Project Title: Super Store Sales Analysis and Visualization

Problem Statement:

Analyze the sales data of a Super Store to uncover insights and trends that can help improve business decisions and strategies.

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv('/content/1719219914-Analysis of Super Store - DA
(1).csv')  # Update the path with the location where you uploaded the
file in Google Colab

# Display the first few rows of the dataframe
print("First few rows of the dataset:")
print(df.head())
```

```
First few rows of the dataset:
        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

   Postal Code Region         Category Sub-Category     Sales
Quantity  \
0        42420  South        Furniture    Bookcases  261.9600
2
1        42420  South        Furniture       Chairs  731.9400
3
2        90036   West  Office Supplies       Labels   14.6200
2
```

```
3        33311  South        Furniture       Tables  957.5775
5
4        33311  South  Office Supplies      Storage   22.3680
2

   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
```

```
# Drop unnecessary columns (if any)
# For example, let's drop a column named 'Unnamed: 0' if it exists
if 'Unnamed: 0' in df.columns:
    df = df.drop('Unnamed: 0', axis=1)

# Display unique values for each column
print("\nUnique values for each column:")
for column in df.columns:
    print(f"{column}: {df[column].unique()}")
```

```
Unique values for each column:
Ship Mode: ['Second Class' 'Standard Class' 'First Class' 'Same Day']
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' 'Minneapolis' 'Portland' 'Saint Paul' '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' 'Redlands' 'Hamilton' 'Westfield' 'Akron'
 'Denver' 'Dallas' 'Whittier' 'Saginaw' 'Medina' 'Dublin' 'Detroit'
 'Tampa' 'Santa Clara' 'Lakeville' 'San Diego' 'Brentwood' 'Chapel
Hill'
 'Morristown' 'Cincinnati' 'Inglewood' 'Tamarac' 'Colorado Springs'
 'Belleville' 'Taylor' 'Lakewood' 'Arlington' 'Arvada' 'Hackensack'
 'Saint Petersburg' 'Long Beach' 'Hesperia' 'Murfreesboro' 'Layton'
 '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' 'Norman' 'Park
Ridge'
 'Amarillo' 'Lindenhurst' 'Huntsville' 'Fayetteville' 'Costa Mesa'
 'Parker' 'Atlanta' 'Gladstone' 'Great Falls' 'Lakeland' 'Montgomery'
 'Mesa' 'Green Bay' 'Anaheim' 'Marysville' 'Salem' 'Laredo' 'Grove
City'
 'Dearborn' 'Warner Robins' 'Vallejo' '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' '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'
 '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' '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' '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' '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' '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' 'Beaumont' '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' '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' 'Oregon' 'Colorado' 'Iowa'
'Ohio'
 'Missouri' 'Oklahoma' 'New Mexico' 'Louisiana' 'Connecticut' 'New
Jersey'
 '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 55407 97206 55106 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
92374
 45011  7090 19120 44312 80219 75220 37064 90604 48601 44256 43017
48227
 38401 33614 95051 55044 92037 77506 94513 27514  7960 45231 94110
90301
 33319 80906  7109 48180  8701 22204 80004  7601 33710 19143 90805
92345
 37130 84041 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 73071 94521 60068 79109 11757 90008 92024
77340
 14609 72701 92627 80134 30318 64118 59405 48234 33801 36116 85204
60653
 54302 45503 92804 98270 97301 78041 75217 43123 10011 48126 31088
94591
 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 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 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 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 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
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
 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 77705 33407 79605 61761 63301 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
33134
 60441  1752 20707 77803 71603 57401 21740  7017 60004 60543 55433
92672
 94568 93405 72762 95240 77571 45040 30188]
Region: ['South' 'West' 'Central' 'East']
Category: ['Furniture' 'Office Supplies' 'Technology']
Sub-Category: ['Bookcases' 'Chairs' 'Labels' 'Tables' 'Storage'
'Furnishings' 'Art'
 'Phones' 'Binders' 'Appliances' 'Paper' 'Accessories' 'Envelopes'
 'Fasteners' 'Supplies' 'Machines' 'Copiers']
Sales: [261.96  731.94   14.62  ... 437.472  97.98  243.16 ]
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 ...  16.124    4.1028  72.948 ]
```

```
# Display summary statistics
print("\nSummary statistics:")
print(df.describe())
```

```
Summary statistics:
       Postal Code         Sales     Quantity     Discount
Profit
count   9994.000000   9994.000000  9994.000000  9994.000000
9994.000000
mean   55190.379428    229.858001     3.789574     0.156203
28.656896
std    32063.693350    623.245101     2.225110     0.206452
234.260108
min     1040.000000      0.444000     1.000000     0.000000 -
6599.978000
25%    23223.000000     17.280000     2.000000     0.000000
1.728750
50%    56430.500000     54.490000     3.000000     0.200000
```

```
8.666500
75%    90008.000000    209.940000     5.000000     0.200000
29.364000
max    99301.000000  22638.480000    14.000000     0.800000
8399.976000
```

```
# Display information about the dataframe
print("\nDataframe information:")
print(df.info())
```

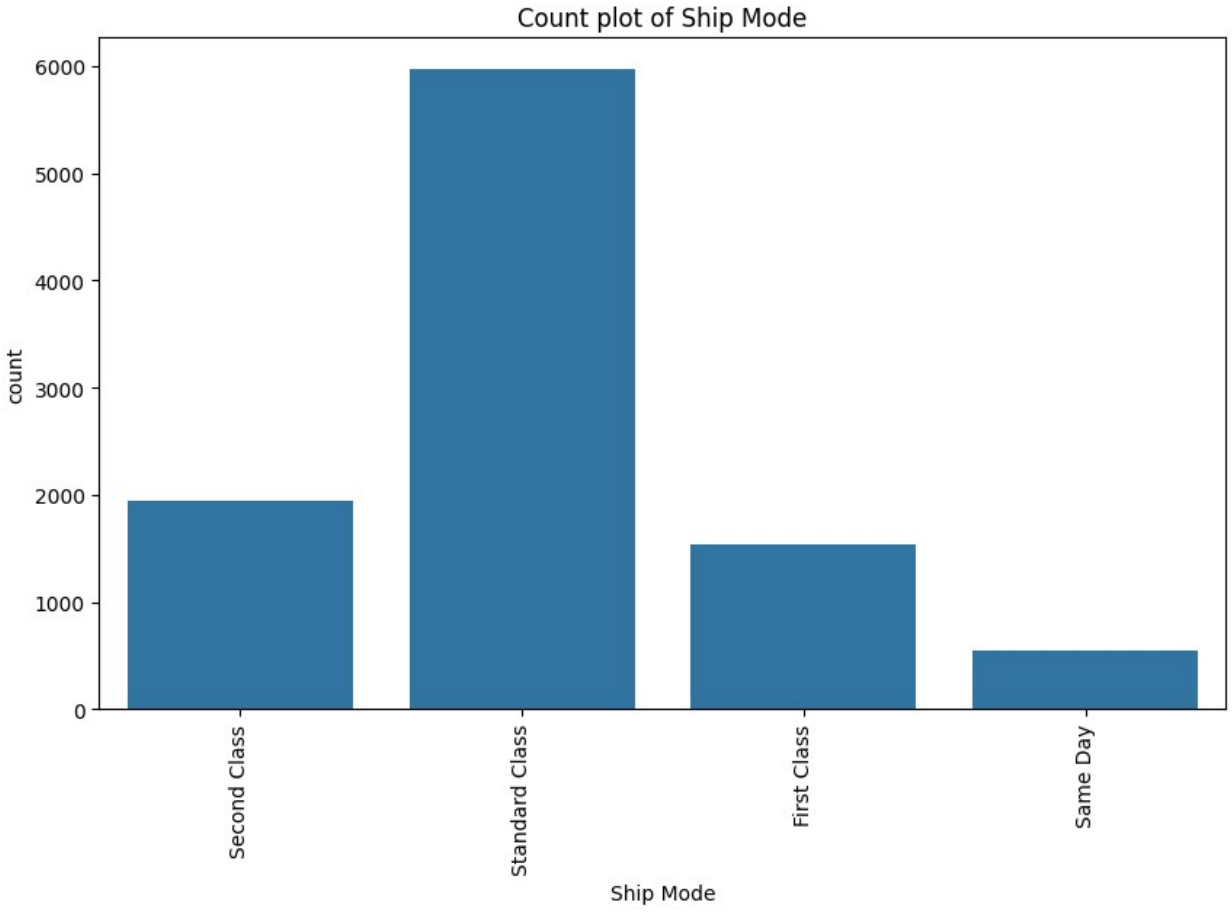
```
Dataframe information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Ship Mode     9994 non-null   object
 1   Segment       9994 non-null   object
 2   Country       9994 non-null   object
 3   City          9994 non-null   object
 4   State         9994 non-null   object
 5   Postal Code   9994 non-null   int64
 6   Region        9994 non-null   object
 7   Category      9994 non-null   object
 8   Sub-Category  9994 non-null   object
 9   Sales         9994 non-null   float64
 10  Quantity      9994 non-null   int64
 11  Discount      9994 non-null   float64
 12  Profit        9994 non-null   float64
dtypes: float64(3), int64(2), object(8)
memory usage: 1015.1+ KB
None
```

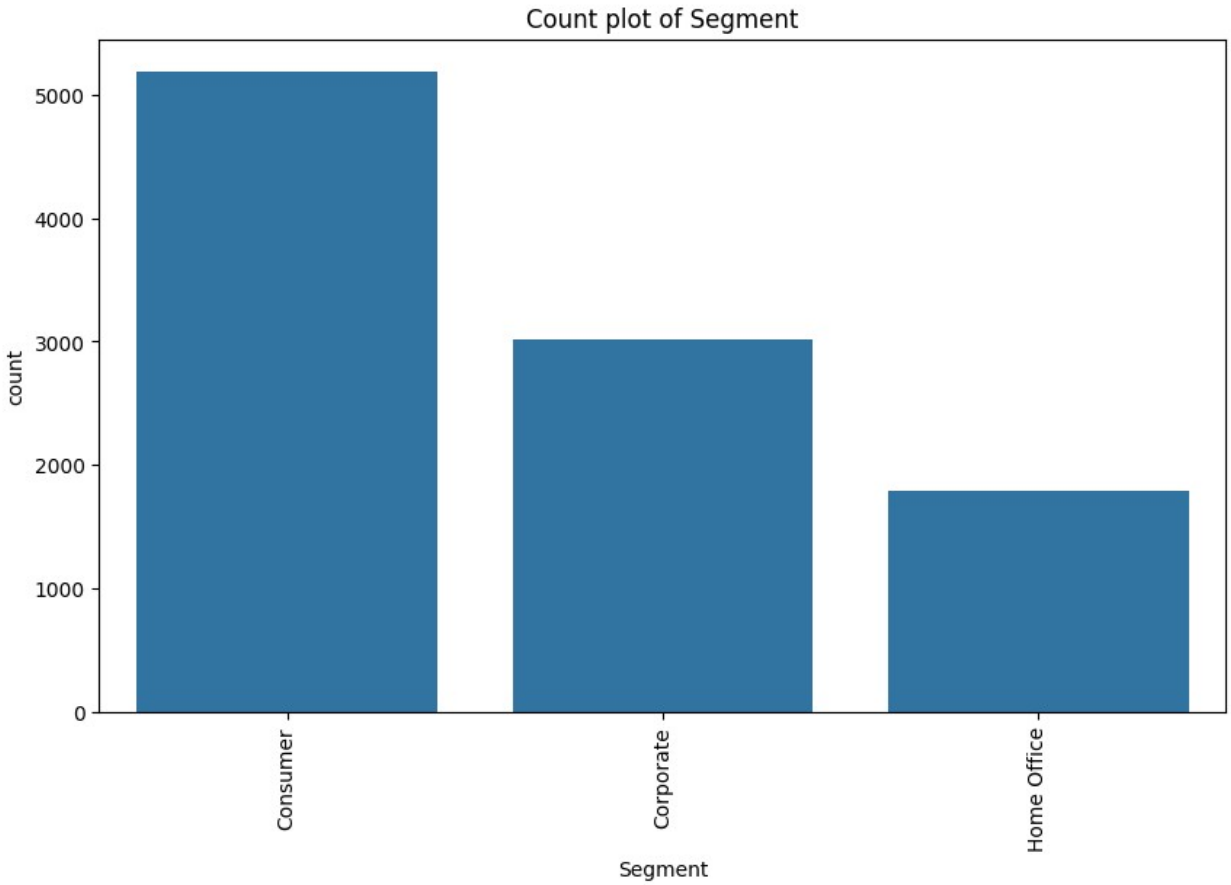
```
# Check for missing values
print("\nMissing values in the dataset:")
print(df.isna().sum())
```

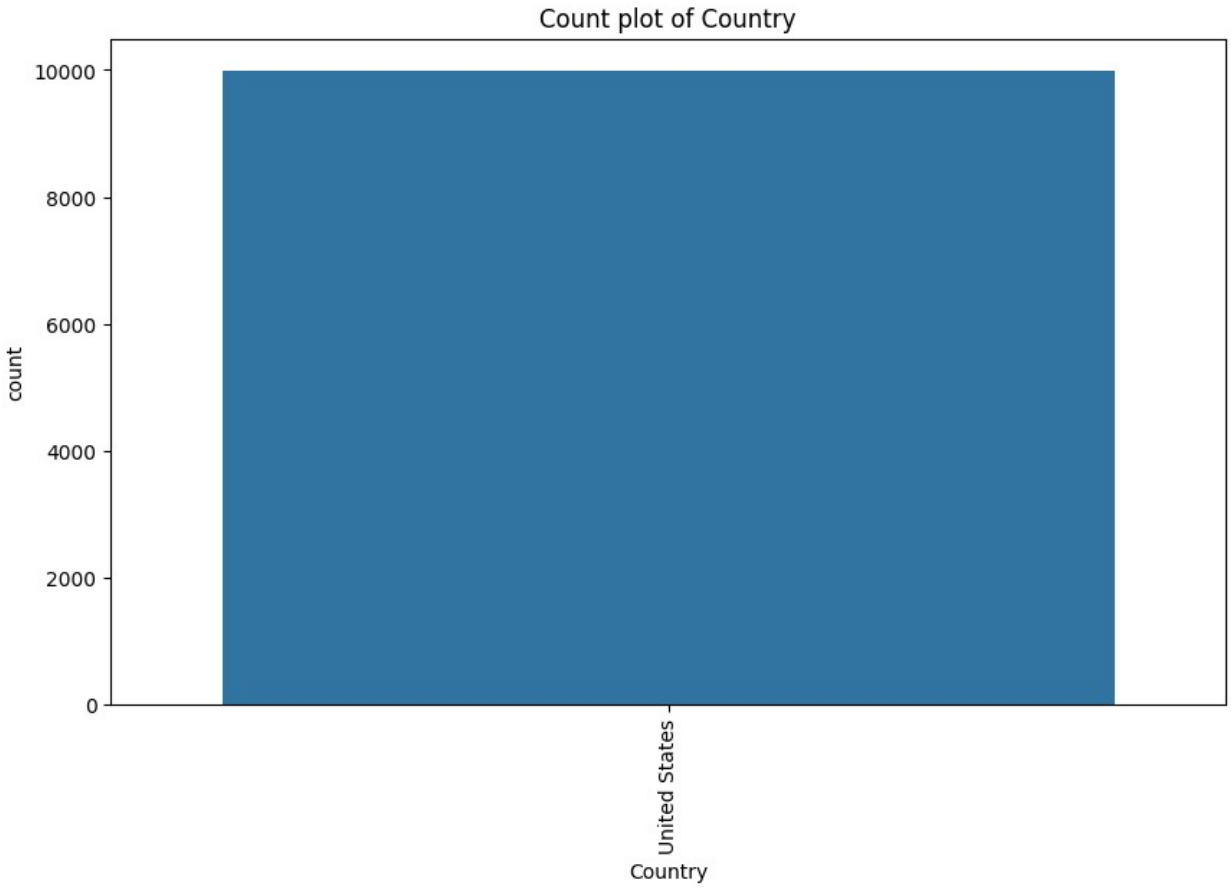
```
Missing values in the dataset:
Ship Mode       0
Segment         0
Country         0
City            0
State           0
Postal Code     0
Region          0
Category        0
Sub-Category    0
Sales           0
```

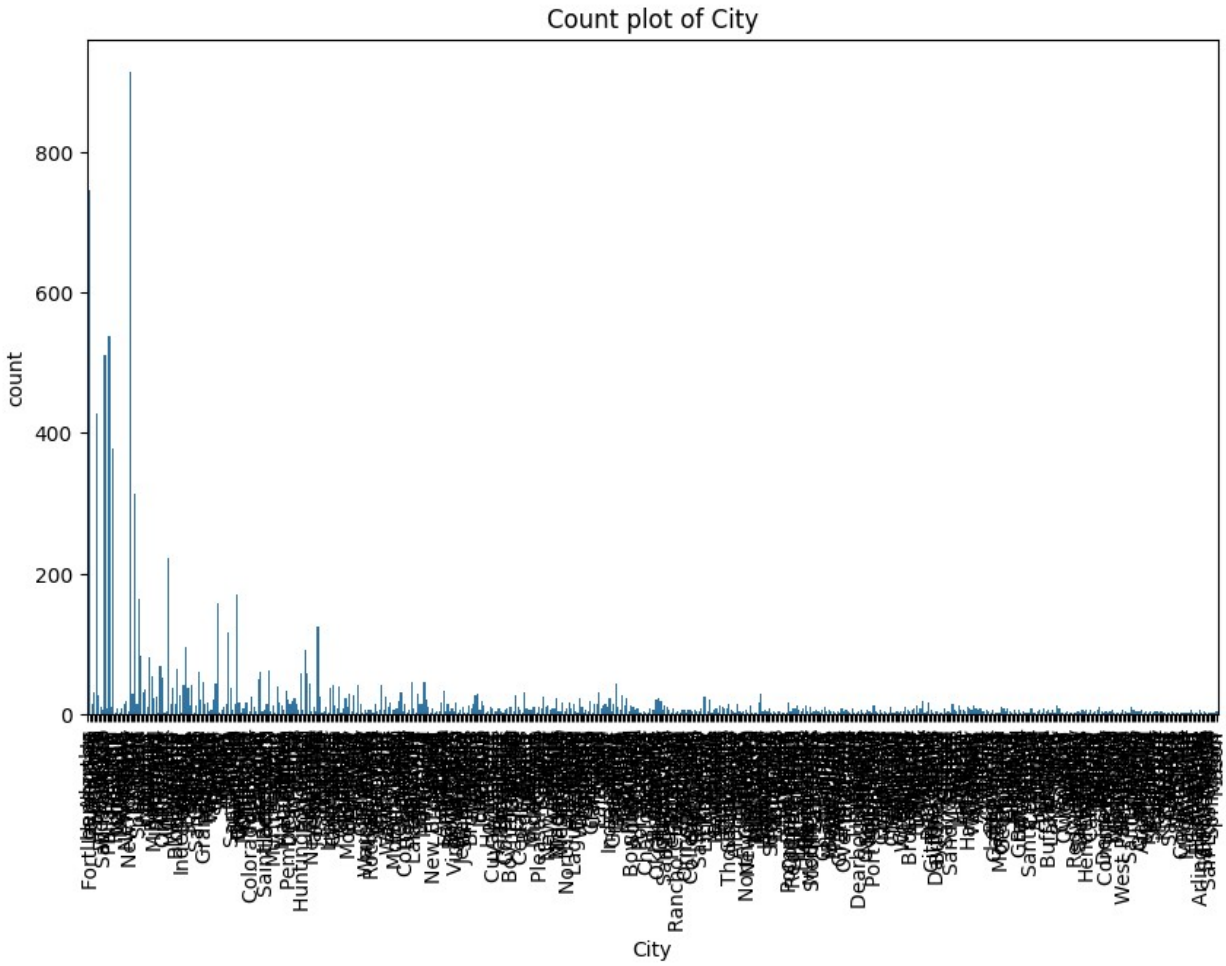
```
Quantity        0
Discount        0
Profit          0
dtype: int64
```

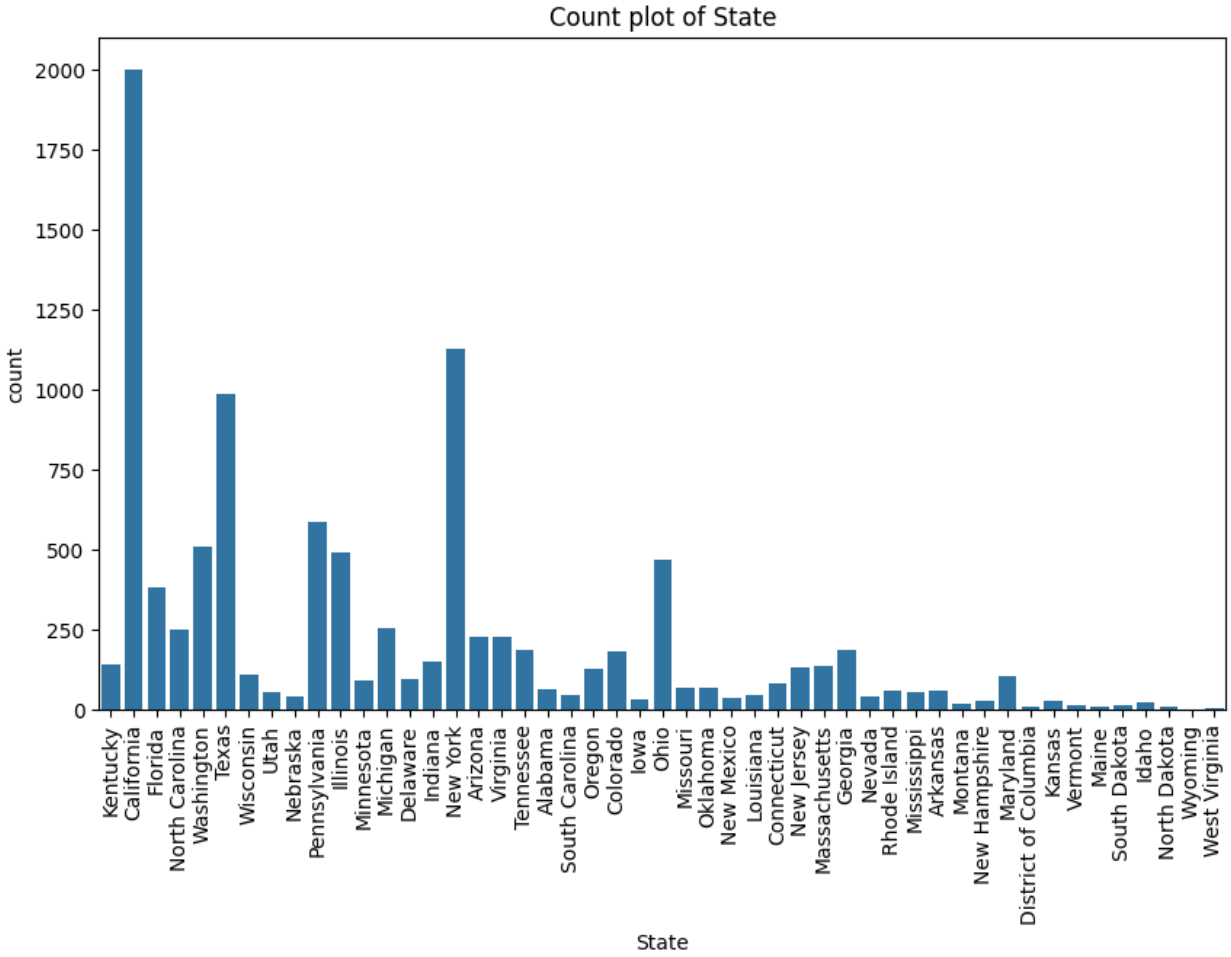
```
# Visualizations

# Bar graph for categorical columns
categorical_columns = df.select_dtypes(include=['object']).columns
for column in categorical_columns:
    plt.figure(figsize=(10, 6))
    sns.countplot(data=df, x=column)
    plt.xticks(rotation=90)
    plt.title(f'Count plot of {column}')
    plt.show()
```

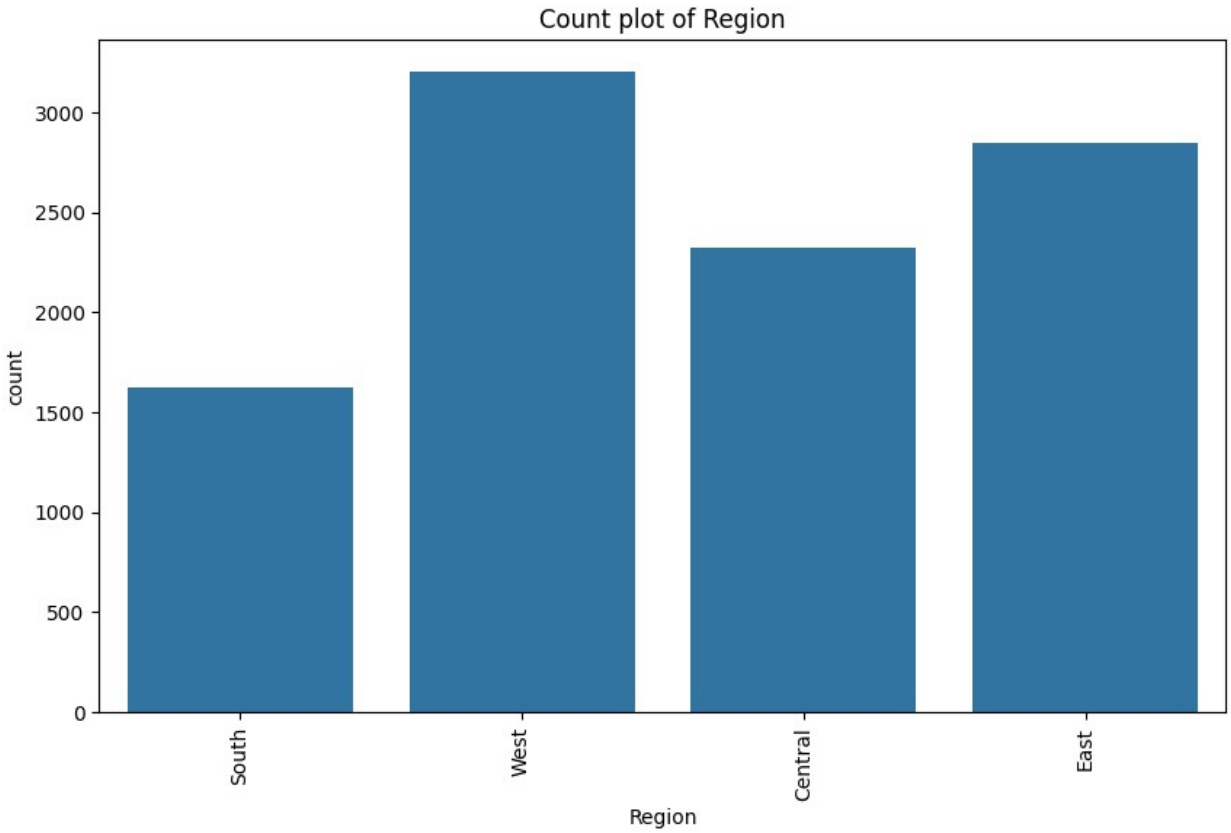


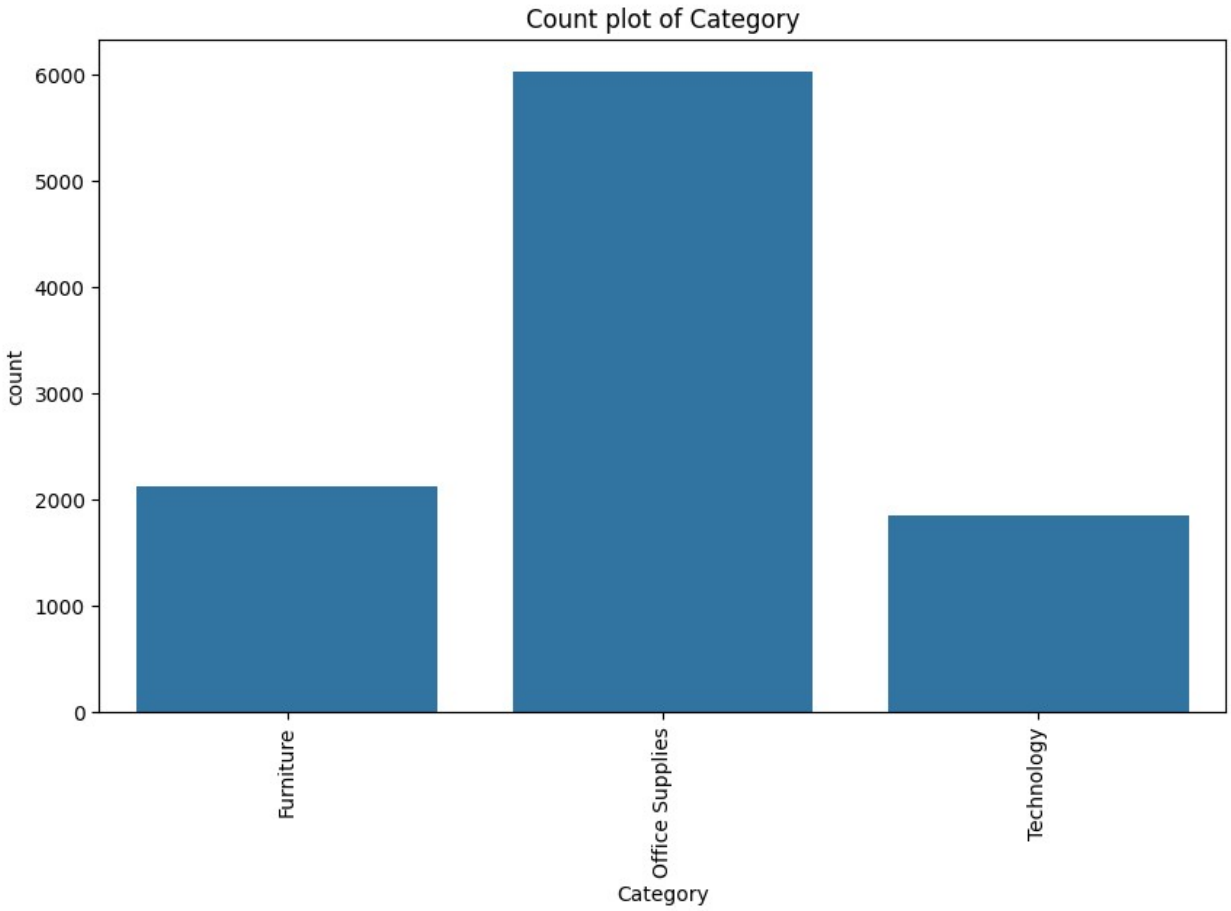

Count plot of Segment
count
5000
4000
3000
2000
1000
0
Consumer
Corporate
Home Office
Segment

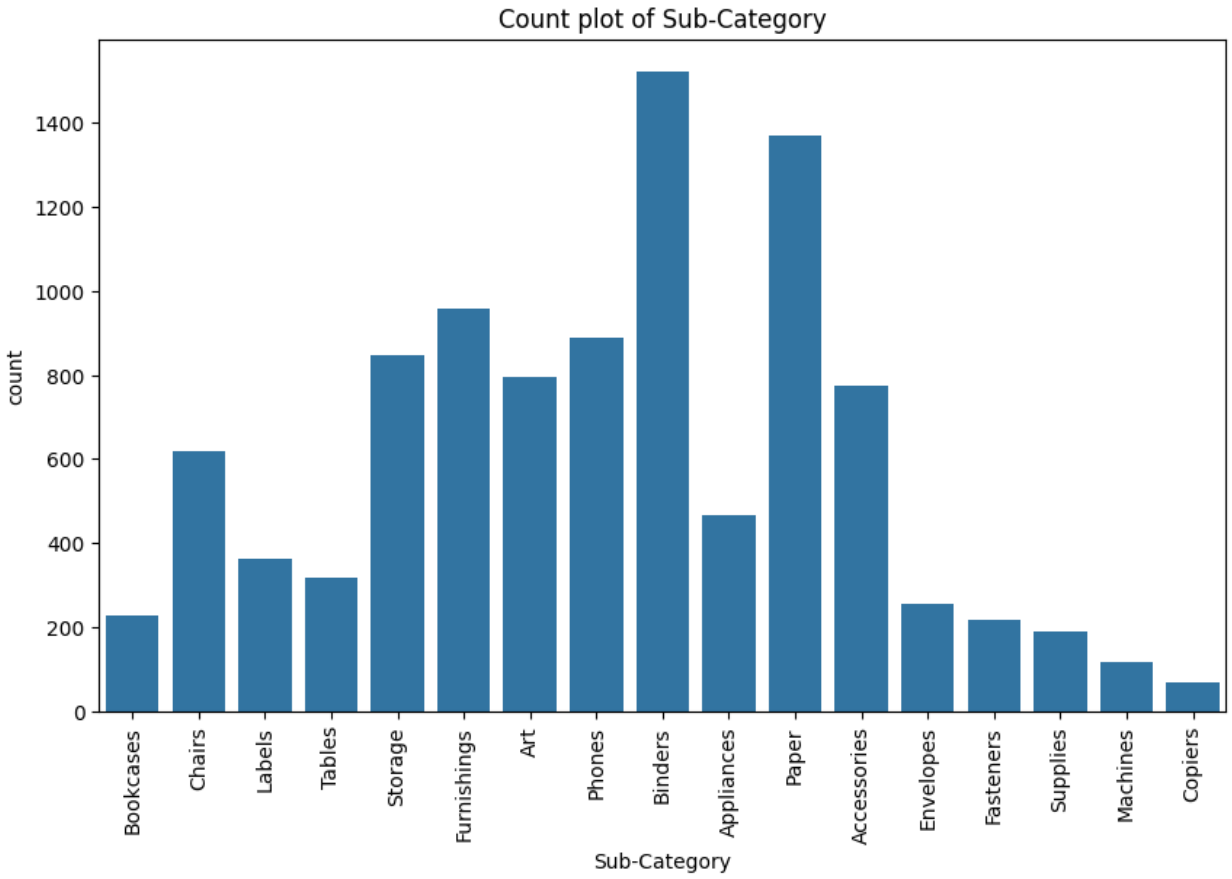

Count plot of Country
10000
8000
6000
4000
2000
0
count
United States
Country


Count plot of City
count
0
200
400
600
800
City

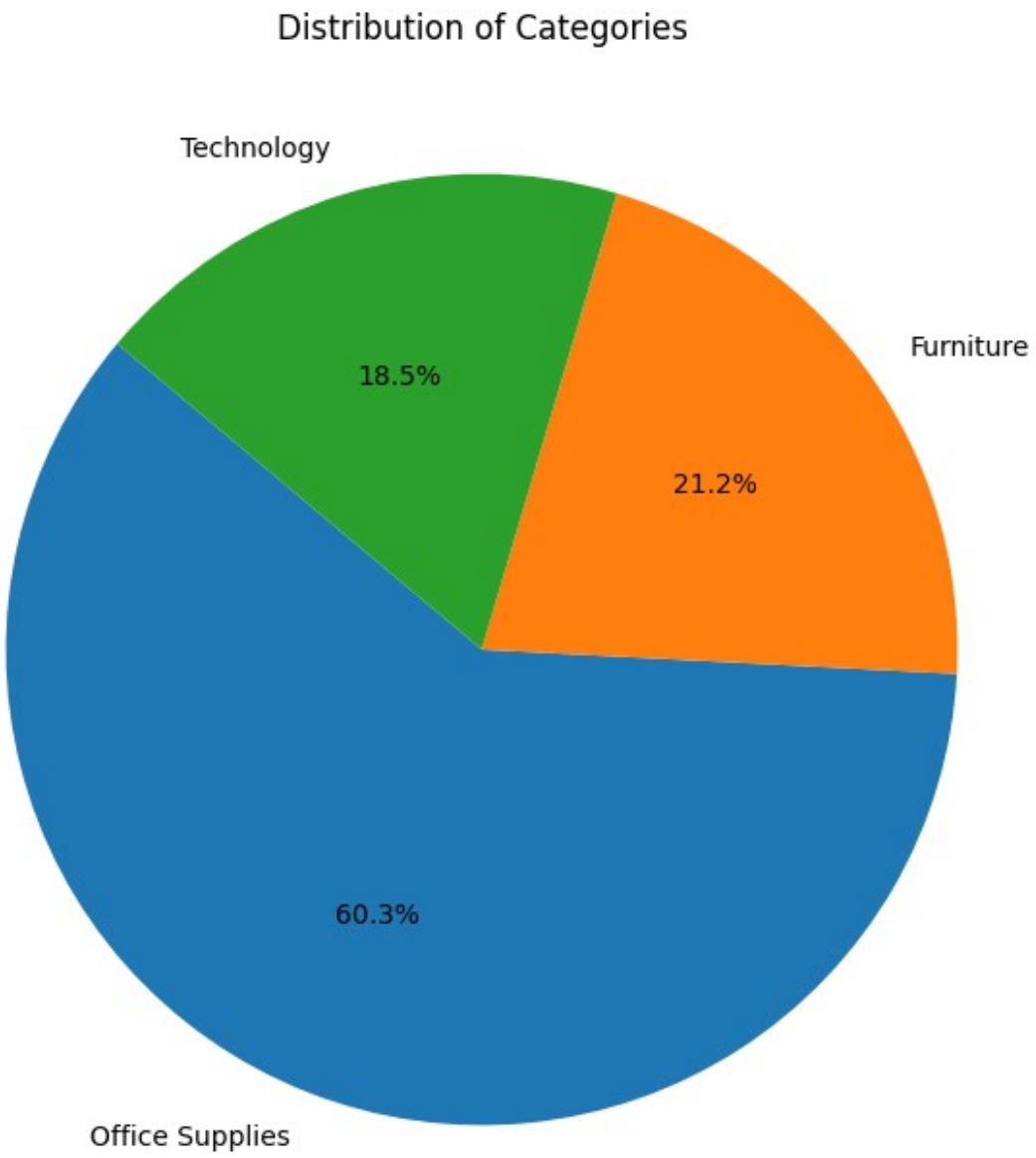

Count plot of State
count
2000
1750
1500
1250
1000
750
500
250
0
Kentucky
California
Florida
North Carolina
Washington
Texas
Wisconsin
Utah
Nebraska
Pennsylvania
Illinois
Minnesota
Michigan
Delaware
Indiana
New York
Arizona
Virginia
Tennessee
Alabama
South Carolina
Oregon
Colorado
Iowa
Ohio
Missouri
Oklahoma
New Mexico
Louisiana
Connecticut
New Jersey
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
State


Count plot of Region
count
3000
2500
2000
1500
1000
500
0
South
West
Central
East
Region

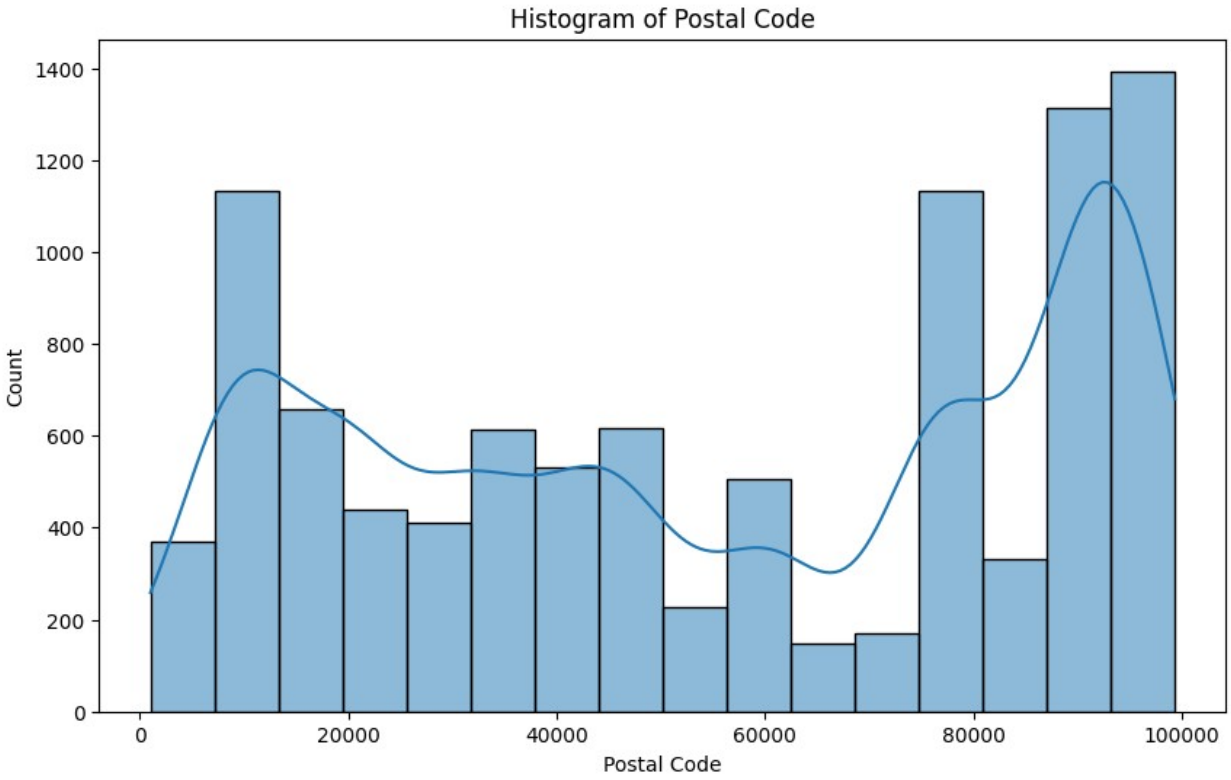

Count plot of Category
6000
5000
4000
3000
2000
1000
0
count
Furniture
Office Supplies
Technology
Category

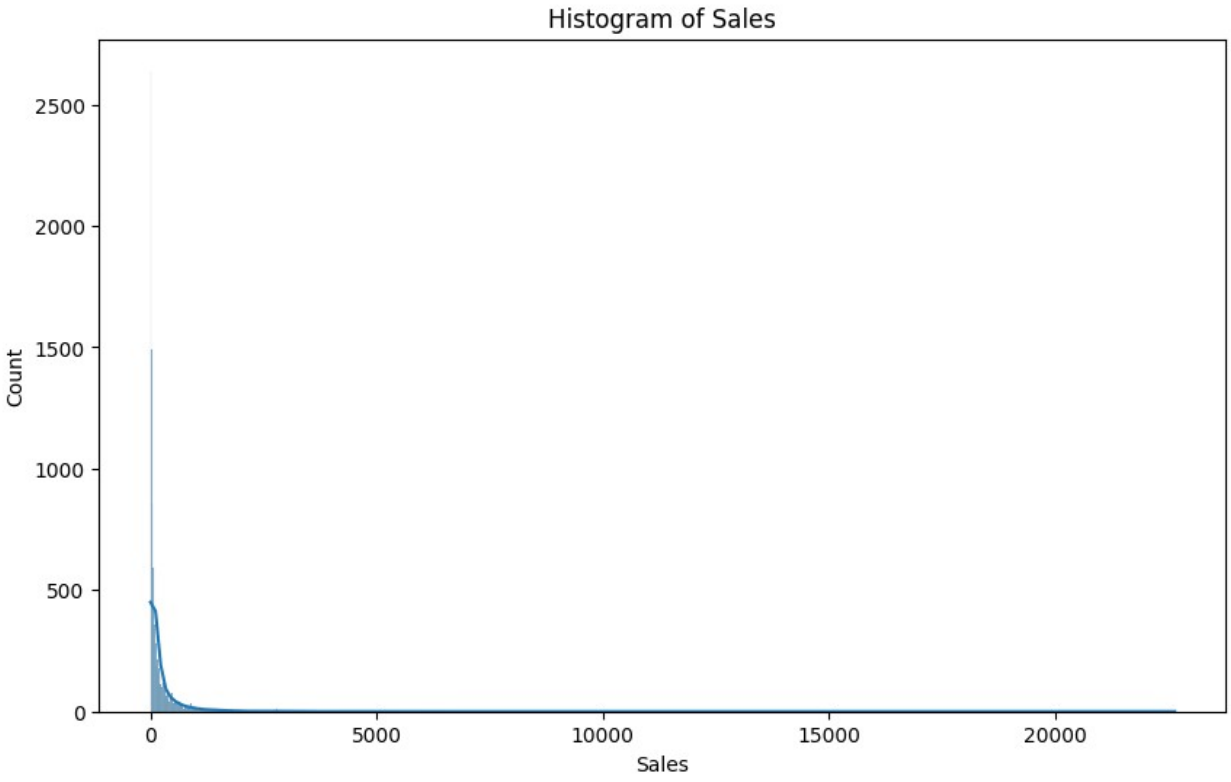


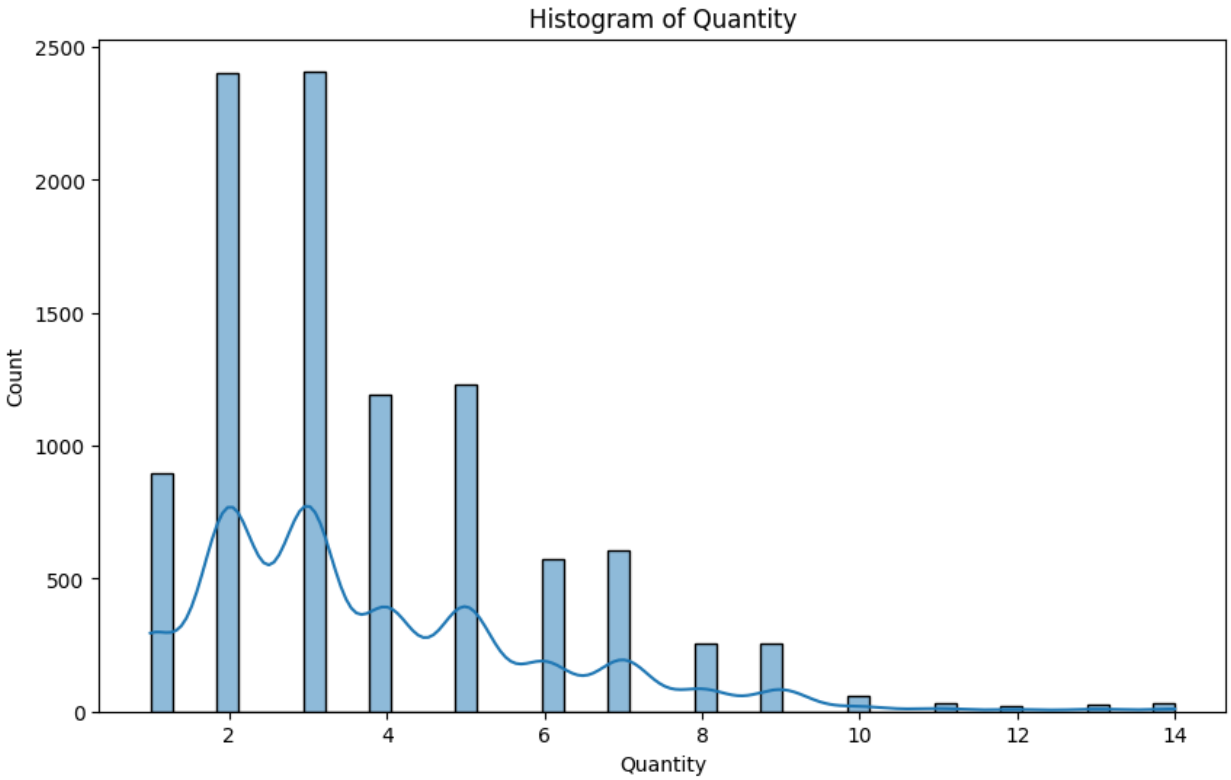

