SI_618_Homework_01

September 13, 2023

1 SI 618 - Homework #1: Data Manipulation

Version 2023.09.06.01.CT (Fall 2023)

1.1 Background

This homework assignment focuses on the analysis of biometric data and exercise performance.

Your main task in this assignment is to explore the data *using the data manipulation methods we covered in class*, as well as those in the assigned readings and video resources. You may need to consult pandas documentation, Stack Overflow, Copilot, or other online resources.

A total of 100 points is available in this homework assignment, broken down as shown beside each question and in the accompanying rubric in Canvas. Please note that to receive full points, code should conform to PEP 8 guidelines, and written responses should be grammatically correct, free of spelling errors, and generally follow the Strunk & White guidelines. You should review those guidelines before proceding with the assignment.

1.2 Download the data from:

https://www.kaggle.com/kukuroo3/body-performance-data

[ ]:
```
MY_UNIQNAME = 'yanlunar' # please fill in your uniqname
```

1.3 Answer the following questions.

Points for each question are shown below.

For each question, you should 1. Write code using Python and pandas that can help you answer the following questions, and 2. Explain your answers in plain English. You should use complete sentences that would be understood by an educated professional who is not necessarily a data scientist (like a product manager). When we ask for an explanation of your answer, we are interested in your interpretation of the analyses that you produce. With the exception of Question 1a, we are not interested in a description of the steps you took to arrive at your answer.

1.3.1 Q1: 8 pt(total):

Q1a (3pt): Write out the steps you need to follow in order to describe the dataset, in terms of size, number of rows, and number of columns. Replace this with your answer. Remember that markdown supports bulleted lists: * this * that * the other thing

and numbered lists: 1. one 1. two 1. three

Q1b (5pt): Write and execute the code that will do the steps you identified in Part (a) of this question.

[ ]:
```
# import libraries
import pandas as pd

# read in the data
df = pd.read_csv('bodyPerformance.csv')

# describe the dataset
print("The size of the dataset is:", df.shape)
print("The number of rows of the dataset is:", len(df))
print("The number of columns of the dataset is:", len(df.columns))
```

```
The size of the dataset is: (13393, 12)
The number of rows of the dataset is: 13393
The number of columns of the dataset is: 12
```

[ ]:
```
# print the first 5 rows of the dataset
df.head(5)
```

[ ]:

	age	gender	height_cm	weight_kg	body fat_%	diastolic	systolic	\
0	27.0	M	172.3	75.24	21.3	80.0	130.0	
1	25.0	M	165.0	55.80	15.7	77.0	126.0	
2	31.0	M	179.6	78.00	20.1	92.0	152.0	
3	32.0	M	174.5	71.10	18.4	76.0	147.0	
4	28.0	M	173.8	67.70	17.1	70.0	127.0	

	gripForce	sit and bend forward_cm	sit-ups counts	broad jump_cm	class
0	54.9	18.4	60.0	217.0	C
1	36.4	16.3	53.0	229.0	A
2	44.8	12.0	49.0	181.0	C
3	41.4	15.2	53.0	219.0	B
4	43.5	27.1	45.0	217.0	B

[ ]:
```
df.describe().round(1)
```

[ ]:

	age	height_cm	weight_kg	body fat_%	diastolic	systolic	\
count	13393.0	13393.0	13393.0	13393.0	13393.0	13393.0	
mean	36.8	168.6	67.4	23.2	78.8	130.2	
std	13.6	8.4	11.9	7.3	10.7	14.7	
min	21.0	125.0	26.3	3.0	0.0	0.0	
25%	25.0	162.4	58.2	18.0	71.0	120.0	
50%	32.0	169.2	67.4	22.8	79.0	130.0	
75%	48.0	174.8	75.3	28.0	86.0	141.0	
max	64.0	193.8	138.1	78.4	156.2	201.0	

	gripForce	sit and bend forward_cm	sit-ups counts	broad jump_cm
count	13393.0	13393.0	13393.0	13393.0

mean	37.0	15.2	39.8	190.1
std	10.6	8.5	14.3	39.9
min	0.0	-25.0	0.0	0.0
25%	27.5	10.9	30.0	162.0
50%	37.9	16.2	41.0	193.0
75%	45.2	20.7	50.0	221.0
max	70.5	213.0	80.0	303.0

My Explanation

- **Size**: The dataset consists 13,393 rows, i.e. 13,393 participants,which is quite large; it has 12 columns, which gives much exploration content
- **Describe**: This dataset includes some data related to human body, e.g. age, height, body fat, diatolic, etc
- **Extreme data**: By comparing the mean and 50% value, which are close to each other, we can infer that there aren't much extreme data
- **Comparison between average and median**: It is obvious that for all columns, the average is larger than the median, which means that the dataset has very small value in each column

1.3.2 Q2: 5 pt:

Without using pd.DataFrame.describe(), report the median age, height, and weight for people in the data. Round your answer to the nearest whole number. Be sure to indicate units for each variable in your write-up.

[ ]:
```
print("median age: ", int(df['age'].median().round()), "years old")
print("median height: ", int(df['height_cm'].median().round()), "cm")
print("median weight: ", int(df['weight_kg'].median().round()), "kg")
```

```
median age:  32 years old
median height:  169 cm
median weight:  67 kg
```

My Explanation

- The median age is 32 years old, larger than the average, which means some people in the dataset has very yound age
- The median height is 169cm, larger than the average, which means some people in the dataset has very short height
- The median weight is 67kg, larger than the average, which means some people in the dataset has very light weight

1.3.3 Q3: 5 pt:

Look for any anomalous data points. For example, are there cases where diastolic blood pressure is higher than systolic blood pressure (which is physiologically impossible)? Do you notice any other anomalies?

