Final Project - Analyzing Sales Data

Date: 19 May 2023

Author: Chanakan Srichaiwat (Pang-pond)

Course: `Pandas Foundation`

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

```python
# 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	Henderson
1	2	CA-2019-152156	11/8/2019	11/11/2019	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson
2	3	CA-2019-138688	6/12/2019	6/16/2019	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles
3	4	US-2018-108966	10/11/2018	10/18/2018	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale
4	5	US-2018-108966	10/11/2018	10/18/2018	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale

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
trial = pd.to_datetime(df['Order Date'].head(10), format='%m/%d/%Y')
trial.dt.date
```

```
0    2019-11-08
1    2019-11-08
2    2019-06-12
3    2018-10-11
4    2018-10-11
5    2017-06-09
6    2017-06-09
7    2017-06-09
8    2017-06-09
9    2017-06-09
Name: Order Date, dtype: object
```

```
# 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')

df.head(10)
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country/Region	City	...	Po Cc
0	1	CA-2019-152156	2019-11-08	2019-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42
1	2	CA-2019-152156	2019-11-08	2019-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42
2	3	CA-2019-138688	2019-06-12	2019-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90
3	4	US-2018-108966	2018-10-11	2018-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33
4	5	US-2018-108966	2018-10-11	2018-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33
5	6	CA-2017-115812	2017-06-09	2017-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90
6	7	CA-2017-115812	2017-06-09	2017-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90
7	8	CA-2017-115812	2017-06-09	2017-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90
8	9	CA-2017-115812	2017-06-09	2017-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90
9	10	CA-2017-115812	2017-06-09	2017-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90

10 rows × 21 columns

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

```
11
```

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

	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	2020-12-10	Standard Class	QJ-19255	Quincy Jones	Corporate	United States	Burlington	...
5274	5275	CA-2018-162887	2018-11-07	2018-11-09	Second Class	SV-20785	Stewart Visinsky	Consumer	United States	Burlington	...
8798	8799	US-2019-150140	2019-04-06	2019-04-10	Standard Class	VM-21685	Valerie Mitchum	Home Office	United States	Burlington	...
9146	9147	US-2019-165505	2019-01-23	2019-01-27	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States	Burlington	...
9147	9148	US-2019-165505	2019-01-23	2019-01-27	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States	Burlington	...
9148	9149	US-2019-165505	2019-01-23	2019-01-27	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States	Burlington	...
9386	9387	US-2020-127292	2020-01-19	2020-01-23	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burlington	...
9387	9388	US-2020-127292	2020-01-19	2020-01-23	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burlington	...
9388	9389	US-2020-127292	2020-01-19	2020-01-23	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burlington	...
9389	9390	US-2020-127292	2020-01-19	2020-01-23	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burlington	...
9741	9742	CA-2018-117086	2018-11-08	2018-11-12	Standard Class	QJ-19255	Quincy Jones	Corporate	United States	Burlington	...

11 rows × 21 columns

```
# TODO - Explore this dataset on your owns, ask your own questions
# summarise dataframe
df.describe().round(decimals=2)
```

	Row ID	Postal Code	Sales	Quantity	Discount	Profit
count	9994.00	9983.00	9994.00	9994.00	9994.00	9994.00
mean	4997.50	55245.23	229.86	3.79	0.16	28.66
std	2885.16	32038.72	623.25	2.23	0.21	234.26
min	1.00	1040.00	0.44	1.00	0.00	-6599.98
25%	2499.25	23223.00	17.28	2.00	0.00	1.73
50%	4997.50	57103.00	54.49	3.00	0.20	8.67
75%	7495.75	90008.00	209.94	5.00	0.20	29.36
max	9994.00	99301.00	22638.48	14.00	0.80	8399.98

```
# Sales by region
df['Region'].unique() # South, West, Central, East

sales_by_region = df.groupby('Region')['Sales'].agg(['min', 'mean', 'median', 'std'

sales_by_region
```

	min	mean	median	std	max
Region					
Central	0.444	215.772661	45.980	632.779010	17499.950
East	0.852	238.336110	54.900	620.712652	11199.968
South	1.167	241.803645	54.594	774.796273	22638.480
West	0.990	226.493233	60.840	524.876877	13999.960

```
# Number of transactions by state and ship mode
result = df[ ['State', 'Ship Mode'] ].value_counts().reset_index()

result.columns = ['State', 'Ship Mode', 'Count']

