Final Project : Analyzing Sales Data

Date : 20 Feb 2023

Author : Kantapon Phasee

Course : Pandas Foundation

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

```
# Preview the first 10 rows
df.head(10)
```

	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	Henderso
1	2	CA-2019-152156	11/8/2019	11/11/2019	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderso
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 Lauderdal
4	5	US-2018-108966	10/11/2018	10/18/2018	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdal
5	6	CA-2017-115812	6/9/2017	6/14/2017	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles
6	7	CA-2017-115812	6/9/2017	6/14/2017	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles
7	8	CA-2017-115812	6/9/2017	6/14/2017	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles
8	9	CA-2017-115812	6/9/2017	6/14/2017	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles
9	10	CA-2017-115812	6/9/2017	6/14/2017	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles

10 rows × 21 columns

```
# shape of dataframe -- check (row, column)
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')
```

```
# 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 Order"] = pd.to_datetime(df["Ship Date"], format="%m/%d/%Y")

df[["Order Date", "Ship Order"]]
```

	Order Date	Ship Order
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
...	...	...
9989	2017-01-21	2017-01-23
9990	2020-02-26	2020-03-03
9991	2020-02-26	2020-03-03
9992	2020-02-26	2020-03-03
9993	2020-05-04	2020-05-09

9994 rows × 2 columns

```
# TODO - count nan in postal code column
# nan is a not a number (missing value)

df["Postal Code"].isna().sum()
```

11

```
# 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	12/10/2020	Standard Class	QJ-19255	Quincy Jones	Corporate	United States	Burlington
5274	5275	CA-2018-162887	2018-11-07	11/9/2018	Second Class	SV-20785	Stewart Visinsky	Consumer	United States	Burlington
8798	8799	US-2019-150140	2019-04-06	4/10/2019	Standard Class	VM-21685	Valerie Mitchum	Home Office	United States	Burlington
9146	9147	US-2019-165505	2019-01-23	1/27/2019	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States	Burlington
9147	9148	US-2019-165505	2019-01-23	1/27/2019	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States	Burlington
9148	9149	US-2019-165505	2019-01-23	1/27/2019	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States	Burlington
9386	9387	US-2020-127292	2020-01-19	1/23/2020	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burlington
9387	9388	US-2020-127292	2020-01-19	1/23/2020	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burlington
9388	9389	US-2020-127292	2020-01-19	1/23/2020	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burlington
9389	9390	US-2020-127292	2020-01-19	1/23/2020	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burlington
9741	9742	CA-2018-117086	2018-11-08	11/12/2018	Standard Class	QJ-19255	Quincy Jones	Corporate	United States	Burlington

11 rows × 22 columns

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

# How many are there in each sub-category ?

df_subcat = df["Sub-Category"].value_counts().reset_index()
df_subcat.columns = ["Sub-Category", "Number of Sub-Category"]
df_subcat
```

	Sub-Category	Number of Sub-Category
0	Binders	1523
1	Paper	1370
2	Furnishings	957
3	Phones	889
4	Storage	846
5	Art	796
6	Accessories	775
7	Chairs	617
8	Appliances	466
9	Labels	364
10	Tables	319
11	Envelopes	254
12	Bookcases	228
13	Fasteners	217
14	Supplies	190
15	Machines	115
16	Copiers	68

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

# Show that the maximum, minimum, mean and median profit in sub-category

stat = df.groupby("Sub-Category")["Profit"].agg(["max", "min", "mean", "median"])
stat
```

	Sub-Category	max	min	mean	median
0	Accessories	829.3754	-75.5958	54.111788	21.00000
1	Appliances	793.7160	-1181.2824	38.922758	17.44650
2	Art	112.5740	0.1533	8.200737	3.72360
3	Binders	4946.3700	-3701.8928	19.843574	3.97710
4	Bookcases	1013.1270	-1665.0522	-15.230509	4.13330
5	Chairs	770.3520	-630.8820	43.095894	13.31760
6	Copiers	8399.9760	59.9980	817.909190	332.99420
7	Envelopes	204.0714	0.5508	27.418019	12.71835
8	Fasteners	21.8880	-11.8256	4.375660	2.84160
9	Furnishings	387.5676	-427.4500	13.645918	9.10200
10	Labels	385.3752	0.6786	15.236962	6.87140
11	Machines	2799.9840	-6599.9780	29.432669	38.99740
12	Paper	352.2960	1.0700	24.856620	11.54320
13	Phones	1228.1787	-386.3916	50.073938	23.52480
14	Storage	792.2691	-337.8060	25.152277	7.75700
15	Supplies	327.5060	-1049.3406	-6.258418	3.95930
16	Tables	629.0100	-1862.3124	-55.565771	-31.37220

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

# It has 9994 columns amd 21 rows
```

```
(9994, 21)
```

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

df.info()

