In [99]:
```
!pip install pandas numpy matplotlib seaborn
```

```
Requirement already satisfied: pandas in /opt/anaconda3/lib/python3.12/site-
packages (2.2.2)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.12/site-p
ackages (1.26.4)
Requirement already satisfied: matplotlib in /opt/anaconda3/lib/python3.12/s
ite-packages (3.9.2)
Requirement already satisfied: seaborn in /opt/anaconda3/lib/python3.12/site
-packages (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/anaconda3/lib/
python3.12/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/anaconda3/lib/python3.1
2/site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /opt/anaconda3/lib/python3.
12/site-packages (from pandas) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in /opt/anaconda3/lib/python
3.12/site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/anaconda3/lib/python3.1
2/site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /opt/anaconda3/lib/pytho
n3.12/site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/anaconda3/lib/pytho
n3.12/site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /opt/anaconda3/lib/python
3.12/site-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in /opt/anaconda3/lib/python3.12/si
te-packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/anaconda3/lib/python
3.12/site-packages (from matplotlib) (3.1.2)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.12/sit
e-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

In [100…
```
import pandas as pd

# Load the dataset
df = pd.read_csv("Superstore Sales.csv", encoding="latin1")

# Display the first 5 rows
df.head()
```

Out[100…

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Co
0	1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	U S
1	2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	U S
2	3	CA-2016-138688	6/12/2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	U S
3	4	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	U S
4	5	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	U S

5 rows × 21 columns

In [101…

```
# Check basic info (columns, data types, missing values)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   object
 3   Ship Date      9994 non-null   object
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64
 12  Region         9994 non-null   object
 13  Product ID     9994 non-null   object
 14  Category       9994 non-null   object
 15  Sub-Category   9994 non-null   object
 16  Product Name   9994 non-null   object
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.6+ MB
```

In [102…

```
# Summary statistics for numerical columns
df.describe()
```

Out[102…

	Row ID	Postal Code	Sales	Quantity	Discount	
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	9994.
mean	4997.500000	55190.379428	229.858001	3.789574	0.156203	28.
std	2885.163629	32063.693350	623.245101	2.225110	0.206452	234
min	1.000000	1040.000000	0.444000	1.000000	0.000000	-6599
25%	2499.250000	23223.000000	17.280000	2.000000	0.000000	1
50%	4997.500000	56430.500000	54.490000	3.000000	0.200000	8.
75%	7495.750000	90008.000000	209.940000	5.000000	0.200000	29.
max	9994.000000	99301.000000	22638.480000	14.000000	0.800000	8399

In [103…

```
# Count missing values in each column
df.isnull().sum()
```

Out[103…
```
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
dtype: int64
```

In [104…
```
# Count duplicate rows
df.duplicated().sum()
```

Out[104…
```
0
```

In [105…
```
df.dtypes
```

Out[105…
```
Row ID            int64
Order ID         object
Order Date       object
Ship Date        object
Ship Mode        object
Customer ID      object
Customer Name    object
Segment          object
Country          object
City             object
State            object
Postal Code       int64
Region           object
Product ID       object
Category         object
Sub-Category     object
Product Name     object
Sales           float64
Quantity          int64
Discount        float64
Profit          float64
dtype: object
```

In [106…
```
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Ship Date'] = pd.to_datetime(df['Ship Date'])
```

```
df["Postal Code"] = df["Postal Code"].astype(str)
```

In [107…
```
# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns
print("Categorical Columns:", categorical_cols)
```

```
Categorical Columns: Index(['Order ID', 'Ship Mode', 'Customer ID', 'Custome
r Name', 'Segment',
       'Country', 'City', 'State', 'Postal Code', 'Region', 'Product ID',
       'Category', 'Sub-Category', 'Product Name'],
      dtype='object')
```

In [108…
```
# List unique values in each relevant categorical column
relevant_cats = ['Ship Mode', 'Segment', 'Country', 'City', 'State', 'Regior
                 'Category', 'Sub-Category', 'Product Name']

for col in relevant_cats:
    print(f"Unique values in '{col}':")
    print(df[col].unique(), "\n")
```

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

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

Unique values in 'Country':
['United States']

Unique values in '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']

Unique values in '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']

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

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

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

Unique values in 'Product Name':
['Bush Somerset Collection Bookcase'
 'Hon Deluxe Fabric Upholstered Stacking Chairs, Rounded Back'
 'Self-Adhesive Address Labels for Typewriters by Universal' ...
 'Eureka Hand Vacuum, Bagless' 'LG G2'
 'Eldon Jumbo ProFile Portable File Boxes Graphite/Black']
```

In [109…

```python
for col in relevant_cats:
    df[col] = df[col].str.strip().str.title()
```

In [110…

```python
for col in relevant_cats:
    print(f"Cleaned values in '{col}':")
    print(df[col].unique(), "\n")
```

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

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

Cleaned values in 'Country':
['United States']

Cleaned values in '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']

Cleaned values in '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']

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

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

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

Cleaned values in 'Product Name':
['Bush Somerset Collection Bookcase'
 'Hon Deluxe Fabric Upholstered Stacking Chairs, Rounded Back'
 'Self-Adhesive Address Labels For Typewriters By Universal' ...
 'Eureka Hand Vacuum, Bagless' 'Lg G2'
 'Eldon Jumbo Profile Portable File Boxes Graphite/Black']
```

In [111…
```
df.to_csv("Superstore_Cleaned.csv", index=False)
```

In [112…
```
df.describe()
```

Out[112…

	Row ID	Order Date	Ship Date	Sales	Quan
count	9994.000000	9994	9994	9994.000000	9994.000
mean	4997.500000	2016-04-30 00:07:12.259355648	2016-05-03 23:06:58.571142912	229.858001	3.789
min	1.000000	2014-01-03 00:00:00	2014-01-07 00:00:00	0.444000	1.000
25%	2499.250000	2015-05-23 00:00:00	2015-05-27 00:00:00	17.280000	2.000
50%	4997.500000	2016-06-26 00:00:00	2016-06-29 00:00:00	54.490000	3.000
75%	7495.750000	2017-05-14 00:00:00	2017-05-18 00:00:00	209.940000	5.000
max	9994.000000	2017-12-30 00:00:00	2018-01-05 00:00:00	22638.480000	14.000
std	2885.163629	NaN	NaN	623.245101	2.225

In [113…
```
df.nunique()
```

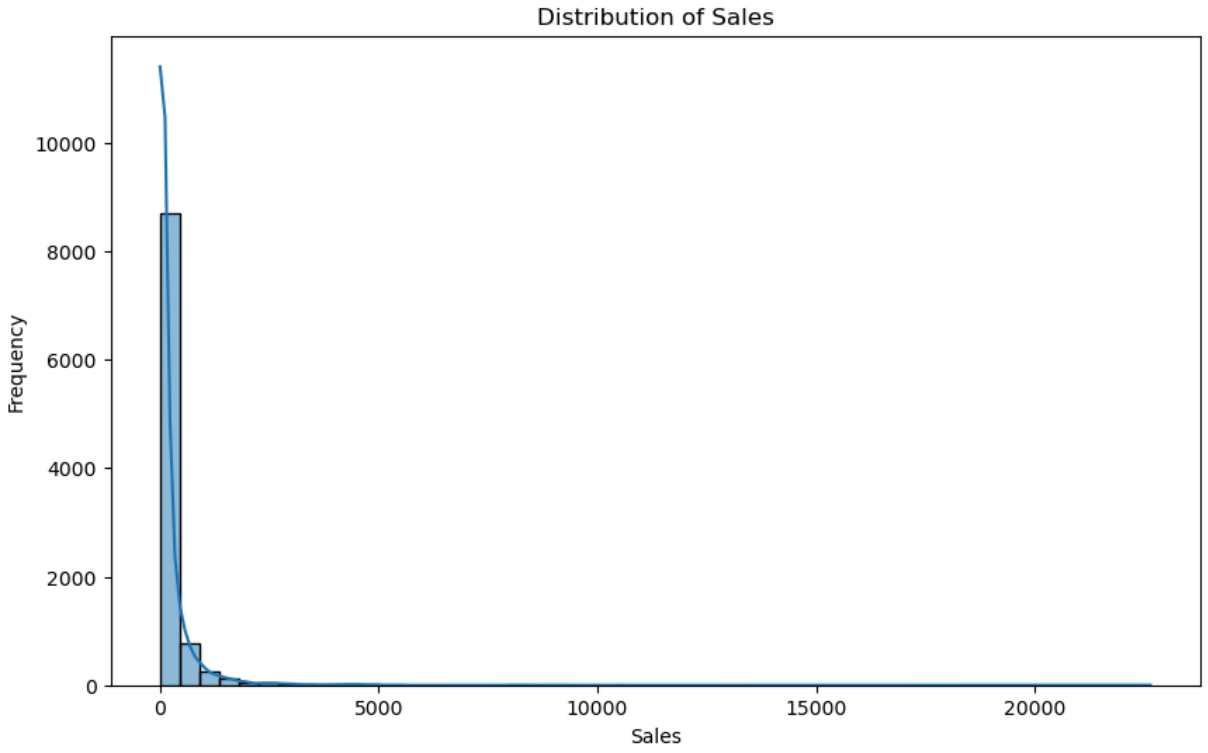
Out[113…

