project4

April 19, 2024

1 Import Lib

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import KNNImputer
import statsmodels.api as sm
from scipy.stats import pearsonr
from scipy.stats import f_oneway
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from scipy.stats import ttest_ind
```

2 Reading Dataset

[3]:
```
df=pd.read_excel(r"C:\Users\ganesh\Desktop\Sample - Superstore (1).xlsx")  #␣
 ↪reading excel file
pd.set_option("display.max_column",22)
df.head()     #to display top 5 rows
```

[3]:
```
   Row ID        Order ID Order Date  Ship Date       Ship Mode Customer ID  \
0     1.0  CA-2016-152156 2016-11-08 2016-11-11    Second Class    CG-12520
1     2.0  CA-2016-152156 2016-11-08 2016-11-11    Second Class    CG-12520
2     3.0  CA-2016-138688 2016-06-12 2016-06-16    Second Class    DV-13045
3     4.0  US-2015-108966 2015-10-11 2015-10-18  Standard Class    SO-20335
4     5.0  US-2015-108966 2015-10-11 2015-10-18  Standard Class    SO-20335

     Customer Name    Segment        Country             City       State  \
0      Claire Gute   Consumer  United States        Henderson    Kentucky
1      Claire Gute   Consumer  United States        Henderson    Kentucky
2  Darrin Van Huff  Corporate  United States      Los Angeles  California
3   Sean O'Donnell   Consumer  United States  Fort Lauderdale     Florida
4   Sean O'Donnell   Consumer  United States  Fort Lauderdale     Florida

   Postal Code Region       Product ID         Category Sub-Category  \
0     42420.0  South  FUR-BO-10001798        Furniture    Bookcases
```

```
1      42420.0  South  FUR-CH-10000454        Furniture       Chairs
2      90036.0   West  OFF-LA-10000240  Office Supplies       Labels
3      33311.0  South  FUR-TA-10000577        Furniture       Tables
4      33311.0  South  OFF-ST-10000760  Office Supplies      Storage

                                        Product Name     Sales  Quantity  \
0                    Bush Somerset Collection Bookcase    261.96       2.0
1  Hon Deluxe Fabric Upholstered Stacking Chairs,…    731.94       3.0
2  Self-Adhesive Address Labels for Typewriters b…     14.62       2.0
3      Bretford CR4500 Series Slim Rectangular Table  957.5775       5.0
4                         Eldon Fold 'N Roll Cart System    22.368       2.0

   Discount    Profit
0      0.00   41.9136
1      0.00  219.5820
2      0.00    6.8714
3      0.45 -383.0310
4      0.20    2.5164
```

[4]:
```
df.shape    #shape of dataframe
```

[4]:
```
(9994, 21)
```

[5]:
```
df.info()  #information of dataframe
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9985 non-null   float64
 1   Order ID       9981 non-null   object
 2   Order Date     9981 non-null   datetime64[ns]
 3   Ship Date      9979 non-null   datetime64[ns]
 4   Ship Mode      9975 non-null   object
 5   Customer ID    9968 non-null   object
 6   Customer Name  9937 non-null   object
 7   Segment        9942 non-null   object
 8   Country        9930 non-null   object
 9   City           9949 non-null   object
 10  State          9937 non-null   object
 11  Postal Code    9957 non-null   float64
 12  Region         9954 non-null   object
 13  Product ID     9956 non-null   object
 14  Category       9965 non-null   object
 15  Sub-Category   9952 non-null   object
 16  Product Name   9936 non-null   object
```

```
 17  Sales          9932 non-null   object
 18  Quantity       9948 non-null   float64
 19  Discount       9957 non-null   float64
 20  Profit         9944 non-null   float64
dtypes: datetime64[ns](2), float64(5), object(14)
memory usage: 1.6+ MB
```

[6]:
```
df.isnull().sum() #finding number of null values in each column
```

[6]:
```
Row ID           9
Order ID        13
Order Date      13
Ship Date       15
Ship Mode       19
Customer ID     26
Customer Name   57
Segment         52
Country         64
City            45
State           57
Postal Code     37
Region          40
Product ID      38
Category        29
Sub-Category    42
Product Name    58
Sales           62
Quantity        46
Discount        37
Profit          50
dtype: int64
```

[7]:
```
df.isnull().sum()/df.shape[0]*100   #percentage of missing values in each␣
↪columns
```

[7]:
```
Row ID          0.090054
Order ID        0.130078
Order Date      0.130078
Ship Date       0.150090
Ship Mode       0.190114
Customer ID     0.260156
Customer Name   0.570342
Segment         0.520312
Country         0.640384
City            0.450270
State           0.570342
Postal Code     0.370222
```

```
Region           0.400240
Product ID       0.380228
Category         0.290174
Sub-Category     0.420252
Product Name     0.580348
Sales            0.620372
Quantity         0.460276
Discount         0.370222
Profit           0.500300
dtype: float64
```

[8]:
```
df.duplicated().sum()  #duplicate rows
```

[8]:
```
10
```

[9]:
```
for i in df.select_dtypes(include="object").columns:   #counts number of unique values
    print(df[i].value_counts())
    print("***"*10)
```

```
CA-2017-100111    14
CA-2017-157987    12
US-2016-108504    11
CA-2016-165330    11
US-2015-126977    10
                  ..
US-2017-107636     1
US-2014-165862     1
CA-2016-101448     1
US-2017-117331     1
CA-2017-119914     1
Name: Order ID, Length: 5000, dtype: int64
******************************
Standard Class    5958
Second Class      1940
First Class       1534
Same Day           543
Name: Ship Mode, dtype: int64
******************************
WB-21850    37
AP-10915    35
PP-18955    34
JL-15835    34
MA-17560    34
            ..
LD-16855     1
AO-10810     1
```

```
CJ-11875     1
RE-19405     1
JR-15700     1
Name: Customer ID, Length: 793, dtype: int64
******************************
William Brown        37
Arthur Prichep       35
John Lee             34
Paul Prost           34
Matt Abelman         34
                     ..
Lela Donovan          1
Jocasta Rupert        1
Carl Jackson          1
Anthony O'Donnell     1
Ricardo Emerson       1
Name: Customer Name, Length: 792, dtype: int64
******************************
Consumer       5171
Corporate      2997
Home Office    1774
Name: Segment, dtype: int64
******************************
United States    9930
Name: Country, dtype: int64
******************************
New York City     913
Los Angeles       744
Philadelphia      529
San Francisco     507
Seattle           428
                  …
Cedar Rapids        1
Palatine            1
Jefferson City      1
Waterloo            1
Iowa City           1
Name: City, Length: 530, dtype: int64
******************************
California            1991
New York              1122
Texas                  975
Pennsylvania           578
Washington             506
                       …
31.744                   1
18.180000000000003       1
471.92                   1
```

```
89.584                  1
12.624                  1
Name: State, Length: 68, dtype: int64
******************************
West       3197
East       2828
Central    2305
South      1605
0            11
0.2           6
0.7           1
0.1           1
Name: Region, dtype: int64
******************************
OFF-PA-10001970    19
TEC-AC-10003832    18
FUR-FU-10004270    16
FUR-CH-10001146    15
TEC-AC-10002049    15
                   ..
TEC-MA-10002937     1
TEC-PH-10003535     1
OFF-AP-10002734     1
TEC-MA-10003353     1
OFF-ST-10001627     1
Name: Product ID, Length: 1878, dtype: int64
******************************
Office Supplies    6003
Furniture          2116
Technology         1846
Name: Category, dtype: int64
******************************
Binders        1512
Paper          1367
Furnishings     956
Phones          890
Storage         843
Art             787
Accessories     774
Chairs          612
Appliances      464
Labels          363
Tables          317
Envelopes       252
Bookcases       228
Fasteners       217
Supplies        190
Machines        114
```

```
Copiers          66
Name: Sub-Category, dtype: int64
******************************
Staple envelope                                                   48
Easy-staple paper                                                 46
Staples                                                           46
Avery Non-Stick Binders                                           20
Staples in misc. colors                                           18
                                                                  ..
Eldon File Chest Portable File                                     1
Hewlett-Packard Deskjet D4360 Printer                              1
Jiffy Padded Mailers with Self-Seal Closure                        1
Hunt BOSTON Model 1606 High-Volume Electric Pencil Sharpener, Beige    1
Eldon Jumbo ProFile Portable File Boxes Graphite/Black             1
Name: Product Name, Length: 1857, dtype: int64
******************************
12.960     54
15.552     39
19.440     39
10.368     35
32.400     28
           ..
487.920     1
25.920      1
95.736      1
3.392       1
275.880     1
Name: Sales, Length: 6128, dtype: int64
******************************
```

#Converting DataType of sales from object to float (datatype was object because there were some string values in column,this process convert string to nan)

[10]:
```
df["Sales"]=pd.to_numeric(df["Sales"],errors='coerce')
pro=df[df["Sales"].isna()]
print(pro)
```

```
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
193    194.0  CA-2015-102281 2015-10-12 2015-10-14     First Class
194    195.0  CA-2015-131457 2015-10-31 2015-11-06  Standard Class
195    196.0  CA-2014-140004 2014-03-21 2014-03-25  Standard Class
196    197.0  CA-2014-140004 2014-03-21 2014-03-25  Standard Class
197    198.0  CA-2017-107720 2017-11-06 2017-11-13  Standard Class
…      …              …         …         …              …
7177  7187.0  CA-2017-133102 2017-08-17 2017-08-24  Standard Class
7178  7188.0  CA-2017-133102 2017-08-17 2017-08-24  Standard Class
7179  7189.0  CA-2017-133102 2017-08-17 2017-08-24  Standard Class
7180  7190.0  CA-2016-164399 2016-11-12 2016-11-15     First Class
```

```
7181  7191.0  CA-2016-164399 2016-11-12 2016-11-15     First Class

     Customer ID      Customer Name      Segment        Country  \
193     MP-17470        Mark Packer  Home Office  United States
194     MZ-17515          Mary Zewe    Corporate  United States
195     CB-12025  Cassandra Brandow     Consumer  United States
196     CB-12025  Cassandra Brandow     Consumer  United States
197     VM-21685    Valerie Mitchum  Home Office  United States
…          …                …           …            …
7177    ED-13885       Emily Ducich  Home Office  United States
7178    ED-13885       Emily Ducich  Home Office  United States
7179    ED-13885       Emily Ducich  Home Office  United States
7180    DW-13480      Dianna Wilson  Home Office  United States
7181    DW-13480      Dianna Wilson  Home Office  United States

               City       State  Postal Code   Region       Product ID  \
193   New York City    New York      10035.0     East              NaN
194        Redlands  California      92374.0     West              NaN
195        Hamilton        Ohio      45011.0     East              NaN
196        Hamilton        Ohio      45011.0     East              NaN
197       Westfield  New Jersey       7090.0     East              NaN
…             …         …           …       …              …
7177        Houston       Texas      77095.0  Central  FUR-CH-10002017
7178        Houston       Texas      77095.0  Central  FUR-FU-10003247
7179        Houston       Texas      77095.0  Central  OFF-AP-10001563
7180      San Diego  California      92024.0     West  TEC-PH-10004908
7181      San Diego  California      92024.0     West  FUR-TA-10003392

             Category Sub-Category Product Name  Sales  Quantity  Discount  \
193               NaN          NaN          NaN    NaN       NaN       NaN
194               NaN          NaN          NaN    NaN       NaN       NaN
195               NaN          NaN          NaN    NaN       NaN       NaN
196               NaN          NaN          NaN    NaN       NaN       NaN
197               NaN          NaN          NaN    NaN       NaN       NaN
…                 …            …            …      …         …         …
7177        Furniture       Chairs          NaN    NaN       NaN       NaN
7178        Furniture  Furnishings          NaN    NaN       NaN       NaN
7179  Office Supplies   Appliances          NaN    NaN       NaN       NaN
7180       Technology       Phones          NaN    NaN       NaN       NaN
7181        Furniture       Tables          NaN    NaN       NaN       NaN

      Profit
193      NaN
194      NaN
195      NaN
196      NaN
197      NaN
…        …
```

```
7177     NaN
7178     NaN
7179     NaN
7180     NaN
7181     NaN

[83 rows x 21 columns]
```

[11]:
```
df.isnull().sum() #checking number of null values for eah column
```

[11]:
```
Row ID           9
Order ID        13
Order Date      13
Ship Date       15
Ship Mode       19
Customer ID     26
Customer Name   57
Segment         52
Country         64
City            45
State           57
Postal Code     37
Region          40
Product ID      38
Category        29
Sub-Category    42
Product Name    58
Sales           83
Quantity        46
Discount        37
Profit          50
dtype: int64
```

[12]:
```
df[df["Row ID"]==6858] #checking row that has changed from string to nan
```

[12]:
```
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6857  6858.0  CA-2017-128965 2017-04-17 2017-04-22  Standard Class

     Customer ID Customer Name   Segment        Country         City  \
6857    PS-18760  Pamela Stobb  Consumer  United States  Los Angeles

           State  Postal Code Region       Product ID         Category  \
6857  California      90008.0   West  OFF-PA-10004911  Office Supplies

     Sub-Category Product Name  Sales  Quantity  Discount  Profit
6857        Paper        Paper    NaN     28.14       3.0     0.0
```

[13]:
```
df.dropna(subset=["Sales"],inplace=True)  #droping null values of sales(we cant␣
↪impute data in sales(dependent variable) because it creates a arbitary␣
↪values whih affects model prediction)
```

[14]:
```
df.shape #shape of dataframe
```

[14]:
```
(9911, 21)
```

[15]:
```
df["Sales"].dtypes #cheking dataframe of sales column
```

[15]:
```
dtype('float64')
```

[16]:
```
df.isnull().sum() #checking number of null values
```

[16]:
```
Row ID            9
Order ID         13
Order Date       13
Ship Date        15
Ship Mode        19
Customer ID      26
Customer Name    46
Segment          41
Country          53
City             45
State            51
Postal Code      31
Region           31
Product ID       11
Category          2
Sub-Category      4
Product Name      4
Sales             0
Quantity          0
Discount          0
Profit           13
dtype: int64
```

[17]:
```
df["City"].isnull().sum() #cheking number of null values in city
```

[17]:
```
45
```

[18]:
```
df["City"].unique() #hecking unique values in city column
```

[18]:
```
array(['Henderson', 'Los Angeles', 'Fort Lauderdale', 'Concord',
       'Seattle', 'Fort Worth', 'Madison', 'West Jordan', 'San Francisco',
       'Fremont', 'Philadelphia', 'Orem', 'Houston', 'Richardson',
       'Naperville', 'Melbourne', 'Eagan', 'Westland', 'Dover',
```

'New Albany', 'New York City', 'Troy', 'Chicago', 'Gilbert',
'Springfield', 'Jackson', 'Memphis', 'Decatur', 'Durham',
'Columbia', 'Rochester', nan, 'Aurora', 'Charlotte', 'Orland Park',
'Urbandale', 'Columbus', 'Bristol', 'Wilmington', 'Bloomington',
'Phoenix', 'Roseville', 'Independence', 'Pasadena', 'Newark',
'Franklin', 'Scottsdale', 'San Jose', 'Edmond', 'Carlsbad',
'San Antonio', 'Monroe', 'Fairfield', 'Grand Prairie', 'Denver',
'Dallas', 'Whittier', 'Saginaw', 'Medina', 'Detroit', 'Tampa',
'Santa Clara', 'Lakeville', 'San Diego', 'Brentwood',
'Chapel Hill', 'Morristown', 'Cincinnati', 'Inglewood', 'Portland',
'Tamarac', 'Colorado Springs', 'Belleville', 'Taylor', 'Lakewood',
'Arlington', 'Arvada', 'Hackensack', 'Saint Petersburg',
'Long Beach', 'Hesperia', 'Murfreesboro', 'Austin', 'Lowell',
'Manchester', 'Harlingen', 'Tucson', 'Quincy', 'Pembroke Pines',
'Des Moines', 'Peoria', 'Las Vegas', 'Warwick', 'Miami',
'Huntington Beach', 'Richmond', 'Louisville', 'Lawrence', 'Canton',
'New Rochelle', 'Gastonia', 'Jacksonville', 'Auburn', 'Akron',
'Norman', 'Park Ridge', 'Amarillo', 'Lindenhurst', 'Huntsville',
'Fayetteville', 'Costa Mesa', 'Parker', 'Atlanta', 'Gladstone',
'Great Falls', 'Montgomery', 'Mesa', 'Green Bay', 'Anaheim',
'Marysville', 'Salem', 'Laredo', 'Grove City', 'Dearborn',
'Warner Robins', 'Vallejo', 'Minneapolis', 'Mission Viejo',
'Rochester Hills', 'Plainfield', 'Sierra Vista', 'Vancouver',
'Cleveland', 'Tyler', 'Burlington', 'Waynesboro', 'Chester',
'Cary', 'Palm Coast', 'Mount Vernon', 'Hialeah', 'Oceanside',
'Evanston', 'Trenton', 'Cottage Grove', 'Bossier City',
'Lancaster', 'Asheville', 'Lake Elsinore', 'Omaha', 'Edmonds',
'Santa Ana', 'Milwaukee', 'Florence', 'Lorain', 'Linden',
'Salinas', 'New Brunswick', 'Garland', 'Norwich', 'Alexandria',
'Toledo', 'Farmington', 'Riverside', 'Torrance', 'Round Rock',
'Boca Raton', 'Virginia Beach', 'Murrieta', 'Olympia',
'Washington', 'Jefferson City', 'Saint Peters', 'Rockford',
'Brownsville', 'Yonkers', 'Oakland', 'Clinton', 'Encinitas',
'Roswell', 'Jonesboro', 'Antioch', 'Homestead', 'La Porte',
'Lansing', 'Cuyahoga Falls', 'Reno', 'Harrisonburg', 'Escondido',
'Royal Oak', 'Rockville', 'Coral Springs', 'Buffalo',
'Boynton Beach', 'Gulfport', 'Fresno', 'Greenville', 'Macon',
'Cedar Rapids', 'Providence', 'Pueblo', 'Saint Paul', 'Deltona',
'Murray', 'Middletown', 'Freeport', 'Pico Rivera', 'Provo',
'Pleasant Grove', 'Smyrna', 'Parma', 'Mobile', 'New Bedford',
'Irving', 'Vineland', 'Glendale', 'Niagara Falls', 'Thomasville',
'Westminster', 'Coppell', 'Pomona', 'North Las Vegas', 'Allentown',
'Tempe', 'Laguna Niguel', 'Bridgeton', 'Everett', 'Watertown',
'Appleton', 'Bellevue', 'Allen', 'El Paso', 'Grapevine',
'Carrollton', 'Kent', 'Lafayette', 'Tigard', 'Skokie', 'Plano',
'Suffolk', 'Indianapolis', 'Bayonne', 'Dublin', 'Greensboro',
'Baltimore', 'Kenosha', 'Olathe', 'Tulsa', 'Redmond', 'Raleigh',

'Muskogee', 'Meriden', 'Bowling Green', 'South Bend', 'Spokane',
'Keller', 'Port Orange', 'Medford', 'Charlottesville', 'Missoula',
'Apopka', 'Reading', 'Broomfield', 'Paterson', 'Oklahoma City',
'Chesapeake', 'Lubbock', 'Johnson City', 'San Bernardino',
'Leominster', 'Bozeman', 'Perth Amboy', 'Ontario',
'Rancho Cucamonga', 'Moorhead', 'Mesquite', 'Redlands', 'Stockton',
'Ormond Beach', 'Sunnyvale', 'York', 'College Station',
'Saint Louis', 'Manteca', 'San Angelo', 'Salt Lake City',
'Knoxville', 'Little Rock', 'Lincoln Park', 'Marion', 'Littleton',
'Bangor', 'Southaven', 'New Castle', 'Midland', 'Sioux Falls',
'Fort Collins', 'Clarksville', 'Sacramento', 'Thousand Oaks',
'Malden', 'Holyoke', 'Albuquerque', 'Sparks', 'Coachella',
'Elmhurst', 'Passaic', 'North Charleston', 'Newport News',
'Jamestown', 'Mishawaka', 'Westfield', 'La Quinta', 'Tallahassee',
'Nashville', 'Bellingham', 'Woodstock', 'Haltom City', 'Wheeling',
'Summerville', 'Hot Springs', 'Englewood', 'Las Cruces', 'Hoover',
'Frisco', 'Vacaville', 'Waukesha', 'Bakersfield', 'Pompano Beach',
'Corpus Christi', 'Redondo Beach', 'Orlando', 'Orange',
'Lake Charles', 'Highland Park', 'Hempstead', 'Noblesville',
'Apple Valley', 'Mount Pleasant', 'Sterling Heights', 'Eau Claire',
'Pharr', 'Billings', 'Gresham', 'Chattanooga', 'Meridian',
'Bolingbrook', 'Lakeland', 'Maple Grove', 'Woodland',
'Missouri City', 'Pearland', 'San Mateo', 'Grand Rapids',
'Visalia', 'Overland Park', 'Temecula', 'Yucaipa', 'Revere',
'Conroe', 'Tinley Park', 'Dubuque', 'Dearborn Heights', 'Santa Fe',
'Hickory', 'Carol Stream', 'Saint Cloud', 'North Miami',
'Plantation', 'Port Saint Lucie', 'Rock Hill', 'Odessa',
'West Allis', 'Chula Vista', 'Manhattan', 'Altoona', 'Thornton',
'Champaign', 'Texarkana', 'Edinburg', 'Baytown', 'Greenwood',
'Woonsocket', 'Superior', 'Bedford', 'Covington', 'Broken Arrow',
'Miramar', 'Hollywood', 'Deer Park', 'Wichita', 'Mcallen',
'Iowa City', 'Boise', 'Cranston', 'Port Arthur', 'Citrus Heights',
'The Colony', 'Daytona Beach', 'Bullhead City', 'Portage', 'Fargo',
'Elkhart', 'San Gabriel', 'Hamilton', 'Margate', 'Sandy Springs',
'Mentor', 'Lawton', 'Hampton', 'Rome', 'La Crosse', 'Lewiston',
'Hattiesburg', 'Danville', 'Logan', 'Waterbury', 'Athens',
'Avondale', 'Marietta', 'Yuma', 'Wausau', 'Pasco', 'Oak Park',
'Pensacola', 'League City', 'Gaithersburg', 'Lehi', 'Tuscaloosa',
'Moreno Valley', 'Georgetown', 'Loveland', 'Chandler', 'Helena',
'Kirkwood', 'Waco', 'Frankfort', 'Bethlehem', 'Grand Island',
'Woodbury', 'Rogers', 'Clovis', 'Jupiter', 'Santa Barbara',
'Cedar Hill', 'Norfolk', 'Draper', 'Ann Arbor', 'La Mesa',
'Pocatello', 'Holland', 'Milford', 'Buffalo Grove', 'Lake Forest',
'Redding', 'Chico', 'Utica', 'Conway', 'Cheyenne', 'Owensboro',
'Caldwell', 'Kenner', 'Nashua', 'Bartlett', 'Redwood City',
'Lebanon', 'Santa Maria', 'Des Plaines', 'Longview',
'Hendersonville', 'Waterloo', 'Cambridge', 'Palatine', 'Beverly',

```
       'Eugene', 'Oxnard', 'Renton', 'Glenview', 'Delray Beach',
       'Commerce City', 'Texas City', 'Wilson', 'Rio Rancho', 'Goldsboro',
       'Montebello', 'El Cajon', 'West Palm Beach', 'Abilene', 'Normal',
       'Saint Charles', 'Camarillo', 'Hillsboro', 'Burbank', 'Modesto',
       'Garden City', 'Atlantic City', 'Longmont', 'Davis', 'Morgan Hill',
       'Clifton', 'Sheboygan', 'East Point', 'Rapid City', 'Andover',
       'Kissimmee', 'Shelton', 'Danbury', 'Sanford', 'San Marcos',
       'Greeley', 'Mansfield', 'Elyria', 'Twin Falls', 'Coral Gables',
       'Romeoville', 'Marlborough', 'Laurel', 'Bryan', 'Pine Bluff',
       'Aberdeen', 'Hagerstown', 'East Orange', 'Arlington Heights',
       'Oswego', 'Beaumont', 'Coon Rapids', 'San Clemente',
       'San Luis Obispo', 'Springdale', 'Lodi', 'Mason'], dtype=object)
```

[19]:
```
df.dropna(subset=["City"],inplace=True) #droped nan values in city
```

[20]:
```
df.isnull().sum() #checking null values
```

[20]:
```
Row ID            5
Order ID          9
Order Date        9
Ship Date        11
Ship Mode         8
Customer ID       5
Customer Name    23
Segment          18
Country          21
City              0
State             6
Postal Code       0
Region            0
Product ID        2
Category          2
Sub-Category      4
Product Name      4
Sales             0
Quantity          0
Discount          0
Profit           13
dtype: int64
```

[21]:
```
df.shape #shape of dataframe
```

[21]:
```
(9866, 21)
```

[22]:
```
df["State"].isnull().sum() #checking null values in state
```

[22]:
```
6
```

[23]:
```
df["State"].unique() #finding unique values in state column
```

[23]:
```
array(['Kentucky', 'California', 'Florida', 'North Carolina',
       'Washington', 'Texas', 'Wisconsin', 'Utah', 'Nebraska',
       'Pennsylvania', 'Illinois', 'Minnesota', 'Michigan', 'Delaware',
       'Indiana', 'New York', 'Arizona', 'Virginia', 'Tennessee',
       'Alabama', 'South Carolina', 'Colorado', 'Iowa', 'Ohio',
       'Missouri', 'Oklahoma', 'New Mexico', 'Louisiana', 'Connecticut',
       'New Jersey', 'Oregon', 'Massachusetts', 'Georgia', 'Nevada',
       'Rhode Island', nan, 'Mississippi', 'Arkansas', 'Montana',
       'New Hampshire', 'Maryland', 'District of Columbia', 'Kansas',
       'Vermont', 'Maine', 'South Dakota', 'Idaho', 'North Dakota',
       'Wyoming', 24.849999999999998, 12.624, 89.584, 471.92,
       18.180000000000003, 31.744, 5.904, 621.7600000000001, 59.98, 48.87,
       154.9, 5.92, 30.18, 24.1, 8.78, 376.74, 29.52, 11.96,
       26.400000000000002, 'West Virginia'], dtype=object)
```

[24]:
```
state_city=df[["State","City"]]  #created dataframe for state, city
unique_state_city=state_city.drop_duplicates() #drop dupliates
unique_state_city.head(10)  #hecking respective city name for state to change␣
 ↪errors in state column
```

[24]:

	State	City
0	Kentucky	Henderson
2	California	Los Angeles
3	Florida	Fort Lauderdale
12	North Carolina	Concord
13	Washington	Seattle
14	Texas	Fort Worth
16	Wisconsin	Madison
17	Utah	West Jordan
18	California	San Francisco
21	Nebraska	Fremont

[25]:
```
a=[24.849999999999998, 12.624, 89.584, 471.92,
       18.180000000000003, 31.744, 5.904, 621.7600000000001, 59.98, 48.87,
       154.9, 5.92, 30.18, 24.1, 8.78, 376.74, 29.52, 11.96,
       26.400000000000002]   #values are errors in state column
for i in a:
    print(df[df["State"]==i])
```

```
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6972  6973.0  CA-2017-153822 2017-09-19 2017-09-25  Standard Class

     Customer ID  Customer Name   Segment        Country     City  State  \
6972    AB-10105  Adrian Barton  Consumer  United States  Phoenix  24.85
```

```
      Postal Code Region Product ID    Category Sub-Category  \
6972          5.0      0     7.7035  Technology       Phones

                                    Product Name   Sales  Quantity  Discount  \
6972  Polycom VoiceStation 500 Conference phone  471.92       2.0       0.2

      Profit
6972  29.495
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6973  6974.0  CA-2017-153822 2017-09-19 2017-09-25  Standard Class

     Customer ID  Customer Name   Segment        Country     City   State  \
6973    AB-10105  Adrian Barton  Consumer  United States  Phoenix  12.624

      Postal Code Region Product ID         Category Sub-Category  \
6973          2.0    0.2    -2.5248  Office Supplies      Binders

               Product Name  Sales  Quantity  Discount  Profit
6973  Plastic Binding Combs  18.18       4.0       0.7 -13.938
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6974  6975.0  CA-2017-146185 2017-09-15 2017-09-19  Standard Class

     Customer ID     Customer Name   Segment        Country     City   State  \
6974    CC-12145  Charles Crestani  Consumer  United States  Houston  89.584

      Postal Code Region Product ID         Category Sub-Category  \
6974          2.0    0.2     4.4792  Office Supplies          Art

                     Product Name   Sales  Quantity  Discount  Profit
6974  Prismacolor Color Pencil Set  31.744       2.0       0.2  8.3328
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6975  6976.0  CA-2015-112144 2015-06-28 2015-07-02  Standard Class

     Customer ID Customer Name    Segment        Country     City   State  \
6975    CY-12745  Craig Yedwab  Corporate  United States  Gilbert  471.92

      Postal Code Region Product ID         Category Sub-Category  \
6975          2.0    0.2     29.495  Office Supplies       Labels

     Product Name  Sales  Quantity  Discount  Profit
6975    Avery 501  5.904       2.0       0.2  1.9926
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6976  6977.0  CA-2015-112144 2015-06-28 2015-07-02  Standard Class

     Customer ID Customer Name    Segment        Country     City  State  \
6976    CY-12745  Craig Yedwab  Corporate  United States  Gilbert  18.18

      Postal Code Region Product ID   Category Sub-Category  \
```

```
6976          4.0    0.7    -13.938  Furniture  Furnishings

                        Product Name   Sales  Quantity  Discount  Profit
6976  Electrix Halogen Magnifier Lamp  621.76       4.0       0.2  46.632
      Row ID        Order ID Order Date  Ship Date    Ship Mode Customer ID  \
6977  6978.0  US-2016-119298 2016-11-25 2016-11-28  First Class    EP-13915

     Customer Name   Segment        Country       City   State  Postal Code  \
6977    Emily Phan  Consumer  United States  Jonesboro  31.744          2.0

     Region Product ID    Category Sub-Category  \
6977    0.2     8.3328  Technology       Phones

                                        Product Name  Sales  Quantity  \
6977  OtterBox Defender Series Case - Samsung Galaxy S4  59.98       2.0

      Discount  Profit
6977       0.0  17.994
      Row ID        Order ID Order Date  Ship Date     Ship Mode Customer ID  \
6978  6979.0  CA-2017-155159 2017-11-25 2017-11-29  Second Class    DL-13315

         Customer Name   Segment        Country     City  State  Postal Code  \
6978  Delfina Latchford  Consumer  United States  Atlanta  5.904          2.0

     Region Product ID         Category Sub-Category  \
6978    0.2     1.9926  Office Supplies        Paper

                           Product Name  Sales  Quantity  Discount   Profit
6978  Wirebound Message Book, 4 per Page  48.87       9.0       0.0  23.9463
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6979  6980.0  CA-2017-149076 2017-01-14 2017-01-19  Standard Class

     Customer ID   Customer Name   Segment        Country         City  \
6979    SO-20335  Sean O'Donnell  Consumer  United States  Los Angeles

       State  Postal Code Region Product ID         Category Sub-Category  \
6979  621.76          4.0    0.2     46.632  Office Supplies        Paper

     Product Name  Sales  Quantity  Discount  Profit
6979     Xerox 19  154.9       5.0       0.0  69.705
      Row ID        Order ID Order Date  Ship Date    Ship Mode Customer ID  \
6980  6981.0  CA-2014-146990 2014-11-07 2014-11-08  First Class    BP-11095

     Customer Name    Segment        Country           City  State  \
6980  Bart Pistole  Corporate  United States  New York City  59.98

      Postal Code Region Product ID         Category Sub-Category  \
6980          2.0      0     17.994  Office Supplies    Fasteners
```

```
             Product Name  Sales  Quantity  Discount  Profit
6980  Binder Clips by OIC   5.92       4.0       0.0  2.8416
      Row ID        Order ID Order Date  Ship Date    Ship Mode Customer ID  \
6981  6982.0  CA-2014-146990 2014-11-07 2014-11-08  First Class    BP-11095

     Customer Name    Segment        Country           City  State  \
6981  Bart Pistole  Corporate  United States  New York City  48.87

      Postal Code Region Product ID         Category Sub-Category  \
6981         9.0      0    23.9463  Office Supplies        Paper

                                  Product Name  Sales  Quantity  Discount  \
6981  Riverleaf Stik-Withit Designer Note Cubes  30.18       3.0       0.0

       Profit
6981  13.8828
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6982  6983.0  CA-2016-116526 2016-09-01 2016-09-05  Standard Class

     Customer ID Customer Name   Segment        Country     City  State  \
6982    JA-15970  Joseph Airdo  Consumer  United States  Detroit  154.9

      Postal Code Region Product ID         Category Sub-Category  \
6982         5.0      0     69.705  Office Supplies      Binders

                                        Product Name  Sales  Quantity  \
6982  Wilson Jones Turn Tabs Binder Tool for Ring Bi…   24.1       5.0

      Discount  Profit
6982       0.0  11.086
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6983  6984.0  CA-2016-116526 2016-09-01 2016-09-05  Standard Class

     Customer ID Customer Name   Segment        Country     City State  \
6983    JA-15970  Joseph Airdo  Consumer  United States  Detroit  5.92

      Postal Code Region Product ID    Category Sub-Category  \
6983         4.0      0     2.8416  Technology       Phones

                                        Product Name  Sales  Quantity  \
6983  Belkin Grip Candy Sheer Case / Cover for iPhon…   8.78       1.0

      Discount  Profit
6983       0.0  2.2828
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6984  6985.0  CA-2016-116526 2016-09-01 2016-09-05  Standard Class
```

```
     Customer ID Customer Name   Segment        Country     City  State  \
6984    JA-15970  Joseph Airdo  Consumer  United States  Detroit  30.18

      Postal Code Region Product ID         Category Sub-Category  \
6984          3.0      0    13.8828  Office Supplies   Appliances

                                              Product Name   Sales  Quantity  \
6984  Eureka The Boss Plus 12-Amp Hard Box Upright V…  376.74       4.0

      Discount  Profit
6984       0.1  71.162
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6985  6986.0  CA-2016-116526 2016-09-01 2016-09-05  Standard Class

     Customer ID Customer Name   Segment        Country     City State  \
6985    JA-15970  Joseph Airdo  Consumer  United States  Detroit  24.1

      Postal Code Region Product ID         Category Sub-Category  \
6985          5.0      0     11.086  Office Supplies      Binders

                 Product Name  Sales  Quantity  Discount   Profit
6985  GBC Plastic Binding Combs  29.52       4.0       0.0  14.4648
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6986  6987.0  CA-2016-116526 2016-09-01 2016-09-05  Standard Class

     Customer ID Customer Name   Segment        Country     City State  \
6986    JA-15970  Joseph Airdo  Consumer  United States  Detroit  8.78

      Postal Code Region Product ID         Category Sub-Category  \
6986          1.0      0     2.2828  Office Supplies          Art

     Product Name  Sales  Quantity  Discount  Profit
6986   Newell 315  11.96       2.0       0.0    2.99
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6987  6988.0  CA-2016-116526 2016-09-01 2016-09-05  Standard Class

     Customer ID Customer Name   Segment        Country     City   State  \
6987    JA-15970  Joseph Airdo  Consumer  United States  Detroit  376.74

      Postal Code Region Product ID         Category Sub-Category  \
6987          4.0    0.1     71.162  Office Supplies      Binders

