analysis-retail-samplesuperstore

October 20, 2023

GRIP - THE SPARK FOUNDATION

AUTHOR - SHIVAM DUBEY

DATA SCIENCE & BUSINESS ANALYTICS TASK -

Exploratory Data Analysis-Retail(SampleSuperStore) TASK - 3

Exploratory Data Analysis-Retail(SampleSuperStore)

Import libraries

```
[ ]: import numpy as np # linear algebra
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import plotly.express as px
     import plotly.graph_objects as go
     from plotly.subplots import make_subplots
```

Loading the DataSet

```
[ ]: df = pd.read_csv('SampleSuperstore.csv')
     df
```

```
[ ]:           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
     …               …         …            …                …         …
     9989    Second Class   Consumer  United States            Miami     Florida
     9990  Standard Class   Consumer  United States       Costa Mesa  California
     9991  Standard Class   Consumer  United States       Costa Mesa  California
     9992  Standard Class   Consumer  United States       Costa Mesa  California
```

```
9993    Second Class   Consumer  United States      Westminster  California

      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         2
4           33311  South  Office Supplies      Storage   22.3680         2
…           …    …          …          …       …       …
9989        33180  South        Furniture  Furnishings   25.2480         3
9990        92627   West        Furniture  Furnishings   91.9600         2
9991        92627   West       Technology       Phones  258.5760         2
9992        92627   West  Office Supplies        Paper   29.6000         4
9993        92683   West  Office Supplies   Appliances  243.1600         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
…        …       …
9989      0.20    4.1028
9990      0.00   15.6332
9991      0.20   19.3932
9992      0.00   13.3200
9993      0.00   72.9480

[9994 rows x 13 columns]
```

Reading Datasets The following code would imply these instructions

df_orders = is the name of the variable, that will be using throughout the example of this tutorial. pd = stands for Panda, it's the convention the community is using. .read_csv = is a method within to read the CSV file.

[ ]:

```
df.sample(9)
```

[ ]:

```
           Ship Mode      Segment        Country           City         State  \
499   Standard Class     Consumer  United States     Costa Mesa    California
1361  Standard Class    Corporate  United States   Philadelphia  Pennsylvania
8530  Standard Class     Consumer  United States        Detroit      Michigan
6469  Standard Class  Home Office  United States     Providence  Rhode Island
5387    Second Class    Corporate  United States    Los Angeles    California
2930        Same Day     Consumer  United States  San Francisco    California
```

```
9877     First Class  Home Office  United States       Cleveland          Ohio
3933  Standard Class     Consumer  United States     Bakersfield    California
119      First Class     Consumer  United States      Wilmington      Delaware

      Postal Code   Region        Category Sub-Category    Sales  Quantity  \
499         92627     West        Furniture  Furnishings    69.300         9
1361        19120     East  Office Supplies        Paper    23.680         4
8530        48227  Central  Office Supplies        Paper    33.360         4
6469         2908     East        Furniture  Furnishings    72.420         6
5387        90045     West       Technology       Phones   167.976         3
2930        94109     West  Office Supplies      Binders     6.608         2
9877        44105     East  Office Supplies      Binders     8.700         5
3933        93309     West  Office Supplies          Art     9.400         5
119         19805     East        Furniture  Furnishings    47.040         3

      Discount   Profit
499        0.0  22.8690
1361       0.2   7.4000
8530       0.0  16.6800
6469       0.0  23.8986
5387       0.2  10.4985
2930       0.2   2.2302
9877       0.7  -6.3800
3933       0.0   2.7260
119        0.0  18.3456
```

[ ]:
```
df.tail()
```

[ ]:
```
           Ship Mode   Segment        Country         City       State  \
9989    Second Class  Consumer  United States        Miami      Florida
9990  Standard Class  Consumer  United States   Costa Mesa   California
9991  Standard Class  Consumer  United States   Costa Mesa   California
9992  Standard Class  Consumer  United States   Costa Mesa   California
9993    Second Class  Consumer  United States  Westminster   California

      Postal Code Region         Category Sub-Category    Sales  Quantity  \
9989        33180  South        Furniture  Furnishings   25.248         3
9990        92627   West        Furniture  Furnishings   91.960         2
9991        92627   West       Technology       Phones  258.576         2
9992        92627   West  Office Supplies        Paper   29.600         4
9993        92683   West  Office Supplies   Appliances  243.160         2

      Discount   Profit
9989       0.2   4.1028
9990       0.0  15.6332
9991       0.2  19.3932
9992       0.0  13.3200
```

```
9993       0.0  72.9480
```

check the missing value

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

```
[ ]: 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
```

Finding Total number of null values in a dataset

```
[ ]: print("total number of null values = ",df.isnull().sum().sum())
```

```
total number of null values =  0
```

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

```
<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
```

Statistical details of the dataset

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

```
[ ]:         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
```

shape of the DataSet

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

```
[ ]: (9994, 13)
```

Find the dtypes in the Dataset

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

```
[ ]: Ship Mode        object
     Segment          object
     Country          object
     City             object
     State            object
     Postal Code       int64
     Region           object
     Category         object
     Sub-Category     object
     Sales           float64
     Quantity          int64
     Discount        float64
     Profit          float64
     dtype: object
```

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

```
[ ]: Index(['Ship Mode', 'Segment', 'Country', 'City', 'State', 'Postal Code',
            'Region', 'Category', 'Sub-Category', 'Sales', 'Quantity', 'Discount',
            'Profit'],
           dtype='object')
```

Statistic Figures The following code would imply these instructions

- df_orders = is the name of the variable, that will be using throughout the example of this tutorial.
- .count = is the count value to a specific column.
- .mean = is the mean value to a specific column.
- .std = is the std value to a specific column.
- .min = is the min value to a specific column.

```
[ ]: df["Sales"].count()
```

```
[ ]: 9994
```

```
[ ]: df["Sales"].mean()
```

```
[ ]: 229.85800083049833
```

```
[ ]: df["Sales"].std()
```

```
[ ]: 623.2451005086807
```

```
[ ]: df["Sales"].min()
```

```
[ ]: 0.444
```

Exporting Dataset Once that we've satisfied with out results, let's export them a new CSV dataset, so we could work with them on the next notebook.

- df_orders = is the name of the variable, that will be using throughout the example of this tutorial.
- .to_csv = is the export method to a CSV dataset.
- index = False = we need to definet this index value set to False, since we don't want the index column.

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

```
<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
```

Bonus Stage Now that we've come a long way of exploring our superstore.csv dataset, it's time to dive a little bit deeper of what, both Python and Pandas capable of delivering. Let's try to create a custom class in Python by leveraging our builtin Pandas method available in the library.

[ ]:

```python
# Let's create a class named `display_all`, by which later we call on the next command.

def display_all(df_orders):
    with pd.option_context("display.max_rows", 20, "display.max_columns", 20):
        display(df_orders)
```

[ ]:

```python
display_all(df.head(10).T)
```

```
                           0              1               2                3  \
Ship Mode       Second Class   Second Class    Second Class   Standard Class
Segment             Consumer       Consumer       Corporate         Consumer
Country        United States  United States   United States    United States
City               Henderson      Henderson     Los Angeles  Fort Lauderdale
State               Kentucky       Kentucky      California          Florida
Postal Code            42420          42420           90036            33311
Region                 South          South            West            South
Category           Furniture      Furniture  Office Supplies        Furniture
Sub-Category       Bookcases         Chairs          Labels           Tables
Sales                 261.96         731.94           14.62         957.5775
Quantity                   2              3               2                5
Discount                 0.0            0.0             0.0             0.45
Profit               41.9136        219.582          6.8714         -383.031

                             4               5                6  \
Ship Mode       Standard Class  Standard Class   Standard Class
Segment               Consumer        Consumer         Consumer
Country          United States   United States    United States
City           Fort Lauderdale     Los Angeles      Los Angeles
State                  Florida      California       California
Postal Code              33311           90032            90032
Region                   South            West             West
Category       Office Supplies       Furniture  Office Supplies
Sub-Category           Storage     Furnishings              Art
Sales                   22.368           48.86             7.28
Quantity                     2               7                4
```

```
Discount                  0.2             0.0              0.0
Profit                 2.5164         14.1694           1.9656

                           7                8                9
Ship Mode     Standard Class   Standard Class   Standard Class
Segment             Consumer         Consumer         Consumer
Country        United States    United States    United States
City             Los Angeles      Los Angeles      Los Angeles
State             California       California       California
Postal Code            90032            90032            90032
Region                  West             West             West
Category          Technology  Office Supplies  Office Supplies
Sub-Category          Phones          Binders       Appliances
Sales                907.152           18.504            114.9
Quantity                   6                3                5
Discount                 0.2              0.2              0.0
Profit               90.7152           5.7825            34.47
```

```
[ ]: display_all(df.describe(include='all').T)
```

```
               count unique              top  freq          mean          std  \
Ship Mode       9994      4   Standard Class  5968           NaN          NaN
Segment         9994      3         Consumer  5191           NaN          NaN
Country         9994      1    United States  9994           NaN          NaN
City            9994    531    New York City   915           NaN          NaN
State           9994     49       California  2001           NaN          NaN
Postal Code   9994.0    NaN              NaN   NaN  55190.379428  32063.69335
Region          9994      4             West  3203           NaN          NaN
Category        9994      3  Office Supplies  6026           NaN          NaN
Sub-Category    9994     17          Binders  1523           NaN          NaN
Sales         9994.0    NaN              NaN   NaN    229.858001   623.245101
Quantity      9994.0    NaN              NaN   NaN      3.789574      2.22511
Discount      9994.0    NaN              NaN   NaN      0.156203     0.206452
Profit        9994.0    NaN              NaN   NaN     28.656896   234.260108

                   min      25%      50%      75%       max
Ship Mode          NaN      NaN      NaN      NaN       NaN
Segment            NaN      NaN      NaN      NaN       NaN
Country            NaN      NaN      NaN      NaN       NaN
City               NaN      NaN      NaN      NaN       NaN
State              NaN      NaN      NaN      NaN       NaN
Postal Code     1040.0  23223.0  56430.5  90008.0   99301.0
Region             NaN      NaN      NaN      NaN       NaN
Category           NaN      NaN      NaN      NaN       NaN
Sub-Category       NaN      NaN      NaN      NaN       NaN
Sales            0.444    17.28    54.49   209.94  22638.48
Quantity           1.0      2.0      3.0      5.0      14.0
Discount           0.0      0.0      0.2      0.2       0.8
```

```
Profit       -6599.978  1.72875   8.6665   29.364  8399.976
```

