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.

[1]:
```
import pandas as pd
```

```
<ipython-input-1-7dd3504c366f>:1: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

Load the dataset into a dataframe.

Read Data

We utilize the `pandas.read_csv()` function for reading CSV files. However, in this version of the lab, which operates on JupyterLite, the dataset needs to be downloaded to the interface using the provided code below.

The functions below will download the dataset into your browser:

[2]:
```
from pyodide.http import pyfetch

async def download(url, filename):
    response = await pyfetch(url)
    if response.status == 200:
        with open(filename, "wb") as f:
            f.write(await response.bytes())
```

[3]:
```
file_path = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/LargeData/m1_survey_data.csv"
```

To obtain the dataset, utilize the download() function as defined above:

[4]:
```
await download(file_path, "m1_survey_data.csv")
file_name="m1_survey_data.csv"
```

Utilize the Pandas method read_csv() to load the data into a dataframe.

[5]:
```
df = pd.read_csv(file_name)
df.head()
```

[5]:

	Respondent	MainBranch	Hobbyist	OpenSourcer	OpenSource	Employment	Country	Student	EdLevel	UndergradMajor	...	WelcomeChange	SONewContent	Age
0	4	I am a developer by profession	No	Never	The quality of OSS and closed source software ...	Employed full-time	United States	No	Bachelor's degree (BA, BS, B.Eng., etc.)	Computer science, computer engineering, or sof...	...	Just as welcome now as I felt last year	Tech articles written by other developers;Indu...	22.0
1	9	I am a developer by	Yes	Once a month or	The quality of OSS and closed	Employed full-time	New Zealand	No	Some college/university study without	Computer science, computer	...	Just as welcome now as I felt last	NaN	23.0
2	13	I am a developer by profession	Yes	Less than once a month but more than once per ...	OSS is, on average, of HIGHER quality than pro...	Employed full-time	United States	No	Master's degree (MA, MS, M.Eng., MBA, etc.)	Computer science, computer engineering, or sof...	...	Somewhat more welcome now than last year	Tech articles written by other developers;Cour...	28.0	Man	No
3	16	I am a developer by profession	Yes	Never	The quality of OSS and closed source software ...	Employed full-time	United Kingdom	No	Master's degree (MA, MS, M.Eng., MBA, etc.)	NaN	...	Just as welcome now as I felt last year	Tech articles written by other developers;Indu...	26.0	Man	No
4	17	I am a developer by profession	Yes	Less than once a month but more than once per ...	The quality of OSS and closed source software ...	Employed full-time	Australia	No	Bachelor's degree (BA, BS, B.Eng., etc.)	Computer science, computer engineering, or sof...	...	Just as welcome now as I felt last year	Tech articles written by other developers;Indu...	29.0	Man	No

5 rows × 85 columns

> Note: This version of the lab is working on JupyterLite, which requires the dataset to be downloaded to the interface.While working on the downloaded version of this notebook on their local machines(Jupyter Anaconda), the learners can simply **skip the steps above,** and simply use the URL directly in the `pandas.read_csv()` function. You can uncomment and run the statements in the cell below.

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

[8]:
```
Index(['Respondent', 'MainBranch', 'Hobbyist', 'OpenSourcer', 'OpenSource',
       'Employment', 'Country', 'Student', 'EdLevel', 'UndergradMajor',
       'EduOther', 'OrgSize', 'DevType', 'YearsCode', 'Age1stCode',
       'YearsCodePro', 'CareerSat', 'JobSat', 'MgrIdiot', 'MgrMoney',
       'MgrWant', 'JobSeek', 'LastHireDate', 'LastInt', 'FizzBuzz',
       'JobFactors', 'ResumeUpdate', 'CurrencySymbol', 'CurrencyDesc',
       'CompTotal', 'CompFreq', 'ConvertedComp', 'WorkWeekHrs', 'WorkPlan',
       'WorkChallenge', 'WorkRemote', 'WorkLoc', 'ImpSyn', 'CodeRev',
       'CodeRevHrs', 'UnitTests', 'PurchaseHow', 'PurchaseWhat',
       'LanguageWorkedWith', 'LanguageDesireNextYear', 'DatabaseWorkedWith',
       'DatabaseDesireNextYear', 'PlatformWorkedWith',
       'PlatformDesireNextYear', 'WebFrameWorkedWith',
       'WebFrameDesireNextYear', 'MiscTechWorkedWith',
       'MiscTechDesireNextYear', 'DevEnviron', 'OpSys', 'Containers',
       'BlockchainOrg', 'BlockchainIs', 'BetterLife', 'ITperson', 'OffOn',
       'SocialMedia', 'Extraversion', 'ScreenName', 'SOVisit1st',
       'SOVisitFreq', 'SOVisitTo', 'SOFindAnswer', 'SOTimeSaved',
       'SOHowMuchTime', 'SOAccount', 'SOPartFreq', 'SOJobs', 'EntTeams',
       'SOComm', 'WelcomeChange', 'SONewContent', 'Age', 'Gender', 'Trans',
       'Sexuality', 'Ethnicity', 'Dependents', 'SurveyLength', 'SurveyEase'],
      dtype='object')
```

Finding duplicates

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

Find how many duplicate rows exist in the dataframe.

[9]:
```
# your code goes here
len(df)-len(df.drop_duplicates())
```

[9]:
```
154
```

Removing duplicates ¶

Remove the duplicate rows from the dataframe.

[10]:
```
# your code goes here
df = df.drop_duplicates()
```

Verify if duplicates were actually dropped.

[11]:
```
# your code goes here
if df.duplicated().any():
    print("Duplicates still exist in the DataFrame.")
else:
    print("No duplicates found.")
```

```
No duplicates found.
```

Finding Missing values

Find the missing values for all columns.

[12]:
```
# your code goes here
df.isnull()
```

	Respondent	MainBranch	Hobbyist	OpenSourcer	OpenSource	Employment	Country	Student	EdLevel	UndergradMajor	...	WelcomeChange	SONewContent	Age	Gender	Trans	Se
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	...	False	True	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	True	...	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11547	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
11548	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
11549	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
11550	False	False	False	False	False	False	False	False	False	True	...	False	True	False	False	False	
11551	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	

11398 rows × 85 columns

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

```
# your code goes here
len(df['Country'])-len(df['Country'].dropna())
```

```
0
```

Imputing missing values ¶

Find the value counts for the column WorkLoc.

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

```
WorkLoc
Office                                        6806
Home                                          3589
Other place, such as a coworking space or cafe     971
Name: count, dtype: int64
```

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

```
#make a note of the majority value here, for future reference
max = df['ConvertedComp'].median()
print(max)
```

```
57745.0
```

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

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

```
<ipython-input-16-768b06206376>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the oper
ation inplace on the original object.


  df['WorkLoc'].fillna('Office', inplace=True)
```

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

Verify if imputing was successful.

[17]:
```
# your code goes here
df['WorkLoc'].isna().sum()
```

[17]:
```
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'

[18]:
```
# your code goes here
df['CompFreq'].unique()
```

[18]:
```
array(['Yearly', 'Monthly', 'Weekly', nan], dtype=object)
```

[41]:
```
df['CompFreq'].dropna()
```

[41]:
```
0         Yearly
1         Yearly
2         Yearly
3        Monthly
4         Yearly
           ...
11546    Monthly
11547     Yearly
11548     Yearly
11549     Yearly
11550     Yearly
Name: CompFreq, Length: 11192, dtype: object
```

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

Double click to see the **Hint**.

[20]:
```
# your code goes here
annualcompensation = []

def anncomp():
    for x,y in zip(df['CompFreq'], df['CompTotal']):
        if x=='Monthly':
            annualcompensation.append(y*12)
        elif x=='Weekly':
            annualcompensation.append(y*52)
        else:
            annualcompensation.append(y)

anncomp()

df['NormalizedAnnualCompensation']=annualcompensation
df[['NormalizedAnnualCompensation']]
```

[20]:

	NormalizedAnnualCompensation
0	61000.0
1	138000.0
2	90000.0
3	348000.0
4	90000.0
...	...
11547	130000.0
11548	74400.0
11549	105000.0
11550	80000.0
11551	NaN

11398 rows × 1 columns

[46]:
```
df['NormalizedAnnualCompensation'].median()
```

[46]:
```
100000.0
```