
 'Arizona' 'Arkansas' 'Michigan' 'Tennessee' 'Florida' 'Ontario' 'Indiana'
 'Nevada' 'South Dakota' 'New York' 'Wisconsin' 'Washington' 'New Jersey'
 'Missouri' 'North Carolina' 'Colorado' 'Alberta' 'Utah' 'Minnesota'
 'Mississippi' 'Iowa' 'New Mexico' 'Massachusetts' 'Alabama' 'Idaho'
 'Montana' 'Maryland' 'Connecticut' 'New Hampshire' 'British Columbia'
 'Quebec' 'Nova Scotia' 'Oklahoma' 'Nebraska' 'Maine' 'Kansas'
 'Rhode Island' 'Newfoundland and Labrador' 'New Brunswick'
 'Prince Edward Island' 'District of Columbia' 'Vermont' 'Manitoba'
 'Saskatchewan' 'Wyoming' 'North Dakota' 'West Virginia']
```

```
----------------------------------------

Unique values in 'Postal Code':
['77095' '60540' '19143' '42420' '30605' '90049' '77340' '78041' '22153'
 '19901' '94109' '29464' '43055' '71111' '19140' '30076' '19134' '97477'
 '85254' '72401' '48185' '37167' '33180' 'M7A' '47905' '89115' '57701'
 '22304' '92037' '10024' '48234' '92691' '54302' '33710' '98105' '92025'
 '92024' '60441' '23320' '07036' '89031' '65203' '94521' '98103' '75220'
 '60653' '79424' '76017' '47374' '60098' '92553' '79907' '97504' '43229'
 '60126' '98115' '77036' '10035' '10009' '19120' '28403' '33063' '29203'
 '10701' '98198' '80219' '48073' '40475' '95661' 'T2C' '92646' '84321'
 '32216' '33614' '27604' '55044' '39212' '75217' '94110' '91505' '33801'
 '37918' '45011' '28806' '85705' '46368' '27405' '33445' '90036' '93727'
 '94122' '91767' '45503' '50315' '87105' '75023' '78521' '11561' '55124'
 '60610' '94591' '02151' '23464' '90008' '48205' '70506' '48127' '62521'
 '90004' '17602' '36608' '30062' '35601' '43615' '85301' '85224' '83501'
 '59405' '78745' '90278' '95240' '47401' '21215' '95123' '22204' '90045'
 '77070' '12180' '91776' '14701' '38109' '70601' '14609' '40214' '54915'
 '06457' '49201' '78207' '61032' '01841' 'M3C' '98031' '76106' '13601'
 '53209' '02038' '33012' '84084' '94601' '54703' '44312' '44105' '48640'
 '03820' '78666' 'V6Z' 'H1A' '98006' '60623' '84107' '92105' '60089'
 '23223' '72209' '08701' '84057' '60505' '61604' '37620' '90032' '22801'
 '46544' '11550' '19711' 'B3H' '73505' '22980' '06450' '81001' '19013'
 '85023' '55407' '80027' '28027' '97301' '38671' '45231' '10011' '32725'
 '07060' '80013' '32137' '92020' '14215' '07601' '14304' '77573' '57103'
 '10801' '06010' '92503' '68104' '30318' '84020' '32712' '28205' '65807'
 '04401' '84062' '77590' '48183' '95687' 'M5H' '30188' '33021' '06824'
 '75081' '31907' 'H1B' '23666' '44107' '63301' '33161' '49505' '11572'
 '59102' '42301' '33178' '87505' '72701' '43402' '74133' '60543' '95051'
 '99301' '75701' '31204' '01852' '43130' '48227' '27834' '97030' '07011'
 'H1C' '93030' '66062' '27511' '45014' '79762' '85281' '78415' '91911'
 '90805' '21044' '61701' 'M2N' '75043' '33433' '77041' '75150' '28314'
 '37042' '33437' '89502' '60201' '27707' 'T1Y' 'T5A' '46203' '28540'
 '95336' '91104' '98026' '10550' '02149' '03301' '44134' '77705' '36116'
 '71854' '90712' 'V6G' '23602' '91730' '29730' '33311' '67846' '47201'
 '02920' '44052' '85323' '45040' '07050' '97206' '44256' '75061' '37211'
 '54401' '96003' '19601' '53711' '75007' '77506' '37604' '66502' '56560'
 '75104' '60016' '53132' '84604' '19805' '84106' '55433' '71203' '60440'
 '95823' '63116' 'A0A' '02895' '94513' '13501' '97224' '11520' '60076'
 '32839' '20735' '30328' 'E1A' '73120' 'V6B' '85234' '98502' '85204'
 '83605' '43302' '35630' '38401' '75051' '80229' 'V5K' '77642' '80906'
 '01915' '92804' '60174' '44240' '55016' 'G1B' '48180' 'C0A' '02908'
 '77520' '55125' '60068' '38134' '59715' '33407' '13440' '23434' '53142'
 '08861' '71901' '88001' '48310' '01453' '16602' '75019' '18018' '47362'
 '33317' '95928' '84043' '36830' '92404' '93534' '35810' '91360' '29501'
 '98042' '33065' '33142' '61761' '48911' '99207' '06360' '68701' '87401'
 '78501' '47150' '93454' '32114' '20016' '45373' '60477' '75002' '44221'
```