[ ]:
```
df[df['diastolic'] > df['systolic']]
```

[ ]:

	age	gender	height_cm	weight_kg	body fat_%	diastolic	systolic	\
7495	30.0	F	156.2	52.80	28.2	156.2	104.0	
7597	30.0	M	166.6	66.08	14.4	67.0	14.0	
8217	31.0	M	176.7	80.32	20.8	46.2	43.9	
12949	24.0	M	171.0	78.70	20.0	99.0	95.0	

	gripForce	sit and bend forward_cm	sit-ups counts	broad jump_cm	class
7495	21.7	27.2	43.0	162.0	C
7597	53.8	17.2	57.0	221.0	A
8217	43.9	17.2	57.0	221.0	B
12949	46.5	12.8	50.0	197.0	D

[ ]:
```
df.sort_values('diastolic').head(10)
```

[ ]:

	age	gender	height_cm	weight_kg	body fat_%	diastolic	systolic	\
10624	26.0	F	160.0	63.56	32.0	0.0	0.0	
3524	44.0	M	177.0	78.28	22.1	6.0	145.0	
336	35.0	M	173.1	77.60	22.8	8.0	122.0	
6786	22.0	F	164.1	60.80	32.4	30.0	122.0	
344	22.0	F	158.5	49.30	27.0	37.0	77.0	
6234	59.0	F	152.1	50.60	35.5	40.0	105.0	
12698	25.0	M	174.4	77.00	12.5	41.0	97.0	
8560	22.0	M	173.4	76.00	16.5	41.0	98.0	
11205	21.0	M	175.1	71.40	18.4	42.0	93.0	
11121	22.0	M	170.0	58.20	9.4	42.0	113.0	

	gripForce	sit and bend forward_cm	sit-ups counts	broad jump_cm	class
10624	23.7	17.1	23.0	128.0	D
3524	47.4	10.5	42.0	215.0	B
336	46.4	16.7	47.0	216.0	B
6786	29.7	-4.6	31.0	159.0	D
344	27.1	21.0	59.0	178.0	B
6234	22.8	15.1	9.0	110.0	C
12698	54.0	18.9	57.0	252.0	A
8560	45.0	9.8	56.0	215.0	C
11205	39.6	28.6	57.0	241.0	A
11121	38.3	8.2	53.0	186.0	C

[ ]:
```
df.sort_values('systolic').head(10)
```

[ ]:

	age	gender	height_cm	weight_kg	body fat_%	diastolic	systolic	\
10624	26.0	F	160.0	63.56	32.0	0.0	0.0	
7597	30.0	M	166.6	66.08	14.4	67.0	14.0	
8217	31.0	M	176.7	80.32	20.8	46.2	43.9	
344	22.0	F	158.5	49.30	27.0	37.0	77.0	
7341	50.0	F	161.5	52.00	17.9	42.0	82.0	
9440	21.0	F	155.9	42.00	21.9	60.0	84.0	

```
4412   26.0      F      158.1      62.80        39.8       60.0      86.0
12473  27.0      F      157.0      49.10        30.7       70.0      86.0
8965   27.0      F      157.0      49.10        30.7       70.0      86.0
9549   23.0      F      167.0      52.10        15.3       49.0      88.0
```

	gripForce	sit and bend forward_cm	sit-ups counts	broad jump_cm	class
10624	23.7	17.1	23.0	128.0	D
7597	53.8	17.2	57.0	221.0	A
8217	43.9	17.2	57.0	221.0	B
344	27.1	21.0	59.0	178.0	B
7341	24.5	30.2	25.0	163.0	A
9440	19.9	17.2	41.0	177.0	B
4412	18.9	18.0	25.0	139.0	D
12473	27.7	19.7	51.0	167.0	A
8965	27.7	19.7	51.0	167.0	A
9549	29.1	19.9	41.0	175.0	A

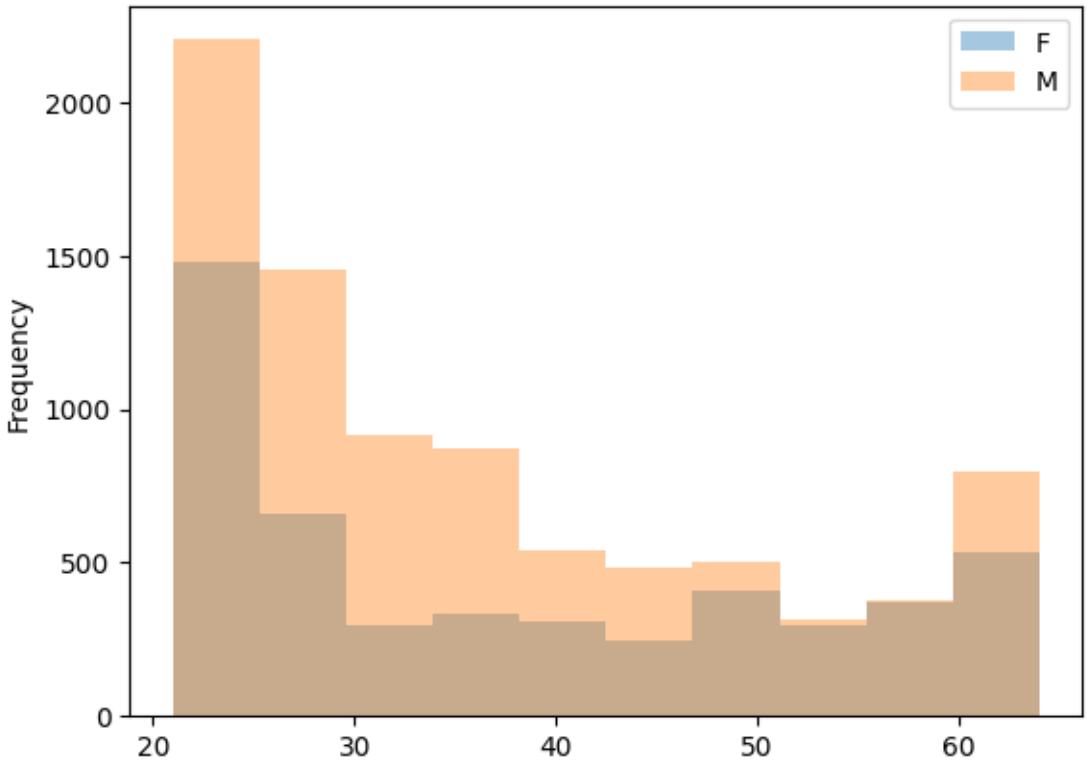
My Explanation Anomalous data finding: - It is found that there are 4 cases where the diastolic pressure is larger than the systolic pressure, which is physiologically impossble - It is found that some diastolic and systolic pressure values reaches 0, which is also impossible

1.3.4 Q4: 11 pt:

Generate histogram plots for the age, weight, and height for that allow you to visualize differences between gender M and gender F participants. Describe the plots and point out any interesting aspects. You are not permitted to use Seaborn, Bokeh, or Plotly to create your visualizations (i.e. you can only use the matplotlib-pandas integrations).