Data Cleaning

```
[ ]: # Set the total rows and columns default numbers.
     pd.set_option('display.max_columns', 21)
     pd.set_option('display.max_rows', 5)
```

```
[ ]: Cat = [i for i in df.columns if df.dtypes[i] == 'object']
     for j in Cat:
         print('\033[95m' + j + '\033[0m')
         print(sorted(df[j].unique()))
```

```
Ship Mode
['First Class', 'Same Day', 'Second Class', 'Standard Class']
Segment
['Consumer', 'Corporate', 'Home Office']
Country
['United States']
City
['Aberdeen', 'Abilene', 'Akron', 'Albuquerque', 'Alexandria', 'Allen',
'Allentown', 'Altoona', 'Amarillo', 'Anaheim', 'Andover', 'Ann Arbor',
'Antioch', 'Apopka', 'Apple Valley', 'Appleton', 'Arlington', 'Arlington
Heights', 'Arvada', 'Asheville', 'Athens', 'Atlanta', 'Atlantic City', 'Auburn',
'Aurora', 'Austin', 'Avondale', 'Bakersfield', 'Baltimore', 'Bangor',
'Bartlett', 'Bayonne', 'Baytown', 'Beaumont', 'Bedford', 'Belleville',
'Bellevue', 'Bellingham', 'Bethlehem', 'Beverly', 'Billings', 'Bloomington',
'Boca Raton', 'Boise', 'Bolingbrook', 'Bossier City', 'Bowling Green', 'Boynton
Beach', 'Bozeman', 'Brentwood', 'Bridgeton', 'Bristol', 'Broken Arrow',
'Broomfield', 'Brownsville', 'Bryan', 'Buffalo', 'Buffalo Grove', 'Bullhead
City', 'Burbank', 'Burlington', 'Caldwell', 'Camarillo', 'Cambridge', 'Canton',
'Carlsbad', 'Carol Stream', 'Carrollton', 'Cary', 'Cedar Hill', 'Cedar Rapids',
'Champaign', 'Chandler', 'Chapel Hill', 'Charlotte', 'Charlottesville',
'Chattanooga', 'Chesapeake', 'Chester', 'Cheyenne', 'Chicago', 'Chico', 'Chula
Vista', 'Cincinnati', 'Citrus Heights', 'Clarksville', 'Cleveland', 'Clifton',
'Clinton', 'Clovis', 'Coachella', 'College Station', 'Colorado Springs',
'Columbia', 'Columbus', 'Commerce City', 'Concord', 'Conroe', 'Conway', 'Coon
Rapids', 'Coppell', 'Coral Gables', 'Coral Springs', 'Corpus Christi', 'Costa
Mesa', 'Cottage Grove', 'Covington', 'Cranston', 'Cuyahoga Falls', 'Dallas',
'Danbury', 'Danville', 'Davis', 'Daytona Beach', 'Dearborn', 'Dearborn Heights',
'Decatur', 'Deer Park', 'Delray Beach', 'Deltona', 'Denver', 'Des Moines', 'Des
Plaines', 'Detroit', 'Dover', 'Draper', 'Dublin', 'Dubuque', 'Durham', 'Eagan',
'East Orange', 'East Point', 'Eau Claire', 'Edinburg', 'Edmond', 'Edmonds', 'El
Cajon', 'El Paso', 'Elkhart', 'Elmhurst', 'Elyria', 'Encinitas', 'Englewood',
'Escondido', 'Eugene', 'Evanston', 'Everett', 'Fairfield', 'Fargo',
'Farmington', 'Fayetteville', 'Florence', 'Fort Collins', 'Fort Lauderdale',
'Fort Worth', 'Frankfort', 'Franklin', 'Freeport', 'Fremont', 'Fresno',
'Frisco', 'Gaithersburg', 'Garden City', 'Garland', 'Gastonia', 'Georgetown',
'Gilbert', 'Gladstone', 'Glendale', 'Glenview', 'Goldsboro', 'Grand Island',
'Grand Prairie', 'Grand Rapids', 'Grapevine', 'Great Falls', 'Greeley', 'Green
Bay', 'Greensboro', 'Greenville', 'Greenwood', 'Gresham', 'Grove City',
'Gulfport', 'Hackensack', 'Hagerstown', 'Haltom City', 'Hamilton', 'Hampton',
'Harlingen', 'Harrisonburg', 'Hattiesburg', 'Helena', 'Hempstead', 'Henderson',
'Hendersonville', 'Hesperia', 'Hialeah', 'Hickory', 'Highland Park',
'Hillsboro', 'Holland', 'Hollywood', 'Holyoke', 'Homestead', 'Hoover', 'Hot
Springs', 'Houston', 'Huntington Beach', 'Huntsville', 'Independence',
'Indianapolis', 'Inglewood', 'Iowa City', 'Irving', 'Jackson', 'Jacksonville',
'Jamestown', 'Jefferson City', 'Johnson City', 'Jonesboro', 'Jupiter', 'Keller',
'Kenner', 'Kenosha', 'Kent', 'Kirkwood', 'Kissimmee', 'Knoxville', 'La Crosse',
'La Mesa', 'La Porte', 'La Quinta', 'Lafayette', 'Laguna Niguel', 'Lake
Charles', 'Lake Elsinore', 'Lake Forest', 'Lakeland', 'Lakeville', 'Lakewood',
'Lancaster', 'Lansing', 'Laredo', 'Las Cruces', 'Las Vegas', 'Laurel',
'Lawrence', 'Lawton', 'Layton', 'League City', 'Lebanon', 'Lehi', 'Leominster',
'Lewiston', 'Lincoln Park', 'Linden', 'Lindenhurst', 'Little Rock', 'Littleton',
'Lodi', 'Logan', 'Long Beach', 'Longmont', 'Longview', 'Lorain', 'Los Angeles',
'Louisville', 'Loveland', 'Lowell', 'Lubbock', 'Macon', 'Madison', 'Malden',
'Manchester', 'Manhattan', 'Mansfield', 'Manteca', 'Maple Grove', 'Margate',
'Marietta', 'Marion', 'Marlborough', 'Marysville', 'Mason', 'Mcallen',
'Medford', 'Medina', 'Melbourne', 'Memphis', 'Mentor', 'Meriden', 'Meridian',
'Mesa', 'Mesquite', 'Miami', 'Middletown', 'Midland', 'Milford', 'Milwaukee',
'Minneapolis', 'Miramar', 'Mishawaka', 'Mission Viejo', 'Missoula', 'Missouri
City', 'Mobile', 'Modesto', 'Monroe', 'Montebello', 'Montgomery', 'Moorhead',
'Moreno Valley', 'Morgan Hill', 'Morristown', 'Mount Pleasant', 'Mount Vernon',
'Murfreesboro', 'Murray', 'Murrieta', 'Muskogee', 'Naperville', 'Nashua',
'Nashville', 'New Albany', 'New Bedford', 'New Brunswick', 'New Castle', 'New
Rochelle', 'New York City', 'Newark', 'Newport News', 'Niagara Falls',
'Noblesville', 'Norfolk', 'Normal', 'Norman', 'North Charleston', 'North Las
Vegas', 'North Miami', 'Norwich', 'Oak Park', 'Oakland', 'Oceanside', 'Odessa',
'Oklahoma City', 'Olathe', 'Olympia', 'Omaha', 'Ontario', 'Orange', 'Orem',
'Orland Park', 'Orlando', 'Ormond Beach', 'Oswego', 'Overland Park',
'Owensboro', 'Oxnard', 'Palatine', 'Palm Coast', 'Park Ridge', 'Parker',
'Parma', 'Pasadena', 'Pasco', 'Passaic', 'Paterson', 'Pearland', 'Pembroke
Pines', 'Pensacola', 'Peoria', 'Perth Amboy', 'Pharr', 'Philadelphia',
'Phoenix', 'Pico Rivera', 'Pine Bluff', 'Plainfield', 'Plano', 'Plantation',
'Pleasant Grove', 'Pocatello', 'Pomona', 'Pompano Beach', 'Port Arthur', 'Port
Orange', 'Port Saint Lucie', 'Portage', 'Portland', 'Providence', 'Provo',
'Pueblo', 'Quincy', 'Raleigh', 'Rancho Cucamonga', 'Rapid City', 'Reading',
'Redding', 'Redlands', 'Redmond', 'Redondo Beach', 'Redwood City', 'Reno',
'Renton', 'Revere', 'Richardson', 'Richmond', 'Rio Rancho', 'Riverside',
'Rochester', 'Rochester Hills', 'Rock Hill', 'Rockford', 'Rockville', 'Rogers',
'Rome', 'Romeoville', 'Roseville', 'Roswell', 'Round Rock', 'Royal Oak',
'Sacramento', 'Saginaw', 'Saint Charles', 'Saint Cloud', 'Saint Louis', 'Saint
Paul', 'Saint Peters', 'Saint Petersburg', 'Salem', 'Salinas', 'Salt Lake City',
'San Angelo', 'San Antonio', 'San Bernardino', 'San Clemente', 'San Diego', 'San
Francisco', 'San Gabriel', 'San Jose', 'San Luis Obispo', 'San Marcos', 'San
Mateo', 'Sandy Springs', 'Sanford', 'Santa Ana', 'Santa Barbara', 'Santa Clara',
'Santa Fe', 'Santa Maria', 'Scottsdale', 'Seattle', 'Sheboygan', 'Shelton',
'Sierra Vista', 'Sioux Falls', 'Skokie', 'Smyrna', 'South Bend', 'Southaven',
'Sparks', 'Spokane', 'Springdale', 'Springfield', 'Sterling Heights',
'Stockton', 'Suffolk', 'Summerville', 'Sunnyvale', 'Superior', 'Tallahassee',
'Tamarac', 'Tampa', 'Taylor', 'Temecula', 'Tempe', 'Texarkana', 'Texas City',
'The Colony', 'Thomasville', 'Thornton', 'Thousand Oaks', 'Tigard', 'Tinley
Park', 'Toledo', 'Torrance', 'Trenton', 'Troy', 'Tucson', 'Tulsa', 'Tuscaloosa',
'Twin Falls', 'Tyler', 'Urbandale', 'Utica', 'Vacaville', 'Vallejo',
'Vancouver', 'Vineland', 'Virginia Beach', 'Visalia', 'Waco', 'Warner Robins',
'Warwick', 'Washington', 'Waterbury', 'Waterloo', 'Watertown', 'Waukesha',
'Wausau', 'Waynesboro', 'West Allis', 'West Jordan', 'West Palm Beach',
'Westfield', 'Westland', 'Westminster', 'Wheeling', 'Whittier', 'Wichita',
'Wilmington', 'Wilson', 'Woodbury', 'Woodland', 'Woodstock', 'Woonsocket',
'Yonkers', 'York', 'Yucaipa', 'Yuma']
State
['Alabama', 'Arizona', 'Arkansas', 'California', 'Colorado', 'Connecticut',
'Delaware', 'District of Columbia', 'Florida', 'Georgia', 'Idaho', 'Illinois',
'Indiana', 'Iowa', 'Kansas', 'Kentucky', 'Louisiana', 'Maine', 'Maryland',
'Massachusetts', 'Michigan', 'Minnesota', 'Mississippi', 'Missouri', 'Montana',
'Nebraska', 'Nevada', 'New Hampshire', 'New Jersey', 'New Mexico', 'New York',
'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon', 'Pennsylvania',
'Rhode Island', 'South Carolina', 'South Dakota', 'Tennessee', 'Texas', 'Utah',
'Vermont', 'Virginia', 'Washington', 'West Virginia', 'Wisconsin', 'Wyoming']
Region
['Central', 'East', 'South', 'West']
Category
['Furniture', 'Office Supplies', 'Technology']
Sub-Category
['Accessories', 'Appliances', 'Art', 'Binders', 'Bookcases', 'Chairs',
'Copiers', 'Envelopes', 'Fasteners', 'Furnishings', 'Labels', 'Machines',
'Paper', 'Phones', 'Storage', 'Supplies', 'Tables']
```

```
[ ]: df.dtypes.to_frame()
```

```
[ ]:                  0
      Ship Mode   object
      Segment     object
      …             …
      Discount   float64
      Profit     float64

      [13 rows x 1 columns]
```

```
[ ]: palette_color = sns.color_palette("flare")

     sns.palplot(palette_color, size = 3)
```

```
plt.text(-0.75,-0.75, "Superstore's Retails: Visualizations &␣
↪Classifications",{'font':'serif', 'size':24, 'weight':'bold'})
plt.text(-0.75,-0.65, 'Lets try to stick to these colors throughout␣
↪presentation.',{'font':'serif', 'size':16},alpha = 0.9)
plt.gcf().set_facecolor('#f5f6f6')
plt.gcf().set_dpi(100)
plt.box(None)
plt.axis('off')
plt.show()
```

