Module 1 Homework

ISE-529 Predictive Analytics
Chinmay Gherde

In [1]:

```
# importing libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1. For this problem, we will be using the file "Cars Data.csv"

a. Load the file Cars Data.csv into a dataframe named cars and display the first 10 rows of the dataframe.

In [2]:

```
#loading the file from local path

cars = pd.read_csv(r'C:\Users\Chinmay\Downloads\cars.csv', header=3, index_col=None)
cars.head(10)
```

Out[2]:

	Make	Model	DriveTrain	Origin	Type	Cylinders	Engine Size (L)	Horsepower	Invoice	Len
0	Acura	3.5 RL 4dr	Front	Asia	Sedan	6.0	3.5	225	$39,014	
1	Acura	3.5 RL w/Navigation 4dr	Front	Asia	Sedan	6.0	3.5	225	$41,100	
2	Acura	MDX	All	Asia	SUV	6.0	3.5	265	$33,337	
3	Acura	NSX coupe 2dr manual S	Rear	Asia	Sports	6.0	3.2	290	$79,978	
4	Acura	RSX Type S 2dr	Front	Asia	Sedan	4.0	2.0	200	$21,761	
5	Acura	TL 4dr	Front	Asia	Sedan	6.0	3.2	270	$30,299	
6	Acura	TSX 4dr	Front	Asia	Sedan	4.0	2.4	200	$24,647	
7	Audi	A4 1.8T 4dr	Front	Europe	Sedan	4.0	1.8	170	$23,508	
8	Audi	A4 3.0 4dr	Front	Europe	Sedan	6.0	3.0	220	$28,846	
9	Audi	A4 3.0 convertible 2dr	Front	Europe	Sedan	6.0	3.0	220	$38,325	

Answer 1a complete

b. Use the describe() function to produce a numerical summary of the variables in the dataset. (10 points)

In [3]:

```
cars.describe()
```

Out[3]:

	Cylinders	Engine Size (L)	Horsepower	Length (IN)	MPG (City)	MPG (Highway)	Weight (LBS)
count	426.000000	428.000000	428.000000	428.000000	428.000000	428.000000	428.000000
mean	5.807512	3.196729	215.885514	186.362150	20.060748	26.843458	3577.953271
std	1.558443	1.108595	71.836032	14.357991	5.238218	5.741201	758.983215
min	3.000000	1.300000	73.000000	143.000000	10.000000	12.000000	1850.000000
25%	4.000000	2.375000	165.000000	178.000000	17.000000	24.000000	3104.000000
50%	6.000000	3.000000	210.000000	187.000000	19.000000	26.000000	3474.500000
75%	6.000000	3.900000	255.000000	194.000000	21.250000	29.000000	3977.750000
max	12.000000	8.300000	500.000000	238.000000	60.000000	66.000000	7190.000000

Answer 1b complete

c. Use the groupby() and size() functions to create a table of the number of observations in the dataset of each car Make (e.g., Acura, Audi, etc.) (15 points)

In [4]:

```
grouped_by_make_size = cars.groupby(["Make"]).size()
print(grouped_by_make_size)
```

```
Make
Acura             7
Audi             19
BMW              20
Buick             9
Cadillac          8
Chevrolet        27
Chrysler         15
Dodge            13
Ford             23
GMC               8
Honda            17
Hummer            1
Hyundai          12
Infiniti          8
Isuzu             2
Jaguar           12
Jeep              3
Kia              11
Land Rover        3
Lexus            11
Lincoln           9
MINI              2
Mazda            11
Mercedes-Benz    26
Mercury           9
Mitsubishi       13
Nissan           17
Oldsmobile        3
Pontiac          11
Porsche           7
Saab              7
Saturn            8
Scion             2
Subaru           11
Suzuki            8
Toyota           28
Volkswagen       15
Volvo            12
dtype: int64
```

Answer 1c complete

d. Use the corr() function to create a correlation matrix of the numeric attributes in the cars dataset. Use the "pearson" correlation coefficient algorithm (15 points)

In [5]:

```
# In order to create correlation matrix, only numerical attributes will have to be consider
#Checking the dtypes of the data

cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 428 entries, 0 to 427
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Make             428 non-null    object 
 1   Model            428 non-null    object 
 2   DriveTrain       428 non-null    object 
 3   Origin           428 non-null    object 
 4   Type             428 non-null    object 
 5   Cylinders        426 non-null    float64
 6   Engine Size (L)  428 non-null    float64
 7   Horsepower       428 non-null    int64  
 8   Invoice          428 non-null    object 
 9   Length (IN)      428 non-null    int64  
 10  MPG (City)       428 non-null    int64  
 11  MPG (Highway)    428 non-null    int64  
 12  MSRP             428 non-null    object 
 13  Weight (LBS)     428 non-null    int64  
 14  Wheelbase (IN)   428 non-null    int64  
dtypes: float64(2), int64(6), object(7)
memory usage: 50.3+ KB
```

In [6]:

```
correlation_matrix = cars.corr(method="pearson")
print(correlation_matrix)
```

```
                Cylinders  Engine Size (L)  Horsepower  Length (IN)  \
Cylinders        1.000000         0.908002    0.810341     0.547783   
Engine Size (L)  0.908002         1.000000    0.787435     0.637448   
Horsepower       0.810341         0.787435    1.000000     0.381554   
Length (IN)      0.547783         0.637448    0.381554     1.000000   
MPG (City)      -0.684402        -0.709471   -0.676699    -0.501526   
MPG (Highway)   -0.676100        -0.717302   -0.647195    -0.466092   
Weight (LBS)     0.742209         0.807867    0.630796     0.690021   
Wheelbase (IN)   0.546730         0.636517    0.387398     0.889195   

                MPG (City)  MPG (Highway)  Weight (LBS)  Wheelbase (IN)  
Cylinders        -0.684402      -0.676100      0.742209        0.546730  
Engine Size (L)  -0.709471      -0.717302      0.807867        0.636517  
Horsepower       -0.676699      -0.647195      0.630796        0.387398  
Length (IN)      -0.501526      -0.466092      0.690021        0.889195  
MPG (City)        1.000000       0.941021     -0.737966       -0.507284  
MPG (Highway)     0.941021       1.000000     -0.790989       -0.524661  
Weight (LBS)     -0.737966      -0.790989      1.000000        0.760703  
Wheelbase (IN)   -0.507284      -0.524661      0.760703        1.000000  
```

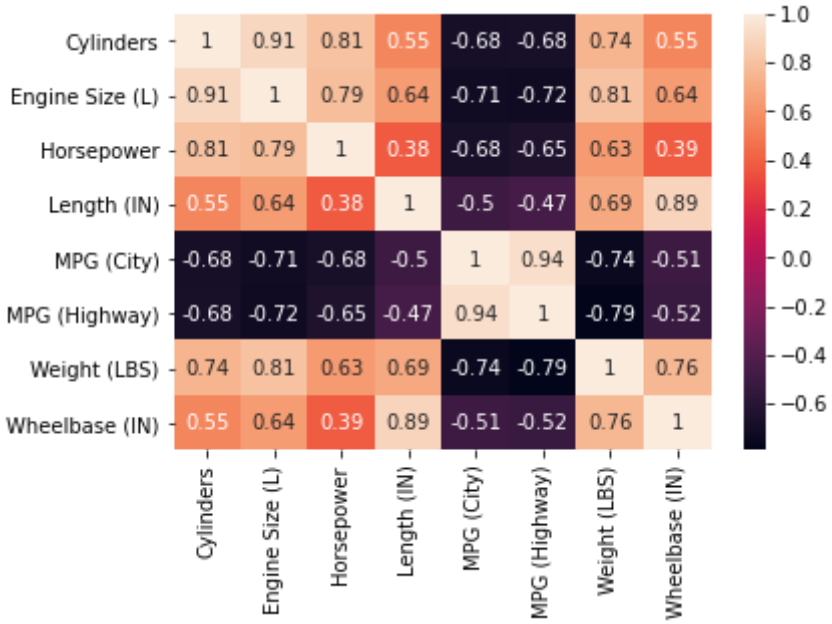
In [7]:

```
# In otrder to better visualize this:

sns.heatmap(correlation_matrix, annot=True)
```

Out[7]:

<AxesSubplot:>



Answer 1d complete

e. Add a new attribute to the dataframe called HP_Groups which has one of the following values:

- "Low": Horsepower is <= 200
- "Medium": Horsepower is > 200 and <= 300
- "High": Horsepower is > 300

Hint: create a function that takes a number and return one of the three bins and then use the apply method

In [8]:

```
#Creating a function first

def groups(hp):
    if hp <= 200:
        return "Low"
    elif hp>200 and hp<=300:
        return "Medium"
    elif hp>300:
        return "High"
```

In [9]:

```
cars["HP_Groups"] = cars["Horsepower"].apply(groups)
```

In [10]:

```
cars[["Horsepower", "HP_Groups"]].head()
```

Out[10]:

	Horsepower	HP_Groups
0	225	Medium
1	225	Medium
2	265	Medium
3	290	Medium
4	200	Low

Answer 1e complete

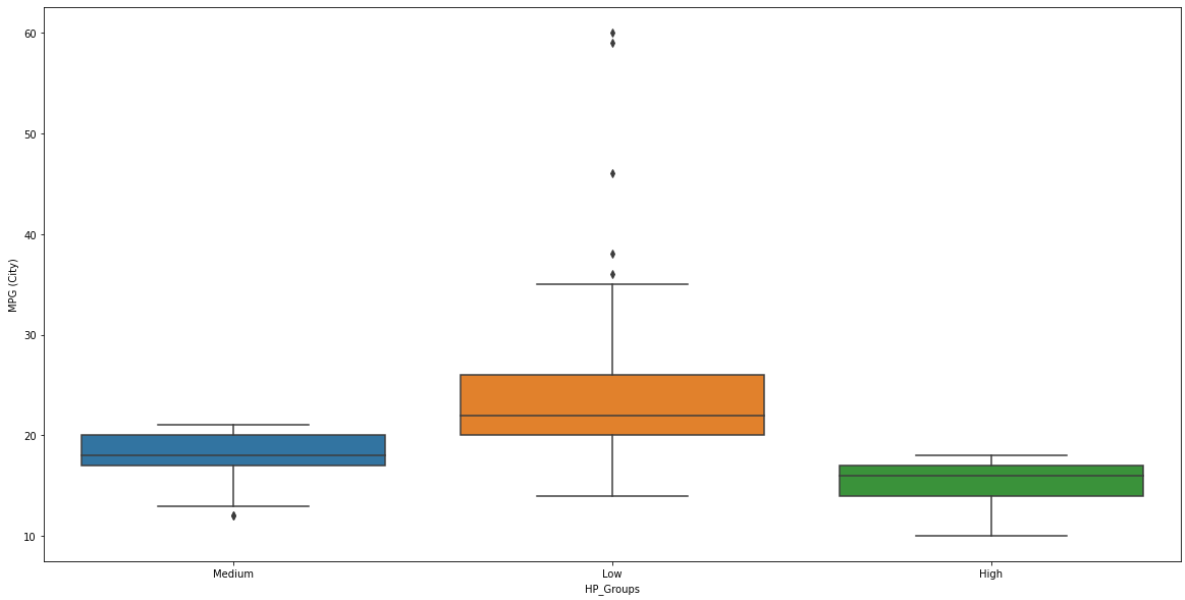
f. Using a Seaborn, create side-by-side boxplots showing the MPG (City) for each of the three HP_Groups

In [11]:

```
plt.figure(figsize=[20,10])
sns.boxplot(x=cars["HP_Groups"], y=cars["MPG (City)"])
plt.plot()
```

Out[11]:

```
[]
```



Answer 1f complete

2. Function Definition

A common metric of health is the Body Mass Index (BMI), which is calculated simply as the weight in kilograms divided by the height in meters squared (BMI = kg/m2)

a. Create a function called calculate_bmi() that takes height in inches and weight in pounds as parameters and returns the BMI. Use the conversions inches $*$ 0.025 = m and pounds $*$ 0.453592 = kg. Test the function with a height of 72" and a weight of 190 lbs. (15 points)

In [12]:

```
def calculate_bmi(height, weight):
    height_m = height*0.025
    weight_kgs = weight*0.45392
    bmi = weight_kgs/(height_m)**2
    return bmi
```

In [13]:

```
height = 72
weight = 190

calculate_bmi(height, weight)
```

Out[13]:

```
26.618765432098762
```

Answer 2a complete

b. General standard categories for BMI are given by:

- BMI < 18.5: Underweight
- BMI 18.5 - 24.9: Normal weight
- BMI 25 - 29.9: Overweight
- BMI 30 or greater: Obese

Create a function called determine_weight_category that takes height in inches and weight in pounds as parameters and returns the category the person falls into. Test your function with the following values: (15 points)

- Height 60" / Weight 160 lbs
- Height 68" / Weight 160 lbs
- Height 72" / Weight 160 lbs

In [14]:

```
def determine_weight_category(height, weight):
    bmi = calculate_bmi(height, weight)
    if bmi<18.5:
        return "Underweight"
    elif bmi>=18.5 and bmi<=24.9:
        return "Normal weight"
    elif bmi>=25 and bmi<=29.9:
        return "Overweight"
    elif bmi>=30:
        return "Obese"
```

In [15]:

```
case1 = determine_weight_category(60,160)
case2 = determine_weight_category(68,160)
case3 = determine_weight_category(72,160)

print(case1)
print(case2)
print(case3)
```

```
Obese
Overweight
Normal weight
```

Answer 2b complete

c. You want to create a plot of the BMI for a 180-pound individual at various heights from 60" to 84" (in one-inch increments). Create a vector ‘heights’ to hold the various heights from 60 to 84 and write a for-loop to call your calculate_bmi() function for each value of heights. Then, use these two vectors to create a scatter plot showing the relationship. Give the chart a title of ‘BMI at Various Heights for a 180-Pound Individual’. For full credit, label the axes and give the chart a title.

In [44]:

```
heights_vector = np.arange(60,84+1,1)
print(heights_vector)
```

```
[60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
 84]
```

In [45]:

```
weight_vector = 180

bmi_vector = []
for h in heights_vector:
    ans = calculate_bmi(h, weight_vector)
    bmi_vector.append(ans)
```

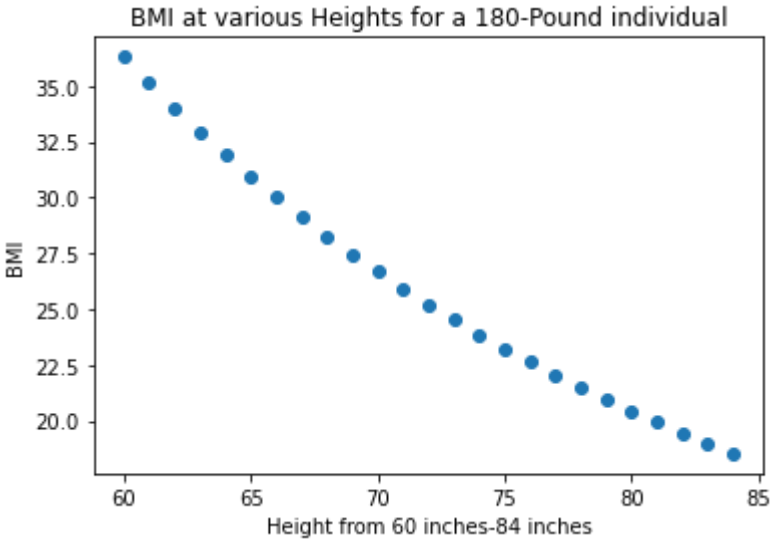
In [49]:

```
plt.scatter(x=heights_vector, y=bmi_vector)
plt.title("BMI at various Heights for a 180-Pound individual")
plt.xlabel("Height from 60 inches-84 inches")
plt.ylabel("BMI")

plt.plot()
```

Out[49]:

```
[]
```



Answer 2c complete

d. Repeat part c from above but without using a loop

In [63]:

```
## Using the map function

weight_vector_2 = [weight_vector]*25

bmi_vector_2 = list(map(calculate_bmi, heights_vector,weight_vector_2))
```

In [66]:

```
print(bmi_vector_2)
```

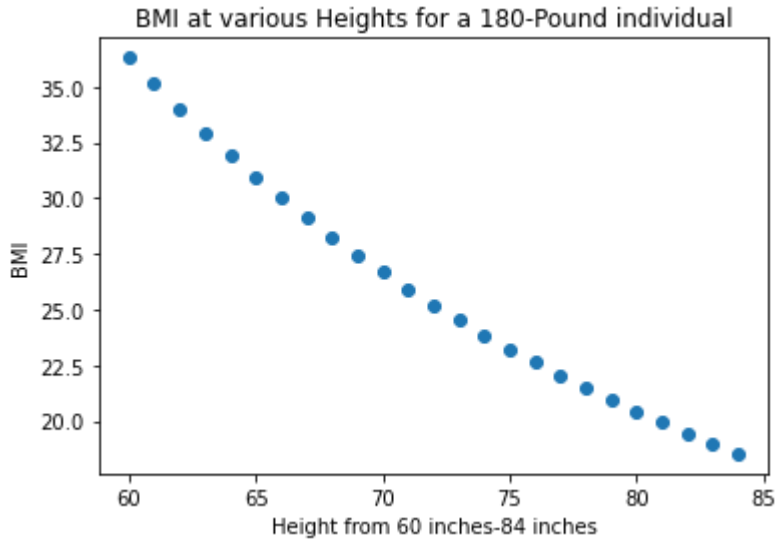
```
[36.3136, 35.13274926095135, 34.00857440166493, 32.93750566893423, 31.916249
999999994, 30.941765680473374, 30.01123966942148, 29.122067275562486, 28.271
833910034598, 27.45829867674858, 26.679379591836735, 25.9331402499504, 25.21
7777777777776, 24.531611934696937, 23.87307523739956, 23.240704, 22.63313019
3905817, 22.049074042840275, 21.487337278106505, 20.946796987662232, 20.426
4, 19.925157750342937, 19.44214158239143, 18.976478443896063, 18.52734693877
5512]
```

In [67]:

```
plt.scatter(x=heights_vector, y=bmi_vector_2)
plt.xlabel("Height from 60 inches-84 inches")
plt.ylabel("BMI")
plt.title("BMI at various Heights for a 180-Pound individual")
plt.plot()
```

Out[67]:

```
[]
```



Answer 2d complete

3) Demographics Dataset Analysis

a. For this problem, load the file "demographics.csv" into a DataFrame called "demos". Uset the "type" function to verify that it is a dataframe

In [16]:

```
demos = pd.read_csv(r'C:\Users\Chinmay\Downloads\demographics.csv')
demos.head()
```

Out[16]:

	CONT	NAME	Region	Population	Urban Pop Percentage	AdultLiteracyPct	MaleSchoolPct	FemaleS
0	91	BAHAMAS	AMR	323063	0.90	NaN	0.85	
1	91	BELIZE	AMR	269736	0.49	0.77	0.98	
2	91	CANADA	AMR	32268243	0.81	NaN	1.00	
3	91	COSTA RICA	AMR	4327228	0.62	0.96	0.90	
4	91	CUBA	AMR	11269400	0.76	1.00	0.96	

In [18]:

```
print(type(demos))
```

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

Answer 3a complete

b. For each continent, summarize the average (mean) city size

In [19]:

```
summary_avg_size = demos.groupby(["Region"]).agg({"Population":["mean"]})
print(summary_avg_size)
```

```
          Population
                mean
Region
AFR     1.604529e+07
AMR     2.532383e+07
EMR     2.561912e+07
EUR     1.930423e+07
SEAR    1.505935e+08
WPR     6.178509e+07
```

Answer 3b complete

c. Find the size of the largest country in each continent

In [124]:

```
largest_country_cont = demos[['Region','NAME','Population']].sort_values('Population',ascen
largest_country_cont
```

Out[124]:

Region	NAME	Population
AFR	NIGERIA	131529669
AMR	UNITED STATES	298212895
EMR	PAKISTAN	157935075
EUR	RUSSIA	143201572
SEAR	INDIA	1103370802
WPR	CHINA	1323344591

Answer 3c complete

d. What is the average percentage of males and females that attend school in each continent?

In [20]:

```
avg_school_attendance = demos.groupby(["Region"]).agg({"MaleSchoolPct":['mean'], "FemaleSch
print(avg_school_attendance)
```

```
       MaleSchoolPct FemaleSchoolPct
                mean             mean
Region
AFR         0.733250         0.673750
AMR         0.931290         0.924839
EMR         0.816111         0.767222
EUR         0.941020         0.935510
SEAR        0.861429         0.844286
WPR         0.932857         0.920476
```

Answer 3d complete

e. What is the average percentage of females that attend school in each continent. Sort the list from highest to lowest.

In [31]:

```
avg_female_attendance = demos.groupby(["Region"]).agg({"FemaleSchoolPct": ["mean"]})
#desc_avg_female_attendance = avg_female_attendance.sort_values(by=['FemaleSchoolPct mean']

sorted_ans = avg_female_attendance.sort_values(by=[("FemaleSchoolPct", "mean")], ascending=
print(sorted_ans)
```

```
       FemaleSchoolPct
                  mean
Region
EUR           0.935510
AMR           0.924839
WPR           0.920476
SEAR          0.844286
EMR           0.767222
AFR           0.673750
```

Answer 3e complete