Project : EDA on Superstore Dataset

```
#Python basics

#Libraries: numpy[numeric calcuations], pandas [data manipulation], matplotli, seaborn [visualization]
```

```
#Objective: Exploratory Data Analysis [EDA]

# I performed Exploratory data analysis [EDA] on superstore dataset using Python.
# In Data Analysis  advanced libraries of python Numpy, Pandas, Matplotlib, Seaborn for data manipulation and visualization purpose.
# In EDA Univariate Analysis and Bivariate Analysis is performed and results are visualizaed.
```

```
#Import libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
pip install xlrd
```

```
Collecting xlrd
  Downloading xlrd-2.0.1-py2.py3-none-any.whl.metadata (3.4 kB)
Downloading xlrd-2.0.1-py2.py3-none-any.whl (96 kB)
   ---------------------------------------- 0.0/96.5 kB ? eta -:--:--
   ---- ----------------------------------- 10.2/96.5 kB ? eta -:--:--
   ------------ --------------------------- 30.7/96.5 kB 435.7 kB/s eta 0:00:01
   ---------------- ----------------------- 41.0/96.5 kB 393.8 kB/s eta 0:00:01
   ----------------------------- ---------- 71.7/96.5 kB 491.5 kB/s eta 0:00:01
   -------------------------------------- - 92.2/96.5 kB 476.3 kB/s eta 0:00:01
   ---------------------------------------- 96.5/96.5 kB 394.0 kB/s eta 0:00:00
Installing collected packages: xlrd
Successfully installed xlrd-2.0.1
Note: you may need to restart the kernel to use updated packages.
```

```
#read dataset

df=pd.read_excel('superstore.xls')
```

```
#display data

df
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9989	9990	CA-2014-110422	2014-01-21	2014-01-23	Second Class	TB-21400	Tom Boeckenhauer	Consumer	United States	Miami	...	33180	South	FUR-FU-10001889
9990	9991	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	...	92627	West	FUR-FU-10000747
9991	9992	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	...	92627	West	TEC-PH-10003645
9992	9993	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	...	92627	West	OFF-PA-10004041
9993	9994	CA-2017-119914	2017-05-04	2017-05-09	Second Class	CC-12220	Chris Cortes	Consumer	United States	Westminster	...	92683	West	OFF-AP-10002684

9994 rows × 21 columns

```
#2. Display total number of rows and columns

df.shape
print("Total Number of Rows: ",df.shape[0])
print("Total Number of Coulmns: ",df.shape[1])
```

```
Total Number of Rows:  9994
Total Number of Coulmns:  21
```

```
#3. Display first rows.   ---- head()
#4. Display last 10 rows  ---- tail()
#5. Display random 10 rows. ---- sample()
```

```
#3. Display first rows.   ---- head()

df.head(10)
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	E
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies	
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture	
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760	Office Supplies	
5	6	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90032	West	FUR-FU-10001487	Furniture	Fu
6	7	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90032	West	OFF-AR-10002833	Office Supplies	
7	8	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90032	West	TEC-PH-10002275	Technology	
8	9	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90032	West	OFF-BI-10003910	Office Supplies	
9	10	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90032	West	OFF-AP-10002892	Office Supplies	A

10 rows × 21 columns

```
#4. Display last 10 rows  ---- tail()

df.tail(10)
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	
9984	9985	CA-2015-100251	2015-05-17	2015-05-23	Standard Class	DV-13465	Dianna Vittorini	Consumer	United States	Long Beach	...	11561	East	OFF-LA-10003766	
9985	9986	CA-2015-100251	2015-05-17	2015-05-23	Standard Class	DV-13465	Dianna Vittorini	Consumer	United States	Long Beach	...	11561	East	OFF-SU-10000898	
9986	9987	CA-2016-125794	2016-09-29	2016-10-03	Standard Class	ML-17410	Maris LaWare	Consumer	United States	Los Angeles	...	90008	West	TEC-AC-10003399	T
9987	9988	CA-2017-163629	2017-11-17	2017-11-21	Standard Class	RA-19885	Ruben Ausman	Corporate	United States	Athens	...	30605	South	TEC-AC-10001539	T
9988	9989	CA-2017-163629	2017-11-17	2017-11-21	Standard Class	RA-19885	Ruben Ausman	Corporate	United States	Athens	...	30605	South	TEC-PH-10004006	T
9989	9990	CA-2014-110422	2014-01-21	2014-01-23	Second Class	TB-21400	Tom Boeckenhauer	Consumer	United States	Miami	...	33180	South	FUR-FU-10001889	
9990	9991	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	...	92627	West	FUR-FU-10000747	
9991	9992	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	...	92627	West	TEC-PH-10003645	T
9992	9993	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	...	92627	West	OFF-PA-10004041	
9993	9994	CA-2017-119914	2017-05-04	2017-05-09	Second Class	CC-12220	Chris Cortes	Consumer	United States	Westminster	...	92683	West	OFF-AP-10002684	

10 rows × 21 columns