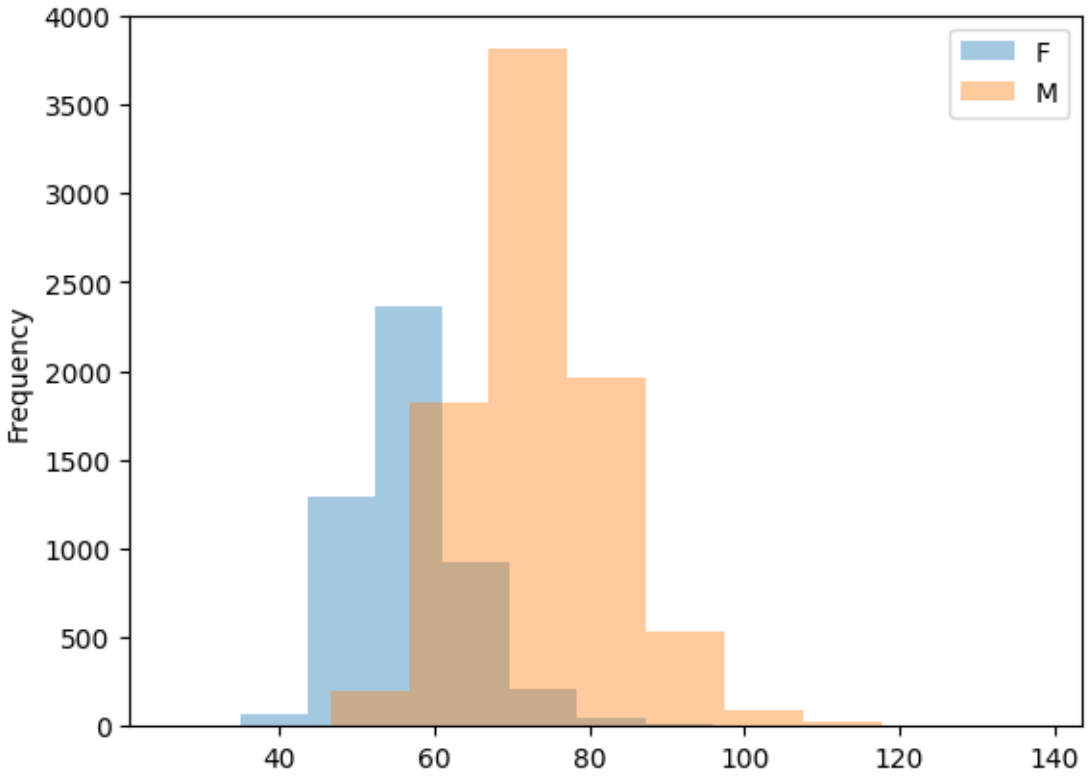
[ ]:
```
df.groupby('gender')['age'].plot.hist(bins=10, alpha=0.4, legend=True)
```

[ ]:
```
gender
F    Axes(0.125,0.11;0.775x0.77)
M    Axes(0.125,0.11;0.775x0.77)
Name: age, dtype: object
```



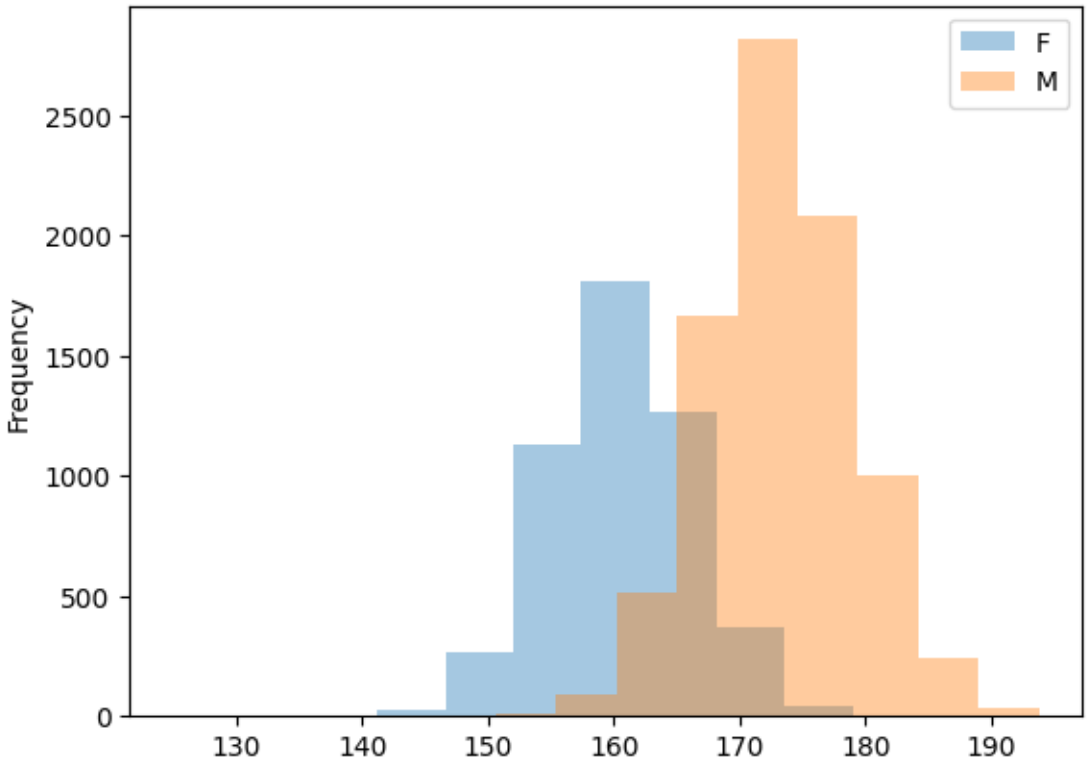
[ ]:
```
df.groupby('gender')['weight_kg'].plot.hist(bins=10, alpha=0.4, legend=True)
```

[ ]:
```
gender
F    Axes(0.125,0.11;0.775x0.77)
M    Axes(0.125,0.11;0.775x0.77)
Name: weight_kg, dtype: object
```



[ ]:
```
df.groupby('gender')['height_cm'].plot.hist(bins=10, alpha=0.4, legend=True)
```

[ ]:
```
gender
F    Axes(0.125,0.11;0.775x0.77)
M    Axes(0.125,0.11;0.775x0.77)
Name: height_cm, dtype: object
```



My Explanation

- **Age**: The age distribution for female is more average, while male are generally younger
- **Weight**:The weight for female is generally lighter than male
- **Height**: The height for male is generally higher than female

1.3.5 Q5: 16 pt:

Blood pressure is reported as two numbers: diastolic (the lower number) and systolic (the higher number). So, for example, if a person has a diastolic measurement of 80 and a systolic measurement of 120, the person's blood pressure would be reported as 120/80.

