****

**ANL252**

**Python for Data Analysis**

**Tutor-Marked Assignment**

**July 2022 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Kris Lennings** | **J1910196** |

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

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

**Table of Contents**

[**1. Qn1a 1**](#_Toc64796393)

[**2. Qn1b 3**](#_Toc64796394)

[**3. Qn1c 9**](#_Toc64796395)

[**4. Qn1d 10**](#_Toc64796403)

**Qn1a**

**i.**

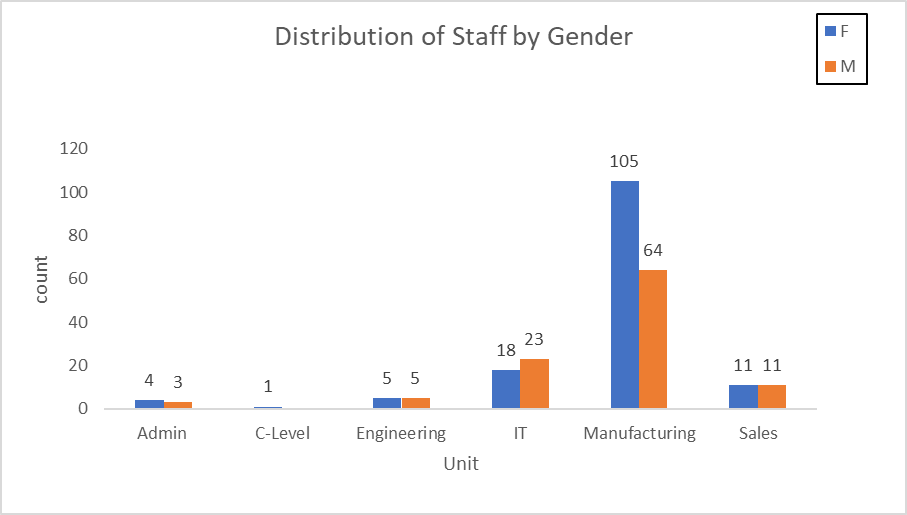
The TMA-dataset has 250 rows of data with 15 data fields. Their data types are described as follows:

|  |  |
| --- | --- |
| **Field** | **Data Type** |
| ID | Nominal |
| Staff | Nominal |
| Birth Year | Ordinal |
| Gender | Categorical |
| Marital | Nominal |
| Citizenship | Nominal |
| Minority | Categorical |
| Join Date | Ordinal |
| LeftDate | Ordinal |
| Unit | Nominal |
| Salary | Continuous |
| Performance Score | Categorical |
| Survey | Ordinal |
| Satisfaction | Ordinal |
| Absence | Continuous |

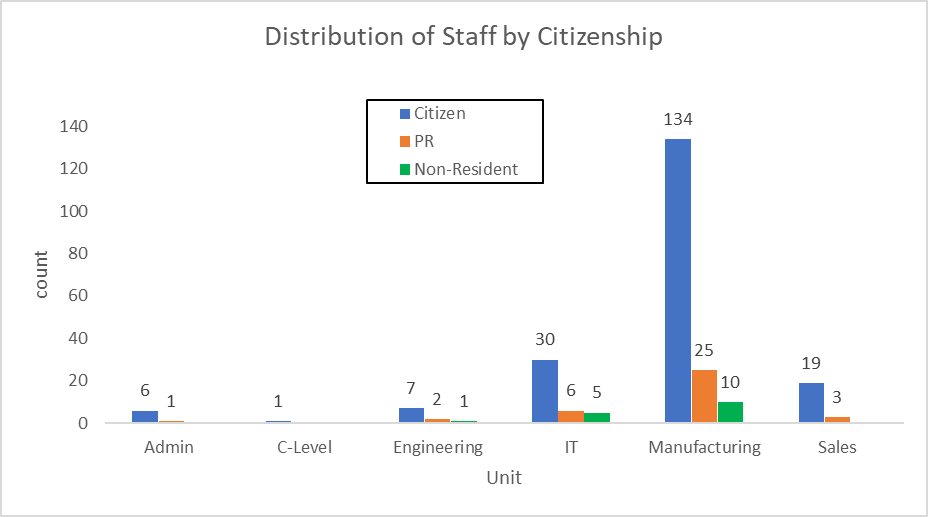
*Table 1. Data type for 15 data fields*

**ii.**

From the excel pivot table and math function, we can generate the following charts and table.



*Chart 1. A clustered column chart depicting the gender distribution by business unit*



*Chart 2. A clustered column chart depicting the citizenship distribution by business unit*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Admin | C-Level | Engineering | IT | Manufacturing | Sales | Total |
| Female | 4 | 1 | 5 | 18 | 105 | 11 | 144 |
| Male | 3 | 0 | 5 | 23 | 64 | 11 | 106 |
| Citizen | 6 | 1 | 7 | 30 | 134 | 19 | 197 |
| Non-Resident | 0 | 0 | 1 | 5 | 10 | 0 | 16 |
| PR | 1 | 0 | 2 | 6 | 25 | 3 | 37 |

*Table 2. Corresponding summarised tables of Chart 1 and Chart 2*

**iii.**

Based on Chart 1, we can assume that the company is in the manufacturing industry. This is due to the manufacturing unit, which employs approximately 169 workers out of the total 250 working population. Furthermore, the manufacturing unit is dominated by female employees. For the both the Engineering and Sales units, we can observe that the female-to-male staff ratio is nearly proportionate. On the other hand, the Admin and IT units have a slightly disproportionate female-to-male staff ratio.

Based on Chart 2, the company primarily employs their own citizens to build their workforce. The total number of citizens accounts for approximately 78 percent of the company's total workforce. This could imply that the company is a local business entity. The hiring pattern is seen to be consistent across all of the company's business units. We can observe that the vast majority of citizens are placed within the manufacturing unit. We can also observe that the company hires approximately 14 percent of non-citizens in the Administration unit, 30 percent in the Engineering unit, 27 percent in the IT unit, and 14 percent in the Sales unit. The data in Chart 1 and Chart 2 can also be found in the corresponding summarised table to view the overall number and distribution of workers in the company by gender and citizenship in supporting the company's business operation.

**Qn1b**

**i.**

In order to generate the above charts and tables, we need to import the following packages and modules into the Jupyter environment.

*# import pandas as pd: This module provides users to access the pandas module functions that enables them to read a csv or xlsx data file.*

*# import numpy as np: This module allows users to work the high multidimensional array and matrices contain in the data file.*

*# from matplotlib import pyplot as plt. This module allows users to create figures, plots and labels in the Jupyter environment*

*# import seaborn as sns. This module allows users to visualise data by creating of graphs and plots.*

*import pandas as pd*

*import numpy as np*

*from matplotlib import pyplot as plt*

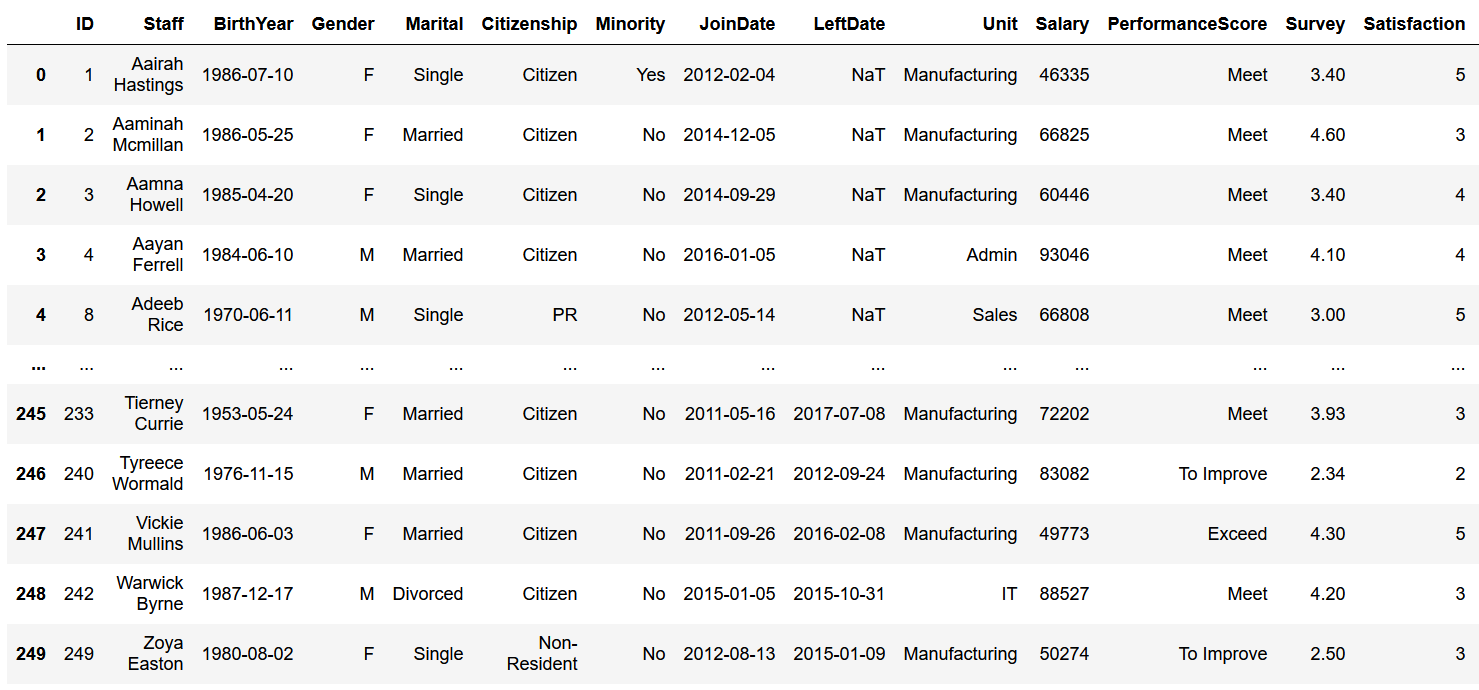
*import seaborn as sns*

**ii.**

Next, we need to load the TMA\_Data as a data frame (df) and inspect its structure.

*df = pd.read\_excel('D:\TMA\_Data.xlsx')*

*df*

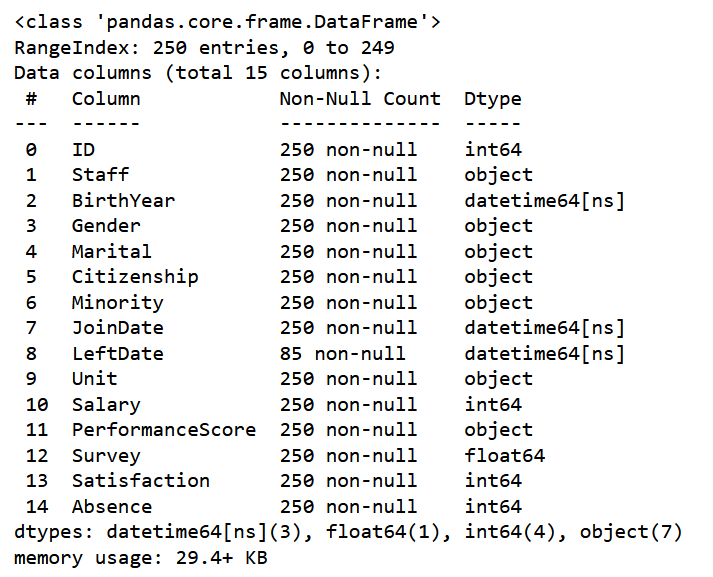


*Table 3. Table output with missing data (NaT)*

**iii.**

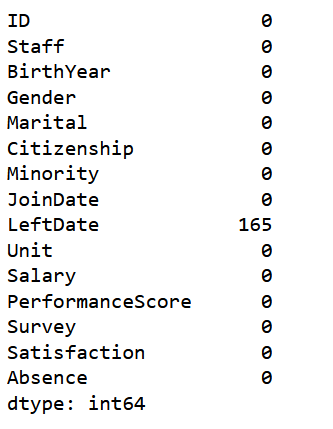
Next, we must explore the data types and check for any missing data in the data frame (df). This is to help users understand the data types and objects contained in the dataset. From the output in Figure 2, we can observe that the ‘LeftDate’ column contains about 165 (250-85) missing values.

*df.info()*



*Figure 1. Output obtained for df.info() code*

*df.isna().sum()*



*Figure 2. Output obtained for df.isna().sum() code*

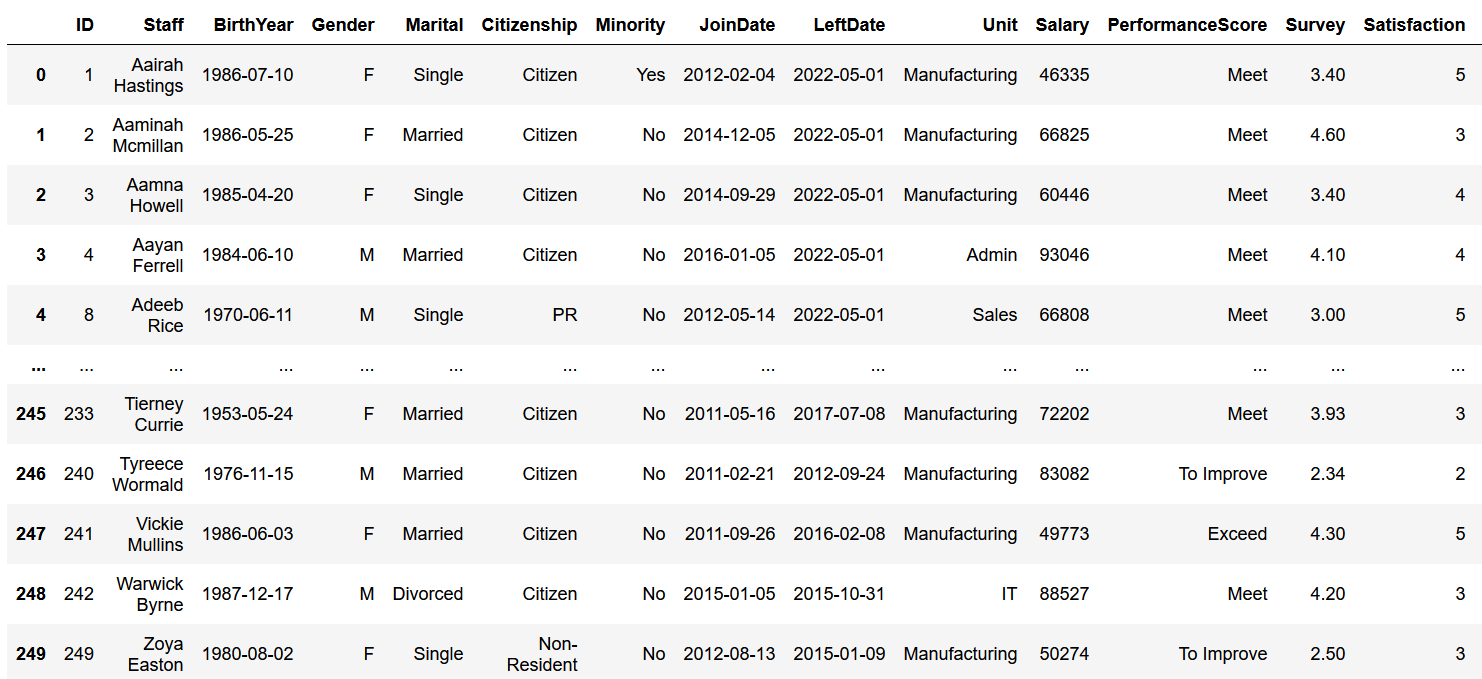
**iv.**

To replace the missing values in the ‘LeftDate’ column with 2022-05-01, we need to input the following codes below. Once the missing data has been replaced, we can begin generating the necessary charts and tables in the Jupyter environment.

*end = pd.to\_datetime('2022-05-01')*

*df['LeftDate'] = df['LeftDate'].fillna(end)*

*df*



*Table 4. Table output after missing data has been replaced*

**v.**

To generate the chart for ‘Distribution of Staff by Gender', we need to input the following codes below.

*Unit\_order = ['Admin','C-Level','Engineering','IT','Manufacturing','Sales']*

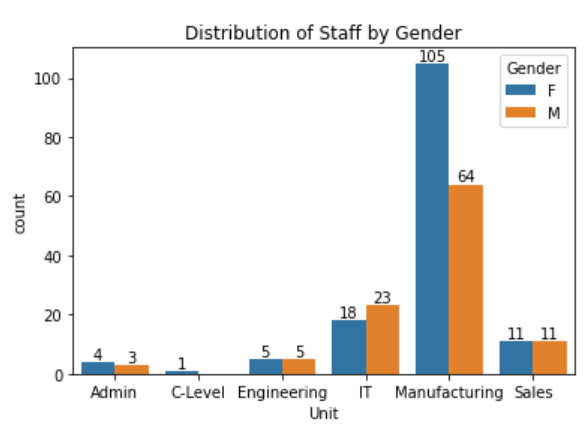
*ax = sns.countplot(x='Unit',data=df, hue='Gender', order=Unit\_order)*

*for container in ax.containers:*

*ax.bar\_label(container)*

*p = ax*

*p.set\_title("Distribution of Staff by Gender")*



*Figure 3. Output chart for ‘*Distribution of Staff by Gender'

**vi.**

To generate the chart for ‘Distribution of Staff by Citizenship', we need to input the following codes below.

*Unit\_order = ['Admin','C-Level','Engineering','IT','Manufacturing','Sales']*

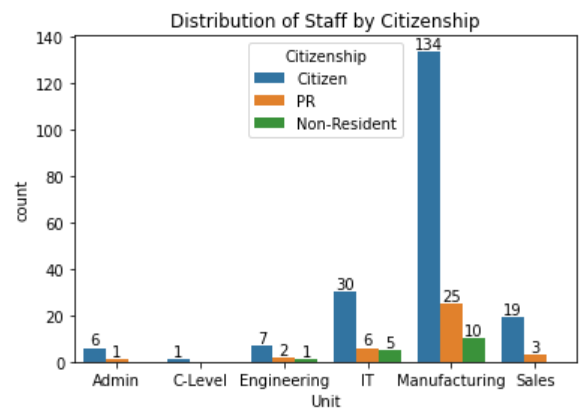
*ax = sns.countplot(x='Unit',data=df, hue='Citizenship', order=Unit\_order)*

*for container in ax.containers:*

*ax.bar\_label(container)*

*p = ax*

*p.set\_title("Distribution of Staff by Citizenship")*



*Figure 4. Output chart for ‘*Distribution of Staff by Citizenship'

**vii.**

Before we can generate the 'Corresponding Summarised tables,' we must first install the following packages in the Jupyter environment. First, we need to install the ‘pip install prettytable’ module. This module allows users to create an ASCII table in the Jupyter environment.

*pip install prettytable*

Next, to generate the ‘Corresponding Summarised tables’, we need to input the following codes below.

*from prettytable import PrettyTable*

*SummarisedTable = PrettyTable(['','Admin','C-Level','Engineering','IT', 'Manufacturing', 'Sales', 'Total'])*

*SummarisedTable.add\_row(['Female','4','1','5','18','105','11','144'])*

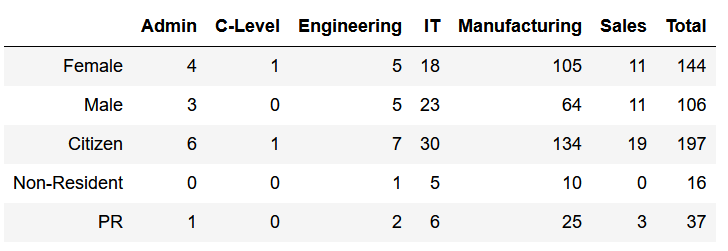
*SummarisedTable.add\_row(['Male','3','0','5','23','64','11','106'])*

*SummarisedTable.add\_row(['Citizen','6','1','7','30','134','19','197'])*

*SummarisedTable.add\_row(['Non-Resident','0','0','1','5','10','0','16'])*

*SummarisedTable.add\_row(['PR','1','0','2','6','25','3','37'])*

*(SummarisedTable)*



*Table 5. Output table for ‘Corresponding Summarised tables’*

**Qn1c**

**i.**

To calculate the length of service for all staff, we must create a new column that subtracts the values from the 'LeftDate' column from the values in the 'JoinDate’ column. The obtained subtracted values will then be converted into time-delta days before being divided by the value 365. The new column's final values will then be rounded to one decimal place.

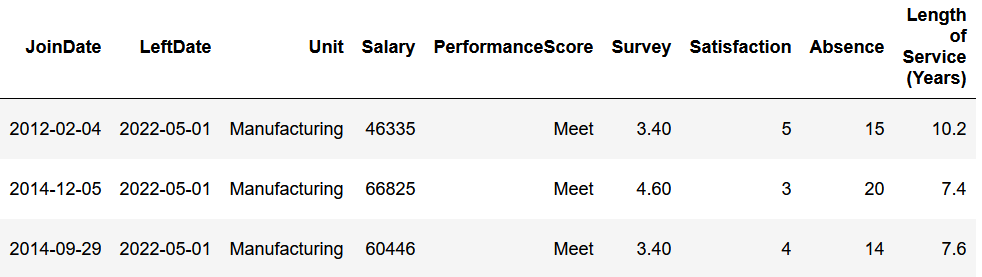
*df['Length of Service (Years)'] = (df['LeftDate'] - df['JoinDate'])*

*df['Length of Service (Years)'] = df['Length of Service (Years)'] / np.timedelta64(1, 'D')*

*df['Length of Service (Years)'] = df['Length of Service (Years)'] /365*

*df = df.round({'Length of Service (Years)': 1})*

*df*



*Table 6. Output table for ‘Length of Service (Years)’*

**ii.**

To generate the min, max and average ‘Length of Service (Years)’, we need to input the following codes below.

*df['Length of Service (Years)'].min(), df['Length of Service (Years)'].max(), df['Length of Service (Years)'].mean()*

*mean = df['Length of Service (Years)'].mean()*

*(round(mean,1))*

*print (f"The Min, Max and Avg Length of Service (Years) is {df['Length of Service (Years)'].min()}, {df['Length of Service (Years)'].max()} and {(round(mean,1))} respectively")*



*Figure 5. Output values for ‘Min, Max and Average Length of Service (Years)’*

**Qn1d**

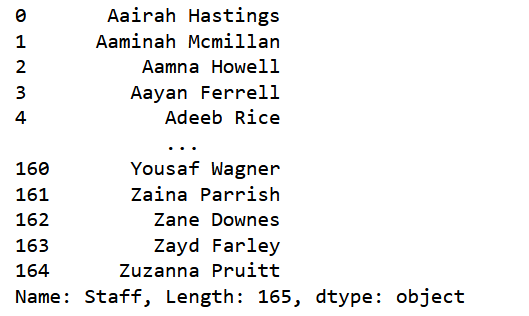
**i.**

We need to generate a list of ‘Previous\_Staff’ names so that users can enter their query about employees who have left the company. To accomplish this, we need to input the following codes below.

*df['Staff'].where(df['Unit'] == 'Previous\_Staff')*

*Previous\_Staff = df['Staff'].where(df['LeftDate'] == '2022-05-01')*

*print(Previous\_Staff.dropna())*



*Figure 6. Output values for ‘Previous\_Staff’ names*

After we have generated the list of 'Previous\_Staff' names, we need to enter the following codes so that the user can determine whether or not a particular person was an ex-member of the organization's staff. We also need to create a query statement in which users can enter the employees' names to determine their employment status.

*my\_list = list(Previous\_Staff)*

*while True:*

*inp = input("Press 0 to stop or Enter the name of the staff: ")*

*if (inp == "0"):*

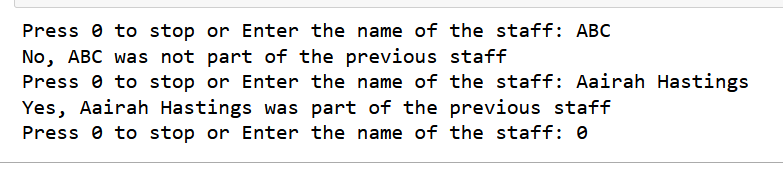
*break*

*if inp in my\_list:*

*print(f"Yes, {inp} was part of the previous staff")*

*else:*

*print(f"No, {inp} was not part of the previous staff")*

  
*Figure 7. User query output for different ‘Previous\_Staff’ names*

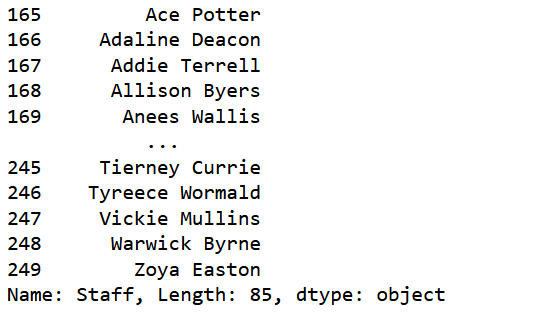
**ii.**

To allow users to query about employees who are still employed by the company, we need to generate a list of 'Current\_Staff' names. To accomplish this, we need to input the following codes below.

*df['Staff'].where(df['Unit'] == 'Current\_Staff')*

*Current\_Staff = df['Staff'].where(df['LeftDate'] != '2022-05-01')*

*print(Current\_Staff.dropna())*



*Figure 8. Output values for ‘Current\_Staff’ names*

After we have generated the list of 'Current\_Staff' names, we need to enter the following codes so that the user can determine whether or not a particular person is still a member of the organization's staff. We also need to create a query statement in which users can enter the employees' names to determine their employment status.

*my\_list2 = list(Current\_Staff)*

*while True:*

*inp = input("Press 0 to stop or Enter the name of the staff: ")*

*if (inp == "0"):*

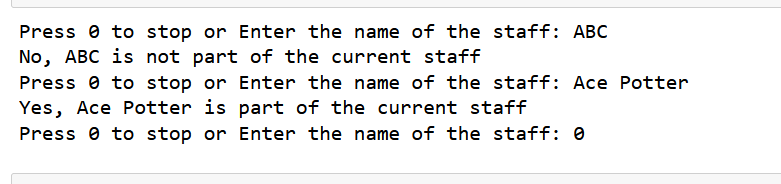
*break*

*if inp in my\_list2:*

*print(f"Yes, {inp} is part of the current staff")*

*else:*

*print(f"No, {inp} is not part of the current staff")*

  
*Figure 9. User query output for different ‘Current\_Staff’ names*