```
Row ID           9994
Order ID         5009
Order Date       1237
Ship Date        1334
Ship Mode           4
Customer ID       793
Customer Name     793
Segment             3
Country             1
City              531
State              49
Postal Code       631
Region              4
Product ID       1862
Category            3
Sub-Category       17
Product Name     1850
Sales            5825
Quantity           14
Discount           12
Profit           7287
dtype: int64
```

In [114…

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,6))
sns.histplot(df['Sales'], bins=50, kde=True)
plt.title('Distribution of Sales')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.show()
```



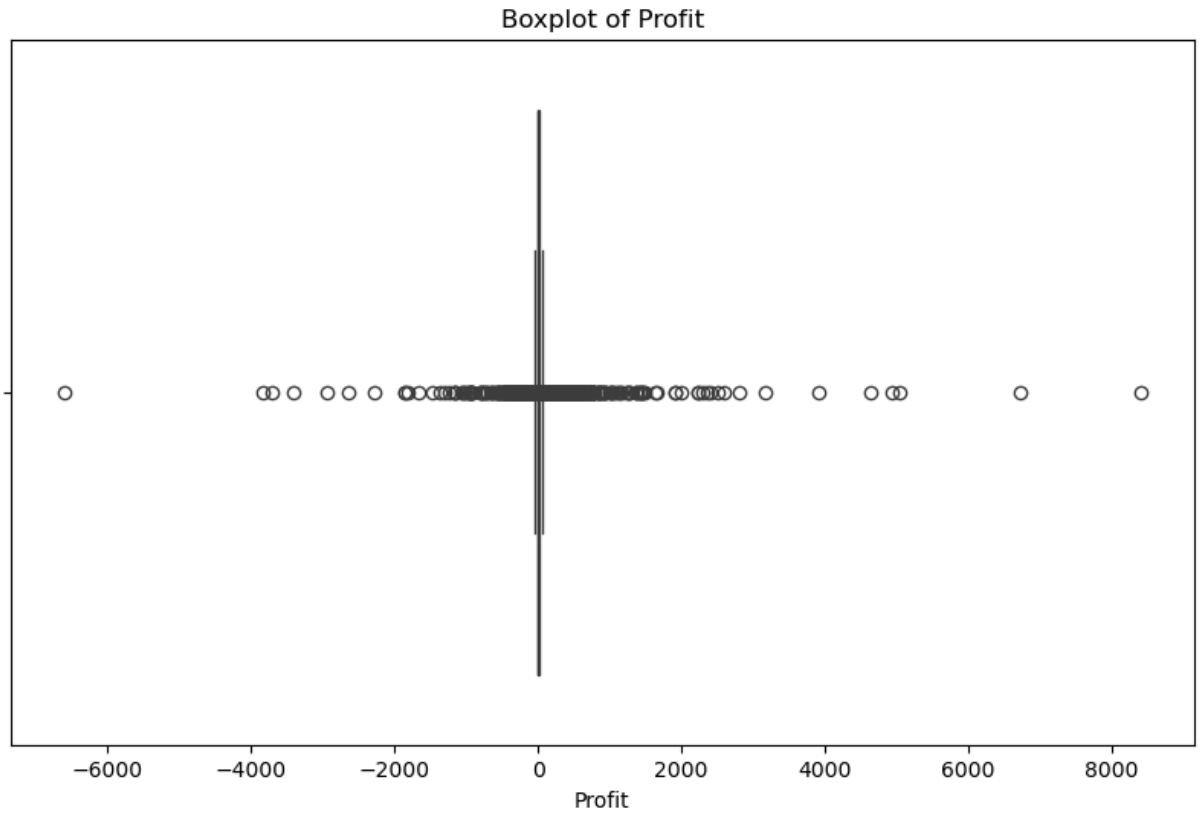
Sales Distribution & Outliers

Finding

- The majority of sales values are relatively low, but a few transactions contribute significantly high revenue.
- The sales distribution is **right-skewed**, meaning most transactions involve smaller amounts.

In [116…

```
plt.figure(figsize=(10,6))
sns.boxplot(x=df['Profit'])
plt.title('Boxplot of Profit')
plt.show()
```



Profitability Analysis

Finding

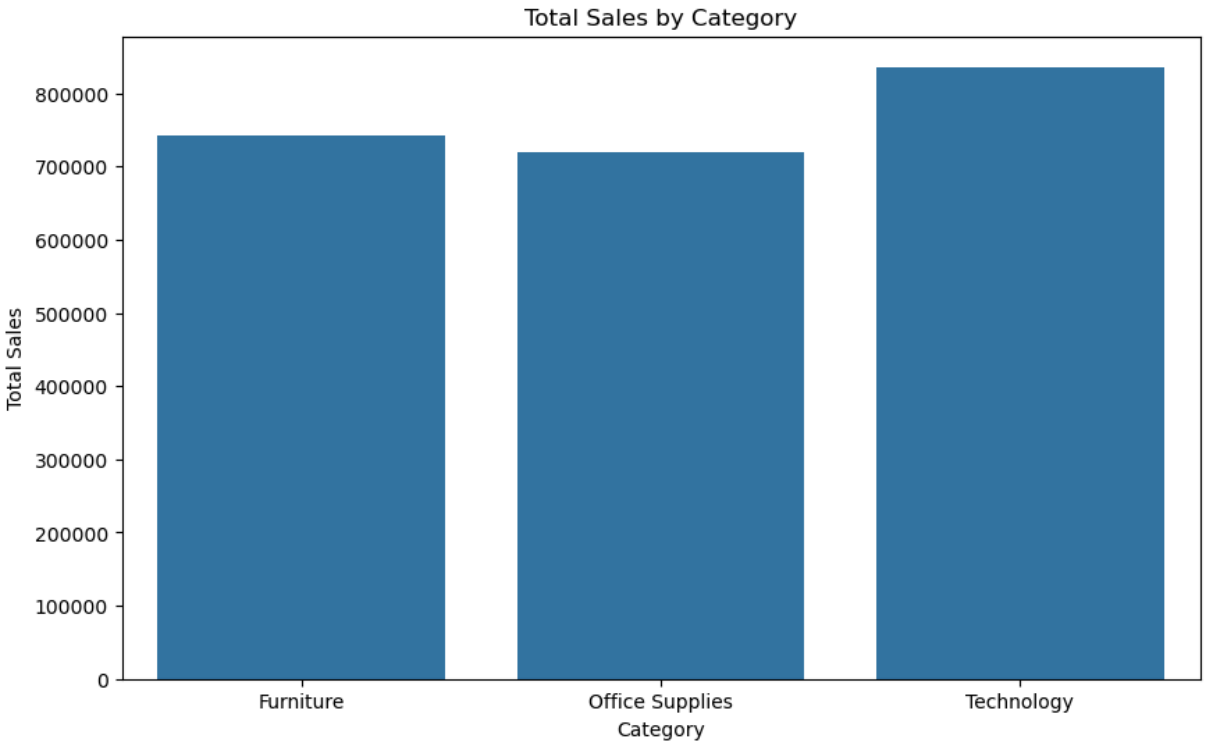
- The **boxplot** of profit reveals *__negative profit outliers__*, indicating loss-making transactions.
- Some high-sales transactions are unprofitable, meaning high revenue doesn't always mean high profit.