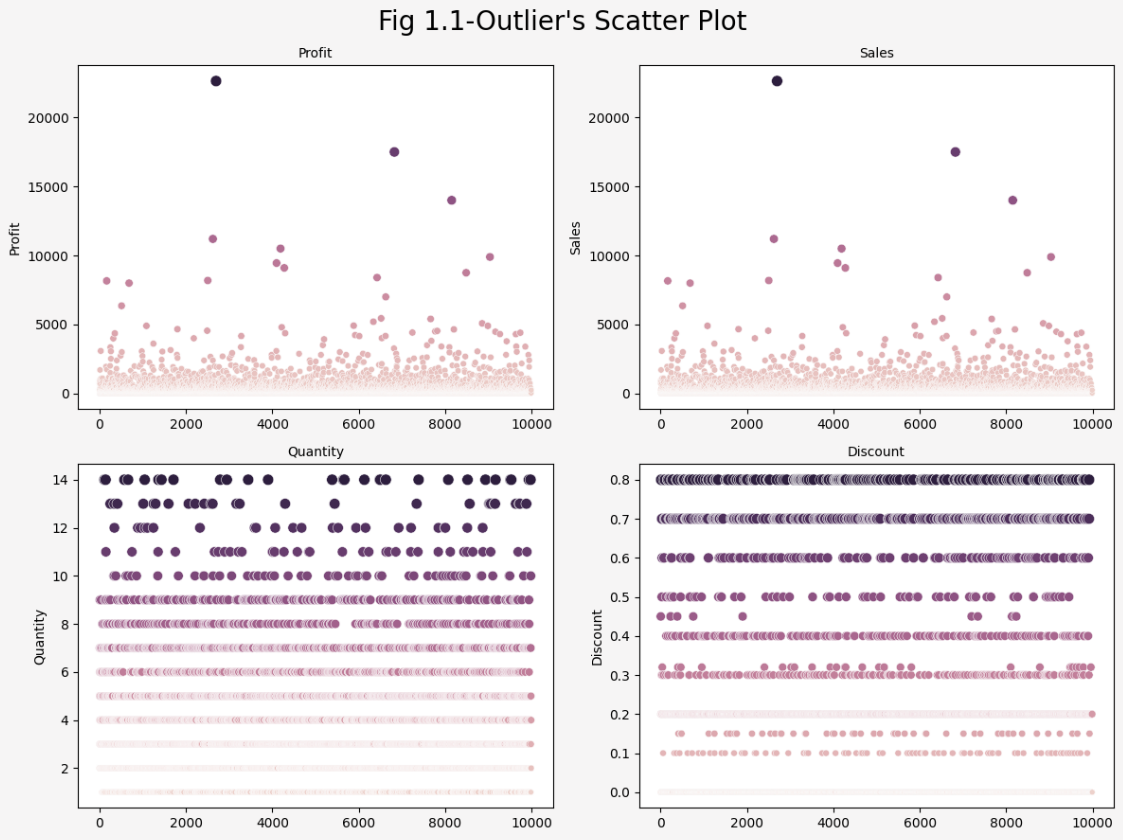
Superstore's Retails: Visualizations & Classifications
Lets try to stick to these colors throughout presentation.

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

[ ]:
```
        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
…               …             …            …            …            …
75%    90008.000000    209.940000     5.000000     0.200000    29.364000
max    99301.000000  22638.480000    14.000000     0.800000  8399.976000

[8 rows x 5 columns]
```

[ ]:
```
fig, axs = plt.subplots(nrows = 2, ncols = 2, figsize=(12, 9));
fig.patch.set_facecolor('#f6f5f5')

# Create a new column named `Profit`
df['Profit'] = df['Sales'] - df['Discount']

sns.scatterplot(data=df, y="Profit", x = df.index, ax = axs[0][0],hue =␣
↪"Profit", size = "Profit", legend=False)
axs[0][0].set_title('Profit', fontsize = 10)

sns.scatterplot(data=df, y ="Sales", x = df.index, ax = axs[0][1], hue␣
↪="Sales", size = "Sales" , legend=False)
axs[0][1].set_title('Sales', fontsize = 10)

sns.scatterplot(data=df, y = "Quantity", x = df.index, ax = axs[1][0], hue =␣
↪"Quantity", size = "Quantity", legend=False)
```

```
axs[1][0].set_title('Quantity', fontsize = 10)

sns.scatterplot(data= df, y = "Discount", x = df.index, ax = axs[1][1], hue =␣
↪"Discount", size = "Discount", legend=False)
axs[1][1].set_title('Discount', fontsize = 10)
plt.suptitle("Fig 1.1-Outlier's Scatter Plot",fontsize = 20)

plt.tight_layout()
plt.show()
```



Find the covariance of dataset

```
[ ]: df.cov()
```

```
<ipython-input-38-6f98a29763d5>:1: FutureWarning: The default value of
numeric_only in DataFrame.cov is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  df.cov()
```

```
[ ]:              Postal Code          Sales    Quantity    Discount  \
     Postal Code  1.028080e+09 -476682.766590  910.415885  386.870404
     Sales       -4.766828e+05  388434.455308  278.459923   -3.627228
     Quantity     9.104159e+02     278.459923    4.951113    0.003961
     Discount     3.868704e+02      -3.627228    0.003961    0.042622
     Profit      -4.770696e+05  388438.082536  278.455961   -3.669851

                         Profit
     Postal Code -477069.636994
     Sales        388438.082536
     Quantity        278.455961
     Discount         -3.669851
     Profit       388441.752387
```

Find the Series containing counts of unique values

```
[ ]: df.value_counts()
```

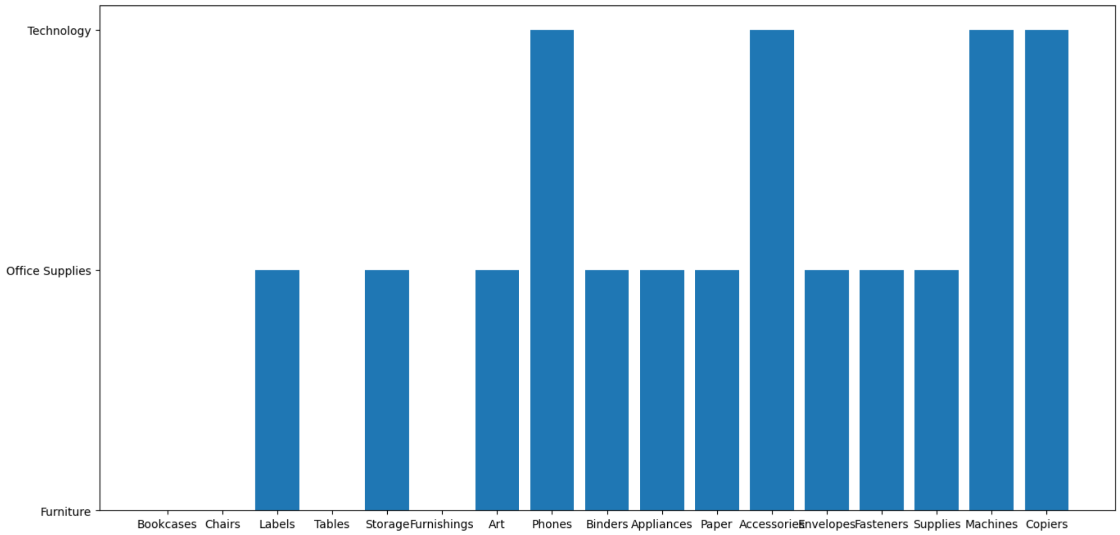
```
[ ]: Ship Mode       Segment      Country        City         State       Postal Code
     Region   Category         Sub-Category  Sales    Quantity  Discount  Profit
     Second Class    Consumer     United States  Seattle      Washington  98115
     West     Office Supplies  Paper         12.960   2         0.0       12.960
     2
                     Corporate    United States  Chicago      Illinois    60653
     Central  Office Supplies  Binders       3.564    3         0.8       2.764
     2
      ..
                                                 Little Rock  Arkansas    72209
     South    Office Supplies  Storage       62.040   4         0.0       62.040
     1
     Standard Class  Home Office  United States  Yuma         Arizona     85364
     West     Technology       Machines      599.985  5         0.7       599.285
     1
     Length: 9971, dtype: int64
```

Deleting the Variable

```
[ ]: col=['Postal Code']
     df1=df.drop(columns=col,axis=1)
```

Proper Visualization of the data set

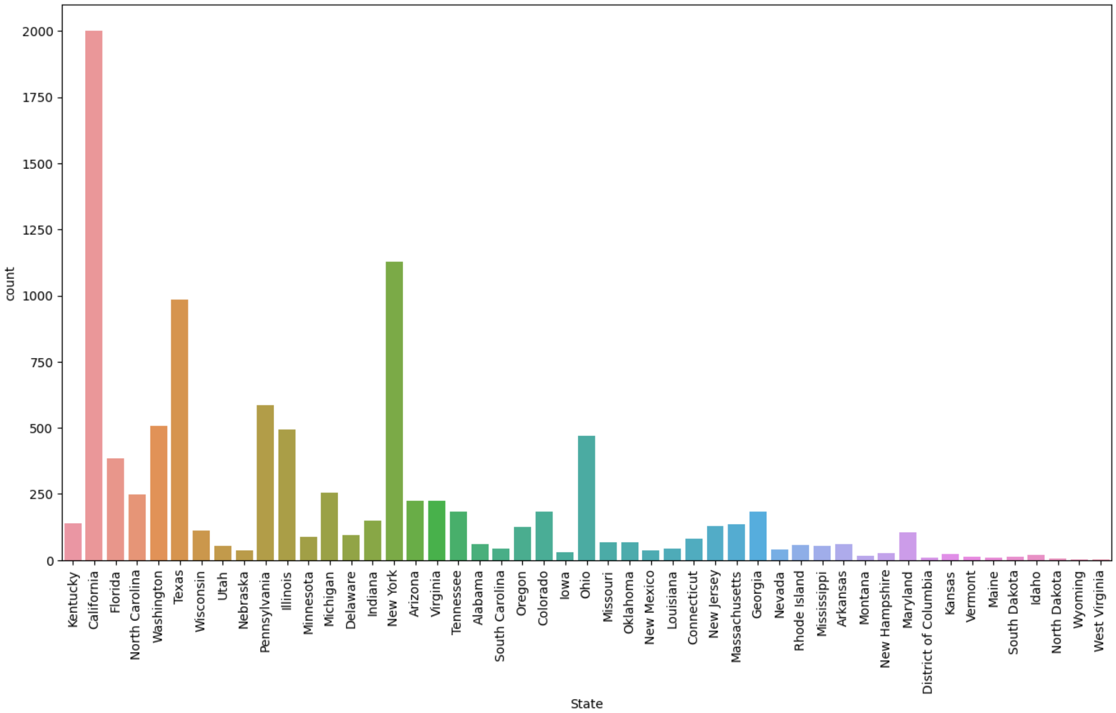
```
[ ]: plt.figure(figsize=(16,8))
     plt.bar('Sub-Category','Category', data=df)
     plt.show()
```



[ ]:

```
print(df1['State'].value_counts())
plt.figure(figsize=(15,8))
sns.countplot(x=df1['State'])
plt.xticks(rotation=90)
plt.show()
```

```
California       2001
New York         1128
                 …
West Virginia       4
Wyoming             1
Name: State, Length: 49, dtype: int64
```



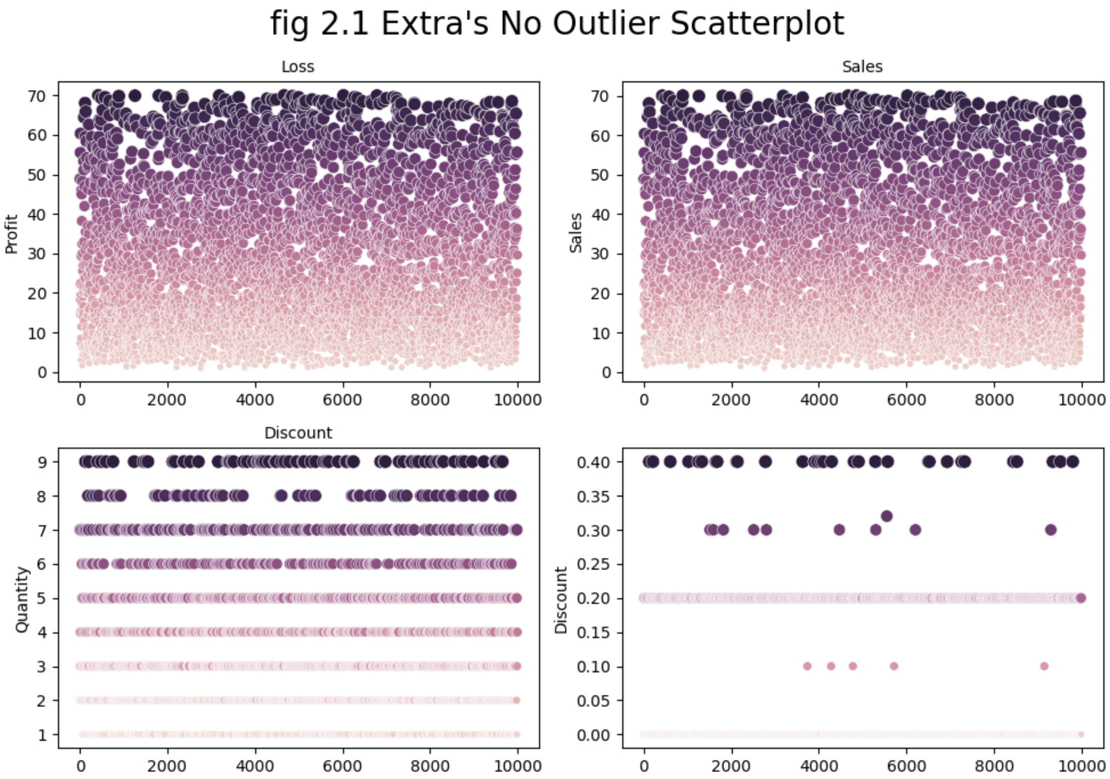
[ ]:

```
data_outliers = df[df["Profit"]<70]; data_outliers =␣
  ↪data_outliers[data_outliers["Profit"]>-40]
data_outliers = data_outliers[df["Sales"]<498.93] ; data_outliers =␣
  ↪data_outliers[data_outliers["Quantity"]<9.5]
data_outliers = data_outliers[data_outliers["Discount"]<0.5]


print(f"The total number of records containing outliers = {df.
  ↪shape[0]-data_outliers.shape[0]}\nThe outliers forms {round((df.
  ↪shape[0]-data_outliers.shape[0])/df.shape[0]*100,2)}% of superstore's␣
  ↪dataset")

fig, axs = plt.subplots(nrows = 2, ncols = 2, figsize=(10, 7));

sns.scatterplot(data=data_outliers, y = "Profit", x = data_outliers.index, ax =␣
  ↪axs[0][0], hue="Profit", size="Profit", legend=False)
sns.scatterplot(data=data_outliers, y = "Sales", x = data_outliers.index, ax =␣
  ↪axs[0][1], hue ="Sales", size ="Sales", legend=False)
sns.scatterplot(data=data_outliers,y = "Quantity",x = data_outliers.index, ax =␣
  ↪axs[1][0], hue = "Quantity", size ="Quantity", legend=False)
sns.scatterplot(data=data_outliers,y = "Discount",x = data_outliers.index, ax =␣
  ↪axs[1][1], hue = "Discount", size = "Discount", legend=False)
axs[0][0].set_title('Loss', fontsize = 10)
```

```
axs[0][1].set_title('Sales', fontsize = 10)
axs[1][0].set_title('Quantity', fontsize = 10)
axs[1][0].set_title('Discount', fontsize = 10)
plt.suptitle("fig 2.1 Extra's No Outlier Scatterplot",fontsize=20)
plt.tight_layout()
```

```
<ipython-input-49-ed00e5a4e069>:2: UserWarning: Boolean Series key will be
reindexed to match DataFrame index.
  data_outliers = data_outliers[df["Sales"]<498.93] ; data_outliers =
data_outliers[data_outliers["Quantity"]<9.5]
```

```
The total number of records containing outliers = 5245
The outliers forms 52.48% of superstore's dataset
```



Data Reduction

Data Reduction Dropping the variable ‘Country’ via attribute dimensionality reduction, because it contains 100% identical values of “United States” for all records. ‘Postal Code’ as we wont be using it in our data analysis as we also have the variabl e of “citys”

[ ]:

```
clean_data=df.drop(['Country','Postal Code'], axis=1)
```

Treating duplicate values

Duplicated rows or records can now by dropped from the dataset, as this redundancy may cause inaccurate results and outcomes (an assumption on the dataset).

```
[ ]: print(f"Number of duplicate recocrds present in the dataset = {clean_data.
       ↪duplicated().sum()}")
```

```
Number of duplicate recocrds present in the dataset = 63
```

How does Sales, Quantity and Discount affects SuperStores Profits?

We can check this out by ploting a heatmap showcasing the corealation between each numrical columns across we won't be including the feature we added in the dataframe for this correlation.

```
[ ]: fig = px.scatter(clean_data,x="Profit",y="Sales",color="Discount",
                      size="Quantity",symbol="Segment",title="How diffrent factors␣
       ↪affects Superstore's sales ")
     fig.update_layout(height=600, width=800,
                       legend=dict(yanchor="top", y=0.99,
                                   xanchor="left", x=0.01))
     fig.show()
```

Profit/ Loss, Sales and Discounts diversity by States! Now We're going to dive into the state wise analysis:

We're going to see the sales in every state We're going to see the Loss ccured in every state We're also going to if the state which causes the most is also causing more loss in average? We're going to give the discounts given off in every state

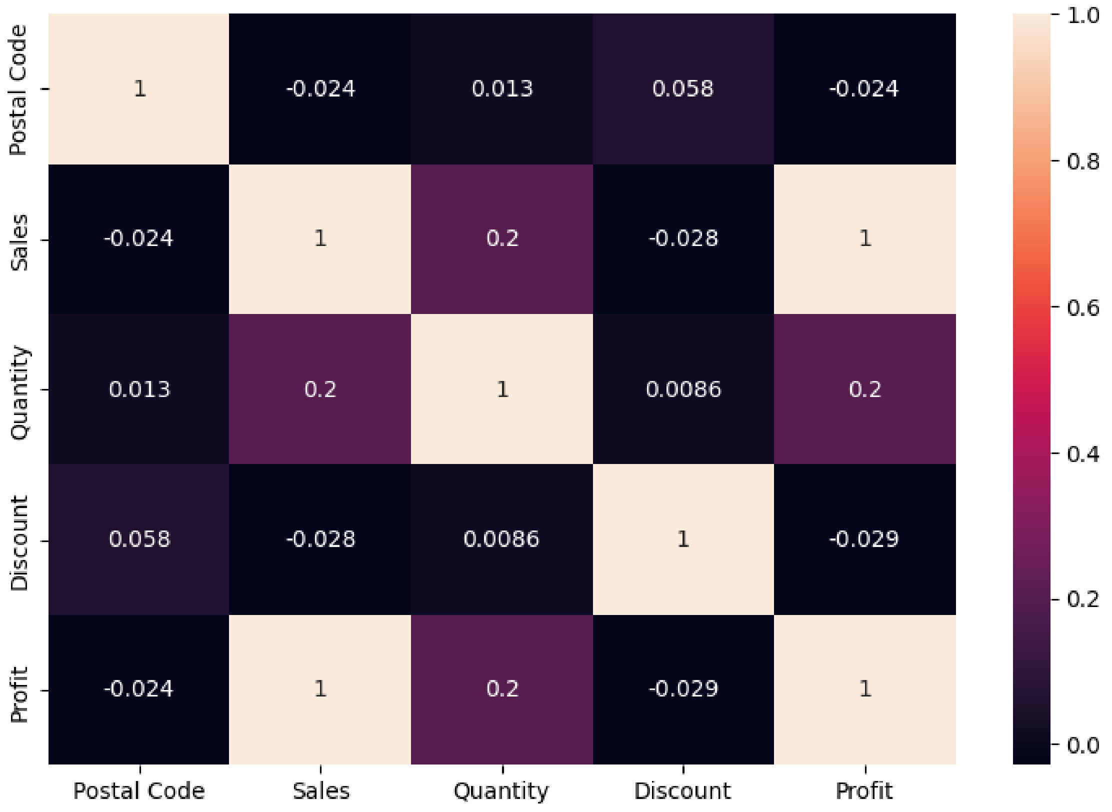
```
[ ]: state_s=clean_data.groupby(["State","City","Category","Sub-Category"])["Sales"].
       ↪sum().reset_index()
     fig = px.treemap(state_s, path=["State","City","Category","Sub-Category"],␣
       ↪values='Sales',
                       color='Sales',
                       color_continuous_scale='RdBu',
                       color_continuous_midpoint=np.average(state_s['Sales'],␣
       ↪weights=state_s['Sales']),
                     title="Fig 4.1-Total State/City X Sales With Category &␣
       ↪Sub-Category Distribution")
     fig.data[0].textinfo = 'label+text+value'

     fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
     fig.show()
```

```
[ ]: state_profit=clean_data[clean_data.Profit>0]
     state_profit=state_profit.
       ↪groupby(["State","City","Category","Sub-Category"])["Profit"].sum().
       ↪reset_index()
     fig = px.treemap(state_profit, path=["State","City","Category","Sub-Category"],␣
       ↪values='Profit',
                        color='Profit',
                        color_continuous_scale='RdBu',
                        color_continuous_midpoint=np.average(clean_data['Profit'],␣
       ↪weights=clean_data['Profit']),
                        title="Fig 4.2-Total State/City X Profit With Category &␣
       ↪Sub-Category Distribution")
     fig.data[0].textinfo = 'label+text+value'
     fig.layout.hovermode = False
     fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
     fig.show()
```

HeatMap of DataSet

```
[ ]: fig,axes = plt.subplots(1,1,figsize=(9,6))
     sns.heatmap(df.corr(), annot= True)
     plt.show()
```

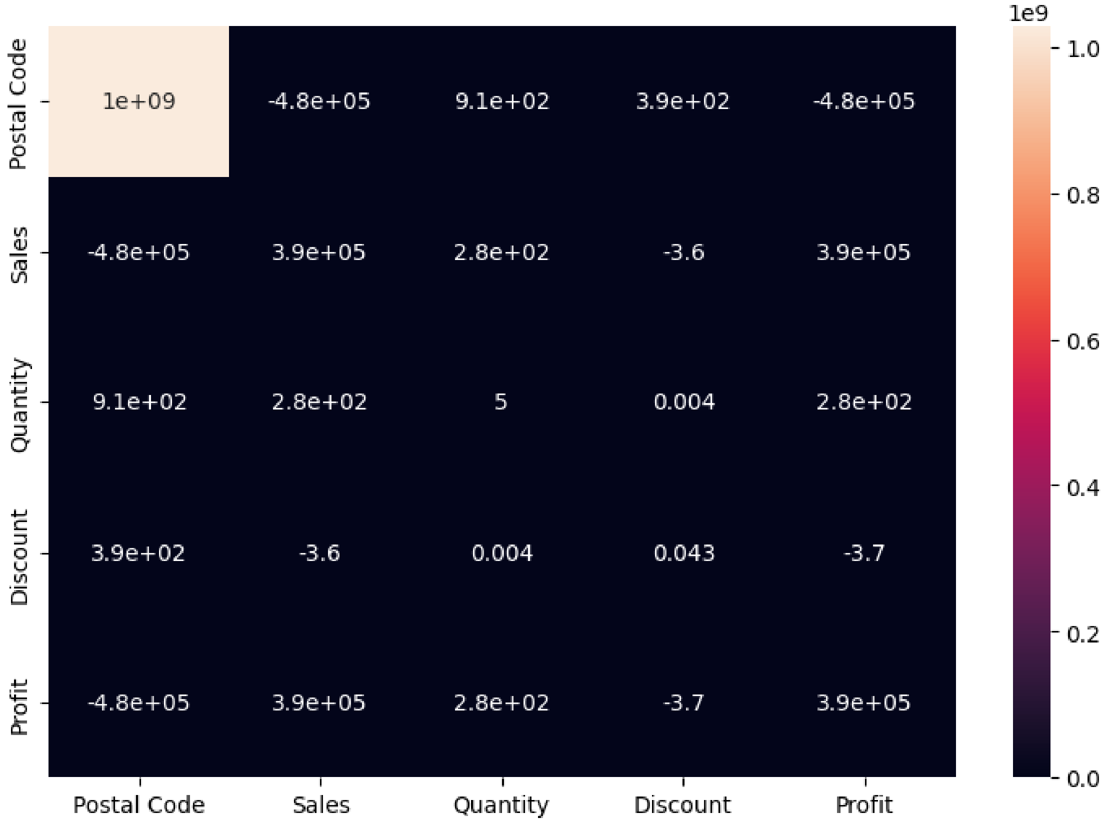
```
<ipython-input-62-7d19acb0cf78>:2: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
```



[ ]:

```
fig,axes = plt.subplots(1,1,figsize=(9,6))
sns.heatmap(df.cov(), annot= True)
plt.show()
```

```
<ipython-input-63-d3c100f5e8f3>:2: FutureWarning:

The default value of numeric_only in DataFrame.cov is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
```



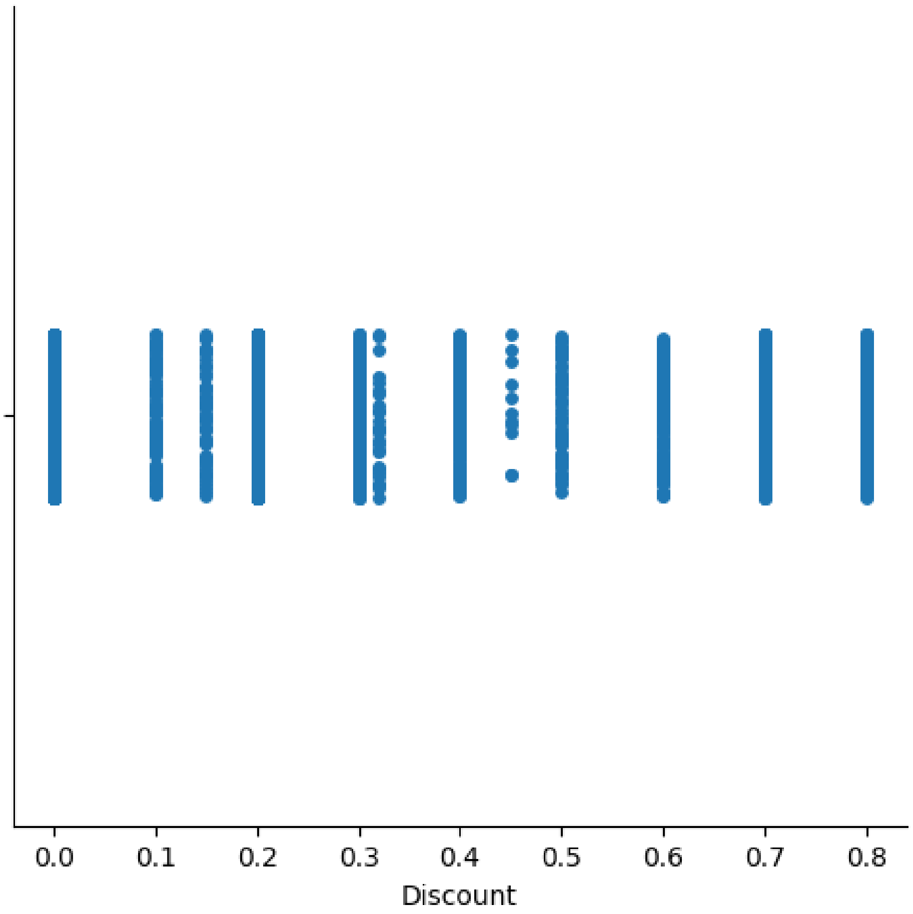
	Postal Code	Sales	Quantity	Discount	Profit
Postal Code	1e+09	-4.8e+05	9.1e+02	3.9e+02	-4.8e+05
Sales	-4.8e+05	3.9e+05	2.8e+02	-3.6	3.9e+05
Quantity	9.1e+02	2.8e+02	5	0.004	2.8e+02
Discount	3.9e+02	-3.6	0.004	0.043	-3.7
Profit	-4.8e+05	3.9e+05	2.8e+02	-3.7	3.9e+05

[ ]:

```
sns.catplot(x=df['Discount'])
```

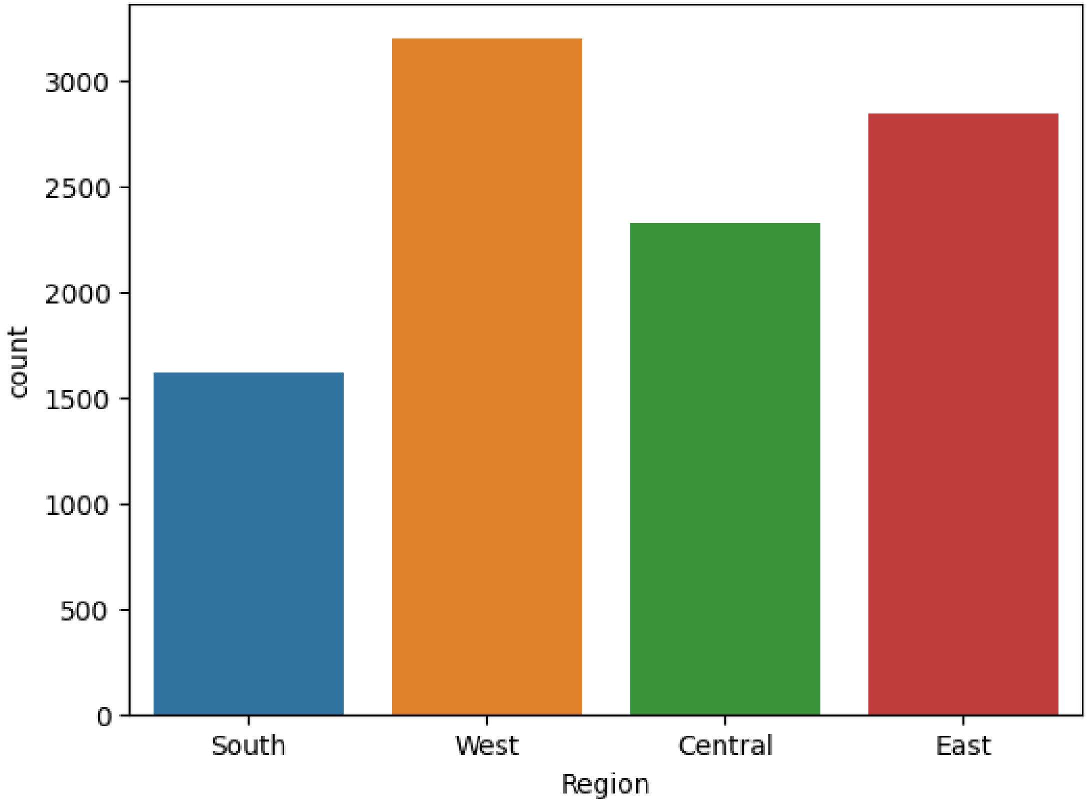
[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7d0fd6b57640>
```



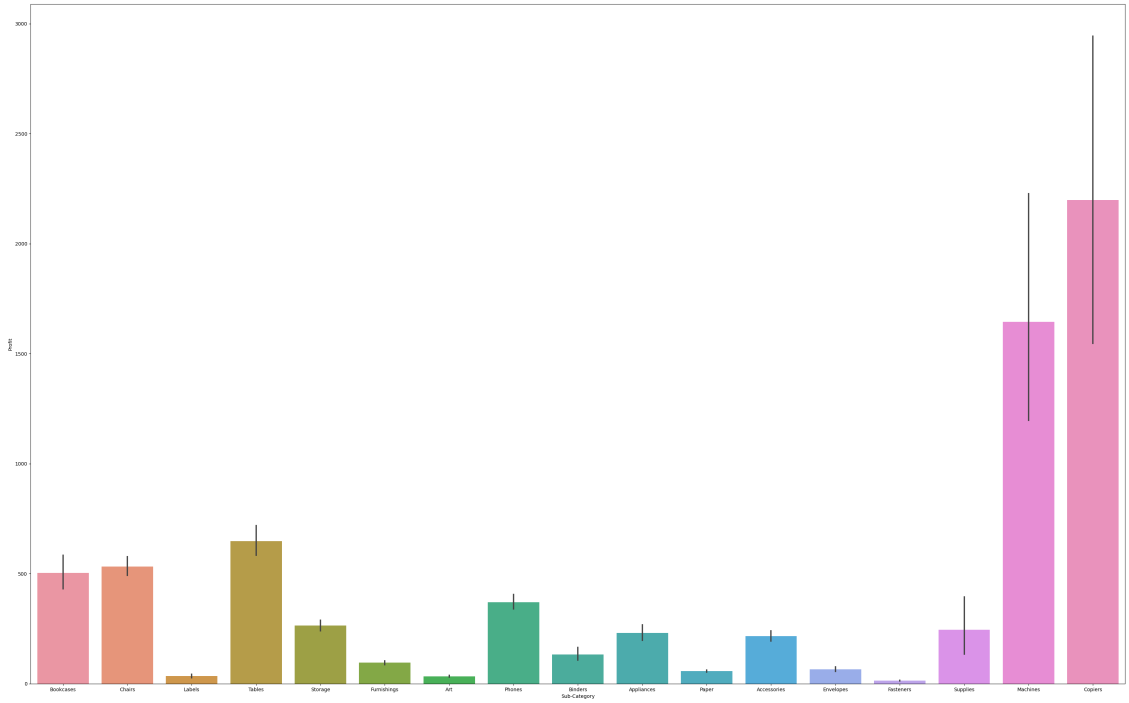
```
[ ]: sns.countplot(x=df['Region'])
```

```
[ ]: <Axes: xlabel='Region', ylabel='count'>
```



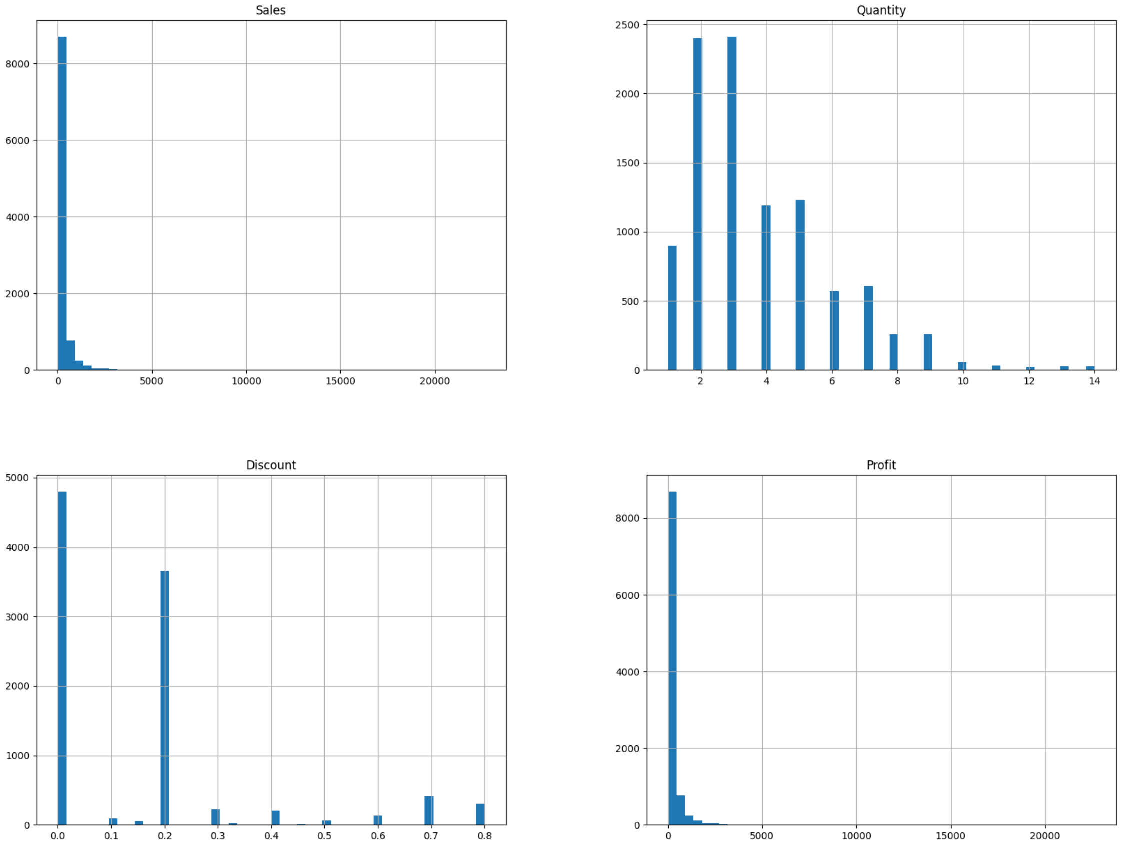
```
[ ]: plt.figure(figsize=(40,25))
     sns.barplot(x=df['Sub-Category'], y=df['Profit'])
```

```
[ ]: <Axes: xlabel='Sub-Category', ylabel='Profit'>
```



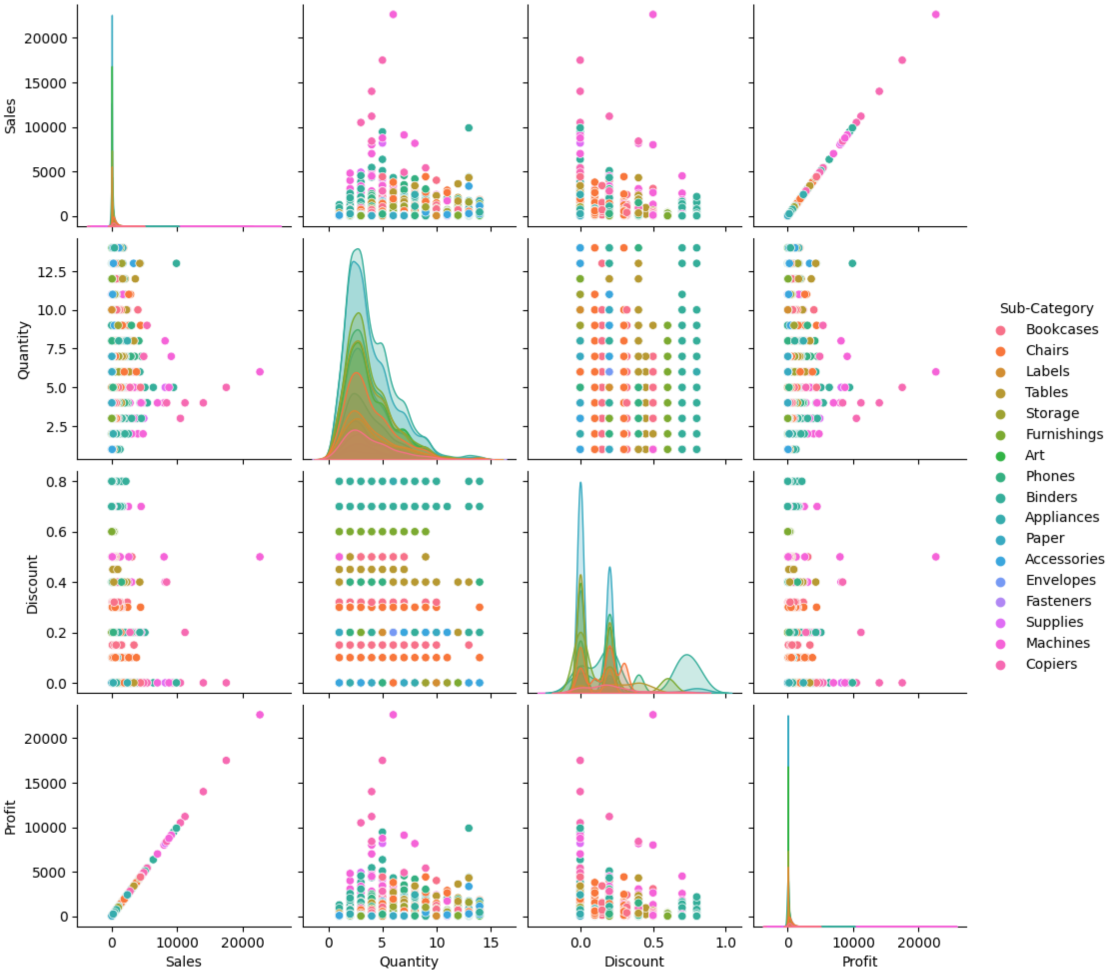
[ ]:

```
df1.hist(bins=50 ,figsize=(20,15))
plt.show()
```



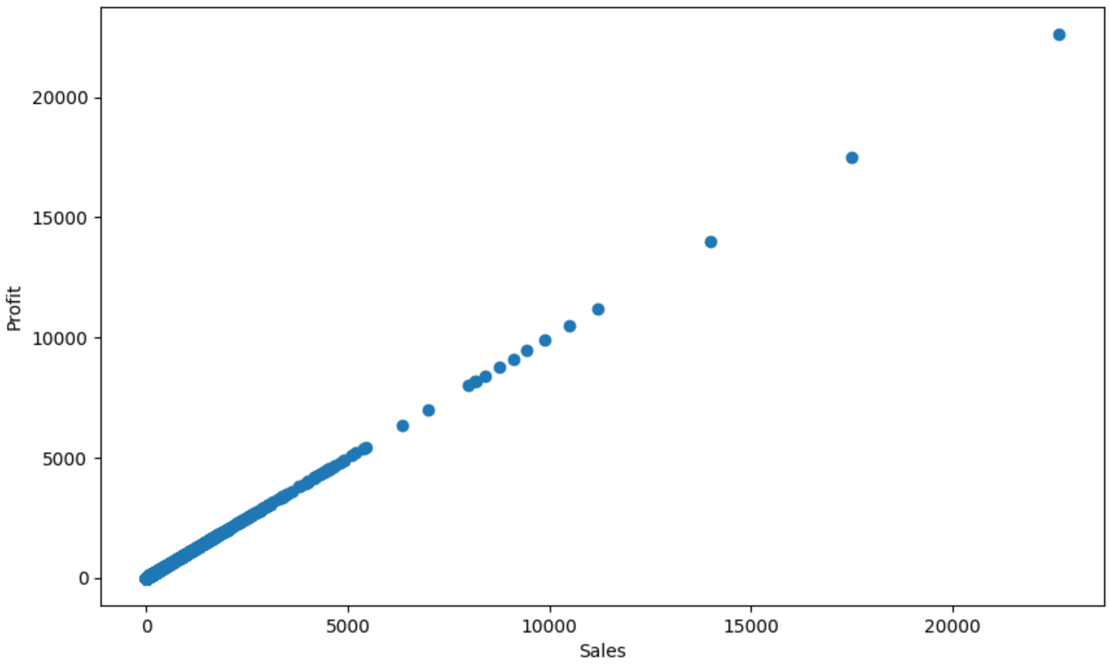
[77]:
```
figsize=(10,10)
sns.pairplot(df1,hue='Sub-Category')
```

[77]:
```
<seaborn.axisgrid.PairGrid at 0x7d0fd1b194b0>
```



SCATTER PLOT

[78]:
```
fig, ax = plt.subplots(figsize = (10 , 6))
ax.scatter(df["Sales"] , df["Profit"])
ax.set_xlabel('Sales')
ax.set_ylabel('Profit')
plt.show()
```

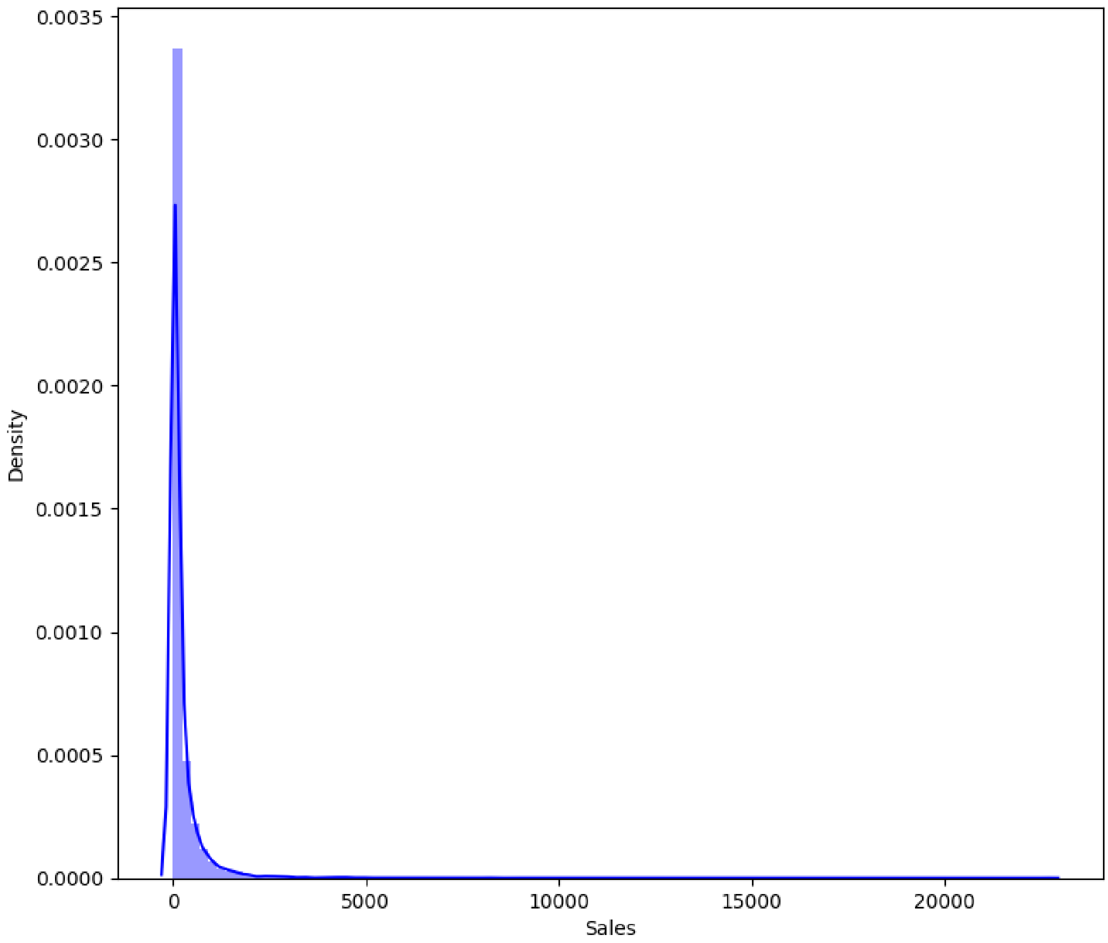


Distribution Plot

[79]:
```
print(df['Sales'].describe())
plt.figure(figsize = (9 , 8))
sns.distplot(df['Sales'], color = 'b', bins = 100, hist_kws = {'alpha': 0.4});
```

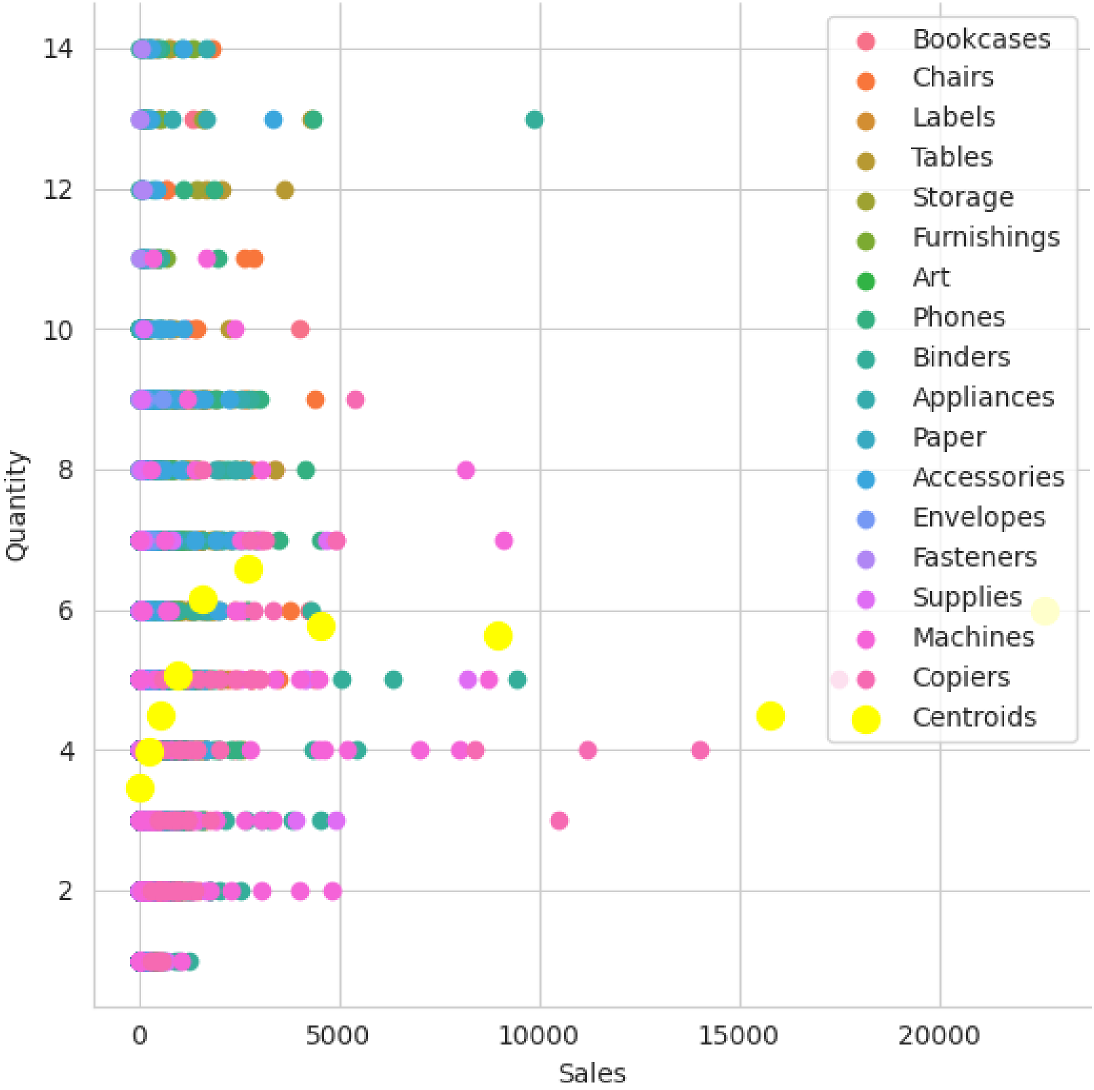
```
count     9994.000000
mean       229.858001
              …
75%        209.940000
max      22638.480000
Name: Sales, Length: 8, dtype: float64
```

```
<ipython-input-79-036949231e31>:3: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```



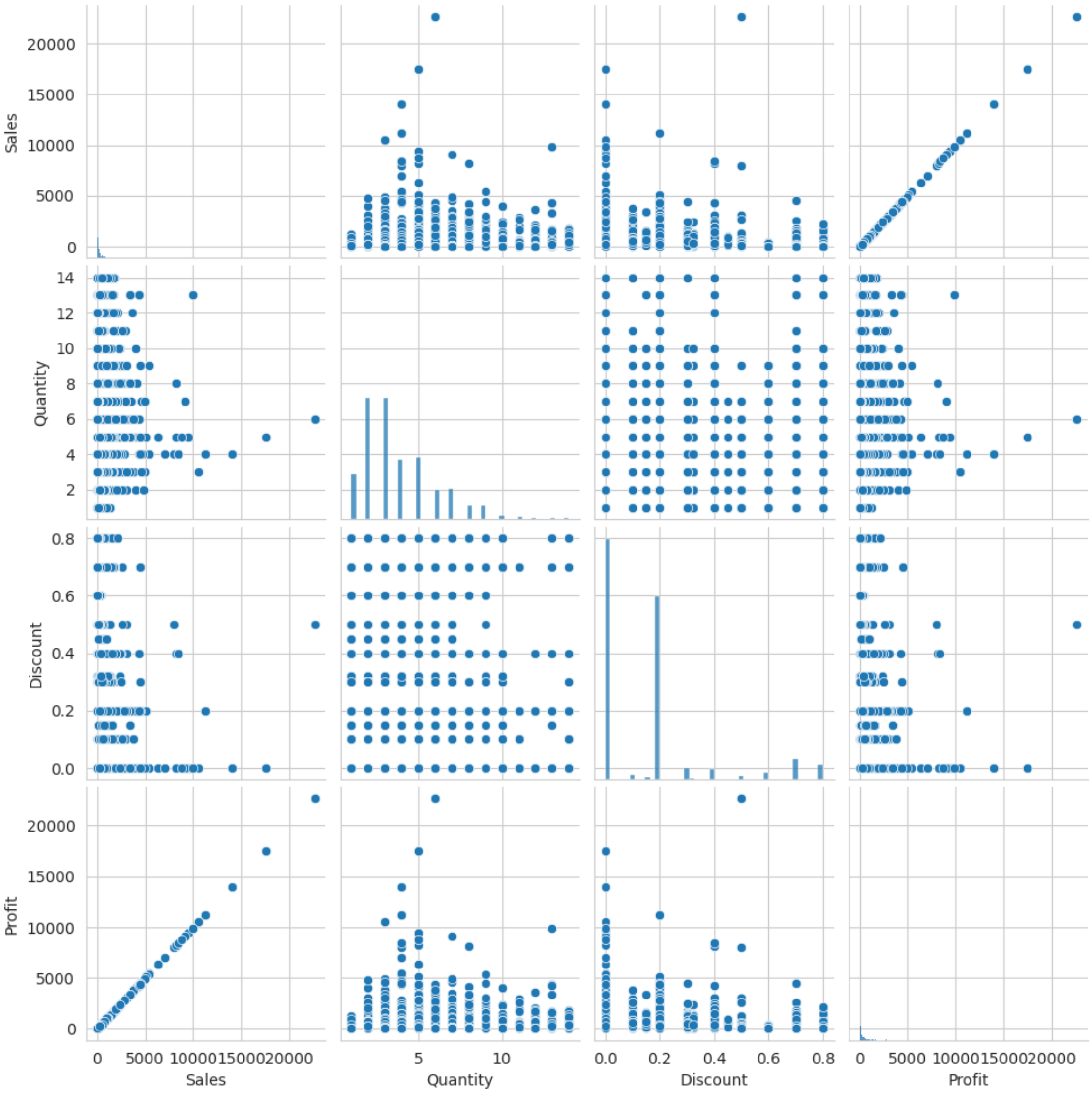
[80]:
```
x = df.iloc[:, [9, 10, 11, 12]].values
from sklearn.cluster import KMeans
wcss = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
                    max_iter = 300, n_init = 10, random_state = 0).fit(x)
    wcss.append(kmeans.inertia_)

sns.set_style("whitegrid")
sns.FacetGrid(df, hue ="Sub-Category",height = 6).map(plt.
 ↪scatter,'Sales','Quantity')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],
            s = 100, c = 'yellow', label = 'Centroids')

plt.legend()
```

```
[80]: <matplotlib.legend.Legend at 0x7d0fd0ecb070>
```



