Homework 1 - Solutions

ISE-529 Predictive Analytics

Student Name

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. (10 points)

In [36]:
```
import csv
import numpy as np
import pandas as pd
cars = pd.read_csv('Cars Data.csv', skiprows = 3)
cars.head(10)
```

Out[36]:

	Make	Model	DriveTrain	Origin	Type	Cylinders	Engine Size (L)	Horsepower	Invoice	Length (IN)	MPG (City)	MPG (Highway)	MSRP	Weight (LBS)	Wheelbase (IN)
0	Acura	3.5 RL 4dr	Front	Asia	Sedan	6.0	3.5	225	$39,014	197	18	24	$43,755	3880	115
1	Acura	3.5 RL w/Navigation 4dr	Front	Asia	Sedan	6.0	3.5	225	$41,100	197	18	24	$46,100	3893	115
2	Acura	MDX	All	Asia	SUV	6.0	3.5	265	$33,337	189	17	23	$36,945	4451	106
3	Acura	NSX coupe 2dr manual S	Rear	Asia	Sports	6.0	3.2	290	$79,978	174	17	24	$89,765	3153	100
4	Acura	RSX Type S 2dr	Front	Asia	Sedan	4.0	2.0	200	$21,761	172	24	31	$23,820	2778	101
5	Acura	TL 4dr	Front	Asia	Sedan	6.0	3.2	270	$30,299	186	20	28	$33,195	3575	108
6	Acura	TSX 4dr	Front	Asia	Sedan	4.0	2.4	200	$24,647	183	22	29	$26,990	3230	105
7	Audi	A4 1.8T 4dr	Front	Europe	Sedan	4.0	1.8	170	$23,508	179	22	31	$25,940	3252	104
8	Audi	A4 3.0 4dr	Front	Europe	Sedan	6.0	3.0	220	$28,846	179	20	28	$31,840	3462	104
9	Audi	A4 3.0 convertible 2dr	Front	Europe	Sedan	6.0	3.0	220	$38,325	180	20	27	$42,490	3814	105

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

In [37]:
```
cars.describe()
```

Out[37]:

	Cylinders	Engine Size (L)	Horsepower	Length (IN)	MPG (City)	MPG (Highway)	Weight (LBS)	Wheelbase (IN)
count	426.000000	428.000000	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	108.154206
std	1.558443	1.108595	71.836032	14.357991	5.238218	5.741201	758.983215	8.311813
min	3.000000	1.300000	73.000000	143.000000	10.000000	12.000000	1850.000000	89.000000
25%	4.000000	2.375000	165.000000	178.000000	17.000000	24.000000	3104.000000	103.000000
50%	6.000000	3.000000	210.000000	187.000000	19.000000	26.000000	3474.500000	107.000000
75%	6.000000	3.900000	255.000000	194.000000	21.250000	29.000000	3977.750000	112.000000
max	12.000000	8.300000	500.000000	238.000000	60.000000	66.000000	7190.000000	144.000000

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 [38]:
```
cars.groupby('Make').size()
```

