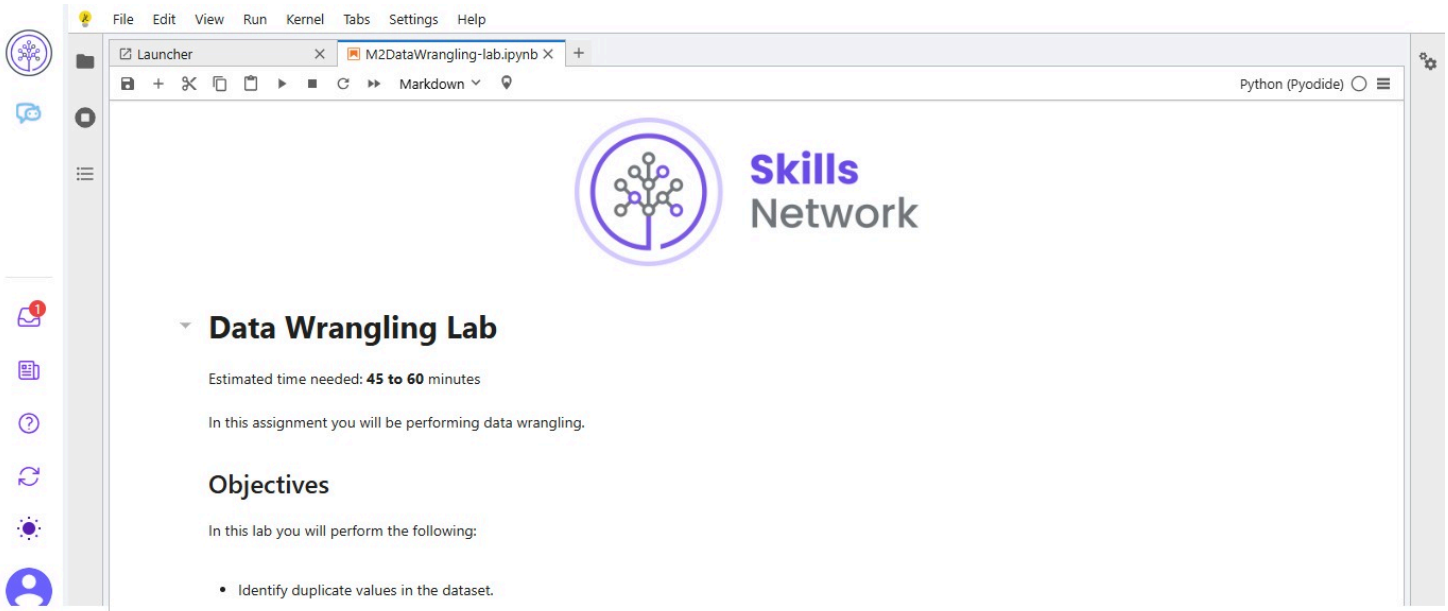



# Data Wrangling Lab

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.

[19]:
```
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:

[20]:
```
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())
```

[21]:
```
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:

[22]:
```
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.

[23]:
```
df = pd.read_csv(file_name)
```

> 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.

[24]:
```
#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 you will identify duplicate values in the dataset.

Find how many duplicate rows exist in the dataframe.

[25]:
```
# your code goes here
df.duplicated().sum()
```

[25]: 154

## Removing duplicates

Remove the duplicate rows from the dataframe.

[26]:
```
# your code goes here
df.drop_duplicates(inplace=True)
```

Verify if duplicates were actually dropped.

[27]:
```
# your code goes here
df.duplicated().sum()
```

[27]: 0

## Finding Missing values

Find the missing values for all columns.

[28]:
```
# your code goes here
print(len(df.index))
```

```
11398
```

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

[29]:
```
# your code goes here
missing_data = df.isnull()

for column in missing_data.columns.values.tolist():
    print(column)
    print (missing_data[column].value_counts())
    print("")
```

```
Respondent
Respondent
False    11398
Name: count, dtype: int64

MainBranch
MainBranch
False    11398
Name: count, dtype: int64

Hobbyist
Hobbyist
False    11398
Name: count, dtype: int64

OpenSourcer
OpenSourcer
False    11398
Name: count, dtype: int64

OpenSource
OpenSource
False    11317
True        81
Name: count, dtype: int64

Employment
Employment
False    11398
Name: count, dtype: int64

Country
Country
False    11398
Name: count, dtype: int64

Student
Student
False    11347
True        51
Name: count, dtype: int64

EdLevel
EdLevel
False    11286
True       112
Name: count, dtype: int64

UndergradMajor
UndergradMajor
False    10661
True       737
Name: count, dtype: int64

EduOther
EduOther
False    11234
True       164
Name: count, dtype: int64

OrgSize
OrgSize
False    11302
True        96
Name: count, dtype: int64
```

```
DevType
DevType
False    11333
True        65
Name: count, dtype: int64

YearsCode
YearsCode
False    11389
True         9
Name: count, dtype: int64

Age1stCode
Age1stCode
False    11385
True        13
Name: count, dtype: int64

YearsCodePro
YearsCodePro
False    11382
True        16
Name: count, dtype: int64

CareerSat
CareerSat
False    11398
Name: count, dtype: int64

JobSat
JobSat
False    11397
True         1
Name: count, dtype: int64

MgrIdiot
MgrIdiot
False    10905
True       493
Name: count, dtype: int64

MgrMoney
MgrMoney
False    10901
True       497
Name: count, dtype: int64

MgrWant
MgrWant
False    10905
True       493
Name: count, dtype: int64

JobSeek
JobSeek
False    11398
Name: count, dtype: int64

LastHireDate
LastHireDate
False    11398
Name: count, dtype: int64

LastInt
LastInt
False    10985
True       413
Name: count, dtype: int64

FizzBuzz
FizzBuzz
False    11361
True        37
Name: count, dtype: int64

JobFactors
JobFactors
False    11395
True         3
Name: count, dtype: int64

ResumeUpdate
ResumeUpdate
False    11359
True        39
Name: count, dtype: int64

CurrencySymbol
CurrencySymbol
False    11398
Name: count, dtype: int64

CurrencyDesc
CurrencyDesc
False    11398
Name: count, dtype: int64

CompTotal
CompTotal
False    10589
True       809
Name: count, dtype: int64

CompFreq
CompFreq
False    11192
True       206
Name: count, dtype: int64

ConvertedComp
```

```
ConvertedComp
False    10582
True       816
Name: count, dtype: int64

WorkWeekHrs
WorkWeekHrs
False    11276
True       122
Name: count, dtype: int64

WorkPlan
WorkPlan
False    11277
True       121
Name: count, dtype: int64

WorkChallenge
WorkChallenge
False    11234
True       164
Name: count, dtype: int64

WorkRemote
WorkRemote
False    11390
True         8
Name: count, dtype: int64

WorkLoc
WorkLoc
False    11366
True        32
Name: count, dtype: int64

ImpSyn
ImpSyn
False    11393
True         5
Name: count, dtype: int64

CodeRev
CodeRev
False    11397
True         1
Name: count, dtype: int64

CodeRevHrs
CodeRevHrs
False    8972
True     2426
Name: count, dtype: int64

UnitTests
UnitTests
False    11369
True        29
Name: count, dtype: int64

PurchaseHow
PurchaseHow
False    11202
True       196
Name: count, dtype: int64

PurchaseWhat
PurchaseWhat
False    11360
True        38
Name: count, dtype: int64

LanguageWorkedWith
LanguageWorkedWith
False    11387
True        11
Name: count, dtype: int64

LanguageDesireNextYear
LanguageDesireNextYear
False    11264
True       134
Name: count, dtype: int64

DatabaseWorkedWith
DatabaseWorkedWith
False    10945
True       453
Name: count, dtype: int64

DatabaseDesireNextYear
DatabaseDesireNextYear
False    10356
True      1042
Name: count, dtype: int64

PlatformWorkedWith
PlatformWorkedWith
False    10987
True       411
Name: count, dtype: int64

PlatformDesireNextYear
PlatformDesireNextYear
False    10854
True       544
Name: count, dtype: int64

WebFrameWorkedWith
WebFrameWorkedWith
```

```
False    10005
True      1393
Name: count, dtype: int64

WebFrameDesireNextYear
WebFrameDesireNextYear
False    9781
True     1617
Name: count, dtype: int64

MiscTechWorkedWith
MiscTechWorkedWith
False    9216
True     2182
Name: count, dtype: int64

MiscTechDesireNextYear
MiscTechDesireNextYear
False    9943
True     1455
Name: count, dtype: int64

DevEnviron
DevEnviron
False    11369
True        29
Name: count, dtype: int64

OpSys
OpSys
False    11364
True        34
Name: count, dtype: int64

Containers
Containers
False    11316
True        82
Name: count, dtype: int64

BlockchainOrg
BlockchainOrg
False    9076
True     2322
Name: count, dtype: int64

BlockchainIs
BlockchainIs
False    8788
True     2610
Name: count, dtype: int64

BetterLife
BetterLife
False    11300
True        98
Name: count, dtype: int64

ITperson
ITperson
False    11363
True        35
Name: count, dtype: int64

OffOn
OffOn
False    11360
True        38
Name: count, dtype: int64

SocialMedia
SocialMedia
False    11105
True       293
Name: count, dtype: int64

Extraversion
Extraversion
False    11378
True        20
Name: count, dtype: int64

ScreenName
ScreenName
False    10891
True       507
Name: count, dtype: int64

SOVisit1st
SOVisit1st
False    11073
True       325
Name: count, dtype: int64

SOVisitFreq
SOVisitFreq
False    11393
True         5
Name: count, dtype: int64

SOVisitTo
SOVisitTo
False    11397
True         1
Name: count, dtype: int64

SOFindAnswer
SOFindAnswer
False    11395
```

```
True         5
Name: count, dtype: int64

SOTimeSaved
SOTimeSaved
False    11348
True        50
Name: count, dtype: int64

SOHowMuchTime
SOHowMuchTime
False    9481
True     1917
Name: count, dtype: int64

SOAccount
SOAccount
False    11397
True         1
Name: count, dtype: int64

SOPartFreq
SOPartFreq
False    10270
True      1128
Name: count, dtype: int64

SOJobs
SOJobs
False    11392
True         6
Name: count, dtype: int64

EntTeams
EntTeams
False    11393
True         5
Name: count, dtype: int64

SOComm
SOComm
False    11398
Name: count, dtype: int64

WelcomeChange
WelcomeChange
False    11313
True        85
Name: count, dtype: int64

SONewContent
SONewContent
False    9433
True     1965
Name: count, dtype: int64

Age
Age
False    11111
True       287
Name: count, dtype: int64

Gender
Gender
False    11325
True        73
Name: count, dtype: int64

Trans
Trans
False    11275
True       123
Name: count, dtype: int64

Sexuality
Sexuality
False    10856
True       542
Name: count, dtype: int64

Ethnicity
Ethnicity
False    10723
True       675
Name: count, dtype: int64

Dependents
Dependents
False    11258
True       140
Name: count, dtype: int64

SurveyLength
SurveyLength
False    11379
True        19
Name: count, dtype: int64

SurveyEase
SurveyEase
False    11384
True        14
Name: count, dtype: int64
```

## Imputing missing values

Find the value counts for the column WorkLoc.

[30]:
```
# your code goes here
df['WorkLoc'].isna().sum()
```

[30]:
```
32
```

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

[31]:
```
#make a note of the majority value here, for future reference
#
majority = df['WorkLoc'].value_counts().idxmax()
print(majority)
```

```
Office
```

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

[35]:
```
# your code goes here
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.

[36]:
```
# your code goes here
df['WorkLoc'].isna().sum()
```

[36]:
```
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'

[37]:
```
# your code goes here
df['CompFreq'].unique()
```

[37]:
```
array(['Yearly', 'Monthly', 'Weekly', nan], dtype=object)
```

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

```
Double click to see the **Hint**.

<!--

Use the below logic to arrive at the values for the column NormalizedAnnualCompensation.

If the CompFreq is Yearly then use the exising value in CompTotal
If the CompFreq is Monthly then multiply the value in CompTotal with 12 (months in an year)
If the CompFreq is Weekly then multiply the value in CompTotal with 52 (weeks in an year)

-->
```

[38]:
```
# your code goes here
df['CompFreq'].value_counts()
```

[38]:
```
CompFreq
Yearly     6073
Monthly    4788
Weekly      331
Name: count, dtype: int64
```

[39]:
```
def conditions(s):
  if (s['CompFreq'] == 'Yearly'):
    return s['CompTotal']
  elif (s['CompFreq'] == 'Monthly'):
    return (s['CompTotal'] * 12)
  else:
    return (s['CompTotal'] * 52)

df['NormalizedAnnualCompensation'] = df.apply(conditions, axis=1)
df.head()
```

[39]:

| | Respondent | MainBranch | Hobbyist | OpenSourcer | OpenSource | Employment | Country | Student | EdLevel | UndergradMajor | ... | SONewContent | Age | Gender |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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... | ... | Tech articles written by other developers;Indu... | 22.0 | Man |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 | No | Some college/university study without earning ... | Computer science, computer engineering, or sof... | ... | NaN | 23.0 | Man |
| **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... | ... | Tech articles written by other developers;Cour... | 28.0 | Man |
| **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 | ... | Tech articles written by other developers;Indu... | 26.0 | Man |
| **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... | ... | Tech articles written by other developers;Indu... | 29.0 | Man |

5 rows × 86 columns

[40]:
```
df[['CompFreq', 'CompTotal', 'NormalizedAnnualCompensation']].head()
```

[40]:

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

[41]:
```
df['NormalizedAnnualCompensation'].describe()
```

[41]:
```
count    1.058900e+04
mean     6.170771e+06
std      9.842866e+07
min      0.000000e+00
25%      5.200000e+04
50%      1.000000e+05
75%      3.600000e+05
max      8.400000e+09
Name: NormalizedAnnualCompensation, dtype: float64
```

## Authors

Ramesh Sannareddy

### Other Contributors

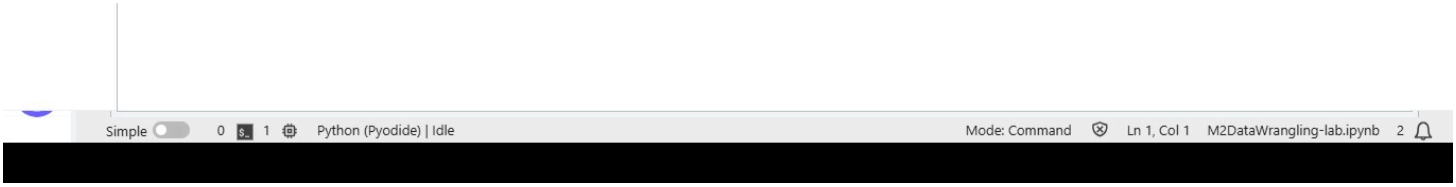
Rav Ahuja

## Change Log

| Date (YYYY-MM-DD) | Version | Changed By | Change Description |
|---|---|---|---|
| 2024-09-24 | 1.1 | Madhusudhan Moole | Updated lab |
| 2024-09-23 | 1.0 | Raghul Ramesh | Created lab |



[ ]:


Simple
0
1
Python (Pyodide) | Idle
Mode: Command
Ln 1, Col 1
M2DataWrangling-lab.ipynb
2