Pandas Exercises

Time to test your new pandas skills! Use the two csv files in this folder to complete the tasks in bold below!

** Import pandas and read in the banklist.csv file into a dataframe called banks. **

In [1]:
```
import pandas as pd
banks=pd.read_csv("banklist.csv")
print(banks)
```

```
                                             Bank Name                City  \
0                                   Fayette County Bank          Saint Elmo
1    Guaranty Bank, (d/b/a BestBank in Georgia & Mi...           Milwaukee
2                                        First NBC Bank         New Orleans
3                                         Proficio Bank  Cottonwood Heights
4                         Seaway Bank and Trust Company             Chicago
..                                                  ...                 ...
546                                  Superior Bank, FSB            Hinsdale
547                                 Malta National Bank               Malta
548                     First Alliance Bank & Trust Co.          Manchester
549                   National State Bank of Metropolis          Metropolis
550                                    Bank of Honolulu            Honolulu

     ST   CERT                Acquiring Institution Closing Date Updated Date
0    IL   1802             United Fidelity Bank, fsb    26-May-17     1-Jun-17
1    WI  30003  First-Citizens Bank & Trust Company      5-May-17     1-Jun-17
2    LA  58302                         Whitney Bank    28-Apr-17    23-May-17
3    UT  35495                    Cache Valley Bank     3-Mar-17    18-May-17
4    IL  19328                  State Bank of Texas    27-Jan-17    18-May-17
..   ..    ...                                  ...          ...          ...
546  IL  32646                Superior Federal, FSB    27-Jul-01    19-Aug-14
547  OH   6629                    North Valley Bank     3-May-01    18-Nov-02
548  NH  34264  Southern New Hampshire Bank & Trust     2-Feb-01    18-Feb-03
549  IL   3815             Banterra Bank of Marion    14-Dec-00    17-Mar-05
550  HI  21029                    Bank of the Orient    13-Oct-00    17-Mar-05

[551 rows x 7 columns]
```

In [25]:

** Show the head of the dataframe **

In [4]:
```
head=banks.head()
print(head)
```

```
                                          Bank Name                City  ST  \
0                               Fayette County Bank          Saint Elmo  IL   
1  Guaranty Bank, (d/b/a BestBank in Georgia & Mi...           Milwaukee  WI   
2                                    First NBC Bank         New Orleans  LA   
3                                     Proficio Bank  Cottonwood Heights  UT   
4                     Seaway Bank and Trust Company             Chicago  IL   

    CERT                Acquiring Institution Closing Date Updated Date  
0   1802           United Fidelity Bank, fsb    26-May-17     1-Jun-17  
1  30003  First-Citizens Bank & Trust Company     5-May-17     1-Jun-17  
2  58302                        Whitney Bank    28-Apr-17    23-May-17  
3  35495                    Cache Valley Bank     3-Mar-17    18-May-17  
4  19328                  State Bank of Texas    27-Jan-17    18-May-17  
```

In [37]:

Out[37]:

	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date	Updated Date
0	Fayette County Bank	Saint Elmo	IL	1802	United Fidelity Bank, fsb	26-May-17	1-Jun-17
1	Guaranty Bank, (d/b/a BestBank in Georgia & Mi...	Milwaukee	WI	30003	First-Citizens Bank & Trust Company	5-May-17	1-Jun-17
2	First NBC Bank	New Orleans	LA	58302	Whitney Bank	28-Apr-17	23-May-17
3	Proficio Bank	Cottonwood Heights	UT	35495	Cache Valley Bank	3-Mar-17	18-May-17
4	Seaway Bank and Trust Company	Chicago	IL	19328	State Bank of Texas	27-Jan-17	18-May-17

** What are the column names? **

In [5]:
```
print(banks.columns)
```

```
Index(['Bank Name', 'City', 'ST', 'CERT', 'Acquiring Institution',
       'Closing Date', 'Updated Date'],
      dtype='object')
```

In [29]:
```

```

Out[29]:
```
Index(['Bank Name', 'City', 'ST', 'CERT', 'Acquiring Institution',
       'Closing Date', 'Updated Date'],
      dtype='object')
```

