Logo

Description automatically generated with low confidence

**ANL252**

**Python for Data Analytics**

**Tutor-Marked Assignment**

**January 2022 Presentation**

|  |  |
| --- | --- |
| **Name** | **PI Number** |
| Chloe Tan Ying Yin | W2082681 |

**Tutorial Group: TG09**

**Instructor’s Name: Professor KUMAR MUNISH**

**Submission Date: 14 AUG 2022**

“I declare that this assignment is my own work, unless otherwise acknowledged or credited by appropriate referencing. I have read and abided by the SUSS Honor Code and I am aware of the penalties associated with plagiarism and collusion listed in the SUSS Student Handbook.”

**Table of contents**

[Question 1(A) 2](#_Toc111339316)

[Excel Chart 1 : Box and Whisker chart 2](#_Toc111339317)

[Excel Chart 2 : Combo chart 4](#_Toc111339318)

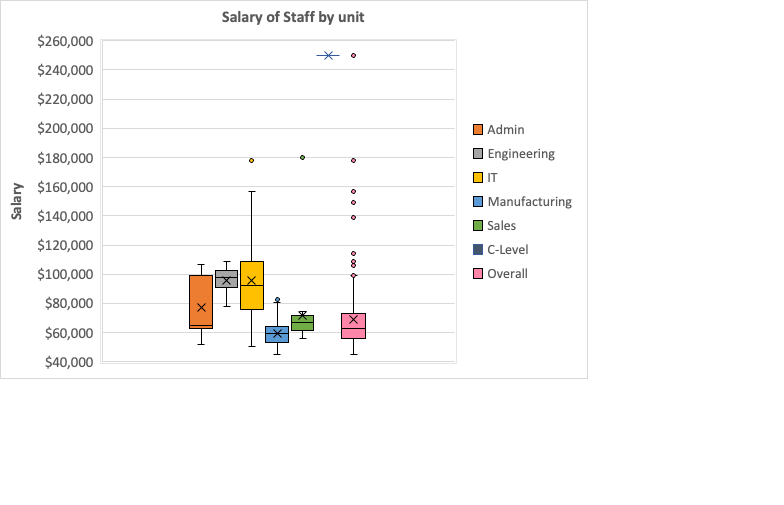
[Question 1(B) 6](#_Toc111339319)

[Question1(C) 15](#_Toc111339320)

[Question1(D) 17](#_Toc111339321)

# Question 1(A)

## Excel Chart 1 : Box and Whisker chart



Descriptive statistics:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Unit** | **Average Salary** | **3rd Quartile** | **Median** | **1st Quartile** | **Upper Whisker** | **Lower Whisker** |
| Admin | $77,315 | $96,199 | $64,520 | $63,002 | $106,367 | $51,920 |
| C-Level | $250,000 | - | - | - | - | - |
| Engineering | $95,867 | $92,989 | $97,470 | $90,583 | $108,987 | $83,363 |
| IT | $95,551 | $107,226 | $92,328 | $76,029 | $150,290 | $50,178 |
| Manufacturing | $59,413 | $64,246 | $59,472 | $53,018 | $80,512 | $45,046 |
| Sales | $71,427 | $71,707 | $67,030 | $61,809 | $74,326 | $55,875 |
| Overall | $69,118 | $72,992 | $63,001 | $55,722 | $97,999 | $45,046 |

Median : The unit with the highest median is C-level, at $250,000 and the lowest median is manufacturing, at $59,472. All units except for the manufacturing unit have medians above the overall median ($63,001).

Dispersion : There are varying interquartile ranges for each unit, with the Admin Unit being the highest ($63,002 - $96,199 ) and C-level being the lowest ($250,000-$250,000).

Skewness : The overall salary distribution is slightly positively skewed. The Admin unit has the largest positive skew, which shows that the staff within the admin unit getting a higher salary is more varied.

Outliers : The IT and manufacturing units have very few outliers but they are fairly close to the interquartile ranges, while the sales unit has an outlier that is a significantly out of the interquartile range.

Overall, the insight that can be gathered from the chart is that the staff working in the C-level unit get a higher salary as compared to the other units. However, it is important to note that there is only one staff working in the C-level unit. If the C-level unit is regarded as an outlier, the unit that tends to receive the highest salary is IT. While staff in the manufacturing unit tend to get a lower salary. The average salary range for all staff in this organisation is from $55,722 to $72,992.

## Excel Chart 2 : Combo chart

Summarised Table :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Unit** | **Count** | **Average Engagement Survey Score** | **Total Number of days Absent** | **Average number of days Absent** |
| Admin | 7 | 4.64 | 55 | 8 |
| C-Level | 1 | 4.83 | 10 | 10 |
| Engineering | 10 | 4.05 | 94 | 9 |
| IT | 41 | 4.24 | 435 | 11 |
| Manufacturing | 169 | 4.18 | 1701 | 10 |
| Sales | 22 | 3.99 | 283 | 13 |
| Overall | 250 | 4.19 | 2578 | 10 |

From the combo chart generated, we can see that the higher the number of average days absent, the lower the average score on the engagement survey. This means that the number of days absent and engagement scores are **negatively correlated**.

The unit with the highest average number of days absent is Sales, at 13 days (to the nearest whole number) . The sales unit also has the lowest average engagement score, at 3.99. On the other hand, the admin unit has the lowest number of average days absent at 8 days (to the nearest whole number) with the second highest engagement score at 4.64. If the C-level unit is considered as an outlier, due to the unit only consisting of one staff, it further proves the negative correlation of the number of days absent and the engagement scores.

# Question 1(B)

**#Question1(B)**

**#Python Chart 1 : Box and Whisker chart**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.patches as mpatches

from numpy import percentile

from matplotlib.lines import Line2D

from matplotlib import ticker

# Read File

DF = pd.read\_csv(r"/Users/cats/Desktop/ANL252/python codes /Untitled Folder/TMA\_Data.csv")

# initialise Variables

data = []

legen\_p = []

legen = ["Admin ","Engineering","IT","Manufacturing","Sales","C-Level"]

colour\_p = []

# color dictionary for reference

color\_dict = {

"Admin " :"#ED7D31",

"Engineering" :"#A5A5A5",

"IT" :"#FEC100",

"Manufacturing":"#5B9BD5",

"Sales" :"#6CA344",

"C-Level" :"#44556A",

"Overall" :"#FF85A9",

}

for i,unit in enumerate(legen):

colour\_p.append(color\_dict[str(unit)])

data.append(list(DF[DF["Unit"]==unit]["Salary"]))

legen\_p.append( mpatches.Patch(color=color\_dict[str(unit)], label=str(unit)) )

# Overall data

data.append(list(DF["Salary"]))

legen\_p.append( mpatches.Patch(color=color\_dict["Overall"], label="Overall") )

colour\_p.append("#FF85A9")

# ====================== plotting figure ===============================

# blank canvas

fig = plt.figure(figsize = (10, 7))

# Creating axes instance

ax = fig.add\_axes([0, 0, 1, 1])

# Creating plot

bp = ax.boxplot(data,patch\_artist=True,whis=2)

# figure settings

plt.grid(True,axis="y")

plt.ylabel("\nSalary",fontweight="bold",fontsize=12,labelpad=20)

plt.title("\nSalary of Staff by unit",fontweight='bold',fontsize=15, y=1.03,)

plt.yticks(list(np.linspace(40000,260000,12)))

ax.yaxis.set\_major\_formatter(ticker.FormatStrFormatter("$%d"))

plt.tick\_params( axis='x', which='both', bottom=False, top=False, labelbottom=False)

plt.legend(handles=legen\_p, bbox\_to\_anchor=(1.01, 0.6),frameon=False)

# iterate through the boxplots and set its color and outlier colors

for i,plot in enumerate(bp["boxes"]):

bp["boxes"][i].set\_facecolor( colour\_p[i] )

bp["medians"][i].set\_color('black')

bp["fliers"][i].set\_markerfacecolor(colour\_p[i])

# Calculate and plot mean data of Salary from each Unit

mean\_dat = []

for i,unit in enumerate(legen):

df\_u = DF[DF["Unit"]==unit]

mean\_dat.append(round(df\_u["Salary"].mean(),2))

plt.scatter(

x = i+1,

y = round(df\_u["Salary"].mean(),2),

color = 'black',zorder=3,marker='x',s=80

)

# Calculate and plot mean data of Overall Salary

plt.scatter(

x = i+2,

y = round(DF["Salary"].mean(),2),

color = 'black',zorder=3,marker='x',s=80

)

# show plot

plt.show()

# ============ Table Summary ============

# initialise variables

units = ["Admin ","C-Level","Engineering","IT","Manufacturing","Sales"]

# Data dictionary that will be used for Dataframe making

df\_dict = {

"Unit": units,

"Average Salary": [],

"3rd Quartile": [],

"Median": [],

"1st Quartile": [],

"Upper Whisker": [],

"Lower Whisker": [],

}

# iterate through the units available and calculate the quartiles

for unit in units:

df\_u = DF[DF["Unit"]==unit]["Salary"]

quartiles = percentile(df\_u, [25, 50, 75])

iqr = quartiles[2] - quartiles[0]

upper\_whisker = df\_u[df\_u<=quartiles[2]+1.5\*iqr].max()

lower\_whisker = df\_u[df\_u>=quartiles[0]-1.5\*iqr].min()

df\_dict["Average Salary"] = df\_dict["Average Salary"] + ["$"+str(int(round(df\_u.mean(),0)))]

# Store the values into dictionary

if str(unit) != "C-Level":

df\_dict["3rd Quartile"] = df\_dict["3rd Quartile"] + ["$"+str(int(round(quartiles[2],0)))]

df\_dict["Median"] = df\_dict["Median"] + ["$"+str(int(round(quartiles[1],0)))]

df\_dict["1st Quartile"] = df\_dict["1st Quartile"] + ["$"+str(int(round(quartiles[0],0)))]

df\_dict["Upper Whisker"] = df\_dict["Upper Whisker"] + ["$"+str(int(round(upper\_whisker,0)))]

df\_dict["Lower Whisker"] = df\_dict["Lower Whisker"] + ["$"+str(int(round(lower\_whisker,0)))]

else:

# if "C-Level" Unit will fill in all fields as blanks

df\_dict["3rd Quartile"] = df\_dict["3rd Quartile"] + ["-"]

df\_dict["Median"] = df\_dict["Median"] + ["-"]

df\_dict["1st Quartile"] = df\_dict["1st Quartile"] + ["-"]

df\_dict["Upper Whisker"] = df\_dict["Upper Whisker"] + ["-"]

df\_dict["Lower Whisker"] = df\_dict["Lower Whisker"] + ["-"]

# calculate Quartile values for overall Salary

quartiles = percentile(DF["Salary"], [25, 50, 75])

iqr = quartiles[2] - quartiles[0]

upper\_whisker = DF["Salary"][DF["Salary"]<=quartiles[2]+1.5\*iqr].max()

lower\_whisker = DF["Salary"][DF["Salary"]>=quartiles[0]-1.5\*iqr].min()

# Store value in dictionary

df\_dict["Unit"] = df\_dict["Unit"] + ["Overall"]

df\_dict["Average Salary"] = df\_dict["Average Salary"] + ["$"+str(int(round(DF["Salary"].mean(),0)))]

df\_dict["3rd Quartile"] = df\_dict["3rd Quartile"] + ["$"+str(int(round(quartiles[2],0)))]

df\_dict["Median"] = df\_dict["Median"] + ["$"+str(int(round(quartiles[1],0)))]

df\_dict["1st Quartile"] = df\_dict["1st Quartile"] + ["$"+str(int(round(quartiles[0],0)))]

df\_dict["Upper Whisker"] = df\_dict["Upper Whisker"] + ["$"+str(int(round(upper\_whisker,0)))]

df\_dict["Lower Whisker"] = df\_dict["Lower Whisker"] + ["$"+str(int(round(lower\_whisker,0)))]

# make dictionary into dataframe and display

pd.DataFrame(df\_dict)

**#Python Chart 2 : Combo chart**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.patches as mpatches

from numpy import percentile

from matplotlib.lines import Line2D

# Initialise variables

units = ["Admin ","C-Level","Engineering","IT","Manufacturing","Sales"]

colour\_p = []

# Dictionary that will be made into dataframe later

df\_dict\_2 = {

"Unit":units,

"Count":[],

"Average Engagement Survey Score":[],

"Total Number of days Absent":[],

"Average number of days Absent":[],

"Average\_absence\_raw": [],

}

# color reference dictionary

color\_dict = {

"Admin " :"#ED7D31",

"Engineering" :"#A5A5A5",

"IT" :"#FEC100",

"Manufacturing":"#5B9BD5",

"Sales" :"#6CA344",

"C-Level" :"#44556A",

"Overall" :"#FF85A9",

"Line" :"#696969"

}

# summarise data first

for unit in units:

df\_u = DF[DF["Unit"]==unit]

# calculate and store values into dictionary

df\_dict\_2["Count"] = df\_dict\_2["Count"] + [str(len(df\_u))]

df\_dict\_2["Average Engagement Survey Score"] = df\_dict\_2["Average Engagement Survey Score"] + [round(df\_u["Survey"].mean(),2)]

df\_dict\_2["Total Number of days Absent"] = df\_dict\_2["Total Number of days Absent"] + [df\_u["Absence"].sum()]

df\_dict\_2["Average number of days Absent"] = df\_dict\_2["Average number of days Absent"] + [round(df\_u["Absence"].mean())]

# For easier plotting later, raw 2DP values

df\_dict\_2["Average\_absence\_raw"] = df\_dict\_2["Average\_absence\_raw"] + [round(df\_u["Absence"].mean(),2)]

colour\_p.append(color\_dict[str(unit)])

# Overall value calculations (row)

df\_dict\_2["Unit"] = df\_dict\_2["Unit"] + ["Overall"]

df\_dict\_2["Count"] = df\_dict\_2["Count"] + [str(len(DF))]

df\_dict\_2["Average Engagement Survey Score"] = df\_dict\_2["Average Engagement Survey Score"] + [round(DF["Survey"].mean(),2)]

df\_dict\_2["Total Number of days Absent"] = df\_dict\_2["Total Number of days Absent"] + [df\_u["Absence"].sum()]

df\_dict\_2["Average number of days Absent"] = df\_dict\_2["Average number of days Absent"] + [round(DF["Absence"].mean())]

# For easier plotting later, raw 2DP values

df\_dict\_2["Average\_absence\_raw"] = df\_dict\_2["Average\_absence\_raw"] + [round(DF["Absence"].mean(),2)]

# Makes Dictionary into dataframe and store into variable DF\_2

DF\_2 = pd.DataFrame(df\_dict\_2)

# Initialise variables for plots

units\_2 = ["Admin ","Engineering","C-Level","Manufacturing","IT","Sales"]

bar\_list = []

line\_list = []

colour\_p = []

# blank canvas

fig2 = plt.figure(figsize = (10, 8))

# define axes

ax2 = fig2.add\_axes([0, 0, 1, 1])

# iterate through the units list and store values that will be used to plot the combo chart lines and bar graph

for unit in units\_2:

df\_u = DF\_2[DF\_2["Unit"]==unit]

bar\_list.append( df\_u["Average\_absence\_raw"].item() )

line\_list.append( df\_u["Average Engagement Survey Score"].item() )

colour\_p.append(color\_dict[str(unit)])

# Plot bar graph

bar = ax2.bar(units\_2,bar\_list,width=0.45, color=colour\_p, zorder=3)

# iterate through the bar values and place it as a text on the bar chart base

for i, val in enumerate(bar\_list):

if val == 10.0: ax2.text(i-0.1-(i\*0.01), 0.2, str(val), color='white',fontsize=12,fontweight='bold')

else: ax2.text(i-0.1-(i\*0.01), 0.2, str(val), color='black',fontsize=12,fontweight='bold')

# Define Y axis, title, grid settings

plt.yticks(list(np.linspace(0,14,8)))

plt.ylabel("\nAverage Absences (days)",fontsize=15,labelpad=15)

plt.title("\nAverage number of Absences Vs Average Engagement by department",fontweight='bold',fontsize=15, y=1.03)

plt.grid(True,axis='y')

ax2.grid(True,axis='y',zorder=0)

# Combo Chart part

# clone the x-axis and plot line

ax3 = ax2.twinx()

ax3.plot(units\_2,line\_list,color=color\_dict["Line"],linewidth=3)

# define the y-axis values and y axis

plt.yticks(list(np.linspace(0,6,7)))

plt.ylabel("\nAverage Engagement (Survey)",fontsize=15,rotation=-90,labelpad=50)

# Iterate through the line values (avg survey score (engagement)) and write the values on the chart

for i, val in enumerate(line\_list):

ax3.text(i-0.1-(i\*0.01), val+0.15, str(val), color='black',fontsize=12,fontweight='bold')

# Define legend settings and place it at the base of the chart

legen = [mpatches.Patch(color=color\_dict["Manufacturing"], label="avg number of days absent"),

Line2D([0], [0], color=color\_dict["Line"], lw=3, label='avg survey score (engagement)')]

plt.legend(handles=legen,ncol=2,bbox\_to\_anchor = (0.772, -0.05),frameon=False)

# displays the chart and table accordingly

plt.show()

DF\_2.iloc[:,:-1]

**#Python Chart 1 : Box and Whisker chart (output)**

Chart, box and whisker chart

Description automatically generated

Table

Description automatically generated

**#Python Chart 2 : Combo chart (output)**

Chart, bar chart

Description automatically generated

Table

Description automatically generated

# Question1(C)

import csv

from datetime import date

# function for getting particular row of staff details from csv file

def getRow(rowNo):

with open('/Users/cats/Desktop/ANL252/python codes /Untitled Folder/TMA\_Data.csv', 'r') as file:

reader = csv.reader(file)

next(reader) # to skip the header row

# to skip rows and get to the particular row that is needed

for count in range(rowNo):

next(reader)

row = next(reader) # returning the particular row that is needed

return row

# function to format the Join date and Left Date

def formatDate(dateString):

countSlash = 0 #counter for the / in the date string, used to differentiate the day, month and year

day = 0

month = 0

year = 0

value = "" # variable to temporarily hold the value for day, month or year

y = 0

for y in range(len(dateString)):

if countSlash == 2: # countSlash == 2 means that the remaining of the date string forms the year value

value = dateString[y:]

year = int(value)

countSlash = 0

else:

if dateString[y] == "/": # used to differentiate between day, month and year values

if countSlash == 0:

month = int(value)

value = ""

countSlash += 1

elif countSlash == 1:

day = int(value)

value = ""

countSlash += 1

else:

value += dateString[y] # concatenate the string values until it reaches the next / in the date string

return date(year, month, day)

x = 0

listOfDurations = [] # a list to store the duration of service of all staff

while x != 250:

staffRow = getRow(x)

joinDate = staffRow[7]

if staffRow[8] == "": # assign "1st May 2022 for rows that do not have Left date"

leftDate = "5/1/2022"

else:

leftDate = staffRow[8]

joinDate = formatDate(joinDate)

leftDate = formatDate(leftDate)

duration = leftDate - joinDate

# add 1 to include the end date

durationInDays = int(duration.days) + 1

durationInYears = round((durationInDays / 365), 1)

listOfDurations.append(durationInYears) #append duration to list

x += 1 # reiterate the while loop

print("Maximum length of service is", str(max(listOfDurations)), "year(s)")

print("Minimum length of service is", str(min(listOfDurations)), "year(s)")

# calculation for average length of service

sum = 0

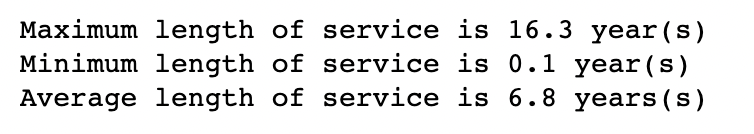
for duration in listOfDurations:

sum += duration

averageLength = round(sum / len(listOfDurations), 1)

print("Average length of service is", str(averageLength), "years(s)")

output:



# Question1(D)

import csv

# function to read the csv file

def loadStaffList():

with open('/Users/cats/Desktop/ANL252/python codes /Untitled Folder/TMA\_Data.csv', 'r') as file:

reader = csv.reader(file)

next(reader)

for row in reader:

staffNameList.append(row[1]) # row[1] will return the staff name and append it to the list

staffNameList = []

queryName = (input("Enter staff's full name to search (Enter Q to quit): ")).title()

while queryName != "Q":

loadStaffList()

if queryName == "Q": # for user to quit the program

break

elif queryName in staffNameList: #check if name input by user is a staff

print(queryName, "is a staff in the organization")

else:

print(queryName, "is not a staff in the organization")

queryName = (input("Enter staff's full name to search (Enter Q to quit): ")).title()

print("Program has been stopped")

output:

