Data Wrangling Lab by Kate Rogatina

In this assignment we will be performing data wrangling.

Objectives

In this lab we 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.

Import pandas module.

[5]:

```
import pandas as pd
```

Load the dataset into a dataframe.

[6]:

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

Finding duplicates

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

Find how many duplicate rows exist in the dataframe.

[15]:

```
num_duplicates = df.duplicated().sum()
```

```
print(f"Number of duplicate rows in the DataFrame: {num_duplicates}")
df.shape
```

```
Number of duplicate rows in the DataFrame: 154 Out[15]: (11552, 85)
```

Removing duplicates

Remove the duplicate rows from the dataframe.

[8]:

```
df_no_duplicates = df.drop_duplicates()

# Print the shape of the DataFrame after removing duplicates
print(f"Shape of the DataFrame after removing duplicates: {df_no_duplicates.shape}”)
```

```
Shape of the DataFrame after removing duplicates: (11398, 85)
```

Verify if duplicates were actually dropped.
In [9]:

```
# your code goes here
# Verify if duplicates were dropped
if df.shape[0] > df_no_duplicates.shape[0]:
    print("Duplicates were successfully dropped.")
else:
    print("No duplicates were found.”)
```

```
Duplicates were successfully dropped.
```

Finding Missing values

Find the missing values for all columns.

In [11]:

```
missing_values = df.isnull().sum() missing_values
```

Out[11]:

```
Respondent         0
MainBranch         0
```

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

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

[12]:# Find the number of missing values in the 'WorkLoc' column

```
missing_values_workloc = df['WorkLoc'].isnull().sum()

# Print the number of missing values in the 'WorkLoc' column
print(f"Number of missing values in the 'WorkLoc' column: {missing_values_workloc}")
```

```
Number of missing values in the 'WorkLoc' column: 32
```

Imputing missing values

Find the value counts for the column WorkLoc.

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

Out[17]:

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

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

In [19]:

```
#make a note of the majority value here, for future reference
#'Office'
# Find the mode (most frequent value) in the 'WorkLoc' column
```

```
workloc_mode = df['WorkLoc'].mode().values[0]
workloc_mode
```

Out[19]:

```
'Office'
```

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

In [22]:

```
# Impute (replace) empty rows in the 'WorkLoc' column with the majority value
df['WorkLoc'].fillna(workloc_mode, inplace=True)
```

After imputation there should ideally not be any empty rows in the WorkLoc column.
Verify if imputing was successful.

In [23]:

```
missing_values_after_imputation = df['WorkLoc'].isnull().sum()
print(f"Number of missing values in the 'WorkLoc' column after imputation:
{missing_values_after_imputation}”)
```

```
Number of missing values in the 'WorkLoc' column after imputation: 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 [24]:

```
# List out the various categories in the 'CompFreq' column
compfreq_categories = df['CompFreq'].unique()

# Print the result
print("Various categories in the 'CompFreq' column:")
print(compfreq_categories)
```

```
Various categories in the 'CompFreq' column: ['Yearly' 'Monthly' 'Weekly' nan]
```

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

In [27]:

```
# Create a new column 'NormalizedAnnualCompensation'
# Assuming 'CompFreq' values are 'Yearly', 'Monthly', or 'Weekly'
df['NormalizedAnnualCompensation'] = df.apply(
    lambda row: row['CompTotal'] * 12 if row['CompFreq'] == 'Monthly' else
                row['CompTotal'] * 52 if row['CompFreq'] == 'Weekly' else
                row['CompTotal'] if row['CompFreq'] == 'Yearly' else None,
    axis=1
)

# Print the DataFrame with the new column
df[['CompTotal', 'CompFreq', 'NormalizedAnnualCompensation']].head()
```

Out[27]:

	CompTotal	CompFreq	NormalizedAnnualCompensation
0	61000.0	Yearly	61000.0
1	138000.0	Yearly	138000.0
2	90000.0	Yearly	90000.0
3	29000.0	Monthly	348000.0
4	90000.0	Yearly	90000.0