Assignment 2

MAS DSE200

Due October 19th, 2020

Instructions

- You don't need to explain your approach (unless specified) so please be concise in your submission.
- To obtain full marks for a question, both the answer and the code should be correct.
- Completely wrong (or missing) code with correct answer will result in zero marks.
- Please code the solution in the space provided.
- Please submit **both the PDF and IPynb** files

Imports

Import necessary packages

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

Part 1: Titanic

Preliminaries

Grab the dataset from here and store it in a Pandas dataframe called passengers.

In [2]:
```
passengers = pd.read_csv('https://raw.githubusercontent.com/justmarkham/DAT8/master/data/titanic.csv',
sep=',')
passengers.to_csv('passengers.csv', sep=',', index=False)
```

In [3]:
```
passengers = pd.read_csv('./passengers.csv')
```

1: Get to know your data

1.1: Print the first 10 entries in the dataframe to see what the columns are and what some values will look like.

In [4]:
```
passengers.head(10)
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

1.2: Next, set the index of the dataframe to the `PassengerId` column, and print the first 10 elements again to ensure the change took place.

In [5]:
```
try:
    passengers.set_index('PassengerId', inplace=True)
except KeyError:
    print('Index already set')

passengers.head(10)
```

Out[5]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
PassengerId											
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

1.3: How many samples are there in this dataset?

In [6]:
```
len(passengers)
```

Out[6]: 891

3: Number of passengers in different classes

3.1: What is the number of passenges in different classes according to this dataset? (Hint: Pclass represents the class of a passenger.)

In [7]:
```
for i in range(0,len(passengers.Pclass.unique())):
    num = passengers[passengers['Pclass'] == passengers.Pclass.unique()[i]].Pclass.count()
    print('class = {}, number of passengers = {}.'.format(passengers.Pclass.unique()[i],num))
```

```
class = 3, number of passengers = 491.
class = 1, number of passengers = 216.
class = 2, number of passengers = 184.
```

4: Fares

4.1: How many different fares were charged on the Titanic based on the dataset?

In [8]:
```
len(passengers.Fare.unique())
```

Out[8]: 248

4.2: Find the top 10 fares charged from the passengers. **Report these fare values**, and then **calculate the total number of passengers** who paid one of these fares (every passenger paid a unique fare value).

In [9]:
```
top_fares_with_counts = passengers.Fare.value_counts().head(10)

top_fares = top_fares_with_counts.keys()
passengers_sum = sum(top_fares_with_counts)

print('The most popular fares in the dataset are: ', top_fares)
print(passengers_sum, 'of the passengers paid these fares.')
```

```
The most popular fares in the dataset are:  Float64Index([8.05, 13.0, 7.8958, 7.75, 26.0, 10.5, 7.92
5, 7.775, 26.55, 0.0], dtype='float64')
276 of the passengers paid these fares.
```

4.3: Create a new dataframe, called `passengers_filtered`, that includes only entries of passengers who paid one of these top 10 fares. **Report the number of samples** in the original dataset and in the new dataset to ensure the desired effect took place.

hint: Check out the Pandas Series function isin

In [10]:
```
passengers_filtered = passengers[passengers.Fare.isin(top_fares)]

print('Before:', len(passengers))
print('After:', len(passengers_filtered))
```

```
Before: 891
After: 276
```

5: Ages

5.1: What was the minimum, maximum and average age of passengers on the Titanic?

In [11]:
```
print('std:',  passengers['Age'].std())
print('mean:', passengers['Age'].mean())
print('min:',  passengers['Age'].min())
print('max:',  passengers['Age'].max())
```

```
std: 14.526497332334044
mean: 29.69911764705882
min: 0.42
max: 80.0
```

5.2: How many passengers on the Titanic were within two standard deviations of the mean age calculated in **5.1**?

In [12]:
```
two_std = passengers['Age'].std() * 2
mean = passengers['Age'].std()

within_two_std = passengers[passengers.Age.gt(mean - two_std) & passengers.Age.lt(mean + two_std)]

print(len(within_two_std), 'of the original', len(passengers))
```

```
590 of the original 891
```

5.3: How many of the passengers found in **5.2** were females over the age of 25?

In [13]:
```
print(len(within_two_std[(within_two_std.Sex == 'female') & (within_two_std.Age > 25)]))
```

```
100
```

5.4: What are the 10 **most** common ages of passengers according to this dataset?

In [14]:
```
passengers.Age.value_counts().head(10)
```

Out[14]:
```
24.0    30
22.0    27
18.0    26
19.0    25
30.0    25
28.0    25
21.0    24
25.0    23
36.0    22
29.0    20
Name: Age, dtype: int64
```

Part 2: Alcohol

In [15]:
```
drinks = pd.read_csv('./drinks.csv', sep=',')
```

6: Print the first 5 entries in the dataframe

In [16]:
```
drinks.head(5)
```

Out[16]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
0	Afghanistan	0	0	0	0.0000
1	Albania	89	132	54	4.8675
2	Algeria	25	0	14	0.6903
3	Andorra	245	138	312	12.3015
4	Angola	217	57	45	5.6463

7: What country has the highest number of wine servings, and what is this number? What countries consume no wine?

In [17]:
```
max_wine_servings = drinks['wine_servings'].max()
min_wine_servings = drinks['wine_servings'].min()

max_wine_servings_countries = drinks[drinks.wine_servings == max_wine_servings].country.unique()
min_wine_servings_countries = drinks[drinks.wine_servings == min_wine_servings].country.unique()

print('The following countries have the highest number of wine servings:\n{}\n'.format(max_wine_serving
s_countries))
print('The highest number of wine servings is {}.\n'.format(max_wine_servings))
print('The following countries consume no wine:\n{}'.format(min_wine_servings_countries))
```

```
The following countries have the highest number of wine servings:
['France']

The highest number of wine servings is 370.

The following countries consume no wine:
['Afghanistan' 'Bangladesh' 'Bhutan' 'Burundi' 'Eritrea' 'Ethiopia'
 'India' 'Indonesia' 'Iran' 'Iraq' 'Kuwait' 'Lesotho' 'Libya' 'Malaysia'
 'Maldives' 'Marshall Islands' 'Mauritania' 'Monaco' 'Myanmar' 'Nepal'
 'North Korea' 'Pakistan' 'Rwanda' 'San Marino' 'Saudi Arabia' 'Somalia'
 'Sri Lanka' 'Sudan' 'Tajikistan' 'Uganda' 'Yemen']
```

8: What country has the highest beer + wine consumption?

In [18]:
```
drinks['beer_wine_servings'] = drinks.wine_servings + drinks.beer_servings
max_beer_wine_consumption = drinks['beer_wine_servings'].max()

max_beer_wine_consumption_countries = drinks[drinks.beer_wine_servings == max_beer_wine_consumption].co
untry.unique()

print('The following countries have the highest beer + wine consumption.\n{}\n'.format(max_beer_wine_co
nsumption_countries))
```

```
The following countries have the highest beer + wine consumption.
['Andorra']
```

9: Among the countries that consume at least some beer, which country consumes the least combined alcohol(beer+wine+spirit)?

In [19]:
```
drinks['beer_wine_spirit_servings'] = drinks.beer_servings + drinks.wine_servings + drinks.spirit_servi
ngs
min_beer_wine_spirit_servings_with_beer = drinks[drinks.beer_servings > 0]['beer_wine_spirit_servings']
.min()

min_beer_wine_spirit_servings_with_beer_countries = drinks[(drinks.beer_wine_spirit_servings == min_bee
r_wine_spirit_servings_with_beer) & (drinks['beer_servings'] > 0)].country.unique()

print('The following countries consume the least combined alcohol(beer + wine + spirit) with at least s
ome beer:\n{}\n'.format(min_beer_wine_spirit_servings_with_beer_countries))
```

```
The following countries consume the least combined alcohol(beer + wine + spirit) with at least some b
eer:
['Comoros']
```

10: Are there countries that consume some(non-zero) spirit and some(non-zero) wine and no beer? If so, which countries?

In [20]:
```
countries = drinks[(drinks['spirit_servings'] > 0) & (drinks['wine_servings'] > 0) & (drinks['beer_serv
ings'] == 0)].country.unique()

if(any(countries)):
    print('Yes, there are countries that consume some(non-zero) spirit and some(non-zero) wine and no b
eer.\n')
    print('The following countries consume some(non-zero) spirit and some(non-zero) wine and no beer:\n
{}\n'.format(countries))
else:
    print('No countries consume some(non-zero) spirit and some(non-zero) wine and no beer')
```

```
Yes, there are countries that consume some(non-zero) spirit and some(non-zero) wine and no beer.

The following countries consume some(non-zero) spirit and some(non-zero) wine and no beer:
['Cook Islands']
```

Part 3: Numpy

11: Generate an integer random numpy array, A, of shape (5,5) with elements from 1 to 5(including 1 and 5). Print the generated random array. (Hint: Look at the documentation of numpy.random.randint)

In [21]:
```
A = np.random.randint(1, 6, size=(5,5))
print(A)
```

```
[[2 5 4 4 3]
 [4 1 5 5 3]
 [4 4 1 3 2]
 [2 2 5 3 5]
 [5 2 4 4 1]]
```

12 Print the transpose of A.

In [22]:
```
print(np.transpose(A))
```

```
[[2 4 4 2 5]
 [5 1 4 2 2]
 [4 5 1 5 4]
 [4 5 3 3 4]
 [3 3 2 5 1]]
```

13 Print the first 3 rows of A.

In [23]:
```
print(A[:3,])
```

```
[[2 5 4 4 3]
 [4 1 5 5 3]
 [4 4 1 3 2]]
```

14 Print the first 3 columns of A.

In [24]:
```
print(A[:,:3])
```

```
[[2 5 4]
 [4 1 5]
 [4 4 1]
 [2 2 5]
 [5 2 4]]
```

15 Add this vector B=[1,2,3,4,5] to all the rows of A. (Hint: Use broadcasting)

In [25]:
```
def vector_row_adder(a,b):
    return a + b

B = [1,2,3,4,5]
C = vector_row_adder(A,B)

print(C)
```

```
[[ 3  7  7  8  8]
 [ 5  3  8  9  8]
 [ 5  6  4  7  7]
 [ 3  4  8  7 10]
 [ 6  4  7  8  6]]
```

16 Add this vector B=[1,2,3,4,5] to all the columns of A. (Hint: Use np.reshape and broadcasting)

In [26]:
```
def vector_column_adder(a,b):
    return a + np.array(b).reshape(np.array(b).shape[0], 1)

B = [1,2,3,4,5]
C = vector_column_adder(A,B)

print(C)
```

```
[[ 3  6  5  5  4]
 [ 6  3  7  7  5]
 [ 7  7  4  6  5]
 [ 6  6  9  7  9]
 [10  7  9  9  6]]
```

In [ ]: