

**ANL252**

**Python for Data Analytics**

**Tutor-Marked Assignment (TMA01)**

**July 2022 Presentation**

**Name:**

Pay Jiarre Lyn

**PI Number:**

Y1981896

**Submission Date: 12 Aug 2022**

**Question 1 (a)**

With the given staff data, two charts was designed, namely “Employee Satisfaction by Unit and Gender” and “ Percentage Count by Performance by Unit’.

Chart 1: Average Employee Satisfaction by Unit and Gender (Excel)

Measuring and boosting employee satisfaction is a crucial aspect of building a corporate culture that encourages staff retention. A content employee is one who embraces their work with enthusiasm and is proactive, productive, and committed to promoting the organization's goals.

Chart, bar chart

Description automatically generated

Table

Description automatically generated

The preceding graph depicts the average satisfaction of employees by Unit and Gender. The satisfaction rate is above average, indicating that the majority of employees are satisfied with their jobs. The male employees are more satisfied than the combined female workforce. Females in the Sales and Engineering division are the most satisfied with a satisfaction score of 4.4, while females in the Admin and C-Level division are the least content with a score of 3.0. The satisfaction rating of male administration personnel is much higher than that of female administration employees, demonstrating a considerable difference between the two groups.

Sales and Engineering have the highest levels of customer satisfaction. With open communication and mutual trust, the sales teams are more convivial and mutually supportive. This is a gratifying, value-added position that generates consistent recognition from management and clients. Those with a problem-solving mindset and the ability to make sound decisions have an expanding number of job options in the field of engineering, which also offers a high salary and creative work.

Experts in Human Resources may call women from the Admin and C-Level teams to a gathering or perform surveys to better comprehend their job attitudes. This increases motivation and appreciation for accomplishments at work.

Employers can make work more significant and meaningful to increase employee satisfaction. Particularly essential to younger employees is the sense that their job contributes to the company's goals and has a meaningful purpose. If workers possess the appropriate skills and talents, their job satisfaction will increase. Regular training and workshops are required for employees to excel and support their professional growth goals.

These tips can raise employee happiness, consequently reducing employee turnover and enhancing overall performance and output.

Chart 2: Percentage Count of Performance Score by Unit (Excel)



Table

Description automatically generated

Performance Score is a sort of Key Performance Indicator (KPI) that provides the organization with a means of measuring progress toward goals and conveying the company's strength to investors and other stakeholders (Klipfolio,n.d.). Tracking the performance score by Unit Level enables the organization to gain a deeper understanding of the successful and underperforming regions.

The majority of units, according to the graph, met or surpassed expectations. Admin and C-Level units have achieved the targeted objective of at least meeting all expectations. Engineering, information technology, production, and sales have met, enhanced, and beyond their respective targets. To protect their feelings and sense of responsibility for their work, employees who meet and surpass expectations should receive some form of acknowledgement. Despite the fact that persons who require growth require more supervision and encouragement, a sense of continuous improvement motivates them to exert themselves more and increases their job happiness (Klipfolio,n.d.).

**Question 1 (b)**

### Data Understanding

For data description, here are some functions used to find out more about the data. This include getting statistical data for DataFrame using “.describe()”, getting dimensions of array using “.shape” function and getting datatype of columns using “.dtypes()”. To preview the first five rows of the dataframe, the “.head()” function was used as well.

Code in Words and Output

**# Importing Libraries**

import pandas as pd

from datetime import date, time, datetime

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

**# Reading File to Pandas DataFrame**

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

df.head()

Graphical user interface, application

Description automatically generated

print("Data Description")

print(df.describe())

print("\nShape of data")

print(df.shape)

print("\nData Type")

print(df.dtypes)

Table

Description automatically generated

Chart 1: Average Employee Satisfaction by Unit and Gender (Python)

First, we will plot the bar graph with the unit as the x-axis and the average as the y-axis. The divisions will be organized according to gender, with the purple bar representing female employees and the red bar representing male employees. Next, in order to construct the summary table, we will groupby unit and gender and calculate the overall mean level of satisfaction.

**# Plotting bargraph with plotly, setting x-axis as unit, y-axis as satisfaction, categories by gender**

fig = px.histogram(df, x="Unit", y="Satisfaction",

color='Gender',

barmode='group',

histfunc='avg',

height=400,

title = "Average satisfaction by unit by gender",

text\_auto = ".2s")

**# Update the x-axis, y-axis and title label**

fig.update\_layout(

title="Average satisfaction by unit by gender",

xaxis\_title="Unit",

yaxis\_title="Average Satisfaction",

legend\_title="Gender")

**# Show graph**

fig.show()

Chart, bar chart

Description automatically generated

**# Summarised Table and rounding up values to 2dp**

df\_satisfaction = df.groupby(['Unit', 'Gender'])['Satisfaction'].mean().reset\_index(name = 'Average satisfaction Level')

df\_satisfaction.round(2)

Table

Description automatically generated

Chart 2: Percentage Count of Performance Score by Unit (Python)

First, we will construct the summarize table for the second chart, as we need to obtain the count and percentage for performance score in units. To obtain the counts, we will use ".size()" and "lambda" to determine the percentage. Then, we will plot the chart with the derived data.