In [118…
```
plt.figure(figsize=(10,6))
sns.barplot(x='Category', y='Sales', data=df, estimator=sum, ci=None)
plt.title('Total Sales by Category')
plt.xlabel('Category')
plt.ylabel('Total Sales')
plt.show()
```

```
/var/folders/mf/pjg0mk757xj_7wrzg8xb_7tr0000gn/T/ipykernel_3194/307858191.p
y:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='Category', y='Sales', data=df, estimator=sum, ci=None)
```



Sales by Product Category

Finding

- The Technology category generates the highest total sales, followed by Furniture and Office Supplies.
- Office Supplies category has the lowest total sales.

In [120…
```
plt.figure(figsize=(10,6))
sns.scatterplot(x='Sales', y='Profit', data=df)
plt.title('Sales vs Profit')
plt.xlabel('Sales')
plt.ylabel('Profit')
plt.show()
```

Sales vs. Profit Relationship

Finding

- The **scatter plot** between Sales and Profit shows a *__weak correlation__*.
- Some high-revenue transactions result in low or negative profits.

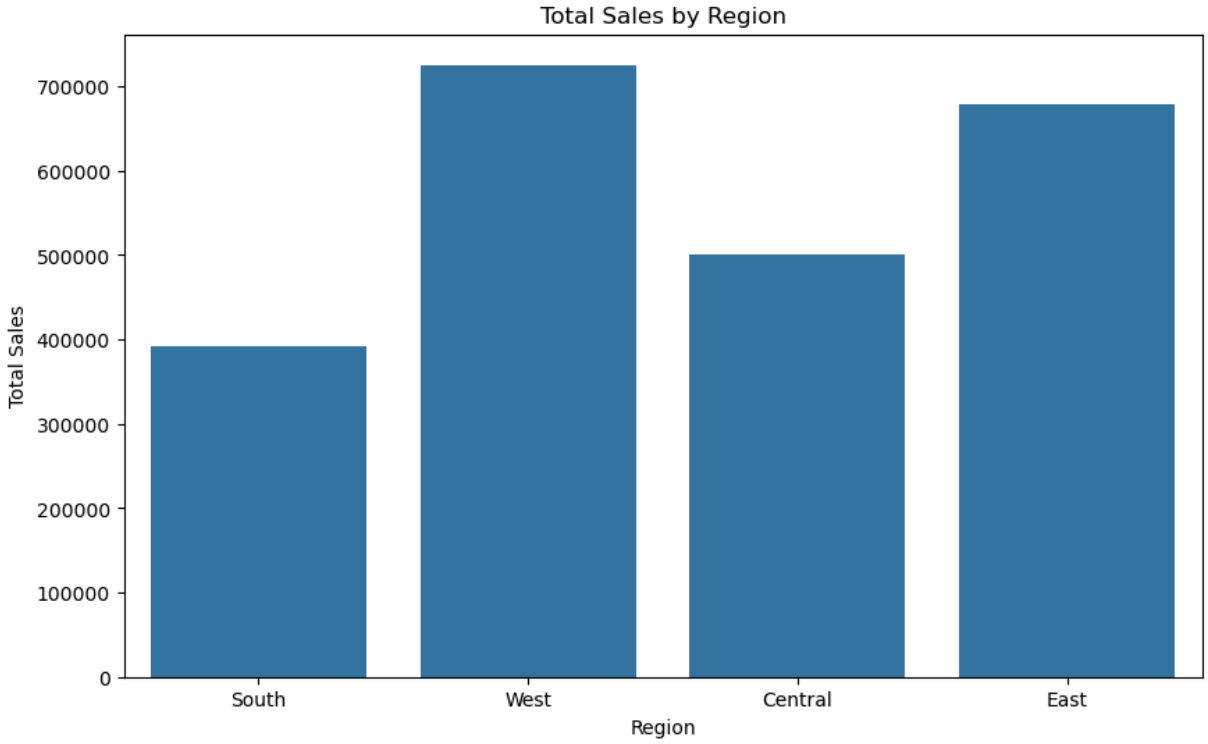
In [122…

```
plt.figure(figsize=(10,6))
sns.barplot(x='Region', y='Sales', data=df, estimator=sum, ci=None)
plt.title('Total Sales by Region')
plt.xlabel('Region')
plt.ylabel('Total Sales')
plt.show()
```

```
/var/folders/mf/pjg0mk757xj_7wrzg8xb_7tr0000gn/T/ipykernel_3194/4118717383.p
y:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='Region', y='Sales', data=df, estimator=sum, ci=None)
```



