Mini Assignment 2

Please add one code cell after each question and create a program to answer the question. Make sure your code runs without error. After you are finished, click on File -> Print -> Save as PDF to create a PDF output and upload it on Brightspace.

NOTE: Make sure your PDF does not have your name or any identifying information in the name or content of the file. Anonymity is essential for the peer-review process.

Project: Auto Fuel Efficiency

In this project we try to use all what we have learned so far to look into auto fuel efficiency and the factors that may contribute to that. Let's import our datafile mpg.csv, which contains fuel economy data for 398 cars. Here are the variables included in the dataset.

- mpg: miles per gallon
- cylinders: # of cylinders
- displacement: engine displacement in liters
- horsepower: engine horsepower
- weight: vehicle weight
- acceleration
- model_year: year manufactured
- origin: 1: North America; 2: Europe; 3: Asia and others
- name: name of the car including make and model

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
from numpy import nan
```

1. Use pandas to read the csv file into a dataframe. Make the 'name' column as the index labels for your dataframe, and identify any value of '?' as NAN in your dataframe (for some reason the missing data is specified as '?' in the csv file). Checkout the head, info, and descriptive statistics of the dataframe.

```
data = pd.DataFrame(pd.read_csv('/content/drive/MyDrive/colabData/mpg.csv', na_values = '?'))


data.set_index('name', inplace = True)
data.replace('?', nan, inplace = True)


data.head()
```

name	mpg	cylinders	displacement	horsepower	weight	acceleration	model_y
chevrolet chevelle malibu	18.0	8	307.0	130.0	3504	12.0	
buick skylark 320	15.0	8	350.0	165.0	3693	11.5	
plymouth satellite	18.0	8	318.0	150.0	3436	11.0	

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 398 entries, mazda glc to hi 1200d
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    float64
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64
 7   origin        398 non-null    int64
 8   mpgcyl        398 non-null    float64
 9   make          398 non-null    category
 10  make_coded    398 non-null    int8
dtypes: category(1), float64(5), int64(4), int8(1)
memory usage: 33.4+ KB
```

```
data.describe()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623	15.568090
std	7.815984	1.701004	104.269838	38.199187	846.841774	2.757689
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000

2. Display the slice to show the mpg for the car named 'ford torino'.

```
data["mpg"]["ford torino"]
```

```
17.0
```

3. Display the slice to show all the rows but only two columns of mpg and cylinders.

```
data[["mpg", "cylinders"]]
```

name	mpg	cylinders
chevrolet chevelle malibu	18.0	8
buick skylark 320	15.0	8
plymouth satellite	18.0	8
amc rebel sst	16.0	8
ford torino	17.0	8
...	...	...
ford mustang gl	27.0	4
vw pickup	44.0	4
dodge rampage	32.0	4
ford ranger	28.0	4
chevy s-10	31.0	4

398 rows × 2 columns

4. Display the slice using index numbers to show all the rows and only the first four columns.

```
data.iloc[:, :4]
```

name	mpg	cylinders	displacement	horsepower
chevrolet chevelle malibu	18.0	8	307.0	130
buick skylark 320	15.0	8	350.0	165
plymouth satellite	18.0	8	318.0	150
amc rebel sst	16.0	8	304.0	150
ford torino	17.0	8	302.0	140
...	...	...	...	...
ford mustang gl	27.0	4	140.0	86
vw pickup	44.0	4	97.0	52
dodge rampage	32.0	4	135.0	84
ford ranger	28.0	4	120.0	79
chevy s-10	31.0	4	119.0	82

398 rows × 4 columns

5. Display the slice to show all the rows with horsepower over 200 and the range of columns from mpg to horsepower

```
data[data["horsepower"]>200].loc[:, "mpg":"horsepower"]
```

	mpg	cylinders	displacement	horsepower

6. Display the slice to show all the rows with mpg lower than 17. Next, display the number of such cars.

buick estate wagon (sw)	14.0	8	455.0	225.0

```
data[data["mpg"] < 17]
```

name	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
buick skylark 320	15.0	8	350.0	165.0	3693	11.5	70
amc rebel sst	16.0	8	304.0	150.0	3433	12.0	70
ford galaxie 500	15.0	8	429.0	198.0	4341	10.0	70
chevrolet impala	14.0	8	454.0	220.0	4354	9.0	70
plymouth fury iii	14.0	8	440.0	215.0	4312	8.5	70
...	...	...	...	...	...	...	...
ford thunderbird	16.0	8	351.0	149.0	4335	14.5	77
peugeot 604sl	16.2	6	163.0	133.0	3410	15.8	78

```
data["mpg"][data["mpg"] < 17].count()
```

```
92
```

7. Display all the rows with horsepower over 150 and cylinders equal to 6; further slice the view to only display the three columns of mpg, horsepower, and cylinders.

```
data[np.logical_and(data["horsepower"] > 150, data["cylinders"] == 6)].iloc[:, :3]
```

name	mpg	cylinders	displacement
buick regal sport coupe (turbo)	17.7	6	231.0

8. Add a new column to the dataframe called "mpgcyl" that calculates mpg/cylinders. Next, sort the dataframe based on the values of this new column from large to small. Make sure the sort is hapenning "inplace", so it changes the order of the rows in the dataframe.

```
data["mpgcyl"] = data.mpg / data.cylinders
```

```
data.sort_values("mpgcyl", ascending = False, inplace = True)
```

9. Write the code to display the percentage of missing values in each column.

```
data.isna().sum()
```

```
mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model_year      0
origin          0
mpgcyl          0
dtype: int64
```

10. Fill the missing values of each column with the average value for that column.

```
data.fillna(data.mean(), inplace = True)
```

11. Add a new column called 'make' that includes only the make of each car. Make of each car is represented as the first word within their name.

    Hint: you may use split() method within a list comprehention, or alternatively use a combination of map() and split() methods.

```
def brand(x):
  namelist = [i for i in x.split()]
  return namelist[0]

data['make'] = data.index.map(brand)
```

12. In the 'make' column, replace 'vw' with 'wolkswagen', and 'maxda' with 'mazda'.

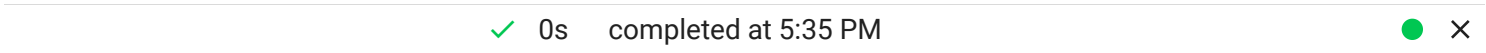
```
data.make.replace(['vw', 'maxda'], ['volkswagen','mazda'], inplace=True)
```

13. Label Encode the 'make' column and add it as a new column called 'make_coded'. It means each make should be represented with a number rather than string.

```
data["make"] = data["make"].astype("category")
data["make_coded"] = data["make"].cat.codes
```

```
data
```

name	mpg	cylinders	displacement	horsepower	weight	acceleration	model_
mazda glc	46.6	4	86.0	65.0	2110	17.9	
honda civic 1500 gl	44.6	4	91.0	67.0	1850	13.8	
vw rabbit c (diesel)	44.3	4	90.0	48.0	2085	21.7	
vw pickup	44.0	4	97.0	52.0	2130	24.6	
vw dasher (diesel)	43.4	4	90.0	48.0	2335	23.7	
...	...	...	...	...	...	...	
dodge d200	11.0	8	318.0	210.0	4382	13.5	
oldsmobile omega	11.0	8	350.0	180.0	3664	11.0	
ford f250	10.0	8	360.0	215.0	4615	14.0	


0s
completed at 5:35 PM