# Yes, it is. Notice the informantion in the non-null count column has 9983 non-n
# Show that postal code has 11 missing values
```

```
<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
```

```
# TODO 03 - your friend ask for `California` data, filter it and export csv for h

df_california = df[ df["State"] == "California" ]

# wirte csv file
df_california.to_csv("mydata1.csv")
```

```
# TODO 04 - your friend ask for all order data in `California` and `Texas` in 201

df1 = df[ (df["State"] == "California") | (df["State"] == "Texas") ]

# Series.dt.year : The year of the datetime
df2 = df1[ df1["Order Date"].dt.year == 2017 ]

df2.to_csv("mydata2.csv")
```

```
# TODO 05 - how much total sales, average sales, and standard deviation of sales

df_y2017= df[ df["Order Date"].dt.year == 2017 ]
result = df_y2017["Sales"].agg(["sum", "mean", "std"]).reset_index()
result.columns = ["statistics value" , "sales"]
result

# total sales = 484248
# average sales = 243
# standard deviation of sales = 754
```

	statistics value	sales
0	sum	484247.498100
1	mean	242.974159
2	std	754.053357

```
# TODO 06 - which Segment has the highest profit in 2018

max_profit_2018 = df[df["Order Date"].dt.year == 2018].groupby("Segment")["Profit
max_profit_2018

# Segment = Consumer
# Profit = 28460
```

	Segment	Profit
0	Consumer	28460.1665
1	Corporate	20688.3248
2	Home Office	12470.1124

```
# TODO 07 - which top 5 States have the least total sales between 15 April 2019 -

df_2019 = df[ (df["Order Date"] >= "2019-04-15") & (df["Order Date"] <= "2019-12-
top5_States = df_2019.groupby("State")["Sales"].sum().reset_index().sort_values("
top5_States
```

	State	Sales
26	New Hampshire	49.05
28	New Mexico	64.08
7	District of Columbia	117.07
16	Louisiana	249.80
36	South Carolina	502.48

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

df_west_central = df[df["Order Date"].dt.year == 2019].query(' Region == "West" |
result_2019 = (df_west_central["Sales"].sum() / df["Sales"].sum())*100
result_2019.__round__(0)

# the proportion of total sales (%) in West + Central in 2019 is 15 %
```

```
15.0
```

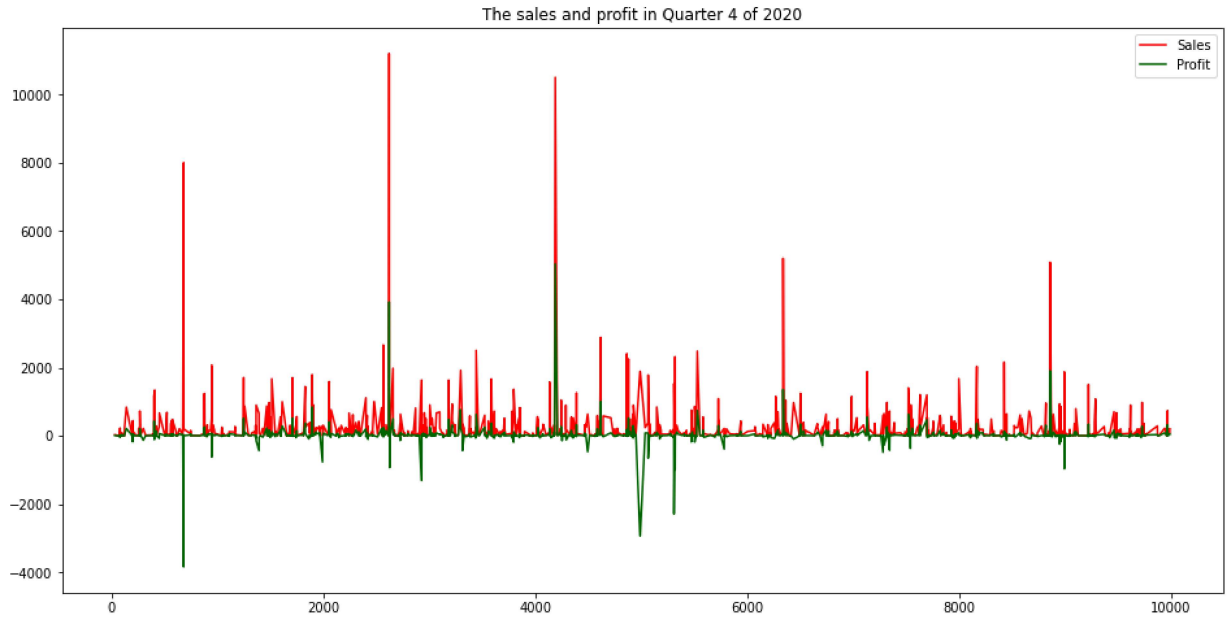
```
# TODO 09 - find top 10 popular products in terms of number of orders vs. total s

total_year = df[ (df["Order Date"] >= "01-01-2019") & (df["Order Date"] <= "12-31

top10_orders = total_year.groupby("Product Name")["Quantity"].agg(["sum"]).sort_v
head(10).reset_index()
top10_orders.columns = ["Product Name", "Number of Orders"]
top10_orders


top10_sales = total_year.groupby("Product Name")["Sales"].agg(["sum"]).sort_value
head(10).reset_index()
top10_sales.columns = ["Product Name", "Total Sales"]
top10_sales

print(top10_orders)
print("")
print(top10_sales)
```

```
                                 Product Name  Number of Orders
0                                      Staples               124
1                          Easy-staple paper                89
2                            Staple envelope                73
3                  Staples in misc. colors                60
4          Chromcraft Round Conference Tables                59
```

```
5                          Storex Dura Pro Binders                49
6         Situations Contoured Folding Chairs, 4/Set              47
7  Wilson Jones Clip & Carry Folder Binder Tool f...              44
8                          Avery Non-Stick Binders                43
9                       Eldon Wave Desk Accessories               42

                                      Product Name  Total Sales
0             Canon imageCLASS 2200 Advanced Copier    61599.824
1              Hewlett Packard LaserJet 3310 Copier    16079.732
2   3D Systems Cube Printer, 2nd Generation, Magenta    14299.890
3   GBC Ibimaster 500 Manual ProClick Binding System    13621.542
4         GBC DocuBind TL300 Electric Binding System    12737.258
5          GBC DocuBind P400 Electric Binding System    12521.108
```