** How many States (ST) are represented in this data set? **

In [6]:
```
banks['ST'].nunique()
```

Out[6]:
```
44
```

In [33]:
```

```

Out[33]:
```
44
```

** Get a list or array of all the states in the data set. **

In [7]:
```
banks['ST'].unique()
```

Out[7]:
```
array(['IL', 'WI', 'LA', 'UT', 'NJ', 'AR', 'GA', 'PA', 'TN', 'WA', 'CO',
       'PR', 'FL', 'MN', 'CA', 'MD', 'OK', 'OH', 'SC', 'VA', 'ID', 'TX',
       'CT', 'AZ', 'NV', 'NC', 'KY', 'MO', 'KS', 'AL', 'MI', 'IN', 'IA',
       'NE', 'MS', 'NM', 'OR', 'NY', 'MA', 'SD', 'WY', 'WV', 'NH', 'HI'],
      dtype=object)
```

In [32]:
```

```

Out[32]:
```
array(['IL', 'WI', 'LA', 'UT', 'NJ', 'AR', 'GA', 'PA', 'TN', 'WA', 'CO',
       'PR', 'FL', 'MN', 'CA', 'MD', 'OK', 'OH', 'SC', 'VA', 'ID', 'TX',
       'CT', 'AZ', 'NV', 'NC', 'KY', 'MO', 'KS', 'AL', 'MI', 'IN', 'IA',
       'NE', 'MS', 'NM', 'OR', 'NY', 'MA', 'SD', 'WY', 'WV', 'NH', 'HI'], dtype=object)
```

** What are the top 5 states with the most failed banks? **

In [8]:
```
banks.groupby("ST").count().sort_values('Bank Name',ascending=False).iloc[:5]['Bank Name']
```

Out[8]:
```
ST
GA    93
FL    75
IL    67
CA    41
MN    23
Name: Bank Name, dtype: int64
```

In [35]:

Out[35]:
```
ST
GA    93
FL    75
IL    67
CA    41
MN    23
Name: Bank Name, dtype: int64
```

** What are the top 5 acquiring institutions? **

In [9]:
```
banks['Acquiring Institution'].value_counts().iloc[:5]
```

Out[9]:
```
No Acquirer                           31
State Bank and Trust Company          12
First-Citizens Bank & Trust Company   11
Ameris Bank                           10
U.S. Bank N.A.                         9
Name: Acquiring Institution, dtype: int64
```

In [14]:

Out[14]:
```
No Acquirer                            31
State Bank and Trust Company           12
First-Citizens Bank & Trust Company    11
Ameris Bank                            10
U.S. Bank N.A.                          9
Name: Acquiring Institution, dtype: int64
```

** How many banks has the State Bank of Texas acquired? How many of them were actually in Texas?**

In [10]:
```
banks[banks['Acquiring Institution']=='State Bank of Texas']
```

Out[10]:

	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date	Updated Date
4	Seaway Bank and Trust Company	Chicago	IL	19328	State Bank of Texas	27-Jan-17	18-May-17
21	The National Republic Bank of Chicago	Chicago	IL	916	State Bank of Texas	24-Oct-14	6-Jan-16
450	Millennium State Bank of Texas	Dallas	TX	57667	State Bank of Texas	2-Jul-09	26-Oct-12

In [15]:

Out[15]:

	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date	Updated Date
4	Seaway Bank and Trust Company	Chicago	IL	19328	State Bank of Texas	27-Jan-17	18-May-17
21	The National Republic Bank of Chicago	Chicago	IL	916	State Bank of Texas	24-Oct-14	6-Jan-16
450	Millennium State Bank of Texas	Dallas	TX	57667	State Bank of Texas	2-Jul-09	26-Oct-12

** What is the most common city in California for a bank to fail in?**

In [11]:
```
banks[banks['ST']=='CA'].groupby('City').count().sort_values('Bank Name',ascending=False).head(1)
```

Out[11]:

City	Bank Name	ST	CERT	Acquiring Institution	Closing Date	Updated Date
Los Angeles	4	4	4	4	4	4

In [24]:

Out[24]:

City	Bank Name	ST	CERT	Acquiring Institution	Closing Date	Updated Date
Los Angeles	4	4	4	4	4	4

** How many failed banks don't have the word "Bank" in their name? **

In [12]:
```
sum(banks['Bank Name'].apply(lambda name: 'Bank' not in name))
```

Out[12]:
```
14
```

In [55]:

Out[55]:
```
14
```

** How many bank names start with the letter 's' ? **

In [13]:
```
sum(banks['Bank Name'].apply(lambda name:name[0].upper() =='S'))
```

Out[13]:
```
53
```

In [58]:

Out[58]:
```
53
```

** How many CERT values are above 20000 ? **

```
In [14]: sum(banks['CERT']>20000)

Out[14]: 417

In [64]:

Out[64]: 417
```

** How many bank names consist of just two words? (e.g. "First Bank" , "Bank Georgia")**

```
In [15]: sum(banks['Bank Name'].apply(lambda name: len(name.split())==2))

Out[15]: 114

In [67]:

Out[67]: 114
```

Bonus: How many banks closed in the year 2008? (this is hard because we technically haven't learned about time series with pandas yet! Feel free to skip this one!

```
In [16]: sum(banks['Closing Date'].apply(lambda date: date[-2:]) == '08')

Out[16]: 25

In [54]:

Out[54]: 25
```

GREAT JOB!

Student Alcohol Consumption

Introduction:

This time you will download a dataset from the UCI.

Step 1. Import the necessary libraries

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

Step 2. Import the dataset from student-alcohal.csv

Step 3. Assign it to a variable called df.

In [14]:
```
csv_url = './students-alcohal.csv'
df = pd.read_csv(csv_url)
df.head()
```

Out[14]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absen
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	

5 rows × 33 columns

Step 4. For the purpose of this exercise slice the dataframe from 'school' until the 'guardian' column

In [3]:
```
stud_alcoh = df.loc[: , "school":"guardian"]
stud_alcoh.head()
```

Out[3]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	course	mother
1	GP	F	17	U	GT3	T	1	1	at_home	other	course	father
2	GP	F	15	U	LE3	T	1	1	at_home	other	other	mother
3	GP	F	15	U	GT3	T	4	2	health	services	home	mother
4	GP	F	16	U	GT3	T	3	3	other	other	home	father

Step 5. Create a lambda function that will capitalize strings.

In [4]:
```
capitalizer = lambda x: x.capitalize()
```

Step 6. Capitalize both Mjob and Fjob

In [5]:
```
stud_alcoh['Mjob'].apply(capitalizer)
stud_alcoh['Fjob'].apply(capitalizer)
```

Out[5]:
```
0       Teacher
1         Other
2         Other
3      Services
4         Other
         ...
390    Services
391    Services
392       Other
393       Other
394     At_home
Name: Fjob, Length: 395, dtype: object
```

Step 7. Print the last elements of the data set.

In [6]:
```
stud_alcoh.tail()
```

Out[6]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian
390	MS	M	20	U	LE3	A	2	2	services	services	course	other
391	MS	M	17	U	LE3	T	3	1	services	services	course	mother
392	MS	M	21	R	GT3	T	1	1	other	other	course	other
393	MS	M	18	R	LE3	T	3	2	services	other	course	mother
394	MS	M	19	U	LE3	T	1	1	other	at_home	course	father

Step 8. Did you notice the original dataframe is still lowercase? Why is that? Fix it and capitalize Mjob and Fjob.

In [7]:
```
stud_alcoh['Mjob'] = stud_alcoh['Mjob'].apply(capitalizer)
stud_alcoh['Fjob'] = stud_alcoh['Fjob'].apply(capitalizer)
stud_alcoh.tail()
```

Out[7]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian
390	MS	M	20	U	LE3	A	2	2	Services	Services	course	other
391	MS	M	17	U	LE3	T	3	1	Services	Services	course	mother
392	MS	M	21	R	GT3	T	1	1	Other	Other	course	other
393	MS	M	18	R	LE3	T	3	2	Services	Other	course	mother
394	MS	M	19	U	LE3	T	1	1	Other	At_home	course	father

Step 9. Create a function called majority that returns a boolean value to a new column called legal_drinker (Consider majority as older than 17 years old)

In [9]:
```
def majority(x):
    if x > 17:
        return True
    else:
        return False
stud_alcoh['legal_drinker'] = stud_alcoh['age'].apply(majority)
stud_alcoh.head()
```

Out[9]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian	legal_drinker
0	GP	F	18	U	GT3	A	4	4	At_home	Teacher	course	mother	True
1	GP	F	17	U	GT3	T	1	1	At_home	Other	course	father	False
2	GP	F	15	U	LE3	T	1	1	At_home	Other	other	mother	False
3	GP	F	15	U	GT3	T	4	2	Health	Services	home	mother	False
4	GP	F	16	U	GT3	T	3	3	Other	Other	home	father	False

In [10]:
```
def times10(x):
    if type(x) is int:
        return 10 * x
    return x
stud_alcoh.applymap(times10).head(10)
```