                                       Product Name  Sales  Quantity  \
6987  Wilson Jones 1" Hanging DublLock Ring Binders   26.4       5.0

      Discount  Profit
6987       0.0  12.672
      Row ID        Order ID Order Date  Ship Date     Ship Mode Customer ID  \
```

```
6988  6989.0  CA-2017-158561 2017-11-11 2017-11-16  Second Class    BB-11545

      Customer Name    Segment        Country             City  State  \
6988  Brenda Bowman  Corporate  United States  Fort Lauderdale  29.52

      Postal Code Region Product ID         Category Sub-Category  \
6988          4.0      0    14.4648  Office Supplies   Appliances

                             Product Name    Sales  Quantity  Discount  \
6988  Hoover Upright Vacuum With Dirt Cup  1158.12       5.0       0.2

        Profit
6988  130.2885
      Row ID        Order ID Order Date  Ship Date    Ship Mode Customer ID  \
6989  6990.0  CA-2017-165099 2017-12-11 2017-12-13  First Class    DK-13375

     Customer Name   Segment        Country     City  State  Postal Code  \
6989   Dennis Kane  Consumer  United States  Abilene  11.96          2.0

     Region Product ID         Category Sub-Category  \
6989      0       2.99  Office Supplies   Appliances

                                     Product Name  Sales  Quantity  Discount  \
6989  Hoover Commercial Lightweight Upright Vacuum  1.392       2.0       0.8

      Profit
6989 -3.7584
      Row ID        Order ID Order Date  Ship Date     Ship Mode Customer ID  \
6990  6991.0  CA-2015-109386 2015-11-08 2015-11-13  Second Class    RH-19600

     Customer Name   Segment        Country     City State  Postal Code  \
6990  Rob Haberlin  Consumer  United States  Hampton  26.4          5.0

     Region Product ID         Category Sub-Category  \
6990      0     12.672  Office Supplies   Appliances

                                          Product Name  Sales  Quantity  \
6990  Holmes Replacement Filter for HEPA Air Cleaner…  44.43       3.0

      Discount   Profit
6990       0.0  18.6606
```

[26]:
```
df[df["State"].isnull()] #checking nul values in state
```

[26]:
```
      Row ID        Order ID Order Date  Ship Date       Ship Mode Customer ID  \
 416   417.0  CA-2017-122105 2017-06-24 2017-06-28  Standard Class    CJ-12010
 423   424.0  CA-2017-125388 2017-10-19 2017-10-23             NaN    MP-17965
```

```
428   429.0  CA-2017-152275 2017-10-01 2017-10-08  Standard Class    KH-16630
430   431.0  US-2016-123750 2016-04-15 2016-04-21  Standard Class    RB-19795
486   487.0  CA-2017-140963 2017-06-10 2017-06-13     First Class    MT-18070
659   660.0  CA-2015-146563 2015-08-24 2015-08-28  Standard Class    CB-12025

       Customer Name      Segment        Country              City State  \
416  Caroline Jumper     Consumer  United States  Huntington Beach   NaN
423    Michael Paige    Corporate  United States          Lawrence   NaN
428       Ken Heidel    Corporate  United States       San Antonio   NaN
430       Ross Baird  Home Office  United States          Gastonia   NaN
486    Michelle Tran  Home Office  United States       Los Angeles   NaN
659              NaN     Consumer  United States         Arlington   NaN

     Postal Code   Region       Product ID         Category Sub-Category  \
416      92646.0     West  OFF-AR-10004344              NaN          Art
423       1841.0     East  OFF-ST-10000918  Office Supplies      Storage
428      78207.0  Central  OFF-AR-10000369              NaN          Art
430      28052.0    South  TEC-AC-10004659       Technology          NaN
486      90045.0     West  TEC-PH-10001924       Technology       Phones
659      76017.0  Central  OFF-ST-10001490  Office Supplies          NaN

                                        Product Name    Sales  Quantity  \
416              Bulldog Vacuum Base Pencil Sharpener   95.920       8.0
423                                     Crate-A-Files   32.700       3.0
428                      Design Ebony Sketching Pencil    6.672       6.0
430  Imation Secure+ Hardware Encrypted USB 2.0 Fla…  408.744       7.0
486                                               NaN  279.960       5.0
659                    Hot File 7-Pocket, Floor Stand  999.432       7.0

     Discount    Profit
416       0.0   25.8984
423       0.0    8.5020
428       0.2    0.5004
430       0.2   76.6395
486       0.2   17.4975
659       0.2  124.9290
```

[27]:
```
df[df["City"]=="Huntington Beach"].head(2) #checkin city name to replace nan␣
↪with respective state name
```

[27]:
```
       Row ID        Order ID Order Date  Ship Date       Ship Mode  \
416     417.0  CA-2017-122105 2017-06-24 2017-06-28  Standard Class
1890   1891.0  CA-2014-157623 2014-03-14 2014-03-18  Standard Class

     Customer ID    Customer Name    Segment        Country             City  \
416     CJ-12010  Caroline Jumper   Consumer  United States  Huntington Beach
1890    DK-13225        Dean Katz  Corporate  United States  Huntington Beach
```

```
           State  Postal Code Region       Product ID         Category  \
416          NaN      92646.0   West  OFF-AR-10004344              NaN
1890  California      92646.0   West  OFF-PA-10001204  Office Supplies

     Sub-Category                         Product Name  Sales  Quantity  \
416           Art  Bulldog Vacuum Base Pencil Sharpener  95.92       8.0
1890        Paper                           Xerox 1972  10.56       2.0

      Discount   Profit
416        0.0  25.8984
1890       0.0   4.7520
```

[28]:
```
df[df["State"]=="Maryland"].head(3)
```

[28]:
```
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
887    888.0  CA-2017-150707 2017-10-14 2017-10-19  Standard Class
1093  1094.0  CA-2015-165085 2015-12-27 2015-12-31  Standard Class
1094  1095.0  CA-2015-165085 2015-12-27 2015-12-31  Standard Class

     Customer ID Customer Name      Segment        Country      City  \
887     EL-13735     Ed Ludwig  Home Office  United States  Columbia
1093    BT-11485   Brad Thomas  Home Office  United States   Clinton
1094    BT-11485   Brad Thomas  Home Office  United States   Clinton

         State  Postal Code Region       Product ID         Category  \
887   Maryland      21044.0   East  OFF-BI-10001078  Office Supplies
1093  Maryland      20735.0   East  OFF-PA-10000605  Office Supplies
1094  Maryland      20735.0   East  OFF-AP-10002518  Office Supplies

     Sub-Category                                          Product Name   Sales  \
887       Binders  Acco PRESSTEX Data Binder with Storage Hooks, …   37.66
1093        Paper                                          Xerox 1950   28.90
1094   Appliances       Kensington 7 Outlet MasterPiece Power Center  355.96

      Quantity  Discount    Profit
887        7.0       0.0   18.4534
1093       5.0       0.0   14.1610
1094       2.0       0.0  103.2284
```

[29]:
```
city_to_state = {
    'Phoenix': 'Arizona',
    'Houston': 'Texas',
    'Gilbert': 'Arizona',
    'Jonesboro': 'Arkansas',
    'Atlanta': 'Georgia',
    'Los Angeles': 'California',
```

```
    'New York City': 'New York',
    'Detroit': 'Michigan',
    'Fort Lauderdale': 'Florida',
    'Hampton': 'Virginia',
    'Arlington':'Virginia',
    'Gastonia':'North Carolina',
    'San Antonio':'Texas',
    'Lawrence':'Massachusetts',
    'Huntington Beach':'California'
}

# Update 'State' column for cities in city_to_state dictionary
df['State'] = df['City'].map(city_to_state).fillna(df['State'])

print(df)
```

```
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
0        1.0  CA-2016-152156 2016-11-08 2016-11-11    Second Class
1        2.0  CA-2016-152156 2016-11-08 2016-11-11    Second Class
2        3.0  CA-2016-138688 2016-06-12 2016-06-16    Second Class
3        4.0  US-2015-108966 2015-10-11 2015-10-18  Standard Class
4        5.0  US-2015-108966 2015-10-11 2015-10-18  Standard Class
…        …              …          …          …              …
9989     NaN            NaN        NaT        NaT  Standard Class
9990     NaN            NaN        NaT        NaT    Second Class
9991     NaN            NaN        NaT        NaT  Standard Class
9992     NaN            NaN        NaT        NaT  Standard Class
9993     NaN            NaN        NaT        NaT    Second Class

     Customer ID    Customer Name      Segment        Country  \
0       CG-12520      Claire Gute     Consumer  United States
1       CG-12520      Claire Gute     Consumer  United States
2       DV-13045  Darrin Van Huff    Corporate  United States
3       SO-20335   Sean O'Donnell     Consumer  United States
4       SO-20335   Sean O'Donnell     Consumer  United States
…          …              …          …              …
9989    SR-20425   Sharelle Roach  Home Office  United States
9990    AG-10330     Alex Grayson     Consumer  United States
9991    BP-11095              NaN          NaN            NaN
9992    JW-16075              NaN          NaN            NaN
9993    LH-16900              NaN          NaN            NaN

                City           State  Postal Code   Region       Product ID  \
0          Henderson        Kentucky      42420.0    South  FUR-BO-10001798
1          Henderson        Kentucky      42420.0    South  FUR-CH-10000454
2        Los Angeles      California      90036.0     West  OFF-LA-10000240
3     Fort Lauderdale         Florida      33311.0    South  FUR-TA-10000577
```

```
4     Fort Lauderdale          Florida      33311.0    South  OFF-ST-10000760
…                 …             …          …       …              …
9989       Tuscaloosa          Alabama      35401.0    South  FUR-CH-10002647
9990             Mesa          Arizona      85204.0     West  FUR-TA-10003008
9991     Jacksonville  North Carolina      28540.0    South  OFF-PA-10004071
9992          Chicago         Illinois      60610.0  Central  OFF-AP-10004980
9993         Columbus          Georgia      31907.0    South  FUR-FU-10000747

             Category Sub-Category  \
0           Furniture    Bookcases
1           Furniture       Chairs
2     Office Supplies       Labels
3           Furniture       Tables
4     Office Supplies      Storage
…                 …          …
9989        Furniture       Chairs
9990        Furniture       Tables
9991  Office Supplies        Paper
9992  Office Supplies   Appliances
9993        Furniture  Furnishings

                                          Product Name     Sales  Quantity  \
0                    Bush Somerset Collection Bookcase  261.9600       2.0
1     Hon Deluxe Fabric Upholstered Stacking Chairs,…  731.9400       3.0
2     Self-Adhesive Address Labels for Typewriters b…   14.6200       2.0
3         Bretford CR4500 Series Slim Rectangular Table  957.5775       5.0
4                       Eldon Fold 'N Roll Cart System   22.3680       2.0
…                                                  …       …       …
9989         Situations Contoured Folding Chairs, 4/Set  141.9600       2.0
9990  Lesro Round Back Collection Coffee Table, End …  182.5500       2.0
9991  Eaton Premium Continuous-Feed Paper, 25% Cotto…   88.7680       2.0
9992  3M Replacement Filter for Office Air Cleaner f…   53.0880       7.0
9993  Tenex B1-RE Series Chair Mats for Low Pile Car…  275.8800       6.0

      Discount    Profit
0         0.00   41.9136
1         0.00  219.5820
2         0.00    6.8714
3         0.45 -383.0310
4         0.20    2.5164
…          …       …
9989      0.00   35.4900
9990      0.50 -135.0870
9991      0.20   31.0688
9992      0.80 -108.8304
9993      0.00   46.8996

[9866 rows x 21 columns]
```

[30]:
```
#checking changes
df[df["City"]=="Huntington Beach"].head(2)
```

[30]:
```
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
416    417.0  CA-2017-122105 2017-06-24 2017-06-28  Standard Class
1890  1891.0  CA-2014-157623 2014-03-14 2014-03-18  Standard Class

     Customer ID    Customer Name    Segment        Country              City  \
416     CJ-12010  Caroline Jumper   Consumer  United States  Huntington Beach
1890    DK-13225        Dean Katz  Corporate  United States  Huntington Beach

           State  Postal Code Region       Product ID         Category  \
416   California      92646.0   West  OFF-AR-10004344              NaN
1890  California      92646.0   West  OFF-PA-10001204  Office Supplies

     Sub-Category                          Product Name  Sales  Quantity  \
416           Art  Bulldog Vacuum Base Pencil Sharpener  95.92       8.0
1890        Paper                            Xerox 1972  10.56       2.0

      Discount   Profit
416        0.0  25.8984
1890       0.0   4.7520
```

[31]:
```
df["State"].isnull().sum() #cheking number of nulls in state
```

[31]:
```
0
```

[32]:
```
#now null also chnaged
df[df["Row ID"]==107]
```

[32]:
```
     Row ID        Order ID Order Date  Ship Date       Ship Mode Customer ID  \
106   107.0  CA-2017-119004 2017-11-23 2017-11-28  Standard Class    JM-15250

    Customer Name    Segment        Country       City           State  \
106  Janet Martin  Consumer  United States  Charlotte  North Carolina

     Postal Code Region       Product ID    Category Sub-Category  \
106      28205.0  South  TEC-AC-10003499  Technology  Accessories

                                          Product Name   Sales  Quantity  \
106  Memorex Mini Travel Drive 8 GB USB 2.0 Flash D…  74.112       8.0

     Discount   Profit
106       0.2  17.6016
```

[33]:
```
#drop 11.96 due to no name of state
df.drop(df[df["State"]==11.96].index,axis=0,inplace=True)
```

[34]:
```
df["State"].unique() #now rechecking uniques values in state
```

[34]:
```
array(['Kentucky', 'California', 'Florida', 'North Carolina',
       'Washington', 'Texas', 'Wisconsin', 'Utah', 'Nebraska',
       'Pennsylvania', 'Illinois', 'Minnesota', 'Michigan', 'Delaware',
       'Indiana', 'New York', 'Arizona', 'Virginia', 'Tennessee',
       'Alabama', 'South Carolina', 'Colorado', 'Iowa', 'Ohio',
       'Missouri', 'Oklahoma', 'New Mexico', 'Louisiana', 'Connecticut',
       'New Jersey', 'Oregon', 'Massachusetts', 'Georgia', 'Nevada',
       'Rhode Island', 'Mississippi', 'Arkansas', 'Montana',
       'New Hampshire', 'Maryland', 'District of Columbia', 'Kansas',
       'Vermont', 'Maine', 'South Dakota', 'Idaho', 'North Dakota',
       'Wyoming', 'West Virginia'], dtype=object)
```

[35]:
```
df["Region"].unique() #hecking unique values in region(found some errors)
```

[35]:
```
array(['South', 'West', 'Central', 'East', 0, 0.2, 0.7, 0.1], dtype=object)
```

[36]:
```
df["Region"].isnull().sum() #checking nan values in region column
```

[36]:
```
0
```

[37]:
```
a = [0, 0.2, 0.7, 0.1]
cities_by_region = {}

for i in a:
    filtered_df = df[df["Region"] == i]
    cities_by_region[i] = filtered_df["City"].tolist()

print(cities_by_region)
```

```
{0: ['Phoenix', 'New York City', 'New York City', 'Detroit', 'Detroit',
'Detroit', 'Detroit', 'Detroit', 'Fort Lauderdale', 'Hampton'], 0.2: ['Phoenix',
'Houston', 'Gilbert', 'Jonesboro', 'Atlanta', 'Los Angeles'], 0.7: ['Gilbert'],
0.1: ['Detroit']}
```

[38]:
```
df[df["City"]=="Huntington Beach"].head(2) #checking region for city
```

[38]:
```
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
416    417.0  CA-2017-122105 2017-06-24 2017-06-28  Standard Class
1890  1891.0  CA-2014-157623 2014-03-14 2014-03-18  Standard Class

     Customer ID    Customer Name    Segment        Country             City  \
416     CJ-12010  Caroline Jumper   Consumer  United States  Huntington Beach
1890    DK-13225        Dean Katz  Corporate  United States  Huntington Beach

           State  Postal Code Region       Product ID         Category  \
```

```
416   California      92646.0   West  OFF-AR-10004344              NaN
1890  California      92646.0   West  OFF-PA-10001204  Office Supplies

     Sub-Category                          Product Name  Sales  Quantity  \
416           Art  Bulldog Vacuum Base Pencil Sharpener  95.92       8.0
1890        Paper                           Xerox 1972  10.56       2.0

      Discount   Profit
416        0.0  25.8984
1890       0.0   4.7520
```

[39]:
```
city_to_region = {
    'Phoenix': 'West',
    'Houston': 'Central',
    'Gilbert': 'West',
    'Jonesboro': 'South',
    'Atlanta': 'South',
    'Los Angeles': 'West',
    'New York City': 'East',
    'Detroit': 'Central',
    'Fort Lauderdale': 'South',
    'Hampton': 'South'
}

# Update 'Region' column permanently
df['Region'] = df['City'].map(city_to_region).fillna(df['Region'])
```

[40]:
```
df["Region"].unique() #cheking unique values in region
```

[40]:
```
array(['South', 'West', 'Central', 'East'], dtype=object)
```

[41]:
```
df.shape #cheking shape of dataframe
```

[41]:
```
(9865, 21)
```

[42]:
```
df.dropna(subset=["Order Date","Ship Date"],inplace=True) #droping null values
 ↪in order date,ship date
```

[43]:
```
df.isnull().sum() #checking null values in columns
```

[43]:
```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        4
Customer ID      1
Customer Name   19
```

```
Segment          15
Country          18
City              0
State             0
Postal Code       0
Region            0
Product ID        2
Category          2
Sub-Category      4
Product Name      4
Sales             0
Quantity          0
Discount          0
Profit           13
dtype: int64
```

[44]:
```
#data formating(creating new columns from existing data)
df["Order Date"] = pd.to_datetime(df["Order Date"], format="%d/%m/%Y")
df["Ship Date"] = pd.to_datetime(df["Ship Date"], format="%d/%m/%Y")

df["order_year"]=df["Order Date"].dt.year
df["order_month"]=df["Order Date"].dt.month
df["order_date"]=df["Order Date"].dt.day

df["Ship_year"]=df["Ship Date"].dt.year
df["Ship_month"]=df["Ship Date"].dt.month
df["Ship_date"]=df["Ship Date"].dt.day
```

[45]:
```
df["Postal Code"].isnull().sum() #sum of null values in Postal code
```

[45]:
```
0
```

[46]:
```
s=df["Postal Code"].astype(int) #unique postal code
s.unique()
```

[46]:
```
array([42420, 90036, 33311, 90032, 28027, 98103, 76106, 53711, 84084,
       94109, 68025, 19140, 84057, 90049, 77095, 75080, 77041, 60540,
       32935, 55122, 48185, 19901, 47150, 10024, 12180, 90004, 60610,
       85234, 22153, 10009, 49201, 38109, 77070, 35601, 94122, 27707,
       60623, 29203, 55901, 80013, 28205, 60462, 10035, 50322, 43229,
       37620, 19805, 61701, 85023, 95661, 64055, 91104, 43055, 53132,
       85254, 95123, 98105, 98115, 73034, 90045, 19134, 88220, 78207,
       77036, 62521, 71203,  6824, 75051, 80219, 75220, 37064, 90604,
       48601, 44256, 48227, 38401, 33614, 95051, 55044, 92037, 77506,
       94513, 27514,  7960, 45231, 94110, 90301, 97206, 33319, 80906,
        7109, 48180,  8701, 22204, 80004,  7601, 33710, 19143, 90805,
       92345, 37130, 78745,  1852, 31907,  6040, 78550, 85705, 62301,
```

 2038, 33024, 98198, 61604, 89115, 2886, 33180, 28403, 92646,
40475, 80027, 1841, 39212, 48187, 10801, 28052, 32216, 47201,
13021, 44312, 73071, 94521, 60068, 79109, 11757, 90008, 92024,
77340, 14609, 72701, 92627, 80134, 30318, 64118, 59405, 48234,
36116, 85204, 60653, 54302, 45503, 92804, 98270, 97301, 78041,
19120, 75217, 43123, 10011, 48126, 31088, 94591, 55407, 92691,
48307, 7060, 85635, 98661, 60505, 76017, 40214, 75081, 44105,
75701, 27217, 22980, 19013, 27511, 32137, 10550, 48205, 33012,
11572, 92105, 60201, 48183, 55016, 71111, 50315, 93534, 23223,
28806, 92530, 68104, 98026, 92704, 53209, 41042, 44052, 7036,
93905, 8901, 17602, 3301, 21044, 75043, 6360, 22304, 43615,
87401, 92503, 90503, 78664, 92054, 33433, 23464, 92563, 28540,
52601, 98502, 20016, 65109, 63376, 61107, 33142, 78521, 10701,
94601, 28110, 20735, 30076, 72401, 47374, 94509, 33030, 46350,
48911, 44221, 89502, 22801, 92025, 48073, 20852, 33065, 14215,
33437, 39503, 93727, 27834, 11561, 35630, 31204, 52402, 2908,
81001, 94533, 55106, 32725, 42071, 6457, 11520, 90660, 84604,
84062, 30080, 24153, 44134, 36608, 2740, 75061, 8360, 85301,
14304, 27360, 92683, 38301, 75019, 91767, 89031, 18103, 19711,
85281, 92677, 8302, 2149, 13601, 54915, 98006, 75002, 79907,
76051, 75007, 37167, 98031, 70506, 97224, 60076, 75023, 23434,
46203, 7002, 43017, 28314, 27405, 21215, 53142, 66062, 98002,
74133, 97756, 27604, 74403, 6450, 42104, 46614, 6010, 89015,
99207, 76248, 45014, 32127, 97504, 22901, 59801, 33178, 29501,
97477, 32712, 19601, 80020, 65807, 7501, 73120, 23320, 79424,
65203, 37604, 36830, 92404, 1453, 59715, 85345, 44107, 8861,
91761, 91730, 56560, 75150, 92374, 95207, 32174, 94086, 3820,
17403, 77840, 63116, 2169, 95336, 44240, 76903, 84106, 35810,
37918, 72209, 48146, 43302, 80122, 5408, 4401, 38671, 47362,
48640, 57103, 80525, 47905, 37042, 95823, 91360, 2148, 1040,
87105, 89431, 92236, 60126, 7055, 29406, 23602, 14701, 46544,
43402, 7090, 92253, 32303, 37211, 98226, 60098, 76117, 60090,
29483, 71901, 80112, 43130, 88001, 35244, 75034, 95687, 84107,
53186, 93309, 33068, 45373, 78415, 90278, 32839, 7050, 70601,
60035, 11550, 46060, 55124, 29464, 48310, 54703, 78577, 59102,
97030, 37421, 83642, 92307, 60440, 33801, 55369, 95695, 77489,
77581, 94403, 49505, 93277, 66212, 92592, 92399, 2151, 77301,
60477, 52001, 48127, 87505, 28601, 60188, 56301, 33161, 46226,
33317, 34952, 29730, 79762, 53214, 91911, 66502, 16602, 80229,
61821, 47401, 71854, 78539, 77520, 46142, 90712, 2895, 54880,
76021, 98042, 74012, 33023, 33021, 77536, 67212, 78501, 52240,
83704, 2920, 61032, 77642, 95610, 75056, 98052, 32114, 86442,
46368, 58103, 46514, 91776, 45011, 33063, 30328, 44060, 73505,
23666, 13440, 54601, 83501, 39401, 94526, 48858, 84321, 6708,
30605, 4240, 61832, 85323, 30062, 85364, 54401, 99301, 60302,
32503, 77573, 20877, 84043, 35401, 92553, 40324, 80538, 85224,
59601, 63122, 76706, 48066, 60423, 18018, 55113, 68801, 55125,

```
       48237, 72756, 88101, 33458, 93101, 75104, 68701, 84020, 48104,
       91941, 83201, 49423,  6460, 60089, 92630, 96003, 95928, 13501,
       72032, 82001, 42301, 83605, 70065,  3060, 38134, 94061, 37087,
       93454, 60016, 98632, 37075, 50701,  2138, 60067,  1915, 97405,
       93030, 98059, 60025, 33445, 80022, 77590, 27893, 87124, 27534,
       98208, 90640, 92020, 33407,     5,     2,     4,     9,     3,
           1, 61761, 60174, 93010, 97123, 91505, 95351, 67846,  8401,
       80501, 95616, 26003, 95037,  7011, 53081, 30344, 57701,  1810,
       34741,  6484,  6810, 52302, 32771, 78666, 80634, 76063, 44035,
       83301, 63301, 33134, 60441,  1752, 20707, 77803, 71603, 57401,
       21740,  7017, 60004, 60543, 77705, 55433, 92672, 94568, 93405,
       72762, 95240, 77571, 45040, 30188])
```

[47]:
```
a=[5,2,4,9,3,1] #errors in postal code
c=df[df["Postal Code"].isin(a)]
c
```

[47]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	\
6972	6973.0	CA-2017-153822	2017-09-19	2017-09-25	Standard Class	
6973	6974.0	CA-2017-153822	2017-09-19	2017-09-25	Standard Class	
6974	6975.0	CA-2017-146185	2017-09-15	2017-09-19	Standard Class	
6975	6976.0	CA-2015-112144	2015-06-28	2015-07-02	Standard Class	
6976	6977.0	CA-2015-112144	2015-06-28	2015-07-02	Standard Class	
6977	6978.0	US-2016-119298	2016-11-25	2016-11-28	First Class	
6978	6979.0	CA-2017-155159	2017-11-25	2017-11-29	Second Class	
6979	6980.0	CA-2017-149076	2017-01-14	2017-01-19	Standard Class	
6980	6981.0	CA-2014-146990	2014-11-07	2014-11-08	First Class	
6981	6982.0	CA-2014-146990	2014-11-07	2014-11-08	First Class	
6982	6983.0	CA-2016-116526	2016-09-01	2016-09-05	Standard Class	
6983	6984.0	CA-2016-116526	2016-09-01	2016-09-05	Standard Class	
6984	6985.0	CA-2016-116526	2016-09-01	2016-09-05	Standard Class	
6985	6986.0	CA-2016-116526	2016-09-01	2016-09-05	Standard Class	
6986	6987.0	CA-2016-116526	2016-09-01	2016-09-05	Standard Class	
6987	6988.0	CA-2016-116526	2016-09-01	2016-09-05	Standard Class	
6988	6989.0	CA-2017-158561	2017-11-11	2017-11-16	Second Class	
6990	6991.0	CA-2015-109386	2015-11-08	2015-11-13	Second Class	

	Customer ID	Customer Name	Segment	Country	\
6972	AB-10105	Adrian Barton	Consumer	United States	
6973	AB-10105	Adrian Barton	Consumer	United States	
6974	CC-12145	Charles Crestani	Consumer	United States	
6975	CY-12745	Craig Yedwab	Corporate	United States	
6976	CY-12745	Craig Yedwab	Corporate	United States	
6977	EP-13915	Emily Phan	Consumer	United States	
6978	DL-13315	Delfina Latchford	Consumer	United States	
6979	SO-20335	Sean O'Donnell	Consumer	United States	
6980	BP-11095	Bart Pistole	Corporate	United States	

```
6981    BP-11095       Bart Pistole  Corporate  United States
6982    JA-15970       Joseph Airdo   Consumer  United States
6983    JA-15970       Joseph Airdo   Consumer  United States
6984    JA-15970       Joseph Airdo   Consumer  United States
6985    JA-15970       Joseph Airdo   Consumer  United States
6986    JA-15970       Joseph Airdo   Consumer  United States
6987    JA-15970       Joseph Airdo   Consumer  United States
6988    BB-11545      Brenda Bowman  Corporate  United States
6990    RH-19600       Rob Haberlin   Consumer  United States

                City       State  …  \
6972         Phoenix     Arizona  …
6973         Phoenix     Arizona  …
6974         Houston       Texas  …
6975         Gilbert     Arizona  …
6976         Gilbert     Arizona  …
6977       Jonesboro    Arkansas  …
6978         Atlanta     Georgia  …
6979     Los Angeles  California  …
6980   New York City    New York  …
6981   New York City    New York  …
6982         Detroit    Michigan  …
6983         Detroit    Michigan  …
6984         Detroit    Michigan  …
6985         Detroit    Michigan  …
6986         Detroit    Michigan  …
6987         Detroit    Michigan  …
6988  Fort Lauderdale     Florida  …
6990         Hampton    Virginia  …

                                          Product Name     Sales Quantity  \
6972         Polycom VoiceStation 500 Conference phone   471.920      2.0
6973                           Plastic Binding Combs    18.180      4.0
6974                    Prismacolor Color Pencil Set    31.744      2.0
6975                                       Avery 501     5.904      2.0
6976                 Electrix Halogen Magnifier Lamp   621.760      4.0
6977  OtterBox Defender Series Case - Samsung Galaxy S4    59.980      2.0
6978               Wirebound Message Book, 4 per Page    48.870      9.0
6979                                        Xerox 19   154.900      5.0
6980                              Binder Clips by OIC     5.920      4.0
6981         Riverleaf Stik-Withit Designer Note Cubes    30.180      3.0
6982  Wilson Jones Turn Tabs Binder Tool for Ring Bi…    24.100      5.0
6983  Belkin Grip Candy Sheer Case / Cover for iPhon…     8.780      1.0
6984  Eureka The Boss Plus 12-Amp Hard Box Upright V…   376.740      4.0
6985                         GBC Plastic Binding Combs    29.520      4.0
6986                                       Newell 315    11.960      2.0
6987     Wilson Jones 1" Hanging DublLock Ring Binders    26.400      5.0
```

```
6988                Hoover Upright Vacuum With Dirt Cup  1158.120      5.0
6990  Holmes Replacement Filter for HEPA Air Cleaner…    44.430      3.0

     Discount    Profit order_year  order_month  order_date  Ship_year  \
6972      0.2   29.4950       2017            9          19       2017
6973      0.7  -13.9380       2017            9          19       2017
6974      0.2    8.3328       2017            9          15       2017
6975      0.2    1.9926       2015            6          28       2015
6976      0.2   46.6320       2015            6          28       2015
6977      0.0   17.9940       2016           11          25       2016
6978      0.0   23.9463       2017           11          25       2017
6979      0.0   69.7050       2017            1          14       2017
6980      0.0    2.8416       2014           11           7       2014
6981      0.0   13.8828       2014           11           7       2014
6982      0.0   11.0860       2016            9           1       2016
6983      0.0    2.2828       2016            9           1       2016
6984      0.1   71.1620       2016            9           1       2016
6985      0.0   14.4648       2016            9           1       2016
6986      0.0    2.9900       2016            9           1       2016
6987      0.0   12.6720       2016            9           1       2016
6988      0.2  130.2885       2017           11          11       2017
6990      0.0   18.6606       2015           11           8       2015

      Ship_month  Ship_date
6972           9         25
6973           9         25
6974           9         19
6975           7          2
6976           7          2
6977          11         28
6978          11         29
6979           1         19
6980          11          8
6981          11          8
6982           9          5
6983           9          5
6984           9          5
6985           9          5
6986           9          5
6987           9          5
6988          11         16
6990          11         13

[18 rows x 27 columns]
```

[48]:
```
#finding most frequent postal code for city
```

```
cities = ["Phoenix", "Houston", "Gilbert", "Jonesboro", "Atlanta", "Los␣
 ↪Angeles", "New York City", "Detroit", "Fort Lauderdale", "Abilene",␣
 ↪"Hampton"]

for city in cities:
    filtered_df = df[df["City"] == city]
    if filtered_df.empty:
        print(f"No rows found for city: {city}")
    else:
        postal_code_mode = filtered_df["Postal Code"].mode()[0]
        print(f"Most frequent postal code for city {city}: {postal_code_mode}")
```

```
Most frequent postal code for city Phoenix: 85023.0
Most frequent postal code for city Houston: 77041.0
Most frequent postal code for city Gilbert: 85234.0
Most frequent postal code for city Jonesboro: 72401.0
Most frequent postal code for city Atlanta: 30318.0
Most frequent postal code for city Los Angeles: 90049.0
Most frequent postal code for city New York City: 10035.0
Most frequent postal code for city Detroit: 48227.0
Most frequent postal code for city Fort Lauderdale: 33311.0
No rows found for city: Abilene
Most frequent postal code for city Hampton: 23666.0
```

[49]:
```
city_to_region = {
    'Phoenix': 85023.0,
    'Houston': 77041.0,
    'Gilbert': 85234.0,
    'Jonesboro': 72401.0,
    'Atlanta': 30318.0,
    'Los Angeles': 90049.0,
    'New York City': 10035.0,
    'Detroit': 48227.0,
    'Fort Lauderdale': 33311.0,
    'Abilene': 77041.0,
    'Hampton': 23666.0
}

# Update 'Region' column permanently
c['Postal Code'] = c['City'].map(city_to_region).fillna(c['Postal Code'])
```

```
C:\Users\ganesh\AppData\Local\Temp\ipykernel_54552\1011837133.py:16:
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
  c['Postal Code'] = c['City'].map(city_to_region).fillna(c['Postal Code'])
```

[50]:
```
c
```

[50]:
```
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
6972  6973.0  CA-2017-153822 2017-09-19 2017-09-25  Standard Class
6973  6974.0  CA-2017-153822 2017-09-19 2017-09-25  Standard Class
6974  6975.0  CA-2017-146185 2017-09-15 2017-09-19  Standard Class
6975  6976.0  CA-2015-112144 2015-06-28 2015-07-02  Standard Class
6976  6977.0  CA-2015-112144 2015-06-28 2015-07-02  Standard Class
6977  6978.0  US-2016-119298 2016-11-25 2016-11-28     First Class
6978  6979.0  CA-2017-155159 2017-11-25 2017-11-29    Second Class
6979  6980.0  CA-2017-149076 2017-01-14 2017-01-19  Standard Class
6980  6981.0  CA-2014-146990 2014-11-07 2014-11-08     First Class
6981  6982.0  CA-2014-146990 2014-11-07 2014-11-08     First Class
6982  6983.0  CA-2016-116526 2016-09-01 2016-09-05  Standard Class
6983  6984.0  CA-2016-116526 2016-09-01 2016-09-05  Standard Class
6984  6985.0  CA-2016-116526 2016-09-01 2016-09-05  Standard Class
6985  6986.0  CA-2016-116526 2016-09-01 2016-09-05  Standard Class
6986  6987.0  CA-2016-116526 2016-09-01 2016-09-05  Standard Class
6987  6988.0  CA-2016-116526 2016-09-01 2016-09-05  Standard Class
6988  6989.0  CA-2017-158561 2017-11-11 2017-11-16    Second Class
6990  6991.0  CA-2015-109386 2015-11-08 2015-11-13    Second Class

     Customer ID      Customer Name    Segment        Country  \
6972    AB-10105      Adrian Barton   Consumer  United States
6973    AB-10105      Adrian Barton   Consumer  United States
6974    CC-12145   Charles Crestani   Consumer  United States
6975    CY-12745       Craig Yedwab  Corporate  United States
6976    CY-12745       Craig Yedwab  Corporate  United States
6977    EP-13915         Emily Phan   Consumer  United States
6978    DL-13315  Delfina Latchford   Consumer  United States
6979    SO-20335     Sean O'Donnell   Consumer  United States
6980    BP-11095       Bart Pistole  Corporate  United States
6981    BP-11095       Bart Pistole  Corporate  United States
6982    JA-15970       Joseph Airdo   Consumer  United States
6983    JA-15970       Joseph Airdo   Consumer  United States
6984    JA-15970       Joseph Airdo   Consumer  United States
6985    JA-15970       Joseph Airdo   Consumer  United States
6986    JA-15970       Joseph Airdo   Consumer  United States
6987    JA-15970       Joseph Airdo   Consumer  United States
6988    BB-11545      Brenda Bowman  Corporate  United States
6990    RH-19600       Rob Haberlin   Consumer  United States

