Final Project - Analyzing Sales Data

Date: 17 January 2023

Author: Phiranat Nawachindaphan

Course: `Pandas Foundation`

```
# import data
import pandas as pd
import numpy as np
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	Henc
1	2	CA-2019-152156	11/8/2019	11/11/2019	Second Class	CG-12520	Claire Gute	Consumer	United States	Henc
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 Laud
4	5	US-2018-108966	10/11/2018	10/18/2018	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Laud

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 datafram
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')
df
```

	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	2019-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henders
1	2	CA-2019-152156	2019-11-08	2019-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henders
2	3	CA-2019-138688	2019-06-12	2019-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Ang
3	4	US-2018-108966	2018-10-11	2018-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderd
4	5	US-2018-108966	2018-10-11	2018-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderd
...	...	...	...	...	...	...	...	...	...	...
9989	9990	CA-2017-110422	2017-01-21	2017-01-23	Second Class	TB-21400	Tom Boeckenhauer	Consumer	United States	Miami
9990	9991	CA-2020-121258	2020-02-26	2020-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa M
9991	9992	CA-2020-121258	2020-02-26	2020-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa M
9992	9993	CA-2020-121258	2020-02-26	2020-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa M
9993	9994	CA-2020-119914	2020-05-04	2020-05-09	Second Class	CC-12220	Chris Cortes	Consumer	United States	Westmir

9994 rows × 21 columns

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

```
11
```

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

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country/Region
2234	2235	CA-2020-104066	12/5/2020	12/10/2020	Standard Class	QJ-19255	Quincy Jones	Corporate	United States
5274	5275	CA-2018-162887	11/7/2018	11/9/2018	Second Class	SV-20785	Stewart Visinsky	Consumer	United States
8798	8799	US-2019-150140	4/6/2019	4/10/2019	Standard Class	VM-21685	Valerie Mitchum	Home Office	United States
9146	9147	US-2019-165505	1/23/2019	1/27/2019	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States
9147	9148	US-2019-165505	1/23/2019	1/27/2019	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States
9148	9149	US-2019-165505	1/23/2019	1/27/2019	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States
9386	9387	US-2020-127292	1/19/2020	1/23/2020	Standard Class	RM-19375	Raymond Messe	Consumer	United States
9387	9388	US-2020-127292	1/19/2020	1/23/2020	Standard Class	RM-19375	Raymond Messe	Consumer	United States
9388	9389	US-2020-127292	1/19/2020	1/23/2020	Standard Class	RM-19375	Raymond Messe	Consumer	United States
9389	9390	US-2020-127292	1/19/2020	1/23/2020	Standard Class	RM-19375	Raymond Messe	Consumer	United States
9741	9742	CA-2018-117086	11/8/2018	11/12/2018	Standard Class	QJ-19255	Quincy Jones	Corporate	United States

11 rows × 21 columns

```
# TODO - Explore this dataset on your owns, ask your own questions
# What city has the most customers in this dataset
df['City'].value_counts()
```

```
New York City      915
Los Angeles        747
Philadelphia       537
San Francisco      510
Seattle            428
                  ...
Glenview             1
Missouri City        1
Rochester Hills      1
Palatine             1
Manhattan            1
```

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, 21)
```

```
# TODO 02 - is there any missing values?, if there is, which colunm? how many
df.isna().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/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
```

```python
# TODO 03 - your friend ask for `California` data, filter it and export csv fo
california = df[df['State'] == 'California']
california.to_csv('california.csv')
```

```python
# TODO 04 - your friend ask for all order data in `California` and `Texas` in
cf_tx_2017 = df[(df['Order Date'].dt.year == 2017) & ((df['State'] == 'Califor
cf_tx_2017.to_csv('cf_tx_2017.csv')
```

```python
# TODO 05 - how much total sales, average sales, and standard deviation of sal
df[df['Order Date'].dt.year == 2017]['Sales'].agg(['sum','mean','std'])
```

```
sum     484247.498100
mean       242.974159
std        754.053357
Name: Sales, dtype: float64
```

```python
# TODO 06 - which Segment has the highest profit in 2018
df[df['Order Date'].dt.year == 2018].groupby('Segment')['Profit'].sum().sort_v
```

```
Segment
Consumer       28460.1665
Corporate      20688.3248
Home Office    12470.1124
Name: Profit, dtype: float64
```

```python
# TODO 07 - which top 5 States have the least total sales between 15 April 201
df[(df['Order Date'] > '2019-04-15') & (df['Order Date'] < '2019-12-31')].grou
    .sort_values().head()
```

```
State
New Hampshire            49.05
New Mexico               64.08
District of Columbia    117.07
Louisiana               249.80
South Carolina          502.48
Name: Sales, dtype: float64
```

```python
# TODO 08 - what is the proportion of total sales (%) in West + Central in 201
df_2019 = df[df['Order Date'].dt.year == 2019]
total_sales = df_2019['Sales'].sum()
sales_w_c = df_2019[(df_2019['Region'] == 'West') | (df_2019['Region'] == 'Cen
sales_w_c / total_sales * 100
```

```
54.97479891837763
```

```python
# TODO 09 - find top 10 popular products in terms of number of orders vs. tota
df_2019_2020 = df[(df['Order Date'].dt.year == 2019) | (df['Order Date'] == 20
df_pd_order = df_2019_2020.value_counts('Product Name').sort_values(ascending=
df_pd_sale = df_2019_2020.groupby('Product Name')['Sales'].sum().sort_values(a
print("Top 10 orders", df_pd_order)
print()
print("Top 10 Sales", df_pd_sale)
```

```
Top 10 orders Product Name
Staple envelope                                                  11
Easy-staple paper                                                11
Staples                                                           9
Chromcraft Round Conference Tables                                9
XtraLife ClearVue Slant-D Ring Binder, White, 3"                  7
Premium Transparent Presentation Covers by GBC                    6
Avery Non-Stick Binders                                           6
Hon Deluxe Fabric Upholstered Stacking Chairs, Rounded Back       6
Acco Perma 4000 Stacking Storage Drawers                          6
Ibico EB-19 Dual Function Manual Binding System                   6
dtype: int64

Top 10 Sales Product Name
Canon imageCLASS 2200 Advanced Copier                        25899.926
GBC Ibimaster 500 Manual ProClick Binding System             12860.562
3D Systems Cube Printer, 2nd Generation, Magenta              9099.930
High Speed Automatic Electric Letter Opener                   8842.662
HP Designjet T520 Inkjet Large Format Printer - 24" Color     8749.950
Okidata MB760 Printer                                         7834.400
```

```
# TODO 10 - plot at least 2 plots, any plot you think interesting :)
# chart 1
df['Region'].value_counts().plot(kind='bar', color=['salmon','orange','yellow'
                                 title='Number of Region',xlabel='Region', yla
# chart 2
df['stack_profit'] = np.cumsum(df['Profit'])
df['stack_profit'].plot(kind='line', y='stack_profit', x='Row ID', figsize = (
                        xlabel='Total of customers', ylabel='Profit')
```

```
<AxesSubplot:title={'center':'Cumulative Sum of Profit'}, xlabel='Total of c
```

```
# TODO Bonus - use np.where() to create new column in dataframe to help you an
df['check_profit'] = np.where(df['Profit'] > 0, 'True', 'False')
profit = df[df['check_profit'] == 'True']['Profit'].sum()
loss = df[df['check_profit'] == 'False']['Profit'].sum()
print(f"Profit: {profit} \nLoss: {abs(loss).round(4)}")
```

```
Profit: 442528.3074
Loss: 156131.2857
```