```
 '93010' '27893' '60423' '67212' '76117' '06040' '07501' '83201' '04240'
 '85345' '84041' '30344' '30080' '60188' '01040' '08360' '79109' '93309'
 '34952' '60035' '46614' '39401' '63122' '83704' '97756' '55122' '92399'
 '02740' '41042' '18103' '92563' '76021' '49423' '22901' '55901' '33319'
 '92253' '92374' '13021' '29406' '48146' '02169' '92054' '37064' '52001'
 '74012' '61107' '37130' '07002' '02138' '62301' '97123' '20852' '31088'
 '48104' '93101' '46060' '48858' '60462' '89431' '93905' '72032' '28110'
 '05401' '46226' '55113' '59601' '37087' '87124' '75034' '07960' '38301'
 '92530' '33024' '61821' '48126' '92704' '32303' '92592' '92627' '60025'
 '86442' '11757' '54880' '43017' '93277' '59801' '20877' '98632' '07090'
 '39503' '08401' '85635' '37421' '07109' '54601' '78664' '01810' '06460'
 '78550' '94061' '77803' '02148' '80122' '63376' '73071' '76051' '28052'
 'V6E' '65109' '92672' '92345' '89015' '24153' '85364' '06708' '02886'
 '07055' '80134' '80501' '17403' '48066' '80020' 'R0H' '42104' '32503'
 '37075' '53214' '70065' '95616' '78539' '80525' '78577' '53081' '80112'
 '76706' '53186' '40324' '52402' '55106' '27360' '60090' '95207' '80004'
 '83301' '92677' '01752' '98052' '95695' 'S0G' '52240' '46350' '92630'
 '33134' '75056' '98270' '77571' 'R3R' '08302' '93405' '77301' '50322'
 '97405' '82001' '60004' '98002' '98661' '27217' '88220' '73034' '90640'
 '06484' '07017' '66212' '28601' '76903' '52302' '95037' '42071' '94509'
 '75080' '68025' '46142' '33030' '90503' '92307' '27534' '03060' '77536'
 '35401' '32174' '76248' '35244' '74403' '72756' '33068' '94533' '48237'
 '08901' '80634' '34741' '32771' '61832' '92683' '76063' '58103' '20707'
 '98226' '77489' '77581' '48307' '55369' '27514' '80022' '26003' '95610'
 '90660' '94403' '50701' '46514' '80538' '90301' '91941' '95351' '64055'
 '88101' '68801' '32935' '71603' '56301' '33023' '44060' '83642' '94526'
 '72762' '94086' '92236' '57401' '33458' '94568' '43123' '44035' '21740'
 '48601' '77840' '91761' '98059' '48187' '29483' '64118' '90604' '79605'
 '52601' '60067' '60302' '32127' '06810' '98208']

----------------------------------------

Unique values in 'Region':
['Central' 'East' 'South' 'West']

----------------------------------------

Unique values in 'Product ID':
['OFF-PA-10000174' 'OFF-BI-10004094' 'OFF-LA-10003223' …
 'TEC-PH-10001468' 'TEC-MA-10001856' 'FUR-BO-10002206']

----------------------------------------

Unique values in 'Category':
['Office Supplies' 'Furniture' 'Technology']