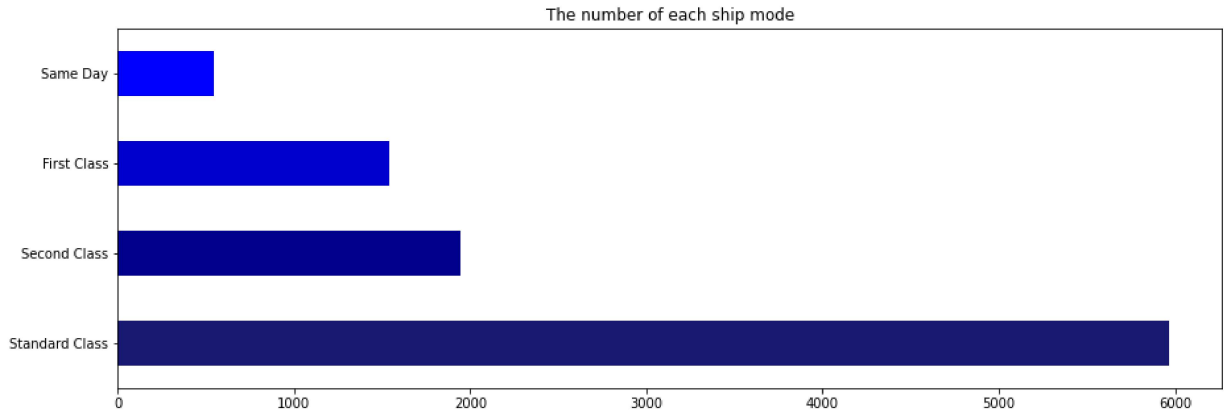
```
# TODO 10 - plot at least 2 plots, any plot you think interesting :)

# Plot the sales and profit in Quarter 4 of 2020
# Q4 is October to December

q4 = df[ (df["Order Date"] >= "10-01-2020") & (df["Order Date"] <= "31-12-2020")
q4[["Sales", "Profit"]].plot(kind = "line", color= ["red", "darkgreen"],figsize=[
 title="The sales and profit in Quarter 4 of 2020");
```

Download



```
# The number of each ship mode

df["Ship Mode"].value_counts().plot.barh( color = ["midnightblue", "darkblue", "m
 figsize=[15,5], title="The number of each ship mode");
```

Download




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

import numpy as np

# find the city that make profit more than 6000 ?

df_city = df.groupby("City")["Profit"].sum().reset_index()
top_profit = df_city.sort_values("Profit", ascending = False)
top_profit

top_profit["Profit More Than 6000"] = np.where(top_profit["Profit"] > 6000, True,
top_profit

# True is profit more than 6000 and False is profit less than 6000
# It has 11 city
```

	City	Profit	Profit More Than 6000
329	New York City	62036.9837	True
266	Los Angeles	30440.7579	True
452	Seattle	29156.0967	True
438	San Francisco	17507.3854	True
123	Detroit	13181.7908	True
...	...	...	...
80	Chicago	-6654.5688	False
241	Lancaster	-7239.0684	False
434	San Antonio	-7299.0502	False
207	Houston	-10153.5485	False
374	Philadelphia	-13837.7674	False

531 rows × 3 columns