# Final Project - Analyzing Sales Data

**Date**: 29 January 2023

**Author**: Tanapol Yootaworn

**Course**: `Pandas Foundation`

```
# import data
import pandas as pd
df = pd.read_csv("sample-store.csv")
```

```
# preview top 5 rows
df.head()
```

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country/Region | City |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CA-2019-152156 | 11/8/2019 | 11/11/2019 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Hend |
| 1 | 2 | CA-2019-152156 | 11/8/2019 | 11/11/2019 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Hend |
| 2 | 3 | CA-2019-138688 | 6/12/2019 | 6/16/2019 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Ange |
| 3 | 4 | US-2018-108966 | 10/11/2018 | 10/18/2018 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Laude |
| 4 | 5 | US-2018-108966 | 10/11/2018 | 10/18/2018 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Laude |

5 rows × 21 columns

```
# shape of dataframe
df.shape
```

```
(9994, 21)
```

```
# see data frame information using .info()
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/Region  9994 non-null   object
 9   City            9994 non-null   object
 10  State           9994 non-null   object
 11  Postal Code     9983 non-null   float64
 12  Region          9994 non-null   object
 13  Product ID      9994 non-null   object
 14  Category        9994 non-null   object
```

We can use `pd.to_datetime()` function to convert columns 'Order Date' and 'Ship Date' to datetime.

```
# example of pd.to_datetime() function
pd.to_datetime(df['Order Date'].head(), format='%m/%d/%Y')
```

```
0   2019-11-08
1   2019-11-08
2   2019-06-12
3   2018-10-11
4   2018-10-11
Name: Order Date, dtype: datetime64[ns]
```

```
# TODO - convert order date and ship date to datetime in the original dataframe
df['Order Date'] = pd.to_datetime(df['Order Date'], format='%m/%d/%Y')
df['Ship Date'] = pd.to_datetime(df['Ship Date'], format='%m/%d/%Y')

# preview 5 rows of column order date and ship date.
df[['Order Date', 'Ship Date']].head()
```

| | Order Date | Ship Date |
|---|---|---|
| 0 | 2019-11-08 | 2019-11-11 |
| 1 | 2019-11-08 | 2019-11-11 |
| 2 | 2019-06-12 | 2019-06-16 |
| 3 | 2018-10-11 | 2018-10-18 |
| 4 | 2018-10-11 | 2018-10-18 |

```
# TODO - count nan in postal code column
df['Postal Code'].isna().sum()
```

11

```
# TODO - filter rows with missing values
df[df['Postal Code'].isna()].head()
```

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country/Region | City |
|---|---|---|---|---|---|---|---|---|---|---|
| 2234 | 2235 | CA-2020-104066 | 2020-12-05 | 12/10/2020 | Standard Class | QJ-19255 | Quincy Jones | Corporate | United States | Burlir |
| 5274 | 5275 | CA-2018-162887 | 2018-11-07 | 11/9/2018 | Second Class | SV-20785 | Stewart Visinsky | Consumer | United States | Burlir |
| 8798 | 8799 | US-2019-150140 | 2019-04-06 | 4/10/2019 | Standard Class | VM-21685 | Valerie Mitchum | Home Office | United States | Burlir |
| 9146 | 9147 | US-2019-165505 | 2019-01-23 | 1/27/2019 | Standard Class | CB-12535 | Claudia Bergmann | Corporate | United States | Burlir |
| 9147 | 9148 | US-2019-165505 | 2019-01-23 | 1/27/2019 | Standard Class | CB-12535 | Claudia Bergmann | Corporate | United States | Burlir |

5 rows × 21 columns

```
# TODO - Explore this dataset on your owns, ask your own questions

# Which columns have missing values, how many ?
df.isna().sum()

# Answer: only column 'Postal Code' has 11 missing value rows
```

```
Row ID                 0
Order ID               0
Order Date             0
Ship Date              0
Ship Mode              0
Customer ID            0
Customer Name          0
Segment                0
Country/Region         0
City                   0
State                  0
Postal Code           11
Region                 0
Product ID             0
Category               0
Sub-Category           0
Product Name           0
Sales                  0
Quantity               0
Discount               0
Profit                 0
Order_Year             0
More Average Sales     0
dtype: int64
```

## Data Analysis Part

Answer 10 below questions to get credit from this course. Write `pandas` code to find answers.

```
# TODO 01 - how many columns, rows in this dataset
df.shape

# 9994 column, 21 rows
```

```
(9994, 21)
```

```
# TODO 02 - is there any missing values?, if there is, which colunm? how many nan

df.isna().sum()

## Answer: Yes, there is. In column 'Postal Code', has 11 nan value.
```

```
Row ID             0
Order ID           0
Order Date         0
Ship Date          0
Ship Mode          0
Customer ID        0
Customer Name      0
Segment            0
Country/Region     0
City               0
State              0
Postal Code       11
Region             0
Product ID         0
Category           0
Sub-Category       0
Product Name       0
Sales              0
Quantity           0
Discount           0
Profit             0
dtype: int64
```

```
# TODO 03 - your friend ask for `California` data, filter it and export csv for h
california_data = df.query('State == "California"')
california_data.to_csv('california.csv')
# preview top 5 rows
california_data.head()
```

