Module 2A Homework

ISE-529 Predictive Analytics

Jui-Li Chen

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 [1]:

```
import pandas as pd
cars = pd.read_csv(r'C:\Users\Jerry\OneDrive - University of Southern California\Desktop\ISE_529\HW2\Cars Data.csv', skiprows=3)
cars.head(10)
```

Out[1]:

	Make	Model	DriveTrain	Origin	Type	Cylinders	Engine Size (L)	Horsepower	Invoice	L
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	

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

In [2]:

```
cars.describe()
```

Out[2]:

	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

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 [3]:

```
cars.groupby(['Make']).size()
```

Out[3]:

```
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
```

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 [4]:

```
cars.corr(method='pearson')
```

Out[4]:

	Cylinders	Engine Size (L)	Horsepower	Length (IN)	MPG (City)	MPG (Highway)	Weight (LBS)	W
Cylinders	1.000000	0.908002	0.810341	0.547783	-0.684402	-0.676100	0.742209	
Engine Size (L)	0.908002	1.000000	0.787435	0.637448	-0.709471	-0.717302	0.807867	
Horsepower	0.810341	0.787435	1.000000	0.381554	-0.676699	-0.647195	0.630796	
Length (IN)	0.547783	0.637448	0.381554	1.000000	-0.501526	-0.466092	0.690021	
MPG (City)	-0.684402	-0.709471	-0.676699	-0.501526	1.000000	0.941021	-0.737966	
MPG (Highway)	-0.676100	-0.717302	-0.647195	-0.466092	0.941021	1.000000	-0.790989	
Weight (LBS)	0.742209	0.807867	0.630796	0.690021	-0.737966	-0.790989	1.000000	
Wheelbase (IN)	0.546730	0.636517	0.387398	0.889195	-0.507284	-0.524661	0.760703	

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 [5]:

```
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]:

```
def HML(row):
    if row['Horsepower'] <= 200:
        return "Low"
    elif row['Horsepower'] > 200 and row['Horsepower'] <= 300:
        return "Medium"
    elif row['Horsepower'] > 300:
        return "High"

cars['HP_Groups'] = cars.apply(lambda row: HML(row), axis=1)
cars['HP_Groups']
```

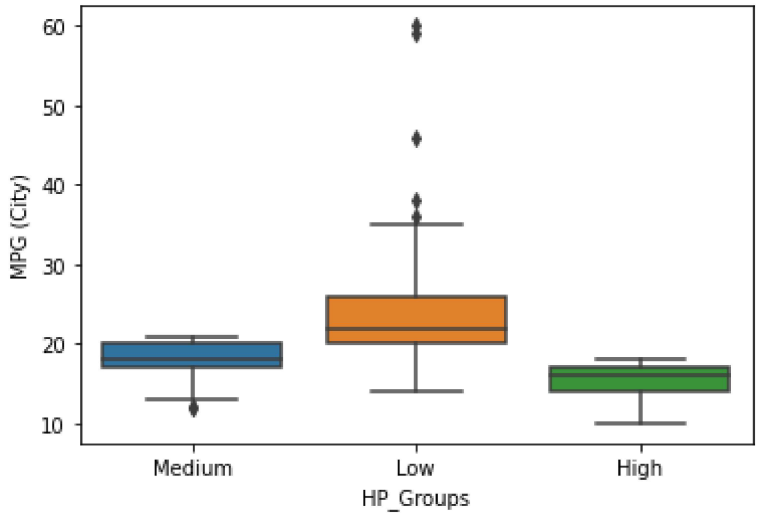
Out[6]:

```
0      Medium
1      Medium
2      Medium
3      Medium
4         Low
       ...
423    Medium
424    Medium
425       Low
426    Medium
427    Medium
Name: HP_Groups, Length: 428, dtype: object
```

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

In [7]:

```
import seaborn as sns

ax = sns.boxplot(x=cars['HP_Groups'], y=cars['MPG (City)'], data=cars)
```



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 [8]:

```
def calculate_bmi(inches, pounds):
    m = inches * 0.025
    kg = pounds * 0.453592
    return (kg/(m*m))

calculate_bmi(72,190)
```

Out[8]:

```
26.59953086419753
```

b. General standard categories for BMI are given by:

- BMI < 18.5: Underweight
- 18.5 <= BMI < 25: Normal weight
- 25 <= BMI < 30: Overweight
- BMI >= 30: 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 [9]:

```
def determine_weight_category(inches, pounds):
    m = inches * 0.025
    kg = pounds * 0.453592
    BMI = (kg/(m*m))
    if BMI < 18.5:
        return "Underweight"
    elif BMI >= 18.5 and BMI < 25:
        return "Normal weight"
    elif BMI >= 25 and BMI < 30:
        return "Overweight"
    elif BMI >= 30:
        return "Obese"

print("Height 60 / Weight 160 lbs is : " + determine_weight_category(60, 160))
print("Height 68 / Weight 160 lbs is : " + determine_weight_category(68, 160))
print("Height 72 / Weight 160 lbs is : " + determine_weight_category(72, 160))
```

```
Height 60 / Weight 160 lbs is : Obese
Height 68 / Weight 160 lbs is : Overweight
Height 72 / Weight 160 lbs is : Normal weight
```

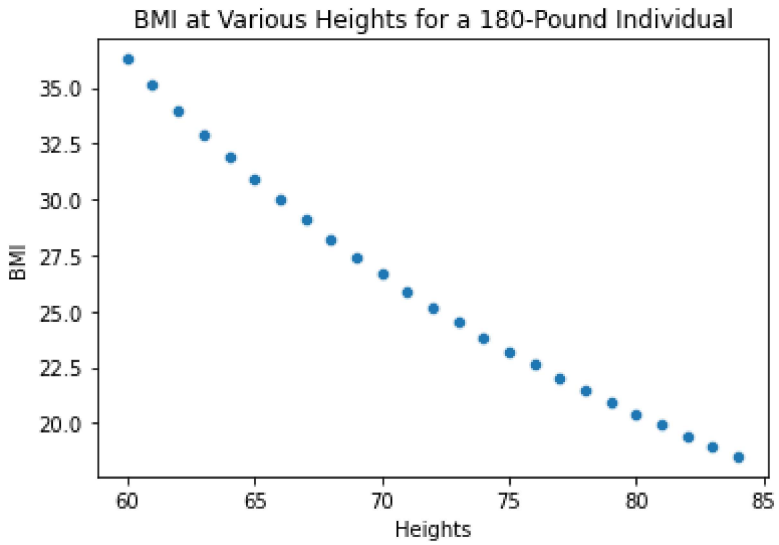
c. You want to create a plot of the BMI for a 180-pound individual at various heights from 60" to 84" (in one-inch increments). Create a vector ‘heights’ to hold the various heights from 60 to 84 and write a for-loop to call your calculate_bmi() function for each value of heights. Then, use these two 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 [10]:

```
heights = [i for i in range(60, 85)]

BMIat180 = []

for n in heights:
    BMIat180.append(calculate_bmi(n, 180))

bmi_plot = sns.scatterplot(x=heights, y=BMIat180)
bmi_plot.set_xlabel("Heights", fontsize = 10)
bmi_plot.set_ylabel("BMI", fontsize = 10)
bmi_plot.set_title("BMI at Various Heights for a 180-Pound Individual")
```

Out[10]:

```
Text(0.5, 1.0, 'BMI at Various Heights for a 180-Pound Individual')
```



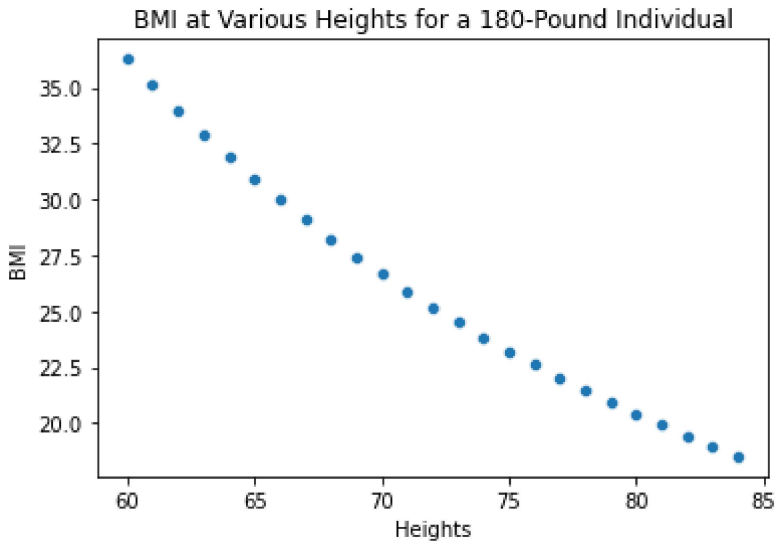
d. Repeat part c from above but without using a loop

In [11]:

```
import numpy as np
heights = [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]
m = np.array(heights)*0.025
kg = np.array([180] * len(heights))*0.453592
bmi = kg/(m*m)

bmi = sns.scatterplot(x=heights, y=bmi)
bmi.set_xlabel("Heights", fontsize = 10)
bmi.set_ylabel("BMI", fontsize = 10)
bmi.set_title("BMI at Various Heights for a 180-Pound Individual")
```

Out[11]:

```
Text(0.5, 1.0, 'BMI at Various Heights for a 180-Pound Individual')
```



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 [12]:

```
demos = pd.read_csv(r'C:\Users\Jerry\OneDrive - University of Southern California\Deskt
op\ISE_529\HW2\demographics.csv')
type(demos)
```

Out[12]:

```
pandas.core.frame.DataFrame
```

In [13]:

```
demos
```

Out[13]:

	CONT	NAME	Region	Population	Urban Pop Percentage	AdultLiteracyPct	MaleSchoolPct	Fe
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	
...	...	...	...	...	...	...	...	
192	96	NIUE	WPR	1445	0.37	NaN	0.99	
193	96	PAPUA NEW GUINEA	WPR	5887138	0.13	0.57	0.79	
194	96	TONGA	WPR	102311	0.34	0.99	1.00	
195	96	TUVALU	WPR	10441	0.57	NaN	NaN	
196	96	SAMOA	WPR	184984	0.23	0.99	0.99	

197 rows × 10 columns

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

In [14]:

```
demos.groupby(['CONT'])[['Population']].mean()
```

Out[14]:

CONT	Population
91	3.182618e+07
92	1.984816e+07
93	1.695122e+07
94	1.707178e+07
95	8.953362e+07
96	4.030014e+06

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

In [15]:

```
largestCountry = demos.groupby(['CONT'])['Population'].transform(max) == demos['Population']
```

In [16]:

```
demos[largestCountry]
```

Out[16]:

	CONT	NAME	Region	Population	Urban Pop Percentage	AdultLiteracyPct	MaleSch
15	91	UNITED STATES	AMR	298212895	0.81	NaN	
20	92	BRAZIL	AMR	186404913	0.84	0.88	
50	93	GERMANY	EUR	82689210	0.89	NaN	
114	94	NIGERIA	AFR	131529669	0.48	0.67	
142	95	CHINA	WPR	1323344591	0.41	0.91	
182	96	ASHMORE/CARTIER	WPR	20155129	0.93	NaN	
183	96	AUSTRALIA	WPR	20155129	0.93	NaN	
186	96	CORAL SEA ISLANDS	WPR	20155129	0.93	NaN	

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

In [17]:

```
demos.groupby(['CONT'])[['MaleSchoolPct' , 'FemaleSchoolPct']].mean()
```

Out[17]:

CONT	MaleSchoolPct	FemaleSchoolPct
91	0.932143	0.931429
92	0.930588	0.919412
93	0.942381	0.939286
94	0.734444	0.676222
95	0.885429	0.856000
96	0.933077	0.923077

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

In [18]:

```
demos.groupby(['CONT'])[['FemaleSchoolPct']].mean().sort_values(by=['FemaleSchoolPct'],
ascending=False)
```

Out[18]:

CONT	FemaleSchoolPct
93	0.939286
91	0.931429
96	0.923077
92	0.919412
95	0.856000
94	0.676222