----------------------------------------
```

```
Unique values in 'Sub-Category':
['Paper' 'Binders' 'Labels' 'Storage' 'Art' 'Chairs' 'Fasteners' 'Phones'
 'Furnishings' 'Accessories' 'Bookcases' 'Envelopes' 'Appliances' 'Tables'
 'Supplies' 'Machines' 'Copiers']

----------------------------------------

Unique values in 'Product Name':
['Message Book, Wirebound, Four 5 1/2" X 4" Forms/Pg., 200 Dupl. Sets/Book'
 'GBC Standard Plastic Binding Systems Combs' 'Avery 508' … 'Xerox 1901'
 'Panasonic Business\xa0Telephones\xa0KX-T7736'
 'Bush Saratoga Collection 5-Shelf Bookcase, Hanover Cherry, *Special Order']

----------------------------------------
```

check out unique value in segment columns

[26]:
```
df['Segment'].unique()
```

[26]:
```
array(['Consumer', 'Home Office', 'Corporate'], dtype=object)
```

group based on sum values and sales & segment.

[27]:
```
df_segment_sales = df.groupby('Segment')['Sales'].sum()
```

[28]:
```
df_segment_sales
```

[28]:
```
Segment
Consumer       1.170660e+06
Corporate      7.158061e+05
Home Office    4.400684e+05
Name: Sales, dtype: float64
```

[29]:
```
df_segment_sales = df.groupby('Segment')['Sales'].sum().reset_index()
fig = px.pie(df_segment_sales, values='Sales', names ='Segment', title='sales␣
 ↪by segment')
fig.show()
```

#Sales breakdown by segment: Corporate leads at 50.3%, followed by Consumer (30.8%) and Home Office (18.9%).

[30]:
```
df.columns
```

[30]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country/Region', 'City',
       'State/Province', 'Postal Code', 'Region', 'Product ID', 'Category',
       'Sub-Category', 'Product Name', 'Sales', 'Quantity', 'Discount',
       'Profit'],
```

```
       dtype='object')
```

[31]:
```
df.drop(["Order ID","Customer Name","Row ID","Product ID","Customer ID","Postal
↪Code"],axis=1,inplace=True)
```

[32]:
```
df.columns
```

[32]:
```
Index(['Order Date', 'Ship Date', 'Ship Mode', 'Segment', 'Country/Region',
       'City', 'State/Province', 'Region', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

[33]:
```
df["Sub-Category"]
```

[33]:
```
0            Paper
1          Binders
2           Labels
3          Storage
4              Art
           …
10189      Binders
10190      Binders
10191    Fasteners
10192       Phones
10193      Binders
Name: Sub-Category, Length: 10194, dtype: object
```

[34]:
```
df_category =df.groupby('Sub-Category')['Sales'].sum().reset_index()
df_category
```

[34]:

	Sub-Category	Sales
0	Accessories	167380.3180
1	Appliances	108213.1850
2	Art	27659.0140
3	Binders	207354.8810
4	Bookcases	115361.2043
5	Chairs	335768.2490
6	Copiers	150745.2900
7	Envelopes	16528.3620
8	Fasteners	8532.2400
9	Furnishings	95598.1260
10	Labels	12695.0420
11	Machines	189925.0310
12	Paper	79540.5380
13	Phones	331842.6400
14	Storage	224644.5540
15	Supplies	46725.4980

```
16       Tables  208020.1820
```

[35]:
```
fig =px.bar(df_category, y = 'Sub-Category', x = 'Sales', title = "Sales by␣
 ↪Sub-Category")
fig.show()
```

#It uses data (likely sales figures) from a DataFrame and displays a graph with:

Bars showing how many sales fall into each price range. A smooth line (optional) for a more complete view of the distribution. Labels for the axes (Sales and Frequency).

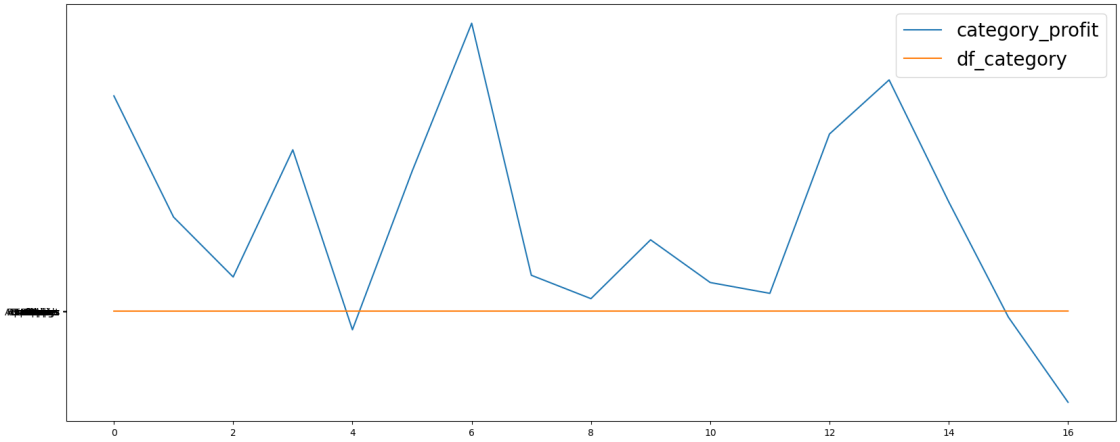
[36]:
```
category_profit =df.groupby('Sub-Category')['Profit'].sum().reset_index()
```

[37]:
```
category_profit
```

[37]:

	Sub-Category	Profit
0	Accessories	41936.6357
1	Appliances	18329.4844
2	Art	6653.1962
3	Binders	31426.1003
4	Bookcases	-3632.0736
5	Chairs	27223.5323
6	Copiers	56093.9365
7	Envelopes	6988.0247
8	Fasteners	2428.6358
9	Furnishings	13891.7430
10	Labels	5572.7780
11	Machines	3461.9769
12	Paper	34511.5070
13	Phones	45050.8265
14	Storage	21285.1115
15	Supplies	-1171.3945
16	Tables	-17753.2061

[38]:
```
plt.figure(figsize = (20,8))
#plt.title("Profits of Sub-Category", fontsize = 20)
plt.plot(category_profit.index,category_profit['Profit'], label =␣
 ↪"category_profit")
plt.plot(category_profit.index,category_profit['Sub-Category'],label =␣
 ↪"df_category")
plt.legend(fontsize = 20)
plt.show()
```



The graph shows sales distribution across product categories. It uses boxes to represent the spread of sales figures for each category:

The box shows the middle (median), the lower portion (Q1), and the upper portion (Q3) of sales in that category. Lines (whiskers) extend from the box showing the range of most data points. Diamonds (optional) might show the average sales for each category. By comparing boxes, you can see how sales differ between categories.

[39]:
```
df_Sub_Category_sale_profit = df.groupby('Sub-Category')[['Profit', 'Sales',␣
↪'Discount']].sum().reset_index()
```

[40]:
```
df_Sub_Category_sale_profit
```

[40]:

	Sub-Category	Profit	Sales	Discount
0	Accessories	41936.6357	167380.3180	60.80
1	Appliances	18329.4844	108213.1850	78.00
2	Art	6653.1962	27659.0140	61.60
3	Binders	31426.1003	207354.8810	571.30
4	Bookcases	-3632.0736	115361.2043	49.94
5	Chairs	27223.5323	335768.2490	107.30
6	Copiers	56093.9365	150745.2900	11.00
7	Envelopes	6988.0247	16528.3620	20.40
8	Fasteners	2428.6358	8532.2400	18.00
9	Furnishings	13891.7430	95598.1260	139.30
10	Labels	5572.7780	12695.0420	26.10
11	Machines	3461.9769	189925.0310	35.60
12	Paper	34511.5070	79540.5380	103.80
13	Phones	45050.8265	331842.6400	137.80
14	Storage	21285.1115	224644.5540	64.30
15	Supplies	-1171.3945	46725.4980	14.60
16	Tables	-17753.2061	208020.1820	84.15

[41]:
```
fig = px.bar(df_Sub_Category_sale_profit, x = 'Sub-Category',y =␣
↪['Profit','Sales','Discount'])
fig.show()
```

This graph shows the relationship between profit and discount for products. Each dot represents a product, with its position showing its profit (horizontal) and discount offered (vertical). There's a positive trend, suggesting higher discounts are associated with higher profits for these products. However, keep in mind that correlation doesn't equal causation - other factors could be at play.

[42]:
```
df_region_sales = df.groupby('Region')[['Profit', 'Sales', 'Discount']].sum().
↪reset_index()
```

[43]:
```
df_region_sales
```

[43]:

	Region	Profit	Sales	Discount
0	Central	39865.3070	503170.6728	562.74