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country/Region | City | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 3 | CA-2019-138688 | 2019-06-12 | 2019-06-16 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | ... |
| 5 | 6 | CA-2017-115812 | 2017-06-09 | 2017-06-14 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... |
| 6 | 7 | CA-2017-115812 | 2017-06-09 | 2017-06-14 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... |
| 7 | 8 | CA-2017-115812 | 2017-06-09 | 2017-06-14 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... |
| 8 | 9 | CA-2017-115812 | 2017-06-09 | 2017-06-14 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... |

5 rows × 21 columns

```
# TODO 04 - your friend ask for all order data in `California` and `Texas` in 201
df['Order Date'] = pd.to_datetime(df['Order Date'], format='%m/%d/%Y')

# All Data in 'California' and 'Texas'
df_cali_texas_2017 = df[(df["Order Date"].dt.strftime("%Y") == "2017") & (df["Sta

# Save csv file
df_cali_texas_2017.to_csv("df_cali_texas_2017.csv")

# preview top 5 rows
df_cali_texas_2017.head()
```

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country/Region | City | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 6 | CA-2017-115812 | 2017-06-09 | 6/14/2017 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... |
| 6 | 7 | CA-2017-115812 | 2017-06-09 | 6/14/2017 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... |
| 7 | 8 | CA-2017-115812 | 2017-06-09 | 6/14/2017 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... |
| 8 | 9 | CA-2017-115812 | 2017-06-09 | 6/14/2017 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... |
| 9 | 10 | CA-2017-115812 | 2017-06-09 | 6/14/2017 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... |

5 rows × 21 columns

```
# TODO 05 - how much total sales, average sales, and standard deviation of sales
df['Order Date'] = pd.to_datetime(df['Order Date'], format='%m/%d/%Y')

# filter data in 2017
df["Sales"][df["Order Date"].dt.strftime('%Y') == "2017"].agg(['sum', "mean", "st
```

```
sum     484247.50
mean       242.97
std        754.05
Name: Sales, dtype: float64
```

```
# TODO 06 - which Segment has the highest profit in 2018
df['Order Date'] = pd.to_datetime(df['Order Date'], format='%m/%d/%Y')

df[["Segment", "Profit"]][df["Order Date"].dt.strftime('%Y') == "2018"]\
    .sort_values(by=["Profit"], ascending=False).round(2).head()

# Answer: Segment=> Consumer = 3177.48
```

| | Segment | Profit |
|---|---|---|
| 509 | Consumer | 3177.48 |
| 8990 | Corporate | 2302.97 |
| 8204 | Corporate | 2229.02 |
| 7683 | Home Office | 1906.48 |
| 1644 | Corporate | 1480.47 |

```
# TODO 07 - which top 5 States have the least total sales between 15 April 2019 -
df['Order Date'] = pd.to_datetime(df['Order Date'], format='%m/%d/%Y')

# fiter date between 15 Api 2019 - 31 Dec 2019
df_apr_to_dec_2019 = df[(df["Order Date"] >= "2019-04-15") & (df["Order Date"] <=

df_state_sales = df_apr_to_dec_2019[["State", "Sales"]].groupby(by=['State']).sum

# sort ascending
df_state_sales.sort_values(by=["Sales"], ascending=True).head(5).reset_index()

# Answer: top 5 States have the least total sales between 15 Apr 2019 - 31 Dec 20

# 1.New Hampshire
# 2.New Mexico
# 3.District of Columbia
# 4.Louisiana
# 5.South Carolina
```

| | State | Sales |
|---|---|---|
| 0 | New Hampshire | 49.05 |
| 1 | New Mexico | 64.08 |
| 2 | District of Columbia | 117.07 |
| 3 | Louisiana | 249.80 |
| 4 | South Carolina | 502.48 |

```
# TODO 08 - what is the proportion of total sales (%) in West + Central in 2019 e

# Total Region
total_region = df['Sales'][df["Order Date"].dt.strftime('%Y') == "2019"].sum()

# West + Central
west_and_central_region = df['Sales'][(df["Region"].isin(["West", "Central"])) &

# the proportion total sales (%)
result = (west_and_central_region/total_region)*100

print(f"total sales in West and Central = {result.__round__()} %")
```

