Logo

Description automatically generated

**ANL252**

**Python for Data Analytics**

**Tutor-Marked Assignment**

**July 2022 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Timothy Yap** | **K2111660** |

**Tutorial Group: T09**

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

**Submission Date: 14 August 2022**

Contents

[Q1(a) 3](#_Toc111390823)

[Q1(b) 6](#_Toc111390824)

[Q1(c) 11](#_Toc111390825)

[Q1(d) 13](#_Toc111390826)

[References 15](#_Toc111390827)

## Q1(a)

An important statistic that may be of interest to both the organisation’s HR and its employees as well is that of payroll. To understand the distribution and variability in the salary of the organisation’s employees, we can compute both the mean salary and standard deviation aggregated at a Unit level.

Both statistics can be computed via Excel’s “AVERAGEIF” and “STDEV.P” functions. The results are appended in the table below:

|  |  |  |
| --- | --- | --- |
|  | **Mean Salary** | **Standard Deviation of Salary** |
| Admin | $77,315.29 | $19,983.70 |
| C-level | $250,000.00 | $0.00 |
| Engineering | $95,867.00 | $9,113.02 |
| IT | $95,550.76 | $28,782.06 |
| Manufacturing | $59,412.91 | $8,520.51 |
| Sales | $71,426.59 | $24,315.80 |
| *Organisational Average (excluding C-level)* | *$68,392.06* | *$21,495.68* |

It should be noted that there is only one C-level executive in the company. Hence, the salary standard deviation in this category is $0. As such, this data point may not be exactly meaningful. Nonetheless, it is retained in the above table for the sake of completeness.

To obtain a simple benchmark, we can compute both the mean salary and standard deviation of salary at an organizational level. The salary of the C-level executive is excluded due for two reasons – 1) it is not representative of the salary of most workers in the organization, 2) due to how high the salary is, it will skew the standard deviation statistic.

The unit with the lowest mean salary is Manufacturing, having a mean salary of $59,412.91. However, it has the lowest standard deviation of all units at $8,520.51. This indicates that there is lesser variation in the salaries of manufacturing employees.

The unit with the highest mean salary (excluding C-level) is IT, having a mean salary of $95,550.76. However, it has the highest standard deviation of all units at $28,782.06. This indicates that there is a higher variation in the salaries of manufacturing employees.

Graphically, we can use excel to plot a comparative grouped vertical bar chart of Mean of Standard Deviation of Salary aggregated at a unit level:

In organisations, there is a gradual but certainly increasing emphasis on diversity and inclusion. This is especially true when it comes to an organisation’s employment practice. To this end, a particular area of emphasis is that of gender diversity. Studies conducted has revealed that gender discrimination, more often not, exists in workplaces. In many organisations, there are more male employees than female employees. While in certain countries such discriminatory hiring practice is illegal, nonetheless they are prevalent enough to be of particular concern. In this regard, many employee-centric companies have made a concerted effort to make gender diversity a key thrust in their HR practice. One way this is done is through equal representation whereby approximately an equal number of female and male employees are hired in a workplace. In this regard, we can compute the number of employees aggregated at both a gender and unit level.

Both statistics can be computed using Excel’s “COUNTIFS” functions. The results are appended in the table below:

|  |  |  |
| --- | --- | --- |
|  | **Female** | **Male** |
| Admin | 4 | 3 |
| C-level | 1 | 0 |
| Engineering | 5 | 5 |
| IT | 18 | 23 |
| Manufacturing | 105 | 64 |
| Sales | 11 | 11 |
| *Total* | *144* | *106* |

Overall, the organisation hires more female than males employees (144 vs 106).

At a unit level, the organization approximately hires an equal proportion of female and male employees. The only two exceptions are IT and Manufacturing. In IT, there are marginally more male employees than female employees (18 vs 23). In Manufacturing, there is a significantly higher number of female employees than male employees (105 vs 64).

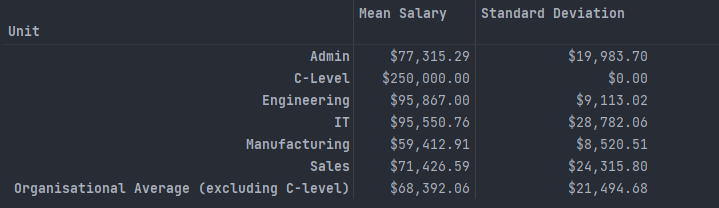
Graphically, we can use excel to plot a comparative grouped horizontal bar chart of the number of female and male employees aggregated at a unit level:

## Q1(b)

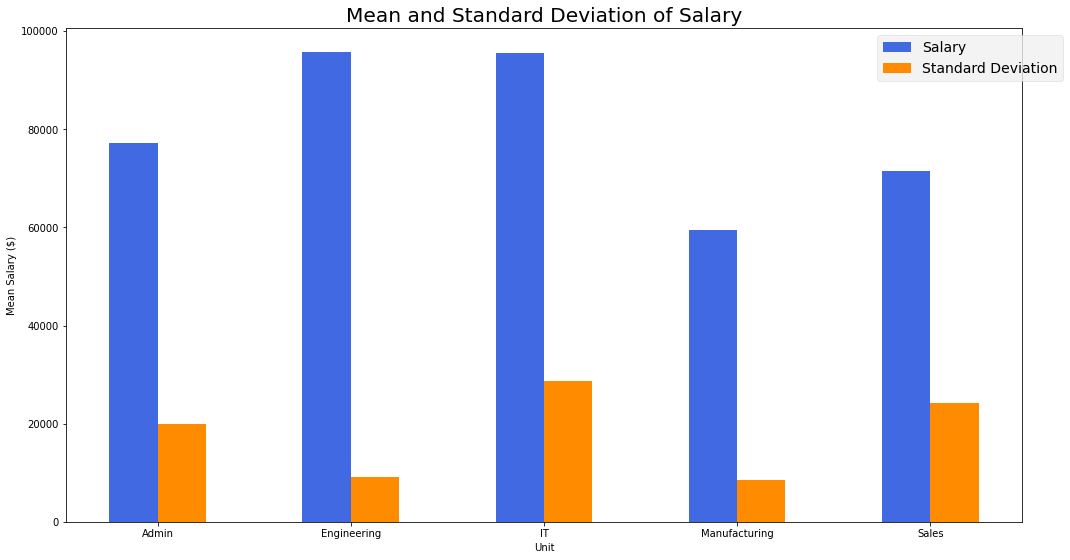
To read and manipulate the csv file in Python, the Pandas library can be imported and then the csv file read as a DataFrame.

To generate the necessary graphs, the Matplotlib library is imported and thereafter the necessary graphs generated.

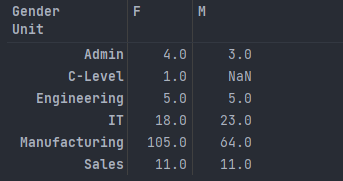
After some data sorting, grouping and transformation, we can arrive at the following data table with the mean and standard deviation of employees’ salary aggregated at a unit level:



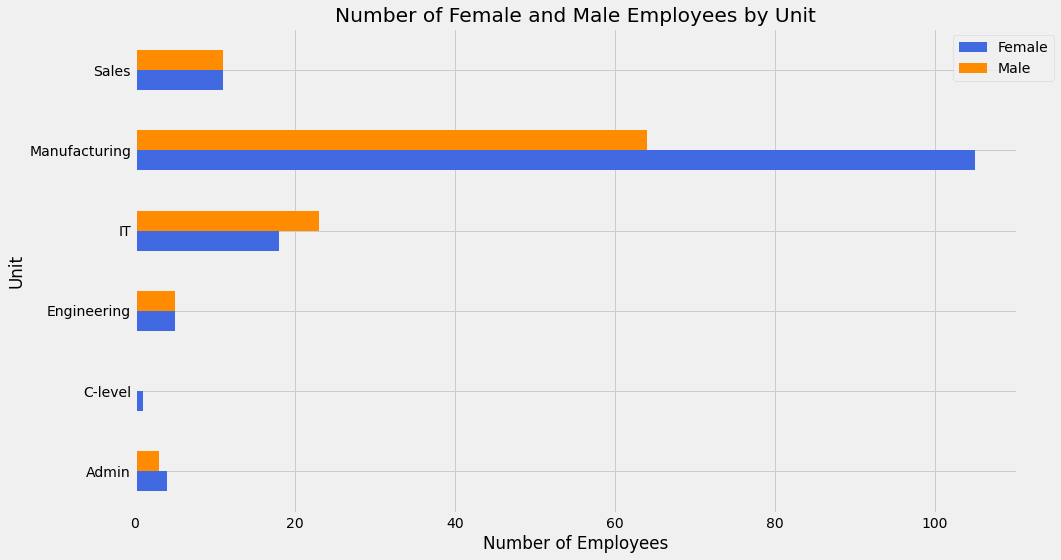
Using this DataFrame, we can make use of Matplotlib to plot the following comparative grouped vertical bar chart with the mean and standard deviation of employees’ salary aggregated at a unit level:



Similarly for the number of female and male employees aggregated at a unit level, we can sort, group and transform the DataFrame and arrive at the following table:



Using this DataFrame, we can make use of Matplotlib to plot the following comparative grouped horizontal bar chart with the number of female and male employees aggregated at a unit level:



The python code with documentation is appended below:

*#To import the necessary libraries that are required for the code to run  
import* pandas *as* pd  
*import* numpy *as* np  
*import* matplotlib.pyplot *as* plt  
  
*#to read the CSV as a Pandas DataFrame*raw\_df = pd.read\_csv(r"C:\Users\timot\OneDrive\Documents\SUSS\ANL252 - Python\TMA\Raw dataset\TMA\_Data.csv")  
  
*#a copy of raw\_df is created so that it can be cleaned up and transformed for the purposes of 1(b)*df\_1b = raw\_df.copy()  
*#As 'ID' is a unique identifier for each employee, it is set as the index of the df\_1b DataFrame*df\_1b.set\_index("ID", inplace=*True*)  
  
*#---SALARY STATISTICS---  
  
#This is to aggregate the df\_1b DataFrame at a 'Unit' level*unit\_group = df\_1b.groupby("Unit")  
*#This computes the mean salary aggregated at a 'Unit' level, and then stores it in a DataFrame called salary\_output\_table*salary\_output\_table = unit\_group["Salary"].mean()  
*#This computes the population standard deviation aggregated at a 'Unit' level*salary\_std\_dev = unit\_group["Salary"].std(ddof=0)  
salary\_output\_table = pd.DataFrame(salary\_output\_table)  
salary\_output\_table.rename(columns={'Salary':'Mean Salary'}, inplace=*True*)  
  
*#To create a new column for standard deviation of salary in the salary\_output\_table DataFrame*salary\_output\_table["Standard Deviation"] = salary\_std\_dev  
  
*#This step serves to filter out the C-level employee record, and subsequently delete that row in the df\_1b DataFrame. The purpose is to compute the organisational mean and standard deviation of salary excluding that of the C-level employee*c\_level\_filt = (df\_1b["Unit"] == "C-Level")  
df\_1b.drop(index=df\_1b[c\_level\_filt].index, inplace=*True*)  
  
organisational\_average\_salary = df\_1b["Salary"].mean()  
organisational\_StdDev\_salary = df\_1b["Salary"].std(ddof=0)  
  
*#This is to add a new row in the salary\_output\_table DataFrame for the data on the Organisational mean and standard deviation of salary excluding that of the C-level employee*salary\_output\_table.loc['Organisational Average (excluding C-level)'] = [organisational\_average\_salary, organisational\_StdDev\_salary]  
  
*#This is to cast the format of the salary and standard deviation figures to be that of currency to 2 d.p.*salary\_output\_table['Mean Salary'] = salary\_output\_table['Mean Salary'].map('${:,.2f}'.format)  
salary\_output\_table['Standard Deviation'] = salary\_output\_table['Standard Deviation'].map('${:,.2f}'.format)  
print(salary\_output\_table)  
  
*#as the step above converted the salary and standard deviation figures to be strings, we will need to recast them to be a float again so that we can plot a graph later on. To simplify the code, parsing is done using regex*salary\_output\_table\_for\_graph = salary\_output\_table.copy()  
salary\_output\_table\_for\_graph.drop(index=['C-Level', 'Organisational Average (excluding C-level)'], inplace=*True*)  
salary\_output\_table\_for\_graph  
  
salary\_output\_table\_for\_graph['Mean Salary'] = salary\_output\_table\_for\_graph['Mean Salary'].replace({'\$': '', ',': ''}, regex=*True*).astype(float)  
salary\_output\_table\_for\_graph['Standard Deviation'] = salary\_output\_table\_for\_graph['Standard Deviation'].replace({'\$': '', ',': ''}, regex=*True*).astype(float)  
  
*#matplotlib is used to create a comparative grouped vertical bar chart. The parameters (e.g. title, colour, legend location, etc.) are specified accordingly. As there is no direct method to create a comparative bar chart in matplotlib, both bars have to be manually offset via the 'width' variable*plt.figure(figsize=(15,8))  
  
title = 'Mean and Standard Deviation of Salary'  
  
x\_index\_list = ['Admin','Engineering','IT','Manufacturing','Sales']  
x\_index\_values = np.arange(len(x\_index\_list))  
width = 0.25  
  
plt.bar(x\_index\_values-width/2, salary\_output\_table\_for\_graph['Mean Salary'], width=width, color='royalblue', label='Salary')  
plt.bar(x\_index\_values+width/2, salary\_output\_table\_for\_graph['Standard Deviation'], width=width, color='darkorange',label='Standard Deviation')  
  
plt.style.use('fivethirtyeight')  
  
plt.legend(bbox\_to\_anchor = (1.05,1))  
plt.xticks(ticks=x\_index\_values, labels=x\_index\_list)  
plt.xlabel('Unit')  
plt.ylabel("Mean Salary ($)")  
plt.title(title)  
  
plt.tight\_layout()  
  
plt.show()  
  
  
*#---GENDER STATISTICS---  
  
#This is to aggregate the df\_1b DataFrame at a 'Unit' and then 'Gender' level as new DataFrame, i.e. gender\_group*gender\_group = raw\_df.groupby(['Unit','Gender'])  
*#this serves to count the number of non-NA values in each row/column in the gender\_group DataFrame*gender\_count\_table = gender\_group.count()  
  
*#Due to the structure of the DataFrame and aggregation type, the gender\_group DataFrame is multiIndexed. For ease of plotting the bar chart later on, we can transform it to a DataFrame that has a single index via the unstack() function*gender\_count\_table = gender\_count\_table.unstack()  
gender\_count\_table = gender\_count\_table['Staff']  
print(gender\_count\_table)  
  
*#matplotlib is used to create a comparative grouped horizontal bar chart. The parameters (e.g. title, colour, legend location, etc.) are specified accordingly. As there is no direct method to create a comparative bar chart in matplotlib, both bars have to be manually offset via the 'height' variable*plt.figure(figsize=(15,8))  
  
title = 'Number of Female and Male Employees by Unit'  
  
y\_index\_list = ['Admin','C-level','Engineering','IT','Manufacturing','Sales']  
y\_index\_values = np.arange(len(y\_index\_list))  
  
height = 0.25  
  
plt.barh(y\_index\_values-height/2, gender\_count\_table['F'],height=height, color='royalblue', label='Female')  
plt.barh(y\_index\_values+height/2, gender\_count\_table['M '],height=height, color='darkorange', label='Male')  
  
plt.style.use('fivethirtyeight')  
  
plt.legend(bbox\_to\_anchor = (1.05,1))  
plt.yticks(ticks=y\_index\_values, labels=y\_index\_list)  
plt.xlabel('Number of Employees')  
plt.ylabel("Unit")  
plt.title(title)  
  
plt.tight\_layout()  
  
plt.show()

## Q1(c)

The minimum length of service rounded to 1 d.p. is 0.1 years

The maximum length of service rounded to 1 d.p. is 16.3 years

The average length of service rounded to 1 d.p. is 6.8 years

The python code with documentation is appended below:

*#To import the necessary libraries that are required for the code to run  
import* pandas *as* pd  
  
*#to read the CSV as a Pandas DataFrame*raw\_df = pd.read\_csv(r"C:\Users\timot\OneDrive\Documents\SUSS\ANL252 - Python\TMA\Raw dataset\TMA\_Data.csv")  
  
*#a copy of raw\_df is created so that it can be cleaned up and transformed for the purposes of 1(c)*df\_1c = raw\_df.copy()  
  
*#This is to convert the data 'JoinDate' and 'LeftDate' to a datetime datatype. This will allow us to compute the length of service later via a timedelta*df\_1c['JoinDate'] = pd.to\_datetime(df\_1c['JoinDate'])  
df\_1c['LeftDate'] = pd.to\_datetime(df\_1c['LeftDate'])  
*#This is to fill in the blank values in the LeftDate column (i.e. staff who are still in the organisation) with a value of 1 May 2022*df\_1c['LeftDate'].fillna(value=pd.to\_datetime('1/5/2022', format='%d/%m/%Y'), inplace=*True*)  
  
*#This computes the length of service of each staff (in days) as a timedelta object*df\_1c["Length of service"] = df\_1c["LeftDate"] - df\_1c["JoinDate"]  
  
*#As arithmetical operations cannot be performed on timedelta objects directly, the 'Length of service' column can be casted into an int16 data type, and thereafter divided by 365 days to derive the length of service in years*df\_1c['Length of service (in years)'] = df\_1c['Length of service'].dt.days.astype('int16', errors='ignore')/365  
df\_1c  
  
*#This computes the minimum length of service (rounded to 1 d.p.) and displays the output in a formatted string*min\_length\_of\_service = round(df\_1c['Length of service (in years)'].min(),1)  
print(f'The minimum length of service rounded to 1 d.p. is {min\_length\_of\_service} years')  
  
*#This computes the maximum length of service (rounded to 1 d.p.) and displays the output in a formatted string*max\_length\_of\_service = round(df\_1c['Length of service (in years)'].max(), 1)  
print(f'The maximum length of service rounded to 1 d.p. is {max\_length\_of\_service} years')  
  
*#This computes the average length of service (rounded to 1 d.p.) and displays the output dynamically in a formatted string*average\_length\_of\_service = round(df\_1c['Length of service (in years)'].mean(), 1)  
print(f'The average length of service rounded to 1 d.p. is {average\_length\_of\_service} years')

## Q1(d)

The python code for the interactive search function with documentation is appended below:

*#To import the necessary libraries that are required for the code to run  
import* pandas *as* pd  
  
*#to read the CSV as a Pandas DataFrame*raw\_df = pd.read\_csv(r"C:\Users\timot\OneDrive\Documents\SUSS\ANL252 - Python\TMA\Raw dataset\TMA\_Data.csv")  
  
*#a copy of raw\_df is created so that it can be cleaned up and transformed for the purposes of 1(d)*df\_1d = raw\_df.copy()  
df\_1d.set\_index("ID", inplace=*True*)  
*#the staff name is transformed to lower case in a new column called 'Staff\_lower'. this is to allow for a case-agnostic validation of user input in the next step*df\_1d["Staff\_lower"] = df\_1d["Staff"].str.lower()  
*#this serves to clean the source data by removing all leading and trailing whitespaces (if any) from the "Staff\_lower" column*df\_1d["Staff\_lower"].str.rstrip()  
df\_1d["Staff\_lower"].str.lstrip()  
  
*#a function is defined to check if the name entered by the user is in the DataFrame  
def* name\_check():  
 name = input("Please enter the username that you will like to check:\n")  
 *#this removes all trailing and leading whitespaces from the user input, thus allowing for the programme to function properly if the user improperly enters the name in this regard* name = name.rstrip()  
 name = name.lstrip()  
 *#the input is converted to lower case, thus allowing for the check to be case-agnostic  
 if* name.lower() *in* df\_1d["Staff\_lower"].values:  
 print(f"The name entered ({name.title()}) can be found in the organisation's records")  
 *else*:  
 print(f"The name entered ({name.title()}) cannot be found in the organisation's records")  
  
name\_check()  
  
*#this step serves to allow users to choose whether to continue searching for more names or quit the program  
#note that the input is case-agnostic as it is converted to lower case  
#a while loop is used to execute this block of code  
 #if user input is 'Y', the name\_check() function is executed again  
 #if user input is 'N', the loop is broken out of and the program ends  
 #if the user input is some other value besides 'Y' or 'N', the user is informed that their input is invalid and the loop continues through another iteration  
while True*:  
 user\_option = input("Do you want to continue searching for more names (Y/N)?\n")  
 *if* user\_option.lower() =="y":  
 name\_check()  
 *continue  
 elif* user\_option.lower() == "n":  
 print('You have successfully exit the program.')  
 *break  
 else*:  
 print(f'{user\_option} is not a valid input. Please either enter "Y" or "N".\n')  
 *continue*

## References

Dixon-Fyle, S., Dolan, K., Hunt, V., & Prince, S. (2020, May 19). *Diversity wins: How inclusion matters*. McKinsey & Company. Retrieved 11 August 2022, from https://www.mckinsey.com/featured-insights/diversity-and-inclusion/diversity-wins-how-inclusion-matters

Downey, A. (2016). *Think Python: How to Think Like a Computer Scientist* (2nd ed.). O’Reilly Media.

Equal Rights Advocates. (n.d.). *Gender Discrimination at Work*. Retrieved 11 August 2022, from https://www.equalrights.org/issue/economic-workplace-equality/discrimination-at-work/

Matplotlib. (n.d.). *Grouped bar chart with labels*. Retrieved 12 August 2022, from https://matplotlib.org/stable/gallery/lines\_bars\_and\_markers/barchart.html

Matthes, E. (2015). *Python Crash Course: A Hands-On, Project-Based Introduction to Programming* (1st ed.). No Starch Press.

McKinney, W. (2017). *Python for Data Analysis*. Van Duuren Media.

Moffitt, C. (2019, October 28). *Cleaning Up Currency Data with Pandas*. Practical Business Python. Retrieved 11 August 2022, from https://pbpython.com/currency-cleanup.html

pandas. (n.d.). *pandas.Series.map*. Retrieved 11 August 2022, from https://pandas.pydata.org/docs/reference/api/pandas.Series.map.html

*Python String Formatting*. (n.d.). ThePythonGuru. Retrieved 11 August 2022, from https://thepythonguru.com/python-string-formatting/

R. Severance, C., PhD. (2016). *Python for Everybody*. https://do1.dr-chuck.com/pythonlearn/EN\_us/pythonlearn.pdf

Wu, K. Y. (2022). *ANL252 Python for data analytics (study guide)*. Singapore University of

Social Sciences.