                City       State  …  \
6972         Phoenix     Arizona  …
```

```
6973          Phoenix     Arizona  …
6974          Houston       Texas  …
6975          Gilbert     Arizona  …
6976          Gilbert     Arizona  …
6977        Jonesboro    Arkansas  …
6978          Atlanta     Georgia  …
6979      Los Angeles  California  …
6980    New York City    New York  …
6981    New York City    New York  …
6982          Detroit    Michigan  …
6983          Detroit    Michigan  …
6984          Detroit    Michigan  …
6985          Detroit    Michigan  …
6986          Detroit    Michigan  …
6987          Detroit    Michigan  …
6988  Fort Lauderdale     Florida  …
6990          Hampton    Virginia  …

                                                Product Name     Sales Quantity  \
6972          Polycom VoiceStation 500 Conference phone   471.920      2.0
6973                              Plastic Binding Combs    18.180      4.0
6974                      Prismacolor Color Pencil Set    31.744      2.0
6975                                        Avery 501     5.904      2.0
6976                    Electrix Halogen Magnifier Lamp   621.760      4.0
6977  OtterBox Defender Series Case - Samsung Galaxy S4    59.980      2.0
6978                Wirebound Message Book, 4 per Page    48.870      9.0
6979                                        Xerox 19   154.900      5.0
6980                               Binder Clips by OIC     5.920      4.0
6981          Riverleaf Stik-Withit Designer Note Cubes    30.180      3.0
6982  Wilson Jones Turn Tabs Binder Tool for Ring Bi…    24.100      5.0
6983  Belkin Grip Candy Sheer Case / Cover for iPhon…     8.780      1.0
6984  Eureka The Boss Plus 12-Amp Hard Box Upright V…   376.740      4.0
6985                          GBC Plastic Binding Combs    29.520      4.0
6986                                       Newell 315    11.960      2.0
6987      Wilson Jones 1" Hanging DublLock Ring Binders    26.400      5.0
6988                Hoover Upright Vacuum With Dirt Cup  1158.120      5.0
6990  Holmes Replacement Filter for HEPA Air Cleaner…    44.430      3.0

     Discount    Profit order_year  order_month  order_date  Ship_year  \
6972      0.2   29.4950       2017            9          19       2017
6973      0.7  -13.9380       2017            9          19       2017
6974      0.2    8.3328       2017            9          15       2017
6975      0.2    1.9926       2015            6          28       2015
6976      0.2   46.6320       2015            6          28       2015
6977      0.0   17.9940       2016           11          25       2016
6978      0.0   23.9463       2017           11          25       2017
6979      0.0   69.7050       2017            1          14       2017
```

```
6980      0.0    2.8416       2014           11            7       2014
6981      0.0   13.8828       2014           11            7       2014
6982      0.0   11.0860       2016            9            1       2016
6983      0.0    2.2828       2016            9            1       2016
6984      0.1   71.1620       2016            9            1       2016
6985      0.0   14.4648       2016            9            1       2016
6986      0.0    2.9900       2016            9            1       2016
6987      0.0   12.6720       2016            9            1       2016
6988      0.2  130.2885       2017           11           11       2017
6990      0.0   18.6606       2015           11            8       2015

      Ship_month  Ship_date
6972           9         25
6973           9         25
6974           9         19
6975           7          2
6976           7          2
6977          11         28
6978          11         29
6979           1         19
6980          11          8
6981          11          8
6982           9          5
6983           9          5
6984           9          5
6985           9          5
6986           9          5
6987           9          5
6988          11         16
6990          11         13

[18 rows x 27 columns]
```

[51]:
```
df = pd.concat([df, c], ignore_index=True) #conactinating the new dataframe␣
 ↪with existing dataframe df
```

[52]:
```
df.shape #shape of data frame
```

[52]:
```
(9872, 27)
```

[53]:
```
df.drop(df[df["Postal Code"].isin(a)].index,inplace=True) #droping rows that␣
 ↪have error in postal code after imputing with correct values
```

[54]:
```
df[df["Row ID"]==6973]
```

[54]:
```
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
9854  6973.0  CA-2017-153822 2017-09-19 2017-09-25  Standard Class
```

```
     Customer ID  Customer Name   Segment        Country     City    State  \
9854    AB-10105  Adrian Barton  Consumer  United States  Phoenix  Arizona

      …                                 Product Name   Sales Quantity  \
9854  …  Polycom VoiceStation 500 Conference phone  471.92      2.0

     Discount  Profit order_year  order_month  order_date  Ship_year  \
9854      0.2  29.495       2017            9          19       2017

      Ship_month  Ship_date
9854           9         25

[1 rows x 27 columns]
```

[55]:
```
df["Postal Code"].unique() #unique values in data frame
```

[55]:
```
array([42420., 90036., 33311., 90032., 28027., 98103., 76106., 53711.,
       84084., 94109., 68025., 19140., 84057., 90049., 77095., 75080.,
       77041., 60540., 32935., 55122., 48185., 19901., 47150., 10024.,
       12180., 90004., 60610., 85234., 22153., 10009., 49201., 38109.,
       77070., 35601., 94122., 27707., 60623., 29203., 55901., 80013.,
       28205., 60462., 10035., 50322., 43229., 37620., 19805., 61701.,
       85023., 95661., 64055., 91104., 43055., 53132., 85254., 95123.,
       98105., 98115., 73034., 90045., 19134., 88220., 78207., 77036.,
       62521., 71203.,  6824., 75051., 80219., 75220., 37064., 90604.,
       48601., 44256., 48227., 38401., 33614., 95051., 55044., 92037.,
       77506., 94513., 27514.,  7960., 45231., 94110., 90301., 97206.,
       33319., 80906.,  7109., 48180.,  8701., 22204., 80004.,  7601.,
       33710., 19143., 90805., 92345., 37130., 78745.,  1852., 31907.,
        6040., 78550., 85705., 62301.,  2038., 33024., 98198., 61604.,
       89115.,  2886., 33180., 28403., 92646., 40475., 80027.,  1841.,
       39212., 48187., 10801., 28052., 32216., 47201., 13021., 44312.,
       73071., 94521., 60068., 79109., 11757., 90008., 92024., 77340.,
       14609., 72701., 92627., 80134., 30318., 64118., 59405., 48234.,
       36116., 85204., 60653., 54302., 45503., 92804., 98270., 97301.,
       78041., 19120., 75217., 43123., 10011., 48126., 31088., 94591.,
       55407., 92691., 48307.,  7060., 85635., 98661., 60505., 76017.,
       40214., 75081., 44105., 75701., 27217., 22980., 19013., 27511.,
       32137., 10550., 48205., 33012., 11572., 92105., 60201., 48183.,
       55016., 71111., 50315., 93534., 23223., 28806., 92530., 68104.,
       98026., 92704., 53209., 41042., 44052.,  7036., 93905.,  8901.,
       17602.,  3301., 21044., 75043.,  6360., 22304., 43615., 87401.,
       92503., 90503., 78664., 92054., 33433., 23464., 92563., 28540.,
       52601., 98502., 20016., 65109., 63376., 61107., 33142., 78521.,
       10701., 94601., 28110., 20735., 30076., 72401., 47374., 94509.,
       33030., 46350., 48911., 44221., 89502., 22801., 92025., 48073.,
```

```
20852., 33065., 14215., 33437., 39503., 93727., 27834., 11561.,
35630., 31204., 52402.,  2908., 81001., 94533., 55106., 32725.,
42071.,  6457., 11520., 90660., 84604., 84062., 30080., 24153.,
44134., 36608.,  2740., 75061.,  8360., 85301., 14304., 27360.,
92683., 38301., 75019., 91767., 89031., 18103., 19711., 85281.,
92677.,  8302.,  2149., 13601., 54915., 98006., 75002., 79907.,
76051., 75007., 37167., 98031., 70506., 97224., 60076., 75023.,
23434., 46203.,  7002., 43017., 28314., 27405., 21215., 53142.,
66062., 98002., 74133., 97756., 27604., 74403.,  6450., 42104.,
46614.,  6010., 89015., 99207., 76248., 45014., 32127., 97504.,
22901., 59801., 33178., 29501., 97477., 32712., 19601., 80020.,
65807.,  7501., 73120., 23320., 79424., 65203., 37604., 36830.,
92404.,  1453., 59715., 85345., 44107.,  8861., 91761., 91730.,
56560., 75150., 92374., 95207., 32174., 94086.,  3820., 17403.,
77840., 63116.,  2169., 95336., 44240., 76903., 84106., 35810.,
37918., 72209., 48146., 43302., 80122.,  5408.,  4401., 38671.,
47362., 48640., 57103., 80525., 47905., 37042., 95823., 91360.,
 2148.,  1040., 87105., 89431., 92236., 60126.,  7055., 29406.,
23602., 14701., 46544., 43402.,  7090., 92253., 32303., 37211.,
98226., 60098., 76117., 60090., 29483., 71901., 80112., 43130.,
88001., 35244., 75034., 95687., 84107., 53186., 93309., 33068.,
45373., 78415., 90278., 32839.,  7050., 70601., 60035., 11550.,
46060., 55124., 29464., 48310., 54703., 78577., 59102., 97030.,
37421., 83642., 92307., 60440., 33801., 55369., 95695., 77489.,
77581., 94403., 49505., 93277., 66212., 92592., 92399.,  2151.,
77301., 60477., 52001., 48127., 87505., 28601., 60188., 56301.,
33161., 46226., 33317., 34952., 29730., 79762., 53214., 91911.,
66502., 16602., 80229., 61821., 47401., 71854., 78539., 77520.,
46142., 90712.,  2895., 54880., 76021., 98042., 74012., 33023.,
33021., 77536., 67212., 78501., 52240., 83704.,  2920., 61032.,
77642., 95610., 75056., 98052., 32114., 86442., 46368., 58103.,
46514., 91776., 45011., 33063., 30328., 44060., 73505., 23666.,
13440., 54601., 83501., 39401., 94526., 48858., 84321.,  6708.,
30605.,  4240., 61832., 85323., 30062., 85364., 54401., 99301.,
60302., 32503., 77573., 20877., 84043., 35401., 92553., 40324.,
80538., 85224., 59601., 63122., 76706., 48066., 60423., 18018.,
55113., 68801., 55125., 48237., 72756., 88101., 33458., 93101.,
75104., 68701., 84020., 48104., 91941., 83201., 49423.,  6460.,
60089., 92630., 96003., 95928., 13501., 72032., 82001., 42301.,
83605., 70065.,  3060., 38134., 94061., 37087., 93454., 60016.,
98632., 37075., 50701.,  2138., 60067.,  1915., 97405., 93030.,
98059., 60025., 33445., 80022., 77590., 27893., 87124., 27534.,
98208., 90640., 92020., 33407., 61761., 60174., 93010., 97123.,
91505., 95351., 67846.,  8401., 80501., 95616., 26003., 95037.,
 7011., 53081., 30344., 57701.,  1810., 34741.,  6484.,  6810.,
52302., 32771., 78666., 80634., 76063., 44035., 83301., 63301.,
33134., 60441.,  1752., 20707., 77803., 71603., 57401., 21740.,
```

```
       7017., 60004., 60543., 77705., 55433., 92672., 94568., 93405.,
      72762., 95240., 77571., 45040., 30188.])
```

[56]:
```
df[df["Postal Code"].isin(a)] #rechecking is there any error in postal code
```

[56]:
```
Empty DataFrame
Columns: [Row ID, Order ID, Order Date, Ship Date, Ship Mode, Customer ID,
Customer Name, Segment, Country, City, State, Postal Code, Region, Product ID,
Category, Sub-Category, Product Name, Sales, Quantity, Discount, Profit,
order_year, order_month, order_date, Ship_year, Ship_month, Ship_date]
Index: []

[0 rows x 27 columns]
```

[57]:
```
df.isnull().sum() #checking sum of null values
```

[57]:
```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        4
Customer ID      1
Customer Name    19
Segment          15
Country          18
City             0
State            0
Postal Code      0
Region           0
Product ID       2
Category         2
Sub-Category     4
Product Name     4
Sales            0
Quantity         0
Discount         0
Profit           13
order_year       0
order_month      0
order_date       0
Ship_year        0
Ship_month       0
Ship_date        0
dtype: int64
```

[58]:
```
# filled with united states as it has only one country
df['Country'].fillna('United States', inplace=True)
```

[59]:
```
#filling nan values in object dtype columns with mode
for i in df.select_dtypes(include="object"):
    df[i].fillna(df[i].mode()[0],inplace=True)
```

[60]:
```
#used knnimputer to impute null values in numerical columns except sales(as it␣
 ↪is dependent variable)
impute=KNNImputer()
for i in df.select_dtypes(include="number").columns:
    if i != 'Sales':
        df[i] = impute.fit_transform(df[[i]])
```

[61]:
```
df.isnull().sum() #checking sum of null values
```

[61]:
```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            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
order_year       0
order_month      0
order_date       0
Ship_year        0
Ship_month       0
Ship_date        0
dtype: int64
```

[62]:
```
df.dtypes #checking dtypes
```

[62]:
```
Row ID                  float64
Order ID                 object
Order Date       datetime64[ns]
```

```
Ship Date        datetime64[ns]
Ship Mode                object
Customer ID              object
Customer Name            object
Segment                  object
Country                  object
City                     object
State                    object
Postal Code             float64
Region                   object
Product ID               object
Category                 object
Sub-Category             object
Product Name             object
Sales                   float64
Quantity                float64
Discount                float64
Profit                  float64
order_year              float64
order_month             float64
order_date              float64
Ship_year               float64
Ship_month              float64
Ship_date               float64
dtype: object
```

[63]:
```
df.dropna(inplace=True) #dropping all nan value rows
```

[64]:
```
df.isnull().sum() #cheking null values in columns
```

[64]:
```
Row ID          0
Order ID        0
Order Date      0
Ship Date       0
Ship Mode       0
Customer ID     0
Customer Name   0
Segment         0
Country         0
City            0
State           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
order_year       0
order_month      0
order_date       0
Ship_year        0
Ship_month       0
Ship_date        0
dtype: int64
```

[65]:
```
df.shape #checking shape
```

[65]:
```
(9854, 27)
```

[66]:
```
((9994-9854)/9994)*100 #checking perecntage of rows i deleted
```

[66]:
```
1.4008405043025816
```

[67]:
```
df.duplicated().sum() #checking number of duplicate rows
```

[67]:
```
10
```

[68]:
```
df.drop_duplicates(inplace=True) #droping duplicate rows
```

[69]:
```
for i in df.columns:
    print(f"{i}:{df[i].unique()}\n")  #checking all unique values for each␣
 ↪column
```

```
Row ID:[1.000e+00 2.000e+00 3.000e+00 … 6.988e+03 6.989e+03 6.991e+03]

Order ID:['CA-2016-152156' 'CA-2016-138688' 'US-2015-108966' …
'CA-2014-146990'
 'CA-2016-116526' 'CA-2017-158561']

Order Date:['2016-11-08T00:00:00.000000000' '2016-06-12T00:00:00.000000000'
 '2015-10-11T00:00:00.000000000' … '2016-06-03T00:00:00.000000000'
 '2015-04-12T00:00:00.000000000' '2014-01-21T00:00:00.000000000']

Ship Date:['2016-11-11T00:00:00.000000000' '2016-06-16T00:00:00.000000000'
 '2015-10-18T00:00:00.000000000' … '2015-05-23T00:00:00.000000000'
 '2014-01-23T00:00:00.000000000' '2017-03-03T00:00:00.000000000']

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

Customer ID:['CG-12520' 'DV-13045' 'SO-20335' 'BH-11710' 'AA-10480' 'IM-15070'
```

'HP-14815' 'PK-19075' 'AG-10270' 'ZD-21925' 'KB-16585' 'SF-20065'
'EB-13870' 'EH-13945' 'TB-21520' 'MA-17560' 'GH-14485' 'SN-20710'
'LC-16930' 'RA-19885' 'ES-14080' 'ON-18715' 'PO-18865' 'LH-16900'
'DP-13000' 'JM-15265' 'TB-21055' 'KM-16720' 'PS-18970' 'BS-11590'
'KD-16270' 'HM-14980' 'JE-15745' 'KB-16600' 'SC-20770' 'DN-13690'
'JC-16105' 'CS-12400' 'PG-18895' 'GM-14455' 'JS-15685' 'RB-19465'
'GZ-14470' 'LC-16870' 'JM-15250' 'PA-19060' 'CV-12805' 'CL-12565'
'RC-19960' 'DK-13090' 'GG-14650' 'SC-20725' 'AD-10180' 'PF-19165'
'TS-21610' 'LS-16975' 'DW-13585' 'LC-16885' 'JD-15895' 'SH-19975'
'SG-20080' 'HA-14920' 'MG-17680' 'JE-16165' 'TW-21025' 'SP-20650'
'NK-18490' 'DB-13060' 'NP-18670' 'TT-21070' 'EM-13960' 'RD-19900'
'MJ-17740' 'BM-11140' 'CS-12130' 'JB-15400' 'SJ-20500' 'JK-15640'
'DK-13150' 'RM-19675' 'SK-19990' 'FM-14290' 'AM-10360' 'MP-17470'
'BS-11755' 'LC-17140' 'HK-14890' 'LE-16810' 'JH-15985' 'MS-17980'
'VW-21775' 'JH-15910' 'DS-13180' 'VD-21670' 'EA-14035' 'DB-13120'
'KL-16645' 'DW-13480' 'LH-17155' 'KC-16540' 'DL-13315' 'DR-12880'
'CC-12670' 'Dl-13600' 'SB-20290' 'RC-19825' 'AH-10210' 'CB-12535'
'CA-12310' 'KH-16690' 'BB-10990' 'AG-10495' 'JO-15280' 'AH-10195'
'NZ-18565' 'KL-16555' 'AS-10225' 'CR-12625' 'SH-20395' 'BP-11185'
'TS-21205' 'AG-10525' 'SP-20860' 'NM-18445' 'FA-14230' 'GK-14620'
'DJ-13510' 'PO-18850' 'JL-15850' 'DB-13615' 'CC-12550' 'TD-20995'
'AB-10060' 'JL-15505' 'VB-21745' 'KW-16435' 'JD-16060' 'MK-17905'
'GT-14755' 'AG-10900' 'MM-18280' 'AR-10405' 'RA-19915' 'AS-10285'
'LA-16780' 'DO-13435' 'DK-13225' 'NG-18430' 'MV-18190' 'JG-15115'
'BP-11095' 'VP-21730' 'SS-20140' 'AG-10675' 'LF-17185' 'RF-19840'
'KH-16510' 'KC-16675' 'CJ-12010' 'PB-19150' 'MP-17965' 'NF-18385'
'SD-20485' 'KH-16630' 'RB-19795' 'MK-18160' 'PO-19180' 'BB-11545'
'TB-21595' 'RB-19360' 'EB-13705' 'SC-20095' 'TN-21040' 'JS-15940'
'MH-17785' 'JP-15520' 'JE-15475' 'JG-15805' 'XP-21865' 'EM-14065'
'MT-18070' 'SA-20830' 'CW-11905' 'AJ-10960' 'SS-20590' 'RO-19780'
'MD-17350' 'MY-17380' 'CM-12385' 'LS-17245' 'BN-11515' 'DB-13210'
'MC-17605' 'BD-11605' 'MG-18145' 'KB-16240' 'JC-15340' 'RL-19615'
'AA-10375' 'EP-13915' 'DK-12985' 'BD-11500' 'LM-17065' 'AS-10135'
'BD-11320' 'GT-14710' 'AJ-10945' 'OT-18730' 'LP-17080' 'CA-12775'
'JF-15490' 'FP-14320' 'EB-13840' 'JF-15415' 'SF-20200' 'TG-21640'
'WB-21850' 'CC-12145' 'DV-13465' 'BD-11725' 'ZC-21910' 'MS-17830'
'LR-16915' 'TP-21130' 'CK-12205' 'AS-10240' 'AR-10510' 'NB-18655'
'GD-14590' 'CK-12595' 'NG-18355' 'CA-12265' 'SF-20965' 'MO-17800'
'AT-10735' 'FM-14380' 'DJ-13420' 'ME-17725' 'JD-16150' 'JL-15835'
'SC-20305' 'CC-12430' 'AR-10825' 'SR-20740' 'CR-12730' 'EH-14125'
'CB-12025' 'SP-20545' 'TH-21235' 'RP-19390' 'RB-19570' 'CD-11980'
'DJ-13630' 'GT-14635' 'MC-17845' 'RA-19285' 'NP-18325' 'AB-10165'
'JO-15550' 'JK-15370' 'BN-11470' 'DP-13165' 'TH-21550' 'AP-10915'
'RS-19765' 'SV-20365' 'CK-12325' 'RD-19810' 'MR-17545' 'SC-20695'
'JF-15355' 'EG-13900' 'DS-13030' 'PO-19195' 'SS-20875' 'PB-19105'
'RF-19735' 'YC-21895' 'DC-13285' 'CP-12340' 'BF-11020' 'LH-17020'
'CS-12250' 'AJ-10795' 'BV-11245' 'DL-12865' 'BM-11785' 'LT-17110'
'JK-15730' 'ES-14020' 'RH-19495' 'CD-11920' 'HW-14935' 'MC-18130'

'GM-14440' 'PJ-19015' 'BW-11110' 'TR-21325' 'PG-18820' 'JL-15175'
'BM-11650' 'EM-14095' 'AF-10885' 'GA-14725' 'CK-12760' 'DP-13105'
'BK-11260' 'SJ-20125' 'CM-12445' 'AJ-10780' 'LS-16945' 'GP-14740'
'PK-18910' 'SM-20005' 'AG-10765' 'PM-19135' 'LL-16840' 'JS-15595'
'EL-13735' 'PC-18745' 'HL-15040' 'MS-17365' 'GB-14530' 'JR-16210'
'BE-11335' 'SC-20050' 'RW-19630' 'SE-20110' 'AH-10075' 'JM-15535'
'JJ-15760' 'RK-19300' 'CG-12040' 'RP-19270' 'KC-16255' 'KH-16360'
'GH-14665' 'SW-20275' 'JA-15970' 'DL-12925' 'LW-16990' 'TB-21190'
'BS-11800' 'RW-19690' 'TZ-21580' 'AS-10630' 'TS-21340' 'SL-20155'
'MW-18235' 'RD-19585' 'RA-19945' 'MT-17815' 'VG-21790' 'JS-15880'
'KM-16225' 'HR-14770' 'DE-13255' 'AG-10390' 'JJ-15445' 'JH-15430'
'RD-19660' 'MO-17500' 'NS-18640' 'DG-13300' 'NF-18595' 'MG-17650'
'TS-21160' 'BD-11620' 'CM-12160' 'SN-20560' 'EH-14005' 'FO-14305'
'MS-17710' 'CC-12100' 'DW-13540' 'BT-11395' 'CY-12745' 'BT-11485'
'PS-19045' 'PV-18985' 'NM-18520' 'DL-13495' 'CS-12355' 'FH-14275'
'NC-18340' 'AA-10315' 'LT-16765' 'AP-10720' 'PM-18940' 'AT-10435'
'CA-12055' 'HR-14830' 'BT-11530' 'LH-16750' 'SW-20755' 'SP-20620'
'BF-11170' 'KT-16480' 'BG-11695' 'GM-14680' 'EJ-14155' 'NP-18700'
'MH-18115' 'JR-15700' 'SM-20950' 'CC-12220' 'PF-19225' 'DC-12850'
'BD-11770' 'GM-14500' 'TB-21355' 'JH-16180' 'EB-13975' 'QJ-19255'
'TC-21535' 'CS-12460' 'HG-14965' 'LW-16825' 'MC-17575' 'LP-17095'
'EB-14170' 'GZ-14545' 'CP-12085' 'FG-14260' 'LD-17005' 'AB-10255'
'MN-17935' 'JR-15670' 'JF-15190' 'CM-12115' 'AS-10045' 'KB-16315'
'BP-11290' 'ND-18370' 'LB-16735' 'KT-16465' 'HM-14860' 'AB-10600'
'SZ-20035' 'MG-17890' 'JK-16120' 'PP-18955' 'YS-21880' 'KM-16375'
'AB-10105' 'HA-14905' 'BT-11305' 'SV-20815' 'RW-19540' 'DK-12835'
'ST-20530' 'MM-17920' 'PW-19030' 'SC-20440' 'TS-21085' 'MC-17425'
'ME-17320' 'NH-18610' 'MB-18085' 'KD-16495' 'MB-17305' 'KN-16390'
'NP-18685' 'CS-12505' 'KD-16345' 'MS-17770' 'CM-12655' 'Co-12640'
'TS-21370' 'JW-15220' 'JD-15790' 'PC-19000' 'AR-10540' 'AI-10855'
'TB-21400' 'PL-18925' 'GH-14425' 'MP-18175' 'JM-15655' 'CL-11890'
'DB-13270' 'IG-15085' 'BO-11425' 'AB-10150' 'JW-16075' 'EB-13750'
'SG-20470' 'CM-12190' 'AW-10840' 'MC-18100' 'TT-21460' 'VG-21805'
'MY-18295' 'RD-19480' 'DP-13390' 'ML-17395' 'PN-18775' 'JC-15385'
'JG-15160' 'MC-17275' 'NW-18400' 'TB-21280' 'BS-11380' 'FH-14365'
'VM-21685' 'HH-15010' 'CD-12280' 'TH-21100' 'MM-18055' 'NS-18505'
'RB-19645' 'SW-20455' 'EB-13930' 'PS-18760' 'HF-14995' 'HZ-14950'
'CD-12790' 'JK-15205' 'FM-14215' 'ED-13885' 'DA-13450' 'JW-15955'
'RM-19375' 'ML-17755' 'CC-12685' 'JE-15610' 'RP-19855' 'TB-21175'
'BE-11455' 'JF-15565' 'PB-19210' 'BT-11680' 'JL-15235' 'CH-12070'
'ND-18460' 'BF-10975' 'KH-16330' 'GW-14605' 'AC-10420' 'NC-18625'
'ME-18010' 'BP-11230' 'JC-15775' 'AS-10090' 'AC-10450' 'MD-17860'
'DB-13660' 'EH-13990' 'EH-13765' 'MZ-17515' 'SC-20230' 'JE-15715'
'AC-10615' 'JD-16015' 'CB-12415' 'JS-16030' 'LW-17215' 'SC-20800'
'AM-10705' 'RH-19510' 'CT-11995' 'MC-17590' 'CC-12610' 'KA-16525'
'TC-20980' 'BF-11080' 'MM-17260' 'AH-10120' 'BW-11200' 'SW-20245'
'BS-11665' 'RF-19345' 'TB-21625' 'AF-10870' 'RB-19435' 'CS-11950'
'KF-16285' 'JH-15820' 'IL-15100' 'PB-18805' 'RH-19600' 'AW-10930'

```
 'RB-19705' 'ML-17410' 'DB-13555' 'MH-17620' 'DK-13375' 'BT-11440'
 'DB-13405' 'TG-21310' 'BF-11005' 'JM-16195' 'MZ-17335' 'MW-18220'
 'MV-17485' 'SM-20320' 'TP-21415' 'JK-15625' 'PJ-18835' 'RS-19420'
 'SV-20935' 'BC-11125' 'EM-13825' 'BM-11575' 'KN-16705' 'KW-16570'
 'SC-20260' 'CV-12295' 'SG-20605' 'TM-21010' 'EM-13810' 'ML-18040'
 'CR-12580' 'AZ-10750' 'PW-19240' 'SC-20380' 'CM-11935' 'GM-14695'
 'TB-21250' 'JM-15865' 'SC-20575' 'LS-17200' 'RR-19315' 'DB-12910'
 'TT-21220' 'LO-17170' 'KD-16615' 'NB-18580' 'BD-11635' 'CM-12235'
 'EN-13780' 'KN-16450' 'BO-11350' 'AG-10300' 'MC-17635' 'TA-21385'
 'JF-15295' 'TT-21265' 'SB-20170' 'CL-12700' 'HG-15025' 'NL-18310'
 'RR-19525' 'TC-21295' 'SV-20785' 'BE-11410' 'SC-20680' 'DF-13135'
 'FH-14350' 'MS-17530' 'RH-19555' 'GA-14515' 'JP-16135' 'Dp-13240'
 'MO-17950' 'ER-13855' 'DL-13330' 'MH-18025' 'DR-12940' 'DM-13015'
 'CA-11965' 'AC-10660' 'DM-13345' 'VF-21715' 'CC-12370' 'BF-11275'
 'HG-14845' 'BP-11155' 'EM-14140' 'MA-17995' 'AY-10555' 'GB-14575'
 'JB-16045' 'MG-17875' 'SR-20425' 'JB-16000' 'DM-12955' 'TC-21475'
 'SW-20350' 'RE-19450' 'BF-11215' 'KB-16405' 'JG-15310' 'EC-14050'
 'EB-14110' 'JP-15460' 'CS-11845' 'GH-14410' 'PT-19090' 'JL-15130'
 'AH-10030' 'CC-12475' 'DW-13195' 'SJ-20215' 'BG-11740' 'LB-16795'
 'CM-11815' 'EH-14185' 'TS-21505' 'PR-18880' 'LC-17050' 'CS-12490'
 'DH-13075' 'JO-15145' 'AH-10690' 'HJ-14875' 'MH-17455' 'RD-19930'
 'SC-20020' 'SU-20665' 'FC-14335' 'RB-19330' 'NC-18535' 'DB-12970'
 'MF-17665' 'RM-19750' 'VM-21835' 'EJ-13720' 'NC-18415' 'LS-17230'
 'KE-16420' 'DH-13675' 'PF-19120' 'VT-21700' 'MG-17695' 'SP-20920'
 'CS-12175' 'DK-12895' 'KM-16660' 'AA-10645' 'DD-13570' 'AH-10465'
 'TP-21565' 'EK-13795' 'CR-12820' 'SG-20890' 'AH-10585' 'NF-18475'
 'BS-11365' 'SH-20635' 'RD-19720' 'AG-10330' 'GR-14560' 'VS-21820'
 'TM-21490' 'TS-21430' 'DB-13360' 'NR-18550' 'JB-15925' 'CS-11860'
 'MF-18250' 'LW-17125' 'AR-10345' 'KS-16300' 'AB-10015' 'LR-17035'
 'SS-20410' 'JM-15580' 'JK-15325' 'DM-13525' 'ML-18265' 'MH-17290'
 'FC-14245' 'TH-21115' 'JK-16090' 'SB-20185' 'BG-11035' 'BW-11065'
 'MG-18205' 'DO-13645' 'BP-11050' 'TS-21655' 'EM-14200' 'AO-10810'
 'MH-17440' 'SS-20515' 'LD-16855' 'VP-21760' 'TC-21145' 'IM-15055'
 'AR-10570' 'CM-12715' 'FW-14395' 'LC-16960' 'HE-14800' 'BD-11560'
 'HD-14785' 'CJ-11875' 'RS-19870' 'SC-20845' 'RE-19405' 'SM-20905']