1	East	94883.2603	691828.1680	428.30
2	South	46749.4303	391721.9050	238.55
3	West	110798.8170	739813.6085	354.40

[44]:
```
fig = px.bar(df_region_sales, x = 'Region',y = ['Sales','Profit','Discount'])
fig.show()
```

This graph tracks website visits over time. The horizontal axis shows time (days, weeks, etc.), and the vertical axis shows the number of visits. The line connecting the dots represents the trend in visits. By looking at the slope of the line, you can see if visits are increasing, decreasing, or flat over time.

[45]:
```
fig = px.bar(df_region_sales, x = 'Region',y = ['Sales','Profit','Discount'])
fig.show()
```

This graph shows customer preferences by age group for different product categories. The horizontal axis shows age groups, and the vertical bars represent the total number of customers in each age group. Colors within each bar represent product categories (e.g., blue for smartphones). The height of each colored section shows how many customers in that age group bought that product category. For example, it appears young adults (18-24) buy more smartphones than laptops or tablets.

[46]:
```
fig = px.pie(df_region_sales, values= 'Sales', names='Region',title= "Sales by␣
↪Region")
fig.show()
```

This graph summarizes sales across different countries and product types. Countries are listed vertically and product types horizontally. The color intensity in each box shows sales for that combination. Darker colors represent higher sales (e.g., France might have strong laptop sales), while lighter colors represent lower sales (e.g., India might have low appliance sales). This helps identify sales trends by country and product type.

[47]:
```
df_profit_sales = df.groupby('Region')[['Profit']].sum().reset_index()
```

[48]:
```
df_profit_sales
```

[48]:

	Region	Profit
0	Central	39865.3070
1	East	94883.2603
2	South	46749.4303
3	West	110798.8170

[49]:
```
fig = px.pie(df_profit_sales, values='Profit', names='Region',title="profit by␣
↪reg")
fig.show()
```

This graph tracks sales figures over a year. The horizontal axis shows the months, and the vertical axis shows the sales amount in dollars. The line connecting the dots represents the trend in sales. It appears sales are seasonal, with higher sales in the spring and summer months and lower sales in the fall and winter.

[50]:
```
fig = px.bar(df_profit_sales, y='Profit',x='Region',title="sales by Profit")
fig.show()
```

This graph shows sales broken down by product type. The blue bars represent total sales for each type (e.g., shirts, jeans). The higher the bar, the higher the sales for that product. The line shows the cumulative percentage of total sales (right axis, likely labeled "% of Total Sales").

It seems clothing like shirts, jeans, and sweaters contribute the most (tall blue bars). The line reaching 80% around "Sweater" suggests these top sellers might make up 80% of total sales, following the 80/20 rule (where a small portion of items contributes to a large portion of sales).

[51]:
```
df.Sales.sum()
```

[51]:
```
2326534.3543
```

[52]:
```
df.Profit.sum()
```

[52]:
```
292296.81460000004
```

[53]:
```
df.Quantity.sum()
```

[53]:
```
38654
```

[54]:
```
df.Discount.sum()
```

[54]:
```
1583.99
```

[55]:
```
df.max()
```

[55]:
```
Order Date                      2022-12-30 00:00:00
Ship Date                       2023-01-05 00:00:00
```