Out[38]:
```
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 [39]:

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

Out[39]:

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

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

In [40]:

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

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

Out[40]:

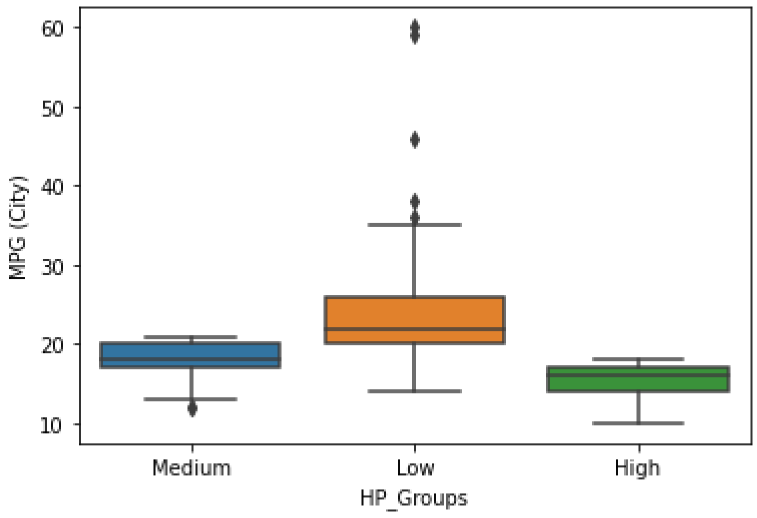
	Make	Model	DriveTrain	Origin	Type	Cylinders	Engine Size (L)	Horsepower	Invoice	Length (IN)	MPG (City)	MPG (Highway)	MSRP	Weight (LBS)	Wheelbase (IN)	H
0	Acura	3.5 RL 4dr	Front	Asia	Sedan	6.0	3.5	225	$39,014	197	18	24	$43,755	3880	115	
1	Acura	3.5 RL w/Navigation 4dr	Front	Asia	Sedan	6.0	3.5	225	$41,100	197	18	24	$46,100	3893	115	
2	Acura	MDX	All	Asia	SUV	6.0	3.5	265	$33,337	189	17	23	$36,945	4451	106	
3	Acura	NSX coupe 2dr manual S	Rear	Asia	Sports	6.0	3.2	290	$79,978	174	17	24	$89,765	3153	100	
4	Acura	RSX Type S 2dr	Front	Asia	Sedan	4.0	2.0	200	$21,761	172	24	31	$23,820	2778	101	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
423	Volvo	S80 2.9 4dr	Front	Europe	Sedan	6.0	2.9	208	$35,542	190	20	28	$37,730	3576	110	
424	Volvo	S80 T6 4dr	Front	Europe	Sedan	6.0	2.9	268	$42,573	190	19	26	$45,210	3653	110	
425	Volvo	V40	Front	Europe	Wagon	4.0	1.9	170	$24,641	180	22	29	$26,135	2822	101	
426	Volvo	XC70	All	Europe	Wagon	5.0	2.5	208	$33,112	186	20	27	$35,145	3823	109	
427	Volvo	XC90 T6	All	Europe	SUV	6.0	2.9	268	$38,851	189	15	20	$41,250	4638	113	

428 rows × 16 columns

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

In [41]:
```
import seaborn as sns
sns.boxplot(x="HP_Groups", y="MPG (City)", data=cars)
```

Out[41]: <AxesSubplot:xlabel='HP_Groups', ylabel='MPG (City)'>



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 [42]:
```
def calculate_bmi(height_in, weight_lbs):
    height_m = height_in * 0.025
    weight_kg = weight_lbs * 0.453592
    return weight_kg/height_m**2

calculate_bmi(72, 190)
```

Out[42]: 26.59953086419753

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 [43]:
```
def determine_weight_category(height_in, weight_lbs):
    bmi = calculate_bmi(height_in, weight_lbs)
    if bmi <= 18.5:
        return 'Underweight'
    elif bmi < 24.9:
        return 'Normal Weight'
    elif bmi < 29.9:
        return 'Overweight'
    else:
        return 'Obese'

print(determine_weight_category(60,160))
print(determine_weight_category(68,160))
print(determine_weight_category(72,160))
```

```
Obese
Overweight
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.

