Final Project - Analyzing Sales Data

Date: 1 February 2022

Author: Palakorn Kerdsap (Boss DataRockie)

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	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 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
```

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')
df.head()
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country/Region	City	...	P C
0	1	CA-2019-152156	2019-11-08	2019-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	4
1	2	CA-2019-152156	2019-11-08	2019-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	4
2	3	CA-2019-138688	2019-06-12	2019-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	9
3	4	US-2018-108966	2018-10-11	2018-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	3
4	5	US-2018-108966	2018-10-11	2018-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	3

5 rows × 21 columns

```
# TODO - convert order date and ship date to datetime in the original dataframe
df.reset_index()
```

	index	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country/Region	...
0	0	1	CA-2019-152156	2019-11-08	2019-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	...
1	1	2	CA-2019-152156	2019-11-08	2019-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	...
2	2	3	CA-2019-138688	2019-06-12	2019-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	...
3	3	4	US-2018-108966	2018-10-11	2018-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	...
4	4	5	US-2018-108966	2018-10-11	2018-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	...
...	...	...	...	...	...	...	...	...	...	...	...
9989	9989	9990	CA-2017-110422	2017-01-21	2017-01-23	Second Class	TB-21400	Tom Boeckenhauer	Consumer	United States	...
9990	9990	9991	CA-2020-121258	2020-02-26	2020-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	...
9991	9991	9992	CA-2020-121258	2020-02-26	2020-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	...
9992	9992	9993	CA-2020-121258	2020-02-26	2020-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	...
9993	9993	9994	CA-2020-119914	2020-05-04	2020-05-09	Second Class	CC-12220	Chris Cortes	Consumer	United States	...

9994 rows × 22 columns

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

```
11
```

```python
# TODO - filter rows with missing values
df[df.isna().any(axis=1)]
```

	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
df['year'] = pd.DatetimeIndex(df['Order Date']).year
# past sales per year
df.groupby('year') ['Sales'].sum()
```

```
year
2017    484247.4981
2018    470532.5090
2019    609205.5980
2020    733215.2552
Name: Sales, dtype: float64
```

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
resultjoi01 = df.shape

print("Answer :")
print(resultjoi01)

print(f" Total Dataset, there are {df.shape[0]} rows and {df.shape[1]} columns.")
```

```
Answer :
(9994, 22)
 Total Dataset, there are 9994 rows and 22 columns.
```

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

resultjoi02 = df.isna().sum()
print("Answer :")
print("Yes, there are missing values . All in Postal Code colunm  11 nan values."


print(resultjoi02)

resultjoi02[resultjoi02 > 0]
```

```
Answer :
Yes, there are missing values . All in Postal Code colunm  11 nan values.
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


Postal Code    11
dtype: int64
```

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

result_california.to_csv('California.csv')
```

```python
# TODO 04 - your friend ask for all order data in `California` and `Texas` in 201
df[((df['State']=='California') | (df['State']=='Texas')) \
   & (df['Order Date'].dt.year==2017)]\
    .to_csv('California_Texas_2017.csv')
```

```python
# TODO 05 - how much total sales, average sales, and standard deviation of sales
print("Answer:")

df['Order Year'] = df['Order Date'].dt.year
resultjoi05 = df[df['Order Year']==2017].groupby('Order Year')['Sales'].agg(['sum
print(resultjoi05)
```

```
Answer:
                    sum        mean         std
Order Year
2017        484247.4981  242.974159  754.053357
```

```
# TODO 06 - which Segment has the highest profit in 2018
# set df
print("Answer:")
resultjoi06 = df[df['Order Year']==2018].groupby(['Order Year','Segment'])['Profi
    .sort_values('sum', ascending = False).reset_index().head(1)



print(f"In 2018,{resultjoi06.iloc[0,1]} segment has the highest profit with total
resultjoi06
```

```
Answer:
In 2018,Consumer segment has the highest profit with total profit 28460.1665
```

	Order Year	Segment	sum
0	2018	Consumer	28460.1665

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

resultjoi07 = df[(df['Order Date']>='2019-04-15') & (df['Order Date']<='2019-12-3
    .groupby('State')['Sales']\
    .sum()\
    .sort_values()\
    .head(5)\
    .round(2)
print("Answer :")
print(resultjoi07)
```

```
Answer :
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 2019 e

resultjoiWC = df[df['Order Date'].dt.year == 2019]\
    .query("Region == 'West' | Region == 'Central'")['Sales']\
    .sum()
result08_2019 = df[df['Order Date'].dt.year == 2019]['Sales'].sum()
result08_percentage_sales = 100 * resultjoiWC/result08_2019

print("Answer :")
print(f"{(result08_percentage_sales).round(2)}%")
```

```
Answer :
54.97%
```

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


resulty19_20= df[(df['Order Date'].dt.year >= 2019) & (df['Order Date'].dt.year <

resultjoi09_by_orders = resulty19_20.value_counts('Product Name')\
    .sort_values(ascending=False).head(10).reset_index()

print("Answer :")
print("Top 10 Product by Orders VS. Sales")

resultjoi09_by_orders.columns = ['Product by Orders', 'Count of Orders']
resultjoi09_by_sales = resulty19_20\
    .groupby('Product Name')[['Product Name', 'Sales']]\
    .agg('sum')\
    .sort_values(by='Sales', ascending=False)\
    .head(10)\
    .round(2)\
    .reset_index()

resultjoi09_by_sales.columns = ['Product by Sales', 'Total Sales']
resultjoi09_by_orders_vs_sales = pd.concat([resultjoi09_by_orders,resultjoi09_by_
resultjoi09_by_orders_vs_sales
```

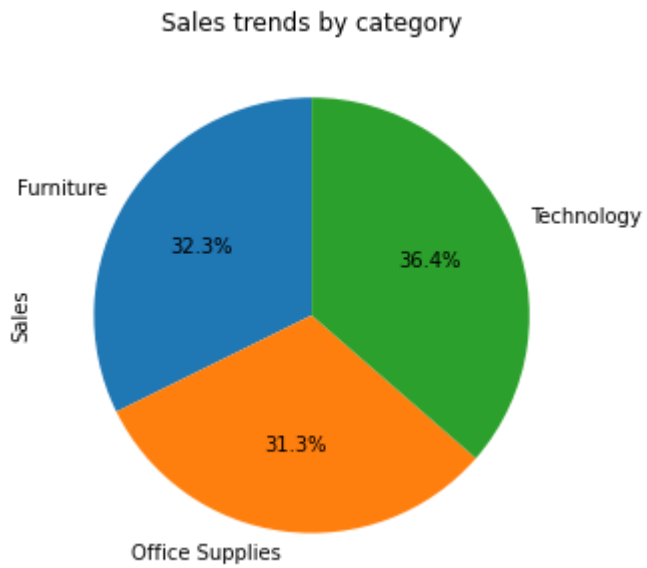
```
Answer :
Top 10 Product by Orders VS. Sales
```

	Product by Orders	Count of Orders	Product by Sales	Total Sales
0	Easy-staple paper	27	Canon imageCLASS 2200 Advanced Copier	61599.82
1	Staples	24	Hewlett Packard LaserJet 3310 Copier	16079.73
2	Staple envelope	22	3D Systems Cube Printer, 2nd Generation, Magenta	14299.89
3	Staples in misc. colors	13	GBC Ibimaster 500 Manual ProClick Binding System	13621.54
4	Staple remover	12	GBC DocuBind TL300 Electric Binding System	12737.26
5	Storex Dura Pro Binders	12	GBC DocuBind P400 Electric Binding System	12521.11
6	Chromcraft Round Conference Tables	12	Samsung Galaxy Mega 6.3	12263.71
7	Global Wood Trimmed Manager's Task Chair, Khaki	11	HON 5400 Series Task Chairs for Big and Tall	11846.56
8	Avery Non-Stick Binders	11	Martin Yale Chadless Opener Electric Letter Op...	11825.90
9	Staple-based wall hangings	10	Global Troy Executive Leather Low-Back Tilter	10169.89

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

import matplotlib as mpl
import matplotlib.pyplot as plt

print("Answer:")
print( " Sales trends by category plot ")

df.groupby('Category')['Sales'].sum()\
.plot.pie(y='Category',  figsize=(5,5), autopct='%1.1f%%', startangle=90)\
.set_title('Sales trends by category')
plt.show()
```

```
Answer:
 Sales trends by category plot
```

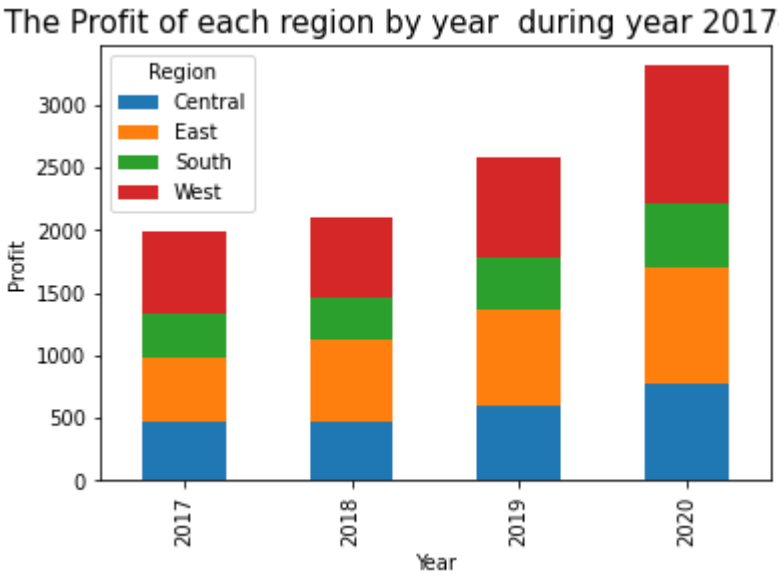
Download



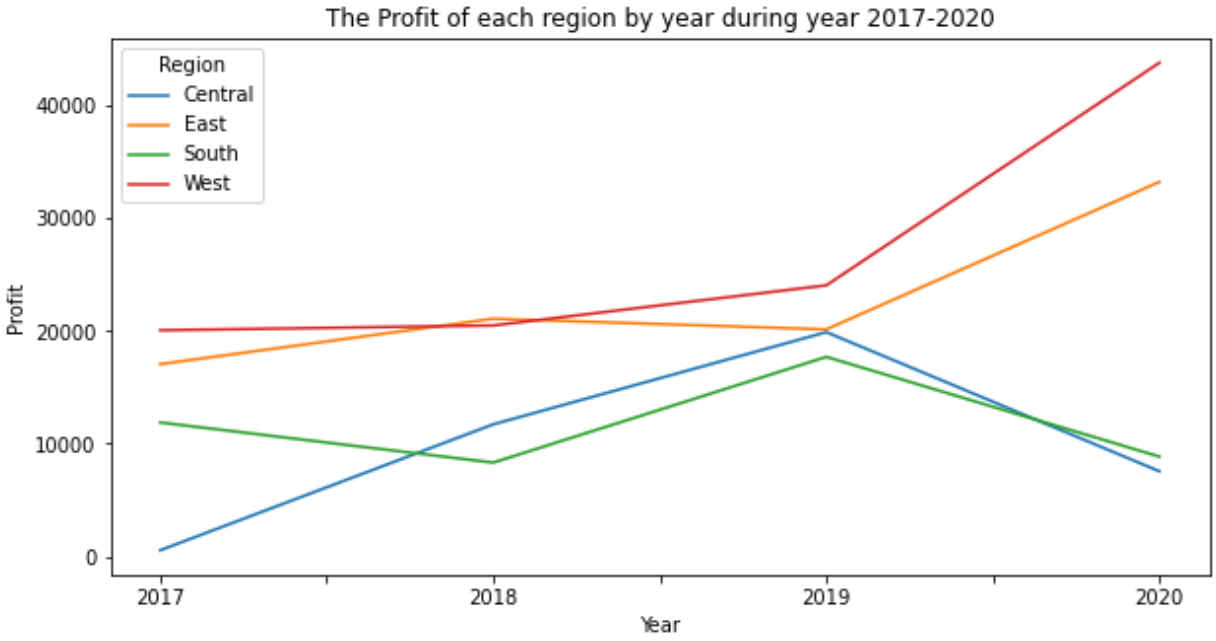

```
print("Answer:")

print(" 2 Plots I think interesting")


df['Year'] = df['Order Date'].dt.strftime('%Y')
table =pd.DataFrame( df.groupby(['Year','Region'])['Profit'].sum().reset_index())
df.groupby(['Year','Region']).size().unstack().plot(kind='bar', stacked=True)
plt.xlabel("Year")
plt.ylabel("Profit")
plt.title('The Profit of each region by year  during year 2017-2020', size = 15)
plt.show()


df['Year'] = df['Order Date'].dt.strftime('%Y')
profit_by_region = df.groupby(['Year', 'Region'])['Profit']\
    .agg('sum')\
    .reset_index()
profit_by_region.pivot(columns='Region', index='Year', values='Profit')\
    .plot(kind='line', xlabel='Year', ylabel='Profit',
    title='The Profit of each region by year during year 2017-2020',figsize=(10,5
```

```
Answer:
 This is Plot I think interesting
```

