Refrence

Answer

End answer

Worksheet 3

This worksheet is due Monday night of Week 3. You are encouraged to work in groups of up to 3 total students, but each student should submit their own file. (It's fine for everyone in the group to upload the same file.)

These questions refer to the attached vending machines csv file, `vend.csv`.

- Load the file as a pandas DataFrame using `pd.read_csv` and store it as the variable `df`. (You will need to import `pandas` first.)

In [1]:
```
import pandas as pd
import numpy as np

df = pd.read_csv("../Data/vend.csv")
df.head()
```

Out[1]:

	Status	Device ID	Location	Machine	Product	Category	Transaction	TransDa
0	Processed	VJ300320611	Brunswick Sq Mall	BSQ Mall x1366 - ATT	Red Bull - Energy Drink - Sugar Free	Carbonated	14515778905	Saturd January 20
1	Processed	VJ300320611	Brunswick Sq Mall	BSQ Mall x1366 - ATT	Red Bull - Energy Drink - Sugar Free	Carbonated	14516018629	Saturd January 20
2	Processed	VJ300320611	Brunswick Sq Mall	BSQ Mall x1366 - ATT	Takis - Hot Chilli Pepper & Lime	Food	14516018629	Saturd January 20
3	Processed	VJ300320611	Brunswick Sq Mall	BSQ Mall x1366 - ATT	Takis - Hot Chilli Pepper & Lime	Food	14516020373	Saturd January 20
4	Processed	VJ300320611	Brunswick Sq Mall	BSQ Mall x1366 - ATT	Red Bull - Energy Drink - Sugar Free	Carbonated	14516021756	Saturd January 20

- How many rows are there in this DataFrame? How many columns? Use the `shape` attribute. (When we refer to something as an attribute, it usually means we will not be using parentheses with it. Methods are like functions and attributes are like variables. Both methods and attributes are attached to an object and are accessed using a period `.`.)

In [2]:
```
print(df.shape)
```

```
(6445, 18)
```

Answer

6445 Rows, 18 Columns

End answer

- Using the `dtypes` attribute of this DataFrame, check how the data type of the "Location" column is represented. Among all the columns, what different data types are listed?

In [3]:
```
print(f"Location dtype: {df['Location'].dtype}")
print(np.unique(df.dtypes))
```

```
Location dtype: object
[dtype('int64') dtype('float64') dtype('O')]
```

Answer

Location is a series, and the columns contain integers, floats, and objects

End answer

- Access the row at integer location `2420` using `iloc` and square brackets. Store this in the variable `x`.

In [4]:
```
x = df.iloc[2420]
print(x)
```

```
Status                              Processed
Device ID                         VJ300320686
Location                        Earle Asphalt
Machine                   Earle Asphalt x1371
Product        Belvita Snack Packs - Blueberry
Category                                  Food
Transaction                     15041961028
TransDate               Friday, April 29, 2022
Type                                    Credit
RCoil                                      122
RPrice                                    1.25
RQty                                         1
MCoil                                      122
MPrice                                    1.25
MQty                                         1
LineTotal                                 1.25
TransTotal                                1.25
Prcd Date                             4/29/2022
Name: 2420, dtype: object
```

- Using the Python built-in function `type`, what is the data type of `x`?

In [5]:
```
print(type(x))
```

```
<class 'pandas.core.series.Series'>
```

- What is the value of `x.loc["Location"]`? Is there any difference if you use `x["Location"]`? What about `x("Location")`?

In [6]:
```
print(x.loc["Location"])
print(x["Location"])
# print(x("Location"))
```

```
Earle Asphalt
Earle Asphalt
```

Answer

There is no difference between using `x.loc["Location"]` and `x["Location"]`. `x("Location")` however, leads to an error

End answer

- What is the `type` of `x.loc["Location"]`? (Notice how this type was not directly reported to us by pandas when we used the `dtypes` attribute. When something is reported as having "object" as its dtype, I usually assume it is a string, but it could also be something else, like a list.)

In [7]:
```
print(f"The type is a: {type(x.loc['Location'])}")
```

```
The type is a: <class 'str'>
```

- Using Boolean indexing, define `df_sub` to be the sub-DataFrame containing all the transactions from this same location.

In [8]:
```
df_sub = df[df["Location"] == "Earle Asphalt"]
print(df_sub)
```

	Status	Device ID	Location	Machine	\
7	Processed	VJ300320686	Earle Asphalt	Earle Asphalt x1371	
9	Processed	VJ300320686	Earle Asphalt	Earle Asphalt x1371	
13	Processed	VJ300320686	Earle Asphalt	Earle Asphalt x1371	
15	Processed	VJ300320686	Earle Asphalt	Earle Asphalt x1371	
16	Processed	VJ300320686	Earle Asphalt	Earle Asphalt x1371	
...	...	...	...	...	
6435	Processed	VJ300320686	Earle Asphalt	Earle Asphalt x1371	
6436	Processed	VJ300320686	Earle Asphalt	Earle Asphalt x1371	
6437	Processed	VJ300320686	Earle Asphalt	Earle Asphalt x1371	
6438	Processed	VJ300320686	Earle Asphalt	Earle Asphalt x1371	
6439	Processed	VJ300320686	Earle Asphalt	Earle Asphalt x1371	

	Product	Category	Transaction	\
7	Miss Vickie's Potato Chip - Lime & Cracked Pe	Food	14519162059	
9	Seapoint Farms Dry Roasted Edamame - Wasabi	Food	14520315330	
13	KitKat - Crisp Wafers	Food	14520549978	
15	Wonderful Pistachios - Variety	Food	14520852487	
16	Wonderful Pistachios - Variety	Food	14521022594	
...	...	...	...	
6435	Lindens - Chocolate Chippers	Food	15601642404	
6436	SunChips Multigrain - Salsa	Food	15602233765	
6437	Keto Bar - Creamy Peanut Butter Chocolate	Food	15602399908	
6438	Wonderful Pistachios - Variety	Food	15602645264	
6439	Robert Irvine's - Fit Crunch - Chocolate Pea	Food	15602818773	

	TransDate	Type	RCoil	RPrice	RQty	MCoil	MPrice	\
7	Monday, January 3, 2022	Credit	110	1.50	1	110	1.50	
9	Monday, January 3, 2022	Credit	134	2.50	1	134	2.50	
13	Monday, January 3, 2022	Credit	136	1.75	1	136	1.75	
15	Monday, January 3, 2022	Cash	132	2.00	1	132	2.00	
16	Monday, January 3, 2022	Cash	132	2.00	1	132	2.00	
...	...	...	...	...	...	...	...	
6435	Wednesday, August 31, 2022	Credit	121	2.25	1	121	2.25	
6436	Wednesday, August 31, 2022	Cash	110	1.50	1	110	1.50	
6437	Wednesday, August 31, 2022	Cash	131	2.00	1	131	2.00	
6438	Wednesday, August 31, 2022	Cash	138	2.00	1	138	2.00	
6439	Wednesday, August 31, 2022	Cash	132	2.00	1	132	2.00	

	MQty	LineTotal	TransTotal	Prcd Date
7	1	1.50	1.50	1/3/2022
9	1	2.50	2.50	1/3/2022
13	1	1.75	1.75	1/3/2022
15	1	2.00	2.00	1/3/2022
16	1	2.00	2.00	1/3/2022
...	...	...	...	...
6435	1	2.25	2.25	8/31/2022
6436	1	1.50	1.50	8/31/2022
6437	1	2.00	2.00	8/31/2022
6438	1	2.00	2.00	8/31/2022
6439	1	2.00	2.00	8/31/2022

[676 rows x 18 columns]

- How many rows in the original DataFrame correspond to this location? Set the variable `a` to be equal to this integer. (Check. It should be between 600 and 700.)

In [9]:
```
a = df_sub.shape[0]
print(f"Number of rows = {a}")
```

```
Number of rows = 676
```

- What values of `b` and `c` are such that `df_sub.loc[13, "Transaction"]` is equal to `df_sub.iloc[b,c]` ? (Remember that counting in Python starts at 0. I don't intend you to have a computer code way of finding these values. Just look at `df_sub` and check.) Store these values.

In [10]:
```
print(df_sub.loc[13, "Transaction"])
b = 2
c = 6
print(df_sub.iloc[b, c])
```

```
14520549978
14520549978
```

- There was exactly one transaction in `df_sub` where the "RPrice" was `1.5` and where "RQty" was `2` (meaning two items were sold in the same transaction). What was the name of that product (i.e., the value in the "Product" column? Store that string with the variable `d` . (Be sure your answer is exact, including spacing and capitalization.)

In [11]:
```
d = df_sub["Product"][(df_sub["RPrice"] == 1.5) & (df_sub["RQty"] == 2)].iloc[0]
print(d)
```

```
Oreo Mini
```

- There is exactly one row in `df` where the `"RPrice"` is not equal to the `"MPrice"` . What is the index of that row? Set `e` to be equal to that index. (The index is the number that's displayed all the way on the left. You can access the index by using the `index` attribute. To check whether two elements are **not** equal, you can use `!=` . Another option is to check for equality and then to negate it using tilde `~` .)

In [12]:
```
e = df[df["RPrice"] != df["MPrice"]].index
print(e)
```

```
Index([5500], dtype='int64')
```

- Put these five values (four integers and one string) into a tuple, `my_tuple = (a,b,c,d,e)` .
- Save `my_tuple` in a pickle file named `"wkst3-ans.pickle"` using the following code. Submit that file on Canvas as your submission for Worksheet 3.

```
import pickle

with open("wkst3-ans.pickle", 'wb') as f:
    pickle.dump(my_tuple, f)
```

- If you want to double-check that this `"wkst3-ans.pickle"` pickle file really contains your answer, you can run the following code. If you then evaluate or print `x`, you should see your original `my_tuple` values. (If you're in a new notebook, you also need to import the pickle module again.)

```
with open("wkst3-ans.pickle", 'rb') as f:
    x = pickle.load(f)
```