

**ANL252**

**Python for Data Analytics**

**Group-based Assignment**

**July 2021 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Tan Mei Ping** | **Q1872131** |
| **Chia Wei Hao Terry** | **H1870852** |
| **Lu Kai Wen** | **J1911368** |
| **Ong Guo Lun** | **M1072994** |

**Tutorial Group: T09, Group 5**

**Instructor’s Name: Dr. Munish Kumar**

**Submission Date: 29/08/2021**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **PI Number** | **Contribution** | **Signature** |
| **Chia Wei Hao Terry** | **H1870852** | **Q2** |  |
| **Tan Mei Ping** | **Q1872131** | **Q2** |  |
| **Lu Kai Wen** | **J1911368** | **Q3** |  |
| **Ong Guo Lun** | **M1072994** | **Q1** |  |

**Question 1**

"""

Created on Sat Aug 21 14:27:00 2021

@author: ANL252 Group5

"""

import numpy as np

import matplotlib.pyplot as plt

**#question 1a**

xxy\_arr = np.array([[4, 6, 8, 4, 10, 1, 9, 5, 2, 7, 8, 2, 2, 8, 7, 9, 1, 4, 6, 9],

              [0.2, 0.1, 0.3, 0.6, 0.1, 0.4, 0.6, 0.3, 0.5, 0.5, 0.1, 0.9, 0.8, 1, 0.9, 0.1, 0.2, 0.2, 0.7, 0.7],

              [1.16, 0.06, -1.79, 1.55, -4.88, 1.37, -1.25, -1.1, 3.23, -2.71, -0.99, 3.23, 4.55, 2.7, -1.13, -0.88, 2.08, 1.62, -0.9, 0.46]])

arr\_y = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

arr\_e = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

**#question 1b**

#y2 = 2 − 0.5(x1) + 2.5(x2)

i = 0

y2 = 0

print("-----Q1B---------")

while i < xxy\_arr.shape[1]:

    y2 = 2 - 0.5\*xxy\_arr[0][i] + 2.5\*xxy\_arr[1][i]

    arr\_y[i] = y2

    print(arr\_y[i])

    i = i + 1

**#question 1c**

#e = arr[2][i] - arr\_y[i]

i = 0

e = 0

print ("-----Q1c---------")

while i <xxy\_arr.shape[1]:

    e = xxy\_arr[2][i] - arr\_y[i]

    #print ("y[2][0]:",arr[2][0])

    #print("e:",e)

    arr\_e[i] = e

    print ("e:",arr\_e[i])

    i = i + 1

**#question 1d**

hist\_e = plt.hist(arr\_e)

plt.xlim(-3,3)

plt.xlabel('Residuals\_e')

plt.ylim(0,4)

plt.ylabel('Frequency')

plt.title('Histogram of Residuals\_e')

plt.show()

**#question 1e**

plt.xlabel('Y2')

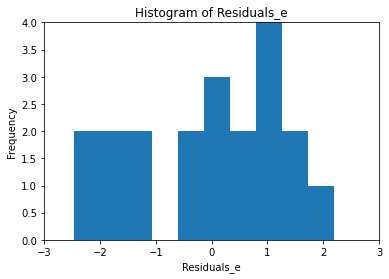
plt.ylabel('Residuals\_e')

plt.scatter(arr\_y, arr\_e)

plt.title("Scatter plot of Residual\_e against Y2")

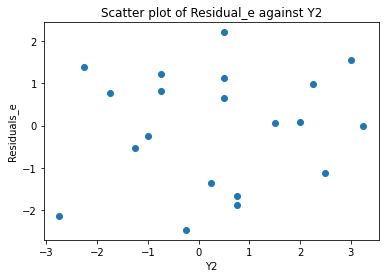
plt.show()

**(d)**



We agreed with the normality assumption with zero mean and the checking of constant variance is not needed. There is a normal distribution between <0 and >-1 to >2 and <3 based on the resulting histogram with a ‘outlier’ between <-2 to <-1.

**(e)**



We agreed that the constant variance assumption is valid when analysing the resulting scatter plot as there is no form of pattern that can be identified and the value  of Residual\_e does not go up or down with the Y2.

**Question 2**

**# Question 2a**

# import pandas

import pandas as pd

# read & convert csv file to DataFrame

class\_df = pd.read\_csv("class.csv")

# convert to NaN values for missing data

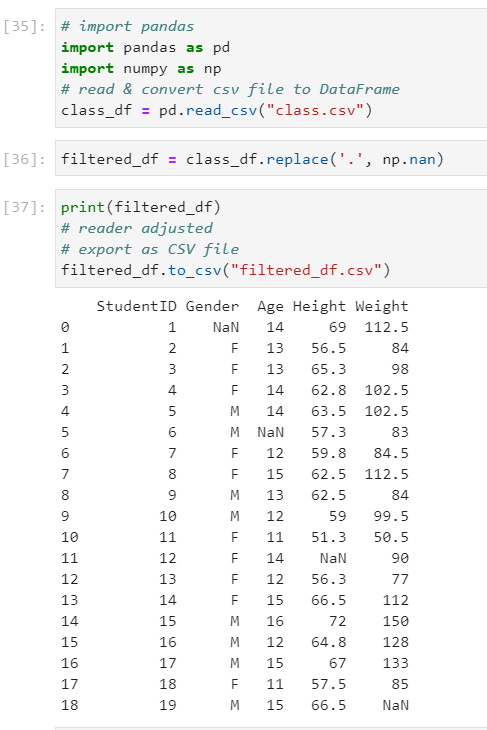
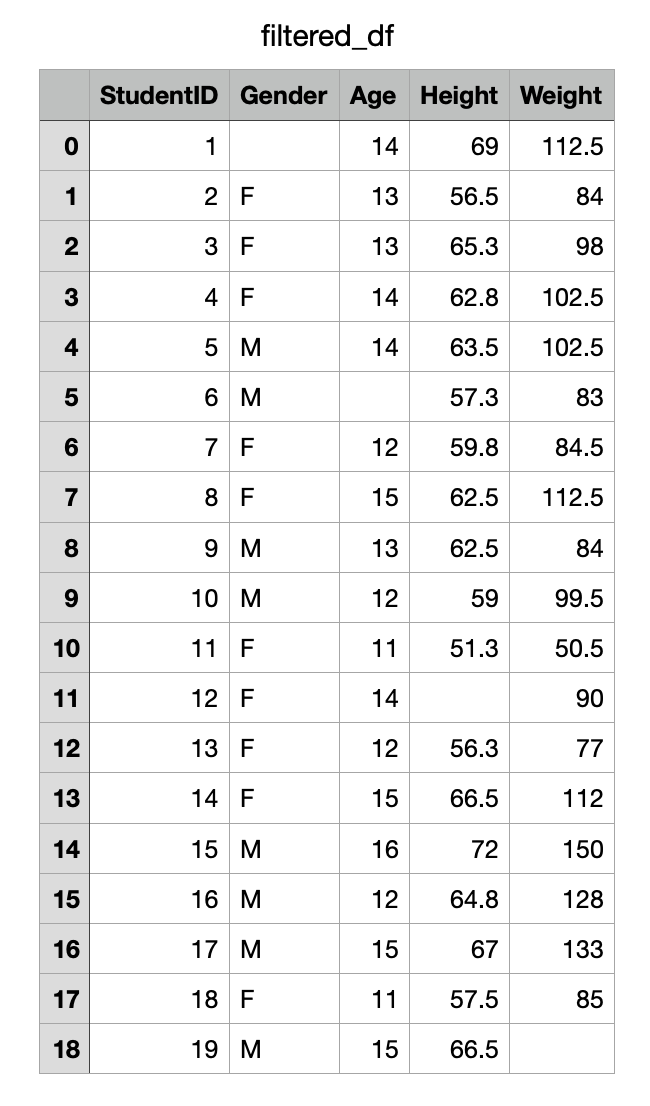
import numpy as np

filtered\_df = class\_df.replace('.', np.nan)

# reader adjusted

# export as CSV file

filtered\_df.to\_csv("filtered\_df.csv")  
  
Output for Jupyter lab:   Output for exported csv file

This is under the assumption that the data file has been placed into the same directory as the working jupyter lab file (directory) so that the program is able to pick up and import the data file.  
Additionally, the data is then exported as a CSV file to check that the data can still be read.

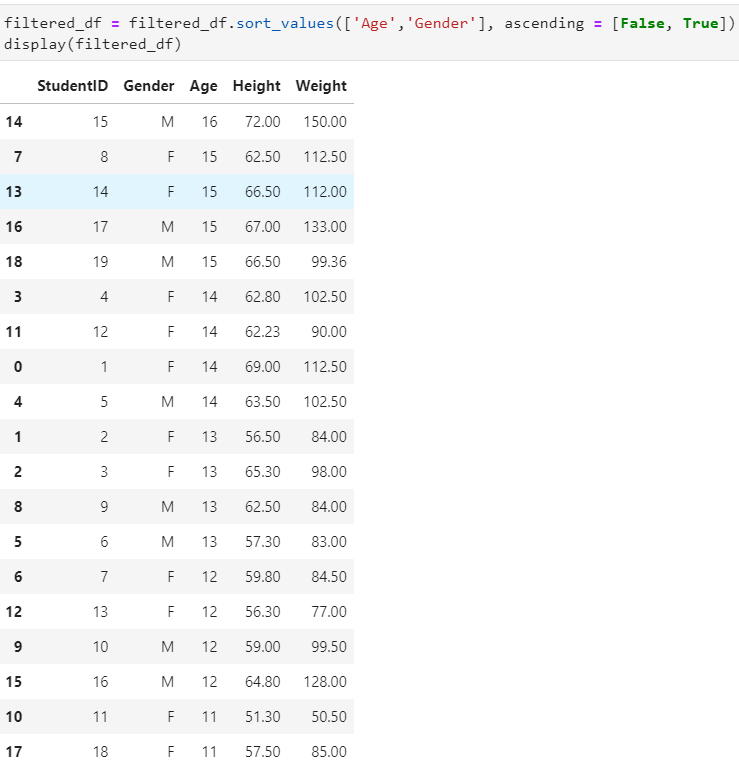
**# Question 2b**

# sort data by age in descending order

# sort data by gender in ascending order

filtered\_df = filtered\_df.sort\_values(['Age','Gender'], ascending = [False, True])

display(filtered\_df)  
  
Output for Jupyter lab:



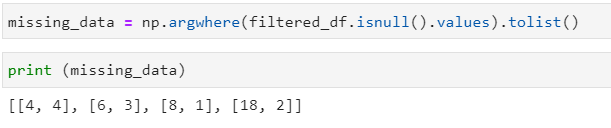
The outputs above are shown after the sorting has been completed by first in a descending order of the data field “age” and then by “gender” which data records containing “F” are shown first due to alphanumeric order.

**# Question 2c**

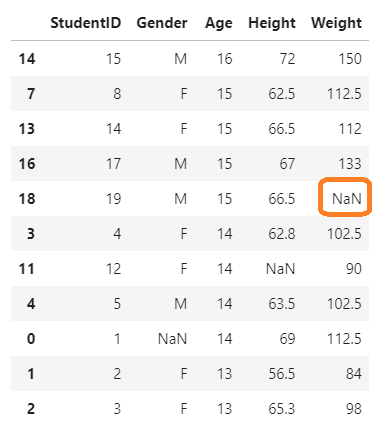
# Print rows and columns where NaN data are found

missing\_data = np.argwhere(filtered\_df.isnull().values).tolist()

print(missing\_data)

Output for jupyter lab:  


The reported locations can be interpret as follows:  
The first number refers to the row number and the second number refers to the column number.   
Taking the first set of data as an example (4,4), you can find the missing data value as highlighted in the data frame.   
Since the starting is numbered off with 0, the “fifth” row and “fifth” column is maintained with the NaN value as reported. (highlighted in orange in the screenshot below)



**# Question 2d**

#change datatype from object to float

filtered\_df['Height'] = filtered\_df['Height'].astype(float)

filtered\_df['Weight'] = filtered\_df['Weight'].astype(float)

# replace missing value

# gender - highest frequency

# age - median

# height - mean

# weight - mean

for column in ["Gender"]:

    filtered\_df[column].fillna(filtered\_df[column].mode(), inplace=True)

for column in ["Age"]:

    filtered\_df[column].fillna(filtered\_df[column].median(), inplace=True)

#Change age to int to not show decimal places

filtered\_df['Age'] = filtered\_df['Age'].astype(int)

# Find the mean height

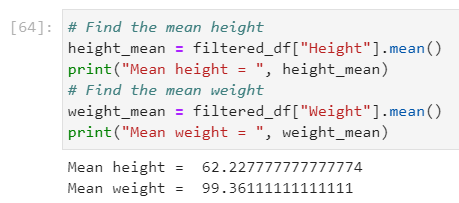
height\_mean = filtered\_df["Height"].mean()

print("Mean height = ", height\_mean)

# Find the mean weight

weight\_mean = filtered\_df["Weight"].mean()

print("Mean weight = ", weight\_mean)



# limit results to 2 decimal places

pd.set\_option('precision', 2)

# Replace NaN value with mean height

for column in ["Height"]:

    filtered\_df[column].fillna(value = height\_mean , inplace=True)

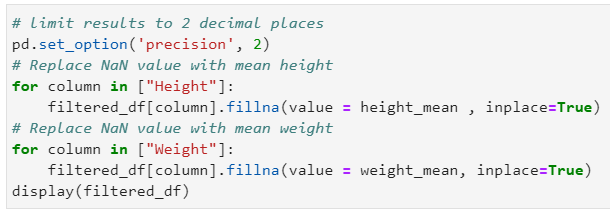
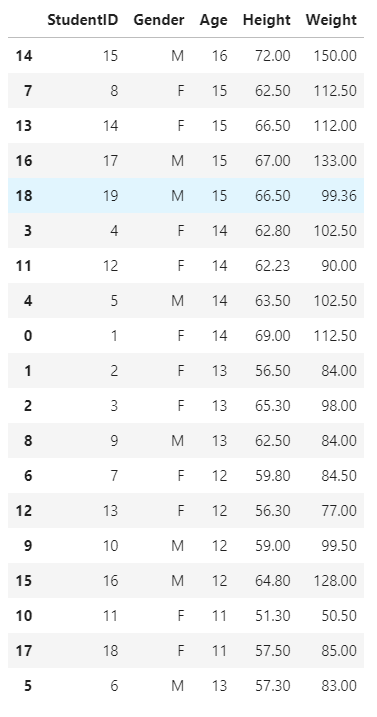
# Replace NaN value with mean weight

for column in ["Weight"]:

    filtered\_df[column].fillna(value = weight\_mean , inplace=True)

display(filtered\_df)

Output from jupyter lab:

**# Question 2e**

#select age,height and weight columns to check for outliers

final\_df = filtered\_df[["Age","Height","Weight"]]

print (final\_df)

#Identifying the interquantile range

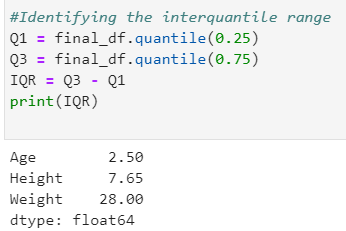
Q1 = final\_df.quantile(0.25)

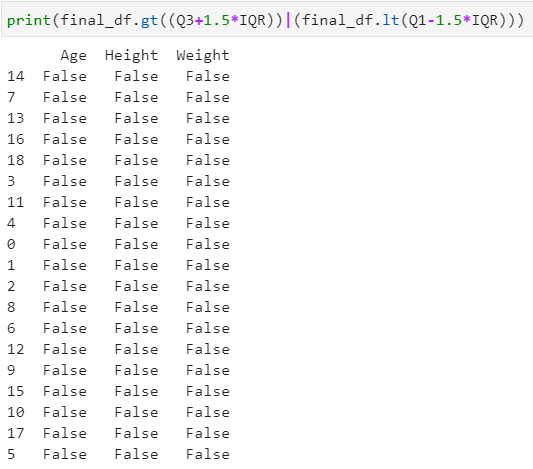
Q3 = final\_df.quantile(0.75)

IQR = Q3 - Q1

print(IQR)

print(final\_df.gt((Q3+1.5\*IQR))|(final\_df.lt(Q1-1.5\*IQR)))





The data frame is further filtered to remove the following columns:  
1.“Student ID”

Student ID is a unique number that is assigned to the data entry in a running sequence;there is no point in identifying any outliers for this data set.

2. “Gender”

Gender is a categorical data type which is either “M” or “F” and that there were checks performed to ensure that there is no missing data and if any, the entry has been replaced. Hence, there is no point in identifying outliers for this data column.

For the output above, if the result returned is “True”, it indicates that the entry is likely to be an outlier. However, for this result, there are no entries indicated with “True”.

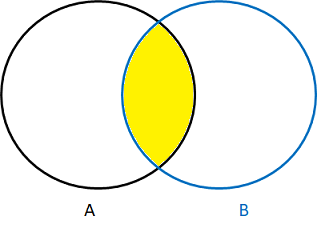
Statistically, we use the formula 1.5 x the Interquartile range to obtain the result and either add it to Quartile 3 to find outliers that are larger or minus it to Quartile 1 to find outliers that are smaller. Since the data values are within this range, we can conclude that there are no strong indications of outliers in this dataset.

It is also noted that in the remaining data columns, there were missing data values that were replaced with either mode, median or mean in this dataset.

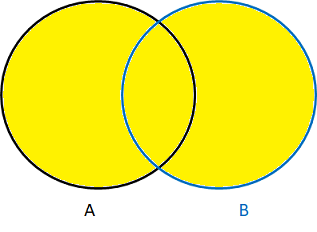
**Question 3**

We can illustrate the differences between inner and outer join by the use of a Venn diagram. For example, taking two dataframes defined as A and B which are distinctly different (A is not equal to B).

For inner join, it can be considered as the intersection of the Venn diagram (marked in yellow). If both dataframes are of the same shape, it will only retain the values from mutual rows or columns from both dataframes. If both dataframes are not of the same shape, it will only retain the values from the mutual rows and columns.



For outer join, it can be considered as the union of the Venn diagram (marked in yellow). Regardless of the shape of both dataframes, it will retain all the values and combine values if they have mutual rows and/or columns. If no value exists for the combined row and column, it will be shown as “NaN”.



We can perform inner or outer join by first importing the pandas package in our program. After which, we can use the concat() function for both datasets, specifying to join either by inner or outer. The axis can also be defined for the desired way of concatenation.

[200 words]