Blood pressure measurements fall into several categories:

Normal blood pressure. Your blood pressure is normal if it's below 120/80 mm Hg.

Elevated blood pressure. Elevated blood pressure is a systolic pressure ranging from 120 to 129 mm Hg and a diastolic pressure below (not above) 80 mm Hg. Elevated blood pressure may also be called prehypertension.

Stage 1 hypertension. Stage 1 hypertension is a systolic pressure ranging from 130 to 139 mm Hg or a diastolic pressure ranging from 80 to 89 mm Hg.

Stage 2 hypertension. More-severe hypertension, stage 2 hypertension is a systolic pressure of 140 mm Hg or higher or a diastolic pressure of 90 mm Hg or higher.

Hypertensive crisis. A blood pressure measurement higher than 180/120 mm Hg is an emergency situation that requires urgent medical care.

a) Report the number of people in each of those categories.

b) The guidelines were recently changed: prior to 2017, the guidelines set the threshold at 140/90 mm Hg for people younger than age 65 and 150/80 mm Hg for those ages 65 and older to be diagnosed with hypertension. Create and implement analyses that allow you to demonstrate how the change in guidelines affects the categorization of people into the different categories. Make reasonable assumptions (and state them) about whether to use diastolic, systolic, both, or either of those measures when implementing your analysis. Note that the new guidelines offer more categories than the older guidelines.

Part a)

[ ]:
```
print("Normal blood pressure:", len(df[(df['systolic'] < 120) &␣
 ↪(df['diastolic'] < 80)]))
print("Elevated blood pressure (prehypertension):", len(df[(df['systolic'] >=␣
 ↪120) & (df['systolic'] <= 129) & (df['diastolic'] < 80)]))
print("Stage 1 hypertension:", len(df[((df['systolic'] >= 130) &␣
 ↪(df['systolic'] <= 139)) | ((df['diastolic'] >= 80) & (df['diastolic'] <=␣
 ↪89))]))
print("Stage 2 hypertension:", len(df[(df['systolic'] >= 140) |␣
 ↪((df['diastolic'] >= 90))]))
print("Hypertensive crisis:", len(df[(df['systolic'] > 180) & (df['diastolic']␣
 ↪> 120)]))
```

```
Normal blood pressure: 2898
Elevated blood pressure (prehypertension): 2090
Stage 1 hypertension: 5936
Stage 2 hypertension: 4418
Hypertensive crisis: 1
```

My Explanation The number of people in the following categories: - Normal blood pressure: 2898, 21.6% of the total people - Elevated blood pressure (prehypertension): 2090, 15.6% of the total people - Stage 1 hypertension: 5936, 44.3% of the total people - Stage 2 hypertension: 4418, 33.0% of the total people - Hypertensive crisis: 1, less than 1% of the total people

*Note: When identifing the **'Hypertensive crisis'**, my understanding is that the both systolic and idastolic have to exceed 180/120 mm Hg.*

Part b)

[ ]:
```
df_change=df.copy(deep=True)
df_change['before_2017']='normal'
df_change['after_2017']='normal'
df_change.loc[(df_change['age'] <= 30) & (df_change['systolic'] >= 140) &␣
 ↪(df_change['diastolic'] >= 90), 'before_2017'] = 'hypertension'
df_change.loc[(df_change['age'] >= 65) & (df_change['systolic'] >= 150) &␣
 ↪(df_change['diastolic'] >= 80), 'before_2017'] = 'hypertension'
```

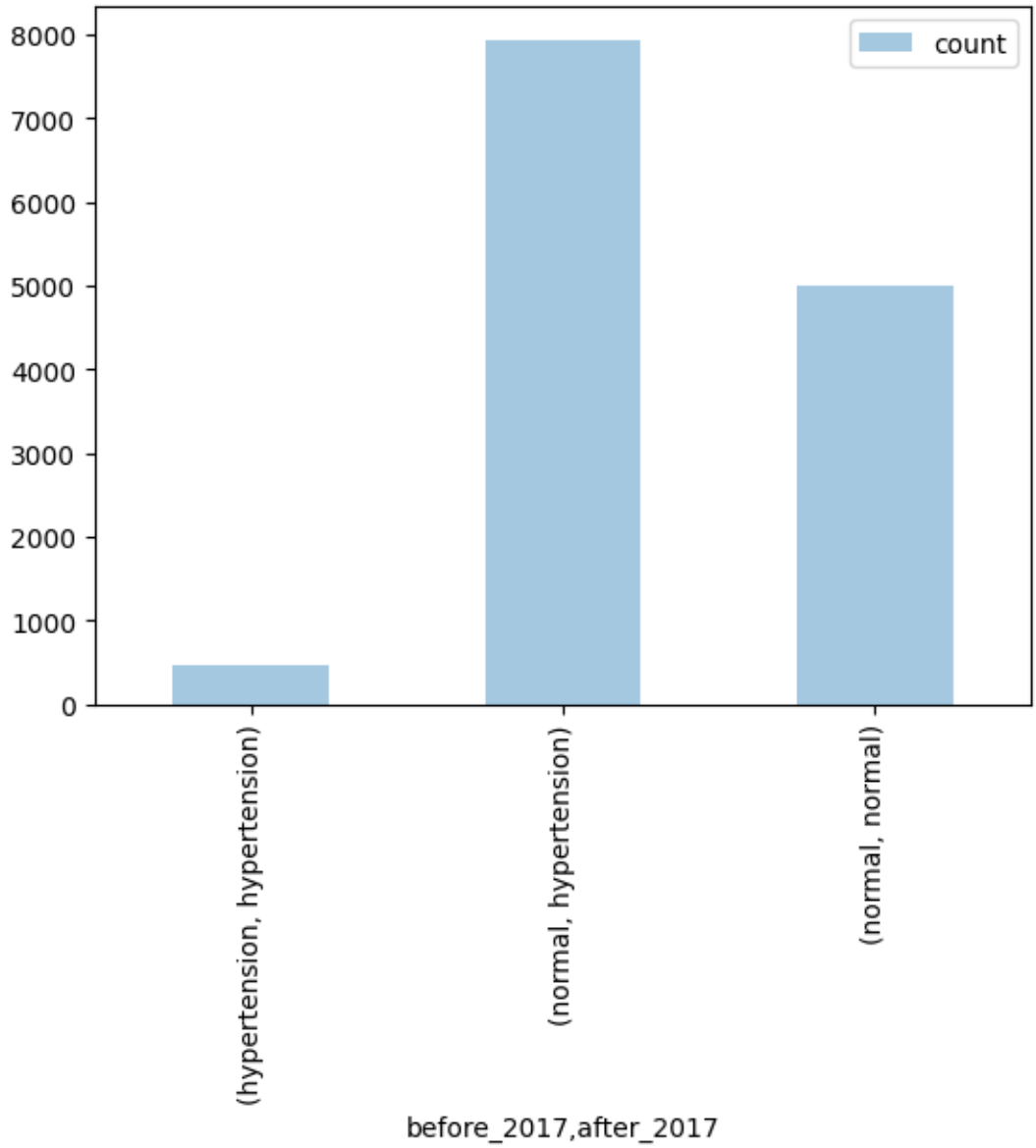
```
df_change.loc[((df_change['systolic'] >= 130) & (df_change['systolic'] <= 139))␣
↪| ((df_change['diastolic'] >= 80) & (df_change['diastolic'] <= 89)),␣
↪'after_2017'] = 'hypertension'
df_change.loc[(df_change['systolic'] >= 140) | ((df_change['diastolic'] >=␣
↪90)), 'after_2017'] = 'hypertension'
```

[ ]:
```
df_change.groupby(['before_2017', 'after_2017'])['before_2017'].value_counts()
```

[ ]:
```
before_2017   after_2017
hypertension  hypertension     468
normal        hypertension    7937
              normal          4988
Name: count, dtype: int64
```

[ ]:
```
df_change.groupby(['before_2017', 'after_2017'])['before_2017'].value_counts().
↪plot.bar(alpha=0.4, legend=True)
```

[ ]:
```
<Axes: xlabel='before_2017,after_2017'>
```



My Explanation

- From the analysis, we can see that there are 468 people remain **'hypertension'** regardless of the guideline change, taking up 3.5%
- There are 7937 people change from **'normal'** into **'hypertension'**, taking up 59.3%
- There are 4988 people remain **'normal'**, taking up 37.2%
- No people change from **'hypertension'** to **'normal'**

So generally, the new guideline seems to be more strict because it classify more people into hypertension

1.3.6 Q6: 10 pt:

Create a new variable "age_class" that is "early" if age < 40, "middle" if age is between 40 and 60 (inclusive) and "late" if age > 60. Report the mean and median number of sit-ups counts for each of those categories.

[ ]:
```
# put your code here
df['age_class']='early'
df.loc[(df['age'] >= 40) & (df['age'] <= 60),'age_class']='middle'
df.loc[(df['age'] > 60),'age_class']='late'
```

[ ]:
```
df.groupby('age_class')['sit-ups counts'].mean().round(1)
```

[ ]:
```
age_class
early     45.3
late      24.1
middle    31.8
Name: sit-ups counts, dtype: float64
```

[ ]:
```
df.groupby('age_class')['sit-ups counts'].median()
```

[ ]:
```
age_class
early     47.0
late      25.0
middle    32.0
Name: sit-ups counts, dtype: float64
```

My Explanation

- The mean for early, middle and late are 45.3, 31.8 and 24.1 respectively
- The median for early, middle and late are 47, 32 and 25 respectively

From which we can see that - With people getting older, they tend to do less sit-ups - The average is a slightly smaller than the median, which means that more people tend to do less sit-ups

1.3.7 Q7: 15 pt:

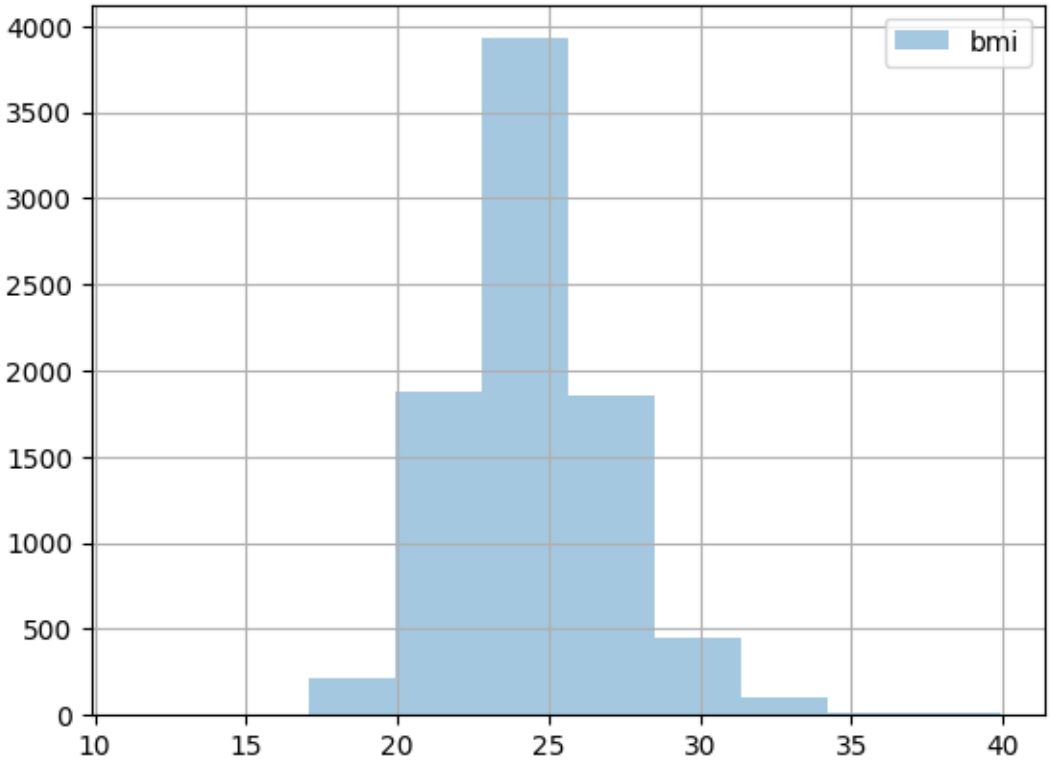
Calculate each person's Body Mass Index (BMI). With the metric system, the formula for BMI is weight in kilograms divided by height in meters squared. Since height is commonly measured in centimeters, an alternate calculation formula, dividing the weight in kilograms by the height in centimeters squared, and then multiplying the result by 10,000, can be used. (https://www.cdc.gov/nccdphp/dnpao/growthcharts/training/bmiage/page5_1.html)

Create two histograms of BMI: one for gender F and one for gender M. Describe similarites and differences of the two histograms.

[ ]:
```
# put your code here
df['bmi']=df['weight_kg']/(df['height_cm']/100)**2
```

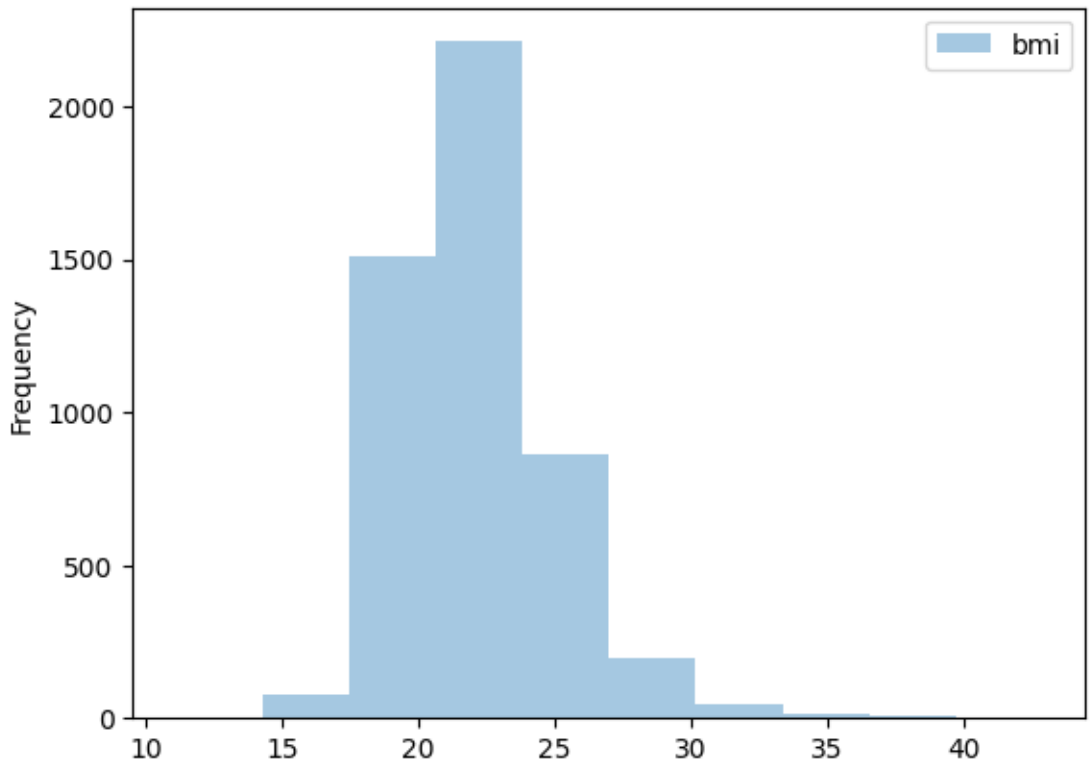
[ ]:
```
df[df['gender']=='M'].bmi.hist(bins=10, alpha=0.4, legend=True)
```

[ ]: <Axes: >



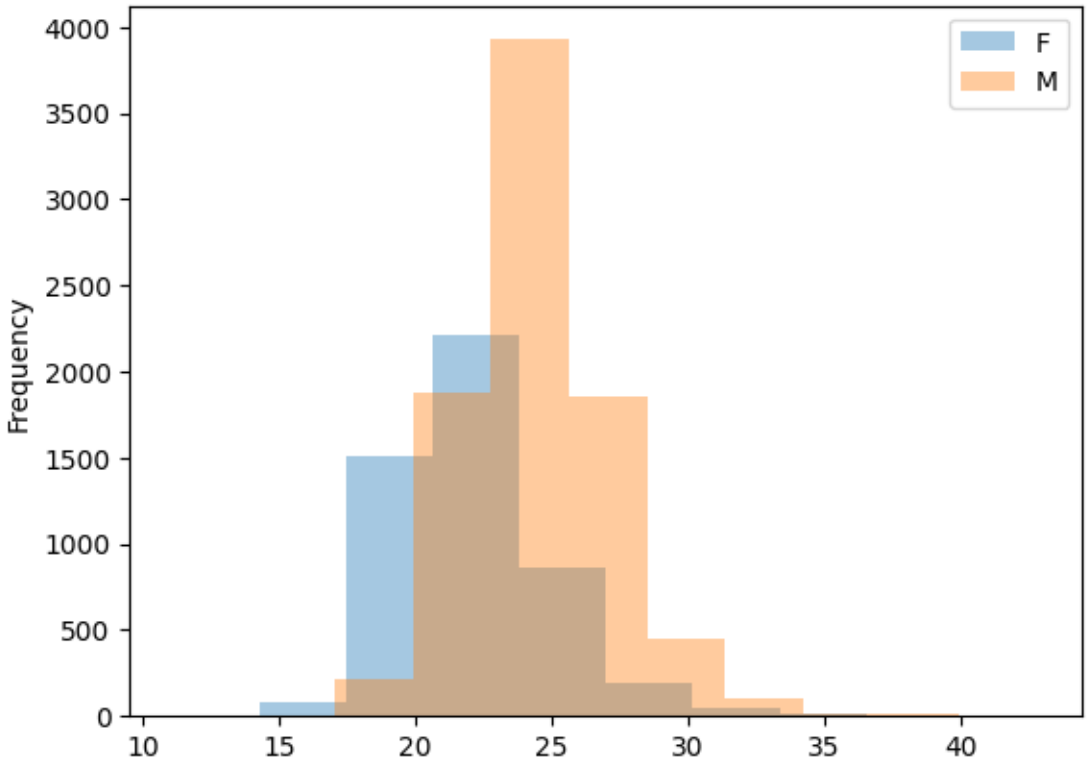
[ ]:
```
df[df['gender']=='F'].bmi.plot.hist(bins=10, alpha=0.4, legend=True)
```

[ ]: <Axes: ylabel='Frequency'>



[ ]:
```
df.groupby('gender').bmi.plot.hist(bins=10, alpha=0.4, legend=True)
```

[ ]:
```
gender
F    Axes(0.125,0.11;0.775x0.77)
M    Axes(0.125,0.11;0.775x0.77)
Name: bmi, dtype: object
```



My Explanation To compare bmi of male and female, put the two histograms into one figure

Similarities: - The two group of people are both "high in the middle and low on both sides", meaning that the distribution of BMI is quite uniform

Differences: - The average BMI of male is higher than female - The distribution of male is more uniform than female - More women tend to have lower BMI - Even the youngest male tend to have higher BMI - In this dataset, male is more than female

1.3.8 Q8: 12 pt:

Compare the mean values of height, weight, body fat %, grip force, sit and bend forward, and sit-ups count for gender F vs. gender M. You should use both quantitative and visual methods in your work.

[ ]:
```
# put your code here
df.groupby('gender')[['height_cm', 'weight_kg', 'body fat_%', 'gripForce', 'sit␣
↪and bend forward_cm', 'sit-ups counts' ]].mean().round(1)
```

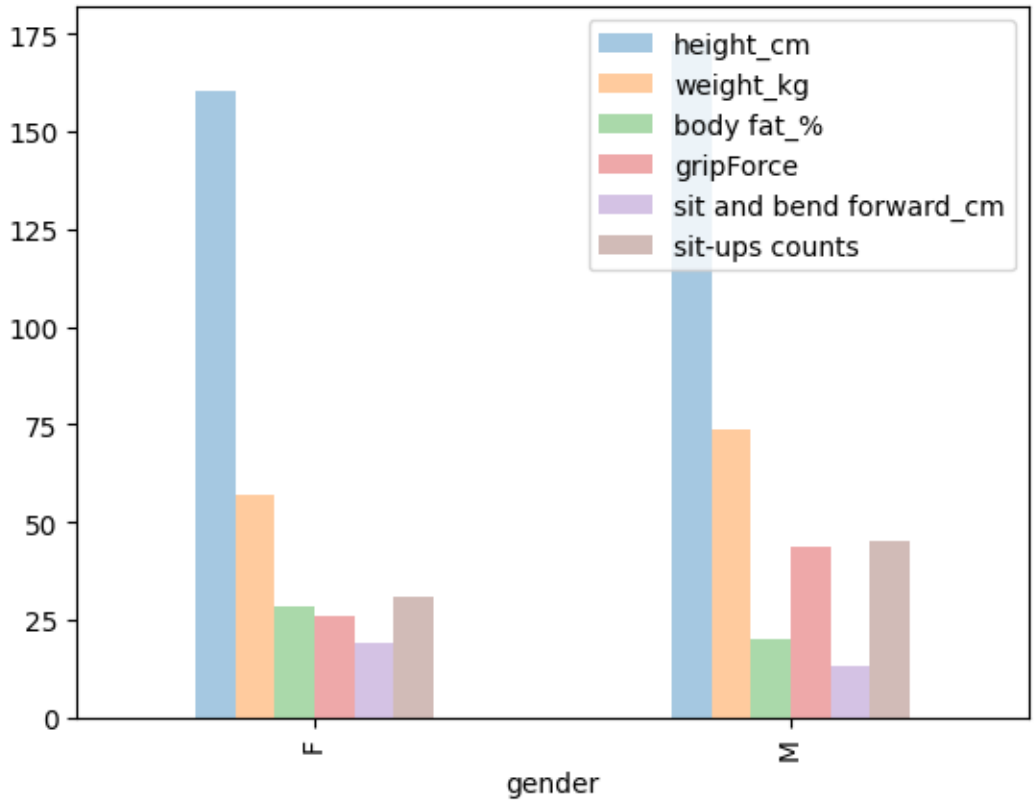
[ ]:

gender	height_cm	weight_kg	body fat_%	gripForce	sit and bend forward_cm	\
F	160.5	56.9	28.5	25.8	18.8	
M	173.3	73.6	20.2	43.4	13.1	

```
        sit-ups counts
gender
F                 30.9
M                 44.9
```

[ ]:
```
df.groupby('gender')[['height_cm', 'weight_kg', 'body fat_%', 'gripForce', 'sit␣
↪and bend forward_cm', 'sit-ups counts' ]].mean().plot.bar(alpha=0.4,␣
↪legend=True)
```

[ ]:
```
<Axes: xlabel='gender'>
```



My Explanation By applying both quantitative and visual methods, we can find that male overwhelm female in almost all fields except for **body fat** and **sit and bend forward**. It may indicate that female are less likely to gain weight or they pay more attention to their body fat. Also, it can be inferred that the female have better body flexibility.

1.3.9 Q9: 18 pt:

Take a look at gripForce (a measure of strength), "sit and bend forward" (a measure of flexibility), and "sit-ups count" (a measures of strength and endurance of the abdominals and hip-flexor mus-

cles). For each of those measures, calculate their z-scores (i.e. subtract the mean of the variable from each observation, then divide by the standard deviation of that score). Create an overall fitness score by adding the z-scores for each of those variables together, then use `pd.qcut()` (not `pd.cut()`) to assign them to five categories ranging from lowest score to highest score.

a) Report the number of individuals in each of the resulting categories.

b) Comment on the degree to which your categories correspond to the `class` column.

Part a)

[ ]:
```
# put your code here
df[['gripForce_z', 'sit and bend forward_cm_z', 'sit-ups counts_z']] =␣
↪(df[['gripForce', 'sit and bend forward_cm', 'sit-ups counts']] -␣
↪df[['gripForce', 'sit and bend forward_cm', 'sit-ups counts']].mean())/
↪df[['gripForce', 'sit and bend forward_cm', 'sit-ups counts']].std()
df[['gripForce_z', 'sit and bend forward_cm_z', 'sit-ups counts_z']]
```

[ ]:
```
       gripForce_z  sit and bend forward_cm_z  sit-ups counts_z
0         1.688127                   0.377303          1.416909
1        -0.053071                   0.128979          0.926599
2         0.737527                  -0.379495          0.646422
3         0.417523                  -0.001096          0.926599
4         0.615172                   1.406076          0.366245
…              …                          …                 …
13388    -0.109543                   0.259053          0.506334
13389    -0.373076                  -1.668418          0.576378
13390     2.497549                   0.140804          0.366245
13391    -1.662504                  -0.710595         -2.785744
13392    -0.100131                  -0.958919          0.786511

[13393 rows x 3 columns]
```

[ ]:
```
df['overall_fitness_score'] = df['gripForce_z'] + df['sit and bend␣
↪forward_cm_z'] + df['sit-ups counts_z']
df['overall_fitness_score']
```

[ ]:
```
0        3.482339
1        1.002506
2        1.004454
3        1.343026
4        2.387494
             …
13388    0.655845
13389   -1.465115
13390    3.004599
13391   -5.158842
13392   -0.272539
Name: overall_fitness_score, Length: 13393, dtype: float64
```

```
[ ]: pd.qcut(df['overall_fitness_score'], q=5)
```

```
[ ]: 0                    (1.773, 24.469]
     1                     (0.642, 1.773]
     2                     (0.642, 1.773]
     3                     (0.642, 1.773]
     4                    (1.773, 24.469]
                         …
     13388                 (0.642, 1.773]
     13389                (-1.738, -0.388]
     13390                 (1.773, 24.469]
     13391    (-8.549999999999999, -1.738]
     13392                 (-0.388, 0.642]
     Name: overall_fitness_score, Length: 13393, dtype: category
     Categories (5, interval[float64, right]): [(-8.549999999999999, -1.738] <
     (-1.738, -0.388] < (-0.388, 0.642] < (0.642, 1.773] < (1.773, 24.469]]
```

```
[ ]: df['overall_category'] = pd.qcut(df['overall_fitness_score'], q=5,␣
      ↪labels=['very poor', 'poor', 'fair', 'good', 'very good'])
```

```
[ ]: df[['overall_fitness_score','overall_category']]
```

```
[ ]:        overall_fitness_score overall_category
     0                   3.482339        very good
     1                   1.002506             good
     2                   1.004454             good
     3                   1.343026             good
     4                   2.387494        very good
     …                      …               …
     13388               0.655845             good
     13389              -1.465115             poor
     13390               3.004599        very good
     13391              -5.158842        very poor
     13392              -0.272539             fair

     [13393 rows x 2 columns]
```

```
[ ]: df.groupby('overall_category').size()
     # df['overall_category'].value_counts()
```

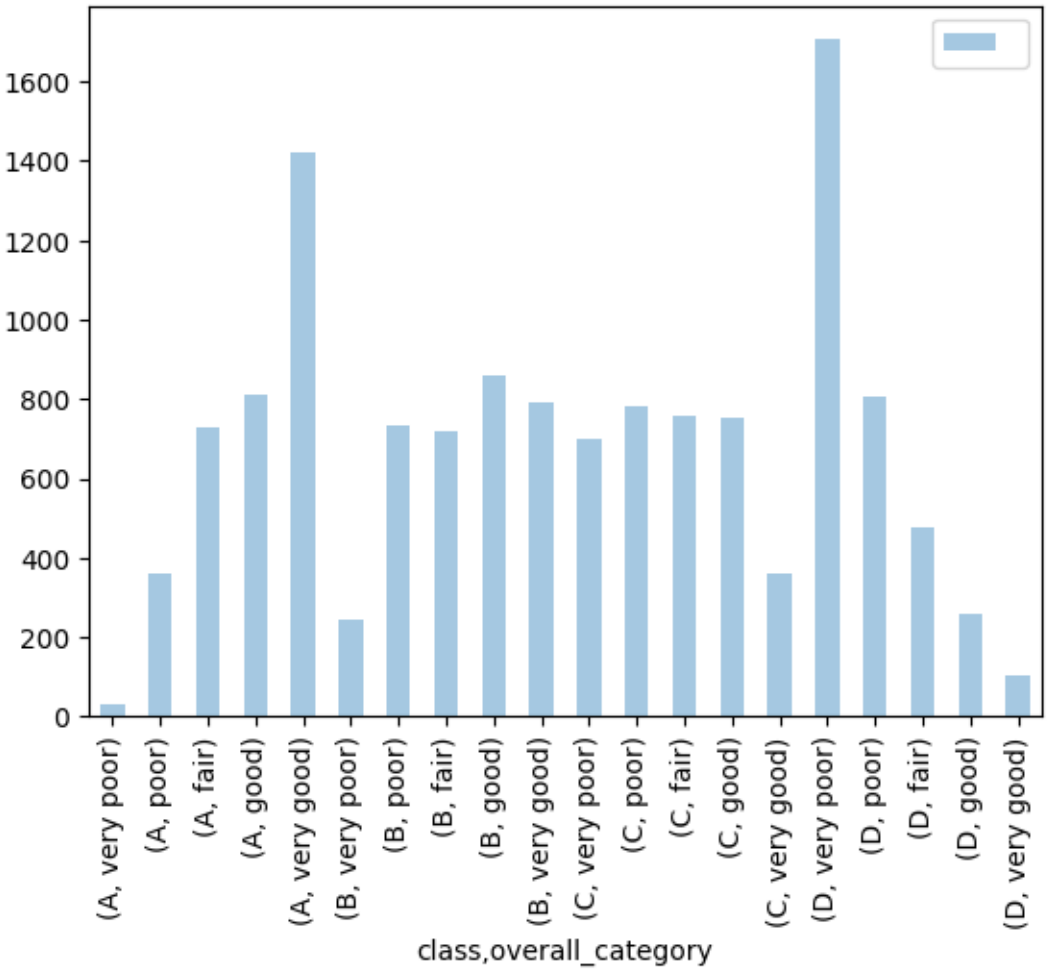
```
[ ]: overall_category
     very poor    2679
     poor         2678
     fair         2679
     good         2678
     very good    2679
     dtype: int64
```

My Explanation We seperate the individual into 5 catugories as the following: - very poor: [-8.549999999999999, -1.738], 2679 people - poor: (-1.738, -0.388], 2678 people - fair: (-0.388, 0.642], 2679 people - good: (0.642, 1.773], 2678 people - very good: (1.773, 24.469], 2679 people

Part b)

```
[ ]: df.groupby(['class', 'overall_category']).size().plot.bar(alpha=0.4,␣
      ↪legend=True)
```

```
[ ]: <Axes: xlabel='class,overall_category'>
```



My Explanation The original **class** 'A' and 'D' performs well since 'very good' and 'very poor' are high respectively. However, since we have 2 remaining **class** 'B' and 'C', and three new intervals 'poor', 'fair', and 'good', they may not match well. The next step should be change the new categories into 4 categories.

1.3.10 IMPORTANT: Ensure your complete notebook runs without errors from top to bottom (check by using "Run All").

1.4 Please submit your completed notebook in .IPYNB and .HTML formats via Canvas