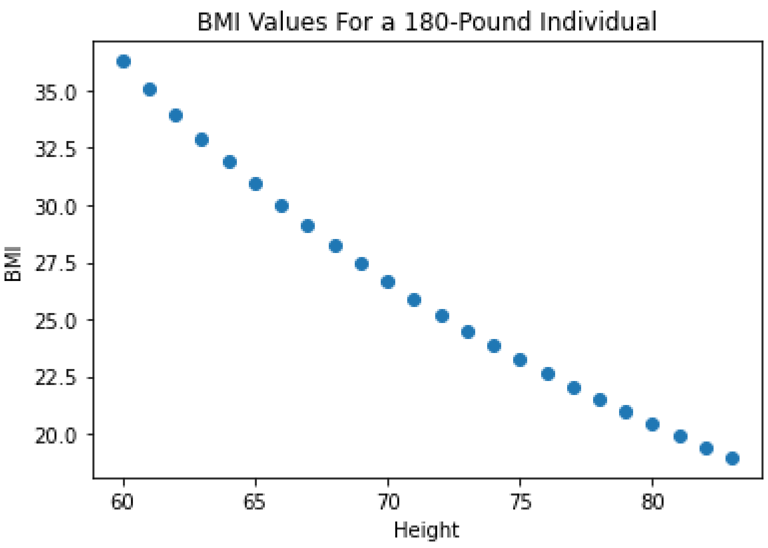
In [44]:
```
import numpy as np
import matplotlib.pyplot as plt
heights = np.arange(start=60, stop=84, step=1)
```

```
bmi_values = np.zeros(24)

for i in range(24):
    bmi_values[i] = calculate_bmi(heights[i], 180)

plt.scatter(heights, bmi_values)
plt.xlabel("Height")
plt.ylabel("BMI")
plt.title("BMI Values For a 180-Pound Individual")
plt.show
```

Out[44]: <function matplotlib.pyplot.show(close=None, block=None)>



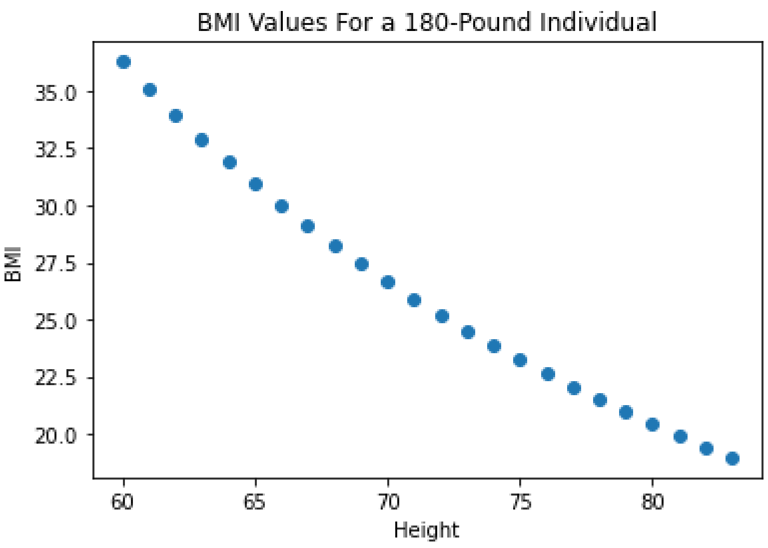
d. Repeat part c from above but without using a loop

In [45]:
```
heights = np.arange(start=60, stop=84, step=1)
bmi_values = np.zeros(24)

bmi_values = calculate_bmi(heights, 180)

plt.scatter(heights, bmi_values)
plt.xlabel("Height")
plt.ylabel("BMI")
plt.title("BMI Values For a 180-Pound Individual")
plt.show
```

Out[45]: <function matplotlib.pyplot.show(close=None, block=None)>



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 [46]:
```
demos = read_csv("demographics.csv")
type(demos)
```

Out[46]: pandas.core.frame.DataFrame

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

In [47]:
```
demos.groupby("CONT")['Population'].mean()
```

Out[47]:
```
CONT
91    3.182618e+07
92    1.984816e+07
93    1.695122e+07
94    1.707178e+07
95    8.953362e+07
96    4.030014e+06
Name: Population, dtype: float64
```

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

In [48]:
```
demos.groupby("CONT")['Population'].max()
```

Out[48]:
```
CONT
91     298212895
92     186404913
93      82689210
94     131529669
95    1323344591
96      20155129
Name: Population, dtype: int64
```

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

In [49]:
```
demos.groupby("CONT")[['FemaleSchoolPct', 'MaleSchoolPct']].mean()
```

Out[49]:

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

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

In [50]:
```
demos.groupby("CONT")[['FemaleSchoolPct', 'MaleSchoolPct']].mean().sort_values("FemaleSchoolPct", ascending = False)
```

Out[50]:

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