```
# Pie chart for a categorical column (e.g., 'Category')
if 'Category' in df.columns:
    plt.figure(figsize=(8, 8))
    df['Category'].value_counts().plot.pie(autopct='%1.1f%%',
startangle=140)
    plt.title('Distribution of Categories')
    plt.ylabel('')
    plt.show()
```

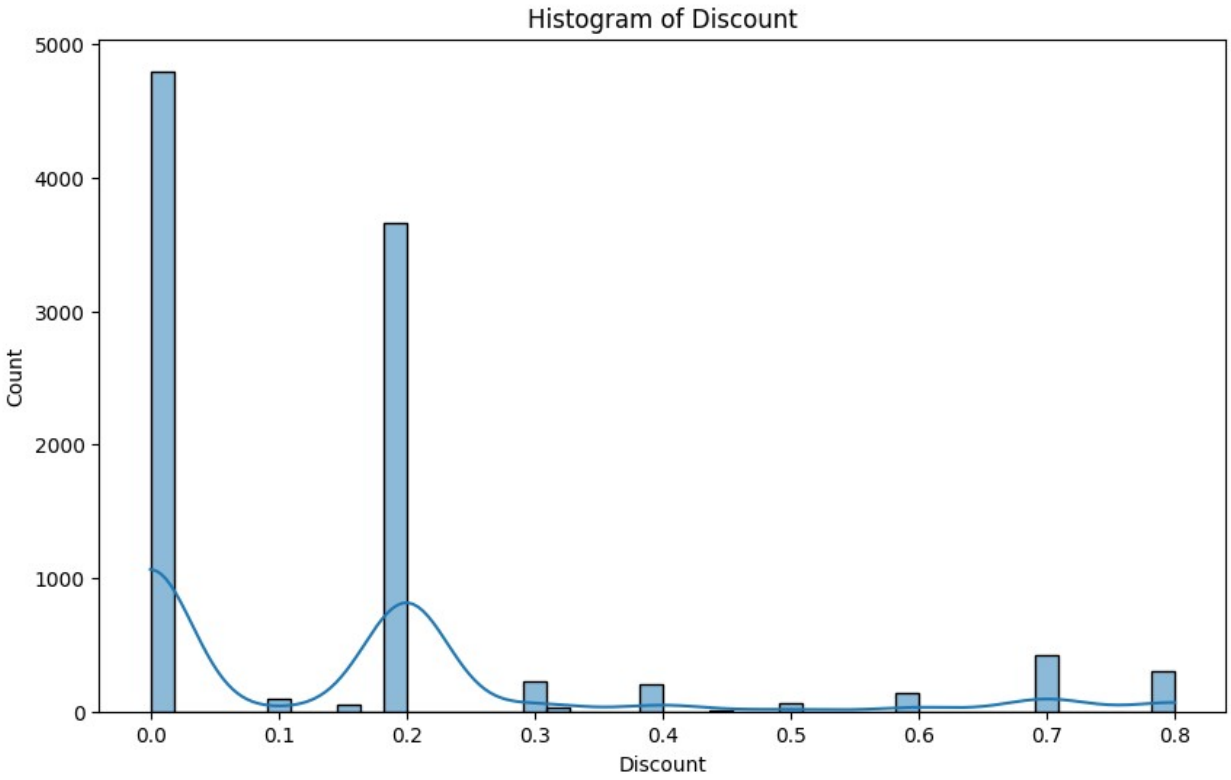


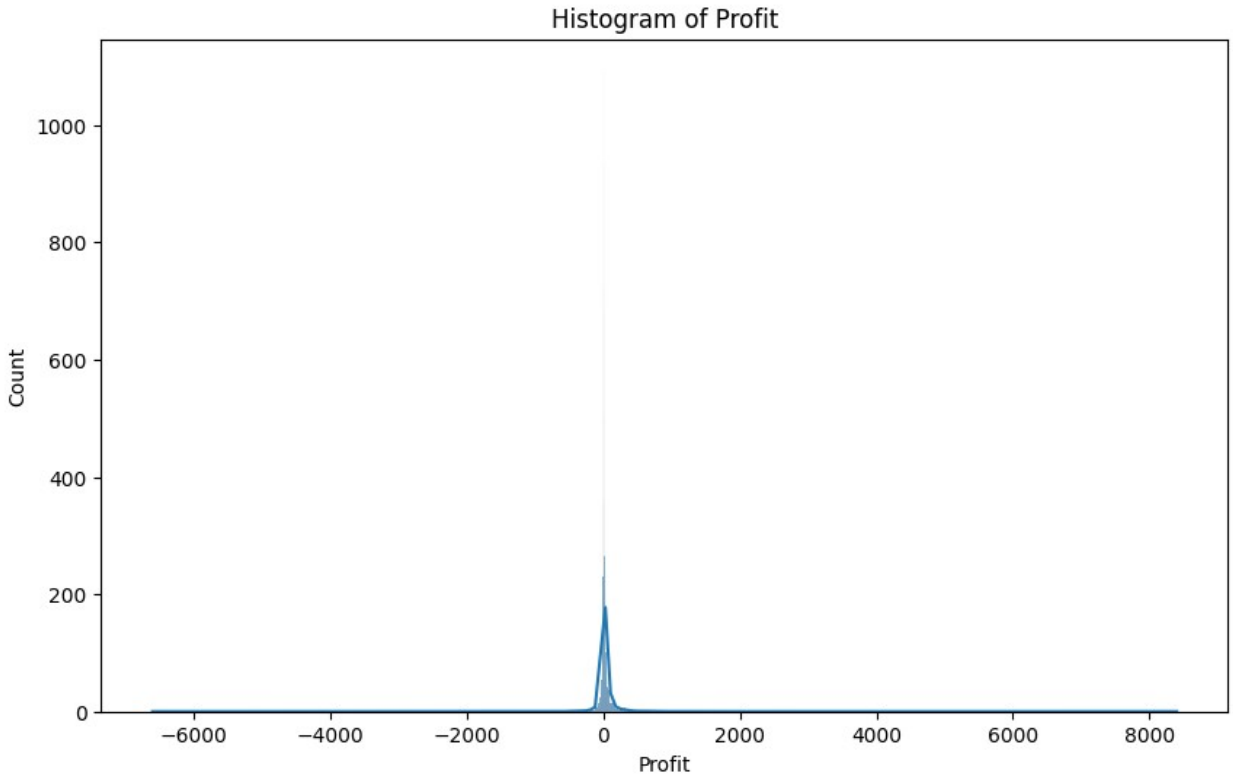

```
# Histogram for numerical columns
numerical_columns = df.select_dtypes(include=['int64',
'float64']).columns
for column in numerical_columns:
    plt.figure(figsize=(10, 6))
    sns.histplot(data=df, x=column, kde=True)
    plt.title(f'Histogram of {column}')
    plt.show()
```


Histogram of Postal Code
Count
Postal Code
0
200
400
600
800
1000
1200
1400
0
20000
40000
60000
80000
100000

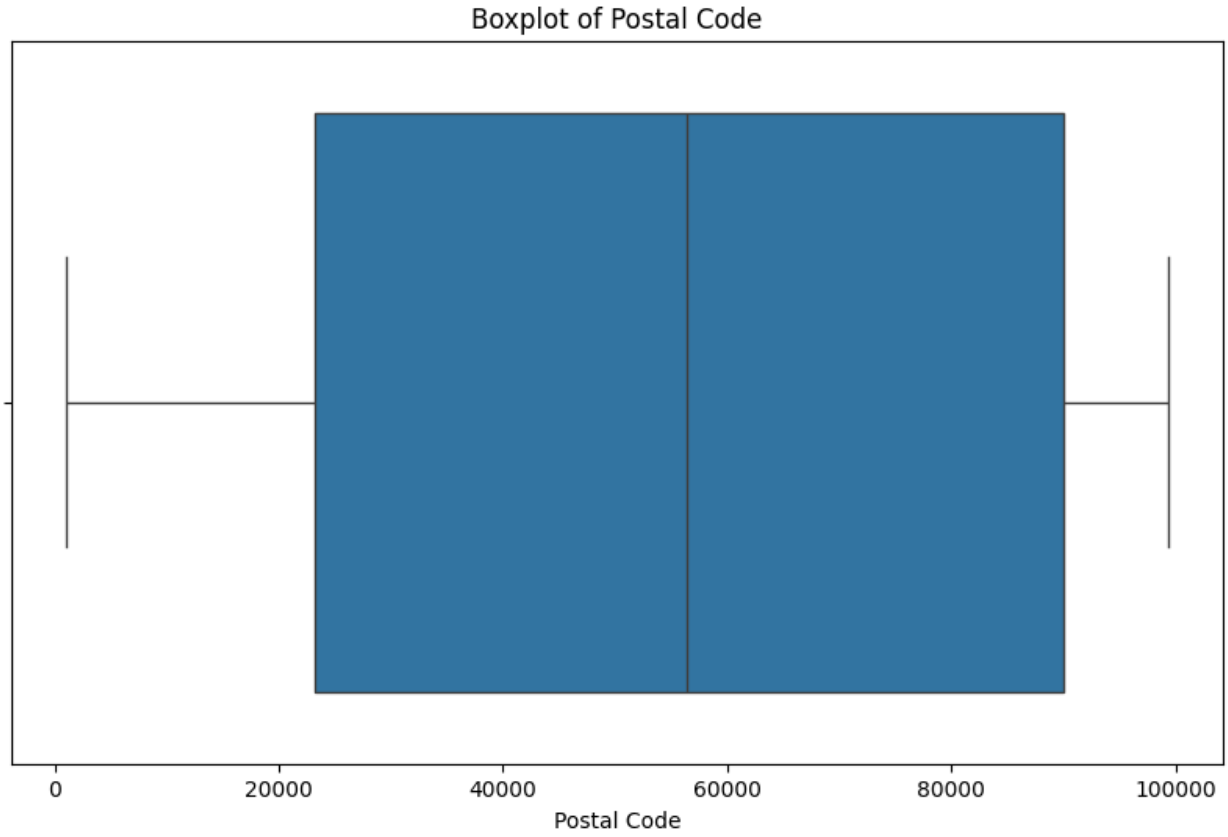

Histogram of Sales
Count
Sales
0
500
1000
1500
2000
2500
0
5000
10000
15000
20000

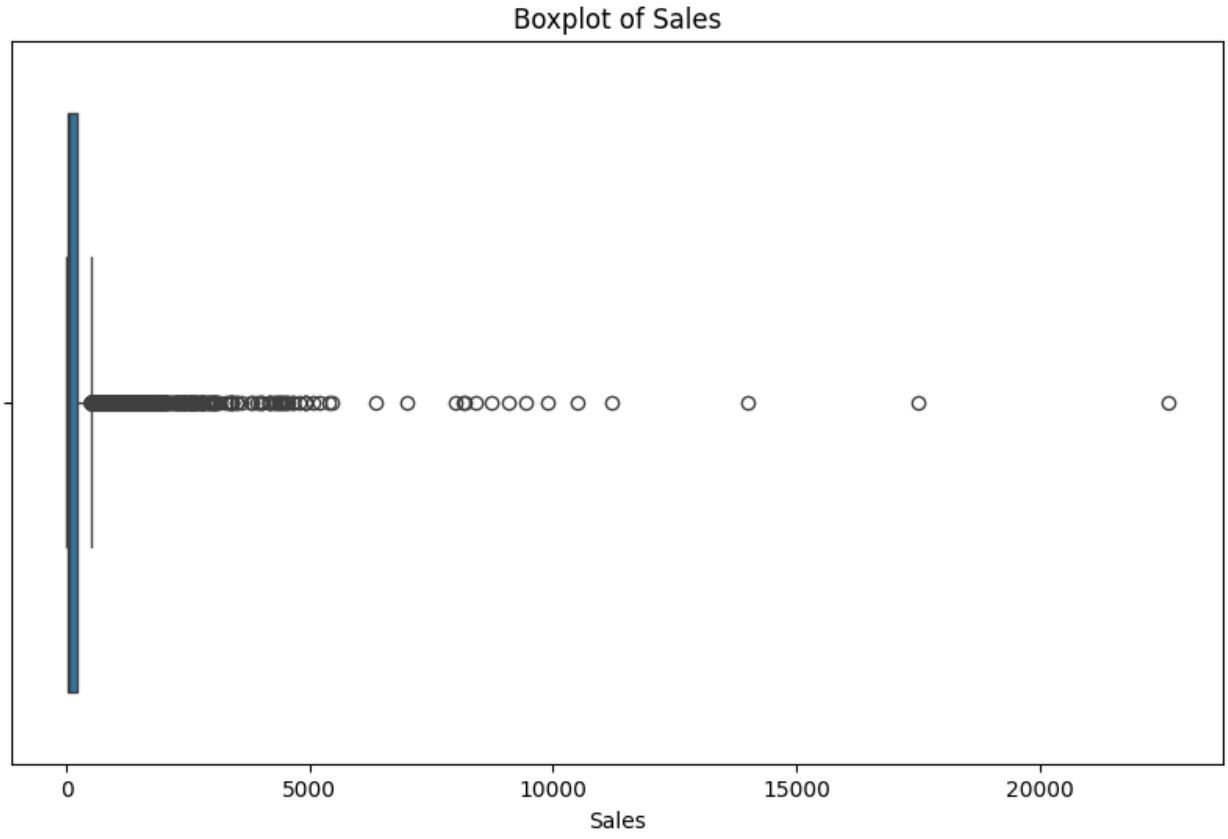

Histogram of Quantity
Count
Quantity
0
500
1000
1500
2000
2500
2
4
6
8
10
12
14

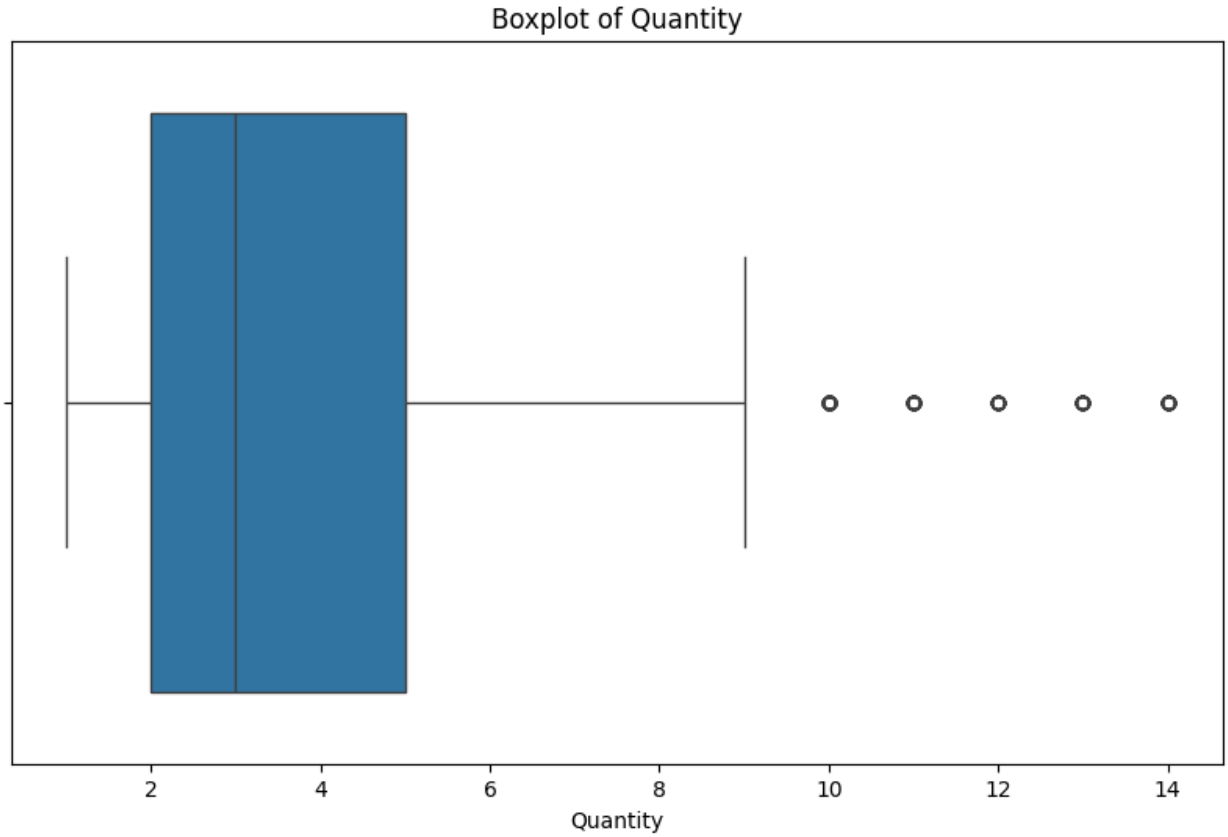

Histogram of Discount
Count
Discount
0
1000
2000
3000
4000
5000
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

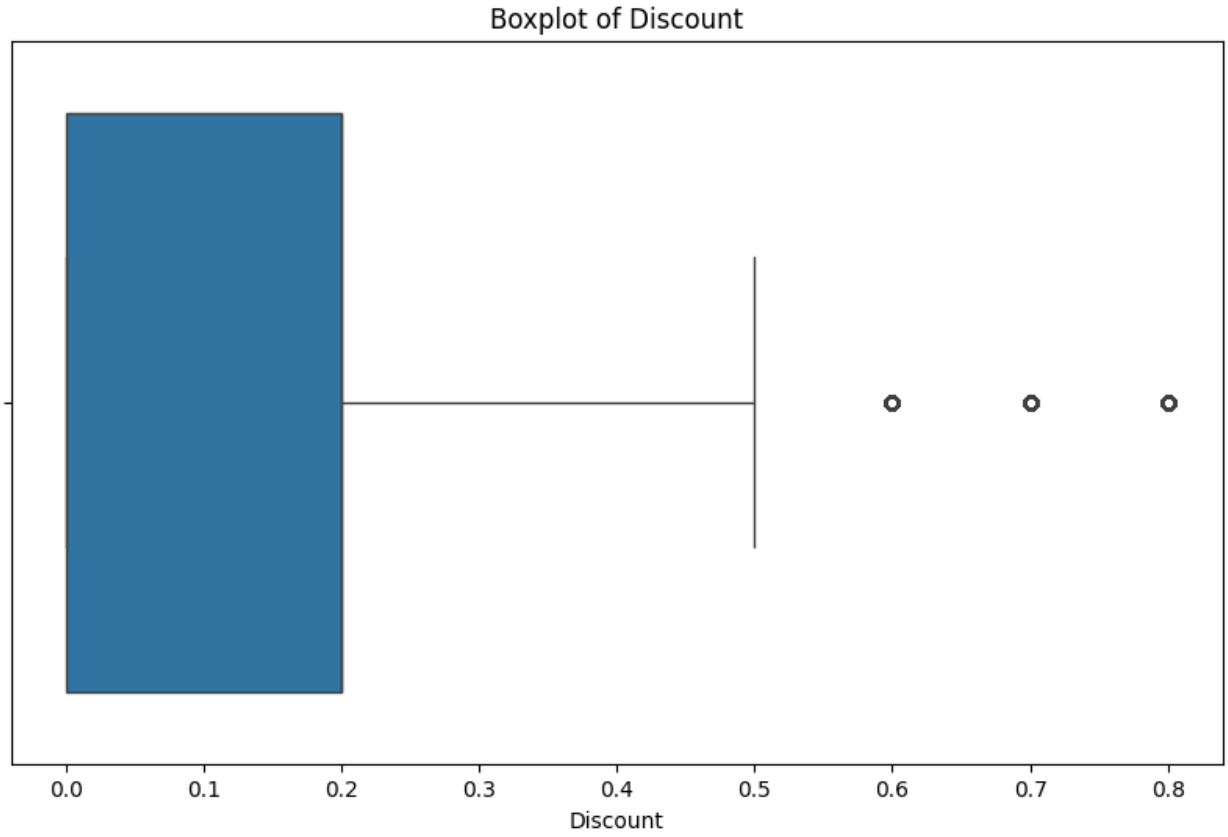


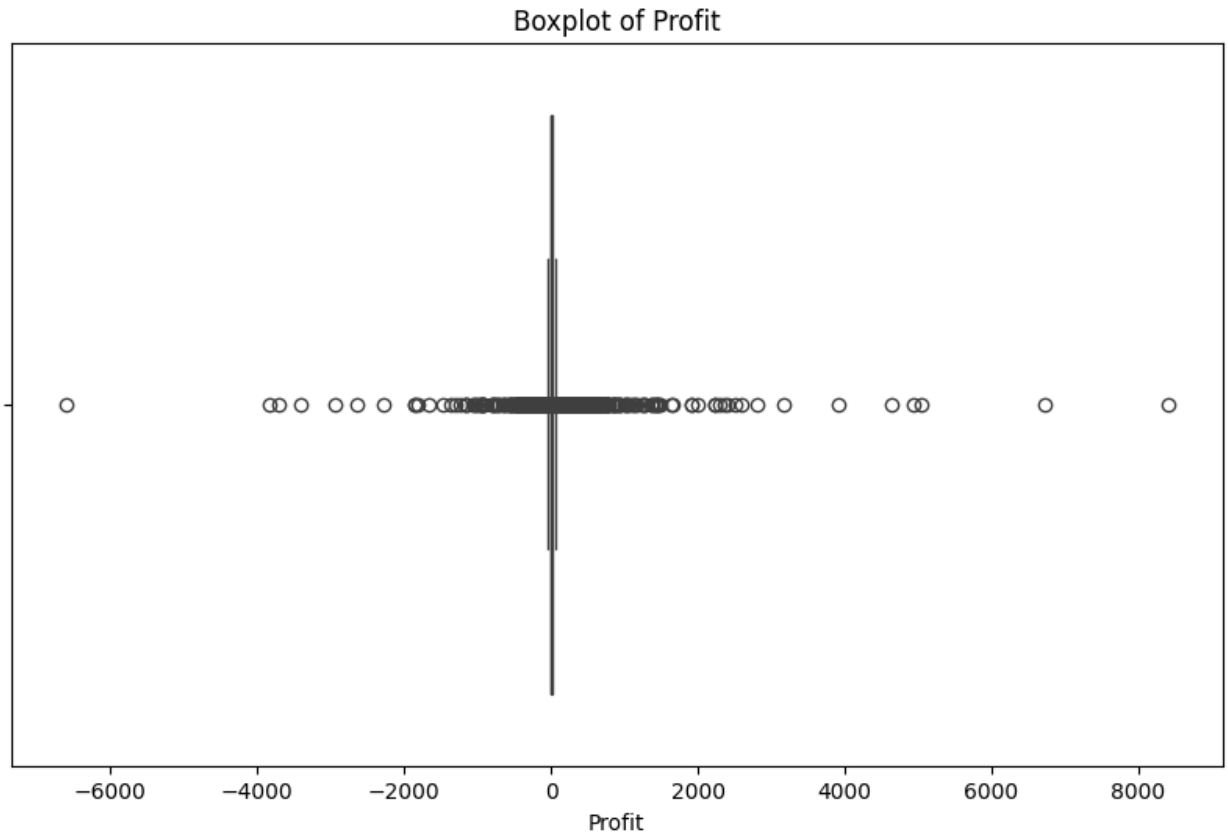

```
# Boxplot for numerical columns to check for outliers
for column in numerical_columns:
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=df, x=column)
    plt.title(f'Boxplot of {column}')
    plt.show()
```


Boxplot of Postal Code
0
20000
40000
60000
80000
100000
Postal Code


Boxplot of Sales
0
5000
10000
15000
20000
Sales


Boxplot of Quantity
2
4
6
8
10
12
14
Quantity


Boxplot of Discount
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Discount


Boxplot of Profit
−6000
−4000
−2000
0
2000
4000
6000
8000
Profit