IBM Developer SKILLS NETWORK

AKRAM ALZAGHIR

Data Wrangling Lab

Estimated time needed: **45 to 60** minutes

In this assignment you will be performing data wrangling.

Objectives

In this lab you will perform the following:

- Identify duplicate values in the dataset.
- Remove duplicate values from the dataset.
- Identify missing values in the dataset.
- Impute the missing values in the dataset.
- Normalize data in the dataset.

Hands on Lab

Import pandas module.

In [78]:

```
import pandas as pd
import numpy as np  # useful for many scientific computing in Python
```

Load the dataset into a dataframe.

In [2]:

```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IB
M-DA0321EN-SkillsNetwork/LargeData/m1_survey_data.csv")
```

In [3]:

```
df.head(2)
```

Out[3]:

	Respondent	MainBranch	Hobbyist	OpenSourcer	OpenSource	Employment	Country	St
0	4	I am a developer by profession	No	Never	The quality of OSS and closed source software ...	Employed full-time	United States	
1	9	I am a developer by profession	Yes	Once a month or more often	The quality of OSS and closed source software ...	Employed full-time	New Zealand	

2 rows × 85 columns

In [4]:

```
df.shape # number of rows and columns
```

Out[4]:

```
(11552, 85)
```

Finding duplicates

In this section you will identify duplicate values in the dataset.

Find how many duplicate rows exist in the dataframe.

In [5]:

```
# your code goes here
df.duplicated()
```

Out[5]:

```
0        False
1        False
2        False
3        False
4        False
         ...
11547    False
11548    False
11549    False
11550    False
11551    False
Length: 11552, dtype: bool
```

In [6]:

```
df.duplicated().sum()
```

Out[6]:

```
154
```

Removing duplicates

Remove the duplicate rows from the dataframe.

In [7]:

```
# your code goes here
drop_dup=df.drop_duplicates().shape # shape is to display the number of rows and column
s
drop_dup
```

Out[7]:

```
(11398, 85)
```

In [8]:

```
#first : Drop duplicated values except for the first value.
#it apply this (keep="first") by default, evev if we did claim it
drop_dup=df.drop_duplicates(keep="first").shape
drop_dup
```

Out[8]:

```
(11398, 85)
```

In [9]:

```
#last : Drop duplicated values except for the last vlaue.
drop_dup=df.drop_duplicates(keep="last").shape
drop_dup
```

Out[9]:

```
(11398, 85)
```

In [10]:

```
#False : Drop all duplicated values.
drop_dup=df.drop_duplicates(keep=False).shape
drop_dup
```

Out[10]:

```
(11305, 85)
```

In [11]:

```
#To remove duplicates on specific column(s)
drop_dup1=df["Respondent"].drop_duplicates().shape
drop_dup1
```

Out[11]:

```
(11398,)
```

In [12]:

```
#To remove duplicates on specific column(s)
drop_dup1=df.drop_duplicates(subset=["Respondent"]).shape
drop_dup1
```

Out[12]:

```
(11398, 85)
```

Verify if duplicates were actually dropped.

In [13]:

```
# your code goes here
drop_dup
```

Out[13]:

```
(11305, 85)
```

Finding Missing values

Find the missing values for all columns.

In [14]:

```
# your code goes here
df_missing=df.isnull().sum() #to show the missing value for each column
df_missing
```

Out[14]:

```
Respondent        0
MainBranch        0
Hobbyist          0
OpenSourcer       0
OpenSource       81
                ...
Sexuality       547
Ethnicity       683
Dependents      144
SurveyLength     19
SurveyEase       14
Length: 85, dtype: int64
```

In [15]:

```
df_missing=df.isnull().tail() # to show the last 5 rows
df_missing
```

Out[15]:

	Respondent	MainBranch	Hobbyist	OpenSourcer	OpenSource	Employment	Country
11547	False	False	False	False	False	False	False
11548	False	False	False	False	False	False	False
11549	False	False	False	False	False	False	False
11550	False	False	False	False	False	False	False
11551	False	False	False	False	False	False	False

5 rows × 85 columns

Find out how many rows are missing in the column 'WorkLoc'

In [16]:

```
df["WorkLoc"].isnull().sum()
```

Out[16]:

```
32
```

In [17]:

```
df["Sexuality"].isnull().sum()
```

Out[17]:

```
547
```

In [18]:

```
df["Country"].isnull().sum()
```

Out[18]:

```
0
```

Imputing missing values

Find the value counts for the column WorkLoc.

In [19]:

```
# your code goes here
df_imputing=df['WorkLoc'].value_counts()
df_imputing
```

Out[19]:

```
Office                                            6905
Home                                              3638
Other place, such as a coworking space or cafe     977
Name: WorkLoc, dtype: int64
```

In [20]:

```
df_imputing=df['WorkLoc'].value_counts().sum() # total imputing missing values
df_imputing
```

Out[20]:

```
11520
```

In [21]:

```
df_imputing=df.value_counts(df['WorkLoc'])
df_imputing
```

Out[21]:

```
WorkLoc
Office                                            6905
Home                                              3638
Other place, such as a coworking space or cafe     977
dtype: int64
```

In [22]:

```
df_imputing=df.value_counts(df['WorkLoc']).sum()
df_imputing
```

Out[22]:

```
11520
```

Identify the value that is most frequent (majority) in the WorkLoc column.

In [23]:

```
#make a note of the majority value here, for future reference
df_imputing=df.value_counts(df['WorkLoc']).idxmax()
df_imputing
```

Out[23]:

```
'Office'
```

In [24]:

```
df_imputing=df.value_counts(df['Employment']).idxmax()
df_imputing
```

Out[24]:

```
'Employed full-time'
```

In [25]:

```
df_imputing=df.value_counts(df['UndergradMajor']).idxmax()
df_imputing
```

Out[25]:

```
'Computer science, computer engineering, or software engineering'
```

Impute (replace) all the empty rows in the column WorkLoc with the value that you have identified as majority.

In [26]:

```
# your code goes here
df_imputing_replace=df['WorkLoc'].fillna('Office', inplace = True)
df_imputing_replace
```

In [27]:

```
df["WorkLoc"].isnull().sum()
```

Out[27]:

```
0
```

In [28]:

```
#alt. approach for replacing the missing value
df_imputing_replace=df.fillna({'WorkLoc' : 'Office'}, inplace = True)
df_imputing_replace
```

After imputation there should ideally not be any empty rows in the WorkLoc column.

Verify if imputing was successful.

In [29]:

```
# your code goes here
df_imputing=df.value_counts(df['WorkLoc'])
df_imputing
```

Out[29]:

```
WorkLoc
Office                                              6937
Home                                                3638
Other place, such as a coworking space or cafe       977
dtype: int64
```

In [30]:

```
df_imputing=df["WorkLoc"].isnull().sum()
df_imputing
```

Out[30]:

```
0
```

Normalizing data

There are two columns in the dataset that talk about compensation.

One is "CompFreq". This column shows how often a developer is paid (Yearly, Monthly, Weekly).

The other is "CompTotal". This column talks about how much the developer is paid per Year, Month, or Week depending upon his/her "CompFreq".

This makes it difficult to compare the total compensation of the developers.

In this section you will create a new column called 'NormalizedAnnualCompensation' which contains the 'Annual Compensation' irrespective of the 'CompFreq'.

Once this column is ready, it makes comparison of salaries easy.

List out the various categories in the column 'CompFreq'

In [31]:

```
# your code goes here
pd.value_counts(df['CompFreq'])
```

Out[31]:

```
Yearly     6163
Monthly    4846
Weekly      337
Name: CompFreq, dtype: int64
```

In [33]:

```
pd.value_counts(df['CompFreq']).sum()
```

Out[33]:

```
11346
```

In [34]:

```
df['CompFreq'].unique()
```

Out[34]:

```
array(['Yearly', 'Monthly', 'Weekly', nan], dtype=object)
```

In [35]:

```
df.CompFreq.unique()
```

Out[35]:

```
array(['Yearly', 'Monthly', 'Weekly', nan], dtype=object)
```

In [35]:

```
df.CompFreq.unique().shape
```

Out[35]:

```
(4,)
```

Create a new column named 'NormalizedAnnualCompensation'. Use the hint given below if needed.

In [98]:

```
mean = df["CompTotal"].mean()
df["CompTotal"].replace(np.nan, mean, inplace = True)
df["NormalizedAnnualCompensation"] = df["CompTotal"]/df["CompTotal"].max()
print(df[["NormalizedAnnualCompensation"]])
```

```
       NormalizedAnnualCompensation
0                          0.000087
1                          0.000197
2                          0.000129
3                          0.000041
4                          0.000129
...                             ...
11547                      0.000186
11548                      0.000106
11549                      0.000150
11550                      0.000114
11551                      0.001071

[11552 rows x 1 columns]
```

In [94]:

```
df["NormalizedAnnualCompensation"] = (df["CompTotal"]-df["CompTotal"].min())/(df["CompT
otal"].max()-df["CompTotal"].min())
print(df[["NormalizedAnnualCompensation"]])
df.head(2)
```

```
       NormalizedAnnualCompensation
0                          0.000087
1                          0.000197
2                          0.000129
3                          0.000041
4                          0.000129
...                             ...
11547                      0.000186
11548                      0.000106
11549                      0.000150
11550                      0.000114
11551                      0.001071

[11552 rows x 1 columns]
```

Out[94]:

	Respondent	MainBranch	Hobbyist	OpenSourcer	OpenSource	Employment	Country	St
0	4	I am a developer by profession	No	Never	The quality of OSS and closed source software ...	Employed full-time	United States	
1	9	I am a developer by profession	Yes	Once a month or more often	The quality of OSS and closed source software ...	Employed full-time	New Zealand	

2 rows × 86 columns

Double click to see the **Hint**.

In [ ]:

```
# your code goes here
```

Authors

Ramesh Sannareddy

Other Contributors

Rav Ahuja

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-10-17	0.1	Ramesh Sannareddy	Created initial version of the lab