Out[10]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian	legal_drinker
0	GP	F	180	U	GT3	A	40	40	At_home	Teacher	course	mother	True
1	GP	F	170	U	GT3	T	10	10	At_home	Other	course	father	False
2	GP	F	150	U	LE3	T	10	10	At_home	Other	other	mother	False
3	GP	F	150	U	GT3	T	40	20	Health	Services	home	mother	False
4	GP	F	160	U	GT3	T	30	30	Other	Other	home	father	False
5	GP	M	160	U	LE3	T	40	30	Services	Other	reputation	mother	False
6	GP	M	160	U	LE3	T	20	20	Other	Other	home	mother	False
7	GP	F	170	U	GT3	A	40	40	Other	Teacher	home	mother	False
8	GP	M	150	U	LE3	A	30	20	Services	Other	home	mother	False
9	GP	M	150	U	GT3	T	30	40	Other	Other	home	mother	False

Step 10. Multiply every number of the dataset by 10.

I know this makes no sense, don't forget it is just an exercise

In [11]:
```
def times10(x):
    if type(x) is int:
        return 10 * x
    return x
stud_alcoh.applymap(times10).head(10)
```

Out[11]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian	legal_drinker
0	GP	F	180	U	GT3	A	40	40	At_home	Teacher	course	mother	True
1	GP	F	170	U	GT3	T	10	10	At_home	Other	course	father	False
2	GP	F	150	U	LE3	T	10	10	At_home	Other	other	mother	False
3	GP	F	150	U	GT3	T	40	20	Health	Services	home	mother	False
4	GP	F	160	U	GT3	T	30	30	Other	Other	home	father	False
5	GP	M	160	U	LE3	T	40	30	Services	Other	reputation	mother	False
6	GP	M	160	U	LE3	T	20	20	Other	Other	home	mother	False
7	GP	F	170	U	GT3	A	40	40	Other	Teacher	home	mother	False
8	GP	M	150	U	LE3	A	30	20	Services	Other	home	mother	False
9	GP	M	150	U	GT3	T	30	40	Other	Other	home	mother	False

In [ ]:

Filtering and Sorting Data

Step 1. Import the necessary libraries

In [72]:
```
import pandas as pd
```

Step 2. Import the dataset from chipotle.tsv

Step 3. Assign it to a variable called chipo.

In [73]:
```
chipo = pd.read_csv('chipotle.tsv', sep="\t")
#print(chipo)
```

Step 4. How many products cost more than $10.00?

In [74]:
```
cost = [float(value[1 : ]) for value in chipo.item_price]
chipo.item_price = cost
chipo[chipo['item_price']>10.00].head(10)
```

Out[74]:

	order_id	quantity	item_name	choice_description	item_price
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	16.98
5	3	1	Chicken Bowl	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	10.98
7	4	1	Steak Burrito	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	11.75
13	7	1	Chicken Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	11.25
23	12	1	Chicken Burrito	[[Tomatillo-Green Chili Salsa (Medium), Tomati...	10.98
39	19	1	Barbacoa Bowl	[Roasted Chili Corn Salsa, [Fajita Vegetables,...	11.75
42	20	1	Chicken Bowl	[Roasted Chili Corn Salsa, [Rice, Black Beans,...	11.25
43	20	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Pinto Beans, Chees...	11.75
45	21	1	Chicken Burrito	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	10.98
52	24	1	Chicken Burrito	[Roasted Chili Corn Salsa (Medium), [Black Bea...	10.98

Step 5. What is the price of each item?

print a data frame with only two columns item_name and item_price

In [83]:
```
chip1= chipo.drop_duplicates(['item_name','quantity'])
chip2 = chip1[chip1.quantity == 1]
chip2.sort_values(by = "item_price", ascending = False).head(40)
```

Out[83]:

	order_id	quantity	item_name	choice_description	item_price
606	250	1	Steak Salad Bowl	[Fresh Tomato Salsa, [Pinto Beans, Cheese, Gua...	11.89
1229	501	1	Barbacoa Salad Bowl	[Fresh Tomato Salsa, [Rice, Fajita Vegetables,...	11.89
1132	468	1	Carnitas Salad Bowl	[Fresh Tomato Salsa, [Rice, Black Beans, Chees...	11.89
7	4	1	Steak Burrito	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	11.75
168	75	1	Barbacoa Crispy Tacos	[Tomatillo Red Chili Salsa, [Rice, Black Beans...	11.75
39	19	1	Barbacoa Bowl	[Roasted Chili Corn Salsa, [Fajita Vegetables,...	11.75
738	304	1	Veggie Soft Tacos	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	11.25
186	83	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	11.25
62	28	1	Veggie Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	11.25
57	26	1	Veggie Burrito	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	11.25
250	109	1	Chicken Salad	[Roasted Chili Corn Salsa (Medium), [Black Bea...	10.98
5	3	1	Chicken Bowl	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	10.98
8	4	1	Steak Soft Tacos	[Tomatillo Green Chili Salsa, [Pinto Beans, Ch...	9.25
554	230	1	Carnitas Crispy Tacos	[Roasted Chili Corn Salsa]	9.25
237	103	1	Carnitas Soft Tacos	[Tomatillo Green Chili Salsa, [Fajita Vegetabl...	9.25
56	26	1	Barbacoa Soft Tacos	[Fresh Tomato Salsa, [Fajita Vegetables, Black...	9.25
92	40	1	Steak Crispy Tacos	[Fresh Tomato Salsa, Sour Cream]	9.25
664	276	1	Steak Salad	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	8.99
54	25	1	Steak Bowl	[Fresh Tomato Salsa (Mild), [Black Beans, Rice...	8.99
3750	1500	1	Carnitas Salad	[[Fresh Tomato Salsa (Mild), Roasted Chili Cor...	8.99
21	11	1	Barbacoa Burrito	[[Fresh Tomato Salsa (Mild), Tomatillo-Green C...	8.99
27	14	1	Carnitas Burrito	[[Tomatillo-Green Chili Salsa (Medium), Roaste...	8.99
33	17	1	Carnitas Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	8.99
11	6	1	Chicken Crispy Tacos	[Roasted Chili Corn Salsa, [Fajita Vegetables,...	8.75
12	6	1	Chicken Soft Tacos	[Roasted Chili Corn Salsa, [Rice, Black Beans,...	8.75
44	20	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	8.75
1653	668	1	Veggie Crispy Tacos	[Fresh Tomato Salsa (Mild), [Pinto Beans, Rice...	8.49
16	8	1	Chicken Burrito	[Tomatillo-Green Chili Salsa (Medium), [Pinto ...	8.49
1694	686	1	Veggie Salad	[[Fresh Tomato Salsa (Mild), Roasted Chili Cor...	8.49
1414	575	1	Salad	[Brown Rice, Adobo-Marinated and Grilled Chick...	7.40
510	214	1	Burrito	[Adobo-Marinated and Grilled Chicken, Pinto Be...	7.40
520	217	1	Crispy Tacos	[Adobo-Marinated and Grilled Steak]	7.40
673	279	1	Bowl	[Adobo-Marinated and Grilled Steak, [Sour Crea...	7.40
298	129	1	6 Pack Soft Drink	[Sprite]	6.49
10	5	1	Chips and Guacamole	NaN	4.45
1	1	1	Izze	[Clementine]	3.39
2	1	1	Nantucket Nectar	[Apple]	3.39
674	279	1	Chips and Mild Fresh Tomato Salsa	NaN	3.00
111	49	1	Chips and Tomatillo Red Chili Salsa	NaN	2.95
233	102	1	Chips and Roasted Chili Corn Salsa	NaN	2.95

Step 6. Sort by the name of the item

```
In [81]: #chipo['item_name'].sort_values()
         chip1= chipo.drop_duplicates(['item_name', 'choice_description'])
         chip1.sort_values(by=["item_name"])
```

Out[81]:

	order_id	quantity	item_name	choice_description	item_price
341	148	1	6 Pack Soft Drink	[Diet Coke]	6.49
298	129	1	6 Pack Soft Drink	[Sprite]	6.49
357	154	1	6 Pack Soft Drink	[Coke]	6.49
721	298	1	6 Pack Soft Drink	[Nestea]	6.49
3141	1253	1	6 Pack Soft Drink	[Lemonade]	6.49
127	56	1	Barbacoa Bowl	[Tomatillo Red Chili Salsa, [Rice, Pinto Beans...	9.25
1264	514	1	Barbacoa Bowl	[Tomatillo Red Chili Salsa, [Rice, Black Beans...	9.25
3376	1356	1	Barbacoa Bowl	[Fresh Tomato Salsa, [Rice, Black Beans, Cheese]]	9.25
3017	1200	1	Barbacoa Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	11.75
2073	836	1	Barbacoa Bowl	[Tomatillo Green Chili Salsa, [Fajita Vegetabl...	11.75
4046	1619	1	Barbacoa Bowl	[Tomatillo Green Chili Salsa, [Fajita Vegetabl...	11.75
2013	812	1	Barbacoa Bowl	[Tomatillo Red Chili Salsa, [Black Beans, Chee...	11.75
2620	1041	1	Barbacoa Bowl	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	9.25
4056	1624	1	Barbacoa Bowl	[[Rice, Cheese]]	8.69
95	42	1	Barbacoa Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	9.25
219	97	1	Barbacoa Bowl	[Tomatillo Red Chili Salsa, [Rice, Black Beans...	9.25
919	380	1	Barbacoa Bowl	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	11.75
750	310	1	Barbacoa Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	8.99
1746	705	1	Barbacoa Bowl	[Tomatillo Red Chili Salsa, [Cheese, Sour Crea...	11.75
1347	550	1	Barbacoa Bowl	[Fresh Tomato Salsa, [Rice, Black Beans, Chees...	9.25
1357	554	1	Barbacoa Bowl	[Fresh Tomato Salsa, [Rice, Black Beans, Pinto...	9.25
3025	1203	1	Barbacoa Bowl	[Roasted Chili Corn Salsa, [Fajita Vegetables,...	11.75
1419	576	1	Barbacoa Bowl	[Roasted Chili Corn Salsa]	9.25
3547	1426	1	Barbacoa Bowl	[Fresh Tomato Salsa, [Rice, Cheese, Sour Cream...	11.75
3549	1426	1	Barbacoa Bowl	[Fresh Tomato Salsa, [Rice, Black Beans, Chees...	9.25
1383	562	1	Barbacoa Bowl	[[Tomatillo-Green Chili Salsa (Medium), Roaste...	11.48
2762	1097	1	Barbacoa Bowl	[Tomatillo Red Chili Salsa, [Rice, Fajita Vege...	11.75
2517	1000	1	Barbacoa Bowl	[Roasted Chili Corn Salsa, [Rice, Black Beans,...	9.25
804	331	1	Barbacoa Bowl	[Tomatillo Red Chili Salsa, [Rice, Fajita Vege...	9.25
1804	730	1	Barbacoa Bowl	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	8.99
...	...	...	...	...	...
891	368	1	Veggie Burrito	[Fresh Tomato Salsa (Mild), [Black Beans, Rice...	8.49
2067	834	1	Veggie Burrito	[Fresh Tomato Salsa, [Cheese, Rice, Pinto Beans]]	8.75
1653	668	1	Veggie Crispy Tacos	[Fresh Tomato Salsa (Mild), [Pinto Beans, Rice...	8.49
2756	1094	1	Veggie Salad	[[Tomatillo-Green Chili Salsa (Medium), Roaste...	8.49
2996	1192	1	Veggie Salad	[Roasted Chili Corn Salsa (Medium), [Black Bea...	8.49
3163	1263	1	Veggie Salad	[[Fresh Tomato Salsa (Mild), Roasted Chili Cor...	8.49
1694	686	1	Veggie Salad	[[Fresh Tomato Salsa (Mild), Roasted Chili Cor...	8.49
2683	1066	1	Veggie Salad Bowl	[Roasted Chili Corn Salsa, [Fajita Vegetables,...	8.75
4201	1677	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Black...	11.25
186	83	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	11.25
960	394	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	8.75
3293	1321	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Rice, Black Beans, Chees...	8.75
4573	1818	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	8.75
455	195	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	11.25
2269	913	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	8.75
4109	1646	1	Veggie Salad Bowl	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	11.25
2156	869	1	Veggie Salad Bowl	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	11.25
1316	536	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	8.75
496	207	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Rice, Lettuce, Guacamole...	11.25
4541	1805	1	Veggie Salad Bowl	[Tomatillo Green Chili Salsa, [Fajita Vegetabl...	8.75
2223	896	1	Veggie Salad Bowl	[Roasted Chili Corn Salsa, Fajita Vegetables]	8.75
4261	1700	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	11.25
295	128	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	11.25
1699	688	1	Veggie Soft Tacos	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	11.25
2851	1132	1	Veggie Soft Tacos	[Roasted Chili Corn Salsa (Medium), [Black Bea...	8.49

	order_id	quantity	item_name	choice_description	item_price
2384	948	1	Veggie Soft Tacos	[Roasted Chili Corn Salsa, [Fajita Vegetables,...	8.75
738	304	1	Veggie Soft Tacos	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	11.25
3889	1559	2	Veggie Soft Tacos	[Fresh Tomato Salsa (Mild), [Black Beans, Rice...	16.98
1395	567	1	Veggie Soft Tacos	[Fresh Tomato Salsa (Mild), [Pinto Beans, Rice...	8.49
781	322	1	Veggie Soft Tacos	[Fresh Tomato Salsa, [Black Beans, Cheese, Sou...	8.75

1871 rows × 5 columns

Step 7. What was the quantity of the most expensive item ordered?

In [26]:
```
chipo.sort_values(by=["quantity"], ascending = False).head(1)
```

Out[26]:

	order_id	quantity	item_name	choice_description	item_price
3598	1443	15	Chips and Fresh Tomato Salsa	NaN	$44.25

Step 8. How many times was a Veggie Salad Bowl ordered?

In [18]:
```
veg_salad = chipo[chipo['item_name'] == "Veggie Salad Bowl"]
len(veg_salad)
```

Out[18]:
```
18
```

Step 9. How many times did someone order more than one Canned Soda?

In [17]:
```
canned_soda = chipo[(chipo.item_name == "Canned Soda") & (chipo.quantity > 1)]
len(canned_soda)
```

Out[17]:
```
20
```

Housing Market

Introduction:

This time we will create our own dataset with fictional numbers to describe a house market. As we are going to create random data don't try to reason of the numbers.

Step 1. Import the necessary libraries

In [1]:

```
import pandas as pd
import numpy as np
```

Step 2. Create 3 differents Series, each of length 100, as follows:

1. The first a random number from 1 to 4
2. The second a random number from 1 to 3
3. The third a random number from 10,000 to 30,000

In [2]:

```
s1 = pd.Series(np.random.randint(1, high=5, size=100, dtype='l'))
s2 = pd.Series(np.random.randint(1, high=4, size=100, dtype='l'))
s3 = pd.Series(np.random.randint(10000, high=30001, size=100, dtype='l'))

print(s1, s2, s3)
```

```
0     1
1     4
2     3
3     1
4     3
     ..
95    3
96    4
97    2
98    3
99    3
Length: 100, dtype: int32 0     1
1     3
2     1
3     1
4     2
     ..
95    1
96    3
97    1
98    1
99    3
Length: 100, dtype: int32 0     10253
1     11390
2     22106
3     16382
4     18029
      ...
95    28474
96    27641
97    17363
98    26937
99    13776
Length: 100, dtype: int32
```

Step 3. Let's create a DataFrame by joinning the Series by column

In [3]:

```
housemkt = pd.concat([s1, s2, s3], axis=1)
housemkt.head()
```

Out[3]:

	0	1	2
0	1	1	10253
1	4	3	11390
2	3	1	22106
3	1	1	16382
4	3	2	18029

Step 4. Change the name of the columns to bedrs, bathrs, price_sqr_meter

In [4]:

```
housemkt.rename(columns = {0: 'bedrs', 1: 'bathrs', 2: 'price_sqr_meter'}, inplace=T
housemkt.head()
```

Out[4]:

	bedrs	bathrs	price_sqr_meter
0	1	1	10253
1	4	3	11390
2	3	1	22106
3	1	1	16382
4	3	2	18029

Step 5. Create a one column DataFrame with the values of the 3 Series and assign it to 'bigcolumn'

In [6]:

```
bigcolumn = pd.concat([s1, s2, s3], axis=0)
bigcolumn = bigcolumn.to_frame()
print(type(bigcolumn))
bigcolumn
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[6]:

	0
0	1
1	4
2	3
3	1
4	3
...	...
95	28474
96	27641
97	17363
98	26937
99	13776

300 rows × 1 columns

Step 6. Oops, it seems it is going only until index 99. Is it true?

In [7]:

```
len(bigcolumn)
```

Out[7]:

```
300
```

Step 7. Reindex the DataFrame so it goes from 0 to 299

In [8]:

```
bigcolumn.reset_index(drop=True, inplace=True)
bigcolumn
```

Out[8]:

```
         0
  0      1
  1      4
  2      3
  3      1
  4      3
...    ...
295  28474
296  27641
297  17363
298  26937
299  13776
```

300 rows × 1 columns

Getting and Knowing your Data

Step 1. Import the necessary libraries

In [1]:

```
import pandas as pd
import numpy as np
```

Step 2. Import the dataset from this chipotle.tsv

Step 3. Assign it to a variable called chipo.

In [27]:

```
url = './chipotle.tsv'
chipo = pd.read_csv(url, sep = '\t')
```

Step 4. See the first 10 entries

In [3]:

```
chipo.head(10)
```

Out[3]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98
5	3	1	Chicken Bowl	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	$10.98
6	3	1	Side of Chips	NaN	$1.69
7	4	1	Steak Burrito	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	$11.75
8	4	1	Steak Soft Tacos	[Tomatillo Green Chili Salsa, [Pinto Beans, Ch...	$9.25
9	5	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Pinto...	$9.25

Step 5. What is the number of observations in the dataset?

In [4]:

```
# Solution 1
chipo.shape[0]  # entries <= 4622 observations

```

Out[4]:

```
4622
```

In [5]:

```
# Solution 2

chipo.info() # entries <= 4622 observations
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4622 entries, 0 to 4621
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   order_id            4622 non-null   int64 
 1   quantity            4622 non-null   int64 
 2   item_name           4622 non-null   object
 3   choice_description  3376 non-null   object
 4   item_price          4622 non-null   object
dtypes: int64(2), object(3)
memory usage: 180.7+ KB
```

Step 6. What is the number of columns in the dataset?

In [6]:

```
chipo.shape[1]
```

Out[6]:

```
5
```

Step 7. Print the name of all the columns.

In [7]:

```
chipo.columns
```

Out[7]:

```
Index(['order_id', 'quantity', 'item_name', 'choice_description',
       'item_price'],
      dtype='object')
```

Step 8. How is the dataset indexed?

In [8]:

```
chipo.index
```

Out[8]:

```
RangeIndex(start=0, stop=4622, step=1)
```

Step 9. Which was the most-ordered item?

In [9]:

```
c = chipo.groupby('item_name')
c = c.sum()
c = c.sort_values(['quantity'], ascending=False)
c.head(1)
```

Out[9]:

	order_id	quantity
item_name		
Chicken Bowl	713926	761

Step 10. For the most-ordered item, how many items were ordered?

In [10]:

```
c = chipo.groupby('item_name')
c = c.sum()
c = c.sort_values(['quantity'], ascending=False)
c.head(1)
```

Out[10]:

	order_id	quantity
item_name		
Chicken Bowl	713926	761

Step 11. What was the most ordered item in the choice_description column?

In [12]:

```
c = chipo.groupby('choice_description').sum()
c = c.sort_values(['quantity'], ascending=False)
c.head(1)

```

Out[12]:

	order_id	quantity
choice_description		
[Diet Coke]	123455	159

Step 12. How many items were orderd in total?

In [13]:

```
total_items_orders = chipo.quantity.sum()
total_items_orders
```

Out[13]:

```
4972
```

Step 13. Turn the item price into a float

Step 13.a. Check the item price type

In [14]:

```
chipo.item_price.dtype
```

Out[14]:

```
dtype('O')
```

Step 13.b. Create a lambda function and change the type of item price

In [15]:

```
dollarizer = lambda x: float(x[1:-1])
chipo.item_price = chipo.item_price.apply(dollarizer)
```

Step 13.c. Check the item price type

In [16]:

```
chipo.item_price.dtype
```

Out[16]:

```
dtype('float64')
```

Step 14. How much was the revenue for the period in the dataset?

In [17]:

```
revenue = (chipo['quantity']* chipo['item_price']).sum()

print('Revenue was: $' + str(np.round(revenue,2)))
```

```
Revenue was: $39237.02
```

Step 15. How many orders were made in the period?

In [18]:

```
orders = chipo.order_id.value_counts().count()
orders
```

Out[18]:

```
1834
```

Step 16. What is the average revenue amount per order?

In [19]:

```
# Solution 1

chipo['revenue'] = chipo['quantity'] * chipo['item_price']
order_grouped = chipo.groupby(by=['order_id']).sum()
order_grouped.mean()['revenue']

```

Out[19]:

```
21.394231188658654
```

In [20]:

```
# Solution 2

chipo.groupby(by=['order_id']).sum().mean()['revenue']
```

Out[20]:

```
21.394231188658654
```

Step 17. How many different items are sold?

In [21]:

```
chipo.item_name.value_counts().count()
```

Out[21]:

```
50
```