

**ANL252 (Online)**

**Python for Data Analytics**

# Tutor-Marked Assignment

**July 2022 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Tricia Ang Li Ying** | **J2081605** |

**Tutorial Group: ­­­­­­­­­­­­ TG 09**

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

**Submission Date: 14/08/2022**

Table of Contents

[Question 1(A) 3](#_Toc111331651)

[Figure 1: [Chart A] Average Employee Satisfaction by Performance Score 3](#_Toc111331652)

[Figure 2: [Chart B] Age distribution by Unit 4](#_Toc111331653)

[Question 1(B) 5](#_Toc111331654)

[Chart A data table output 6](#_Toc111331655)

[Figure 3: Chart A data table output 6](#_Toc111331656)

[Chart A output 8](#_Toc111331657)

[Figure 4: Chart A Python output 8](#_Toc111331658)

[Chart B data table output 11](#_Toc111331659)

[Figure 5: Chart B data table output 11](#_Toc111331660)

[Chart B output 12](#_Toc111331661)

[Figure 6: Chart B Python output 12](#_Toc111331662)

[Question 1(C) 13](#_Toc111331663)

[Question 1(D) 14](#_Toc111331664)

[References 15](#_Toc111331665)

[Appendices 16](#_Toc111331666)

[Appendix A 16](#_Toc111331667)

[Appendix B 17](#_Toc111331668)

# Question 1(A)

*See Appendix A for Chart A and B’s summarised data tables*

Chart

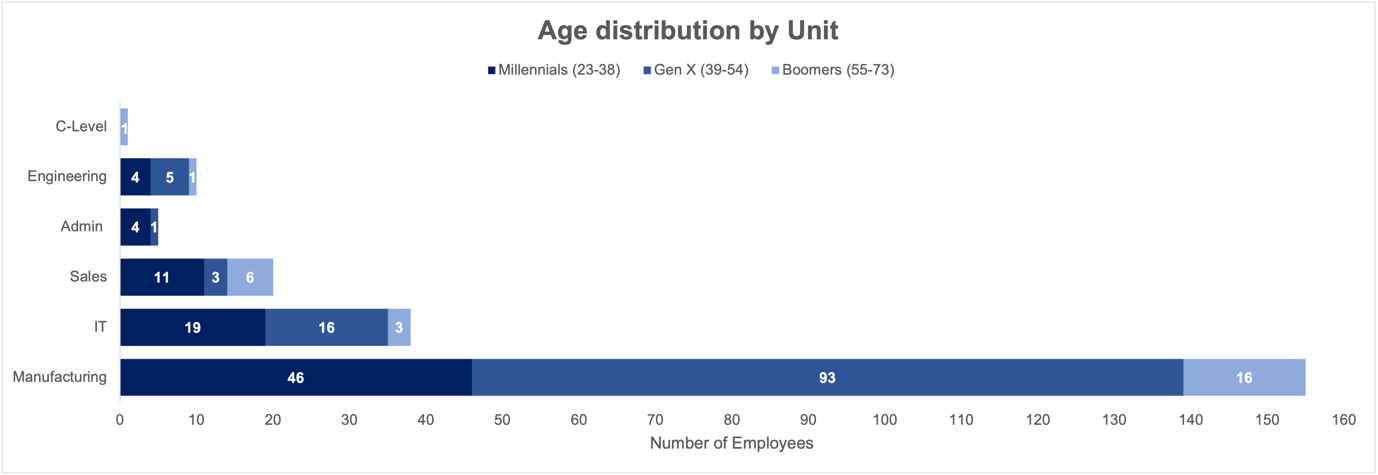
Description automatically generated

### Figure 1: [Chart A] Average Employee Satisfaction by Performance Score

Chart A showcases average employee satisfaction segmented by the different performance score categories. There is a clear positive correlation between the average satisfaction and performance score – the more satisfied employees are, the better their performance. Employees with low satisfaction tend to underperform, with those in the Performance Improvement Program (PIP) scoring an average of 1.5.

With this relationship made known, the Human Resource department can further analyse the underlying reasons behind high employee satisfaction, then implement activities accordingly, ultimately increasing employee work performance.

From the above chart, majority of employees (80.4%) fall into the “Meet” category, followed by “Exceed”, “To Improve”, and “PIP”. This showcases that the employees are generally meeting the overall performance expectations of the company, with 12% (30/250) exceeding expectations. A small minority (7.6%) need improvement or are put on performance improvement plans (PIPs).



### Figure 2: [Chart B] Age distribution by Unit

Chart B displays the age distribution of employees by their respective business units. The company comprises of mostly Generation X (age 39-54) with 46.8% of employees, followed by Millennials (33.6%), and Boomers (19.6%). Manufacturing is the largest business unit, with nearly 160 employees, followed by IT, and Sales. C-Level is the smallest unit, with only 1 employee.

Manufacturing and Engineering are the only business units where it is predominately made up of Gen X, whereas the rest of the units are predominately make up of Millennials, with the exception of C-Level.

Breaking down the different generations segmented by business units can enable superiors to better understand and manage their employees as each generation has different needs, attitudes, and approaches towards career and work (Marston, 2010).

Although each generation is receptive to different leadership styles and learning methods (Waldman, 2021), they can also learn from one another regardless of seniority or age. There is high multi-generational diversity among employees, however, Generation Z (age >=23) employees (0) could be incorporated into the organization to enhance multi-generational learning.

# Question 1(B)

*See Appendix B for side-by-side comparison of excel and python versions of Chart A and B*

**Importing libraries**

# Pandas for managing datasets

import pandas as pd

# Matplotlib for additional customization

from matplotlib import pyplot as plt

%matplotlib inline

# Seaborn for plotting and styling

import seaborn as sns

**Importing dataset from excel file**

# Import data from excel

TMAData = pd.read\_excel(r'/Users/twly/Desktop/ANL252\_Excel\_triciaang001\_TriciaAngLiYing.xlsx',sheet\_name='TMA\_Data') # Replace r'...' with new file location

#Setting dataframe

df\_TMA = pd.DataFrame(TMAData)

#Print TMA Dataset

display(df\_TMA)

**Calculating "Employees by Performance Score" for Chart A data table**

#Calculating number of employees in "PIP"

PIP = (sum(df\_TMA.PerformanceScore =="PIP"))

print(PIP)

#Calculating number of employees in "To Improve"

To\_Improve = (sum(df\_TMA.PerformanceScore =="To Improve"))

print(To\_Improve)

#Calculating number of employees in "Meet"

Meet = (sum(df\_TMA.PerformanceScore =="Meet"))

print(Meet)

#Calculating number of employees in "Exceed"

Exceed = (sum(df\_TMA.PerformanceScore =="Exceed"))

print(Exceed)

**Calculating "Average Satisfaction" for Chart A data table**

#Calculating average satisfaction in "PIP"

AS\_PIP=(df\_TMA.loc[df\_TMA['PerformanceScore'] == 'PIP', 'Satisfaction'].sum() / PIP)

print(AS\_PIP)

#Calculating average satisfaction in "To Improve"

AS\_To\_Improve=(df\_TMA.loc[df\_TMA['PerformanceScore'] == 'To Improve', 'Satisfaction'].sum() / To\_Improve)

print(AS\_To\_Improve)

#Calculating average satisfaction in "Meet"

AS\_Meet=(df\_TMA.loc[df\_TMA['PerformanceScore'] == 'Meet', 'Satisfaction'].sum() / Meet)

print(AS\_Meet)

#Calculating average satisfaction in "Exceed"

AS\_Exceed=(df\_TMA.loc[df\_TMA['PerformanceScore'] == 'Exceed', 'Satisfaction'].sum() / Exceed)

print(AS\_Exceed)

**Creating Chart A data table**

# Creating Chart A data table

ChartA\_data = {'Performance Score': ['PIP', 'To Improve', 'Meet', 'Exceed'],

'Employees by Performance Score': [PIP, To\_Improve, Meet, Exceed],

'Average Satisfaction': [AS\_PIP, AS\_To\_Improve, AS\_Meet, AS\_Exceed]}

# Creating Chart A dataframe

ChartA = pd.DataFrame(ChartA\_data)

# Rounding off Average Satisfaction to 2 decimal places

ChartA['Average Satisfaction'] = ChartA['Average Satisfaction'].round(decimals = 2)

# Print Chart A data

display(ChartA)

## Chart A data table output

Table

Description automatically generated

### Figure 3: Chart A data table output

**Creating Chart A, dual-axis chart**

# Set default font family to 'Arial'

plt.rcParams.update({'font.family':'Arial'})

# Set theme for Chart A

sns.set\_theme(style='dark',rc={'axes.facecolor':'White', 'figure.facecolor':'White'})

fig, ax1 = plt.subplots(figsize=(12,6))

# Plot bar chart as right y axis

BC=sns.barplot(data = ChartA, x = 'Performance Score', y = 'Employees by Performance Score', ax=ax2, color='#002060', label='Employees by Performance Score')

# Show values on bar chart

for i in ax2.containers:

ax2.bar\_label(i,label\_type='edge', fontsize=9, color='#404141', weight='bold',)

# Plot dataset A as dual-axis chart

# Plot line chart as left y axis

LC=sns.lineplot(data = ChartA, y='Average Satisfaction',x='Performance Score', marker='o', ax=ax1, color='#97b0df',label='Average Satisfaction')

ax2 = ax1.twinx()

# Plot bar chart as right y axis

BC=sns.barplot(data = ChartA, x = 'Performance Score', y = 'Employees by Performance Score', ax=ax2, color='#002060', label='Employees by Performance Score')

# Show values on bar chart

for i in ax2.containers:

ax2.bar\_label(i,label\_type='edge', fontsize=9, color='#404141', weight='bold',)

# Show and format alignment of values on line chart

for x, y in zip(ChartA['Performance Score'], ChartA['Average Satisfaction']):

ax1.annotate(y, xy=(x, y), textcoords='data',horizontalalignment='center', verticalalignment='bottom', color='#404141', fontsize=9, weight='bold',)

# Set, format, and move title

ax1.set\_title("Average Satisfaction by Performance Score", fontsize=16 ,weight='bold', y=1.12)

# Add legend and move to top of plot area

ax1.legend(loc='upper center', bbox\_to\_anchor=(0.67, 1.10),frameon=False)

ax2.legend(loc='upper center', bbox\_to\_anchor=(0.38, 1.10),frameon=False)

# Format font size of all axes

plt.tick\_params(axis='both', which='major', labelsize=9)

#Set axis limit for left y axis

ax1.set\_ylim(0, 4.5)

#Set axis limit for right y axis

ax2.set\_ylim(0, 250)

# Turn off grid and ticks on left y xaxis

ax1.grid(False)

ax1.tick\_params(left = False)

# Turn off grid and ticks on the right y axis

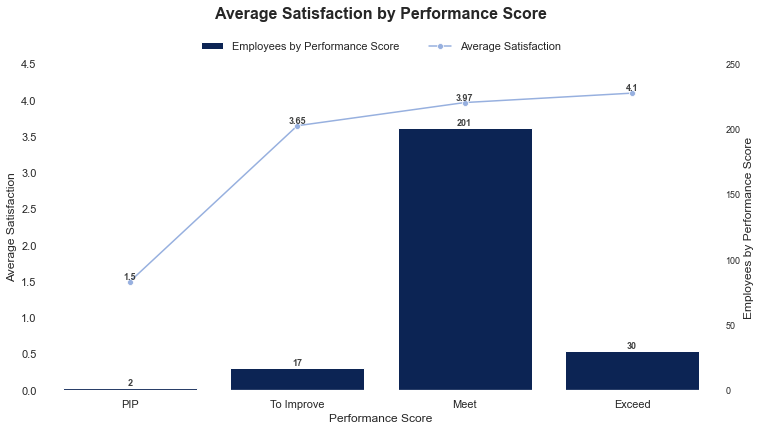
ax2.grid(False)

ax2.tick\_params(right = False)

# Show chart

plt.show(LC,BC)

## Chart A output



### Figure 4: Chart A Python output

**Creating ’Age’ colum from BirthYear for Chart B data table**

#Import date from datetime

from datetime import date

# function

def calc\_age(bd: pd.Series) -> pd.Series:

today = pd.to\_datetime(date.today()) # convert today to a pandas datetime

return (today - bd) / pd.Timedelta(days=365.25) # divide by days to get years

# call function and assign the values to a new column in the dataframe

df\_TMA['Age'] = calc\_age(df\_TMA.BirthYear)

# Print ID, Staff, and Age columns

display(df\_TMA[['ID','Staff','Age']])

**[Manufacturing] Segregating unit into different age generations**

# Count and print number of Millennials (Age 23-38) in unit "Manufacturing"

M\_M = df\_TMA[(df\_TMA['Unit']=='Manufacturing') & (df\_TMA['Age']>=23) & (df\_TMA['Age']<=38)]['Age'].count()

print(M\_M)

# Count and print number of Gen X (Age 39-54) in unit "Manufacturing"

M\_G = df\_TMA[(df\_TMA['Unit']=='Manufacturing') & (df\_TMA['Age']>=39) & (df\_TMA['Age']<=54)]['Age'].count()

print(M\_G)

# Count and print number of Gen X (Age 39-54) in unit "Manufacturing"

M\_B = df\_TMA[(df\_TMA['Unit']=='Manufacturing') & (df\_TMA['Age']>=55) & (df\_TMA['Age']<=73)]['Age'].count()

print(M\_B)

**[IT] Segregating unit into different age generations**

# Count and print number of Millennials (Age 23-38) in unit "IT"

IT\_M = df\_TMA[(df\_TMA['Unit']=='IT') & (df\_TMA['Age']>=23) & (df\_TMA['Age']<=38)]['Age'].count()

print(IT\_M)

# Count and print number of Gen X (Age 39-54) in unit "IT"

IT\_G = df\_TMA[(df\_TMA['Unit']=='IT') & (df\_TMA['Age']>=39) & (df\_TMA['Age']<=54)]['Age'].count()

print(IT\_G)

# Count and print number of Boomers(Age 39-54) in unit "IT"

IT\_B = df\_TMA[(df\_TMA['Unit']=='Manufacturing') & (df\_TMA['Age']>=55) & (df\_TMA['Age']<=73)]['Age'].count()

print(IT\_B)

**[Sales] Segregating unit into different age generations**

# Count and print number of Millennials (Age 23-38) in unit "Sales"

Sales\_M = df\_TMA[(df\_TMA['Unit']=='Sales') & (df\_TMA['Age']>=23) & (df\_TMA['Age']<=38)]['Age'].count()

print(Sales\_M)

# Count and print number of Gen X (Age 39-54) in unit "Sales"

Sales\_G = df\_TMA[(df\_TMA['Unit']=='Sales') & (df\_TMA['Age']>=39) & (df\_TMA['Age']<=54)]['Age'].count()

print(Sales\_G)

# Count and print number of Gen X (Age 39-54) in unit "Sales"

Sales\_B = df\_TMA[(df\_TMA['Unit']=='Sales') & (df\_TMA['Age']>=55) & (df\_TMA['Age']<=73)]['Age'].count()

print(Sales\_B)

**[Admin] Segregating unit into different age generations**

# Count and print number of Millennials (Age 23-38) in unit "Admin"

Admin\_M = df\_TMA[(df\_TMA['Unit']=='Admin ') & (df\_TMA['Age']>=23) & (df\_TMA['Age']<=38)]['Age'].count()

print(Admin\_M)

# Count and print number of Gen X (Age 39-54) in unit "Admin"

Admin\_G = df\_TMA[(df\_TMA['Unit']=='Admin ') & (df\_TMA['Age']>=39) & (df\_TMA['Age']<=54)]['Age'].count()

print(Admin\_G)

# Count and print number of Gen X (Age 39-54) in unit "Admin"

Admin\_B = df\_TMA[(df\_TMA['Unit']=='Admin ') & (df\_TMA['Age']>=55) & (df\_TMA['Age']<=73)]['Age'].count()

print(Admin\_B)

**[Engineering] Segregating unit into different age generations**

# Count and print number of Millennials (Age 23-38) in unit "Engineering"

E\_M = df\_TMA[(df\_TMA['Unit']=='Engineering') & (df\_TMA['Age']>=23) & (df\_TMA['Age']<=38)]['Age'].count()

print(E\_M)

# Count and print number of Gen X (Age 39-54) in unit "Engineering"

E\_G = df\_TMA[(df\_TMA['Unit']=='Engineering') & (df\_TMA['Age']>=39) & (df\_TMA['Age']<=54)]['Age'].count()

print(E\_G)

# Count and print number of Gen X (Age 39-54) in unit "Engineering"

E\_B = df\_TMA[(df\_TMA['Unit']=='Engineering') & (df\_TMA['Age']>=55) & (df\_TMA['Age']<=73)]['Age'].count()

print(E\_B)

**[C-Level] Segregating unit into different age generations**

# Count and print number of Millennials (Age 23-38) in unit "C-Level"

C\_M = df\_TMA[(df\_TMA['Unit']=='C-Level') & (df\_TMA['Age']>=23) & (df\_TMA['Age']<=38)]['Age'].count()

print(C\_M)

# Count and print number of Gen X (Age 39-54) in unit "C-Level"

C\_G = df\_TMA[(df\_TMA['Unit']=='C-Level') & (df\_TMA['Age']>=39) & (df\_TMA['Age']<=54)]['Age'].count()

print(C\_G)

# Count and print number of Gen X (Age 39-54) in unit "C-Level"

C\_B = df\_TMA[(df\_TMA['Unit']=='C-Level') & (df\_TMA['Age']>=55) & (df\_TMA['Age']<=73)]['Age'].count()

print(C\_B)

**Creating Chart B data table**

# Creating Chart B data table

ChartB\_data = {'Unit': ['Manufacturing', 'IT', 'Sales', 'Admin', 'Engineering', 'C-Level'],

'Millennials (23-38)': [M\_M, IT\_M, Sales\_M, Admin\_M, E\_M, C\_M],

'Gen X (39-54)': [M\_G, IT\_G, Sales\_G, Admin\_G, E\_G, C\_G],

'Boomers (55-73)': [M\_B, IT\_B, Sales\_B, Admin\_B, E\_B, C\_B]}

# Creating Chart B dataframe

ChartB\_t = pd.DataFrame(ChartB\_data)

# Print Chart B table

display(ChartB\_t)

## Chart B data table output

Table

Description automatically generated

### Figure 5: Chart B data table output

**Creating Chart B, horizontal stacked bar chart**

# Set default font family to 'Arial'

plt.rcParams.update({'font.family':'Arial'})

# Import Library

import matplotlib.pyplot as plt

# Increase size of plot in jupyter notebook

plt.rcParams["figure.figsize"] = (20,5)

# Set seaborn theme for Chart B

sns.set\_theme(style='dark',rc={'axes.facecolor':'White', # Set axes color to white

'figure.facecolor':'White'}) #Set figure color to white

# Create and format (Color and width) stacked bar chart

ChartB=ChartB\_t.set\_index('Unit').plot(kind='barh', # Sets horizontal

stacked=True, # Sets bar chart to stacked

color=['#002060', '#2f5597','#8fabdd'], # Sets custom color for bars

width=0.7, # Customize width of bar chart to 0.7

edgecolor = "none") # Removes bar borders

# Set and format title

ChartB.set\_title("Age distribution by Unit", # Set chart title

fontsize=16 , # Set font size to 16

weight='bold', # Set title to be bolded

y=1.12) # Set position of title to exceed plot area at y=1.12

# Format legend

ChartB.legend(loc='upper center', # Set location of chart to upper center

bbox\_to\_anchor=(0.5, 1.1), # Set location to exceed plot area at 0.5, 1.1

frameon=False, # Removes frame from legend

ncol=3, # Set legend to one row

fontsize=9) # Set font size to 9

# Plot x axis label

plt.xlabel("Number of Employees")

# Annotate and format data labels

if i != 0:

for i in ChartB.containers:

ChartB.bar\_label(i, label\_type='center', fontsize=9, color='White', weight='bold')

# Show chart

plt.show(ChartB)

## Chart B output

Graphical user interface, application, Teams

Description automatically generated

### Figure 6: Chart B Python output

# Question 1(C)

**Import libraries**

# Import statistics

import statistics

#Import datetime

from datetime import datetime

**Calculate "Total Service Years" using "JoinDate" and "LeftDate"**

emp=[]

# Loop to compare till all data is checked

for i in range(0,len(df\_TMA['JoinDate'])):

if pd.isnull(df\_TMA['LeftDate'].iloc[i])==True: # If the LeftDate value of data row i is empty then this condition is true

x=pd.Timestamp('2022-05-1T12:00:00') # Set the date as 1/5/2022 for those that have not left yet

emp.append(round((x-df\_TMA['JoinDate'].iloc[i]).days/365,1)) # Convert JoinDate days in service into years

else: # If staff have left the organisation, run the else condition

emp.append(round((df\_TMA['LeftDate'].iloc[i]-df\_TMA['JoinDate'].iloc[i]).days/365,1)) # Convert LeftDate days in service into years

df\_TMA['Total Service Years']=emp # Create new column to save the total service years

# Print columns 'ID','Staff', and 'Total Service Years'

print('Total Service Years: ')

df\_TMA[['ID','Staff','Total Service Years']]

**Calculate and print minimum, maximum, and average service**

print("Minimum Service: ","{:.1f}".format(min(emp)),'years')

print("Maximum Service: ","{:.1f}".format(max(emp)),'years')

print("Average Service: ","{:.1f}".format(statistics.mean(emp)),'years')

# Question 1(D)

**Interactive user input to verify employment status**

found=False # Flag to check if record is found

while True: # Run till loop is broken

choice=input("Enter full name of staff to verify employment or 'exit' to quit: ")

for i in df\_TMA['Staff']: # Checking all values in Staff column of the dataframe

if i.lower()==choice.lower(): # Check if name is found, lower() allows checking with capital insensitivity

print(i,"'s record was found in the staff of the organization")

found=True # Found variable is set True

# If 'exit' entered is by user then exit the loop

if choice.lower()=='exit':

print('Quitting...')

break

# If input by user is not found, print record not found

if found==False:

print(choice,"'s record was NOT found in the staff of the organization")

# References

Marston, C. (2010, August 27). Employee training and development across the generations. Training Industry. <https://trainingindustry.com/articles/strategy-alignment-and-planning/employee-training-and-development-across-the-generations/>

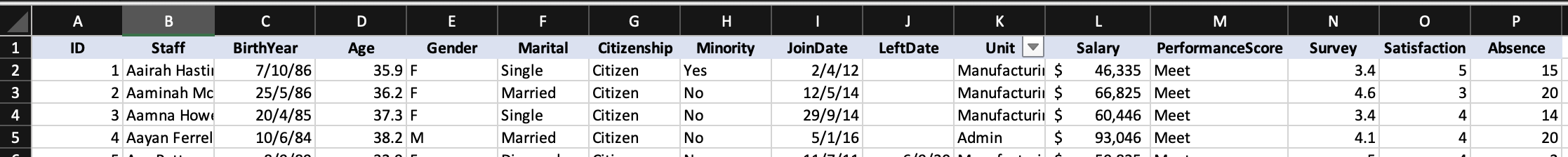
Waldman, E. (2021, August 31). How to manage a multi-generational team. Harvard Business Review. <https://hbr.org/2021/08/how-to-manage-a-multi-generational-team>

# Appendices

## Appendix A

Elaboration on summarised data tables for Chart A and B

*Appendix A1: TMA\_Data sheet’s column labels*

****

*Appendix A2: Summarised data table for Chart A*

Table

Description automatically generated

Formula “=COUNTIF(TMA\_Data!$M:$M,$A3)” used COUNTIF to conditionally count the number of employees in "which fell into the respective Performance Score categories using TMA\_Data “PerformanceScore” Category

Formula “=SUMIF(TMA\_Data!$M:$M, !$A3, TMA\_Data!$O:$O)/B3” was used to conditionally sum the satisfaction scores on employees based on their respective Performance Score categories, then divided by the count of employees in that category to derive the average

*Appendix A2: Summarised data table for Chart B*

Table

Description automatically generated

Row “TMA\_Data!$D:$D” contains current age of the employee, derived subtracting their date of birth from the current date (today) and rounding off to the nearest whole number using the formula “=ROUND(YEARFRAC(C2, TODAY(), 1),0)”

Formula “=COUNTIFS(TMA\_Data!$K:$K, Age!$A2, TMA\_Data!$D:$D, ">=23", TMA\_Data!$D:$D, "<=38")” counts the cell if it meets the specified criteria of Industry and age range

## Appendix B

Side-by-side comparison of excel and python output of Chart A and B

*Appendix B1: [Excel output] Chart A*

Chart

Description automatically generated

*Appendix B2: [Python output] Chart B*

Chart

Description automatically generated

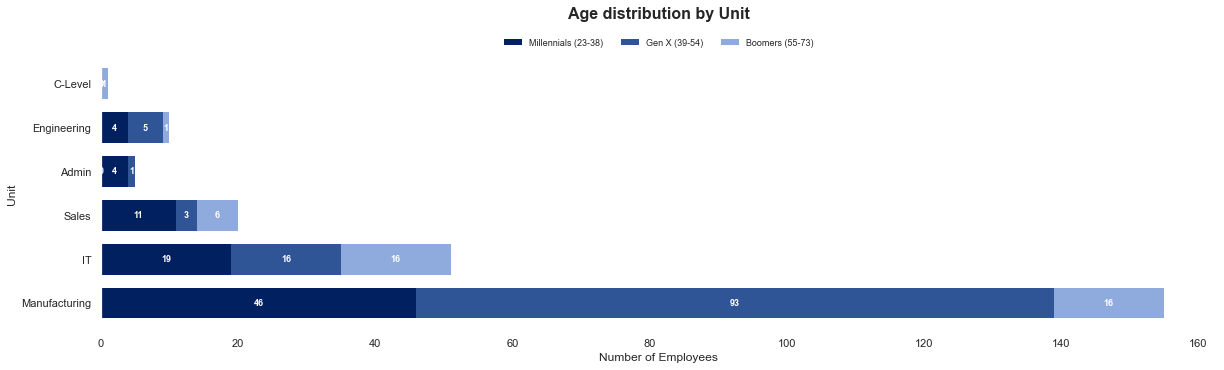
Although there are several visual discrepancies in font size and colour, the same configurations (font size 9 and hex colour code) used in Excel has been applied in Python

*Appendix B3: [Excel output] Chart B*

Graphical user interface, text, application

Description automatically generated

*Appendix B1: [Python output] Chart B*



Although there are several visual discrepancies in font size and colour, the same configurations (font size 9 and hex colour code) used in Excel has been applied in Python