**CHAPTER 3**

**DATA WRANGLING**

**3.1 Data Wrangling**

Data wrangling, sometimes referred to as data munging, is the process of transforming and mapping data from one "raw" data form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes such as analytics. The goal of data wrangling is to assure quality and useful data. Data analysts typically spend the majority of their time in the process of data wrangling compared to the actual analysis of the data.

Till now we have a dataset which need to be pre-processed and cleaned before going to analysis phase. Dataset may contain missing values, outliers, and other in-consistencies. Dataset is processed into different processes until it is declared as cleaned.

**3.2 Removing Duplicates.**

"Duplication" just means that you have repeated data in your dataset. This could be due to things like data entry errors or data collection methods. For example, if you're using a web scraper you may happen to scrape the same webpage more than once, or the same information from two different pages.

First, in data wrangling or pre-processing we need to identify the duplicates first and their numbers:

# get the number of duplicates

duplicates = df\_final[df.duplicated()]

print(len(duplicates))

0

**Code Snap 2.1. Removing Duplicates**

Above code snippets shows that there are No duplicates values in the dataset which need to be removed:

## 2.3.2 Finding Missing values

In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data.

Sometimes missing values are caused by the researcher—for example, when data collection is done improperly or mistakes are made in data entry.

Now the duplicates have been removed and we may have the missing values for each column as:

df\_final.isnull().sum()

The number of missing values for each column are as :

Respondent 0

Age 19015

ConvertedComp 29705

CompFreq 24392

CompTotal 29635

Country 389

DatabaseDesireNextYear 20391

DatabaseWorkedWith 14924

DevType 15091

EdLevel 7030

Gender 13904

LanguageDesireNextYear 10348

LanguageWorkedWith 7083

MiscTechDesireNextYear 22082

MiscTechWorkedWith 24147

OpSys 8233

PlatformDesireNextYear 13856

PlatformWorkedWith 10618

UndergradMajor 13466

WebframeDesireNextYear 24437

WebframeWorkedWith 22182

WorkWeekHrs 23310

YearsCodePro 18112

**Code Snap 2.2 Finding Missing Values**

## 2.3.3 Imputing missing values

We need to replace the missing values with appropriate values i.e with max value or simply drop these rows as:

Hence, we have 0 null or missing values.

df\_final.dropna(axis=0,inplace=True)

df\_final.reset\_index(drop=True,inplace=True)

df\_final.isnull().sum()

Respondent 0

Age 0

ConvertedComp 0

CompFreq 0

CompTotal 0

Country 0

DatabaseDesireNextYear 0

DatabaseWorkedWith 0

DevType 0

EdLevel 0

Gender 0

LanguageDesireNextYear 0

LanguageWorkedWith 0

MiscTechDesireNextYear 0

MiscTechWorkedWith 0

OpSys 0

PlatformDesireNextYear 0

PlatformWorkedWith 0

UndergradMajor 0

WebframeDesireNextYear 0

WebframeWorkedWith 0

WorkWeekHrs 0

YearsCodePro 0

**Code Snap 2.3 Imputing Missing Values**

**2.3.4 Normalizing Data**

Dataset normalization is the process of structuring a dataset, usually a relational dataset, in accordance with a series of so-called normal forms in order to reduce data redundancy

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':

Df\_final['CompFreq'].value\_counts()

Yearly 6073

Monthly 4788

Weekly 331

Name: CompFreq, dtype: int64

**Code Snap 2.4 Count values for each type**

Now we must normalize this column as:

conditions = [(df\_final['CompFreq']=='Yearly'),

(df\_final['CompFreq']=='Monthly'),

(df\_final['CompFreq']=='Weekly') ]

values = [(df\_final['CompTotal']),

(df\_final['CompTotal']\*12),

(df\_final['CompTotal']\*52)]

df\_final['NormalizedAnnualCompensation'] = np.select(conditions,values)

df\_final[['NormalizedAnnualCompensation']].head()

NormalizedAnnualCompensation

0 3.824729e+243

1 1.160000e+05

2 2.500000e+04

3 3.100000e+04

4 6.600000e+04

**Code Snap 2.5 Normalizing Data**

**2.3.5 Outliers**

In statistics, an outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. An outlier can cause serious problems in statistical analyses.

We have to find outliers also for a fair and accurate data analysis so we are now identifying the ootliers first and then we have to remove them.

Find out if outliers exist in the column ConvertedComp using a box plot?

A picture containing graphical user interface

Description automatically generated

**Figure 2.1 Box Plot of ConvertedComp**

Find out the Inter Quartile Range for the column `ConvertedComp`:

Q1 = df['ConvertedComp'].quantile(0.25)

Q3 = df['ConvertedComp'].quantile(0.75)

IQR = Q3 - Q1

IQR

77306.05014961446

**Code Snap 2.6 Finding IQR**

Find out the upper and lower bounds.

**Code Snap 2.7 Finding Upper and Lower bounds**

upper = Q3+(1.5\*IQR)

lower = Q1-(1.5\*IQR)

upper,lower

(209698.0, -82830.0)

Identify how many outliers are there in the ConvertedComp column.

**Code Snap 2.8 Length of outliers**

outliers = [x for x in df['ConvertedComp'] if x < lower or x > upper]

print(len(outliers))

661

Create a new data frame by removing the outliers from the ConvertedComp column and replace it with original column in the dataset.

**Code Snap 2.9 Replacing datasets**

new\_df = [x for x in df['ConvertedComp'] if x >= lower and x <= upper]

new\_df = pd.DataFrame(new\_df)

new\_df.columns = ['ConvertedComp']

new\_df = df\_final[‘ConvertedComp’]

Till now we have removed all the outliers and dataset is free from outliers.