Download


The Profit of each region by year during year 2017-2020
Region
Central
East
South
West
Profit
3000
2500
2000
1500
1000
500
0
2017
2018
2019
2020
Year

Download


The Profit of each region by year during year 2017-2020
Region
Central
East
South
West
Profit
40000
30000
20000
10000
0
2017
2018
2019
2020
Year

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


import numpy as np
print("Question Bonus - What are the top loss product by most number of orders of
print("Answer:")
print("  It can be seen from below table that top loss product by most number of


bonus_filter = (df['Order Date']>=pd.to_datetime('2020-10-01')) & df['Sub-Categor
df99 = df[bonus_filter][['Product Name','Profit']]
df99['OrderLoss'] = np.where(df99['Profit']<0,1,0)
df99 = df99.groupby('Product Name')['OrderLoss'].sum().reset_index().sort_values(
df99.head(10)
```

```
Question Bonus - What are the top loss product by most number of orders of Mach
Answer:
  It can be seen from below table that top loss product by most number of order
```

	index	Product Name	OrderLoss
0	11	Avery Reinforcements for Hole-Punch Pages	2
1	5	Aluminum Screw Posts	2
2	21	Cardinal EasyOpen D-Ring Binders	1
3	24	Computer Printout Index Tabs	1
4	25	Cubify CubeX 3D Printer Triple Head Print	1
5	26	DYMO CardScan Personal V9 Business Card Scanner	1
6	27	Economy Binders	1
7	28	Epson TM-T88V Direct Thermal Printer - Monochr...	1
8	55	Prestige Round Ring Binders	1
9	54	Premium Transparent Presentation Covers, No Pa...	1

```python
# TODO Bonus - use np.where() to create new column in dataframe to help you answe
# What are the states that mostly sold products at a loss?
df['Profitable'] = np.where(df['Profit'] >= 0, 'Break-even or profit', 'Loss')


count_df = pd.crosstab(df['State'], df['Profitable'])
count_df.div(count_df.sum(axis=1), axis=0) \
    .query('Loss > 0') \
    .sort_values('Loss', ascending=False) \
    .plot.bar(stacked=True)

plt.legend(loc='lower right', ncol=2)
```

```
<matplotlib.legend.Legend at 0x7f75c8616220>
```

Download