Customer Name:['Claire Gute' 'Darrin Van Huff' "Sean O'Donnell" 'Brosina
Hoffman'
 'Andrew Allen' 'Irene Maddox' 'Harold Pawlan' 'Pete Kriz'
 'Alejandro Grove' 'Zuschuss Donatelli' 'Ken Black' 'Sandra Flanagan'
 'Emily Burns' 'Eric Hoffmann' 'Tracy Blumstein' 'Matt Abelman'
 'Gene Hale' 'Steve Nguyen' 'Linda Cazamias' 'Ruben Ausman' 'Erin Smith'
 'Odella Nelson' "Patrick O'Donnell" 'Lena Hernandez' 'Darren Powers'
 'Janet Molinari' 'Ted Butterfield' 'Kunst Miller' 'Paul Stevenson'
 'Brendan Sweed' 'Karen Daniels' 'Henry MacAllister' 'Joel Eaton'
 'Ken Brennan' 'Stewart Carmichael' 'Duane Noonan' 'Julie Creighton'
 'Christopher Schild' 'Paul Gonzalez' 'Gary Mitchum' 'Jim Sink'
 'Rick Bensley' 'Gary Zandusky' 'Lena Cacioppo' 'Janet Martin'
```

'Pete Armstrong' 'Cynthia Voltz' 'Clay Ludtke' 'Ryan Crowe' 'Dave Kipp'
'Greg Guthrie' 'Steven Cartwright' 'Alan Dominguez' 'Philip Fox'
'Troy Staebel' 'Lindsay Shagiari' 'Dorothy Wardle' 'Lena Creighton'
'Jonathan Doherty' 'Sally Hughsby' 'Sandra Glassco' 'Helen Andreada'
'Maureen Gastineau' 'Justin Ellison' 'Tamara Willingham'
'Stephanie Phelps' 'Neil Knudson' 'Dave Brooks' 'Nora Paige'
'Ted Trevino' 'Eric Murdock' 'Ruben Dartt' 'Max Jones' 'Becky Martin'
'Chad Sievert' 'Jennifer Braxton' 'Shirley Jackson' 'Jim Kriz'
'David Kendrick' 'Robert Marley' 'Sally Knutson' 'Frank Merwin'
'Alice McCarthy' 'Mark Packer' 'Bruce Stewart' 'Logan Currie'
'Heather Kirkland' 'Laurel Elliston' 'Joseph Holt' 'Michael Stewart'
'Victoria Wilson' 'Jonathan Howell' 'David Smith' 'Valerie Dominguez'
'Erin Ashbrook' 'David Bremer' 'Ken Lonsdale' 'Dianna Wilson'
'Logan Haushalter' 'Kelly Collister' 'Delfina Latchford'
'Dan Reichenbach' 'Craig Carreira' 'Dorris liebe' 'Sean Braxton'
'Roy Collins' 'Alan Hwang' 'Claudia Bergmann' 'Christine Abelman'
'Kristen Hastings' 'Barry Blumstein' 'Andrew Gjertsen' "Jas O'Carroll"
'Alan Haines' 'Nick Zandusky' 'Kelly Lampkin' 'Alan Schoenberger'
'Corey Roper' 'Shahid Hopkins' 'Ben Peterman' 'Thomas Seio'
'Andy Gerbode' 'Sung Pak' 'Nathan Mautz' 'Frank Atkinson' 'Grace Kelly'
'Don Jones' "Patrick O'Brill" 'John Lucas' 'Doug Bickford'
'Clay Cheatham' 'Tamara Dahlen' 'Adam Bellavance' 'Jeremy Lonsdale'
'Victoria Brennan' 'Katrina Willman' 'Julia Dunbar' 'Michael Kennedy'
'Guy Thornton' 'Arthur Gainer' 'Muhammed MacIntyre' 'Allen Rosenblatt'
'Russell Applegate' 'Alejandro Savely' 'Laura Armstrong' 'Denny Ordway'
'Dean Katz' 'Nathan Gelder' 'Mike Vittorini' 'Jack Garza' 'Bart Pistole'
'Victor Preis' 'Saphhira Shifley' 'Anna Gayman' 'Luke Foster'
'Roy Französisch' 'Keith Herrera' 'Kimberly Carter' 'Caroline Jumper'
'Philip Brown' 'William Brown' 'Michael Paige' 'Natalie Fritzler'
'Shirley Daniels' 'Ken Heidel' 'Ross Baird' 'Mike Kennedy'
'Philisse Overcash' 'Brenda Bowman' 'Troy Blackwell' 'Raymond Buch'
'Ed Braxton' 'Sanjit Chand' 'Tanja Norvell' 'Joni Sundaresam'
'Maya Herman' 'Jeremy Pistek' 'Jeremy Ellison' 'John Grady'
'Xylona Preis' 'Erin Mull' 'Michelle Tran' 'Sue Ann Reed' 'Carl Weiss'
'Astrea Jones' 'Sonia Sunley' "Rose O'Brian" 'Maribeth Dona'
'Maribeth Yedwab' 'Christopher Martinez' 'Lynn Smith' 'Bradley Nguyen'
'Dean Braden' 'Matt Connell' 'Brian Dahlen' 'Mike Gockenbach'
'Karen Bern' 'Jasper Cacioppo' 'Rob Lucas' 'Allen Armold' 'Emily Phan'
'Darren Koutras' 'Bradley Drucker' 'Liz MacKendrick' 'Adrian Shami'
'Bill Donatelli' 'Greg Tran' 'Ashley Jarboe' 'Olvera Toch'
'Liz Pelletier' 'Cynthia Arntzen' 'Jeremy Farry' 'Frank Preis'
'Ellis Ballard' 'Jennifer Ferguson' 'Sarah Foster' 'Trudy Glocke'
'Carlos Soltero' 'Charles Crestani' 'Dianna Vittorini' 'Bruce Degenhardt'
'Zuschuss Carroll' 'Melanie Seite' 'Lena Radford' 'Theone Pippenger'
'Chloris Kastensmidt' 'Alan Shonely' 'Andrew Roberts' 'Nona Balk'
'Giulietta Dortch' 'Clytie Kelty' 'Nat Gilpin' 'Christina Anderson'
'Sylvia Foulston' "Meg O'Connel" 'Annie Thurman' 'Fred McMath'
'Denny Joy' 'Max Engle' 'Justin Deggeller' 'John Lee' 'Sean Christensen'

'Chuck Clark' 'Anthony Rawles' 'Steven Roelle' 'Craig Reiter'
'Eugene Hildebrand' 'Cassandra Brandow' 'Sibella Parks' 'Tiffany House'
'Resi Pölking' 'Rob Beeghly' 'Carol Darley' 'Doug Jacobs'
'Grant Thornton' 'Michael Chen' 'Ralph Arnett' 'Naresj Patel'
'Alan Barnes' 'Jesus Ocampo' 'Jay Kimmel' 'Brad Norvell' 'David Philippe'
'Tracy Hopkins' 'Arthur Prichep' 'Roland Schwarz' 'Seth Vernon'
'Christine Kargatis' 'Ross DeVincentis' 'Mathew Reese' 'Steve Chapman'
'Jay Fein' 'Emily Grady' 'Darrin Sayre' 'Phillina Ober' 'Sung Shariari'
'Peter Bühler' 'Roland Fjeld' 'Yoseph Carroll' 'Debra Catini'
'Christine Phan' 'Barry Französisch' 'Lisa Hazard' 'Chris Selesnick'
'Anthony Johnson' 'Benjamin Venier' 'Dan Lawera' 'Bryan Mills'
'Liz Thompson' 'Joe Kamberova' 'Erica Smith' 'Rick Hansen' 'Carlos Daly'
'Helen Wasserman' 'Mike Caudle' 'Gary McGarr' 'Pauline Johnson'
'Bart Watters' 'Toby Ritter' 'Patrick Gardner' 'James Lanier'
'Brian Moss' 'Eudokia Martin' 'Art Foster' 'Guy Armstrong' 'Cyma Kinney'
'Dave Poirier' 'Berenike Kampe' 'Sanjit Jacobs' 'Chuck Magee'
'Anthony Jacobs' 'Linda Southworth' 'Guy Phonely' 'Paul Knutson'
'Sally Matthias' 'Anthony Garverick' 'Peter McVee' 'Lauren Leatherbury'
'Jill Stevenson' 'Ed Ludwig' 'Pamela Coakley' 'Hunter Lopez'
'Maribeth Schnelling' 'George Bell' 'Justin Ritter' 'Bill Eplett'
'Sample Company A' 'Rob Williams' 'Sanjit Engle' 'Adam Hart'
'Jessica Myrick' 'Joel Jenkins' 'Ralph Kennedy' 'Catherine Glotzbach'
'Rachel Payne' 'Karen Carlisle' 'Katherine Hughes' 'Greg Hansen'
'Scott Williamson' 'Joseph Airdo' 'Daniel Lacy' 'Lindsay Williams'
'Thomas Brumley' 'Bryan Spruell' 'Robert Waldorf' 'Tracy Zic'
'Ann Steele' 'Toby Swindell' 'Sara Luxemburg' 'Mitch Willingham'
'Rob Dowd' 'Ryan Akin' 'Meg Tillman' 'Vivek Gonzalez' 'John Stevenson'
'Kalyca Meade' 'Hallie Redmond' 'Deanra Eno' 'Allen Goldenen'
'Jennifer Jackson' 'Jennifer Halladay' 'Robert Dilbeck' "Mary O'Rourke"
'Noel Staavos' 'Deirdre Greer' 'Nicole Fjeld' 'Matthew Grinstein'
'Theresa Swint' 'Brian DeCherney' 'Charles McCrossin' 'Skye Norling'
'Erica Hernandez' 'Frank Olsen' 'Maurice Satty' 'Chad Cunningham'
'Don Weiss' 'Bill Tyler' 'Craig Yedwab' 'Brad Thomas' 'Penelope Sewall'
'Paul Van Hugh' 'Neoma Murray' 'Dionis Lloyd' 'Christine Sundaresam'
'Frank Hawley' 'Nat Carroll' 'Alex Avila' 'Larry Tron' 'Anne Pryor'
'Paul MacIntyre' 'Alyssa Tate' 'Cathy Armstrong' 'Harold Ryan'
'Bradley Talbott' 'Larry Hughes' 'Steven Ward' 'Stefania Perrino'
'Ben Ferrer' 'Kean Thornton' 'Brooke Gillingham' 'Greg Matthias'
'Eva Jacobs' 'Nora Preis' 'Mick Hernandez' 'Jocasta Rupert'
'Suzanne McNair' 'Chris Cortes' 'Phillip Flathmann' 'Dan Campbell'
'Bryan Davis' 'Gene McClure' 'Todd Boyes' 'Justin Hirsh' 'Erica Bern'
'Quincy Jones' 'Tracy Collins' 'Chuck Sachs' 'Henry Goldwyn'
'Laurel Workman' 'Matt Collins' 'Liz Preis' 'Evan Bailliet'
'George Zrebassa' 'Cathy Prescott' 'Frank Gastineau' 'Lisa DeCherney'
'Alejandro Ballentine' 'Michael Nguyen' 'Jim Radford' 'Jamie Frazer'
'Chad McGuire' 'Aaron Smayling' 'Karl Braun' 'Beth Paige'
'Natalie DeCherney' 'Larry Blacks' 'Kean Takahito' 'Harry Marie'
'Ann Blume' 'Sam Zeldin' 'Michael Granlund' 'Julie Kriz' 'Paul Prost'

'Yana Sorensen' 'Katherine Murray' 'Adrian Barton' 'Helen Abelman'
'Beth Thompson' 'Stuart Van' 'Rick Wilson' 'Damala Kotsonis' 'Shui Tom'
'Michael Moore' 'Pauline Webber' 'Shaun Chance' 'Thais Sissman'
'Mark Cousins' 'Maria Etezadi' 'Nicole Hansen' 'Mick Brown'
'Keith Dawkins' 'Maria Bertelson' 'Katherine Nockton' 'Nora Pelletier'
'Cindy Stewart' 'Katherine Ducich' 'Maxwell Schwartz' 'Corinna Mitchell'
'Corey-Lock' 'Todd Sumrall' 'Jane Waco' 'John Dryer' 'Pauline Chand'
'Andy Reiter' 'Arianne Irving' 'Tom Boeckenhauer' 'Paul Lucas'
'Gary Hwang' 'Mike Pelletier' 'Jim Mitchum' 'Carl Ludwig'
'Deborah Brumfield' 'Ivan Gibson' 'Bobby Odegard' 'Aimee Bixby'
'Julia West' 'Edward Becker' 'Sheri Gordon' 'Charlotte Melton'
'Anthony Witt' 'Mick Crebagga' 'Tonja Turnell' 'Vivek Grady'
'Muhammed Yedwab' 'Rick Duston' 'Dennis Pardue' 'Marina Lichtenstein'
'Parhena Norris' 'Jenna Caffey' 'James Galang' 'Marc Crier'
'Natalie Webber' 'Toby Braunhardt' 'Bill Stewart' 'Fred Hopkins'
'Valerie Mitchum' 'Hilary Holden' 'Christina DeMoss' 'Thea Hendricks'
'Michelle Moray' 'Neola Schneider' 'Robert Barroso' 'Shaun Weien'
'Eric Barreto' 'Pamela Stobb' 'Herbert Flentye' 'Henia Zydlo'
'Cynthia Delaney' 'Jamie Kunitz' 'Filia McAdams' 'Emily Ducich'
'Dianna Arnett' 'Joni Wasserman' 'Raymond Messe' 'Max Ludwig'
'Craig Carroll' 'Jim Epp' 'Roy Phan' 'Thomas Boland' 'Brad Eason'
'Jill Fjeld' 'Phillip Breyer' 'Brian Thompson' 'Janet Lee' 'Cathy Hwang'
'Neil Ducich' 'Barbara Fisher' 'Katharine Harms' 'Giulietta Weimer'
'Alyssa Crouse' 'Noah Childs' 'Michelle Ellison' 'Benjamin Patterson'
'John Castell' 'Adam Shillingsburg' 'Amy Cox' 'Michael Dominguez'
'Duane Benoit' 'Erica Hackney' 'Edward Hooks' 'Mary Zewe' 'Scot Coram'
'Joe Elijah' 'Ann Chong' 'Joy Daniels' 'Christy Brittain' 'Joy Smith'
'Luke Weiss' 'Stuart Calhoun' 'Anne McFarland' 'Rick Huthwaite'
'Carol Triggs' 'Matt Collister' 'Corey Catlett' 'Kelly Andreada'
'Tamara Chand' 'Bart Folk' 'Magdelene Morse' 'Adrian Hane' 'Ben Wallace'
'Scot Wooten' 'Brian Stugart' 'Randy Ferguson' 'Trudy Brown'
'Art Ferguson' 'Richard Bierner' 'Karen Ferguson' 'John Huston'
'Ivan Liston' 'Patrick Bzostek' 'Rob Haberlin' 'Arthur Wiediger'
'Roger Barcio' 'Maris LaWare' 'Dorothy Badders' 'Matt Hagelstein'
'Dennis Kane' 'Bobby Trafton' 'Denny Blanton' 'Toby Gnade' 'Barry Franz'
'Justin MacKendrick' 'Maria Zettner' 'Mitch Webber' 'Mark Van Huff'
'Sean Miller' 'Tom Prescott' 'Jim Karlsson' 'Patrick Jones'
'Ricardo Sperren' 'Susan Vittorini' 'Becky Castell' 'Elizabeth Moffitt'
'Brendan Murry' 'Kristina Nunn' 'Kelly Williams' 'Scott Cohen'
'Christina VanderZanden' 'Speros Goranitis' 'Tamara Manning'
'Eleni McCrary' 'Michelle Lonsdale' 'Clay Rozendal' 'Annie Zypern'
'Pierre Wener' 'Shahid Collister' 'Carlos Meador' 'Greg Maxwell'
'Tim Brockman' 'John Murray' 'Sonia Cooley' 'Luke Schmidt' 'Ralph Ritter'
'Daniel Byrd' 'Thomas Thornton' 'Lori Olson' 'Ken Dana' 'Nicole Brennan'
'Brian Derr' 'Chris McAfee' 'Edward Nazzal' 'Kean Nguyen' 'Bill Overfelt'
'Aleksandra Gannaway' 'Matthew Clasen' 'Tom Ashbrook' 'Jason Fortune-'
'Tim Taslimi' 'Sarah Bern' 'Craig Leslie' 'Hunter Glantz'
'Nancy Lomonaco' 'Rick Reed' 'Toby Carlisle' 'Stewart Visinsky'

```
 'Bobby Elias' 'Steve Carroll' 'David Flashing' 'Fred Harton'
 'MaryBeth Skach' 'Ritsa Hightower' 'George Ashbrook' 'Julie Prescott'
 'Dean percer' 'Michael Oakman' 'Elpida Rittenbach' 'Denise Leinenbach'
 'Michelle Huthwaite' 'Daniel Raglin' 'Darrin Martin' 'Carol Adams'
 'Anna Chung' 'Denise Monton' 'Vicky Freymann' 'Christopher Conant'
 'Beth Fritzler' 'Harry Greene' 'Becky Pak' 'Eugene Moren'
 'Michelle Arnett' 'Andy Yotov' 'Giulietta Baptist' 'Julia Barnett'
 'Michael Grace' 'Sharelle Roach' 'Joy Bell-' 'Dario Medina'
 'Tony Chapman' 'Sean Wendt' 'Richard Eichhorn' 'Benjamin Farhat'
 'Katrina Bavinger' 'Jason Gross' 'Erin Creighton' 'Eugene Barchas'
 'Jennifer Patt' 'Cari Sayre' 'Gary Hansen' 'Pete Takahito' 'Jack Lebron'
 'Aaron Hawkins' 'Cindy Chapman' 'David Wiener' 'Sarah Jordon'
 'Bruce Geld' 'Laurel Beltran' 'Candace McMahon' 'Evan Henry' 'Tony Sayre'
 'Patrick Ryan' 'Liz Carlisle' 'Cindy Schnelling' 'Dave Hallsten'
 "Jack O'Briant" 'Anna Häberlin' 'Heather Jas' 'Mark Hamilton'
 "Russell D'Ascenzo" 'Sam Craven' 'Stephanie Ulpright' 'Fred Chung'
 'Randy Bradley' 'Nick Crebassa' 'Darren Budd' 'Maureen Fritzler'
 'Roland Murray' 'Vivian Mathis' 'Ed Jacobs' 'Nathan Cano'
 'Lycoris Saunders' 'Katrina Edelman' 'Duane Huffman' 'Peter Fuller'
 'Valerie Takahito' 'Maureen Gnade' 'Susan Pistek' 'Charles Sheldon'
 'Dana Kaydos' 'Khloe Miller' 'Anna Andreadi' 'Dorothy Dickinson'
 'Amy Hunt' 'Tracy Poddar' 'Eileen Kiefer' 'Cyra Reiten' 'Susan Gilcrest'
 'Angele Hood' 'Neil Französisch' 'Bill Shonely' 'Stefanie Holloman'
 'Roger Demir' 'Alex Grayson' 'Georgia Rosenberg' 'Vivek Sundaresam'
 'Tony Molinari' 'Tom Stivers' 'Dennis Bolton' 'Nick Radford'
 'Joni Blumstein' 'Cari Schnelling' 'Monica Federle' 'Liz Willingham'
 'Alex Russell' 'Karen Seio' 'Aaron Bergman' 'Lisa Ryan' 'Shahid Shariari'
 'Jill Matthias' 'Jason Klamczynski' 'Don Miller' 'Muhammed Lee'
 'Marc Harrigan' 'Frank Carlisle' 'Thea Hudgings' 'Juliana Krohn'
 'Sarah Brown' 'Barry Gonzalez' 'Barry Weirich' 'Mitch Gastineau'
 "Doug O'Connell" 'Barry Pond' 'Trudy Schmidt' 'Evan Minnotte'
 "Anthony O'Donnell" 'Mark Haberlin' 'Shirley Schmidt' 'Lela Donovan'
 'Victoria Pisteka' 'Theresa Coyne' 'Ionia McGrath' 'Anemone Ratner'
 'Craig Molinari' 'Fred Wasserman' 'Lindsay Castell' 'Harold Engle'
 'Brendan Dodson' 'Harold Dahlen' 'Carl Jackson' 'Roy Skaria' 'Sung Chung'
 'Ricardo Emerson' 'Susan MacKendrick']

Segment:['Consumer' 'Corporate' 'Home Office']

Country:['United States']

City:['Henderson' 'Los Angeles' 'Fort Lauderdale' 'Concord' 'Seattle'
 'Fort Worth' 'Madison' 'West Jordan' 'San Francisco' 'Fremont'
 'Philadelphia' 'Orem' 'Houston' 'Richardson' 'Naperville' 'Melbourne'
 'Eagan' 'Westland' 'Dover' 'New Albany' 'New York City' 'Troy' 'Chicago'
 'Gilbert' 'Springfield' 'Jackson' 'Memphis' 'Decatur' 'Durham' 'Columbia'
 'Rochester' 'Aurora' 'Charlotte' 'Orland Park' 'Urbandale' 'Columbus'
 'Bristol' 'Wilmington' 'Bloomington' 'Phoenix' 'Roseville' 'Independence'
```

'Pasadena' 'Newark' 'Franklin' 'Scottsdale' 'San Jose' 'Edmond'
'Carlsbad' 'San Antonio' 'Monroe' 'Fairfield' 'Grand Prairie' 'Denver'
'Dallas' 'Whittier' 'Saginaw' 'Medina' 'Detroit' 'Tampa' 'Santa Clara'
'Lakeville' 'San Diego' 'Brentwood' 'Chapel Hill' 'Morristown'
'Cincinnati' 'Inglewood' 'Portland' 'Tamarac' 'Colorado Springs'
'Belleville' 'Taylor' 'Lakewood' 'Arlington' 'Arvada' 'Hackensack'
'Saint Petersburg' 'Long Beach' 'Hesperia' 'Murfreesboro' 'Austin'
'Lowell' 'Manchester' 'Harlingen' 'Tucson' 'Quincy' 'Pembroke Pines'
'Des Moines' 'Peoria' 'Las Vegas' 'Warwick' 'Miami' 'Huntington Beach'
'Richmond' 'Louisville' 'Lawrence' 'Canton' 'New Rochelle' 'Gastonia'
'Jacksonville' 'Auburn' 'Akron' 'Norman' 'Park Ridge' 'Amarillo'
'Lindenhurst' 'Huntsville' 'Fayetteville' 'Costa Mesa' 'Parker' 'Atlanta'
'Gladstone' 'Great Falls' 'Montgomery' 'Mesa' 'Green Bay' 'Anaheim'
'Marysville' 'Salem' 'Laredo' 'Grove City' 'Dearborn' 'Warner Robins'
'Vallejo' 'Minneapolis' 'Mission Viejo' 'Rochester Hills' 'Plainfield'
'Sierra Vista' 'Vancouver' 'Cleveland' 'Tyler' 'Burlington' 'Waynesboro'
'Chester' 'Cary' 'Palm Coast' 'Mount Vernon' 'Hialeah' 'Oceanside'
'Evanston' 'Trenton' 'Cottage Grove' 'Bossier City' 'Lancaster'
'Asheville' 'Lake Elsinore' 'Omaha' 'Edmonds' 'Santa Ana' 'Milwaukee'
'Florence' 'Lorain' 'Linden' 'Salinas' 'New Brunswick' 'Garland'
'Norwich' 'Alexandria' 'Toledo' 'Farmington' 'Riverside' 'Torrance'
'Round Rock' 'Boca Raton' 'Virginia Beach' 'Murrieta' 'Olympia'
'Washington' 'Jefferson City' 'Saint Peters' 'Rockford' 'Brownsville'
'Yonkers' 'Oakland' 'Clinton' 'Encinitas' 'Roswell' 'Jonesboro' 'Antioch'
'Homestead' 'La Porte' 'Lansing' 'Cuyahoga Falls' 'Reno' 'Harrisonburg'
'Escondido' 'Royal Oak' 'Rockville' 'Coral Springs' 'Buffalo'
'Boynton Beach' 'Gulfport' 'Fresno' 'Greenville' 'Macon' 'Cedar Rapids'
'Providence' 'Pueblo' 'Saint Paul' 'Deltona' 'Murray' 'Middletown'
'Freeport' 'Pico Rivera' 'Provo' 'Pleasant Grove' 'Smyrna' 'Parma'
'Mobile' 'New Bedford' 'Irving' 'Vineland' 'Glendale' 'Niagara Falls'
'Thomasville' 'Westminster' 'Coppell' 'Pomona' 'North Las Vegas'
'Allentown' 'Tempe' 'Laguna Niguel' 'Bridgeton' 'Everett' 'Watertown'
'Appleton' 'Bellevue' 'Allen' 'El Paso' 'Grapevine' 'Carrollton' 'Kent'
'Lafayette' 'Tigard' 'Skokie' 'Plano' 'Suffolk' 'Indianapolis' 'Bayonne'
'Dublin' 'Greensboro' 'Baltimore' 'Kenosha' 'Olathe' 'Tulsa' 'Redmond'
'Raleigh' 'Muskogee' 'Meriden' 'Bowling Green' 'South Bend' 'Spokane'
'Keller' 'Port Orange' 'Medford' 'Charlottesville' 'Missoula' 'Apopka'
'Reading' 'Broomfield' 'Paterson' 'Oklahoma City' 'Chesapeake' 'Lubbock'
'Johnson City' 'San Bernardino' 'Leominster' 'Bozeman' 'Perth Amboy'
'Ontario' 'Rancho Cucamonga' 'Moorhead' 'Mesquite' 'Redlands' 'Stockton'
'Ormond Beach' 'Sunnyvale' 'York' 'College Station' 'Saint Louis'
'Manteca' 'San Angelo' 'Salt Lake City' 'Knoxville' 'Little Rock'
'Lincoln Park' 'Marion' 'Littleton' 'Bangor' 'Southaven' 'New Castle'
'Midland' 'Sioux Falls' 'Fort Collins' 'Clarksville' 'Sacramento'
'Thousand Oaks' 'Malden' 'Holyoke' 'Albuquerque' 'Sparks' 'Coachella'
'Elmhurst' 'Passaic' 'North Charleston' 'Newport News' 'Jamestown'
'Mishawaka' 'Westfield' 'La Quinta' 'Tallahassee' 'Nashville'
'Bellingham' 'Woodstock' 'Haltom City' 'Wheeling' 'Summerville'

```
 'Hot Springs' 'Englewood' 'Las Cruces' 'Hoover' 'Frisco' 'Vacaville'
 'Waukesha' 'Bakersfield' 'Pompano Beach' 'Corpus Christi' 'Redondo Beach'
 'Orlando' 'Orange' 'Lake Charles' 'Highland Park' 'Hempstead'
 'Noblesville' 'Apple Valley' 'Mount Pleasant' 'Sterling Heights'
 'Eau Claire' 'Pharr' 'Billings' 'Gresham' 'Chattanooga' 'Meridian'
 'Bolingbrook' 'Lakeland' 'Maple Grove' 'Woodland' 'Missouri City'
 'Pearland' 'San Mateo' 'Grand Rapids' 'Visalia' 'Overland Park'
 'Temecula' 'Yucaipa' 'Revere' 'Conroe' 'Tinley Park' 'Dubuque'
 'Dearborn Heights' 'Santa Fe' 'Hickory' 'Carol Stream' 'Saint Cloud'
 'North Miami' 'Plantation' 'Port Saint Lucie' 'Rock Hill' 'Odessa'
 'West Allis' 'Chula Vista' 'Manhattan' 'Altoona' 'Thornton' 'Champaign'
 'Texarkana' 'Edinburg' 'Baytown' 'Greenwood' 'Woonsocket' 'Superior'
 'Bedford' 'Covington' 'Broken Arrow' 'Miramar' 'Hollywood' 'Deer Park'
 'Wichita' 'Mcallen' 'Iowa City' 'Boise' 'Cranston' 'Port Arthur'
 'Citrus Heights' 'The Colony' 'Daytona Beach' 'Bullhead City' 'Portage'
 'Fargo' 'Elkhart' 'San Gabriel' 'Hamilton' 'Margate' 'Sandy Springs'
 'Mentor' 'Lawton' 'Hampton' 'Rome' 'La Crosse' 'Lewiston' 'Hattiesburg'
 'Danville' 'Logan' 'Waterbury' 'Athens' 'Avondale' 'Marietta' 'Yuma'
 'Wausau' 'Pasco' 'Oak Park' 'Pensacola' 'League City' 'Gaithersburg'
 'Lehi' 'Tuscaloosa' 'Moreno Valley' 'Georgetown' 'Loveland' 'Chandler'
 'Helena' 'Kirkwood' 'Waco' 'Frankfort' 'Bethlehem' 'Grand Island'
 'Woodbury' 'Rogers' 'Clovis' 'Jupiter' 'Santa Barbara' 'Cedar Hill'
 'Norfolk' 'Draper' 'Ann Arbor' 'La Mesa' 'Pocatello' 'Holland' 'Milford'
 'Buffalo Grove' 'Lake Forest' 'Redding' 'Chico' 'Utica' 'Conway'
 'Cheyenne' 'Owensboro' 'Caldwell' 'Kenner' 'Nashua' 'Bartlett'
 'Redwood City' 'Lebanon' 'Santa Maria' 'Des Plaines' 'Longview'
 'Hendersonville' 'Waterloo' 'Cambridge' 'Palatine' 'Beverly' 'Eugene'
 'Oxnard' 'Renton' 'Glenview' 'Delray Beach' 'Commerce City' 'Texas City'
 'Wilson' 'Rio Rancho' 'Goldsboro' 'Montebello' 'El Cajon'
 'West Palm Beach' 'Normal' 'Saint Charles' 'Camarillo' 'Hillsboro'
 'Burbank' 'Modesto' 'Garden City' 'Atlantic City' 'Longmont' 'Davis'
 'Morgan Hill' 'Clifton' 'Sheboygan' 'East Point' 'Rapid City' 'Andover'
 'Kissimmee' 'Shelton' 'Danbury' 'Sanford' 'San Marcos' 'Greeley'
 'Mansfield' 'Elyria' 'Twin Falls' 'Coral Gables' 'Romeoville'
 'Marlborough' 'Laurel' 'Bryan' 'Pine Bluff' 'Aberdeen' 'Hagerstown'
 'East Orange' 'Arlington Heights' 'Oswego' 'Beaumont' 'Coon Rapids'
 'San Clemente' 'San Luis Obispo' 'Springdale' 'Lodi' 'Mason']

State:['Kentucky' 'California' 'Florida' 'North Carolina' 'Washington' 'Texas'
 'Wisconsin' 'Utah' 'Nebraska' 'Pennsylvania' 'Illinois' 'Minnesota'
 'Michigan' 'Delaware' 'Indiana' 'New York' 'Arizona' 'Virginia'
 'Tennessee' 'Alabama' 'South Carolina' 'Colorado' 'Iowa' 'Ohio'
 'Missouri' 'Oklahoma' 'New Mexico' 'Louisiana' 'Connecticut' 'New Jersey'
 'Oregon' 'Massachusetts' 'Georgia' 'Nevada' 'Rhode Island' 'Mississippi'
 'Arkansas' 'Montana' 'New Hampshire' 'Maryland' 'District of Columbia'
 'Kansas' 'Vermont' 'Maine' 'South Dakota' 'Idaho' 'North Dakota'
 'Wyoming' 'West Virginia']
```

Postal Code:[42420. 90036. 33311. 90032. 28027. 98103. 76106. 53711. 84084.
94109.
 68025. 19140. 84057. 90049. 77095. 75080. 77041. 60540. 32935. 55122.
 48185. 19901. 47150. 10024. 12180. 90004. 60610. 85234. 22153. 10009.
 49201. 38109. 77070. 35601. 94122. 27707. 60623. 29203. 55901. 80013.
 28205. 60462. 10035. 50322. 43229. 37620. 19805. 61701. 85023. 95661.
 64055. 91104. 43055. 53132. 85254. 95123. 98105. 98115. 73034. 90045.
 19134. 88220. 78207. 77036. 62521. 71203. 6824. 75051. 80219. 75220.
 37064. 90604. 48601. 44256. 48227. 38401. 33614. 95051. 55044. 92037.
 77506. 94513. 27514. 7960. 45231. 94110. 90301. 97206. 33319. 80906.
  7109. 48180. 8701. 22204. 80004. 7601. 33710. 19143. 90805. 92345.
 37130. 78745. 1852. 31907. 6040. 78550. 85705. 62301. 2038. 33024.
 98198. 61604. 89115. 2886. 33180. 28403. 92646. 40475. 80027. 1841.
 39212. 48187. 10801. 28052. 32216. 47201. 13021. 44312. 73071. 94521.
 60068. 79109. 11757. 90008. 92024. 77340. 14609. 72701. 92627. 80134.
 30318. 64118. 59405. 48234. 36116. 85204. 60653. 54302. 45503. 92804.
 98270. 97301. 78041. 19120. 75217. 43123. 10011. 48126. 31088. 94591.
 55407. 92691. 48307. 7060. 85635. 98661. 60505. 76017. 40214. 75081.
 44105. 75701. 27217. 22980. 19013. 27511. 32137. 10550. 48205. 33012.
 11572. 92105. 60201. 48183. 55016. 71111. 50315. 93534. 23223. 28806.
 92530. 68104. 98026. 92704. 53209. 41042. 44052. 7036. 93905. 8901.
 17602. 3301. 21044. 75043. 6360. 22304. 43615. 87401. 92503. 90503.
 78664. 92054. 33433. 23464. 92563. 28540. 52601. 98502. 20016. 65109.
 63376. 61107. 33142. 78521. 10701. 94601. 28110. 20735. 30076. 72401.
 47374. 94509. 33030. 46350. 48911. 44221. 89502. 22801. 92025. 48073.
 20852. 33065. 14215. 33437. 39503. 93727. 27834. 11561. 35630. 31204.
 52402. 2908. 81001. 94533. 55106. 32725. 42071. 6457. 11520. 90660.
 84604. 84062. 30080. 24153. 44134. 36608. 2740. 75061. 8360. 85301.
 14304. 27360. 92683. 38301. 75019. 91767. 89031. 18103. 19711. 85281.
 92677. 8302. 2149. 13601. 54915. 98006. 75002. 79907. 76051. 75007.
 37167. 98031. 70506. 97224. 60076. 75023. 23434. 46203. 7002. 43017.
 28314. 27405. 21215. 53142. 66062. 98002. 74133. 97756. 27604. 74403.
  6450. 42104. 46614. 6010. 89015. 99207. 76248. 45014. 32127. 97504.
 22901. 59801. 33178. 29501. 97477. 32712. 19601. 80020. 65807. 7501.
 73120. 23320. 79424. 65203. 37604. 36830. 92404. 1453. 59715. 85345.
 44107. 8861. 91761. 91730. 56560. 75150. 92374. 95207. 32174. 94086.
  3820. 17403. 77840. 63116. 2169. 95336. 44240. 76903. 84106. 35810.
 37918. 72209. 48146. 43302. 80122. 5408. 4401. 38671. 47362. 48640.
 57103. 80525. 47905. 37042. 95823. 91360. 2148. 1040. 87105. 89431.
 92236. 60126. 7055. 29406. 23602. 14701. 46544. 43402. 7090. 92253.
 32303. 37211. 98226. 60098. 76117. 60090. 29483. 71901. 80112. 43130.
 88001. 35244. 75034. 95687. 84107. 53186. 93309. 33068. 45373. 78415.
 90278. 32839. 7050. 70601. 60035. 11550. 46060. 55124. 29464. 48310.
 54703. 78577. 59102. 97030. 37421. 83642. 92307. 60440. 33801. 55369.
 95695. 77489. 77581. 94403. 49505. 93277. 66212. 92592. 92399. 2151.
 77301. 60477. 52001. 48127. 87505. 28601. 60188. 56301. 33161. 46226.
 33317. 34952. 29730. 79762. 53214. 91911. 66502. 16602. 80229. 61821.
 47401. 71854. 78539. 77520. 46142. 90712. 2895. 54880. 76021. 98042.

```
 74012. 33023. 33021. 77536. 67212. 78501. 52240. 83704.  2920. 61032.
 77642. 95610. 75056. 98052. 32114. 86442. 46368. 58103. 46514. 91776.
 45011. 33063. 30328. 44060. 73505. 23666. 13440. 54601. 83501. 39401.
 94526. 48858. 84321.  6708. 30605.  4240. 61832. 85323. 30062. 85364.
 54401. 99301. 60302. 32503. 77573. 20877. 84043. 35401. 92553. 40324.
 80538. 85224. 59601. 63122. 76706. 48066. 60423. 18018. 55113. 68801.
 55125. 48237. 72756. 88101. 33458. 93101. 75104. 68701. 84020. 48104.
 91941. 83201. 49423.  6460. 60089. 92630. 96003. 95928. 13501. 72032.
 82001. 42301. 83605. 70065.  3060. 38134. 94061. 37087. 93454. 60016.
 98632. 37075. 50701.  2138. 60067.  1915. 97405. 93030. 98059. 60025.
 33445. 80022. 77590. 27893. 87124. 27534. 98208. 90640. 92020. 33407.
 61761. 60174. 93010. 97123. 91505. 95351. 67846.  8401. 80501. 95616.
 26003. 95037.  7011. 53081. 30344. 57701.  1810. 34741.  6484.  6810.
 52302. 32771. 78666. 80634. 76063. 44035. 83301. 63301. 33134. 60441.
  1752. 20707. 77803. 71603. 57401. 21740.  7017. 60004. 60543. 77705.
 55433. 92672. 94568. 93405. 72762. 95240. 77571. 45040. 30188.]

Region:['South' 'West' 'Central' 'East']

Product ID:['FUR-BO-10001798' 'FUR-CH-10000454' 'OFF-LA-10000240' …
 71.16199999999998 14.4648 12.672]

Category:['Furniture' 'Office Supplies' 'Technology']

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

Product Name:['Bush Somerset Collection Bookcase'
 'Hon Deluxe Fabric Upholstered Stacking Chairs, Rounded Back'
 'Self-Adhesive Address Labels for Typewriters by Universal' …
 'Hoover Commercial Lightweight Upright Vacuum' 'LG G2'
 'Eldon Jumbo ProFile Portable File Boxes Graphite/Black']

Sales:[261.96 731.94  14.62 …   8.78 376.74  44.43]

Quantity:[ 2.  3.  5.  7.  4.  6.  9.  1.  8. 14. 11. 13. 10. 12.]

Discount:[0.   0.45 0.2  0.8  0.3  0.5  0.7  0.6  0.32 0.1  0.4  0.15]

Profit:[ 41.9136 219.582    6.8714 …   2.99   130.2885  18.6606]

order_year:[2016. 2015. 2014. 2017.]

order_month:[11.  6. 10.  4. 12.  5.  8.  7.  9.  1.  3.  2.]

order_date:[ 8. 12. 11.  9. 15.  5. 22. 13. 27. 16. 25. 17. 19. 10. 20. 18. 24.
```

```
30.
  4. 14. 26.  3. 28. 29.  1. 23.  2.  6.  7. 31. 21.]

Ship_year:[2016. 2015. 2014. 2017. 2018.]

Ship_month:[11.  6. 10.  4. 12.  5.  9.  7.  1.  3.  8.  2.]