```
Ship Mode                              Standard Class
Segment                                   Home Office
Country/Region                         United States
City                                           Yuma
State/Province                              Wyoming
Region                                         West
Category                                 Technology
Sub-Category                                 Tables
Product Name      netTALK DUO VoIP Telephone Service
Sales                                      22638.48
Quantity                                         14
Discount                                        0.8
Profit                                     8399.976
dtype: object
```

[56]:
```
df.min()
```

[56]:
```
Order Date                                    2019-01-03 00:00:00
Ship Date                                     2019-01-07 00:00:00
Ship Mode                                             First Class
Segment                                              Consumer
Country/Region                                         Canada
City                                                 Aberdeen
State/Province                                        Alabama
Region                                               Central
Category                                           Furniture
Sub-Category                                     Accessories
Product Name      "While you Were Out" Message Book, One Form pe…
Sales                                                    0.444
Quantity                                                     1
Discount                                                   0.0
Profit                                                -6599.978
dtype: object
```

[57]:
```
max_quantity=0
for i in df['Quantity']:
    max_quantity+=i
print(max_quantity)
```

```
38654
```

[58]:
```
dir(df)
```

[58]:
```
['Category',
 'City',
 'Discount',
 'Profit',
```

```
'Quantity',
'Region',
'Sales',
'Segment',
'T',
'_AXIS_LEN',
'_AXIS_ORDERS',
'_AXIS_TO_AXIS_NUMBER',
'_HANDLED_TYPES',
'__abs__',
'__add__',
'__and__',
'__annotations__',
'__array__',
'__array_priority__',
'__array_ufunc__',
'__bool__',
'__class__',
'__contains__',
'__copy__',
'__dataframe__',
'__deepcopy__',
'__delattr__',
'__delitem__',
'__dict__',
'__dir__',
'__divmod__',
'__doc__',
'__eq__',
'__finalize__',
'__floordiv__',
'__format__',
'__ge__',
'__getattr__',
'__getattribute__',
'__getitem__',
'__getstate__',
'__gt__',
'__hash__',
'__iadd__',
'__iand__',
'__ifloordiv__',
'__imod__',
'__imul__',
'__init__',
'__init_subclass__',
'__invert__',
```

```
'__ior__',
'__ipow__',
'__isub__',
'__iter__',
'__itruediv__',
'__ixor__',
'__le__',
'__len__',
'__lt__',
'__matmul__',
'__mod__',
'__module__',
'__mul__',
'__ne__',
'__neg__',
'__new__',
'__nonzero__',
'__or__',
'__pos__',
'__pow__',
'__radd__',
'__rand__',
'__rdivmod__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rfloordiv__',
'__rmatmul__',
'__rmod__',
'__rmul__',
'__ror__',
'__round__',
'__rpow__',
'__rsub__',
'__rtruediv__',
'__rxor__',
'__setattr__',
'__setitem__',
'__setstate__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__truediv__',
'__weakref__',
'__xor__',
'_accessors',
```

```
'_accum_func',
'_add_numeric_operations',
'_agg_examples_doc',
'_agg_summary_and_see_also_doc',
'_align_frame',
'_align_series',
'_append',
'_arith_method',
'_as_manager',
'_attrs',
'_box_col_values',
'_can_fast_transpose',
'_check_inplace_and_allows_duplicate_labels',
'_check_inplace_setting',
'_check_is_chained_assignment_possible',
'_check_label_or_level_ambiguity',
'_check_setitem_copy',
'_clear_item_cache',
'_clip_with_one_bound',
'_clip_with_scalar',
'_cmp_method',
'_combine_frame',
'_consolidate',
'_consolidate_inplace',
'_construct_axes_dict',
'_construct_result',
'_constructor',
'_constructor_sliced',
'_create_data_for_split_and_tight_to_dict',
'_data',
'_dir_additions',
'_dir_deletions',
'_dispatch_frame_op',
'_drop_axis',
'_drop_labels_or_levels',
'_ensure_valid_index',
'_find_valid_index',
'_flags',
'_from_arrays',
'_get_agg_axis',
'_get_axis',
'_get_axis_name',
'_get_axis_number',
'_get_axis_resolvers',
'_get_block_manager_axis',
'_get_bool_data',
'_get_cleaned_column_resolvers',
```

```
'_get_column_array',
'_get_index_resolvers',
'_get_item_cache',
'_get_label_or_level_values',
'_get_numeric_data',
'_get_value',
'_getitem_bool_array',
'_getitem_multilevel',
'_getitem_nocopy',
'_gotitem',
'_hidden_attrs',
'_indexed_same',
'_info_axis',
'_info_axis_name',
'_info_axis_number',
'_info_repr',
'_init_mgr',
'_inplace_method',
'_internal_names',
'_internal_names_set',
'_is_copy',
'_is_homogeneous_type',
'_is_label_or_level_reference',
'_is_label_reference',
'_is_level_reference',
'_is_mixed_type',
'_is_view',
'_iset_item',
'_iset_item_mgr',
'_iset_not_inplace',
'_item_cache',
'_iter_column_arrays',
'_ixs',
'_join_compat',
'_logical_func',
'_logical_method',
'_maybe_cache_changed',
'_maybe_update_cacher',
'_metadata',
'_mgr',
'_min_count_stat_function',
'_needs_reindex_multi',
'_protect_consolidate',
'_reduce',
'_reduce_axis1',
'_reindex_axes',
'_reindex_columns',
```