```
#5. Display random 10 rows. ---- sample()

df.sample(10)
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Categor
1250	1251	CA-2017-115602	2017-12-18	2017-12-24	Standard Class	DJ-13630	Doug Jacobs	Consumer	United States	New York City	...	10009	East	OFF-AP-10000891	Offi Suppli
6605	6606	CA-2016-148852	2016-05-26	2016-05-31	Standard Class	SV-20785	Stewart Visinsky	Consumer	United States	Santa Ana	...	92704	West	OFF-PA-10001293	Offi Suppli
9944	9945	CA-2015-145415	2015-04-12	2015-04-17	Second Class	RD-19660	Robert Dilbeck	Home Office	United States	Seattle	...	98103	West	OFF-ST-10000419	Offi Suppli
2437	2438	CA-2014-164210	2014-11-18	2014-11-20	Second Class	PW-19240	Pierre Wener	Consumer	United States	Louisville	...	80027	West	FUR-TA-10000849	Furnitu
2633	2634	US-2017-110604	2017-05-15	2017-05-20	Standard Class	JF-15295	Jason Fortune-	Consumer	United States	Seattle	...	98103	West	FUR-CH-10004287	Furnitu
4470	4471	CA-2017-156895	2017-05-08	2017-05-12	Standard Class	DJ-13510	Don Jones	Corporate	United States	Philadelphia	...	19140	East	FUR-CH-10003535	Furnitu
4293	4294	CA-2017-130967	2017-03-10	2017-03-16	Standard Class	EB-13870	Emily Burns	Consumer	United States	Philadelphia	...	19140	East	TEC-PH-10004896	Technolo
210	211	CA-2017-135860	2017-12-01	2017-12-07	Standard Class	JH-15985	Joseph Holt	Consumer	United States	Saginaw	...	48601	Central	OFF-ST-10001522	Offi Suppli
1437	1438	CA-2015-139731	2015-10-15	2015-10-15	Same Day	JE-15745	Joel Eaton	Consumer	United States	Amarillo	...	79109	Central	TEC-AC-10004975	Technolo
4355	4356	CA-2015-155600	2015-12-04	2015-12-07	Second Class	RO-19780	Rose O'Brian	Consumer	United States	Clarksville	...	37042	South	OFF-BI-10000545	Offi Suppli

10 rows × 21 columns

```
# 6. Display all column names

df.columns
```

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

```
#7. Display datatypes of each column

df.dtypes
```

```
Row ID                    int64
Order ID                 object
Order Date       datetime64[ns]
Ship Date        datetime64[ns]
Ship Mode                object
Customer ID              object
Customer Name            object
Segment                  object
Country                  object
City                     object
State                    object
Postal Code               int64
Region                   object
Product ID               object
Category                 object
Sub-Category             object
Product Name             object
Sales                   float64
Quantity                  int64
Discount                float64
```

```
Profit                  float64
dtype: object
```

```
#8.Identify whether any columns has null values

df.isnull().sum()
```

```
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
```

```
#9.
```

```
#Categorical Features
#Continous Features
```

```
#1. Order Id
#unique
#nunique--count of unique

df['Order ID'].nunique()
```

```
5009
```

```
df.nunique()
```

```
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            6144
Quantity           14
Discount           12
Profit           7545
dtype: int64
```

```
#Identify categorical and continous features.
cat=[]
cont=[]

threshold=20

for i in df.columns:
    if df[i].nunique() >= 20:
        cont.append(i)
    else :
        cat.append(i)
```

```
print("Categorical Features: ",cat)
print()
print("Continous Features: ",cont)
```

```
Categorical Features:  ['Ship Mode', 'Segment', 'Country', 'Region', 'Category', 'Sub-Category', 'Quantity', 'Discount']

Continous Features:  ['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Customer ID', 'Customer Name', 'City', 'State', 'Postal Code
```

```
#10. Display Unique values for each categorical feature.

for i in cat:
    print(i,"\t",df[i].unique())
    print()
```

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

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

Country          ['United States']

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']

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]
```

```
# 11. Display unique values in city and state

print("city: ")
print(df['City'].unique())
```

```
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'
```

```
print("State : ")
print(df['State'].unique())
```

```
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']
```

Uni Variate Analysis

```
cat
```

```
['Ship Mode',
 'Segment',
 'Country',
 'Region',
 'Category',
 'Sub-Category',
 'Quantity',
 'Discount']
```

```
#1. Display count of orders through each shipment mode.

df['Ship Mode'].value_counts()
```

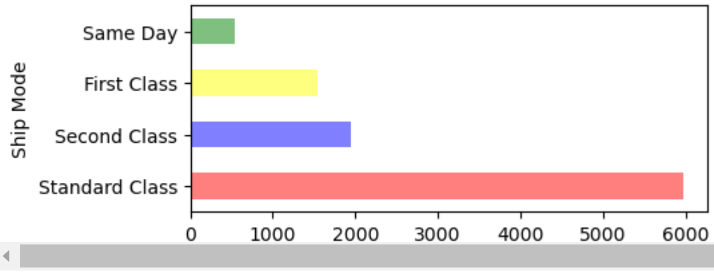
```
Ship Mode
Standard Class    5968
Second Class      1945
First Class       1538
Same Day           543
Name: count, dtype: int64
```

```
#Count of shipment mode in percentages

np.round(df['Ship Mode'].value_counts() / df.shape[0] * 100 , 2)
```

```
Ship Mode
Standard Class    59.72
Second Class      19.46
First Class       15.39
Same Day           5.43
Name: count, dtype: float64
```

```
# Visualization

plt.figure(figsize=(5,2))
df['Ship Mode'].value_counts().plot.barh(color=['red','blue','yellow','green'],alpha=.5)
```

```
<Axes: ylabel='Ship Mode'>
```



```
# 1.Display region wise order count

df['Region'].value_counts()
```

```
Region
West       3203
East       2848
Central    2323
South      1620
Name: count, dtype: int64
```

```
# 2. % Terms

np.round(df['Region'].value_counts() / df.shape[0] * 100 , 2)
```

```
Region
West       32.05
East       28.50
Central    23.24
South      16.21
Name: count, dtype: float64
```

```
# 3.Display ordre count by each segment

df['Segment'].value_counts()
```

```
Segment
Consumer       5191
Corporate      3020
Home Office    1783
Name: count, dtype: int64
```

```
df['Segment'].value_counts() / df.shape[0] * 100
```

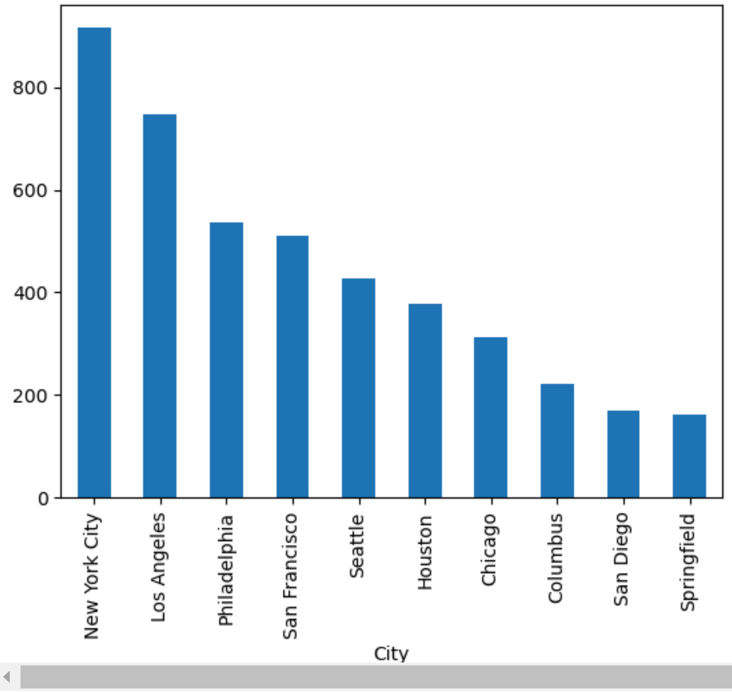
```
Segment
Consumer       51.941165
Corporate      30.218131
Home Office    17.840704
Name: count, dtype: float64
```

```
#4. Display top 5 states by order count

df['City'].value_counts()[:10]
```

```
City
New York City    915
Los Angeles      747
Philadelphia     537
San Francisco    510
Seattle          428
Houston          377
Chicago          314
Columbus         222
San Diego        170
Springfield      163
Name: count, dtype: int64
```

```
df['City'].value_counts()[:10].plot.bar()
```

```
<Axes: xlabel='City'>
```



```
cont
```

```
['Row ID',
 'Order ID',
 'Order Date',
 'Ship Date',
 'Customer ID',
 'Customer Name',
 'City',
 'State',
 'Postal Code',
 'Product ID',
 'Product Name',
 'Sales',
 'Profit']
```

```
# Display total sales.     sum()

print("Total Sales Values: ",df['Sales'].sum().round())
```

```
Total Sales Values:  2297201.0
```

```
#Display Total Profit

print(" Total Profit Value: ",df['Profit'].sum().round())
```

```
 Total Profit Value:  286397.0
```

```
#Display % Profit

print("% Profit : ", np.round(df['Profit'].sum().round() / df['Sales'].sum().round() * 100  , 2 ) , "%" )
```

```
% Profit :  12.47 %
```

2. Bi Variate Analysis

```
#group by()
```

```
cat
```

```
['Ship Mode',
 'Segment',
 'Country',
 'Region',
 'Category',
 'Sub-Category',
 'Quantity',
 'Discount']
```

```
#1. Display Region wise --- i. Total sales    ii.Total profit
```

```
#Context: 'Region' , 'Sales'
```

```
#1. Total Region wise Sales
```

```
df.groupby('Region')['Sales'].sum().round()
```

```
Region
Central    501240.0
East       678781.0
South      391722.0
West       725458.0
Name: Sales, dtype: float64
```