Cardio Data Project

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

Importing data (csv file)

```
#importing data csv file
df = pd.read_csv("cardio_base.csv")
df.head()
```

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	smoke
0	0	18393	2	168	62.0	110	80	1	0
1	1	20228	1	156	85.0	140	90	3	0
2	2	18857	1	165	64.0	130	70	3	0
3	3	17623	2	169	82.0	150	100	1	0
4	4	17474	1	156	56.0	100	60	1	0

```
# determining number of rows and columns
df.shape
```

```
(70000, 9)
```

Manipulating data

since the age column was given in days it needed to be converted into years

```
# dividing by 365 to get value in years
df['age_years'] = df['age'] // 365
df.drop('age', axis=1, inplace=True)
df.head()
```

	id	gender	height	weight	ap_hi	ap_lo	cholesterol	smoke	age_years
0	0	2	168	62.0	110	80	1	0	50
1	1	1	156	85.0	140	90	3	0	55
2	2	1	165	64.0	130	70	3	0	51
3	3	2	169	82.0	150	100	1	0	48
4	4	1	156	56.0	100	60	1	0	47

as observed above now the column of "age" is dropped and "age_years" is added

Relation between weight and age

In this line, the DataFrame df is grouped by the 'age_years' column, and the mean of the 'weight' column is calculated for each age group. The reset_index() function is then applied to reset the index of the resulting DataFrame (avg_weight_by_age) back to the default integer index. This is done because after the groupby operation, the 'age_years' column becomes the new index, and resetting the index brings it back as a regular column. In short

- here the reset index code is inserted so that after the new data frame "avg_weight_by_age" is grouped by age and weight so its index is set back to default otherwise errors might occur

```
avg_weight_by_age = df.groupby('age_years')['weight'].mean().reset_index()
highest_avg_weight = avg_weight_by_age['weight'].idxmax()
highest_avg_weight_age = avg_weight_by_age.loc[highest_avg_weight, 'age_years']
highest_avg_weight_age
```

```
63
```

Here, idxmax() is used to find the index of the row that has the maximum value in the 'weight' column of the avg_weight_by_age DataFrame. This index value corresponds to the row with the highest average weight.

```
lowest_avg_weight = avg_weight_by_age['weight'].idxmin()
lowest_avg_weight_age = avg_weight_by_age.loc[lowest_avg_weight, 'age_years']
lowest_avg_weight_age
```

```
30
```

Similarly idxmin() will give us the lowest value.

now observing the difference

```
print("The age group with the highest average weight is", highest_avg_weight_age)
print("The age group with the lowest average weight is", lowest_avg_weight_age)
```

```
The age group with the highest average weight is 63
The age group with the lowest average weight is 30
```

Now calculating percentage difference

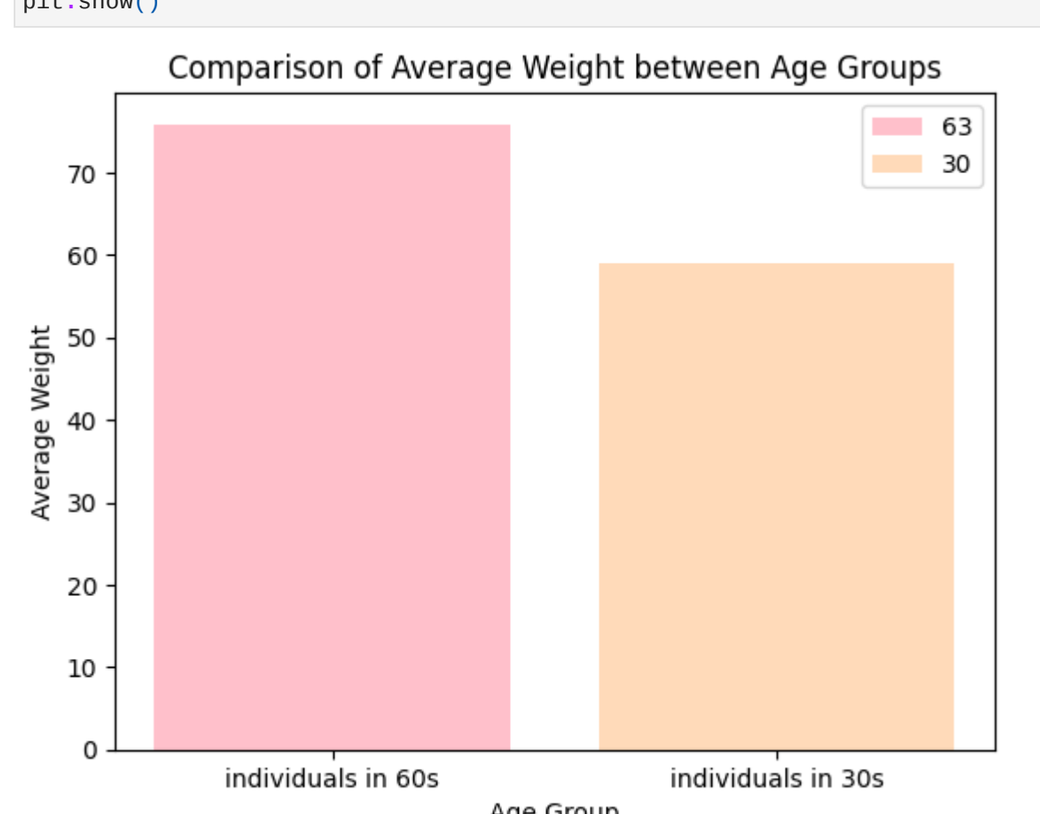
```
percentage_heavier = (avg_weight_by_age.loc[highest_avg_weight, 'weight']  / avg_weight_by_age.loc[lowest_avg_weight, 'weight']) * 100
print("The heavier weight age groups is {:.2f}% heavier.".format(percentage_heavier))
```

```
The heavier weight age groups is 128.60% heavier.
```

it can be observed that the heavier age group is 128.60% havaier

PLotting the relation

```
highest_avg_weight_data = avg_weight_by_age[avg_weight_by_age['age_years'] == highest_avg_weight_age]
lowest_avg_weight_data = avg_weight_by_age[avg_weight_by_age['age_years'] == lowest_avg_weight_age]

plt.bar('individuals in 60s', highest_avg_weight_data['weight'], label=str(highest_avg_weight_age), color="pink")
plt.bar('individuals in 30s', lowest_avg_weight_data['weight'], label=str(lowest_avg_weight_age), color="peachpuff")

plt.xlabel('Age Group')
plt.ylabel('Average Weight')
plt.title('Comparison of Average Weight between Age Groups')
plt.legend()

plt.show()
```



visualised data of age groups and their respective average weights

Cholestrol levels relation with age

setting 50 years of age as threshold and comparing

```
age_cholesterol = df[['age_years', 'cholesterol']]

age_cholesterol.loc[:, 'over_50'] = age_cholesterol['age_years'] > 50

avg_cholesterol_over_50 = age_cholesterol[age_cholesterol['over_50']]['cholesterol'].mean()
avg_cholesterol_under_50 = age_cholesterol[~age_cholesterol['over_50']]['cholesterol'].mean()

if avg_cholesterol_over_50 > avg_cholesterol_under_50:
    print("People over 50 years of age have higher cholesterol levels than the rest.")
else:
    print("People over 50 years of age do not have higher cholesterol levels than the rest.")
```

```
People over 50 years of age have higher cholesterol levels than the rest.
C:\Users\Dell\AppData\Local\Temp\ipykernel_7164\727019440.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_cholesterol.loc[:, 'over_50'] = age_cholesterol['age_years'] > 50
```

It can be observed through data analysis that the people with age greater than 50 years of age have higher cholestrols than the rest.

Now calculating the percentage.

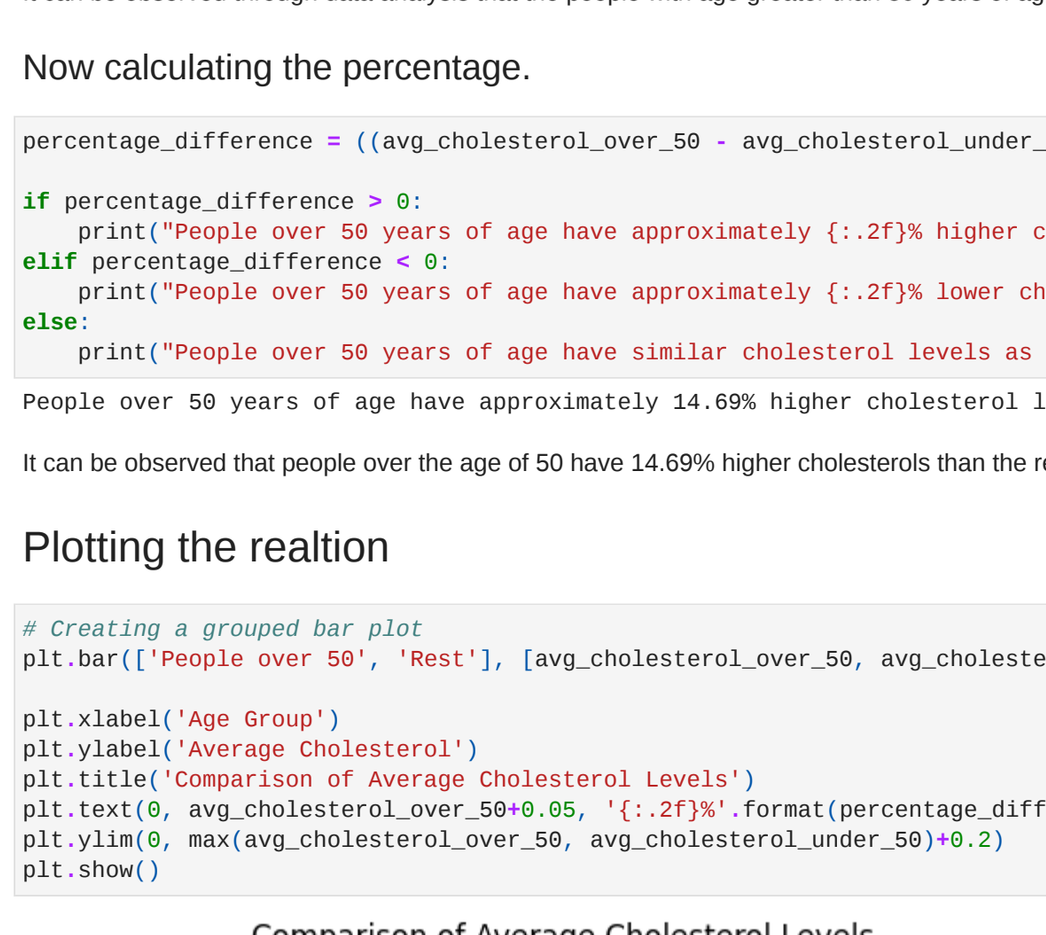
```
percentage_difference = ((avg_cholesterol_over_50 - avg_cholesterol_under_50) / avg_cholesterol_under_50) * 100

if percentage_difference > 0:
    print("People over 50 years of age have approximately {:.2f}% higher cholesterol levels than the rest.".format(percentage_difference))
elif percentage_difference < 0:
    print("People over 50 years of age have approximately {:.2f}% lower cholesterol levels than the rest.".format(abs(percentage_difference)))
else:
    print("People over 50 years of age have similar cholesterol levels as the rest.")
```

```
People over 50 years of age have approximately 14.69% higher cholesterol levels than the rest.
```

It can be observed that people over the age of 50 have 14.69% higher cholesterols than the rest.

Plotting the realtion

```
# Creating a grouped bar plot
plt.bar(['People over 50', 'Rest'], [avg_cholesterol_over_50, avg_cholesterol_under_50],color=["peachpuff","silver"])

plt.xlabel('Age Group')
plt.ylabel('Average Cholesterol')
plt.title('Comparison of Average Cholesterol Levels')
plt.text(0, avg_cholesterol_over_50+0.05, '{:.2f}%'.format(percentage_difference), ha='center')
plt.ylim(0, max(avg_cholesterol_over_50, avg_cholesterol_under_50)+0.2)
plt.show()
```



Visualized data of cholesterol levels comparison

Data manipulation

The given data set had values of gender as 2 and 1 hence it was not clear either 1 was male or female hence using height as a deciding factor since males are typically taller than females hence determining and labeling genders

```
gender_height_subset = df[['gender', 'height']]
avg_height_by_gender = gender_height_subset.groupby('gender')['height'].mean()
if avg_height_by_gender[2] > avg_height_by_gender[1]:
    print("The average height of (gender 2) is higher than that of (gender 1) hence gender 2 is males.")
else:
    print("The average height of (gender 1) is higher than that of (gender 2) hence gender 1 is males.")
```

```
The average height of (gender 2) is higher than that of (gender 1) hence gender 2 is males.
```

Since "2" is males hence for further data analysis introducung a new column and mapping females to 1 and males to 2

```
gender_mapping = {1: 'Female', 2: 'Male'}
df['gender_label'] = df['gender'].map(gender_mapping)
df.head()
```

	id	gender	height	weight	ap_hi	ap_lo	cholesterol	smoke	age_years	gender_label
0	0	2	168	62.0	110	80	1	0	50	Male
1	1	1	156	85.0	140	90	3	0	55	Female
2	2	1	165	64.0	130	70	3	0	51	Female
3	3	2	169	82.0	150	100	1	0	48	Male
4	4	1	156	56.0	100	60	1	0	47	Female

Relationship between gender and status of smoking

Grouping by columns gender and smoke to get results

```
gender_smoke = df[['gender', 'smoke']]
smokers_by_gender = gender_smoke.groupby('gender')['smoke'].mean() * 100
if smokers_by_gender[2] > smokers_by_gender[1]:
    print("Males are more likely to be smokers than females.")
else:
    print("Males are not more likely to be smokers than females.")
```

```
Males are more likely to be smokers than females.
```

Now calculating the ratio

This time using the gender label column directly as its much more convenient and not confusing like using 1 and 2 instead of males and females labels

```
gender_smoke = df[['gender_label', 'smoke']]
smokers_by_genderL = gender_smoke.groupby('gender_label')['smoke'].mean() * 100
odds_ratio = smokers_by_genderL['Male'] / smokers_by_genderL['Female']
if odds_ratio > 1:
    print("Males are {:.2f} times more likely to be smokers than females.".format(odds_ratio))
elif odds_ratio < 1:
    print("Females are {:.2f} times more likely to be smokers than males.".format(1/odds_ratio))
else:
    print("Males and females are equally likely to be smokers.")
```

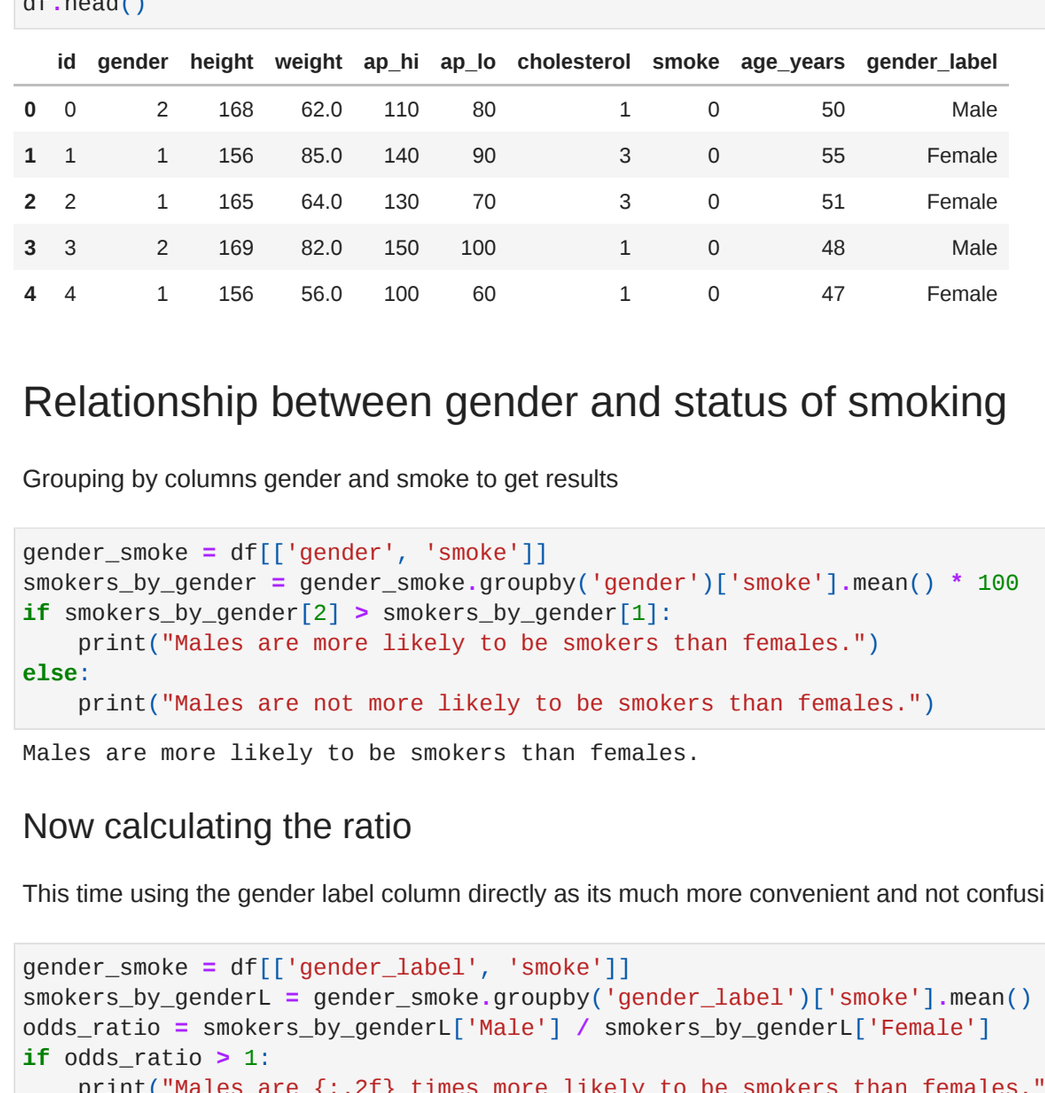
```
Males are 12.26 times more likely to be smokers than females.
```

It can be observed that the males are 12.26 times more liekly to smoke as compared to females.

Plotting the relation

firest we need to determine the smoking rates of males and females to plot the graphs more accurately

```
# Calculating the smoking rates for males and females
male_smoking_rate = df[df['gender_label'] == "Male"]['smoke'].mean() *100
female_smoking_rate = df[df['gender_label'] == "Female"]['smoke'].mean() * 100

# Calculating the ratio of male smoking rate to female smoking rate
smoking_ratio = male_smoking_rate / female_smoking_rate

# Creating a bar plot
plt.bar(['Male', 'Female'], [male_smoking_rate, female_smoking_rate],color=["lightblue","palegreen"])

plt.xlabel('Gender')
plt.ylabel('Smoking Rate %')
plt.title('Comparison of Smoking Rates between Genders')
plt.text(0, male_smoking_rate+0.2, '{:.2f} %'.format(male_smoking_rate), ha="center")
plt.text(1, female_smoking_rate+0.2, '{:.2f} %'.format(female_smoking_rate), ha="center")
plt.text(0.5, max(male_smoking_rate, female_smoking_rate)+2 , 'Male:Female Ratio = {:.2f}'.format(smoking_ratio), ha='center')
plt.ylim(0, max(male_smoking_rate, female_smoking_rate)+3)
plt.show()
```



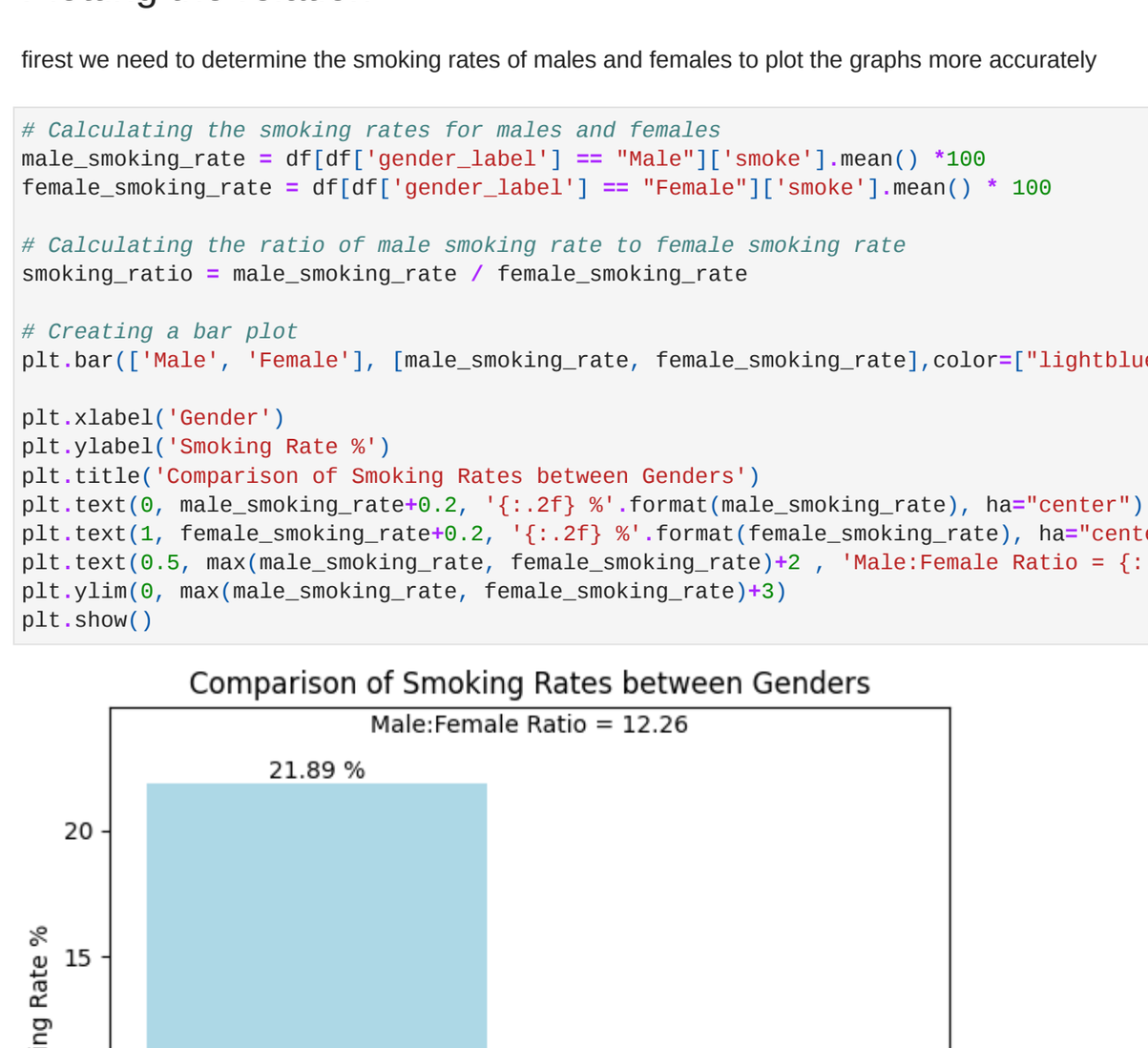
Plotting for smokers and non smokers by genders

First we determine the number of smoker and non smoker male and females by breaking up the data into 4 parts

```
# Counting the number of smokers and non-smokers for each gender
male_smokers = len(df[(df['gender_label'] == 'Male') & (df['smoke'] == 1)])
male_non_smokers = len(df[(df['gender_label'] == 'Male') & (df['smoke'] == 0)])
female_smokers = len(df[(df['gender_label'] == 'Female') & (df['smoke'] == 1)])
female_non_smokers = len(df[(df['gender_label'] == 'Female') & (df['smoke'] == 0)])
```

Now plotting a pie chart for better understanding

```
# Creating labels and sizes for the pie chart
labels = ['Male Smokers', 'Male Non-Smokers', 'Female Smokers', 'Female Non-Smokers']
sizes = [male_smokers, male_non_smokers, female_smokers, female_non_smokers]
colors = ['palegreen', 'lightcyan', 'pink', 'peachpuff']
explode = (0.1, 0, 0.1,0)

# Plotting the pie chart
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=300,explode=explode, shadow= True)
plt.title('Smoking Habits by Gender')
plt.axis('equal')

plt.show()
```



Calculating the average height of the tallest 1% of individuals

Quantile 0.99 gives us the 1% or 0.01 tsllest individuals

```
height_percentile = df['height'].quantile(0.99)
tallest_1_percent = df[df['height'] >= height_percentile]
tallest_heights = tallest_1_percent['height']
tallest_heights
```

```
62       187
76       188
94       185
137      184
193      185
        ...
69772    184
69781    184
69785    185
69873    188
69913    186
Name: height, Length: 761, dtype: int64
```

Now calculating average

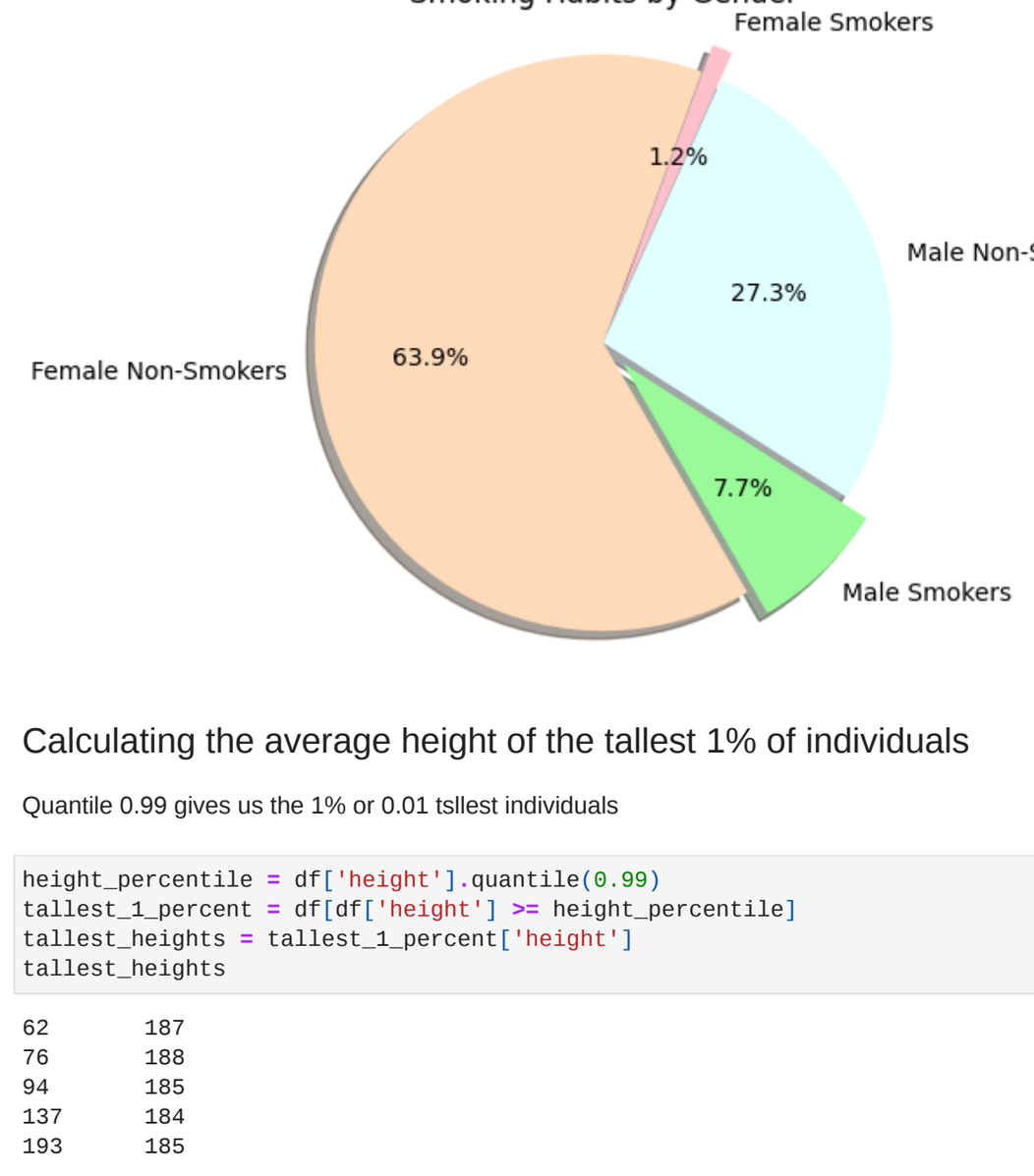
```
avg_height_tallest_1_percent = tallest_1_percent['height'].mean()
print("The average height of the tallest 1% of people is:", avg_height_tallest_1_percent)
```

```
The average height of the tallest 1% of people is: 186.68068331143232
```

It can be observed that the tallest 1% of the people have an average height of 186.68 cm

Visualising this information

```
# Creating a histogram plot of heights
plt.hist(avg_height_tallest_1_percent, bins=3)

plt.xlabel('Height')
plt.ylabel('Frequency')
plt.title('Distribution of Heights')
plt.show()
```



we can observe above that the top 1% tallest people height ranges from between about 186.55 cm to 186.85 cm

Highest spearman correlation between 2 features

In this line, the code calculates the correlation matrix of the DataFrame df using the corr() method with the method parameter set to 'spearman'. The Spearman rank correlation coefficient measures the monotonic relationship between variables.

```
correlation_matrix = df.corr(method='spearman')

# Finding the pair with the highest correlation (excluding self-correlations and duplicate pairs)
highest_correlation = correlation_matrix.unstack().sort_values(ascending=False).drop_duplicates()
highest_correlation = highest_correlation[highest_correlation != 1].index[0]

# Extracting the names of the features with the highest correlation
feature1, feature2 = highest_correlation

print("The pair of features with the highest Spearman rank correlation is:", feature1, "and", feature2)
```

```
The pair of features with the highest Spearman rank correlation is: ap_lo and ap_hi
C:\Users\Dell\AppData\Local\Temp\ipykernel_7164\1714641980.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to Fal
se. Select only valid columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = df.corr(method='spearman')
```

As we can see the highest spearman rank correlation is between ap lo and ap hi

Here, the correlation matrix is transformed to a Series by using unstack(). This reshapes the DataFrame to a multi-index Series, with pairs of column names as the index. Then, sort_values() is applied to sort the Series in descending order of correlation values. The drop_duplicates() function is used to remove self-correlations and duplicate pairs, retaining only unique pairs with the highest correlations.

Height of individuals 2 standard deviations far from average height value

Calculating the avg height as well as the std height and add and subtracting 2 times the std value to get the values that are 2 times the std apart from the average value of height.

```
avg_height = df['height'].mean()
std_dev = df['height'].std()

upper_bound = avg_height + (2 * std_dev)
lower_bound = avg_height - (2 * std_dev)

outliers = df[(df['height'] > upper_bound) | (df['height'] < lower_bound)]

percentage_outliers = (len(outliers) / len(df)) * 100

print("The percentage of people more than 2 standard deviations away from the average height is:", percentage_outliers)
```

```
The percentage of people more than 2 standard deviations away from the average height is: 3.3357142857142854
```

It can be observed that the percentage of people more than 2 standard deviations away from the average height is about 3.33%

Manipulating data

The data was broken into 2 csv files with on csv file only containing 2 columns hence the need to merge data sets arises for further analysis

```
df2 = pd.read_csv('cardio_alco.csv')
df2.head()
```

	id;alco
0	44;0
1	45;0
2	46;0
3	47;0
4	49;0

The csv data contains ";" instead of "," due to which the values aren't speareated hence manipulating the csv file to basically make it semi colon speareated values data frame

```
df2 = pd.read_csv('cardio_alco.csv', sep=';')
df2.head()
```

	id	alco
0	44	0
1	45	0
2	46	0
3	47	0
4	49	0

Now the data is in its proper shape of 2 columns.

```
alcohol = df2['alco']    # .round().astype(int)

# Concatenating the desired column from df2 to df
merged_df = pd.concat([df, alcohol], axis=1)

# Writing the merged dataframe to a new CSV file (optional)
merged_df.to_csv('merged_file.csv', index=False)
merged_df.head()
```

	id	gender	height	weight	ap_hi	ap_lo	cholesterol	smoke	age_years	gender_label	alco
0	0	2	168	62.0	110	80	1	0	50	Male	0.0
1	1	1	156	85.0	140	90	3	0	55	Female	0.0
2	2	1	165	64.0	130	70	3	0	51	Female	0.0
3	3	2	169	82.0	150	100	1	0	48	Male	0.0
4	4	1	156	56.0	100	60	1	0	47	Female	0.0

Now the 2 data sets are successfully merged and a new csv file is created to save the data set by the name of merged_file.csv

calculating the percentage of individuals who are older than 50 years of age but consume alcohol

filtering and dividing the data set into 2 age groups

```
# Filtering the dataframe to include only people above 50 years of age
above_50_df = merged_df[merged_df['age_years'] > 50]

# Calculating the total number of people above 50 years of age who consume alcohol
above50_alcocon = len(above_50_df[above_50_df['alco'] == 1])

# Calculating the total number of people above 50 years of age
num_above_50 = len(above_50_df)

# Calculating the percentage of people above 50 years of age who consume alcohol
percent_above50_alcocon = (above50_alcocon / num_above_50) * 100

# Printing the result
print(f"The percentage of people above 50 years of age who consume alcohol: {percent_above50_alcocon:.2f}%")
```

```
The percentage of people above 50 years of age who consume alcohol: 4.40%
```

it can be observed that the people above 50 years of age who consume alcohol is 4.40%

Visualizing data

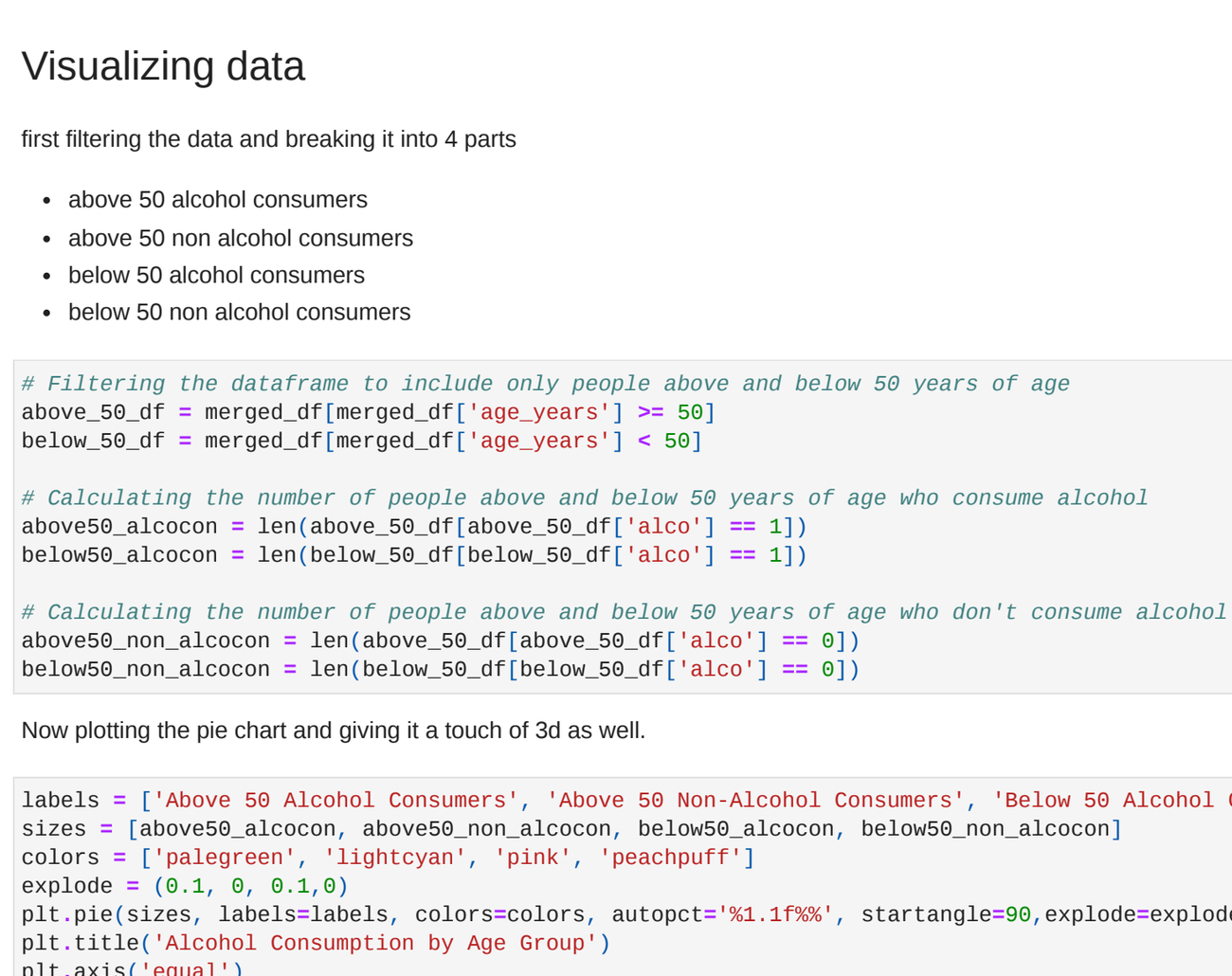
first filtering the data and breaking it into 4 parts

- above 50 alcohol consumers
- above 50 non alcohol consumers
- below 50 alcohol consumers
- below 50 non alcohol consumers

```
# Filtering the dataframe to include only people above and below 50 years of age
above_50_df = merged_df[merged_df['age_years'] >= 50]
below_50_df = merged_df[merged_df['age_years'] < 50]

# Calculating the number of people above and below 50 years of age who consume alcohol
above50_alcocon = len(above_50_df[above_50_df['alco'] == 1])
below50_alcocon = len(below_50_df[below_50_df['alco'] == 1])

# Calculating the number of people above and below 50 years of age who don't consume alcohol
above50_non_alcocon = len(above_50_df[above_50_df['alco'] == 0])
below50_non_alcocon = len(below_50_df[below_50_df['alco'] == 0])
```

Now plotting the pie chart and giving it a touch of 3d as well.

```
labels = ['Above 50 Alcohol Consumers', 'Above 50 Non-Alcohol Consumers', 'Below 50 Alcohol Consumers', 'Below 50 Non-Alcohol Consumers']
sizes = [above50_alcocon, above50_non_alcocon, below50_alcocon, below50_non_alcocon]
colors = ['palegreen', 'lightcyan', 'pink', 'peachpuff']
explode = (0.1, 0, 0.1,0)
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90,explode=explode, shadow= True)
plt.title('Alcohol Consumption by Age Group')
plt.axis('equal')

plt.show()
```



Here we can easily observe all the alcohlic and non alcohlic individsuals by their age groups above 50 and below 50 and a very small percentage in both age groups consume alcohol most people do not consume alcohol

Calculating if smoking has significant effect on blood pressures

Breaking up the data frame into smokers and non smokers catagories and now analysing the data for smokers and non smokers to see the effects of smoking on blood pressure

```
# importing library
import scipy.stats as stats

# Filtering the dataset for smokers and non-smokers
smokers_df = merged_df[merged_df['smoke'] == 1]
non_smokers_df = merged_df[merged_df['smoke'] == 0]

# Calculating the mean values of systolic and diastolic blood pressure for smokers and non-smokers
smokers_mean_ap_hi = smokers_df['ap_hi'].mean()
smokers_mean_ap_lo = smokers_df['ap_lo'].mean()
non_smokers_mean_ap_hi = non_smokers_df['ap_hi'].mean()
non_smokers_mean_ap_lo = non_smokers_df['ap_lo'].mean()

# Performing hypothesis test using t-test
t_statistic, p_value = stats.ttest_ind(smokers_df['ap_hi'], non_smokers_df['ap_hi'], equal_var=False)

# Printing the results
print("Mean ap hi for smokers:", smokers_mean_ap_hi)
print("Mean ap hi for non-smokers:", non_smokers_mean_ap_hi)
print("Mean ap lo for smokers:", smokers_mean_ap_lo)
print("Mean ap lo for non-smokers:", non_smokers_mean_ap_lo)
print("T-statistic:", t_statistic)
print("P-value:", p_value)
```

```
Mean ap hi for smokers: 128.36051223861242
Mean ap hi for non-smokers: 128.86143096614498
Mean ap lo for smokers: 99.77451775004053
Mean ap lo for non-smokers: 96.32654979555389
T-statistic: -0.736802791500064
P-value: 0.461244784929987
```

- calculating the mean (average) values of ap hi and ap lo for smokers and non smokers
- calculating T-statistic and P-value
- as the T-statistic = -0.7 aprox that shows very tiny differnce
- P-value is greater ithan 0.5 there forer the difference does not matter

Hence forth we can conclude that smokers and non smokers had no siginificant differnce in blood pressures

calculating if smoking has significant effect on Cholestrols

```
# Separating the data for smokers and non-smokers
smokers_cholesterol = merged_df[merged_df['smoke'] == 1]['cholesterol']
non_smokers_cholesterol = merged_df[merged_df['smoke'] == 0]['cholesterol']

# Calculating the mean values
mean_cholesterol_smokers = np.mean(smokers_cholesterol)
mean_cholesterol_non_smokers = np.mean(non_smokers_cholesterol)

# Performing hypothesis test (two-sample t-test)
t_statistic, p_value = stats.ttest_ind(smokers_cholesterol, non_smokers_cholesterol)

# Printing the results
print("Mean cholesterol for smokers:", mean_cholesterol_smokers)
print("Mean cholesterol for non-smokers:", mean_cholesterol_non_smokers)
print("T-statistic:", t_statistic)
print("P-value:", p_value)
```

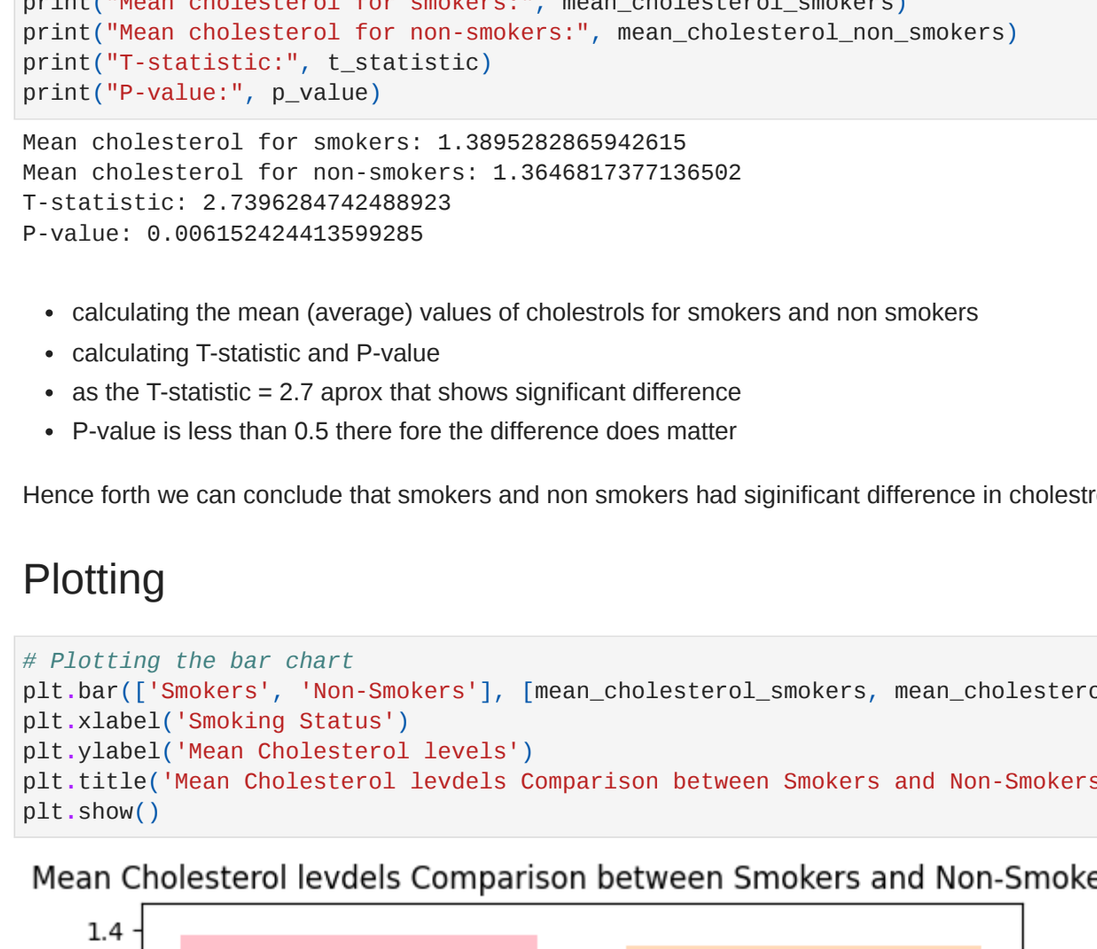
```
Mean cholesterol for smokers: 1.3895282865942615
Mean cholesterol for non-smokers: 1.3646817377136502
T-statistic: 2.7396284742488923
P-value: 0.006152424413599285
```

- calculating the mean (average) values of cholestrols for smokers and non smokers
- calculating T-statistic and P-value
- as the T-statistic = 2.7 aprox that shows significant difference
- P-value is less than 0.5 there fore the difference does matter

Hence forth we can conclude that smokers and non smokers had siginificant difference in cholestrol levels

Plotting

```
# Plotting the bar chart
plt.bar(['Smokers', 'Non-Smokers'], [mean_cholesterol_smokers, mean_cholesterol_non_smokers], color=['pink', 'peachpuff'])
plt.xlabel('Smoking Status')
plt.ylabel('Mean Cholesterol levels')
plt.title('Mean Cholesterol levdels Comparison between Smokers and Non-Smokers')
plt.show()
```



Calculating if smoking has significant effect on weight

```
# Subset the data for smokers and non-smokers
smokers_weight = merged_df[merged_df['smoke'] == 1]['weight']
non_smokers_weight = merged_df[merged_df['smoke'] == 0]['weight']

# Performing independent two-sample t-test
t_statistic, p_value = stats.ttest_ind(smokers_weight, non_smokers_weight)

# Printing the results
print("Mean weight for smokers:", smokers_weight.mean())
print("Mean weight for non-smokers:", non_smokers_weight.mean())
print("T-statistic:", t_statistic)
print("P-value:", p_value)
```

```
Mean weight for smokers: 77.34435078618901
Mean weight for non-smokers: 73.90235152198775
T-statistic: 17.974105126565
P-value: 4.517830901143157e-72
```

- calculating the mean (average) values of weight for smokers and non smokers
- calculating T-statistic and P-value
- as the T-statistic = 17.94 aprox that shows a huge difference
- P-value is very less than 0.5 there fore the difference does matter infact it is a very big difference

Hence forth we can conclude that smokers and non smokers had a large difference in weight

Plotting

```
# Calculating the mean weights
mean_smokers_weight = smokers_weight.mean()
mean_non_smokers_weight = non_smokers_weight.mean()

# Plotting the bar chart
plt.bar(['Smokers', 'Non-Smokers'], [mean_smokers_weight, mean_non_smokers_weight], color=['pink', 'peachpuff'])
plt.xlabel('Smoking Status')
plt.ylabel('Mean Weight')
plt.title('Mean Weight Comparison between Smokers and Non-Smokers')
plt.show()
```