```
'_reindex_index',
'_reindex_multi',
'_reindex_with_indexers',
'_rename',
'_replace_columnwise',
'_repr_data_resource_',
'_repr_fits_horizontal_',
'_repr_fits_vertical_',
'_repr_html_',
'_repr_latex_',
'_reset_cache',
'_reset_cacher',
'_sanitize_column',
'_series',
'_set_axis',
'_set_axis_name',
'_set_axis_nocheck',
'_set_is_copy',
'_set_item',
'_set_item_frame_value',
'_set_item_mgr',
'_set_value',
'_setitem_array',
'_setitem_frame',
'_setitem_slice',
'_slice',
'_stat_axis',
'_stat_axis_name',
'_stat_axis_number',
'_stat_function',
'_stat_function_ddof',
'_take',
'_take_with_is_copy',
'_to_dict_of_blocks',
'_to_latex_via_styler',
'_typ',
'_update_inplace',
'_validate_dtype',
'_values',
'_where',
'abs',
'add',
'add_prefix',
'add_suffix',
'agg',
'aggregate',
'align',
```

```
'all',
'any',
'apply',
'applymap',
'asfreq',
'asof',
'assign',
'astype',
'at',
'at_time',
'attrs',
'axes',
'backfill',
'between_time',
'bfill',
'bool',
'boxplot',
'clip',
'columns',
'combine',
'combine_first',
'compare',
'convert_dtypes',
'copy',
'corr',
'corrwith',
'count',
'cov',
'cummax',
'cummin',
'cumprod',
'cumsum',
'describe',
'diff',
'div',
'divide',
'dot',
'drop',
'drop_duplicates',
'droplevel',
'dropna',
'dtypes',
'duplicated',
'empty',
'eq',
'equals',
'eval',
```

```
'ewm',
'expanding',
'explode',
'ffill',
'fillna',
'filter',
'first',
'first_valid_index',
'flags',
'floordiv',
'from_dict',
'from_records',
'ge',
'get',
'groupby',
'gt',
'head',
'hist',
'iat',
'idxmax',
'idxmin',
'iloc',
'index',
'infer_objects',
'info',
'insert',
'interpolate',
'isetitem',
'isin',
'isna',
'isnull',
'items',
'iterrows',
'itertuples',
'join',
'keys',
'kurt',
'kurtosis',
'last',
'last_valid_index',
'le',
'loc',
'lt',
'mask',
'max',
'mean',
'median',
```

```
'melt',
'memory_usage',
'merge',
'min',
'mod',
'mode',
'mul',
'multiply',
'ndim',
'ne',
'nlargest',
'notna',
'notnull',
'nsmallest',
'nunique',
'pad',
'pct_change',
'pipe',
'pivot',
'pivot_table',
'plot',
'pop',
'pow',
'prod',
'product',
'quantile',
'query',
'radd',
'rank',
'rdiv',
'reindex',
'reindex_like',
'rename',
'rename_axis',
'reorder_levels',
'replace',
'resample',
'reset_index',
'rfloordiv',
'rmod',
'rmul',
'rolling',
'round',
'rpow',
'rsub',
'rtruediv',
'sample',
```

```
'select_dtypes',
'sem',
'set_axis',
'set_flags',
'set_index',
'shape',
'shift',
'size',
'skew',
'sort_index',
'sort_values',
'squeeze',
'stack',
'std',
'style',
'sub',
'subtract',
'sum',
'swapaxes',
'swaplevel',
'tail',
'take',
'to_clipboard',
'to_csv',
'to_dict',
'to_excel',
'to_feather',
'to_gbq',
'to_hdf',
'to_html',
'to_json',
'to_latex',
'to_markdown',
'to_numpy',
'to_orc',
'to_parquet',
'to_period',
'to_pickle',
'to_records',
'to_sql',
'to_stata',
'to_string',
'to_timestamp',
'to_xarray',
'to_xml',
'transform',
'transpose',
```