Code in Words and Output

**# Summarised Table and rounding up values to 2dp**

df\_stack=df.groupby(['Unit','PerformanceScore']).size().reset\_index()

df\_stack['Percentage']=df.groupby(['Unit','PerformanceScore']).size().groupby(level=0).apply(lambda x:100 \* x/float(x.sum())).values

df\_stack.columns= ['Unit','PerformanceScore', 'Counts', 'Percentage']

df\_stack['Percentage'] = df\_stack['Percentage'].astype(float, errors = 'raise')

df\_stack.round(2)

Table

Description automatically generated

**# Plotting bargraph with plotly, setting x-axis as unit, y-axis as Percentage, categories by PerformanceScore**

fig = px.bar(df\_stack, x="Unit", y="Percentage",color='PerformanceScore',

title="Bar Plot",

template="plotly\_white",

height=600,

text = df\_stack['Percentage'].round(2))

**# Updating chart labels**

fig.update\_layout(barmode="relative", title="Percentage count of performance score by unit")

**# Formatting layout**

fig.update\_traces(textposition='inside')

fig.update\_layout(uniformtext\_minsize=8, uniformtext\_mode='hide')

fig.update\_layout(plot\_bgcolor='white')

fig.update\_yaxes(showline=False,showgrid=False)

fig.update\_xaxes(showline=False,showgrid=False)

**# Show graph**

fig.show()

Chart, bar chart

Description automatically generated

We observed from the preceding that both excel and Python provided identical outcomes. This offers credibility and assurance that the results are trustworthy. The primary distinction was the execution procedure. Excel's data analysis and chart charting will be more labor-intensive than Python's. Additionally, we must choose the chart type that best fits the analysis. In fact, the Excel may not function if the data is too huge. Python, on the other hand, can read and manipulate big data sets more quickly and easily. The application of complex statistical and data visualization skills will be streamlined.

However, the two methods can work well together. In general, Excel is better suited for less intricate and straightforward tasks. Python's flexibility, configurability, and transparency all contribute to improved error checking and faster processing of large data sets during advanced data analysis.

**Question 1(c)**

To determine the length of service for all staff members, we will convert the DateTime column to pandas datetime ['JoinDate', 'LeftDate']. Next, we identify any null values in LeftDate and assign the value '2022-05-01' to employees who have not yet left the organization. The year will then be determined by subtracting JoinDate from LeftDate, translating the result to days, and dividing by 365 days in a year.

Code in Words and Output

**# Convert JoinDate and LeftDate to datetime**

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

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

**# Check for null values, if null set value to "2022-05-01"**

df.loc[df['LeftDate'].isnull(), 'LeftDate'] = date(year=2022, month=5, day=1)

**# Get service length**

df['service\_length'] = (df['LeftDate'] - df['JoinDate']).dt.days/365

df.head()

Graphical user interface, table

Description automatically generated

Table

Description automatically generated

**# Get min, max and average of service\_length across all the staff**

df['service\_length'].describe().loc[['min', 'max', 'mean']].round(1).to\_frame()

Table

Description automatically generated

**Question 1(d)**

Getting Questions

For the development of the interactive user input inquiry, we will first get and generate queries pertinent to the dataset. The initial step is to compile a list of all usernames. Next, we will develop a function that produces the amount of questions based on the column names. Thirdly, we will iterate through the columns and attach the formatted column name together with the query ID to the list. Finally, we will concatenate the list values with n to produce the question in a presentable form.

Code in Words and Output

**# Get the names of all the employees and converting them to lower case**

name\_list = [x.lower() for x in df['Staff'].tolist()]

**# Create function to query list of questions based on user input**

def query\_questions(name):

# Create empty list to store questions

text\_message = ["Select your options"]

question\_no = 1

**# Iterate through all the columns to set name of column as part of the question**

for i in df.columns:

**# Add condition such that outputing the staff name of a given staff will not be part of the option**

if i == "service\_length":

text\_message.append(f"{str(question\_no)}: Get {i} (in years) of staff "

else:

text\_message.append(f"{str(question\_no)}: Get {i} of staff ")

question\_no +=

text\_message.append("17: View all information\n")

**# Convert items in list into a string**

return '\n'.join(text\_message)

Querying Employee

Next, we will construct a function that enables employee querying. The first stage is to obtain user input to select a name, with the "End" key used to exit the loop. Next, we will compare the supplied name to the name list values. As there are several users with similar names, a dynamic method may be constructed by determining the substring of values that matches the name list and obtaining a callback answer based on the user's selection. We will then filter the dataframe by staff name.

Code in Words and Output

def query\_employee():

**# Initialise an empty variable**

selected\_name = ""

question = []

list\_info = []

**# Perform iteration to ask for user input**

while True:

input\_name = input('Enter a name: \nClick on "Enter" to end ')

if input\_name == "":

return selected\_name, list\_info

break

**# Match part of user input to the name\_list and output names that matches**

elif any(input\_name.lower() in name for name in name\_list):

matching = [name for name in name\_list if input\_name.lower() in name]

for i in range(len(matching)):

question.append(f"{i + 1}. {matching[i].title()} ?")

print('\n'.join(question))

**# Get user to select users that have matching names**

while True:

input\_number = int(input('Key in the number reprsenting the employee that you want to see \nif none, type 0 to end '))

if input\_number in range(1, len(matching) +1) or input\_number == 0:

break

else:

print("Input is invalid, please key in a valid number ")

**# Store user input as a new variable**

if input\_number == 0:

return selected\_name, list\_info

break

selected\_name = matching[input\_number - 1].title()

print(f"You have selected {selected\_name}")

**# Get information that matches user input**

list\_info = df.loc[df['Staff'] == selected\_name.title(), :].values.tolist()

list\_info = list\_info[0]

return selected\_name, list\_info

else:

print("Invalid input/Employee not found")

Combining Questions and Employee Query

To execute the code as a whole, we will first execute the query\_employee() function to obtain the employee's name and details. At this phase, we must determine if the user has provided a proper response. The query\_question() function will then ask output questions and return user information based on the selected option. The dataframe will be filtered by the staff's name based on the provided input.

Code in Words and Output

def main():

while True:

**# Run above function to get the employee name and info**

employee\_name, employee\_info = query\_employee()

**# If the return value is blank, exit the for loop**

if employee\_name == "":

break

**# Get column name of dataframe and append to list**

search\_value = df.columns.tolist()

**# Get user input to select questions**

x = int(input(query\_questions(employee\_name)))

**# Check if user select values between 1 to 17**

if x in range(1, 17):

**# Print query results**

print(f"\n{search\_value[x - 1]}: {employee\_info[x - 1]}")

**# If user select print all info, display info line by line**

elif x == 17:

infoDict = df.loc[df['Staff'] == employee\_name.title(), :].to\_dict('list')

**# Iterate through dictionary and print line by line**

for key, value in infoDict.items():

print(f"{key}: {value[0]}")

**# Prevent users for typing in blanks**

elif x == "":

print("Please type in a number")

**# If user input an invalid question number**

else:

print("Invalid number, please try again")

main()

Output of Interactive User Input Query

1. User to input the name. (Eg. Aairah Hasting)



1. Once user selected Aairah Hasting, python will verify the selection by prompting user to select “1” to confirm.

A picture containing text

Description automatically generated

1. We can choose what we want to know about the employees with the options seen below. (Eg. 6: Get Citizenship of Staff)

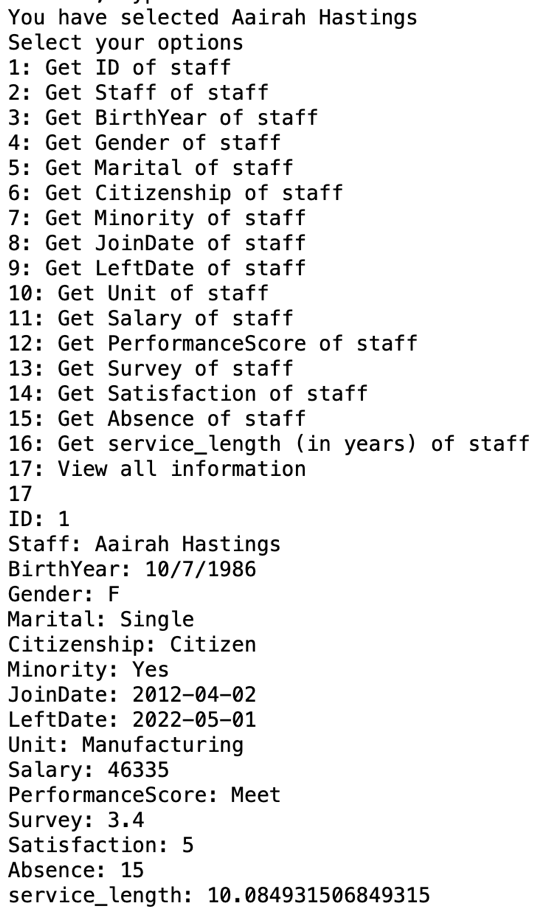
Text

Description automatically generated

Text

Description automatically generated

1. User can also choose to view all employee information by selecting option 17.



Potential Cases

1. When an unknown name is entered, python will indicate as “Invalid Input/Employee not found”

Text

Description automatically generated with medium confidence

1. When similar names are being entered, python will prompt user to select the within the options.

Text

Description automatically generated

**References**

*What is a KPI? definition, best practices, and examples*. Klipfolio. (n.d.). Retrieved August 12, 2022, from https://www.klipfolio.com/resources/articles/what-is-a-key-performance-indicator