Ship_date:[11. 16. 18. 14. 20. 10. 26. 15.  1. 13. 30. 21. 23. 31. 22. 25. 17.
5.
  6.  2.  8.  4. 28. 27. 12. 19.  7.  9.  3. 24. 29.]
```

[70]:
```
df.columns #prints all columns
```

[70]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit', 'order_year',
       'order_month', 'order_date', 'Ship_year', 'Ship_month', 'Ship_date'],
      dtype='object')
```

3 Dropping Irrelevant Columns

[71]:
```
#droping columns that are not required for analysis
df.drop(columns=['Row ID', 'Order ID',
       'Customer ID', 'Customer Name',
       'Postal Code', 'Product ID','Order Date','Ship␣
 ↪Date'],axis=1,inplace=True)
```

[72]:
```
df.sample(5) #display random 5 samples
```

[72]:

	Ship Mode	Segment	Country	City	State	Region	Category	Sub-Category	Product Name	Sales	Quantity	Discount	\
8516	Same Day	Consumer	United States	Seattle	Washington	West	Office Supplies	Paper					
1680	First Class	Consumer	United States	Philadelphia	Pennsylvania	East	Office Supplies	Appliances					
2667	First Class	Corporate	United States	Lakewood	Ohio	East	Furniture	Furnishings					
8800	Second Class	Consumer	United States	North Miami	Florida	South	Office Supplies	Supplies					
9439	First Class	Home Office	United States	New York City	New York	East	Office Supplies	Storage					

```
8516                                Xerox 1942   48.940       1.0       0.0
1680           Avanti 4.4 Cu. Ft. Refrigerator  434.352       3.0       0.2
2667                Nu-Dell Leatherette Frames   45.888       4.0       0.2
8800         Acme Serrated Blade Letter Opener    7.632       3.0       0.2
9439  Sensible Storage WireTech Storage Systems   70.980       1.0       0.0

       Profit  order_year  order_month  order_date  Ship_year  Ship_month  \
8516  24.4700      2014.0          9.0        13.0     2014.0         9.0
1680  43.4352      2015.0         12.0        19.0     2015.0        12.0
2667   9.1776      2017.0         10.0         9.0     2017.0        10.0
8800  -1.8126      2014.0          8.0        22.0     2014.0         8.0
9439   3.5490      2017.0          9.0        14.0     2017.0         9.0

      Ship_date
8516       13.0
1680       20.0
2667       11.0
8800       24.0
9439       16.0
```

4 Descriptive Analysis:

5 1) Generate descriptive statistics for key variables like Sales, Quantity, Discount, and Profit.

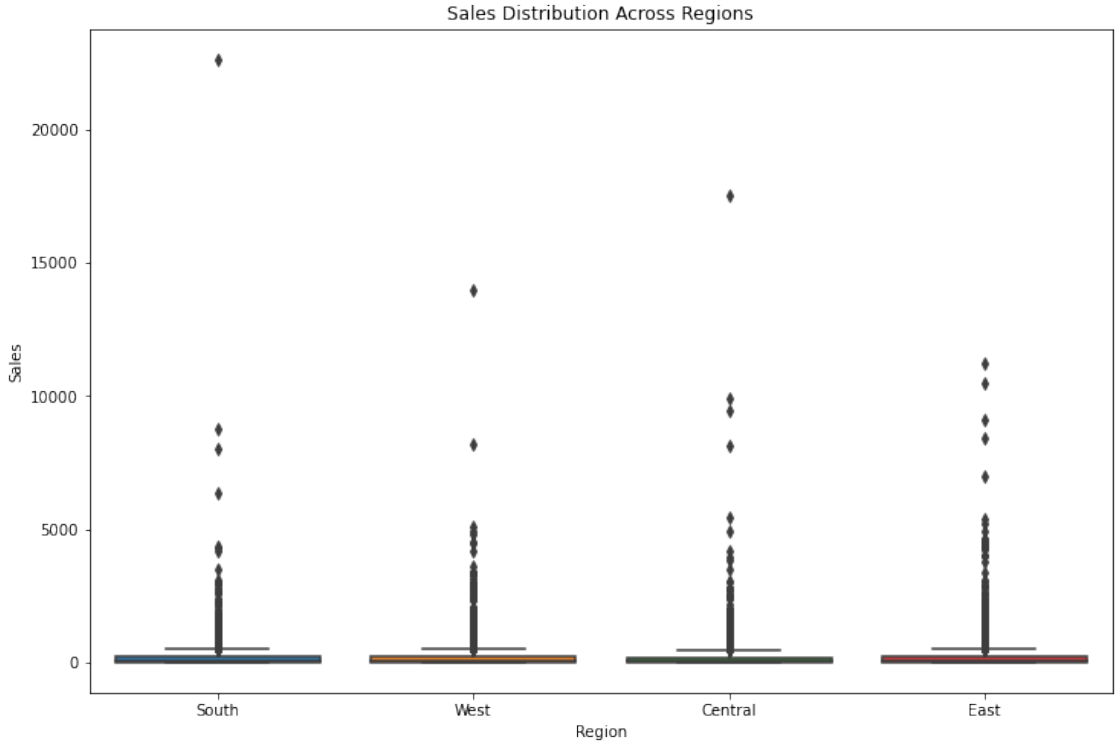
[73]:
```
df[["Sales","Quantity","Discount","Profit"]].describe()
```

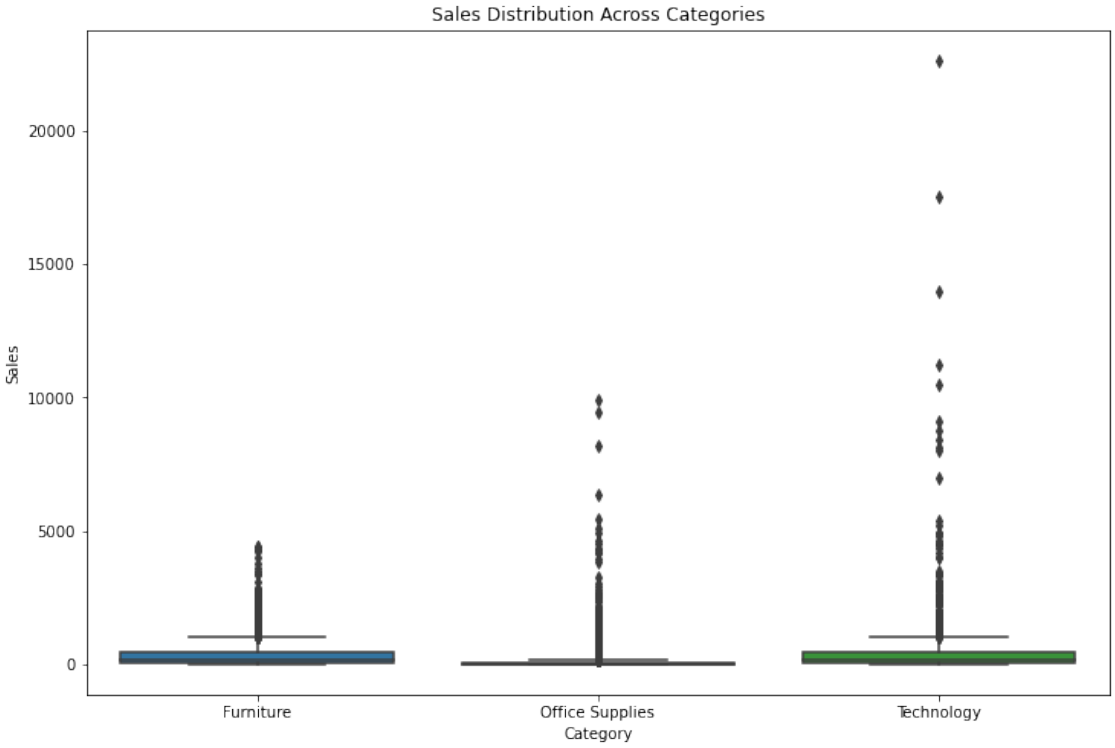
[73]:
```
              Sales     Quantity     Discount       Profit
count   9844.000000  9844.000000  9844.000000  9844.000000
mean     230.386939     3.790431     0.155312    28.970794
std      626.024933     2.224033     0.205817   235.713664
min        0.444000     1.000000     0.000000 -6599.978000
25%       17.310000     2.000000     0.000000     1.757325
50%       54.804000     3.000000     0.200000     8.709500
75%      209.970000     5.000000     0.200000    29.460650
max    22638.480000    14.000000     0.800000  8399.976000
```

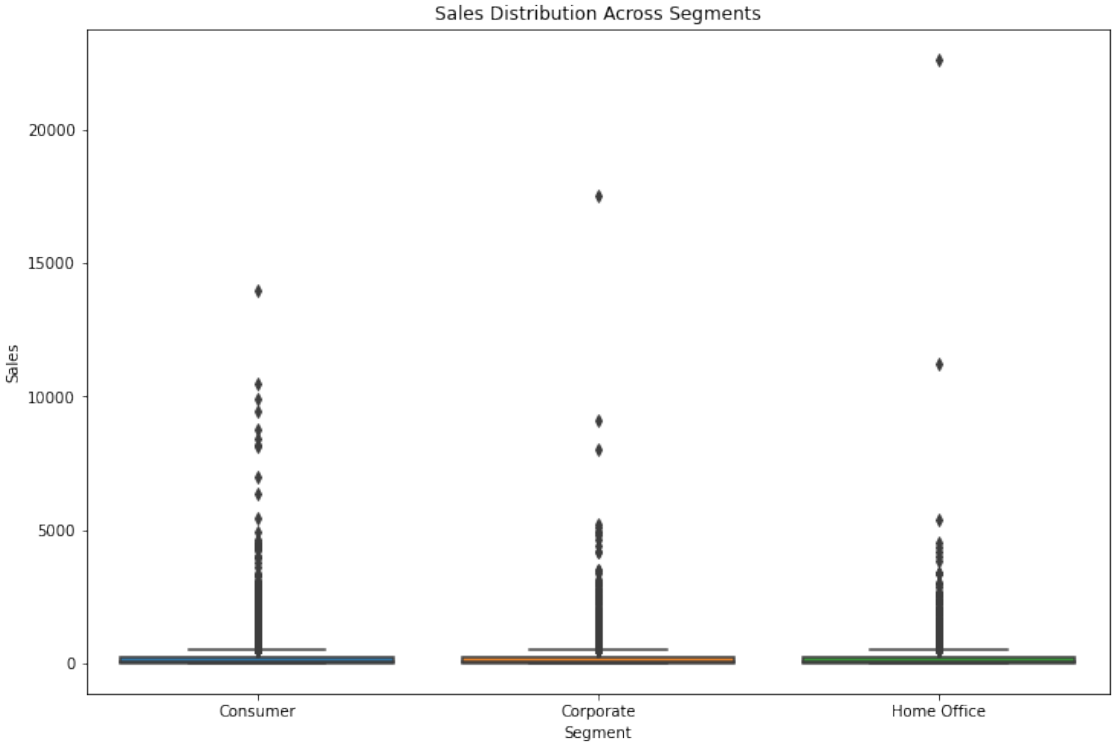
6 2) Create visualizations that provide insights into the distribution of sales across different regions, categories, and segments.

[74]:
```
plt.figure(figsize=(12, 8))
sns.boxplot(x='Region', y='Sales', data=df)
plt.title('Sales Distribution Across Regions')
plt.xlabel('Region')
```

```
plt.ylabel('Sales')
plt.show()

plt.figure(figsize=(12, 8))
sns.boxplot(x='Category', y='Sales', data=df)
plt.title('Sales Distribution Across Categories')
plt.xlabel('Category')
plt.ylabel('Sales')
plt.show()

plt.figure(figsize=(12, 8))
sns.boxplot(x='Segment', y='Sales', data=df)
plt.title('Sales Distribution Across Segments')
plt.xlabel('Segment')
plt.ylabel('Sales')
plt.show()
```

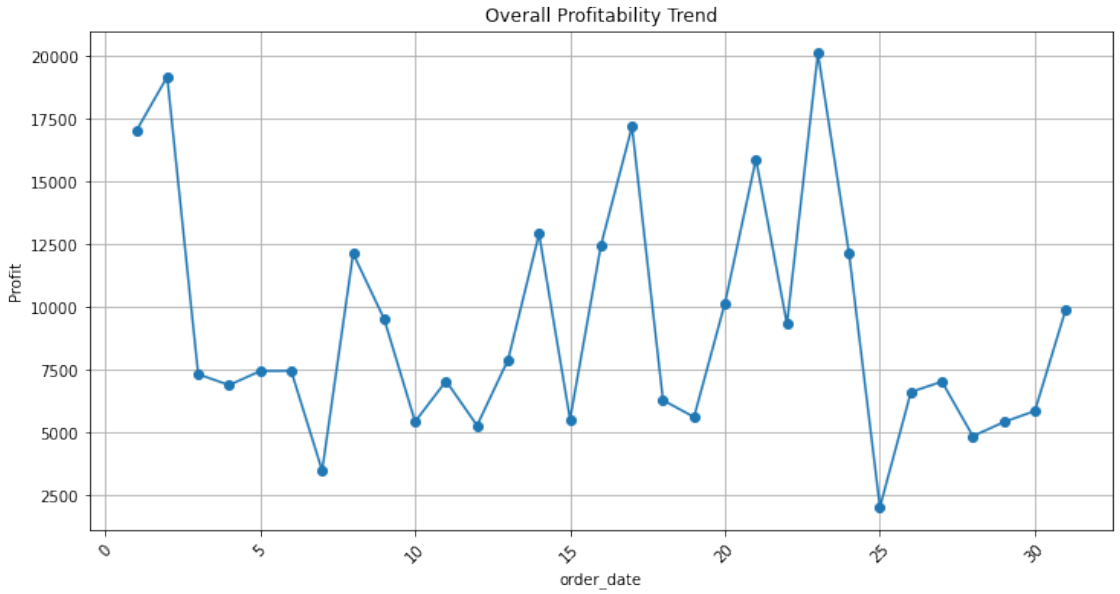



Sales Distribution Across Categories
Sales
20000
15000
10000
5000
0
Furniture
Office Supplies
Technology
Category

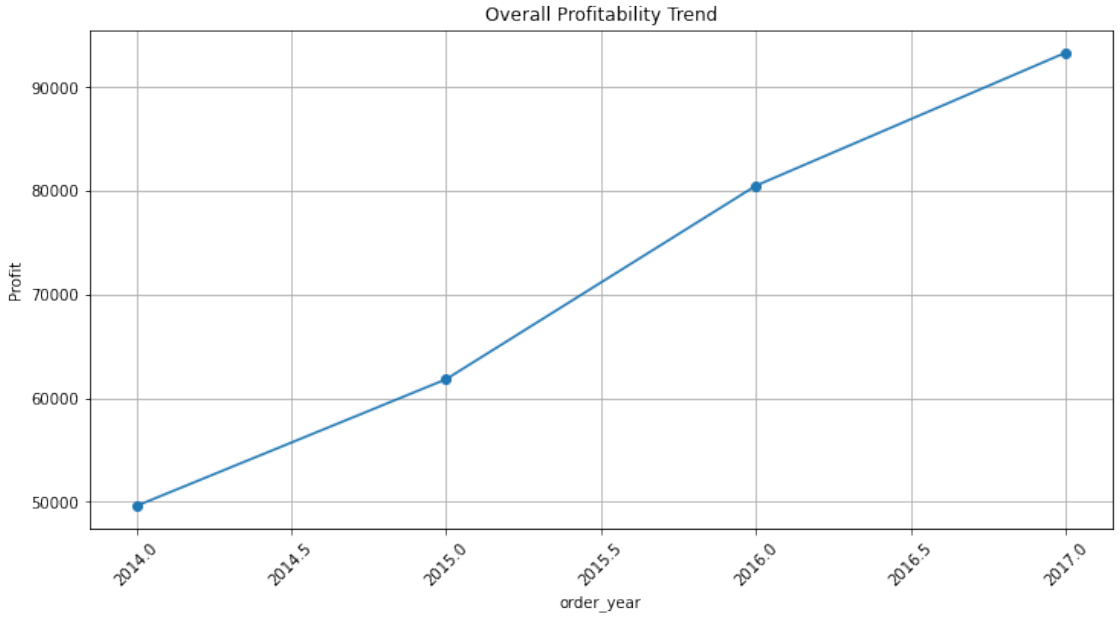

Sales Distribution Across Segments
Sales
20000
15000
10000
5000
0
Consumer
Corporate
Home Office
Segment

7 3) Analyze the overall profitability trend.

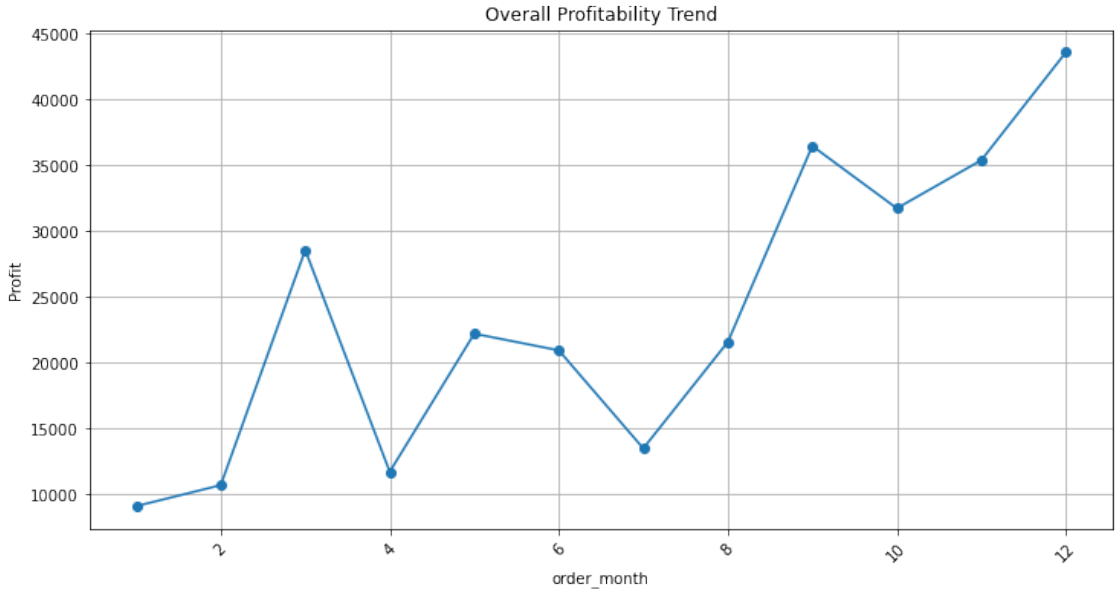
[75]:
```
profitability_trend = df.groupby('order_date')['Profit'].sum().reset_index()
plt.figure(figsize=(12, 6))
plt.plot(profitability_trend['order_date'], profitability_trend['Profit'],␣
 ↪marker='o', linestyle='-')
plt.title('Overall Profitability Trend')
plt.xlabel('order_date')
plt.ylabel('Profit')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



[76]:
```
profitability_trend = df.groupby('order_year')['Profit'].sum().reset_index()
plt.figure(figsize=(12, 6))
plt.plot(profitability_trend['order_year'], profitability_trend['Profit'],␣
 ↪marker='o', linestyle='-')
plt.title('Overall Profitability Trend')
plt.xlabel('order_year')
plt.ylabel('Profit')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



[77]:
```
profitability_trend = df.groupby('order_month')['Profit'].sum().reset_index()
plt.figure(figsize=(12, 6))
plt.plot(profitability_trend['order_month'], profitability_trend['Profit'],␣
 ↪marker='o', linestyle='-')
plt.title('Overall Profitability Trend')
plt.xlabel('order_month')
plt.ylabel('Profit')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



8 4) Identify the least profitable products and categories.

[78]:
```
least_profitable_products = df.groupby("Product Name")["Profit"].mean().
  ↪nsmallest(5)
print("Least Profitable Products:")
print(least_profitable_products)
```

```
Least Profitable Products:
Product Name
Cubify CubeX 3D Printer Triple Head Print               -3839.990400
Cubify CubeX 3D Printer Double Head Print               -2959.990133
Cisco TelePresence System EX90 Videoconferencing Unit   -1811.078400
Lexmark MX611dhe Monochrome Laser Printer               -1147.493250
Zebra GK420t Direct Thermal/Thermal Transfer Printer     -938.280000
Name: Profit, dtype: float64
```

[79]:
```
least_profitable_categories=df.groupby("Category")["Profit"].mean().nsmallest(3)
print("Least Profitable Categories:")
print(least_profitable_categories)
```

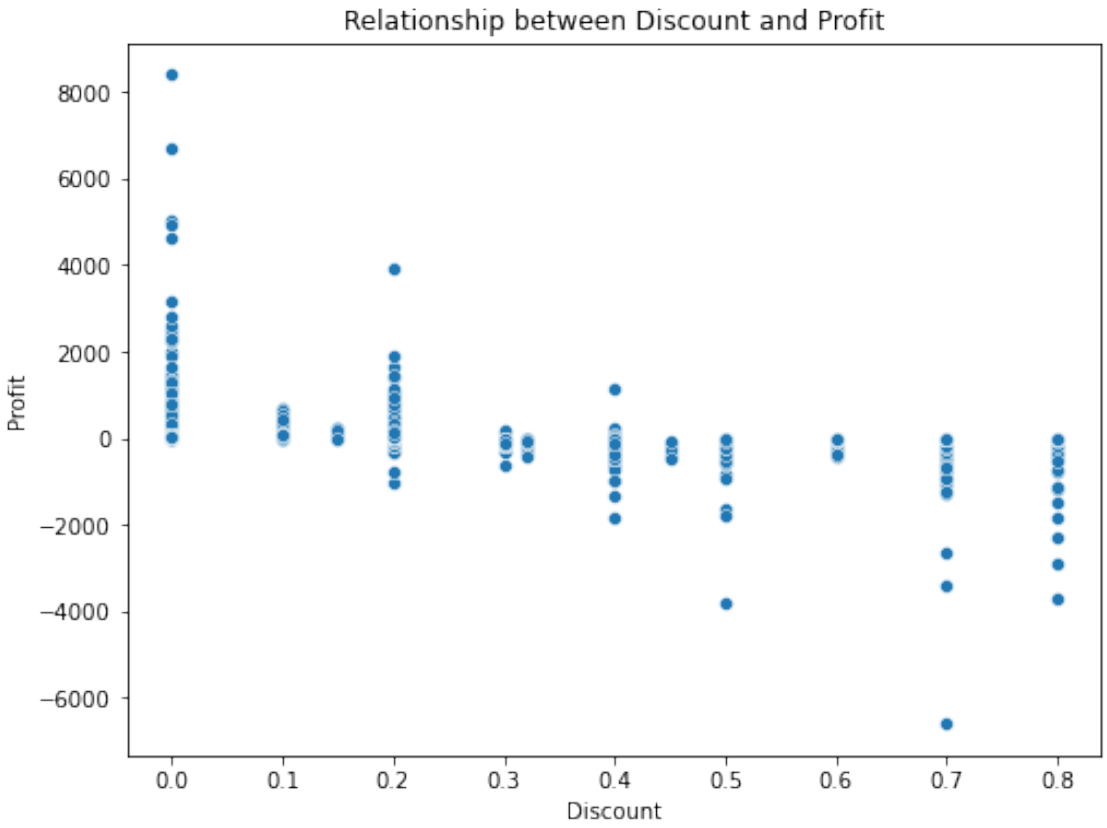
```
Least Profitable Categories:
Category
Furniture            8.299843
Office Supplies     20.703890
Technology          79.402216
Name: Profit, dtype: float64
```

9 5) Explore the relationship between discount and profit.

[80]:
```
avg_profit_by_discount = df.groupby("Discount")["Profit"].mean()
avg_profit_by_discount
```

[80]:
```
Discount
0.00     67.219086
0.10     96.215291
0.15     27.288298
0.20     24.698622
0.30    -46.146675
0.32    -88.560656
0.40   -112.501221
0.45   -226.646464
0.50   -315.772908
0.60    -43.118376
0.70    -96.478232
0.80   -100.906834
Name: Profit, dtype: float64
```

[81]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Discount', y='Profit', data=df)
plt.title('Relationship between Discount and Profit')
plt.xlabel('Discount')
plt.ylabel('Profit')
plt.show()
```



10 Outlier Analysis

11 Quantity Outlier Analysis

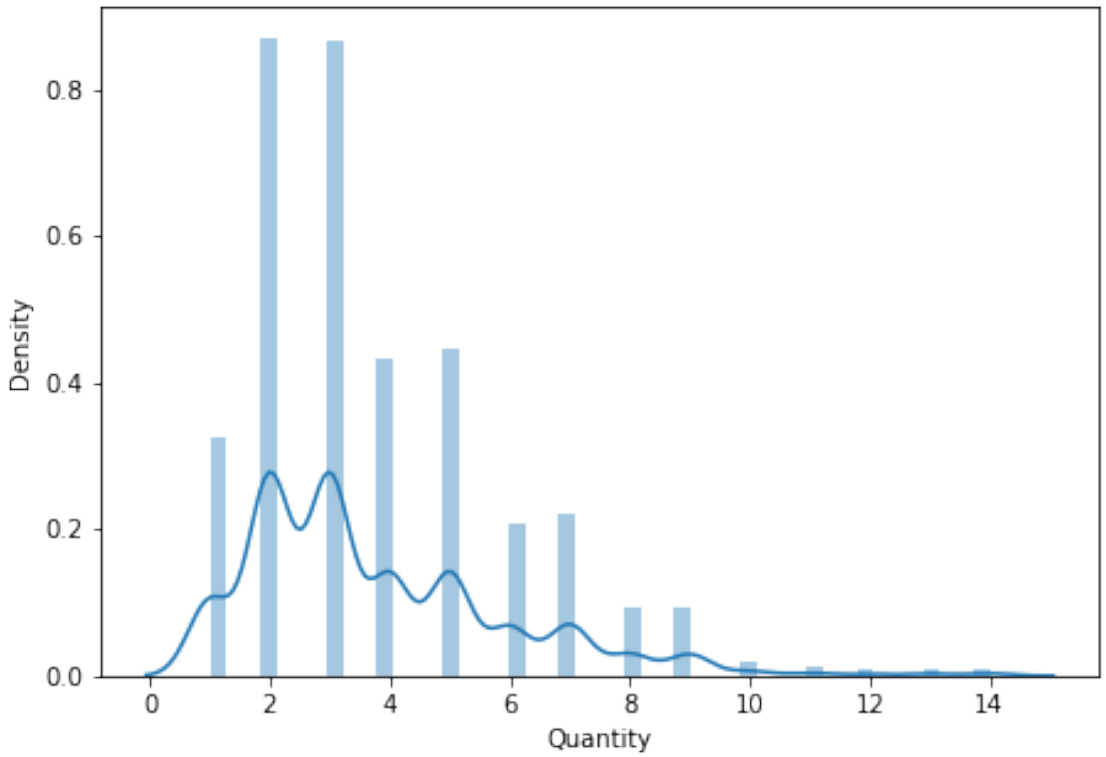
[82]:
```
#unique values
df["Quantity"].unique()
```

[82]:
```
array([ 2.,  3.,  5.,  7.,  4.,  6.,  9.,  1.,  8., 14., 11., 13., 10.,
       12.])
```

[83]:
```
#distribution graph for quantity
plt.figure(figsize=(16,5))
plt.subplot(1,2,2)
sns.distplot(df["Quantity"])
plt.show()
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
```

```
histograms).
  warnings.warn(msg, FutureWarning)
```



[84]:
```
#right skew
df["Quantity"].skew()
```

[84]:
```
1.272389553355905
```

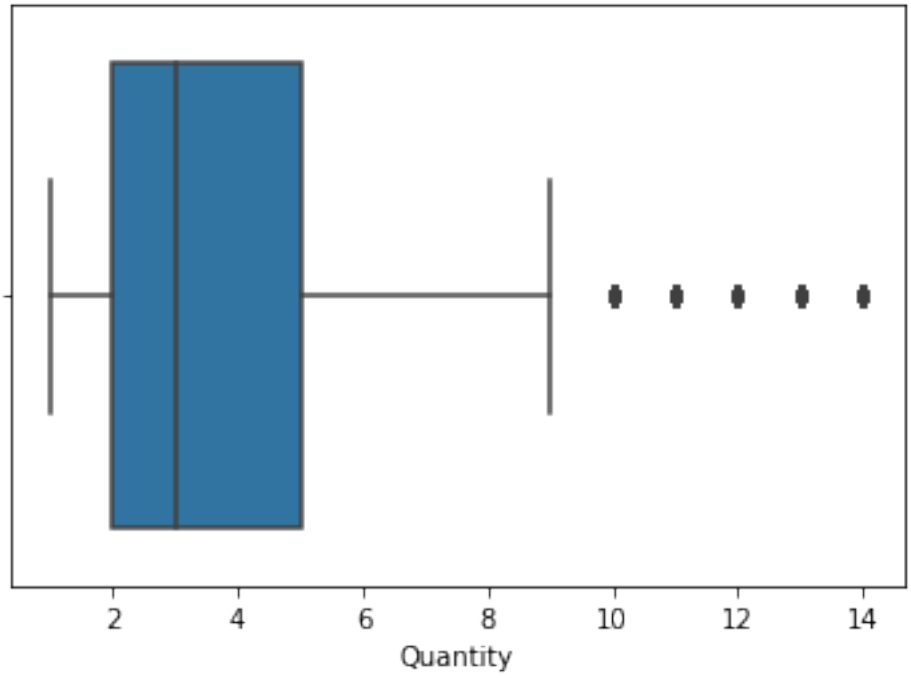
[85]:
```
df["Quantity"].describe()
```

[85]:
```
count    9844.000000
mean        3.790431
std         2.224033
min         1.000000
25%         2.000000
50%         3.000000
75%         5.000000
max        14.000000
Name: Quantity, dtype: float64
```

[86]:
```
#box plot
sns.boxplot(df["Quantity"])
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[86]: <AxesSubplot:xlabel='Quantity'>



[87]:
```
#Percentile Method
low=df["Quantity"].quantile(0.05)
high=df["Quantity"].quantile(0.95)
low,high
```

[87]: (1.0, 8.0)

[88]:
```
df[df["Quantity"]>high]
```

[88]:

	Ship Mode	Segment	Country	City	State	\
10	Standard Class	Consumer	United States	Los Angeles	California	
37	Standard Class	Home Office	United States	Houston	Texas	
102	Second Class	Consumer	United States	Columbus	Ohio	
111	First Class	Consumer	United States	Wilmington	Delaware	
124	Standard Class	Consumer	United States	Roseville	California	
…	…	…	…	…	…	

```
9801  Standard Class     Consumer  United States  San Francisco  California
9802  Standard Class     Consumer  United States        Anaheim  California
9839  Standard Class  Home Office  United States    Los Angeles  California
9844  Standard Class     Consumer  United States     Long Beach    New York
9860    Second Class     Consumer  United States        Atlanta     Georgia

      Region         Category Sub-Category  \
10      West        Furniture       Tables
37   Central  Office Supplies    Envelopes
102     East  Office Supplies    Fasteners
111     East  Office Supplies    Envelopes
124     West        Furniture  Furnishings
…       …              …           …
9801    West       Technology  Accessories
9802    West  Office Supplies      Storage
9839    West  Office Supplies      Binders
9844    East  Office Supplies       Labels
9860   South  Office Supplies        Paper

                                          Product Name     Sales  Quantity  \
10             Chromcraft Rectangular Conference Tables  1706.184       9.0
37   #10-4 1/8" x 9 1/2" Premium Diagonal Seam Enve…   113.328       9.0
102            OIC Colored Binder Clips, Assorted Sizes    40.096      14.0
111        Globe Weis Peel & Seel First Class Envelopes   115.020       9.0
124                         Longer-Life Soft White Bulbs    43.120      14.0
…                                                     …       …       …
9801  Memorex Mini Travel Drive 16 GB USB 2.0 Flash …   223.580      14.0
9802        Carina Mini System Audio Rack, Model AR050B   998.820       9.0
9839                  Ibico Recycled Linen-Style Covers   437.472      14.0
9844                     Self-Adhesive Removable Labels    31.500      10.0
9860                 Wirebound Message Book, 4 per Page    48.870       9.0

      Discount    Profit  order_year  order_month  order_date  Ship_year  \
10         0.2   85.3092      2014.0          6.0         9.0     2014.0
37         0.2   35.4150      2015.0         12.0        27.0     2015.0
102        0.2   14.5348      2014.0          8.0        25.0     2014.0
111        0.0   51.7590      2016.0          6.0        12.0     2016.0
124        0.0   20.6976      2016.0         10.0        13.0     2016.0
…        …       …          …           …          …        …
9801       0.0   87.1962      2017.0         11.0        24.0     2017.0
9802       0.0   29.9646      2014.0         12.0        28.0     2015.0
9839       0.2  153.1152      2016.0         12.0         6.0     2016.0
9844       0.0   15.1200      2015.0          5.0        17.0     2015.0
9860       0.0   23.9463      2017.0         11.0        25.0     2017.0

      Ship_month  Ship_date
10           6.0       14.0
```

```
37          12.0       31.0
102          8.0       27.0
111          6.0       15.0
124         10.0       19.0
…            …          …
9801        11.0       30.0
9802         1.0        3.0
9839        12.0       10.0
9844         5.0       23.0
9860        11.0       29.0

[418 rows x 19 columns]
```

[89]:
```
#did because felt outliers are valid (so used capping)
```

[90]:
```
#capping outliers
df["Quantity"]=np.where(df["Quantity"]>high,high,np.
 ↪where(df["Quantity"]<low,low,df["Quantity"]))
```

[91]:
```
df.shape
```

[91]:
```
(9844, 19)
```

[92]:
```
df[df["Quantity"]>=5]
```

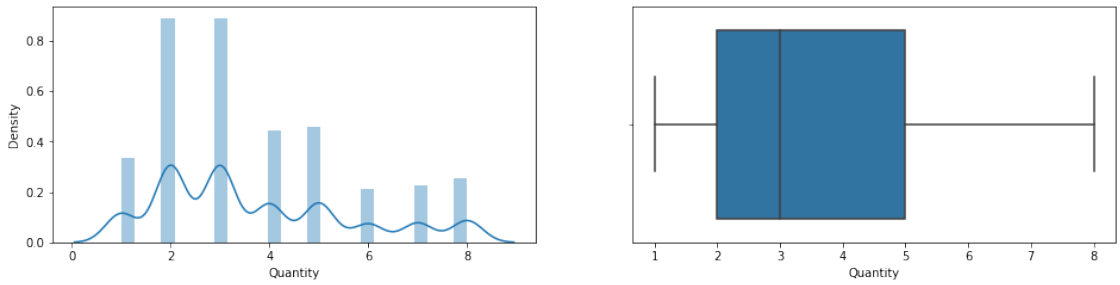
[92]:
```
           Ship Mode    Segment        Country             City       State  \
3     Standard Class   Consumer  United States  Fort Lauderdale     Florida
5     Standard Class   Consumer  United States      Los Angeles  California
7     Standard Class   Consumer  United States      Los Angeles  California
9     Standard Class   Consumer  United States      Los Angeles  California
10    Standard Class   Consumer  United States      Los Angeles  California
…                …          …              …                …          …
9860    Second Class   Consumer  United States          Atlanta     Georgia
9861  Standard Class   Consumer  United States      Los Angeles  California
9864  Standard Class   Consumer  United States          Detroit    Michigan
9869  Standard Class   Consumer  United States          Detroit    Michigan
9870    Second Class  Corporate  United States  Fort Lauderdale     Florida

       Region         Category Sub-Category  \
3       South        Furniture       Tables
5        West        Furniture  Furnishings
7        West       Technology       Phones
9        West  Office Supplies   Appliances
10       West        Furniture       Tables
…         …                …           …
9860    South  Office Supplies        Paper
9861     West  Office Supplies        Paper
```

```
9864  Central  Office Supplies      Binders
9869  Central  Office Supplies      Binders
9870    South  Office Supplies   Appliances

                                          Product Name      Sales  Quantity  \
3         Bretford CR4500 Series Slim Rectangular Table   957.5775       5.0
5     Eldon Expressions Wood and Plastic Desk Access…    48.8600       7.0
7                         Mitel 5320 IP Phone VoIP phone   907.1520       6.0
9                       Belkin F5C206VTEL 6 Outlet Surge   114.9000       5.0
10             Chromcraft Rectangular Conference Tables  1706.1840       8.0
…                                                   …       …       …
9860                Wirebound Message Book, 4 per Page    48.8700       8.0
9861                                        Xerox 19   154.9000       5.0
9864  Wilson Jones Turn Tabs Binder Tool for Ring Bi…    24.1000       5.0
9869      Wilson Jones 1" Hanging DublLock Ring Binders    26.4000       5.0
9870                 Hoover Upright Vacuum With Dirt Cup  1158.1200       5.0

      Discount    Profit  order_year  order_month  order_date  Ship_year  \
3         0.45 -383.0310      2015.0         10.0        11.0     2015.0
5         0.00   14.1694      2014.0          6.0         9.0     2014.0
7         0.20   90.7152      2014.0          6.0         9.0     2014.0
9         0.00   34.4700      2014.0          6.0         9.0     2014.0
10        0.20   85.3092      2014.0          6.0         9.0     2014.0
…          …       …           …           …          …          …
9860      0.00   23.9463      2017.0         11.0        25.0     2017.0
9861      0.00   69.7050      2017.0          1.0        14.0     2017.0
9864      0.00   11.0860      2016.0          9.0         1.0     2016.0
9869      0.00   12.6720      2016.0          9.0         1.0     2016.0
9870      0.20  130.2885      2017.0         11.0        11.0     2017.0

      Ship_month  Ship_date
3           10.0       18.0
5            6.0       14.0
7            6.0       14.0
9            6.0       14.0
10           6.0       14.0
…           …          …
9860        11.0       29.0
9861         1.0       19.0
9864         9.0        5.0
9869         9.0        5.0
9870        11.0       16.0

[3053 rows x 19 columns]
```

[93]:
```
#distribution and box plot for quantity
plt.figure(figsize=(16,8))
```

```
plt.subplot(2,2,1)
sns.distplot(df["Quantity"])

plt.subplot(2,2,2)
sns.boxplot(df["Quantity"])

plt.show()
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



[94]:
```
#right skew
df["Quantity"].skew()
```

[94]: 0.7107622703947052

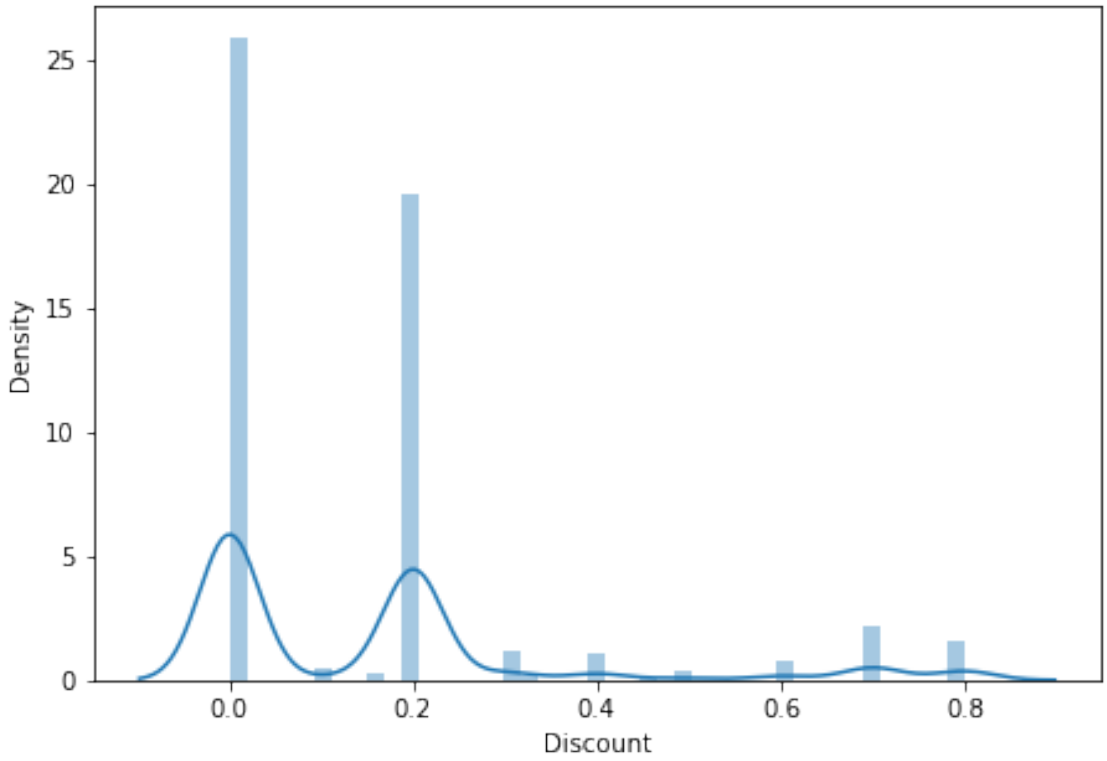
12 Disount Outlier Analysis

[95]:
```
#unique values in discount
df["Discount"].unique()
```

[95]:
```
array([0.  , 0.45, 0.2 , 0.8 , 0.3 , 0.5 , 0.7 , 0.6 , 0.32, 0.1 , 0.4 ,
       0.15])
```

[96]:
```
#distribution graph for discount
plt.figure(figsize=(16,5))
plt.subplot(1,2,2)
sns.distplot(df["Discount"])
plt.show()
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



[97]:
```
#right skew
df["Discount"].skew()
```

[97]:
```
1.6922053394888577
```

[98]:
```
df["Discount"].describe()
```

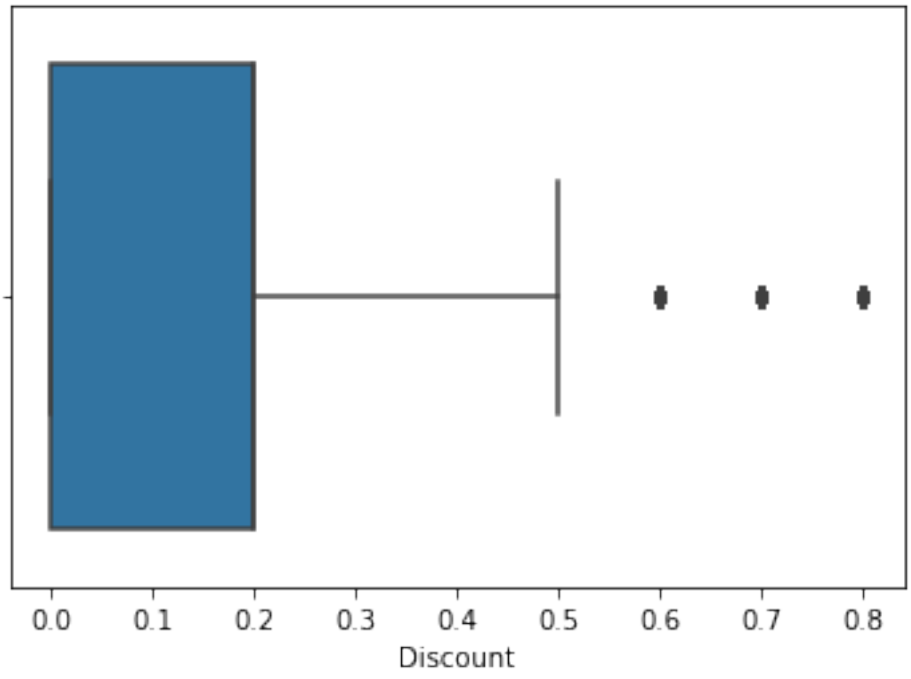
[98]:
```
count    9844.000000
mean        0.155312
```

```
std         0.205817
min         0.000000
25%         0.000000
50%         0.200000
75%         0.200000
max         0.800000
Name: Discount, dtype: float64
```

[99]:
```
#boxplot
sns.boxplot(df["Discount"])
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[99]: <AxesSubplot:xlabel='Discount'>



[100]:
```
per25=df["Discount"].quantile(0.25)
per75=df["Discount"].quantile(0.75)
per25,per75
```

[100]: (0.0, 0.2)

[101]:
```
#IQR(method)
iqr=per75-per25
low=per25-1.5*iqr
high=per75+1.5*iqr
low,high
```

[101]:
```
(-0.30000000000000004, 0.5)
```

[102]:
```
df[df["Discount"]>high]
```

[102]:
```
           Ship Mode      Segment        Country          City          State  \
14    Standard Class  Home Office  United States    Fort Worth          Texas
15    Standard Class  Home Office  United States    Fort Worth          Texas
28    Standard Class     Consumer  United States  Philadelphia   Pennsylvania
32    Standard Class     Consumer  United States  Philadelphia   Pennsylvania
36       First Class    Corporate  United States    Richardson          Texas
…                …            …              …             …             …
9737     First Class  Home Office  United States     Cleveland           Ohio
9763  Standard Class     Consumer  United States    Carrollton          Texas
9780  Standard Class    Corporate  United States         Bryan          Texas
9781  Standard Class  Home Office  United States         Akron           Ohio
9855  Standard Class     Consumer  United States       Phoenix        Arizona

       Region         Category Sub-Category  \
14    Central  Office Supplies   Appliances
15    Central  Office Supplies      Binders
28       East  Office Supplies      Binders
32       East  Office Supplies      Binders
36    Central        Furniture  Furnishings
…       …               …           …
9737     East  Office Supplies      Binders
9763  Central        Furniture  Furnishings
9780  Central  Office Supplies      Binders
9781     East  Office Supplies      Binders
9855     West  Office Supplies      Binders

                                              Product Name    Sales  Quantity  \
14    Holmes Replacement Filter for HEPA Air Cleaner…   68.810       5.0
15     Storex DuraTech Recycled Plastic Frosted Binders    2.544       3.0
28    Avery Recycled Flexi-View Covers for Binding S…    9.618       2.0
32    Acco Pressboard Covers with Storage Hooks, 14 …    6.858       6.0
36    Electrix Architect's Clamp-On Swing Arm Lamp, …  190.920       5.0
…                                                  …       …       …
9737  Wilson Jones Clip & Carry Folder Binder Tool f…    8.700       5.0
9763  GE General Use Halogen Bulbs, 100 Watts, 1 Bul…   25.128       3.0
9780  GBC Pre-Punched Binding Paper, Plastic, White,…   22.386       7.0
9781                   Acco Expandable Hanging Binders    5.742       3.0
```

```
9855                            Plastic Binding Combs   18.180       4.0

      Discount    Profit  order_year  order_month  order_date  Ship_year  \
14         0.8 -123.8580      2015.0         11.0        22.0     2015.0
15         0.8   -3.8160      2015.0         11.0        22.0     2015.0
28         0.7   -7.0532      2015.0          9.0        17.0     2015.0
32         0.7   -5.7150      2015.0          9.0        17.0     2015.0
36         0.6 -147.9630      2016.0         12.0         8.0     2016.0
…          …        …           …            …          …          …
9737       0.7   -6.3800      2017.0          4.0        20.0     2017.0
9763       0.6   -6.9102      2014.0         11.0        12.0     2014.0
9780       0.8  -35.8176      2016.0          3.0        15.0     2016.0
9781       0.7   -4.5936      2014.0         11.0        24.0     2014.0
9855       0.7  -13.9380      2017.0          9.0        19.0     2017.0

      Ship_month  Ship_date
14          11.0       26.0
15          11.0       26.0
28           9.0       21.0
32           9.0       21.0
36          12.0       10.0
…            …          …
9737         4.0       21.0
9763        11.0       18.0
9780         3.0       19.0
9781        11.0       30.0
9855         9.0       25.0

[834 rows x 19 columns]
```

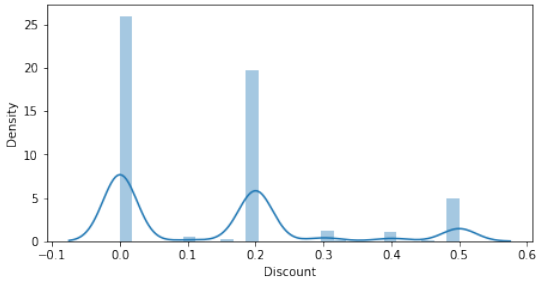
[103]:
```
#capping outliers with high and low values
df["Discount"]=np.where(df["Discount"]>high,high,np.
 ↪where(df["Discount"]<low,low,df["Discount"]))
```

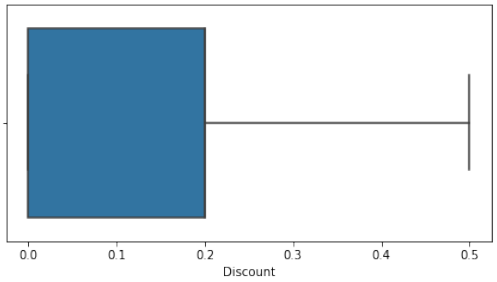
[104]:
```
df.shape
```

[104]:
```
(9844, 19)
```

[105]:
```
#distribution and box plot for discount
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(df["Discount"])
plt.subplot(2,2,2)
sns.boxplot(df["Discount"])

plt.show()
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```





[106]:
```
df.shape
```

[106]:
```
(9844, 19)
```

[107]:
```
#right skew
df["Discount"].skew()
```

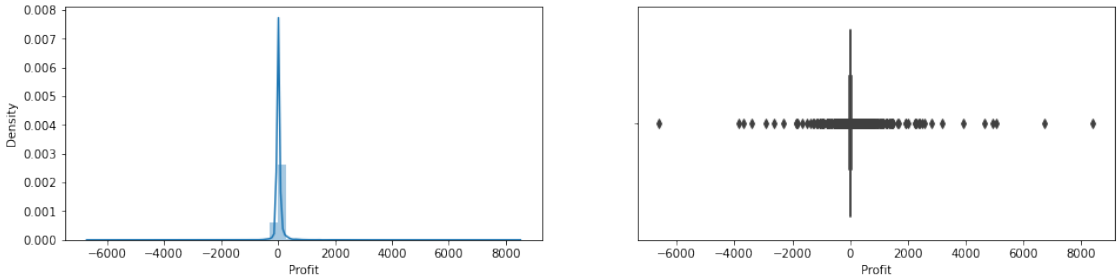
[107]:
```
0.9644235759205968
```

13 Profit Outlier Analysis

[108]:
```
#distribution and boxplot graph for profit
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(df["Profit"])
plt.subplot(2,2,2)
sns.boxplot(df["Profit"])
plt.show()
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
```

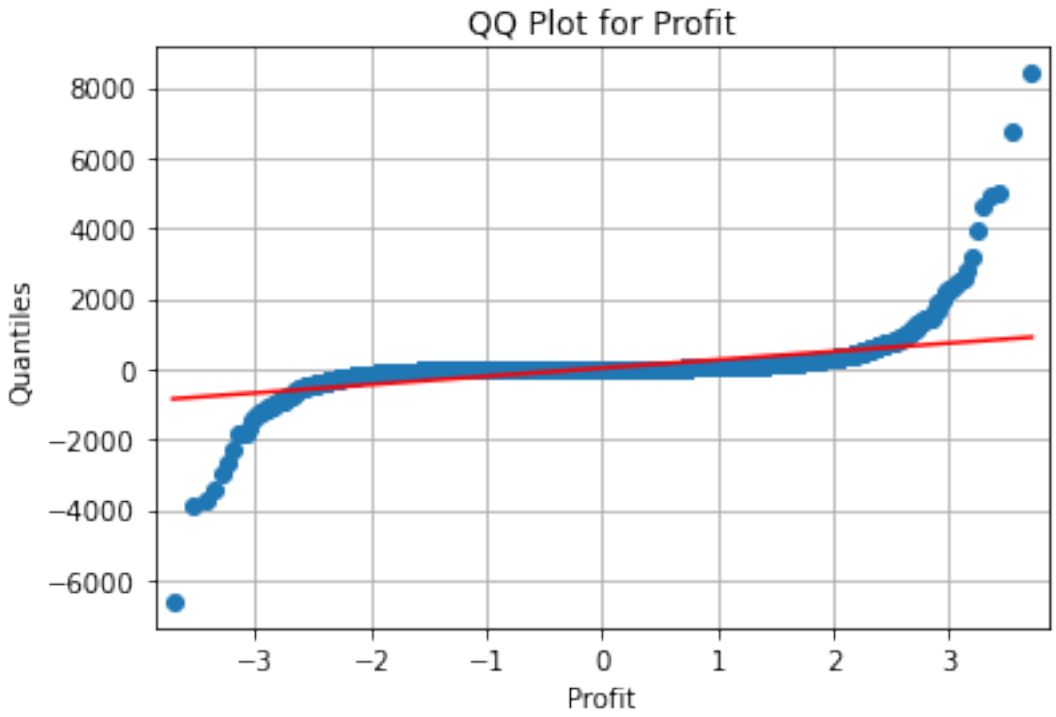
```
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



[109]:
```
#right skew
df["Profit"].skew()
```

[109]: 7.532178434791842

[110]:
```
data = df

# Select the column you want to analyze
column_name = 'Profit'
column_data = data[column_name]

# Create the QQ plot using statsmodels.api
sm.qqplot(column_data, line='s')  # 's' for straight reference line
plt.xlabel(column_name)
plt.ylabel('Quantiles')
plt.title('QQ Plot for ' + column_name)
plt.grid(True)
plt.show()
```



[111]:
```
df[df["Profit"]<=0]
```

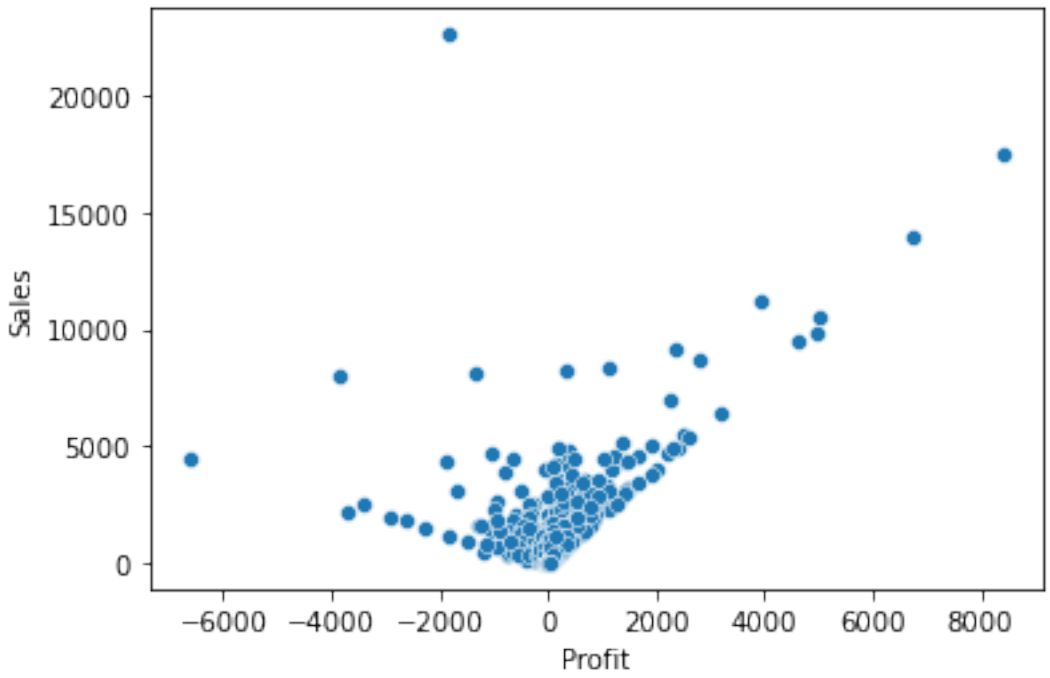
[111]:

	Ship Mode	Segment	Country	City	State	Region	Category	Sub-Category
3	Standard Class	Consumer	United States	Fort Lauderdale	Florida	South	Furniture	Tables
14	Standard Class	Home Office	United States	Fort Worth	Texas	Central	Office Supplies	Appliances
15	Standard Class	Home Office	United States	Fort Worth	Texas	Central	Office Supplies	Binders
23	Second Class	Consumer	United States	Philadelphia	Pennsylvania	East	Furniture	Chairs
27	Standard Class	Consumer	United States	Philadelphia	Pennsylvania	East	Furniture	Bookcases
…	…	…	…	…	…	…	…	…
9791	Standard Class	Consumer	United States	San Bernardino	California	West	Furniture	Bookcases
9797	Second Class	Corporate	United States	Los Angeles				
9822	First Class	Home Office	United States	Houston				
9837	Standard Class	Home Office	United States	Los Angeles				
9855	Standard Class	Consumer	United States	Phoenix				

```
9797    California     West        Furniture       Tables
9822         Texas  Central        Furniture    Bookcases
9837    California     West  Office Supplies    Fasteners
9855       Arizona     West  Office Supplies      Binders

                                          Product Name      Sales  Quantity  \
3         Bretford CR4500 Series Slim Rectangular Table   957.5775       5.0
14    Holmes Replacement Filter for HEPA Air Cleaner…    68.8100       5.0
15     Storex DuraTech Recycled Plastic Frosted Binders     2.5440       3.0
23                   Global Deluxe Stacking Chair, Gray    71.3720       2.0
27    Riverside Palais Royal Lawyers Bookcase, Royal…  3083.4300       7.0
…                                                  …        …       …
9791    O'Sullivan Living Dimensions 3-Shelf Bookcases   683.3320       4.0
9797       Hon 61000 Series Interactive Training Tables    71.0880       2.0
9822  Bush Heritage Pine Collection 5-Shelf Bookcase…   383.4656       4.0
9837           Alliance Big Bands Rubber Bands, 12/Pack    13.8600       7.0
9855                              Plastic Binding Combs    18.1800       4.0

      Discount     Profit  order_year  order_month  order_date  Ship_year  \
3         0.45  -383.0310      2015.0         10.0        11.0     2015.0
14        0.50  -123.8580      2015.0         11.0        22.0     2015.0
15        0.50    -3.8160      2015.0         11.0        22.0     2015.0
23        0.30    -1.0196      2017.0          7.0        16.0     2017.0
27        0.50 -1665.0522      2015.0          9.0        17.0     2015.0
…         …         …          …            …          …
9791      0.15   -40.1960      2015.0         11.0        13.0     2015.0
9797      0.20    -1.7772      2016.0          6.0         3.0     2016.0
9822      0.32   -67.6704      2015.0          3.0        19.0     2015.0
9837      0.00     0.0000      2016.0         12.0         6.0     2016.0
9855      0.50   -13.9380      2017.0          9.0        19.0     2017.0

      Ship_month  Ship_date
3           10.0       18.0
14          11.0       26.0
15          11.0       26.0
23           7.0       18.0
27           9.0       21.0
…            …          …
9791        11.0       17.0
9797         6.0        6.0
9822         3.0       22.0
9837        12.0       10.0
9855         9.0       25.0

[1896 rows x 19 columns]
```

[112]:
```
#scatterplot prfit vs sales
sns.scatterplot(x="Profit",y="Sales",data=df)
```

[112]: <AxesSubplot:xlabel='Profit', ylabel='Sales'>



[113]:
```
df[df["Profit"]<=0]
```

[113]:

	Ship Mode	Segment	Country	City	State	Region	Category	Sub-Category
3	Standard Class	Consumer	United States	Fort Lauderdale	Florida	South	Furniture	Tables
14	Standard Class	Home Office	United States	Fort Worth	Texas	Central	Office Supplies	Appliances
15	Standard Class	Home Office	United States	Fort Worth	Texas	Central	Office Supplies	Binders
23	Second Class	Consumer	United States	Philadelphia	Pennsylvania	East	Furniture	Chairs
27	Standard Class	Consumer	United States	Philadelphia				
…	…	…	…	…				
9791	Standard Class	Consumer	United States	San Bernardino				
9797	Second Class	Corporate	United States	Los Angeles				
9822	First Class	Home Office	United States	Houston				
9837	Standard Class	Home Office	United States	Los Angeles				
9855	Standard Class	Consumer	United States	Phoenix				

```
27    Pennsylvania     East        Furniture    Bookcases
…              …      …              …          …
9791    California     West        Furniture    Bookcases
9797    California     West        Furniture       Tables
9822         Texas  Central        Furniture    Bookcases
9837    California     West  Office Supplies    Fasteners
9855       Arizona     West  Office Supplies      Binders

                                          Product Name      Sales  Quantity  \
3          Bretford CR4500 Series Slim Rectangular Table   957.5775       5.0
14    Holmes Replacement Filter for HEPA Air Cleaner…    68.8100       5.0
15     Storex DuraTech Recycled Plastic Frosted Binders     2.5440       3.0
23                   Global Deluxe Stacking Chair, Gray    71.3720       2.0
27    Riverside Palais Royal Lawyers Bookcase, Royal…  3083.4300       7.0
…                                                  …        …       …
9791     O'Sullivan Living Dimensions 3-Shelf Bookcases   683.3320       4.0
9797       Hon 61000 Series Interactive Training Tables    71.0880       2.0
9822  Bush Heritage Pine Collection 5-Shelf Bookcase…   383.4656       4.0
9837           Alliance Big Bands Rubber Bands, 12/Pack    13.8600       7.0
9855                              Plastic Binding Combs    18.1800       4.0

      Discount     Profit  order_year  order_month  order_date  Ship_year  \
3         0.45  -383.0310      2015.0         10.0        11.0     2015.0
14        0.50  -123.8580      2015.0         11.0        22.0     2015.0
15        0.50    -3.8160      2015.0         11.0        22.0     2015.0
23        0.30    -1.0196      2017.0          7.0        16.0     2017.0
27        0.50 -1665.0522      2015.0          9.0        17.0     2015.0
…            …         …           …            …           …         …
9791      0.15   -40.1960      2015.0         11.0        13.0     2015.0
9797      0.20    -1.7772      2016.0          6.0         3.0     2016.0
9822      0.32   -67.6704      2015.0          3.0        19.0     2015.0
9837      0.00     0.0000      2016.0         12.0         6.0     2016.0
9855      0.50   -13.9380      2017.0          9.0        19.0     2017.0

      Ship_month  Ship_date
3           10.0       18.0
14          11.0       26.0
15          11.0       26.0
23           7.0       18.0
27           9.0       21.0
…            …         …
9791        11.0       17.0
9797         6.0        6.0
9822         3.0       22.0
9837        12.0       10.0
9855         9.0       25.0
```

[1896 rows x 19 columns]

[114]:
```
EPSILON = 1e-8
positive_profit = df['Profit'] + EPSILON

# Take the absolute value
abs_profit = np.abs(positive_profit)

# Apply the logarithm
log_profit = np.log(abs_profit)

# Restore the sign
transformed_profit = np.sign(df['Profit']) * log_profit

# Replace -0 with 0
transformed_profit = np.where(transformed_profit == -0, 0, transformed_profit)

# Add the transformed profit column to the DataFrame
df['Transformed_Profit'] = transformed_profit
```

[115]:
```
df[df["Profit"]==0]
```

[115]:

	Ship Mode	Segment	Country	City	State	Region	Category	Sub-Category
201	Standard Class	Home Office	United States	Tampa	Florida	South	Furniture	Furnishings
509	Second Class	Consumer	United States	San Francisco	California	West	Furniture	Chairs
520	First Class	Consumer	United States	Seattle	Washington	West	Office Supplies	Fasteners
526	Standard Class	Corporate	United States	Seattle	Washington	West	Furniture	Chairs
775	Standard Class	Consumer	United States	Philadelphia	Pennsylvania	East	Furniture	Chairs
…	…	…	…	…	…	…	…	…
9272	First Class	Consumer	United States	Los Angeles	California	West	Furniture	Chairs
9501	Standard Class	Corporate	United States	Seattle	Washington	West	Office Supplies	Storage
9746	Standard Class	Consumer	United States	Lafayette	Indiana	Central	Office Supplies	Fasteners
9758	Standard Class	Consumer	United States	Fairfield	Ohio	East	Furniture	Furnishings
9837	Standard Class	Home Office	United States	Los Angeles	California	West	Office Supplies	Fasteners

```
                                                Product Name     Sales  Quantity  \
201   Tenex Contemporary Contur Chairmats for Low an…   258.072       3.0
509        HON 5400 Series Task Chairs for Big and Tall  1121.568       2.0
520            Alliance Big Bands Rubber Bands, 12/Pack     3.960       2.0
526              Hon Every-Day Series Multi-Task Chairs   451.152       3.0
775                      Global Leather Executive Chair  1228.465       5.0
…                                                    …       …       …
9272       HON 5400 Series Task Chairs for Big and Tall  2803.920       5.0
9501             Contico 72"H Heavy-Duty Storage System   204.900       5.0
9746           Alliance Big Bands Rubber Bands, 12/Pack     5.940       3.0
9758        Deflect-o EconoMat Nonstudded, No Bevel Mat    82.640       2.0
9837           Alliance Big Bands Rubber Bands, 12/Pack    13.860       7.0

      Discount  Profit  order_year  order_month  order_date  Ship_year  \
201        0.2     0.0      2017.0          4.0         7.0     2017.0
509        0.2     0.0      2016.0          4.0        15.0     2016.0
520        0.0     0.0      2015.0         12.0         7.0     2015.0
526        0.2     0.0      2017.0         10.0         1.0     2017.0
775        0.3     0.0      2014.0          6.0        28.0     2014.0
…          …       …           …            …           …        …
9272       0.2     0.0      2015.0          1.0        27.0     2015.0
9501       0.0     0.0      2014.0          3.0         7.0     2014.0
9746       0.0     0.0      2014.0          1.0        23.0     2014.0
9758       0.2     0.0      2016.0          6.0         6.0     2016.0
9837       0.0     0.0      2016.0         12.0         6.0     2016.0

      Ship_month  Ship_date  Transformed_Profit
201          4.0       12.0                 0.0
509          4.0       17.0                 0.0
520         12.0        9.0                 0.0
526         10.0        8.0                 0.0
775          7.0        2.0                 0.0
…            …          …                   …
9272         1.0       29.0                 0.0
9501         3.0       12.0                 0.0
9746         1.0       27.0                 0.0
9758         6.0       10.0                 0.0
9837        12.0       10.0                 0.0

[65 rows x 20 columns]
```

[116]:
```
# sum of null values
df["Transformed_Profit"].isnull().sum()
```

[116]:
```
0
```

[117]:
```
df[df["Transformed_Profit"]<=0]
```

[117]:

	Ship Mode	Segment	Country	City	State	Region	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	order_year	order_month	order_date	Ship_year
3	Standard Class	Consumer	United States	Fort Lauderdale	Florida	South	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	957.5775	5.0	0.45	-383.0310	2015.0	10.0	11.0	2015.0
14	Standard Class	Home Office	United States	Fort Worth	Texas	Central	Office Supplies	Appliances	Holmes Replacement Filter for HEPA Air Cleaner…	68.8100	5.0	0.50	-123.8580	2015.0	11.0	22.0	2015.0
15	Standard Class	Home Office	United States	Fort Worth	Texas	Central	Office Supplies	Binders	Storex DuraTech Recycled Plastic Frosted Binders	2.5440	3.0	0.50	-3.8160	2015.0	11.0	22.0	2015.0
23	Second Class	Consumer	United States	Philadelphia	Pennsylvania	East	Furniture	Chairs	Global Deluxe Stacking Chair, Gray	71.3720	2.0	0.30	-1.0196	2017.0	7.0	16.0	2017.0
27	Standard Class	Consumer	United States	Philadelphia	Pennsylvania	East	Furniture	Bookcases	Riverside Palais Royal Lawyers Bookcase, Royal…	3083.4300	7.0	0.50	-1665.0522	2015.0	9.0	17.0	2015.0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
9797	Second Class	Corporate	United States	Los Angeles	California	West	Furniture	Tables	Hon 61000 Series Interactive Training Tables	71.0880	2.0	0.30	-1.0196	2017.0	7.0	16.0	2017.0
9804	Second Class	Home Office	United States	Seattle	Washington	West	Office Supplies	Storage	Rogers Jumbo File, Granite	40.7400	3.0	0.30	-1.0196	2017.0	7.0	16.0	2017.0
9822	First Class	Home Office	United States	Houston	Texas	Central	Furniture	Bookcases	Bush Heritage Pine Collection 5-Shelf Bookcase…	383.4656	4.0	0.30	-1.0196	2017.0	7.0	16.0	2017.0
9837	Standard Class	Home Office	United States	Los Angeles	California	West	Office Supplies	Fasteners	Alliance Big Bands Rubber Bands, 12/Pack	13.8600	7.0	0.50	-1665.0522	2015.0	9.0	17.0	2015.0
9855	Standard Class	Consumer	United States	Phoenix	Arizona	West	Office Supplies	Binders	Plastic Binding Combs	18.1800	4.0	0.50	-1665.0522	2015.0	9.0	17.0	2015.0

```
…         …         …          …            …          …          …
9797      0.20    -1.7772     2016.0          6.0        3.0     2016.0
9804      0.00     0.4074     2015.0          4.0       12.0     2015.0
9822      0.32   -67.6704     2015.0          3.0       19.0     2015.0
9837      0.00     0.0000     2016.0         12.0        6.0     2016.0
9855      0.50   -13.9380     2017.0          9.0       19.0     2017.0

      Ship_month  Ship_date  Transformed_Profit
3           10.0       18.0           -5.948116
14          11.0       26.0           -4.819136
15          11.0       26.0           -1.339203
23           7.0       18.0           -0.019410
27           9.0       21.0           -7.417612
…            …          …                   …
9797         6.0        6.0           -0.575039
9804         4.0       17.0           -0.897960
9822         3.0       22.0           -4.214649
9837        12.0       10.0            0.000000
9855         9.0       25.0           -2.634619

[2123 rows x 20 columns]
```