```
 'truediv',
 'truncate',
 'tz_convert',
 'tz_localize',
 'unstack',
 'update',
 'value_counts',
 'values',
 'var',
 'where',
 'xs']
```

[59]:
```
total_sales= 0
total_profit= 0
total_quantity= 0
for index,row in df.iterrows():
    total_sales+=row['Sales']
    total_profit+=row['Profit']
    total_quantity+=row['Quantity']
print(f"total_sales are: {total_sales}")
print(f"total_profit are: {total_profit}")
print(f"total_quantity are: {total_quantity}")
print()
```

```
total_sales are: 2326534.3542999476
total_profit are: 292296.8145999994
total_quantity are: 38654
```

[60]:
```
total_data={"total_sales":[total_sales],
"total_profit":[total_profit],
"total_quantity":[total_quantity]}
```

[61]:
```
total_data
```

[61]:
```
{'total_sales': [2326534.3542999476],
 'total_profit': [292296.8145999994],
 'total_quantity': [38654]}
```

[62]:
```
total_value=pd.DataFrame(total_data)
```

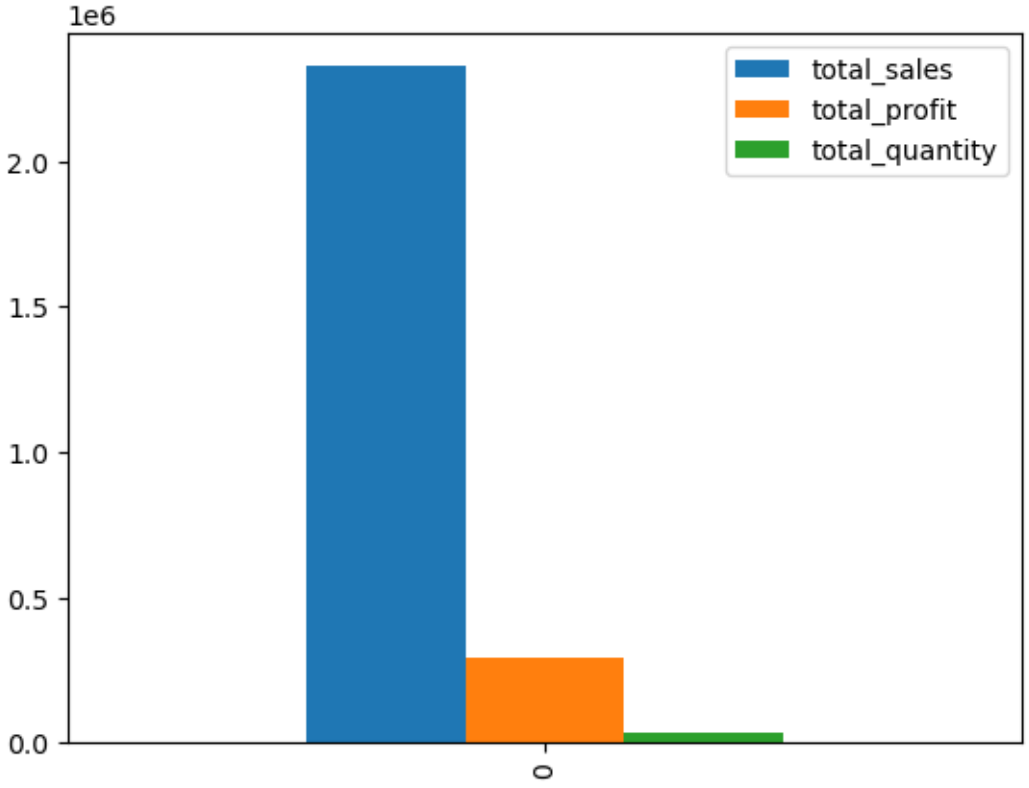
[63]:
```
total_value
```

[63]:

	total_sales	total_profit	total_quantity
0	2.326534e+06	292296.8146	38654

[64]:
```
total_value.plot(kind='bar')

plt.show()
```



This graph shows the relationship between advertising cost and sales for products (or product categories). Each dot represents a product, with its horizontal position showing advertising cost and its vertical position showing sales made. The line shows the general trend: higher advertising cost is associated with higher sales. However, remember that correlation doesn't equal causation. Other factors besides advertising spending could be influencing sales.