In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
bmx = pd.read_sas('BMX_I.XPT')
```

In [3]:
```
bmx.head()
```

Out[3]:

	SEQN	BMDSTATS	BMXWT	BMIWT	BMXRECUM	BMIRECUM	BMXHEAD	BMIHEAD
0	83732.0	1.0	94.8	NaN	NaN	NaN	NaN	NaN
1	83733.0	1.0	90.4	NaN	NaN	NaN	NaN	NaN
2	83734.0	1.0	83.4	NaN	NaN	NaN	NaN	NaN
3	83735.0	1.0	109.8	NaN	NaN	NaN	NaN	NaN
4	83736.0	3.0	55.2	NaN	NaN	NaN	NaN	NaN

5 rows × 26 columns

In [4]:
```
bmx.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9544 entries, 0 to 9543
Data columns (total 26 columns):
SEQN        9544 non-null float64
BMDSTATS    9544 non-null float64
BMXWT       9445 non-null float64
BMIWT       443 non-null float64
BMXRECUM    1073 non-null float64
BMIRECUM    33 non-null float64
BMXHEAD     215 non-null float64
BMIHEAD     0 non-null float64
BMXHT       8769 non-null float64
BMIHT       105 non-null float64
BMXBMI      8756 non-null float64
BMDBMIC     3340 non-null float64
BMXLEG      7110 non-null float64
BMILEG      402 non-null float64
BMXARML     8976 non-null float64
BMIARML     420 non-null float64
BMXARMC     8976 non-null float64
BMIARMC     421 non-null float64
BMXWAIST    8313 non-null float64
BMIWAIST    489 non-null float64
BMXSAD1     6983 non-null float64
BMXSAD2     6983 non-null float64
BMXSAD3     353 non-null float64
BMXSAD4     353 non-null float64
BMDAVSAD    6983 non-null float64
BMDSADCM    446 non-null float64
dtypes: float64(26)
memory usage: 1.9 MB
```

In [5]:
```
demo = pd.read_sas('DEMO_I.XPT')
```

In [6]:
```
demo.head()
```

Out[6]:

	SEQN	SDDSRVYR	RIDSTATR	RIAGENDR	RIDAGEYR	RIDAGEMN	RIDRETH1	RID
0	83732.0	9.0	2.0	1.0	62.0	NaN	3.0	3.0
1	83733.0	9.0	2.0	1.0	53.0	NaN	3.0	3.0
2	83734.0	9.0	2.0	1.0	78.0	NaN	3.0	3.0
3	83735.0	9.0	2.0	2.0	56.0	NaN	3.0	3.0
4	83736.0	9.0	2.0	2.0	42.0	NaN	4.0	4.0

5 rows × 47 columns

In [7]:
```
demo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9971 entries, 0 to 9970
Data columns (total 47 columns):
SEQN        9971 non-null float64
SDDSRVYR    9971 non-null float64
RIDSTATR    9971 non-null float64
RIAGENDR    9971 non-null float64
RIDAGEYR    9971 non-null float64
RIDAGEMN    695 non-null float64
RIDRETH1    9971 non-null float64
RIDRETH3    9971 non-null float64
RIDEXMON    9544 non-null float64
RIDEXAGM    4060 non-null float64
DMQMILIZ    6149 non-null float64
DMQADFC     527 non-null float64
DMDBORN4    9971 non-null float64
DMDCITZN    9969 non-null float64
DMDYRSUS    2236 non-null float64
DMDEDUC3    2647 non-null float64
DMDEDUC2    5719 non-null float64
DMDMARTL    5719 non-null float64
RIDEXPRG    1288 non-null float64
SIALANG     9971 non-null float64
SIAPROXY    9970 non-null float64
SIAINTRP    9971 non-null float64
FIALANG     9642 non-null float64
FIAPROXY    9642 non-null float64
FIAINTRP    9642 non-null float64
MIALANG     6977 non-null float64
MIAPROXY    6978 non-null float64
MIAINTRP    6978 non-null float64
AIALANGA    5962 non-null float64
DMDHHSIZ    9971 non-null float64
DMDFMSIZ    9971 non-null float64
DMDHHSZA    9971 non-null float64
DMDHHSZB    9971 non-null float64
DMDHHSZE    9971 non-null float64
DMDHRGND    9971 non-null float64
DMDHRAGE    9971 non-null float64
DMDHRBR4    9575 non-null float64
DMDHREDU    9575 non-null float64
DMDHRMAR    9909 non-null float64
DMDHSEDU    5226 non-null float64
WTINT2YR    9971 non-null float64
WTMEC2YR    9971 non-null float64
SDMVPSU     9971 non-null float64
SDMVSTRA    9971 non-null float64
INDHHIN2    9626 non-null float64
INDFMIN2    9642 non-null float64
INDFMPIR    8919 non-null float64
dtypes: float64(47)
memory usage: 3.6 MB
```

In [8]:
```
merged = bmx.merge(right = demo, on = 'SEQN')
```

In [9]:
```
#Question 1: Baby Weights
```

In [10]:
```
#Calculate and display the mean weight of baby boys for each month, from month
 0 to 12.
#You’ll produce 13 values, one for each month.
```

In [11]:
```
babies = merged.loc[merged['RIDAGEMN'].isnull() == False]
babies = babies.loc[babies.RIDAGEMN <= 12]
x = babies.loc[babies['DMDHRGND'] == 1,:].groupby(["DMDHRGND", "RIDAGEMN"])["B
MXWT"].mean()

x = pd.DataFrame(x)
x1 = x.melt()
print(x)
```

```
                           BMXWT
DMDHRGND RIDAGEMN
1.0      5.397605e-79   4.841176
         1.000000e+00   5.694737
         2.000000e+00   6.429412
         3.000000e+00   6.793750
         4.000000e+00   7.650000
         5.000000e+00   8.716667
         6.000000e+00   7.943750
         7.000000e+00   9.150000
         8.000000e+00   8.650000
         9.000000e+00   9.808333
         1.000000e+01   9.858333
         1.100000e+01   9.888889
         1.200000e+01  10.644444
```

In [12]:
```
# Calculate and display the mean weight of baby girls for each month, from mon
th 0 to 12.
y = babies.loc[babies['DMDHRGND'] == 2,:].groupby(["DMDHRGND", "RIDAGEMN"])["B
MXWT"].mean()

y = pd.DataFrame(y)
y1 = y.melt()
print(y)
```

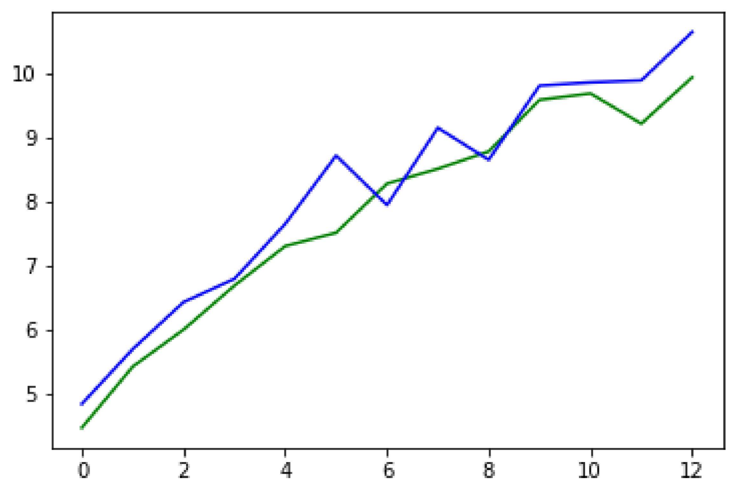
```
                          BMXWT
DMDHRGND RIDAGEMN
2.0      5.397605e-79  4.469231
         1.000000e+00  5.421429
         2.000000e+00  6.000000
         3.000000e+00  6.687500
         4.000000e+00  7.304762
         5.000000e+00  7.511765
         6.000000e+00  8.278571
         7.000000e+00  8.510526
         8.000000e+00  8.780952
         9.000000e+00  9.585000
         1.000000e+01  9.685714
         1.100000e+01  9.214286
         1.200000e+01  9.936364
```

In [13]:
```
#Calculate and display the difference between the mean weights of boys and gir
ls for each month.
print(x.values - y.values)
```

```
[[ 0.3719457 ]
 [ 0.27330827]
 [ 0.42941176]
 [ 0.10625   ]
 [ 0.3452381 ]
 [ 1.20490196]
 [-0.33482143]
 [ 0.63947368]
 [-0.13095238]
 [ 0.22333333]
 [ 0.17261905]
 [ 0.67460317]
 [ 0.70808081]]
```

In [14]:
```
#Make a line plot showing two lines: one for boys' mean weights months 0-12, a
nd one for girls' mean weights 0-12 (in a different color). The month will go
 on the x-axis, and the mean weight will go on the y-axis.
```

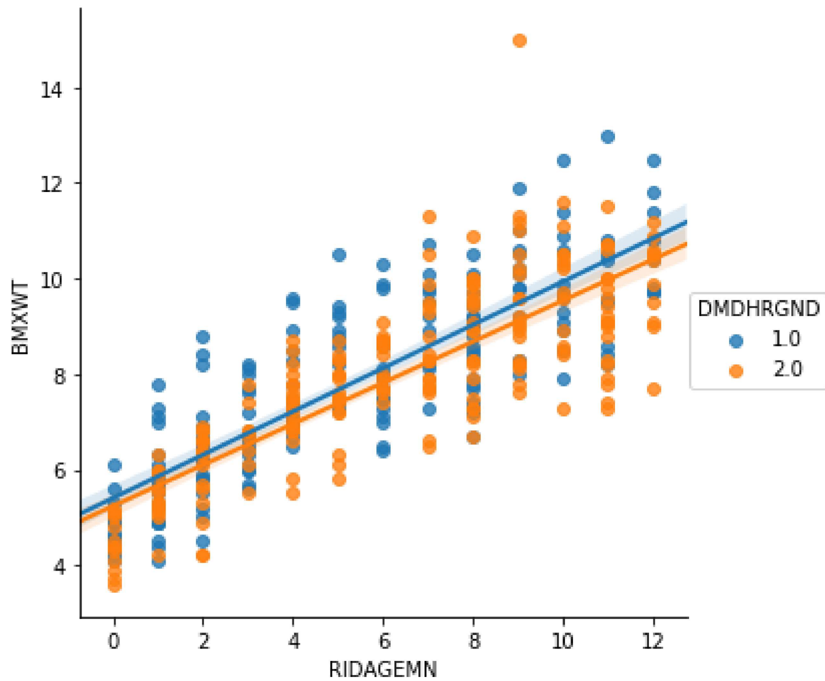
In [15]:
```
plt.plot(range(13),y['BMXWT'],'green')
plt.plot(range(13),x['BMXWT'], 'blue')
plt.show()
```



In [16]:
```
#Make a scatterplot with linear regression lines for the baby boys'
#relationship between age and wt, and the baby girls' relationship between age
 and wt
```

In [17]:
```
sns.lmplot(x="RIDAGEMN", y="BMXWT", data=babies, hue = "DMDHRGND")
```

Out[17]:
```
<seaborn.axisgrid.FacetGrid at 0x1a175dab9e8>
```



In [18]:
```
# Based on our plots and output, we can say that baby boys do weigh more than
 baby girls on average.
# First, the subtraction table between the baby boys' average with the baby gi
rls' showed up as positive for all the months except two meaning that the aver
age weight of baby boys was greater than baby girls for the majority of the ag
e.
# The two months where the baby girls' average was higher only had a slight di
fference.
# Also, the linear regression plot shows a higher slope for baby boys, and the
refore, we can conclude that baby boys' average will increase at a more higher
 rate than baby girsls' as they age, meaning the average of baby boys will be
 higher as age goes up.
# This is also shown in the plot with the two lines, where the baby boy's aver
age is always above the baby girls' average except for two months, month = 6,
8.
```

In [19]:
```
#Question 2: Height vs leg length vs arm length

#For all adults aged 20 and up, who have all three measurements of height, upp
er leg length, and upper arm length
two = merged.loc[merged['RIDAGEYR']>= 20,:]
two = two.loc[two['BMXLEG'].isnull() == False]
two = two.loc[two['BMXHT'].isnull() == False]
two = two.loc[two['BMXARML'].isnull() == False]
```

In [20]:
```
#corr standing height, upper leg
print(two['BMXHT'].corr(two['BMXLEG']))
```

```
0.7874491670325252
```

In [21]:
```
#corr standing height, upper arm
print(two['BMXARML'].corr(two['BMXHT']))
```

```
0.797872682253647
```

In [22]:
```
#corr upper leg, upper arm
print(two['BMXARML'].corr(two['BMXLEG']))
```

```
0.6293694196844941
```

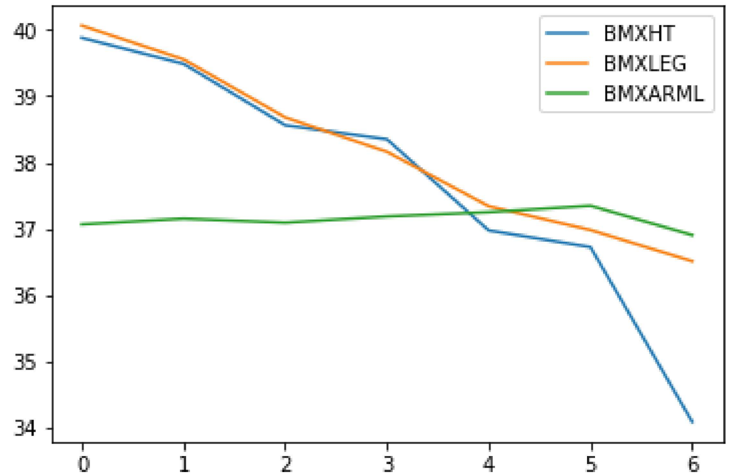
In [23]:
```
#Make adult age groups by decade: i.e. adults aged 20-29.9, adults aged 30-39.
9, … adults aged
#70-79.9, adults aged 80+ (7 groups total). (not required, but recommended: us
e pandas.cut …
#this function was not explicitly covered in the notes, but you should be able
 to read the
#documentation to learn function usage. If you use this, set option: right=Fal
se)

two['range'] = pd.cut(two.RIDAGEYR, [20.00,29.99,39.99,49.99,59.99,69.99,79.99
,300], right = False)
```

In [24]:
```
#[10 pts] For each age group, calculate the mean of the three values.
a = two.groupby('range')["BMXHT"].mean().reset_index()
b = two.groupby('range')["BMXLEG"].mean().reset_index()
c = two.groupby('range')["BMXARML"].mean().reset_index()

a = pd.DataFrame(a)
b = pd.DataFrame(b)
c = pd.DataFrame(c)

print(a)
print(b)
print(c)
```

```
            range       BMXHT
0   [20.0, 29.99)  167.879617
1  [29.99, 39.99)  167.488202
2  [39.99, 49.99)  166.563244
3  [49.99, 59.99)  166.351476
4  [59.99, 69.99)  164.977480
5  [69.99, 79.99)  164.728016
6  [79.99, 300.0)  162.097091
            range     BMXLEG
0   [20.0, 29.99)  40.064189
1  [29.99, 39.99)  39.556128
2  [39.99, 49.99)  38.680863
3  [49.99, 59.99)  38.163400
4  [59.99, 69.99)  37.342707
5  [69.99, 79.99)  36.984049
6  [79.99, 300.0)  36.516000
            range    BMXARML
0   [20.0, 29.99)  37.068919
1  [29.99, 39.99)  37.154754
2  [39.99, 49.99)  37.094632
3  [49.99, 59.99)  37.188784
4  [59.99, 69.99)  37.252859
5  [69.99, 79.99)  37.350102
6  [79.99, 300.0)  36.908000
```

In [25]:
```
a['BMXHT'] = a['BMXHT'] - 128
```

In [26]:
```
#We want to make a plot with three lines showing the relationship between the
 age
#decade and the mean height, leg length, and arm length.

plt.plot(range(7), a['BMXHT'])
plt.plot(range(7), b['BMXLEG'])
plt.plot(range(7), c['BMXARML'])
plt.legend()
plt.show()
```



In [27]:
```
#Comment on what you think the data says about the
#guiding question. Include any additional analysis you deem appropriate