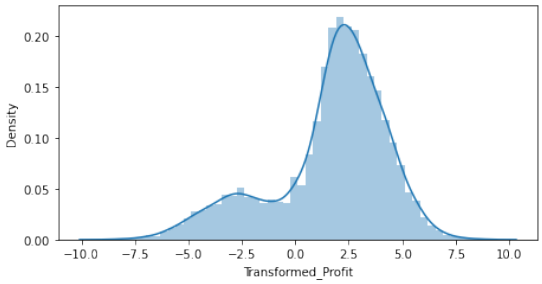
[118]:
```
df.head() #top 5 rows
```

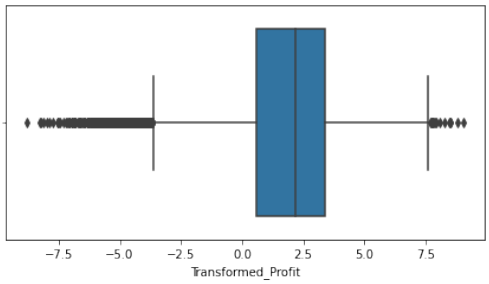
[118]:
```
        Ship Mode    Segment        Country             City       State  \
0    Second Class   Consumer  United States        Henderson    Kentucky
1    Second Class   Consumer  United States        Henderson    Kentucky
2    Second Class  Corporate  United States      Los Angeles  California
3  Standard Class   Consumer  United States  Fort Lauderdale     Florida
4  Standard Class   Consumer  United States  Fort Lauderdale     Florida

  Region         Category Sub-Category  \
0  South        Furniture    Bookcases
1  South        Furniture       Chairs
2   West  Office Supplies       Labels
3  South        Furniture       Tables
4  South  Office Supplies      Storage

                                        Product Name     Sales  Quantity  \
0                 Bush Somerset Collection Bookcase  261.9600       2.0
1  Hon Deluxe Fabric Upholstered Stacking Chairs,…  731.9400       3.0
2  Self-Adhesive Address Labels for Typewriters b…   14.6200       2.0
3      Bretford CR4500 Series Slim Rectangular Table  957.5775       5.0
4                   Eldon Fold 'N Roll Cart System   22.3680       2.0

   Discount    Profit  order_year  order_month  order_date  Ship_year  \
0      0.00   41.9136      2016.0         11.0         8.0     2016.0
```

```
1       0.00  219.5820       2016.0          11.0          8.0      2016.0
2       0.00    6.8714       2016.0           6.0         12.0      2016.0
3       0.45 -383.0310       2015.0          10.0         11.0      2015.0
4       0.20    2.5164       2015.0          10.0         11.0      2015.0

   Ship_month  Ship_date  Transformed_Profit
0        11.0       11.0            3.735610
1        11.0       11.0            5.391726
2         6.0       16.0            1.927368
3        10.0       18.0           -5.948116
4        10.0       18.0            0.922829
```

[119]:
```
#distribution and boxplot for transformed profit
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(df["Transformed_Profit"])
plt.subplot(2,2,2)
sns.boxplot(df["Transformed_Profit"])

plt.show()
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```





[120]:
```
#left skew
df["Transformed_Profit"].skew()
```

[120]:
```
-0.8874539960016707
```

[121]:
```
df.shape
```

[121]:
```
(9844, 20)
```

[122]:
```
Q1 = df['Transformed_Profit'].quantile(0.25)
Q3 = df['Transformed_Profit'].quantile(0.75)

# Calculate IQR(Inter Quantile Range)
IQR = Q3 - Q1

# Define lower and upper bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers
df_no_outliers = df[(df['Transformed_Profit'] >= lower_bound) &␣
 ↪(df['Transformed_Profit'] <= upper_bound)]
```

[123]:
```
df.shape
```

[123]:
```
(9844, 20)
```

[124]:
```
#copying data in df
df=df_no_outliers.copy()
```

[125]:
```
df.shape
```

[125]:
```
(9233, 20)
```

[126]:
```
#sum of null values in transformed profit
df['Transformed_Profit'].isnull().sum()
```

[126]:
```
0
```

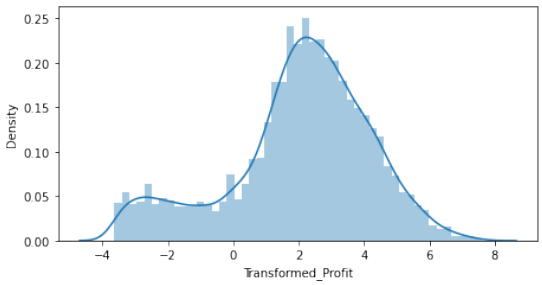
[127]:
```
df.shape
```

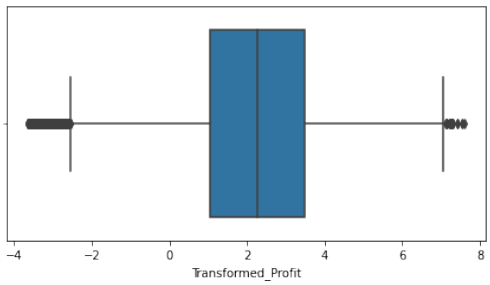
[127]:
```
(9233, 20)
```

[128]:
```
#distribution and box plot for transformed profit
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(df["Transformed_Profit"])
plt.subplot(2,2,2)
sns.boxplot(df["Transformed_Profit"])
```

```
plt.show()
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```





[129]:
```
df["Transformed_Profit"].skew()
```

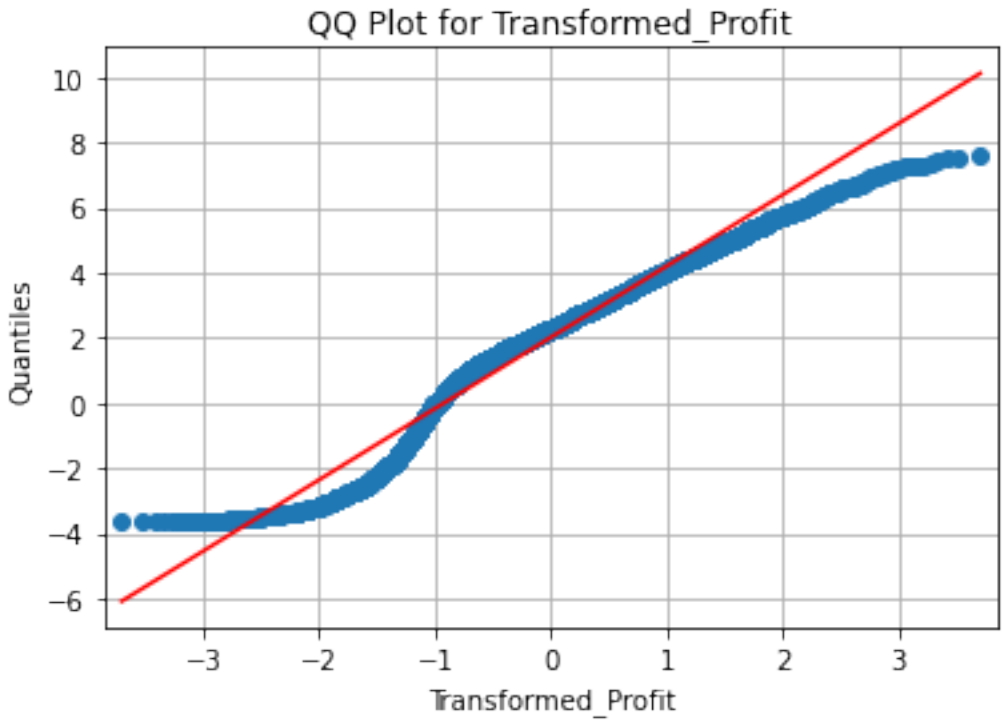
[129]:
```
-0.6187364044245487
```

[130]:
```
#display all columns
df.columns
```

[130]:
```
Index(['Ship Mode', 'Segment', 'Country', 'City', 'State', 'Region',
       'Category', 'Sub-Category', 'Product Name', 'Sales', 'Quantity',
       'Discount', 'Profit', 'order_year', 'order_month', 'order_date',
       'Ship_year', 'Ship_month', 'Ship_date', 'Transformed_Profit'],
      dtype='object')
```

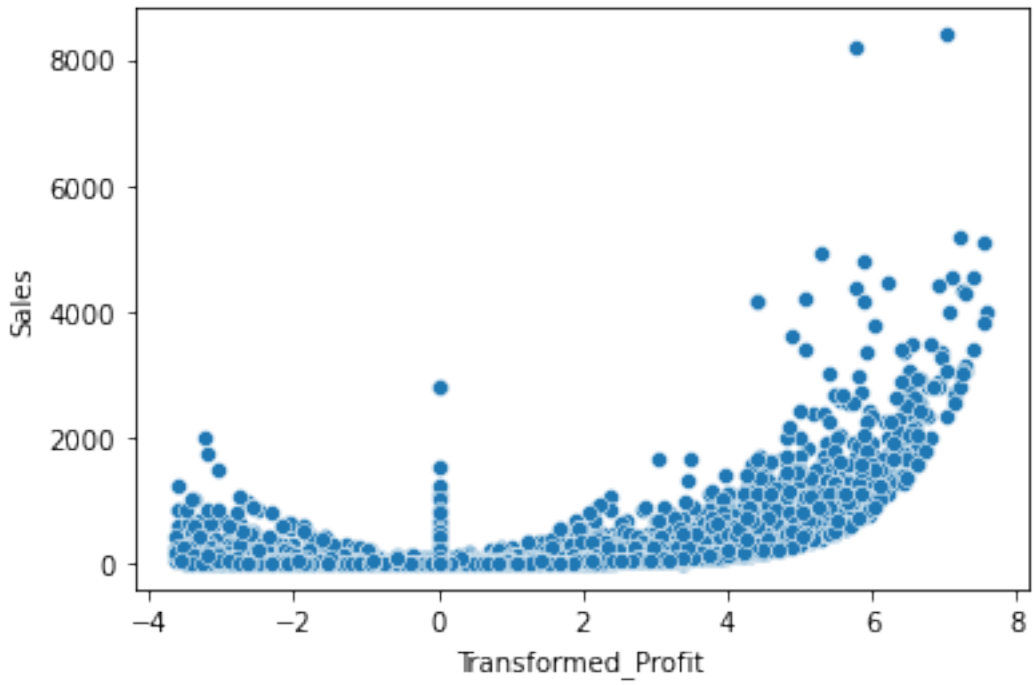
[131]:
```
data = df

# Select the column you want to analyze
column_name = 'Transformed_Profit'
column_data = data[column_name]

# Create the QQ plot using statsmodels.api
sm.qqplot(column_data, line='s')  # 's' for straight reference line
```

```
plt.xlabel(column_name)
plt.ylabel('Quantiles')
plt.title('QQ Plot for ' + column_name)
plt.grid(True)
plt.show()
```



[132]:
```
#Scatter plot sales vs transformed_profit
sns.scatterplot(x="Transformed_Profit",y="Sales",data=df)
```

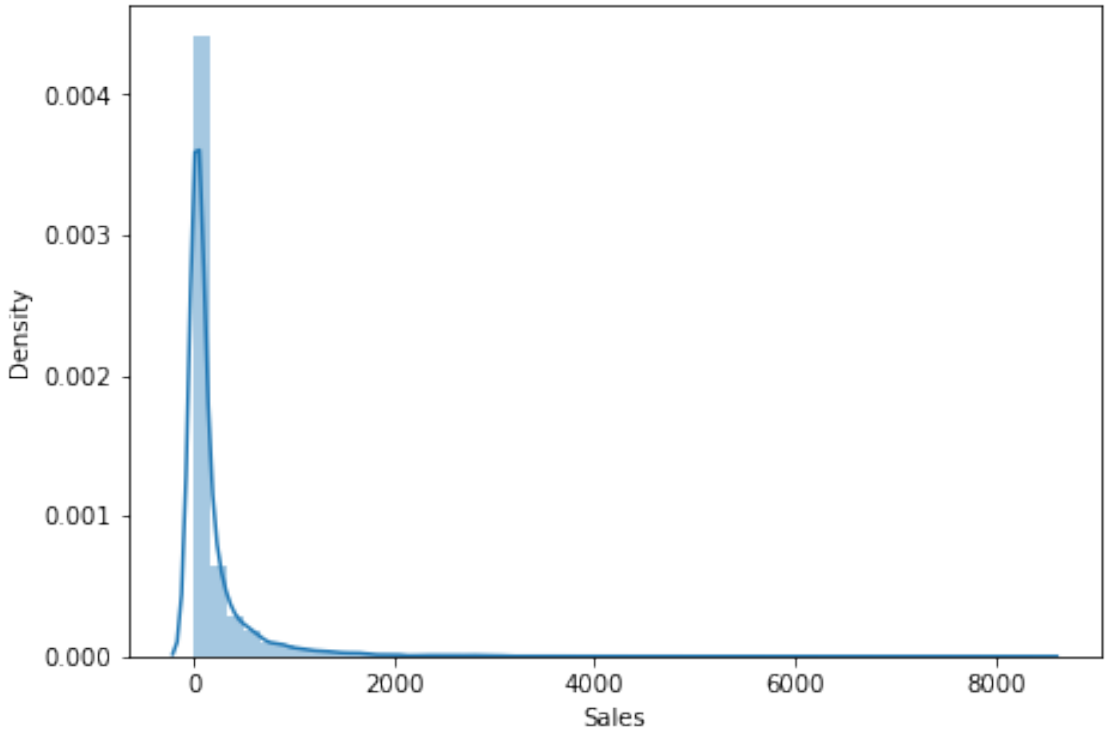
[132]: <AxesSubplot:xlabel='Transformed_Profit', ylabel='Sales'>



14 Sales Outlier Analysis

[133]:
```
# sales data distribution graph

plt.figure(figsize=(16,5))
plt.subplot(1,2,2)
sns.distplot(df["Sales"])
plt.show()
```

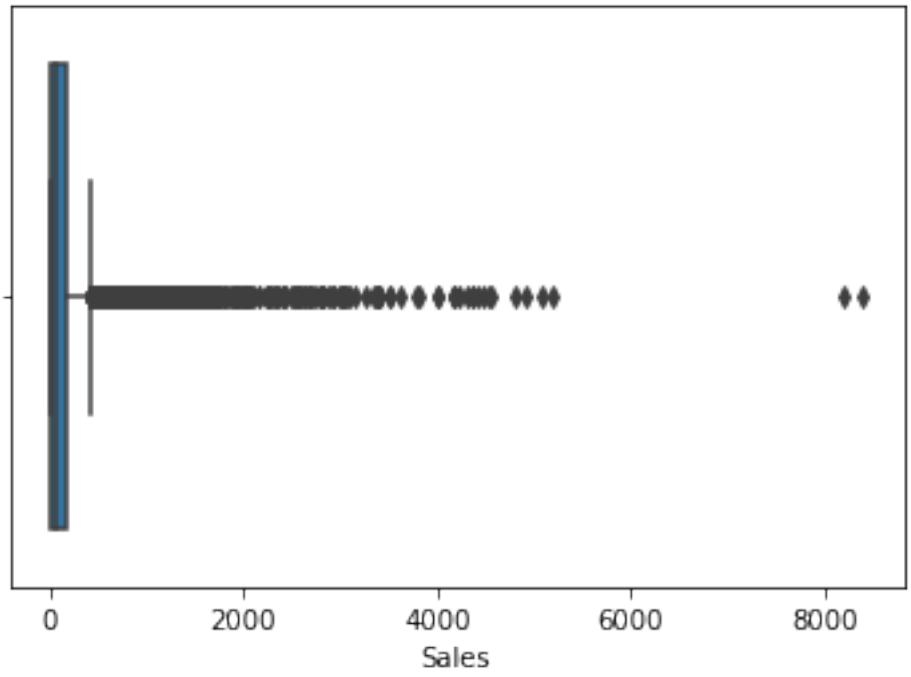
```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



[134]:
```
#box plot to check outliers
sns.boxplot(df["Sales"])
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

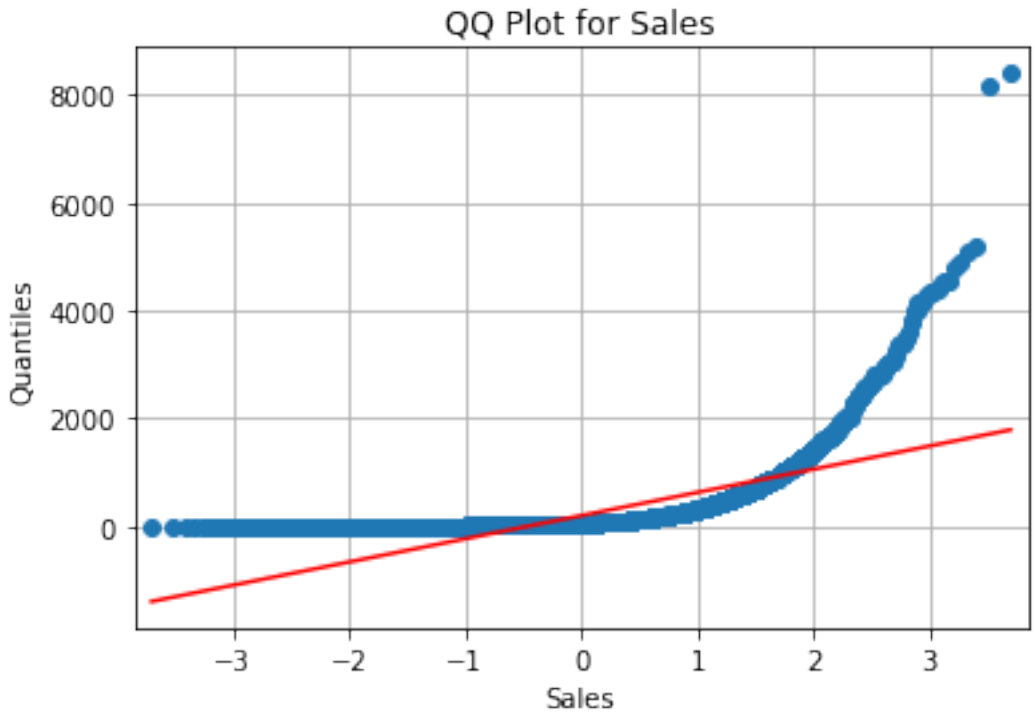
[134]:
```
<AxesSubplot:xlabel='Sales'>
```



[135]:
```
#checking skewnessin data
df["Sales"].skew() #right skew
```

[135]: 5.945170113821082

[136]:
```
data = df

# Select the column you want to analyze
column_name = 'Sales'
column_data = data[column_name]

# Create the QQ plot using statsmodels.api
sm.qqplot(column_data, line='s')  # 's' for straight reference line
plt.xlabel(column_name)
plt.ylabel('Quantiles')
plt.title('QQ Plot for ' + column_name)
plt.grid(True)
plt.show()
```



[137]:
```
#data transformation using log
df['Transformed_Sales'] = np.log(df['Sales'])
```

[138]:
```
df.head() #top 5 rows display
```

[138]:

	Ship Mode	Segment	Country	City	State	Region	Category	Sub-Category	Product Name	Sales	Quantity
0	Second Class	Consumer	United States	Henderson	Kentucky	South	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.960	2.0
1	Second Class	Consumer	United States	Henderson	Kentucky	South	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,…	731.940	3.0
2	Second Class	Corporate	United States	Los Angeles	California	West	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b…	14.620	2.0
4	Standard Class	Consumer	United States	Fort Lauderdale	Florida	South	Office Supplies	Storage			
5	Standard Class	Consumer	United States	Los Angeles	California	West	Furniture	Furnishings			

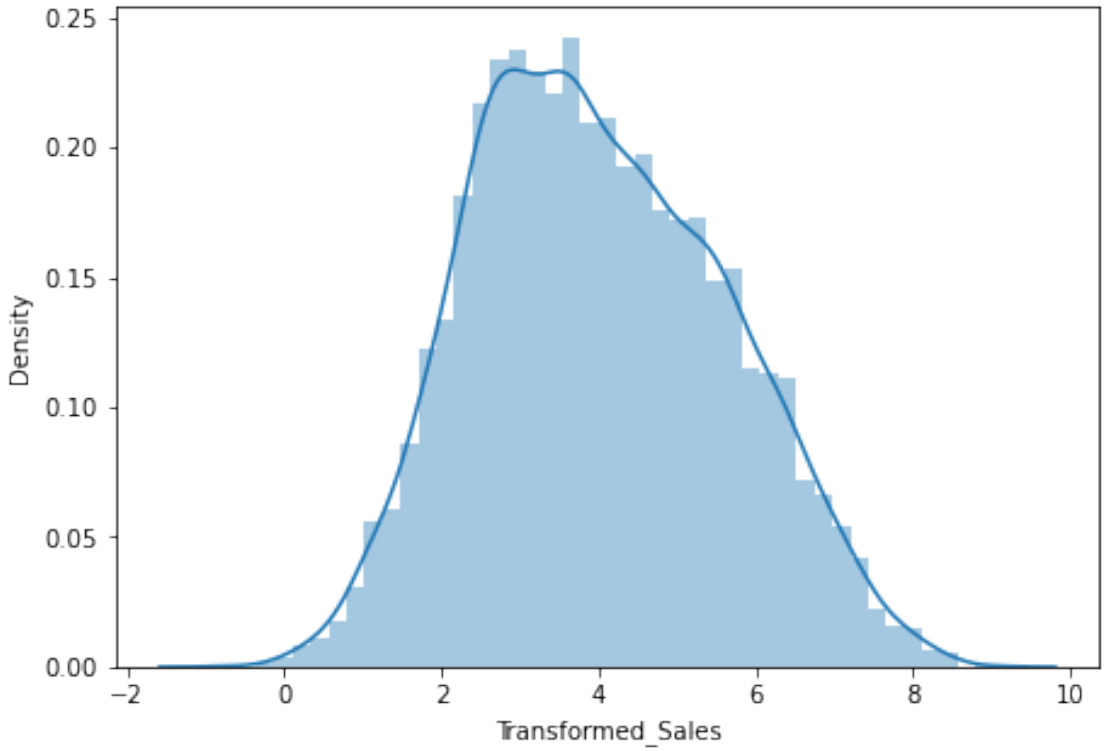
```
4                    Eldon Fold 'N Roll Cart System   22.368       2.0
5  Eldon Expressions Wood and Plastic Desk Access…   48.860       7.0

   Discount    Profit  order_year  order_month  order_date  Ship_year  \
0       0.0   41.9136      2016.0         11.0         8.0     2016.0
1       0.0  219.5820      2016.0         11.0         8.0     2016.0
2       0.0    6.8714      2016.0          6.0        12.0     2016.0
4       0.2    2.5164      2015.0         10.0        11.0     2015.0
5       0.0   14.1694      2014.0          6.0         9.0     2014.0

   Ship_month  Ship_date  Transformed_Profit  Transformed_Sales
0        11.0       11.0            3.735610           5.568192
1        11.0       11.0            5.391726           6.595699
2         6.0       16.0            1.927368           2.682390
4        10.0       18.0            0.922829           3.107631
5         6.0       14.0            2.651085           3.888959
```

[139]:
```
df.shape
```

[139]:
```
(9233, 21)
```

[140]:
```
#distribution graph after transformation
plt.figure(figsize=(16,5))
plt.subplot(1,2,2)
sns.distplot(df["Transformed_Sales"])
plt.show()
```

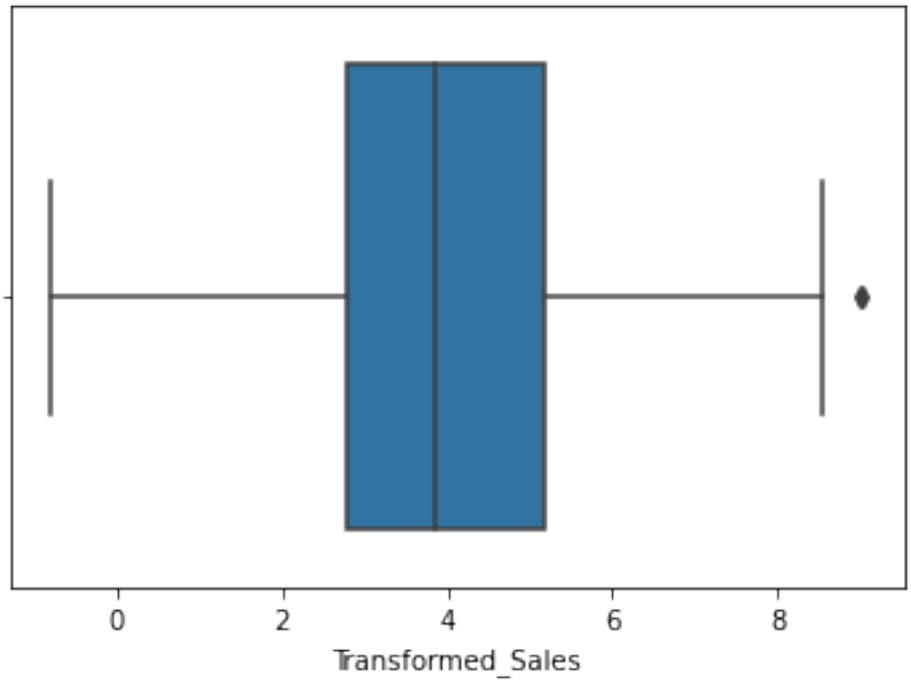
```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



[141]:
```
#boxplot
sns.boxplot(df["Transformed_Sales"])
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

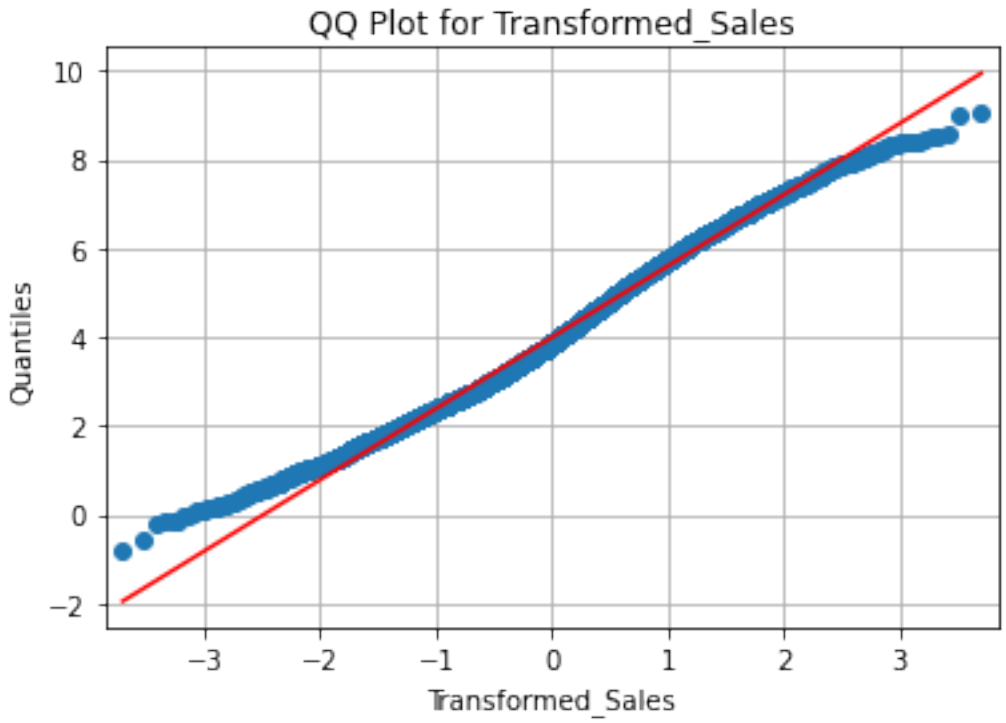
[141]:
```
<AxesSubplot:xlabel='Transformed_Sales'>
```



[142]:
```
#right skewness
df["Transformed_Sales"].skew()
```

[142]:
```
0.2193355558713904
```

[143]:
```
import statsmodels.api as sm
data = df

# Select the column you want to analyze
column_name = 'Transformed_Sales'
column_data = data[column_name]

# Create the QQ plot using statsmodels.api
sm.qqplot(column_data, line='s')  # 's' for straight reference line
plt.xlabel(column_name)
plt.ylabel('Quantiles')
plt.title('QQ Plot for ' + column_name)
plt.grid(True)
plt.show()
```



[144]:
```
df.shape
```

[144]: (9233, 21)

[145]:
```
#z_score technique to delete outliers
z_scores = (df['Transformed_Sales'] - df['Transformed_Sales'].mean()) /␣
 ↪df['Transformed_Sales'].std()
outliers = df[np.abs(z_scores) > 3]
```

[146]:
```
#drop outliers
df = df.drop(outliers.index)
```

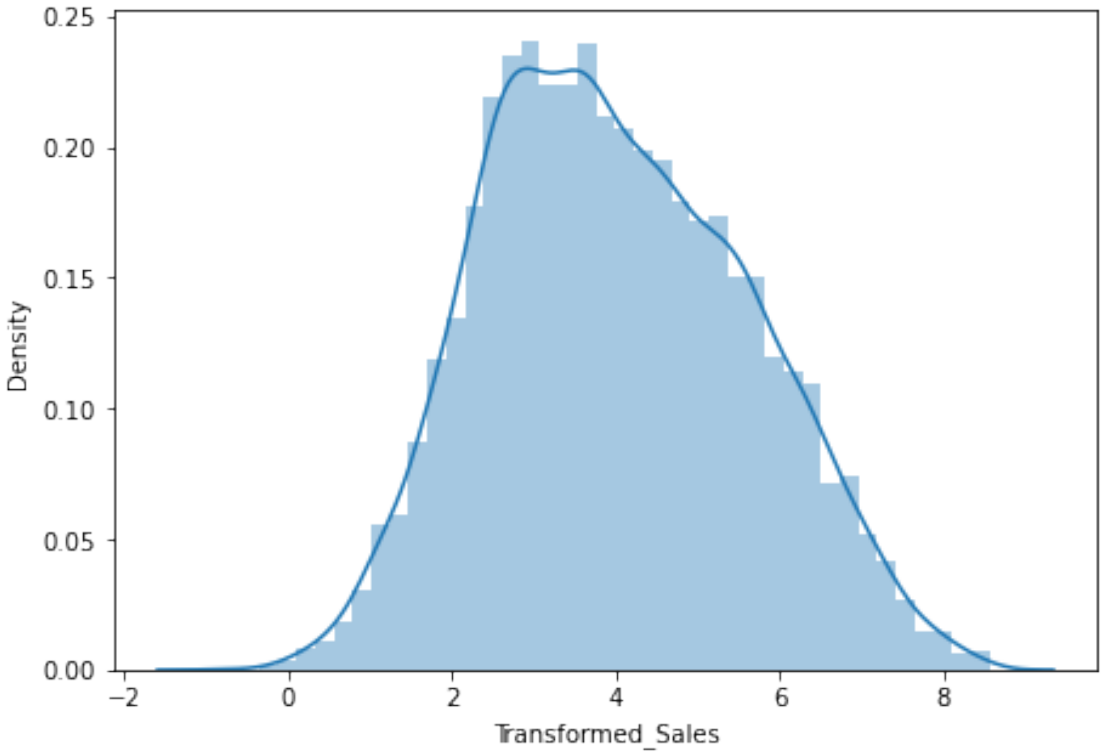
[147]:
```
df.shape
```

[147]: (9231, 21)

[148]:
```
#distribution graph after outlier deletion
plt.figure(figsize=(16,5))
plt.subplot(1,2,2)
sns.distplot(df["Transformed_Sales"])
plt.show()
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
```

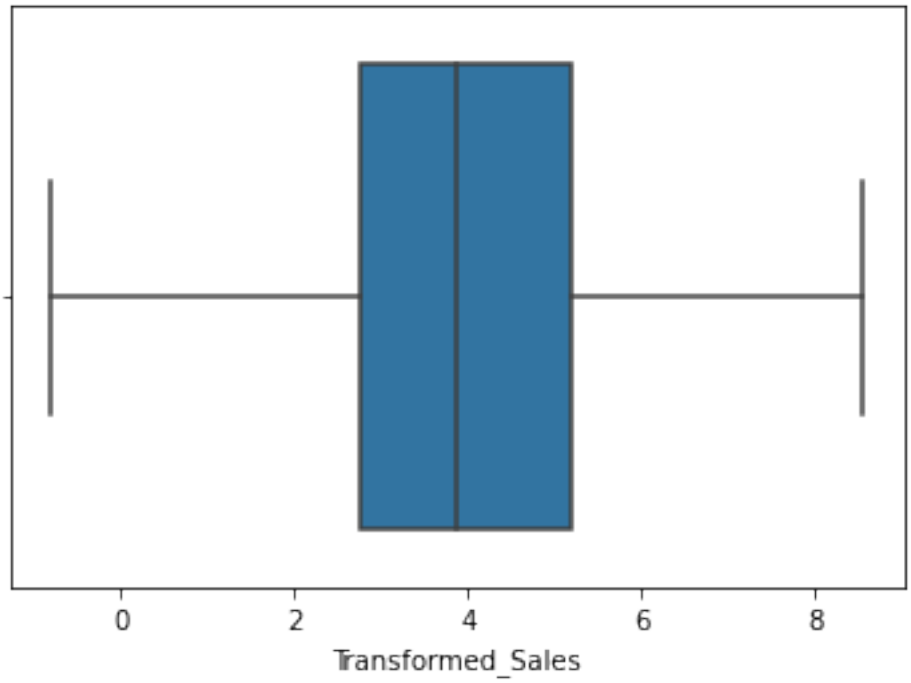
```
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



[149]:
```
#box plot
sns.boxplot(df["Transformed_Sales"])
```

```
C:\Users\ganesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

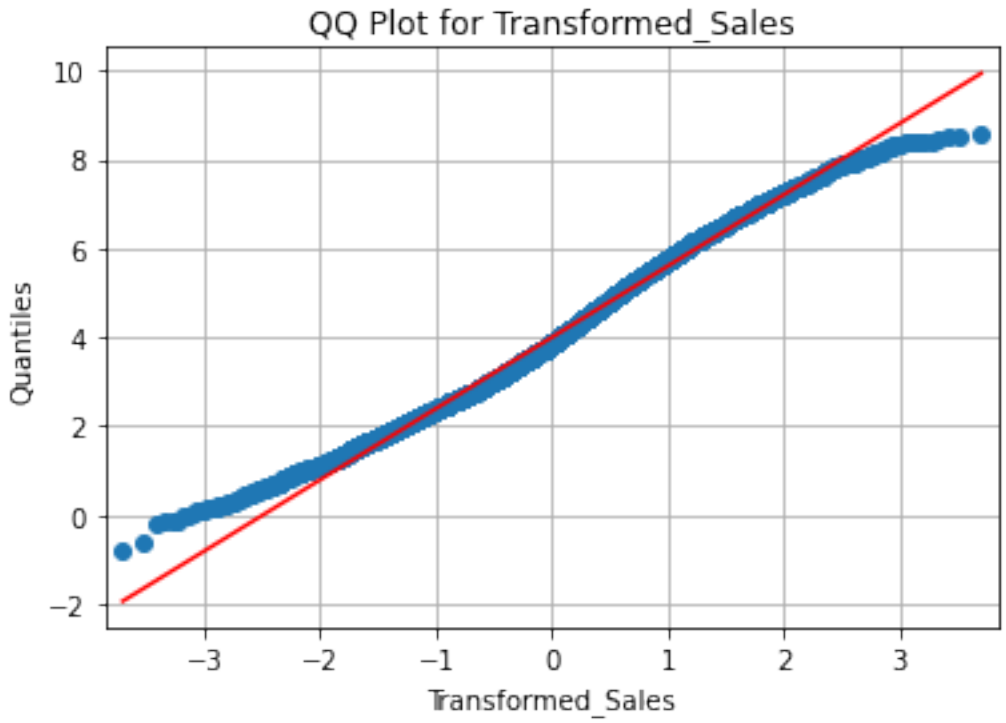
[149]:
```
<AxesSubplot:xlabel='Transformed_Sales'>
```



[150]:
```
#skew
df["Transformed_Sales"].skew()
```

[150]:
```
0.21540098687838372
```

[151]:
```
data = df

# Select the column you want to analyze
column_name = 'Transformed_Sales'
column_data = data[column_name]

# Create the QQ plot using statsmodels.api
sm.qqplot(column_data, line='s')  # 's' for straight reference line
plt.xlabel(column_name)
plt.ylabel('Quantiles')
plt.title('QQ Plot for ' + column_name)
plt.grid(True)
plt.show()
```



[152]:
```
df
```

[152]:

	Ship Mode	Segment	Country	City	State	Region	Category	Sub-Category
0	Second Class	Consumer	United States	Henderson	Kentucky	South	Furniture	Bookcases
1	Second Class	Consumer	United States	Henderson	Kentucky	South	Furniture	Chairs
2	Second Class	Corporate	United States	Los Angeles	California	West	Office Supplies	Labels
4	Standard Class	Consumer	United States	Fort Lauderdale	Florida	South	Office Supplies	Storage
5	Standard Class	Consumer	United States	Los Angeles	California	West	Furniture	Furnishings
…	…	…	…	…	…	…	…	…
9867	Standard Class	Consumer	United States	Detroit	Michigan	Central	Office Supplies	Binders
9868	Standard Class	Consumer	United States	Detroit	Michigan			
9869	Standard Class	Consumer	United States	Detroit	Michigan			
9870	Second Class	Corporate	United States	Fort Lauderdale	Florida			
9871	Second Class	Consumer	United States	Hampton	Virginia			

```
9868  Central  Office Supplies          Art
9869  Central  Office Supplies      Binders
9870    South  Office Supplies   Appliances
9871    South  Office Supplies   Appliances

                                          Product Name     Sales  Quantity  \
0                    Bush Somerset Collection Bookcase   261.960       2.0
1     Hon Deluxe Fabric Upholstered Stacking Chairs,…   731.940       3.0
2     Self-Adhesive Address Labels for Typewriters b…    14.620       2.0
4                       Eldon Fold 'N Roll Cart System    22.368       2.0
5     Eldon Expressions Wood and Plastic Desk Access…    48.860       7.0
…                                                  …       …       …
9867                         GBC Plastic Binding Combs    29.520       4.0
9868                                       Newell 315    11.960       2.0
9869     Wilson Jones 1" Hanging DublLock Ring Binders    26.400       5.0
9870                Hoover Upright Vacuum With Dirt Cup  1158.120       5.0
9871  Holmes Replacement Filter for HEPA Air Cleaner…    44.430       3.0

      Discount    Profit  order_year  order_month  order_date  Ship_year  \
0          0.0   41.9136      2016.0         11.0         8.0     2016.0
1          0.0  219.5820      2016.0         11.0         8.0     2016.0
2          0.0    6.8714      2016.0          6.0        12.0     2016.0
4          0.2    2.5164      2015.0         10.0        11.0     2015.0
5          0.0   14.1694      2014.0          6.0         9.0     2014.0
…        …       …          …           …          …         …
9867       0.0   14.4648      2016.0          9.0         1.0     2016.0
9868       0.0    2.9900      2016.0          9.0         1.0     2016.0
9869       0.0   12.6720      2016.0          9.0         1.0     2016.0
9870       0.2  130.2885      2017.0         11.0        11.0     2017.0
9871       0.0   18.6606      2015.0         11.0         8.0     2015.0

      Ship_month  Ship_date  Transformed_Profit  Transformed_Sales
0           11.0       11.0            3.735610           5.568192
1           11.0       11.0            5.391726           6.595699
2            6.0       16.0            1.927368           2.682390
4           10.0       18.0            0.922829           3.107631
5            6.0       14.0            2.651085           3.888959
…          …         …                 …                  …
9867         9.0        5.0            2.671718           3.385068
9868         9.0        5.0            1.095273           2.481568
9869         9.0        5.0            2.539395           3.273364
9870        11.0       16.0            4.869751           7.054553
9871        11.0       13.0            2.926414           3.793915

[9231 rows x 21 columns]
```

15 Statistical Analysis:

16 Investigate the performance of different customer segments (Consumer, Corporate, Home Office) in terms of sales, quantity, and profit.

