Final Project - Analyzing Sales Data

Date: 26 November 2022

Author: Aekkarach Artpru

Course: `Data Science BootCamp`

```
# 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	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.

```
# 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[['Order Date', 'Ship Date']]
```

	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
...	...	...
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

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

11

```
# 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
```

```
# what segment is a top profit ?

df.groupby('Segment')['Profit'].agg(sum).round(2).reset_index()
```

	Segment	Profit
0	Consumer	134119.21
1	Corporate	91979.13
2	Home Office	60298.68

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

print('columns =',df.shape[0])
print('rows =',df.shape[1])
```

```
columns = 9994
rows = 21
```

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

df.info()
# if check via info method. result = missing values in column 'Postal Code' an
```

```
<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   datetime64[ns]
 3   Ship Date       9994 non-null   datetime64[ns]
 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 fo

df['State'] == 'California' # filter column state in 'California'
df[df['State'] == 'California'] # convert boolean to datafram (2001 rows)
df[df['State'] == 'California'].head(15) # check data
df[df['State'] == 'California'].tail(15) # check data

df_ca = df[df['State'] == 'California'] # assign values to create table 'df_ca
df_ca # check data (2001 rows)

df_ca.to_csv('df_ca.csv')  # use to_csv method for write csv

# if one line coded for write csv : df[df['State'] == 'California'].to_csv('df
```

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

df.query("State == 'California' | State == 'Texas'") # query method to select
df_ca_tx = df.query("State == 'California' | State == 'Texas'") # assign value
df_ca_tx # check data (2986 rows)

df_ca_tx['Order Date'].dt.year == 2017 # dt.year method to convert and then se

df_ca_tx[df_ca_tx['Order Date'].dt.year == 2017].to_csv('df_ca_tx_2017.csv') #
```

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

df[df['Order Date'].dt.year == 2017] # select sales in 2017
df_sales_2017 = df[df['Order Date'].dt.year == 2017] # assign values to create

df_sales_2017['Sales'].agg(['sum', 'mean', 'std']).reset_index() # .agg method
# print result
print(f'Total Sales =', df_sales_2017['Sales'].sum().round(2),\
      '\nAverage Sales =', df_sales_2017['Sales'].mean().round(2),\
      '\nStandard Diviation =', df_sales_2017['Sales'].std().round(2))
```

```
Total Sales = 484247.5
Average Sales = 242.97
Standard Diviation = 754.05
```

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

df_2018 = df[df['Order Date'].dt.year == 2018] # convert, select year 2018, an
df_2018.groupby('Segment').sum('Profit')['Profit'].sort_values(ascending=False

# answer = Consumer Segment
```

```
# TODO 07 - which top 5 States have the least total sales between 15 April 201

df[df['Order Date'].between('2019-04-15', '2019-12-31')] # filter date via bet
apr_dec_2019 = df[df['Order Date'].between('2019-04-15', '2019-12-31')] # crea

apr_dec_2019.groupby('State').sum('Sales')['Sales'].sort_values().head(5) # so
```

```
# TODO 08 - what is the proportion of total sales (%) in West + Central in 201

df_2019 = df[df['Order Date'].dt.year == 2019] # filter 2019 (2587 rows)
df_w_c_2019 = df_2019.query('Region == "West" | Region == "Central"') # west 8
df_w_c_2019['Sales'].sum() # West + Central in 2019 total sales = 334909.5525
df['Sales'].sum() # total sales = 2297200.8603000003

(df_w_c_2019['Sales'].sum() / df['Sales'].sum()).round(2) * 100 # percentage o
(df_w_c_2019['Sales'].sum() / df_2019['Sales'].sum()).round(2) * 100 # percent
```

```
55.00000000000001
```

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

df_19_20 = df[ (df['Order Date'].dt.year == 2019) | (df['Order Date'].dt.year


topten_pro = df_19_20.groupby('Product Name')['Sales'].count().sort_values(asc
```

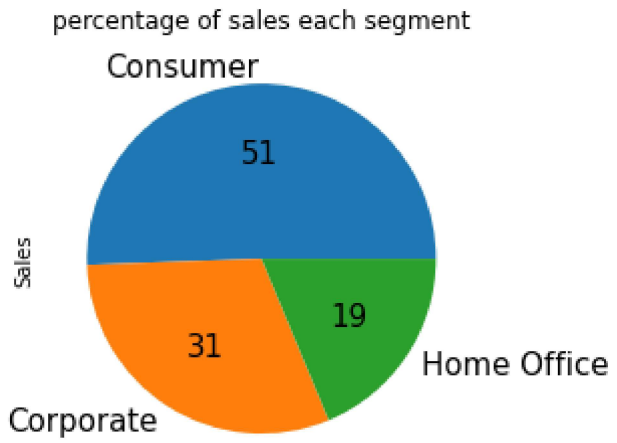
```
topten_sales = df_19_20.groupby('Product Name')['Sales'].sum().sort_values(asc

print(f'Top 10 Products of the year 2019-2000\n', topten_pro)
print(f'Top 10 Sales of the year 2019-2000\n', topten_sales)
```

```
Top 10 Products of the year 2019-2000
 Product Name
Easy-staple paper                                    27
Staples                                              24
Staple envelope                                      22
Staples in misc. colors                              13
Staple remover                                       12
Storex Dura Pro Binders                              12
Chromcraft Round Conference Tables                   12
Global Wood Trimmed Manager's Task Chair, Khaki      11
Avery Non-Stick Binders                              11
Staple-based wall hangings                           10
Name: Sales, dtype: int64
Top 10 Sales of the year 2019-2000
 Product Name
Canon imageCLASS 2200 Advanced Copier                61599.82
Hewlett Packard LaserJet 3310 Copier                 16079.73
3D Systems Cube Printer, 2nd Generation, Magenta     14299.89
GBC Ibimaster 500 Manual ProClick Binding System     13621.54
GBC DocuBind TL300 Electric Binding System           12737.26
```

```
# TODO 10 - plot at least 2 plots, any plot you think interesting :)
# what a percentage of sales each segment ?
df.groupby("Segment")["Sales"].sum().plot(kind = 'pie', title = 'percentage of
                                          , autopct = '%.f');
```

Download




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

```
# how many transaction get profit better than average profit ?
import numpy as np
np.min(df['Profit']) # min = -6599.978
np.max(df['Profit']) # min = 8399.976
np.mean(df['Profit']) # average = 28.65689630778467
np.median(df['Profit']) # median = 8.6665

np.where(df['Profit'] > 28, True, False) # np.where method for query condition
df['ProfitBetterAverage'] = np.where(df['Profit'] > 28, True, False)
profit_better = df.value_counts('ProfitBetterAverage')

profit_better # 2590 transaction get profit more than average profit
```