result
```

	State	Ship Mode	Count
0	California	Standard Class	1165
1	New York	Standard Class	678
2	Texas	Standard Class	602
3	California	Second Class	395
4	Pennsylvania	Standard Class	371
...	...	...	...
171	Iowa	First Class	1
172	South Dakota	Second Class	1
173	Iowa	Same Day	1
174	Vermont	Second Class	1
175	Wyoming	Standard Class	1

176 rows × 3 columns

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 nan v
df.isna().sum()
df['Postal Code'].isna().sum()
```

```
11
```

```
# TODO 03 - your friend ask for `California` data, filter it and export csv for hi
california_data = df.query(" State == 'California' ")

california_data.to_csv("california_data.csv")
```

```
# TODO 04 - your friend ask for all order data in `California` and `Texas` in 2017
data_2017 = df[df['Order Date'].dt.strftime('%Y')=='2017']
california_texas_2017 = data_2017.query(" State == 'California' or State == 'Texas'

california_texas_2017.to_csv("california_texas_2017.csv")
```

```
# TODO 05 - how much total sales, average sales, and standard deviation of sales yo
print(f"Total Sales: $ {data_2017['Sales'].sum()}")
print(f"Average Sales: $ {data_2017['Sales'].mean()}")
print(f"Standard Deviation: {data_2017['Sales'].std()}")
```

```
Total Sales: $ 484247.4981
Average Sales: $ 242.97415860511794
Standard Deviation: 754.0533572593683
```

```
# TODO 06 - which Segment has the highest profit in 2018
data_2018 = df[df['Order Date'].dt.strftime('%Y')=='2018']
data_2018.sort_values('Profit', ascending=False).head(1)
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country/Region	City	...	Pos Coc
509	510	CA-2018-145352	2018-03-16	2018-03-22	Standard Class	CM-12385	Christopher Martinez	Consumer	United States	Atlanta	...	303

1 rows × 21 columns

```
# TODO 07 - which top 5 States have the least total sales between 15 April 2019 - 3
start_date = '2019-04-15'
end_date = '2019-12-31'

mask = (df['Order Date'] > start_date) & (df['Order Date'] <= end_date)
df2 = df.loc[mask]

df2.groupby('State')['Sales'].sum().sort_values(ascending=True).head(5)
```

```
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
```

```
# TODO 08 - what is the proportion of total sales (%) in West + Central in 2019 e.g
data_2019 = df[df['Order Date'].dt.year == 2019]
west_central_2019_sales = data_2019.query(" Region == 'West' or Region == 'Central'
total_2019_sales = data_2019['Sales'].sum()

print(f"Proportion of total sales: {round(west_central_2019_sales / total_2019_sale
```

```
Proportion of total sales: 54.97 %
```

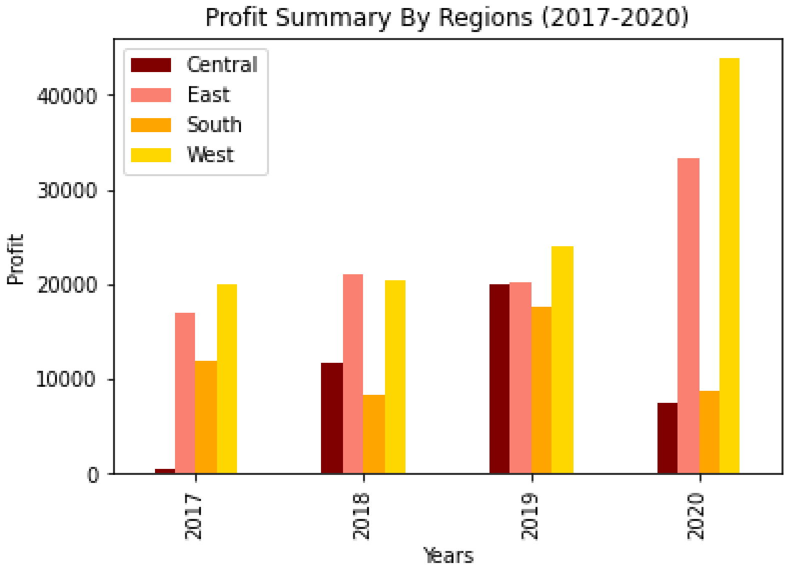
```
# TODO 09 - find top 10 popular products in terms of number of orders vs. total sal
data_2019_2020 = df[(df['Order Date'].dt.year == 2019) | (df['Order Date'].dt.year
df_top_10 = data_2019_2020.groupby('Product Name')[['Quantity', 'Sales']]\
    .sum().reset_index()\
    .sort_values(['Quantity', 'Sales'], ascending=[False, False])\
    .round(decimals=2)\
    .head(10)

df_top_10
```

	Product Name	Quantity	Sales
1412	Staples	124	462.07
512	Easy-staple paper	89	1481.73
1406	Staple envelope	73	644.94
1413	Staples in misc. colors	60	357.16
411	Chromcraft Round Conference Tables	59	7965.05
1421	Storex Dura Pro Binders	49	176.42
1364	Situations Contoured Folding Chairs, 4/Set	47	2612.06
1532	Wilson Jones Clip & Carry Folder Binder Tool f...	44	178.06
250	Avery Non-Stick Binders	43	122.13
662	GBC Premium Transparent Covers with Diagonal L...	42	541.28

```
# TODO 10 - plot at least 2 plots, any plot you think interesting :)
# Plot 1: Grouped bar plot of profit by regions from 2017-2020

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

df_profit_by_region = df.groupby([df['Order Date'].dt.year, 'Region'])['Profit'].su
df_profit_by_region = pd.DataFrame(df_profit_by_region)

plot1 = df_profit_by_region.pivot(index='Order Date', columns='Region', values='Pro
    .plot(kind='bar', stacked=False, color=['Maroon', 'Salmon','Orange', 'Gold'])

plt.xlabel('Years')
plt.ylabel('Profit')
plt.title('Profit Summary By Regions (2017-2020)')
plt.legend(loc='upper left')
```

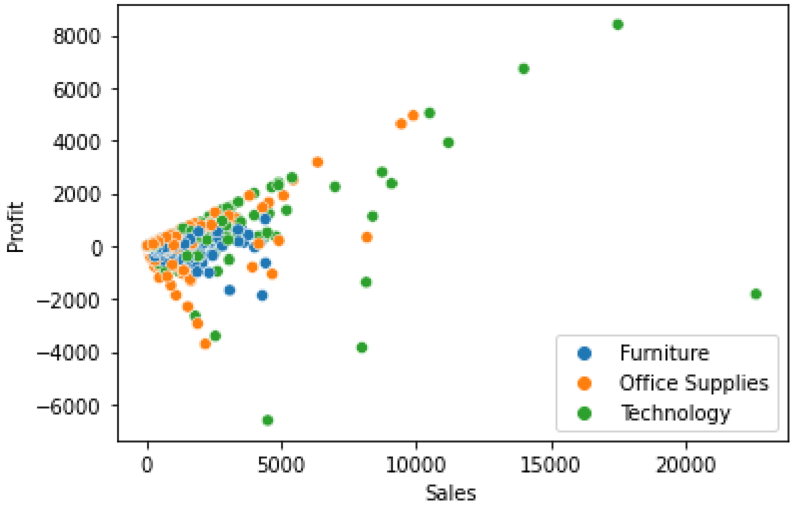
```
<matplotlib.legend.Legend at 0x7fd15c150bb0>
```

Download



```
# Plot 2: Scatter plot of sales vs profit by product category
import matplotlib.pyplot as plt
import seaborn as sns

sns.scatterplot(x='Sales', y='Profit', data=df, hue='Category')
plt.legend(loc='lower right')
plt.show()
```

Download



```
# TODO Bonus - use np.where() to create new column in dataframe to help you answer
# States with average sales that are higher than the mean of total sales
import numpy as np

df_sales_by_state = df.groupby(['Region', 'State'])['Sales'].agg(['sum', 'mean', 'm

df_total_sales = df['Sales'].agg(['sum', 'mean', 'median']).round(decimals=2)
# mean = 229.86

df_sales_by_state['High Spending'] = np.where(df_sales_by_state['mean']>229.86, Tru
df_sales_by_state
```

		sum	mean	median	High Spending
Region	State				
Central	Illinois	80166.10	162.94	36.57	False
	Indiana	53555.36	359.43	70.08	True
	Iowa	4579.76	152.66	33.47	False
	Kansas	2914.31	121.43	63.98	False
	Michigan	76269.61	299.10	85.52	True
	Minnesota	29863.15	335.54	50.40	True
	Missouri	22205.15	336.44	57.68	True
	Nebraska	7464.93	196.45	34.20	False
	North Dakota	919.91	131.42	25.90	False
	Oklahoma	19683.39	298.23	79.55	True
	South Dakota	1315.56	109.63	34.25	False
	Texas	170188.05	172.78	36.29	False
	Wisconsin	32114.61	291.95	93.86	True
East	Connecticut	13384.36	163.22	50.00	False
	Delaware	27451.07	285.95	67.00	True
	District of Columbia	2865.02	286.50	35.80	True
	Maine	1270.53	158.82	105.72	False
	Maryland	23705.52	225.77	89.82	False
	Massachusetts	28634.43	212.11	63.20	False
	New Hampshire	7292.52	270.09	68.62	True
	New Jersey	35764.31	275.11	66.73	True
	New York	310876.27	275.60	60.04	True
	Ohio	78258.14	166.86	44.38	False
	Pennsylvania	116511.91	198.49	41.47	False
	Rhode Island	22627.96	404.07	71.20	True
	Vermont	8929.37	811.76	205.03	True
	West Virginia	1209.82	302.46	265.12	True
South	Alabama	19510.64	319.85	70.98	True
	Arkansas	11678.13	194.64	54.42	False
	Florida	89473.71	233.61	41.47	True
	Georgia	49095.84	266.83	70.96	True
	Kentucky	36591.75	263.25	76.30	True

	Louisiana	9217.03	219.45	64.14	False

```
print(f"There are {df_sales_by_state['High Spending'].sum()} states out of {df_sale
```

23

	South Carolina	8481.71	201.95	69.97	False
	Tennessee	30661.87	167.55	42.05	False
	Virginia	70636.72	315.34	65.25	True
West	Arizona	35282.00	157.51	61.51	False
	California	457687.63	228.73	61.02	False
	Colorado	32108.12	176.42	51.02	False
	Idaho	4382.49	208.69	89.97	False
	Montana	5589.35	372.62	63.98	True
	Nevada	16729.10	428.95	79.14	True
	New Mexico	4783.52	129.28	45.36	False
	Oregon	17431.15	140.57	46.60	False
	Utah	11220.06	211.70	60.12	False
	Washington	138641.27	273.99	65.94	True
	Wyoming	1603.14	1603.14	1603.14	True