[153]:
```
print(df.groupby('Segment').agg({'Sales': 'sum', 'Quantity': 'sum', 'Profit':␣
↪'sum'}))
```

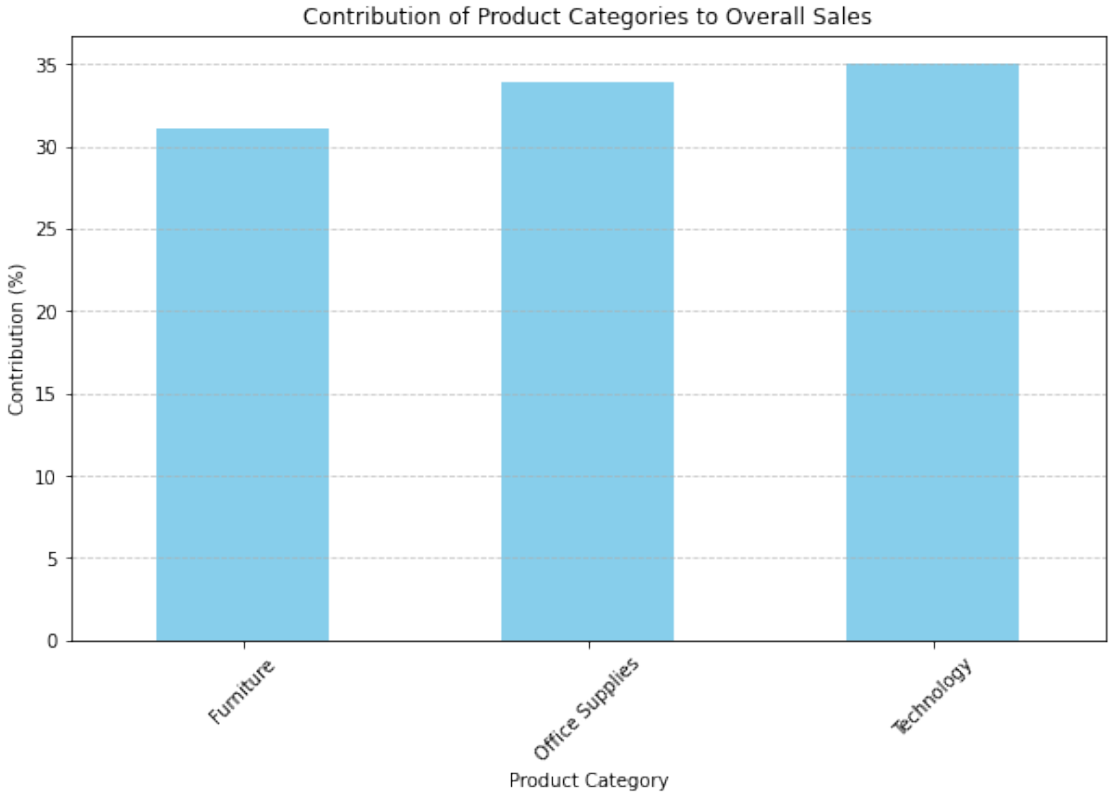
```
                Sales  Quantity         Profit
Segment
Consumer     880281.9858   17434.0  173357.412232
Corporate    558708.6391   10306.0  114919.723887
Home Office  340851.1344    6052.0   77291.453700
```

17 Examine the contribution of each product category to overall sales.

[154]:
```
total_sales = df['Sales'].sum()
category_sales = df.groupby('Category')['Sales'].sum()
category_contribution = (category_sales / total_sales) * 100
category_contribution
```

[154]:
```
Category
Furniture          31.121490
Office Supplies    33.880769
Technology         34.997740
Name: Sales, dtype: float64
```

[155]:
```
plt.figure(figsize=(10, 6))
category_contribution.plot(kind='bar', color='skyblue')
plt.title('Contribution of Product Categories to Overall Sales')
plt.xlabel('Product Category')
plt.ylabel('Contribution (%)')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



18 Compare the mean sales across different regions using statistical tests (e.g., ANOVA).

[156]:
```
# Filter data for unique regions
unique_regions = df['Region'].unique()

# Perform ANOVA test for sales across different regions
anova_results = f_oneway(*[df[df['Region'] == region]['Sales'] for region in␣
 ↪unique_regions])

print("ANOVA Results:")
print("F-statistic:", anova_results.statistic)
print("P-value:", anova_results.pvalue)
```

```
ANOVA Results:
F-statistic: 4.43162405867444
P-value: 0.004057174733622733
```

19 Conduct pairwise comparisons to identify regions with significantly different sales

[157]:
```
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Assuming you have loaded your Superstore dataset into a pandas DataFrame␣
 ↪named 'superstore_data'

# Fit ANOVA model
model = ols('Sales ~ Region', data=df).fit()
anova_table = sm.stats.anova_lm(model, typ=2)

# Conduct Tukey's HSD test for pairwise comparisons
tukey_results = sm.stats.multicomp.pairwise_tukeyhsd(endog=df['Sales'],␣
 ↪groups=df['Region'], alpha=0.05)

# Print Tukey's HSD test results
print("Tukey's HSD test results:")
print(tukey_results)

# Interpret results
print("\nPairwise comparisons:")
print(tukey_results.summary())
```

```
Tukey's HSD test results:
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1 group2 meandiff p-adj   lower    upper  reject
-----------------------------------------------------
Central   East  18.3861 0.4352 -13.0315 49.8038  False
Central  South  22.8597 0.3582 -13.0387 58.7582  False
Central   West  42.2169  0.002  11.9083 72.5255   True
   East  South   4.4736  0.987 -29.7819 38.7291  False
   East   West  23.8308 0.1345  -4.5126 52.1742  False
  South   West  19.3572 0.4397  -13.884 52.5983  False
-----------------------------------------------------

Pairwise comparisons:
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1 group2 meandiff p-adj   lower    upper  reject
-----------------------------------------------------
Central   East  18.3861 0.4352 -13.0315 49.8038  False
Central  South  22.8597 0.3582 -13.0387 58.7582  False
Central   West  42.2169  0.002  11.9083 72.5255   True
```

```
  East  South   4.4736  0.987 -29.7819 38.7291  False
  East   West  23.8308 0.1345  -4.5126 52.1742  False
 South   West  19.3572 0.4397  -13.884 52.5983  False
----------------------------------------------------
```

20 Explore relationships between variables such as Sales, Quantity, Discount, and Profit

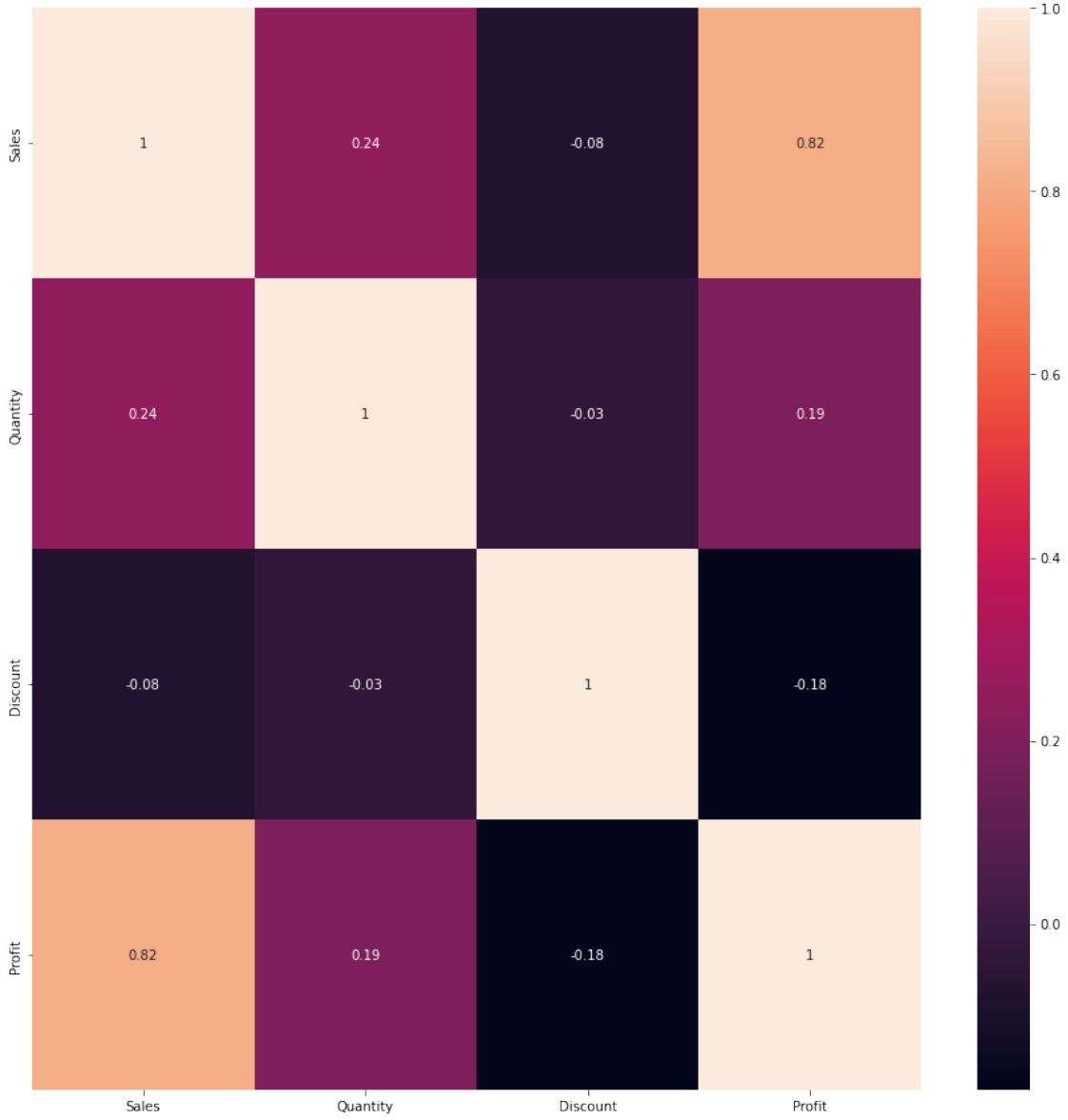
[158]:
```
df[['Sales', 'Quantity', 'Discount', 'Profit']].corr()
```

[158]:

	Sales	Quantity	Discount	Profit
Sales	1.000000	0.243554	-0.079578	0.815014
Quantity	0.243554	1.000000	-0.030481	0.192574
Discount	-0.079578	-0.030481	1.000000	-0.181416
Profit	0.815014	0.192574	-0.181416	1.000000

[159]:
```
plt.figure(figsize=(15,15))
sns.heatmap(df[['Sales', 'Quantity', 'Discount', 'Profit']].corr(),annot=True)
```

[159]:
```
<AxesSubplot:>
```



21 Identify regions with the highest and lowest sales and add this to the graph.

[160]:
```
print("Region with highest sales: ",df.groupby("Region")["Sales"].sum().
↪sort_values(ascending=False).index[0])
```

```
Region with highest sales:  West
```

[161]:
```
print("Lowest Sales value: ",df.groupby("Region")["Sales"].sum().
↪sort_values(ascending=False)[0])
```

```
Lowest Sales value:  650791.8735
```

[162]:
```
print("Region with lowest sales: ",df.groupby("Region")["Sales"].sum().
 ↪sort_values(ascending=True).index[0])
```

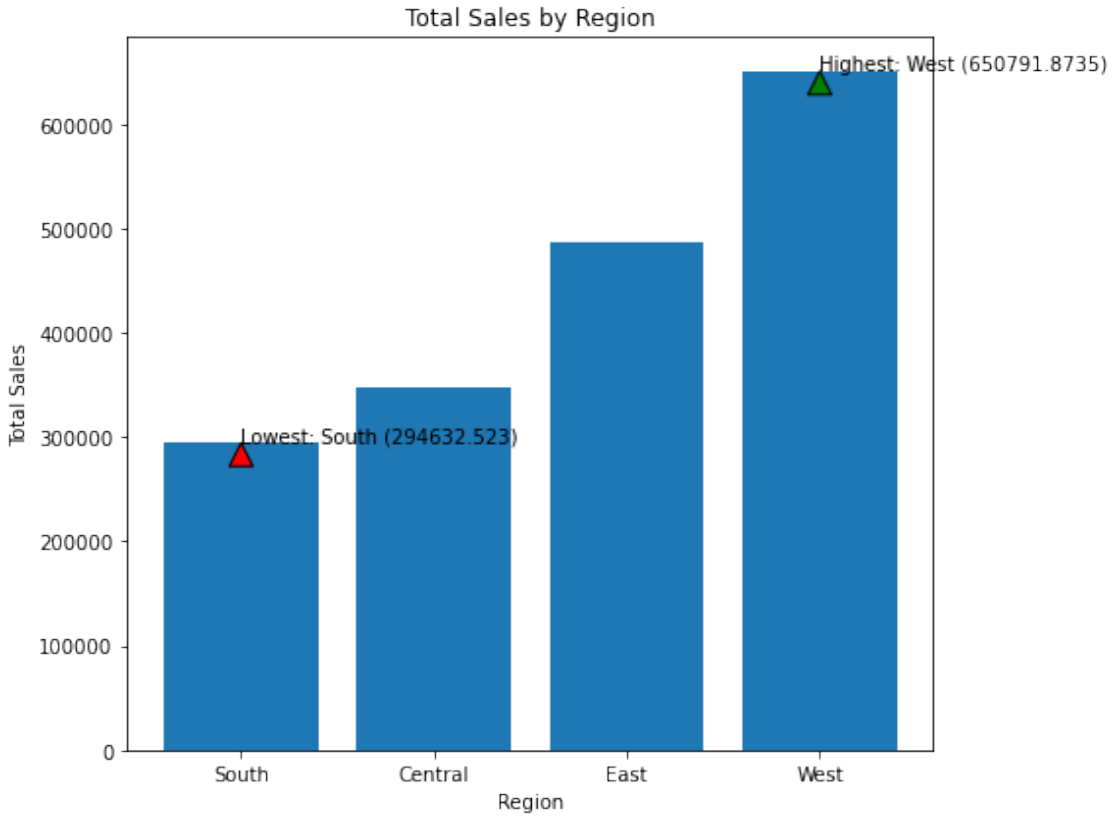
```
Region with lowest sales:  South
```

[163]:
```
print("Lowest Sales value: ",df.groupby("Region")["Sales"].sum().
 ↪sort_values(ascending=True)[0])
```

```
Lowest Sales value:  294632.523
```

[164]:
```
# Calculate total sales per region
total_sales = df.groupby("Region")["Sales"].sum()

# Sort total sales (ascending order for illustration)
sorted_sales = total_sales.sort_values(ascending=True)

# Get the region with the highest sales
highest_sales_region = df.groupby("Region")["Sales"].sum().
 ↪sort_values(ascending=False).index[0]
highest_sales_value = df.groupby("Region")["Sales"].sum().
 ↪sort_values(ascending=False)[0]

# Get the region with the lowest sales
lowest_sales_region = df.groupby("Region")["Sales"].sum().
 ↪sort_values(ascending=True).index[0]
lowest_sales_value = df.groupby("Region")["Sales"].sum().
 ↪sort_values(ascending=True)[0]

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(sorted_sales.index, sorted_sales.values)
plt.xlabel("Region")
plt.ylabel("Total Sales")
plt.title("Total Sales by Region")

# Add annotations for highest and lowest sales
plt.annotate(f"Highest: {highest_sales_region} ({highest_sales_value})",
             xy=(highest_sales_region, highest_sales_value),
             xytext=(highest_sales_region, highest_sales_value + 20),
             arrowprops=dict(facecolor='green', shrink=0.05))

plt.annotate(f"Lowest: {lowest_sales_region} ({lowest_sales_value})",
             xy=(lowest_sales_region, lowest_sales_value),
             xytext=(lowest_sales_region, lowest_sales_value - 20),
             arrowprops=dict(facecolor='red', shrink=0.05))
```

```
plt.xticks(rotation=0)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



22 Identify the top-performing and underperforming products based on sales and profit.

[165]:
```
df.groupby("Product Name")["Sales"].sum().nlargest(5)
```

[165]:
```
Product Name
Fellowes PB500 Electric Punch Plastic Comb Binding Machine with Manual Bind
18810.652
GBC DocuBind TL300 Electric Binding System
15428.228
HON 5400 Series Task Chairs for Big and Tall
14019.600
Samsung Galaxy Mega 6.3
13943.668
```

```
Hewlett Packard LaserJet 3310 Copier
13439.776
Name: Sales, dtype: float64
```

[166]:
```
df.groupby("Product Name")["Sales"].sum().nsmallest(5)
```

[166]:
```
Product Name
Eureka Disposable Bags for Sanitaire Vibra Groomer I Upright Vac    1.624
Avery 5                                                             5.760
Xerox 20                                                            6.480
Grip Seal Envelopes                                                 7.072
Avery Hi-Liter Pen Style Six-Color Fluorescent Set                  7.700
Name: Sales, dtype: float64
```

[167]:
```
df.groupby("Product Name")["Profit"].sum().nlargest(5)
```

[167]:
```
Product Name
Fellowes PB500 Electric Punch Plastic Comb Binding Machine with Manual Bind
8007.2370
GBC DocuBind TL300 Electric Binding System
6395.5387
Hewlett Packard LaserJet 3310 Copier
4391.9268
Plantronics Savi W720 Multi-Device Wireless Headset System
3696.2820
Honeywell Enviracaire Portable HEPA Air Cleaner for 17' x 22' Room
3247.0200
Name: Profit, dtype: float64
```

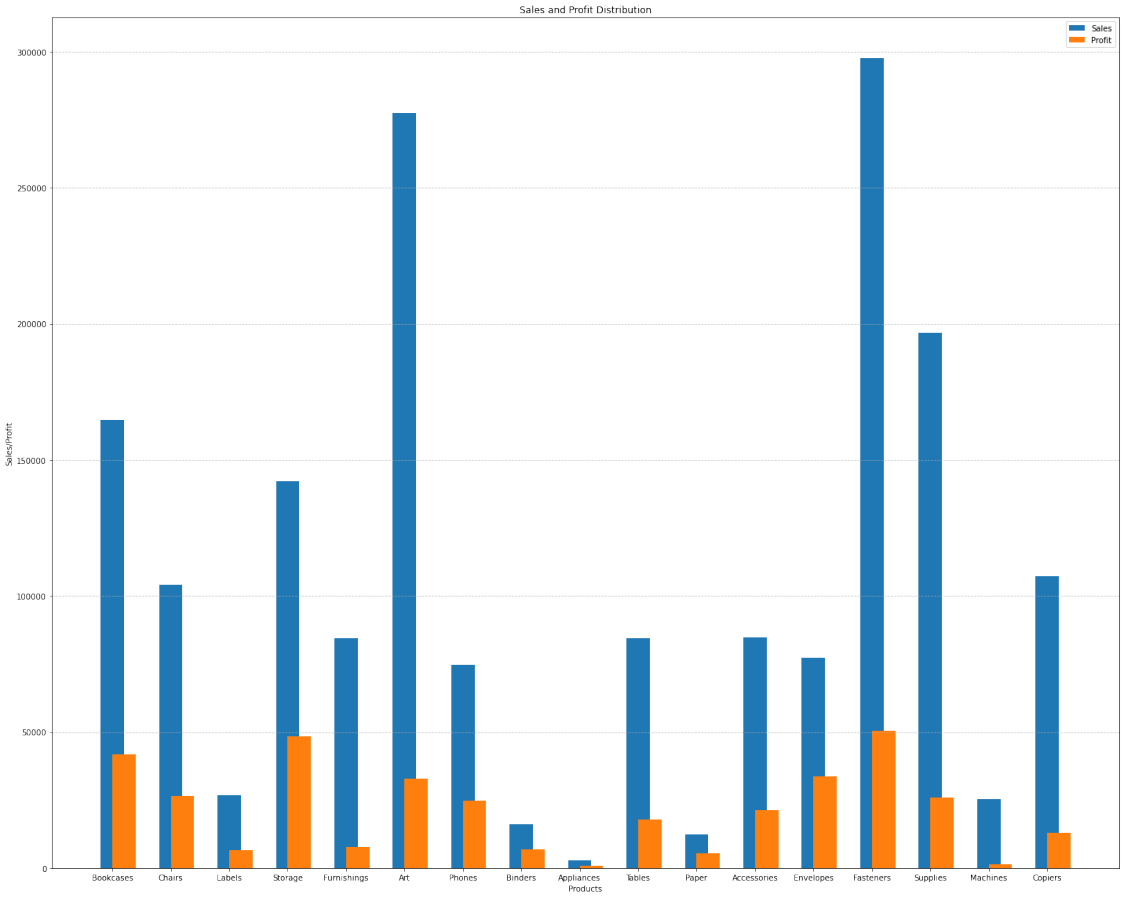
[168]:
```
df.groupby("Product Name")["Profit"].sum().nsmallest(5)
```

[168]:
```
Product Name
Global Deluxe Steno Chair                          -157.8090
Hon 61000 Series Interactive Training Tables       -105.7434
Global Wood Trimmed Manager's Task Chair, Khaki     -90.0702
Eldon "L" Workstation Diamond Chairmat              -85.0976
Belkin 19" Vented Equipment Shelf, Black            -81.8532
Name: Profit, dtype: float64
```

23 Visualize the sales and profit distribution for different products

[173]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Select relevant columns
products = df["Sub-Category"].unique()
```

```
sales = df.groupby("Sub-Category")["Sales"].sum()
profit = df.groupby("Sub-Category")["Profit"].sum()

# Create a bar chart
plt.figure(figsize=(20, 16))
plt.bar(products, sales, label="Sales", width=0.4, align='center')
plt.bar([p for p in products], [p + 0.4 for p in profit], label="Profit",␣
  ↪width=0.4, align='edge')  # Adjust bar positions to avoid overlap
plt.xlabel("Products")
plt.ylabel("Sales/Profit")
plt.title("Sales and Profit Distribution")
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



24 Hypothesis Testing

25 Formulate a hypothesis related to the data (e.g., the impact of discounts on sales).

26 Conduct hypothesis testing using appropriate statistical tests.

[174]:
```
import pandas as pd
from scipy.stats import ttest_ind

# Assuming you have loaded your Superstore dataset into a pandas DataFrame␣
 ↪named 'superstore_data'

# Separate data into two groups: orders with discount and orders without␣
 ↪discount
sales_with_discount = df[df['Discount'] > 0]['Sales']
sales_without_discount = df[df['Discount'] == 0]['Sales']

# Conduct two-sample t-test
t_statistic, p_value = ttest_ind(sales_with_discount, sales_without_discount)

# Print results
print("Two-sample t-test results:")
print("T-statistic:", t_statistic)
print("P-value:", p_value)

# Interpret results
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis. There is a significant difference in␣
 ↪sales between orders with a discount and orders without a discount.")
else:
    print("Fail to reject the null hypothesis. There is no significant␣
 ↪difference in sales between orders with a discount and orders without a␣
 ↪discount.")
```

```
Two-sample t-test results:
T-statistic: -2.1153451183517507
P-value: 0.034427263721101296
Reject the null hypothesis. There is a significant difference in sales between
orders with a discount and orders without a discount.
```

[175]:
```
# Separate data for positive and negative profit
positive_profit_sales = df[df['Profit'] > 0]['Sales']
negative_profit_sales = df[df['Profit'] < 0]['Sales']
```

```
# Perform two-sample t-test for profit and sales
t_statistic_profit, p_value_profit = ttest_ind(positive_profit_sales,␣
 ↪negative_profit_sales)

# Print t-statistic and p-value
print("Profit vs Sales:")
print("t-statistic:", t_statistic_profit)
print("p-value:", p_value_profit)
alpha = 0.05
# Compare p-value to significance level
if p_value_profit < alpha:
    print("Reject null hypothesis: There is a significant difference in average␣
 ↪sales between orders with positive and negative profit.")
else:
    print("Fail to reject null hypothesis: There is no significant difference␣
 ↪in average sales between orders with positive and negative profit.")
```

```
Profit vs Sales:
t-statistic: 8.759604177903055
p-value: 2.3082619533422998e-18
Reject null hypothesis: There is a significant difference in average sales
between orders with positive and negative profit.
```

[176]:
```
# Calculate Pearson correlation coefficient and p-value
correlation_coefficient, p_value_quantity = pearsonr(df['Quantity'],␣
 ↪df['Sales'])

# Print correlation coefficient and p-value
print("Quantity vs Sales:")
print("Correlation coefficient:", correlation_coefficient)
print("p-value:", p_value_quantity)

# Compare p-value to significance level
if p_value_quantity < alpha:
    print("Reject null hypothesis: There is a significant correlation between␣
 ↪quantity and sales.")
else:
    print("Fail to reject null hypothesis: There is no significant correlation␣
 ↪between quantity and sales.")
```

```
Quantity vs Sales:
Correlation coefficient: 0.24355397761476383
p-value: 9.612120704587255e-125
Reject null hypothesis: There is a significant correlation between quantity and
sales.
```

[178]:
```
# Perform ANOVA test
category_groups = df.groupby('Category')['Sales'].apply(list)
f_statistic, p_value = f_oneway(*category_groups)

print("F-statistic:", f_statistic)
print("p-value:", p_value)
```

```
F-statistic: 372.7418591513791
p-value: 2.122671756864925e-156
```

[179]:
```
# Perform ANOVA test
region_groups = df.groupby('Region')['Sales'].apply(list)
f_statistic, p_value = f_oneway(*region_groups)

print("F-statistic:", f_statistic)
print("p-value:", p_value)
```

```
F-statistic: 4.431624058674439
p-value: 0.004057174733622733
```

27 Probability Analysis:

28 What is the probability of an order being shipped using the "Standard Class" mode?

[180]:
```
df["Ship Mode"].value_counts()
```

[180]:
```
Standard Class    5475
Second Class      1817
First Class       1427
Same Day           512
Name: Ship Mode, dtype: int64
```

[181]:
```
print("Total count ",len(df["Ship Mode"]))
```

```
Total count  9231
```

[182]:
```
print("Standard Class ",len(df[df["Ship Mode"]=="Standard Class"]))
```

```
Standard Class  5475
```

[183]:
```
print("Probability of an order being shipped using the Standard Class␣
↪mode",len(df[df["Ship Mode"]=="Standard Class"])/len(df["Ship Mode"]))
```

```
Probability of an order being shipped using the Standard Class mode
0.5931101722456938
```

29 Given that an order is shipped using "Second Class," what is the probability it is from the West region?

[184]:
```
print("Total no of order shipped using second class",len(df[df["Ship␣
↪Mode"]=="Second Class"]))
```

```
Total no of order shipped using second class 1817
```

[185]:
```
print("Number of order shipped using second class and is from west␣
↪region",len(df[(df["Ship Mode"]=="Second Class")&(df["Region"]=="West")]))
```

```
Number of order shipped using second class and is from west region 603
```

[186]:
```
print("Probability that an order is shipped using Second Class ,it is from the␣
↪West region ",len(df[(df["Ship Mode"]=="Second␣
↪Class")&(df["Region"]=="West")])/len(df[df["Ship Mode"]=="Second Class"]))
```

```
Probability that an order is shipped using Second Class ,it is from the West
region  0.3318657127132636
```

30 What is the probability of a customer belonging to the "Corporate" segment?

[187]:
```
df["Segment"].value_counts()
```

[187]:
```
Consumer       4795
Corporate      2780
Home Office    1656
Name: Segment, dtype: int64
```

[188]:
```
print("Total number of customers ",len(df))
```

```
Total number of customers  9231
```

[189]:
```
print("Total number of customers from Corporate Segment␣
↪",len(df[df["Segment"]=="Corporate"]))
```

```
Total number of customers from Corporate Segment  2780
```

[190]:
```
print("Probability of a customer belonging to the Corporate segment␣
↪",len(df[df["Segment"]=="Corporate"])/len(df))
```

```
Probability of a customer belonging to the Corporate segment
0.30115913768822444
```

31 If a customer is from the "Home Office" segment, what is the probability they are from the East region?

[191]:
```
print("Total number of customers from home ofiice and east region␣
↪",len(df[(df["Segment"]=="Home Office")&(df["Region"]=="East")]))
```

```
Total number of customers from home ofiice and east region  463
```

[192]:
```
print("Total number of customers from home office segment␣
↪",len(df[df["Segment"]=="Home Office"]))
```

```
Total number of customers from home office segment  1656
```

[193]:
```
print("Probability that customer is from the Home Office segment and are from␣
↪the East region",len(df[(df["Segment"]=="Home␣
↪Office")&(df["Region"]=="East")])/len(df[df["Segment"]=="Home Office"]))
```

```
Probability that customer is from the Home Office segment and are from the East
region 0.27958937198067635
```

32 What is the probability of a product having a discount greater than 20%?

[194]:
```
print("Total number of Dicount greater than 20%: ",len(df[df["Discount"]>20/
↪100]))
```

```
Total number of Dicount greater than 20%:  876
```

[195]:
```
print("Total number of products: ",len(df))
```

```
Total number of products:  9231
```

[196]:
```
print("Probability of a product having a discount greater than 20%:␣
↪",len(df[df["Discount"]>20/100])/len(df))
```

```
Probability of a product having a discount greater than 20%:
0.09489762755931101
```

33 Given that a product has a discount, what is the probability that it is from the "Office Supplies" category?

[197]:
```
print("Total number of products that as discount and are from office supplies␣
↪category: ",len(df[(df["Category"]=="Office Supplies")&(df["Discount"]>0)]))
```

```
Total number of products that as discount and are from office supplies category:
2645
```

[198]:
```
print("Total number of products that have discount:␣
↪",len(df[(df["Discount"]>0)]))
```

```
Total number of products that have discount:  4501
```

[199]:
```
print("Probability that a product with a discount is from the 'Office Supplies'␣
↪category: ",len(df[(df["Category"]=="Office Supplies")&(df["Discount"]>0)])/
↪len(df[(df["Discount"]>0)]))
```

```
Probability that a product with a discount is from the 'Office Supplies'
category:  0.5876471895134414
```

34 What is the probability of a product having a negative profit?

[200]:
```
print("Total number of products: ",len(df["Product Name"]))
```

```
Total number of products:  9231
```

[201]:
```
print("Total number of products having negative profit:␣
↪",len(df[df["Profit"]<0]))
```

```
Total number of products having negative profit:  1235
```

[202]:
```
print("Probability of a product having a negative profit:␣
↪",len(df[df["Profit"]<0])/len(df["Product Name"]))
```

```
Probability of a product having a negative profit:  0.1337883219586177
```

35 Given that a product is in the "Furniture" category, what is the probability it has a positive profit?

[203]:
```
print("Total number of products in the furniture category:␣
↪",len(df[df["Category"]=="Furniture"]))
```

```
Total number of products in the furniture category:  1772
```

[204]:
```
print("Total number of products in the furniture category that has positive␣
↪profit :",len(df[(df["Category"]=="Furniture")&(df["Profit"]>0)]))
```

```
Total number of products in the furniture category that has positive profit :
1352
```

[205]:
```
print("Probability that a product in the 'Furniture' category has a positive␣
↪profit: ",len(df[(df["Category"]=="Furniture")&(df["Profit"]>0)])/
↪len(df[df["Category"]=="Furniture"]))
```

```
Probability that a product in the 'Furniture' category has a positive profit:
0.7629796839729119
```

36 What is the probability of an order being shipped to California?

[206]:
```
print("Probability of an order being shipped to California:␣
↪",len(df[df["State"]=="California"])/len(df))
```

```
Probability of an order being shipped to California:  0.210486404506554
```

37 Given that an order is shipped to New York, what is the probability it is from the "Consumer" segment?

[207]:
```
print("Probability that an order shipped to New York is from the 'Consumer'␣
↪segment: ",len(df[(df["State"]=="New York")&(df["Segment"]=="Consumer")])/
↪len(df[df["State"]=="New York"]))
```

```
Probability that an order shipped to New York is from the 'Consumer' segment:
0.5743305632502308
```