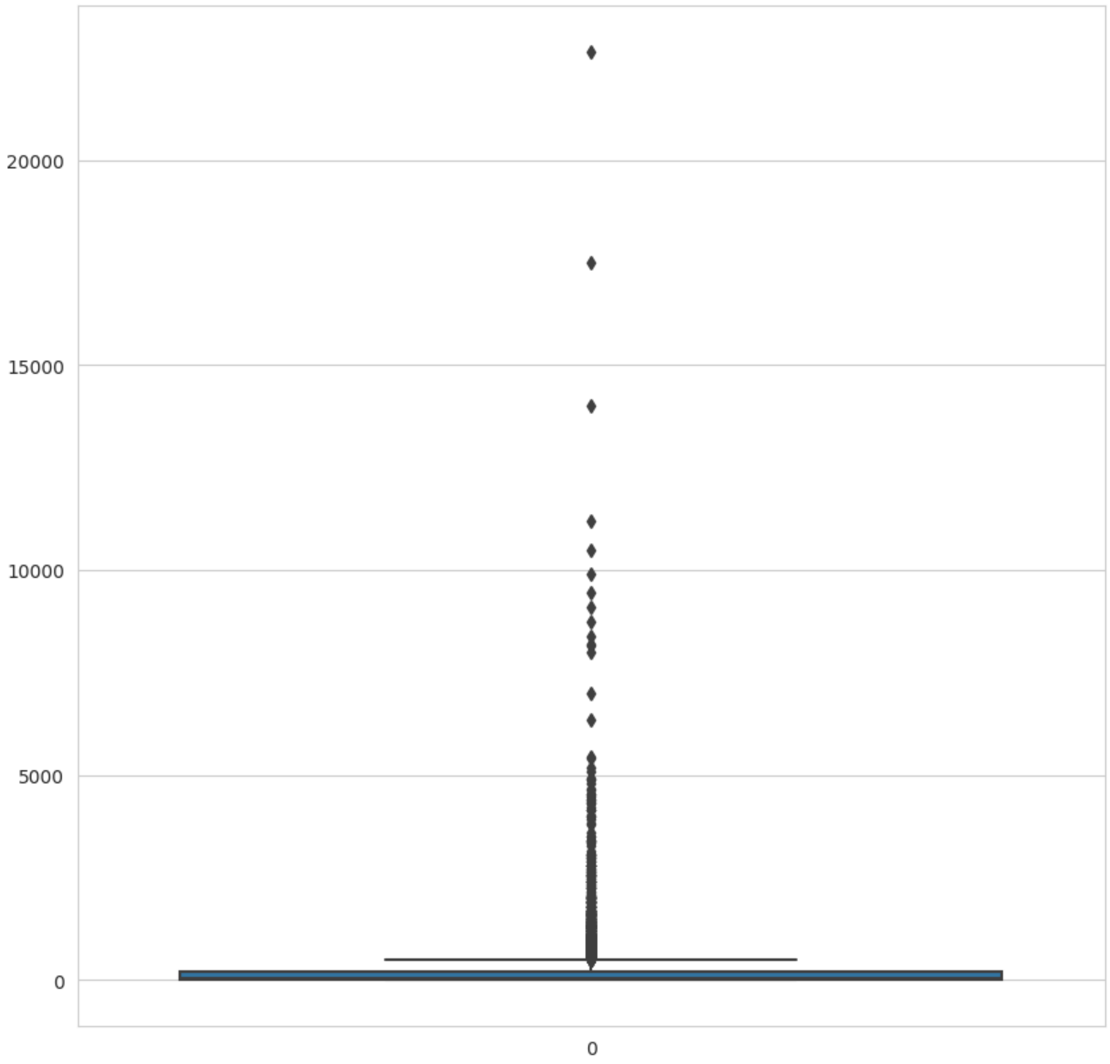
```
[81]: sns.pairplot(df1)
```

```
[81]: <seaborn.axisgrid.PairGrid at 0x7d0fd20625f0>
```



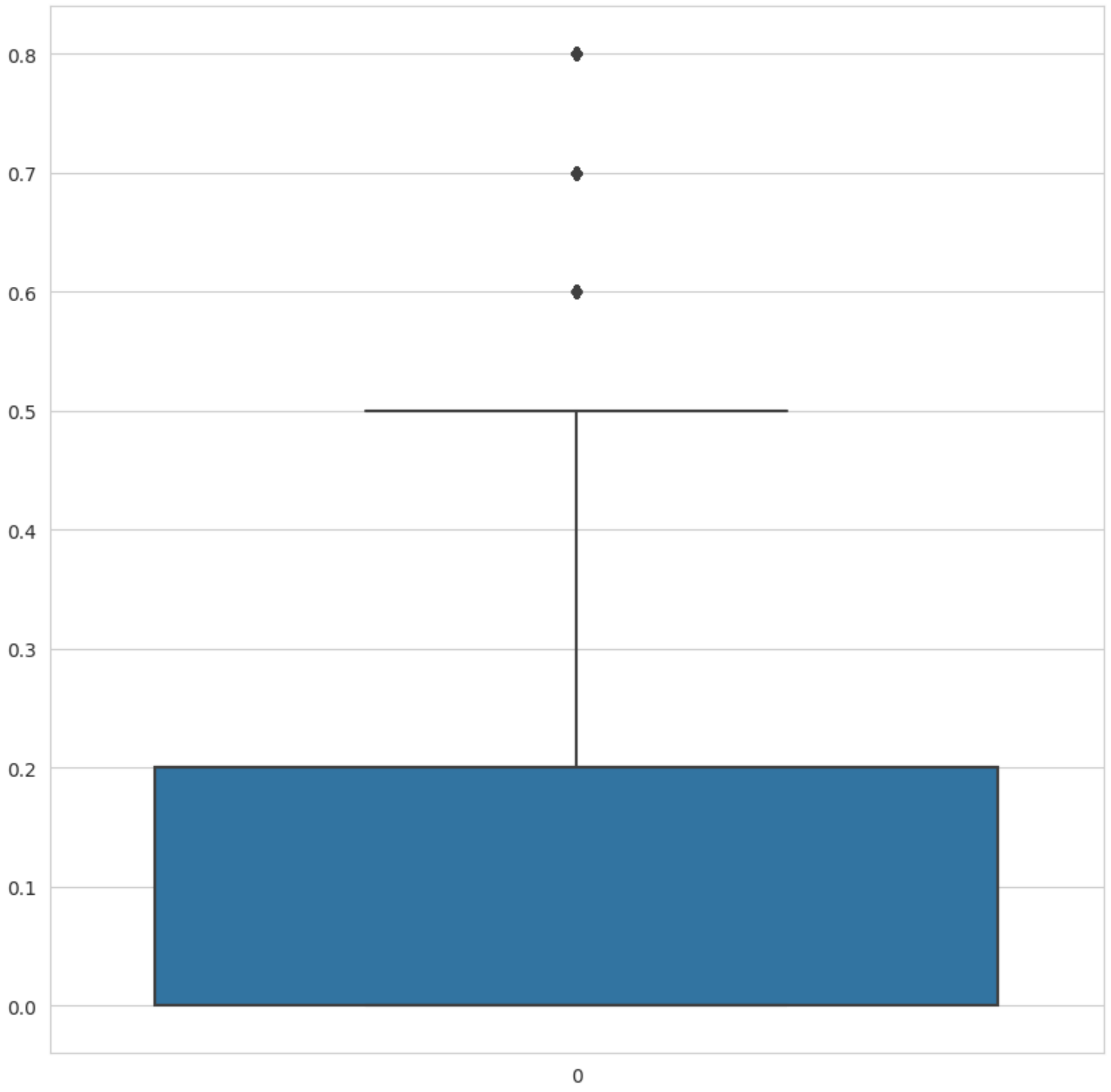
[82]:
```
fig, axes = plt.subplots(figsize = (10 , 10))

sns.boxplot(df['Sales'])
```

[82]:
```
<Axes: >
```



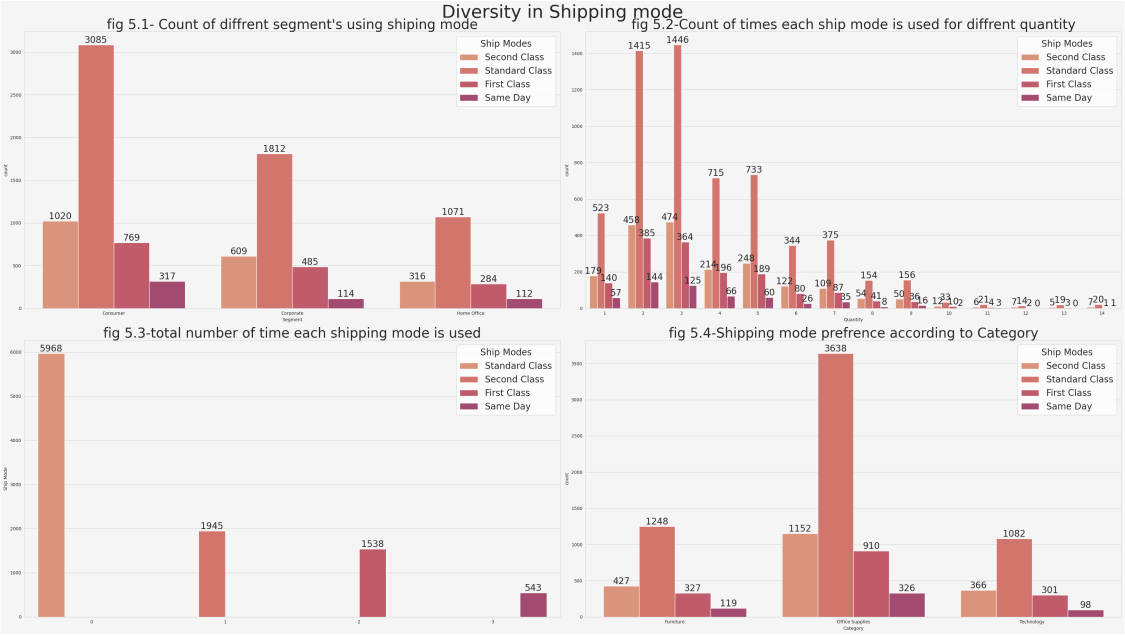
```
[83]: fig, axes = plt.subplots(figsize = (10 , 10))

      sns.boxplot(df['Discount'])
```

```
[83]: <Axes: >
```



[85]:
```
ship=ship=clean_data["Ship Mode"].value_counts().reset_index()
fig, axs = plt.subplots(2,2,figsize=(35,20))
fig.patch.set_facecolor('#f6f5f5')
sns.countplot(data=clean_data, x = "Segment",hue="Ship Mode",␣
 ↪palette=palette_color,ax=axs[0][0])
sns.countplot(data=clean_data, x = "Quantity",hue="Ship Mode",␣
 ↪palette=palette_color,ax=axs[0][1])
sns.barplot(data=ship, x = ship.index,y="Ship Mode",hue="index",␣
 ↪palette=palette_color,ax=axs[1][0])
sns.countplot(data=clean_data,x="Category",hue="Ship Mode",␣
 ↪palette=palette_color,ax=axs[1][1])
```

```
axs[0][0].set_title("fig 5.1- Count of diffrent segment's using shiping␣
 ↪mode",fontsize=30)
axs[0][0].set_facecolor('#f6f5f5')

axs[0][1].set_title("fig 5.2-Count of times each ship mode is used for diffrent␣
 ↪quantity",fontsize=30)
axs[0][1].set_facecolor('#f6f5f5')

axs[1][0].set_title("fig 5.3-total number of time each shipping mode is␣
 ↪used",fontsize=30)
axs[1][0].set_facecolor('#f6f5f5')

axs[1][1].set_title("fig 5.4-Shipping mode prefrence according to␣
 ↪Category",fontsize=30)
axs[1][1].set_facecolor('#f6f5f5')

for m in range(2):
    for n in range(2):
        axs[m][n].legend(fontsize = '11',
           title = 'Ship Modes', title_fontsize = '20',
           prop={'size': 20},
           loc="upper right")
        for i in axs[m][n].containers:
            axs[m][n].bar_label(i, fontsize=20)


plt.suptitle("Diversity in Shipping mode",fontsize=40)



plt.tight_layout()
```



Conclusion (Act Phase) Exploratory Data Analysis (EDA) on the dataset "SampleSuperstore" has been given to us, and as company managers, we are to identify the areas that need improvement in order to increase profit. identifying the weak points in the sales department to increase sales Based on the information discovered during the data analysis, the following techniques have been suggested.

```
[ ]:
```