Regional Sales Performance

Finding

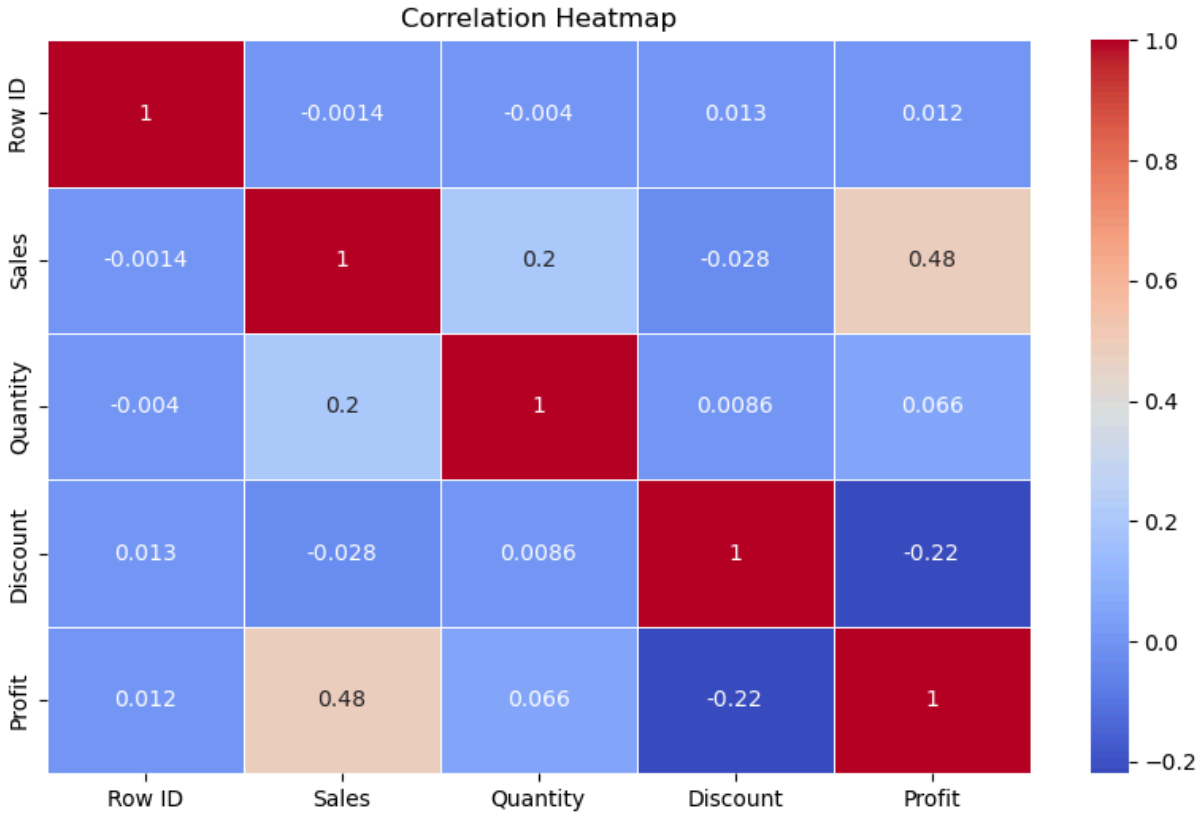
- The Western region has the highest total sales, while the Southern region has the lowest sales.

