A picture containing logo

Description automatically generated

## ANL252

|  |  |
| --- | --- |
| **STUDENT NAME:** | KYNAN TAN |
| **P.I. NO.:** | Z2181984 |
| **TITLE:** | TMA01 |
| **COURSE CODE:** | ANL252 |
| **T:** | 09 |
| **SUBMISSION DATE:** | 14 August 2022 |

**Plagiarism and Collusion**

**Plagiarism** is the act of using or passing as one’s own, the ideas or writings of another without acknowledging or crediting the source from which the ideas are taken from.

**Collusion** is the act of submitting any academic work (including assignment, project or report) that was completed by another person and pass these work off as one’s own.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Declaration** | | | | |
|  | | | | |
| I declare that this assignment is my own work, unless otherwise acknowledge or credited by appropriate referencing. I have read and abide by the SUSS Honour Code and I am aware of the penalties associated with plagiarism and collusion listed in the Student Handbook. | | | | |
|  | | | | |
| Initial: | K |  | Date: | 14 Aug 2022 |
|  | | | | |

(a)

**Figure 1**

*Overall Performance of Company*

*Pie Chart.* This shows the performance of all the employees of the company.

**Table 1**

*Table accounting for the number of employees and their respective performance*

|  |  |
| --- | --- |
| Performance | Employee Count |
| Meet | 201 |
| Exceed | 30 |
| To Improve | 17 |
| PIP | 2 |
| Total | 250 |

|  |
| --- |
| From the pie chart, we can observe that most employees (92.4%) of the company have met or exceeded their performance. In number, 17 employees (6.8%) of the company need to further improve their performance and 2 employees (0.8%) require a performance improvement plan (PIP). |
|  |
|  |

**Figure 2**

*Will citizenship affect the average salary per business unit?*

*Bar Chart.* Comparing the average salary of each business unit to the average salary of each citizenship status within the same unit.

**Table 2**

*Table showing the average salary value of each business unit and the average salary value of each citizenship status within the same unit.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Unit | Average Salary | Citizen | Non-Resident | PR |
| Admin | $77,315.29 | $ 79,701.17 | $ - | $ 63,000.00 |
| Engineering | $95,867.00 | $ 97,033.00 | $ 100,416.00 | $ 89,511.50 |
| IT | $95,550.76 | $ 93,964.23 | $ 94,490.40 | $ 104,367.00 |
| Manufacturing | $59,412.91 | $ 59,883.29 | $ 58,696.10 | $ 57,178.36 |
| Sales | $71,426.59 | $ 71,502.32 | $ - | $ 70,947.00 |

There are no non-resident employees in the Admin and Sales units, hence explaining the zero-value salary in the table.

Analysing the table, we can deduce that the employees of the Manufacturing and Sales units generally earn an average salary similar to the average salary of their respective units. Additionally, we can observe that the average salary of the PR employees in the Admin and Engineering units earn approximately $14,000 and $6,000 lesser than the average salary of their respective units. However, the PR employees in the IT unit earn a higher average salary, approximately $9,000, more than the average salary of the IT unit. Lastly, non-resident employees in the Engineering unit earn a higher average salary, approximately $5,000, more than the average salary of the Engineering unit.

Therefore, we can conclude being a PR or non-resident employee only affects the average earning salary in the Admin, Engineering and IT units whereas citizens generally earn an average salary similar to the average salary of these five business units.

(b)

Re-creating pie-chart

# import relevant libraries.

import pandas as pd

import numpy as np

from datetime import date

from matplotlib import pyplot as plt

# read data into pandas dataframe

df = pd.read\_csv("TMA\_Data.csv")

# creating summary table by counting the number of employees that met their respective

performance

performance\_table = df.groupby(['PerformanceScore'])['PerformanceScore'].count()

# creating a dataframe for performance\_table

performance\_table = pd.DataFrame(performance\_table)

# renaming PerformanceScore to Employee Count to replicate Excel's table output

performance\_table.rename(columns = {'PerformanceScore' : 'Employee Count'}, inplace=True)

# resetting index of performance\_table

performance\_table = performance\_table.reset\_index()

# corresponding summarized table for pie chart

display(performance\_table)

# creating pie chart using the summary table

performance\_table.groupby(['PerformanceScore']).sum().plot(kind='pie', title='Overall Performance of Company', y='Employee Count', autopct='%1.1f%%')

plt.ylabel('') # removing ylabel from pie chart output

plt.show()

Re-creating bar chart

# creating summary table by grouping the average salary of employees by unit

avg\_unit\_salary = df.groupby(['Unit'])['Salary'].mean()

# creating a dataframe for avg\_unit\_salary

avg\_unit\_salary = pd.DataFrame(avg\_unit\_salary)

# removing the only C-level unit from the data because there is only one employee in that unit

avg\_unit\_salary = avg\_unit\_salary.drop('C-Level')

# resetting index of avg\_unit\_salary

avg\_unit\_salary = avg\_unit\_salary.reset\_index()

# creating lists with average salary values of each citizenship status

unit\_list = ['Admin ', 'Engineering', 'IT', 'Manufacturing', 'Sales']

citizen\_list = [79701.17, 97033.00, 93964.23, 59883.29, 71502.32]

non\_resident\_list = [0, 100416.00, 94490.40, 58696.10, 0]

PR\_list = [63000.00, 89511.50, 104367.00, 57178.36, 70947.00]

# creating a dataframe for avg\_citizen\_salary

avg\_citizen\_salary\_df = pd.DataFrame({'Unit':unit\_list, 'Citizen':citizen\_list, 'Non-Resident':non\_resident\_list, 'PR':PR\_list})

# merging both dataframes, avg\_unit\_salary and avg\_citizen\_salary, into one dataframe before plotting bar graph

after\_merging = avg\_unit\_salary.merge(avg\_citizen\_salary\_df)

after\_merging.rename(columns = {'Salary':'Average Salary'}, inplace = True)

# corresponding summarized table for bar chart

display(after\_merging)

# plotting bar chart

after\_merging.plot(x='Unit', kind='bar', stacked=False, title='Will citizenship affect average salary per business unit?')

# rotate x-axis labels vertically

plt.xticks(rotation = 0)

plt.show()

`(c)

# converting JoinDate column to datetime object to perform arithmetic operations

df['JoinDate'] = pd.to\_datetime(df['JoinDate'])

# converting LeftDate column to datetime objects to perform arithmetic operations

df['LeftDate'] = pd.to\_datetime(df['LeftDate'])

# replacing the missing values of LeftDate for staff who has not left the organisation using the date provided, 1st May 2022.

df['LeftDate'].fillna(value = "5/1/2022", inplace = True)

# finding the length of service of all the staff in days

df['Difference in years'] = df['LeftDate'] - df['JoinDate']

# finding the length of service of all the staff in years

df['Difference in years'] = ((df['Difference in years']).dt.days / 365)

service\_in\_years = df['Difference in years']

service\_in\_years = pd.DataFrame(service\_in\_years)

# finding the minimum length of service in years

min\_service = float(df['Difference in years'].min())

print("The minimum length of service is " + str("%.1f" % min\_service) + " years.")

# finding the maximum length of service in years

max\_service = float(df['Difference in years'].max())

print("The maximum length of service is " + str("%.1f" % max\_service) + " years.")

# finding the average length of service in years

mean\_service = float(df['Difference in years'].mean())

print("The average length of service is " + str("%.1f" % mean\_service) + " years.")

(d)

print("This program checks if a particular person was/is a staff of the organization.")

conditional\_date = pd.Timestamp(year=2022, month=5, day=1, hour=0) # setting conditional date to 1st May 2022

status = True # status is the break condition for the while-loop

while status == True:

try:

name\_entry = input("Please enter the person's full name: ")

Name\_Entry = name\_entry.title() # capitalize the first character in every word

check\_person = df.loc[df.Staff == Name\_Entry] # checking if the name entered exists in the organization's records

join\_date = check\_person.values[0,7]

left\_date = check\_person.values[0,8]

if left\_date == conditional\_date:

joindate = join\_date.strftime('%d %B %Y.') # changing the date format to (day of the month, month's full name, full year) to be read easily

print(f"{Name\_Entry} is currently a staff of the organization, and has been a staff since {joindate}")

else:

leftdate = left\_date.strftime("%d %B %Y.") # changing the date format to (day of the month, month's full name, full year) to be read easily

print(f"{Name\_Entry} was a staff of the organization, and has left the organization since {leftdate}")

while True:

go\_again = input("Would you like to enter another person's name (yes/no): ") # allowing the user to choose to enter another person's name

if go\_again == 'yes':

break # break out of this while-loop to start from name\_entry

elif go\_again == 'no':

status = False # setting status to False would indicate that the bigger while-loop would not execute

print('Stopping program.')

break # break out of this while-loop to end the program

else:

print('Invalid input, please enter yes or no.')

continue # continue this while-loop if the input for go\_again is neither 'yes' or 'no'

except IndexError:

print(f"The name {Name\_Entry} is not in the organization's records.")