Financial Data Analysis with Python

Instructor: Luping Yu

Mar 22, 2022

Project 1

It's time to test your new pandas skills! Use the provided .csv file to complete the tasks below!

Student Name 1:

Student Name 2:

Task 1. Bank M&A

Note: failed banks (Bank Name) are bought by another bank (Acquiring Institution).

In [138…

```
import pandas as pd

df = pd.read_csv('banklist.csv')
```

In [139…

```
df
```

Out[139]:

	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
...	...	...	...	...	...	...	...
546	Superior Bank, FSB	Hinsdale	IL	32646	Superior Federal, FSB	27-Jul-01	19-Aug-14
547	Malta National Bank	Malta	OH	6629	North Valley Bank	3-May-01	18-Nov-02
548	First Alliance Bank & Trust Co.	Manchester	NH	34264	Southern New Hampshire Bank & Trust	2-Feb-01	18-Feb-03
549	National State Bank of Metropolis	Metropolis	IL	3815	Banterra Bank of Marion	14-Dec-00	17-Mar-05
550	Bank of Honolulu	Honolulu	HI	21029	Bank of the Orient	13-Oct-00	17-Mar-05

551 rows × 7 columns

- Step 1. How many States (ST) are represented in this data set?

In [140…

```
len(df['ST'].drop_duplicates())
```

Out[140]:

```
44
```

- Step 2. What are the top 5 states with the most failed banks? (tips: groupby 'ST' and use count)

In [141…

```
df.groupby('ST').count()['Bank Name'].sort_values(ascending=False)[:5]
```

Out[141]:

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

- Step 3. What are the top 5 states with the highest **average** CERT (Certificate) value of failed banks? (tips: groupby 'ST' and use mean)

In [142…

```
df.groupby('ST').mean()['CERT'].sort_values(ascending=False)[:5]
```

Out[142]:

```
ST
NY    45616.400000
AZ    45201.062500
NJ    44113.428571
KY    43331.000000
NC    39959.000000
Name: CERT, dtype: float64
```

- Step 4. What are the top 5 acquiring institutions?

In [143…

```
df['Acquiring Institution'].value_counts()[:5]
#df.groupby('Acquiring Institution').count()['Bank Name'].sort_values(ascenc
```

Out[143]:

```
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
```

- Step 5. How many banks has the *State Bank of Texas* acquired?

In [144…

```
df[df['Acquiring Institution']=='State Bank of Texas']
```

Out[144]:

	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

- Step 6. What is the most common city in California for a bank to fail in? (tips: 'ST' == 'CA' and groupby 'City')

In [145…

```
df[df['ST']=='CA'].groupby('City')['Bank Name'].count().sort_values(ascendin
```

Out[145]:

```
City
Los Angeles    4
Name: Bank Name, dtype: int64
```

- Step 7. How many failed banks **don't** have the word "Bank" in their name?

In [146…

```
len(df) - sum(df['Bank Name'].str.contains('Bank'))
#sum(df['Bank Name'].str.contains('Bank') == False)
#sum(~df['Bank Name'].str.contains('Bank'))
#sum(df['Bank Name'].apply(lambda name: 'Bank' not in name))
```

Out[146]:

```
14
```

- Step 8. How many bank names consist of just two words? (e.g. "First Bank" , "Bank Georgia")

In [147…

```
sum(df['Bank Name'].str.split(' ').str.len() == 2)
#sum(df['Bank Name'].str.count(' ') + 1 == 2)
```

Out[147]:

```
113
```

Task 2. Occupation

In [148…

```
import pandas as pd

df = pd.read_csv('occupation.csv', index_col='user_id')
```

In [149…

```
df
```

Out[149]:

	age	gender	occupation	zip_code
user_id				
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213
...	...	...	...	...
939	26	F	student	33319
940	32	M	administrator	02215
941	20	M	student	97229
942	48	F	librarian	78209
943	22	M	student	77841

943 rows × 4 columns

- Step 1. What is the mean age per occupation? (tips: groupby 'occupation')

In [150…

```
df.groupby('occupation')['age'].mean()
#df.groupby('occupation')['age'].apply('mean')
#df.groupby('occupation')['age'].agg('mean')
```

Out[150]:

```
occupation
administrator    38.746835
artist           31.392857
doctor           43.571429
educator         42.010526
engineer         36.388060
entertainment    29.222222
executive        38.718750
healthcare       41.562500
homemaker        32.571429
lawyer           36.750000
librarian        40.000000
marketing        37.615385
none             26.555556
other            34.523810
programmer       33.121212
retired          63.071429
salesman         35.666667
scientist        35.548387
student          22.081633
technician       33.148148
writer           36.311111
Name: age, dtype: float64
```

- Step 2. For each occupation, calculate the minimum and maximum ages.

In [151…

```
df.groupby('occupation')['age'].agg(['min', 'max'])
```

Out[151]:

	min	max
occupation		
administrator	21	70
artist	19	48
doctor	28	64
educator	23	63
engineer	22	70
entertainment	15	50
executive	22	69
healthcare	22	62
homemaker	20	50
lawyer	21	53
librarian	23	69
marketing	24	55
none	11	55
other	13	64
programmer	20	63
retired	51	73
salesman	18	66
scientist	23	55
student	7	42
technician	21	55
writer	18	60

- Step 3. For each combination of occupation and gender, calculate the **median** age.

In [152…

```
df.groupby(['occupation', 'gender'])['age'].median()
```

Out[152]:

```
occupation     gender
administrator  F         38.5
               M         35.0
artist         F         30.0
               M         32.0
doctor         M         45.0
educator       F         40.5
               M         44.0
engineer       F         29.5
               M         36.0
entertainment  F         31.0
               M         25.0
executive      F         44.0
               M         36.0
healthcare     F         43.0
               M         47.0
homemaker      F         33.5
               M         23.0
lawyer         F         39.5
               M         34.0
librarian      F         39.0
               M         38.5
marketing      F         36.5
               M         34.5
none           F         32.5
               M         16.0
other          F         34.0
               M         32.0
programmer     F         32.0
               M         30.0
retired        F         70.0
               M         61.0
salesman       F         30.0
               M         35.0
scientist      F         28.0
               M         38.0
student        F         20.0
               M         22.0
technician     F         38.0
               M         30.0
writer         F         40.0
               M         31.0
Name: age, dtype: float64
```

- Step 4. Create a column named 'male', which equals to 1 if gender is 'M' and zero otherwise. (tips: covert 'gender' column to numeric type: M: 1; F: 0)

In [153…

```python
df['male'] = df['gender']
df['male'] = df['male'].replace('M', 1)
df['male'] = df['male'].replace('F', 0)

df
```

Out[153]:

	age	gender	occupation	zip_code	male
user_id					
1	24	M	technician	85711	1
2	53	F	other	94043	0
3	23	M	writer	32067	1
4	24	M	technician	43537	1
5	33	F	other	15213	0
...	...	...	...	...	...
939	26	F	student	33319	0
940	32	M	administrator	02215	1
941	20	M	student	97229	1
942	48	F	librarian	78209	0
943	22	M	student	77841	1

943 rows × 5 columns

In [154…

```
def gender_to_numeric(x):
    if x == 'M':
        return 1
    if x == 'F':
        return 0

df['male'] = df['gender'].apply(gender_to_numeric)

df
```

Out[154]:

	age	gender	occupation	zip_code	male
user_id					
1	24	M	technician	85711	1
2	53	F	other	94043	0
3	23	M	writer	32067	1
4	24	M	technician	43537	1
5	33	F	other	15213	0
...	...	...	...	...	...
939	26	F	student	33319	0
940	32	M	administrator	02215	1
941	20	M	student	97229	1
942	48	F	librarian	78209	0
943	22	M	student	77841	1

943 rows × 5 columns

In [155…

```
import numpy as np

df['male'] = np.where(df['gender'] == 'M', 1, 0)
```

```
df
```

Out[155]:

	age	gender	occupation	zip_code	male
user_id					
1	24	M	technician	85711	1
2	53	F	other	94043	0
3	23	M	writer	32067	1
4	24	M	technician	43537	1
5	33	F	other	15213	0
...	...	...	...	...	...
939	26	F	student	33319	0
940	32	M	administrator	02215	1
941	20	M	student	97229	1
942	48	F	librarian	78209	0
943	22	M	student	77841	1

943 rows × 5 columns

- Step 5. What is the Male ratio per occupation? Sort it from the most to the least.

In [156…

```
df.groupby('occupation')['male'].mean().sort_values(ascending=False)
```

Out[156]:

```
occupation
doctor           1.000000
engineer         0.970149
technician       0.962963
retired          0.928571
programmer       0.909091
executive        0.906250
scientist        0.903226
entertainment    0.888889
lawyer           0.833333
salesman         0.750000
educator         0.726316
student          0.693878
other            0.657143
marketing        0.615385
writer           0.577778
none             0.555556
administrator    0.544304
artist           0.535714
librarian        0.431373
healthcare       0.312500
homemaker        0.142857
Name: male, dtype: float64
```