#We can see from the mean summary that height and leg length decrease with age
 whereas arm length stays almost consistent throughout.
#This is probably because as you age, your posture gets worse and therefore th
e leg length and height decreases.
#We can also see that there is a high correlation between height and both arml
ength and leglength.
#However, the correlation between arm length and leg length is less than the a
bove correlation.
#Looking at the plot we could see height and leg length drastically decreasing
 with age, whereas the arm length doesn't change at all.
```

In [28]:
```
#Question 3: Education Level and income [25pts]
#Use the variable: INDHHIN2 for household income. Use DMDHREDU for the educati
on level of
#the head of the household.
```

In [29]:
```
# Filter to adults aged 20 and older.

adults = merged.loc[merged['DMDHRAGE'] >= 20]
print(adults.shape)
```

```
(9487, 72)
```

In [30]:
```
#Remove people who are missing, refused to answer, or didn’t know the househol
d income or education levels.
```

In [31]:
```
adults = adults.loc[adults['INDHHIN2'].isnull() == False]
print(adults.shape)
adults = adults.loc[adults['DMDHREDU'].isnull() == False]
print(adults.shape)
adults = adults.loc[adults['INDHHIN2'] <= 15]
print(adults.shape)
adults = adults.loc[adults['DMDHREDU'] <= 5]

print(adults.shape)
```

```
(9210, 72)
(8974, 72)
(8665, 72)
(8640, 72)
```

In [32]:
```
#Remove household income categories 12 ($20,000 and over) and 13 (under $20,00
0) as they
#don’t quite fit in with the other income categories
```

In [33]:
```
adults = adults.loc[adults['INDHHIN2'] != 12]
adults = adults.loc[adults['INDHHIN2'] != 13]
```

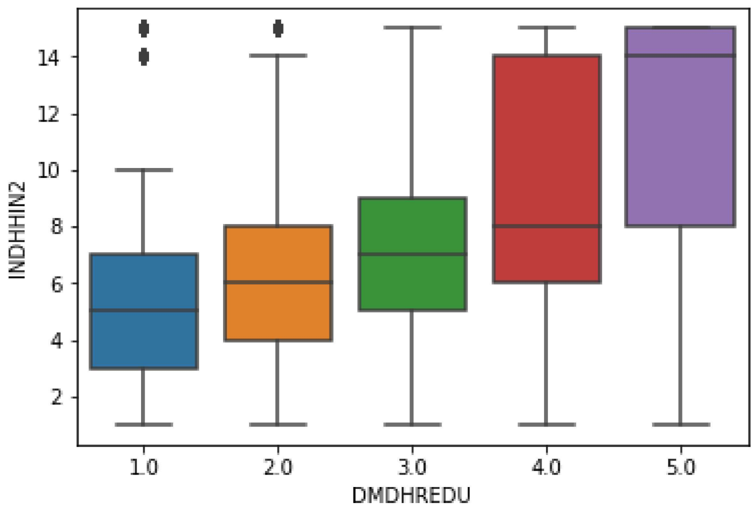
In [34]:
```
# Print the shape of the resulting data before the next step.
print(adults.shape)
```

```
(8262, 72)
```

In [35]:
```
# Summarize and display the data to explore the relationship between education
 level
#and income. This question is purposely open-ended. You choose how best to sum
marize and
#display the data to answer the question regarding education level and househo
ld income.
```

In [36]:
```
sns.boxplot(x="DMDHREDU",y = "INDHHIN2", data = adults)
adults.groupby("DMDHREDU")["INDHHIN2"].mean()
```

Out[36]:
```
DMDHREDU
1.0     5.648438
2.0     6.262626
3.0     7.464968
4.0     8.619637
5.0    11.646460
Name: INDHHIN2, dtype: float64
```



In [37]:
```
#Looking at the result and looking at the outputted graph, we can see that the mean household income is linearly proportional to educational level.
#This means that, by average, households with higher education level will result in a higher income level.
#We see that from education from 1 to 4, the difference of means per education level does not significantly increase the income level as the mean income level increases by about 1.
#However, from education level 5, the mean income level increases signifcantly from the previous education level, by about 3 points.
#This level 5 corresponds to college graduate, and we can therefore make an assumption that college graduates make signficantly more money than non-college graduates.
#Although there is a increasing rate of income by education level, this increasing rate does not seem significant until the last education level = college graduate.
```