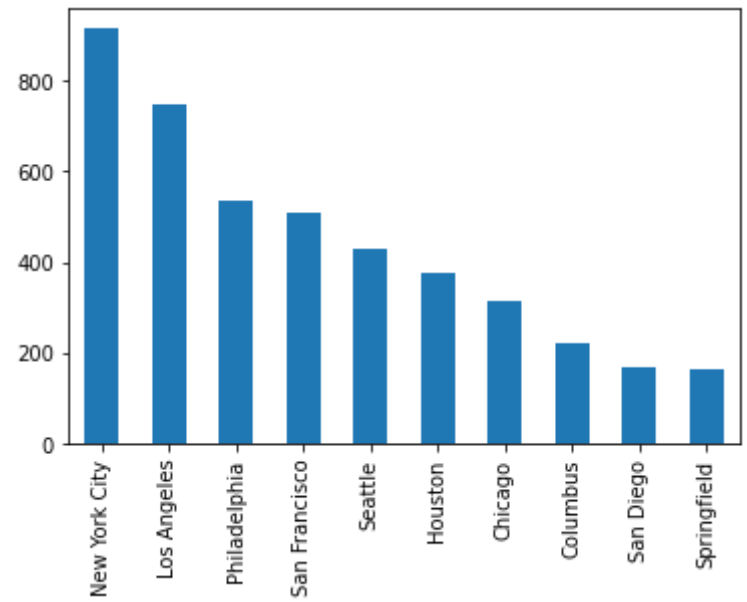
```
total sales in West and Central = 55 %
```

```
# TODO 09 - find top 10 popular products in terms of number of orders vs. total s
df['Order Date'] = pd.to_datetime(df['Order Date'], format='%m/%d/%Y')

# filter data 2019-2020
df_2019_to_2020 = df[(df["Order Date"] >= "2019-01-01") & (df['Order Date'] <= "2

# top 10 products in term of number of orders
df_2019_to_2020[["Product Name", "Quantity"]].groupby("Product Name")\
    .count().sort_values(by="Quantity", ascending=False).reset_index().head(10)
```

| | Product Name | Quantity |
|---|---|---|
| 0 | Easy-staple paper | 26 |
| 1 | Staples | 23 |
| 2 | Staple envelope | 21 |
| 3 | Chromcraft Round Conference Tables | 12 |
| 4 | Staples in misc. colors | 11 |
| 5 | Global Wood Trimmed Manager's Task Chair, Khaki | 11 |
| 6 | Avery Non-Stick Binders | 11 |
| 7 | Staple remover | 10 |
| 8 | GBC Instant Report Kit | 10 |
| 9 | Storex Dura Pro Binders | 10 |

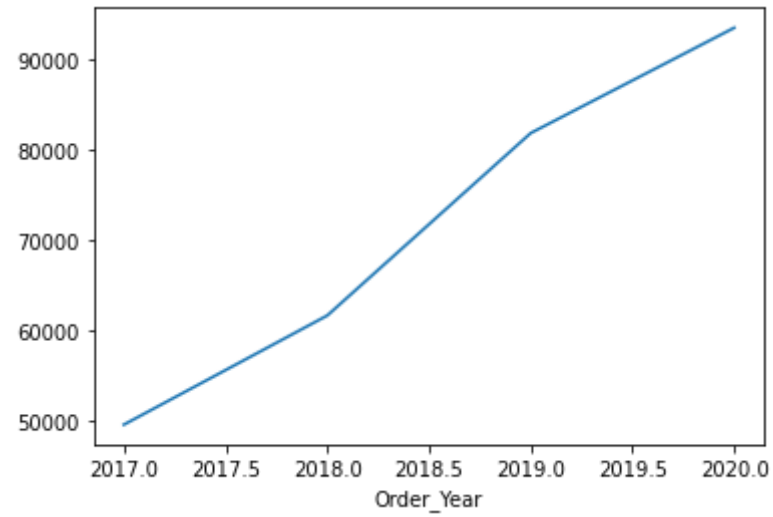
```
# TODO 10 - plot at least 2 plots, any plot you think interesting :)

# histogram top 10 city most sales
df['City'].value_counts().head(10).plot(kind='bar');
```

Download




```
# line plot show company's profits between 2017 - 2020
df['Order_Year'] = df['Order Date'].dt.year
df.groupby("Order_Year")["Profit"].sum().__round__(0).plot(kind='line');
```

Download




```
# TODO Bonus - use np.where() to create new column in dataframe to help you answe
import numpy as np

# which sales are more than average sales?
sales_avg = np.mean(df['Sales']).round() # 230 approximately

df['More Average Sales'] = np.where(df["Sales"] >= sales_avg , True, False)
df.head()
```

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country/Region | City |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CA-2019-152156 | 2019-11-08 | 11/11/2019 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderso |
| 1 | 2 | CA-2019-152156 | 2019-11-08 | 11/11/2019 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderso |
| 2 | 3 | CA-2019-138688 | 2019-06-12 | 6/16/2019 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles |
| 3 | 4 | US-2018-108966 | 2018-10-11 | 10/18/2018 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdal |
| 4 | 5 | US-2018-108966 | 2018-10-11 | 10/18/2018 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdal |

5 rows × 23 columns