In [124…

```
import matplotlib.pyplot as plt
import seaborn as sns

# Select only numeric columns
numeric_df = df.select_dtypes(include=['number'])

# Plot correlation heatmap
plt.figure(figsize=(10,6))
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



Impact of Discount on Profit

Finding

- The **correlation heatmap** shows a negative relationship between Discount and Profit.
- Higher discounts often lead to lower profitability, even though they might drive sales.

Exploratory Data Analysis (EDA) on Superstore Sales Data

1. Introduction

Project Title

Exploratory Data Analysis (EDA) on Superstore Sales Data

Objective

The goal of this analysis is to explore sales and profitability trends, identify key patterns, and extract actionable insights to improve business performance.

Dataset Overview

- The dataset contains **9,994** sales transaction records.
- It includes **21 columns**, such as Order Date, Product Category, Sales, Profit, Discount, and Region.
- The data covers multiple product categories, regions, and customer segments.

Tools & Techniques Used

- **Python** (*Pandas, NumPy, Matplotlib, Seaborn*)
- **Jupyter Notebook** for ***data analysis*** and ***visualization***
- **Descriptive Statistics, Correlation Analysis, Data Visualizations**

2. Data Cleaning & Preparation

- No missing values were found in the dataset.
- Duplicate records were checked and removed.
- Data types were corrected:
  - Order Date and Ship Date converted to **datetime format**.
  - Postal Code converted to **string format**.
- Categorical values were ***standardized*** (removed extra spaces, fixed inconsistent capitalization).

3. Key Insights from the Analysis

3.1 Sales Distribution & Outliers

Finding

- The majority of sales values are relatively low, but a few transactions contribute significantly high revenue.
- The sales distribution is **right-skewed**, meaning most transactions involve smaller amounts.

Business Implication

- The company should focus on high-value customers and products contributing to the highest sales.
- Discount strategies or premium offerings could be optimized to increase profitability from smaller transactions.

3.2 Profitability Analysis

Finding

- The **boxplot** of profit reveals ***negative profit outliers***, indicating loss-making transactions.
- Some high-sales transactions are unprofitable, meaning high revenue doesn't always mean high profit.

Business Implication

- Loss-making products need to be analyzed—with questions like are discounts too high? Are there high operational costs?
- The company should reassess its pricing strategy and optimize discounting policies to minimize losses.

3.3 Sales by Product Category

Finding

- The Technology category generates the highest total sales, followed by Furniture and Office Supplies.
- Office Supplies category has the lowest total sales.

Business Implication

- The company should focus on boosting Office Supplies sales through promotions, better customer targeting, or product diversification.
- Technology products are a strong revenue driver—investing in more tech products or bundling them with Office Supplies could be beneficial.

3.4 Sales vs. Profit Relationship

Finding

- The **scatter plot** between Sales and Profit shows a ***weak correlation***.
- Some high-revenue transactions result in low or negative profits.

Business Implication

- A sales-driven approach isn't enough; profitability must also be considered.
- The company should analyze whether high discounts or high-cost products are affecting profit margins.

3.5 Regional Sales Performance

Finding

- The Western region has the highest total sales, while the Southern region has the lowest sales.

Business Implication

- The company should invest in marketing and expansion in the South to increase sales.
- Investigate why the West is outperforming—better distribution, higher demand, or better product availability?

3.6 Impact of Discount on Profit

Finding

- The **correlation heatmap** shows a negative relationship between Discount and Profit.
- Higher discounts often lead to lower profitability, even though they might drive sales.

Business Implication

- The company should reduce excessive discounts on certain products that don't generate enough revenue.
- Targeted discount strategies (e.g., discounts for bulk purchases) may help increase profit margins.

4. Conclusion & Recommendations

Based on our analysis, the following recommendations can help optimize sales and profitability:

1. Reduce unnecessary discounts to prevent loss-making transactions.
2. Increase focus on high-performing categories (Technology, Furniture) and improve marketing for Office Supplies.
3. Analyze unprofitable transactions and adjust pricing strategies accordingly.
4. Expand marketing efforts in the South region to improve sales.
5. Monitor high-value transactions and develop premium product offerings for top-spending customers.

In [ ]: