DEAKIN UNIVERSITY

Student Name (Data Scientist)	Muhammad Akbar Husnoo
Student ID	219306444
Module Name	Modern Data Science
Module Code	SIT742
Assessment	Task 1: Exploration of Data Scientist Survey Data.
Lecturer	Associate Professor Dr Gang Li
Tutor	Sandya De Alwis
Lab Session	Tuesday [19:00 – 20.00]
Report Name	Exploration & Analysis of Kaggle Data Scientists Information Data

TABLE OF CONTENTS

1.0. INTRODUCTION

A survey was conducted by Kaggle in 2017 on their users who were registered as data scientists. The data gathered included the skillset of the data scientists, the demographic of the data scientists, the feedback of the platform and so on. This assessment, scripted using Python 3 and Jupyter Notebook, consists of two parts namely:

- **Part 1**: Performing exploratory data analysis and visualisation on two datasets, MCQResponses.csv and ConversionRates.csv.
- **Part 2**: Performing text analysis on JobPostings.csv, from Job Pikr.

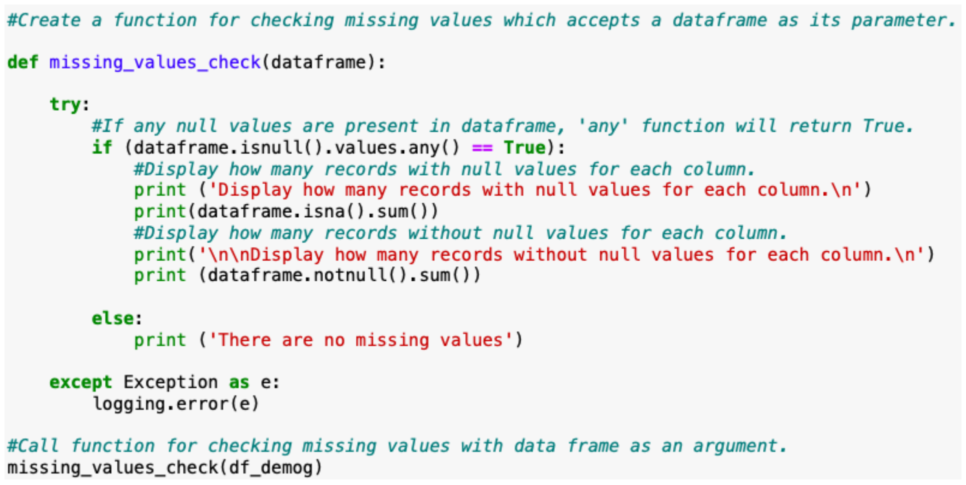
This report addresses each required section thoroughly as shown in the following.

2.0. PART 1: DATA ANALYSIS & VISUALISATION

This section deals with the analysis of the data gathered in the aim of producing some insights about Data Scientists on the Kaggle platform.

2.1. 1.0A: Missing Values Check

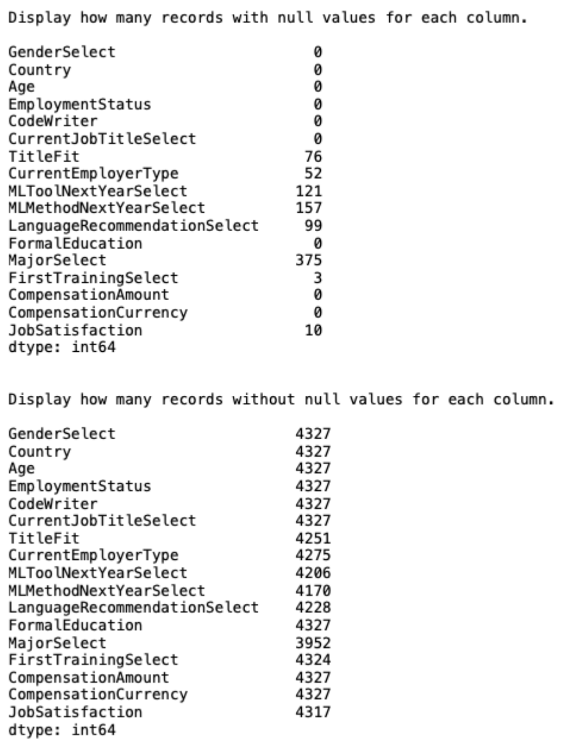
During this task, the data scientist was required to pre-process the data by checking which columns have null values and display how many records with null values.



```python
#Create a function for checking missing values which accepts a dataframe as its parameter.

def missing_values_check(dataframe):

    try:
        #If any null values are present in dataframe, 'any' function will return True.
        if (dataframe.isnull().values.any() == True):
            #Display how many records with null values for each column.
            print ('Display how many records with null values for each column.\n')
            print(dataframe.isna().sum())
            #Display how many records without null values for each column.
            print('\n\nDisplay how many records without null values for each column.\n')
            print (dataframe.notnull().sum())

        else:
            print ('There are no missing values')

    except Exception as e:
        logging.error(e)

#Call function for checking missing values with data frame as an argument.
missing_values_check(df_demog)
```

Figure 1: Code Snippet for 1.0A

Based on the above code snippet, the data scientist created a function which accepts a data frame as parameter. The usage of effective error-handling (try-except block) prevents abnormal termination of the operation. Using a condition-based (if-else) statement to check if the data frame has any NA values, a list of columns with a count of null values is displayed followed by a list of columns with a count of not null values is displayed.



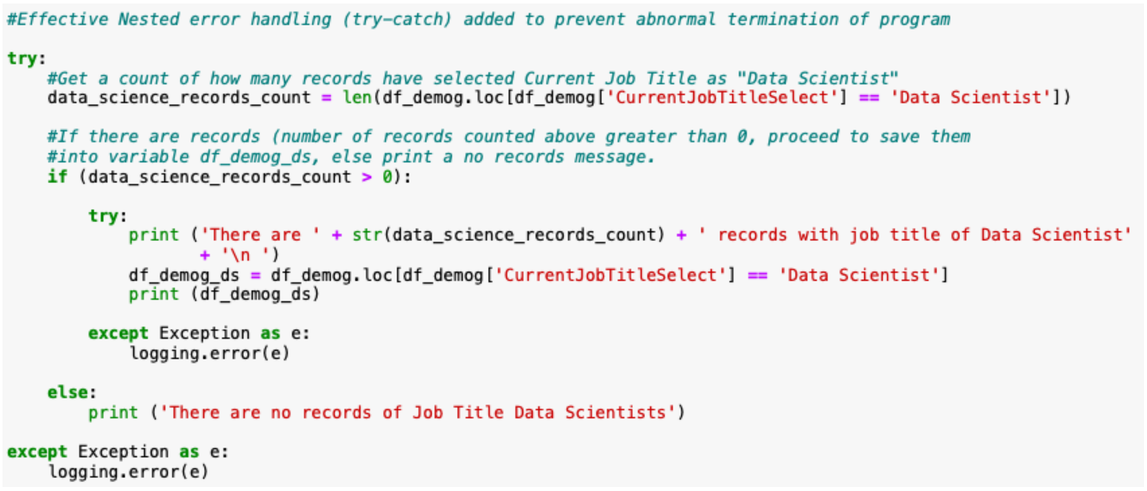
```
Display how many records with null values for each column.

GenderSelect                        0
Country                             0
Age                                 0
EmploymentStatus                    0
CodeWriter                          0
CurrentJobTitleSelect               0
TitleFit                           76
CurrentEmployerType                52
MLToolNextYearSelect              121
MLMethodNextYearSelect            157
LanguageRecommendationSelect       99
FormalEducation                     0
MajorSelect                       375
FirstTrainingSelect                 3
CompensationAmount                  0
CompensationCurrency                0
JobSatisfaction                    10
dtype: int64

Display how many records without null values for each column.

GenderSelect                      4327
Country                           4327
Age                               4327
EmploymentStatus                  4327
CodeWriter                        4327
CurrentJobTitleSelect             4327
TitleFit                          4251
CurrentEmployerType               4275
MLToolNextYearSelect              4206
MLMethodNextYearSelect            4170
LanguageRecommendationSelect      4228
FormalEducation                   4327
MajorSelect                       3952
FirstTrainingSelect               4324
CompensationAmount                4327
CompensationCurrency              4327
JobSatisfaction                   4317
dtype: int64
```

Figure 2: Output of 1.0A

Based on the output above, there are 9 columns (e.g. Country, Age, etc.) without any null values while the other 8 columns (e.g. CurrentEmployerType, MajorSelect, etc.) have null values. It can be concluded that MajorSelect was the survey question with the least number of responses (most number of NA values) while FirstTrainingSelect had the least number of null responses.

2.2. 1.0B: Data Scientist Users

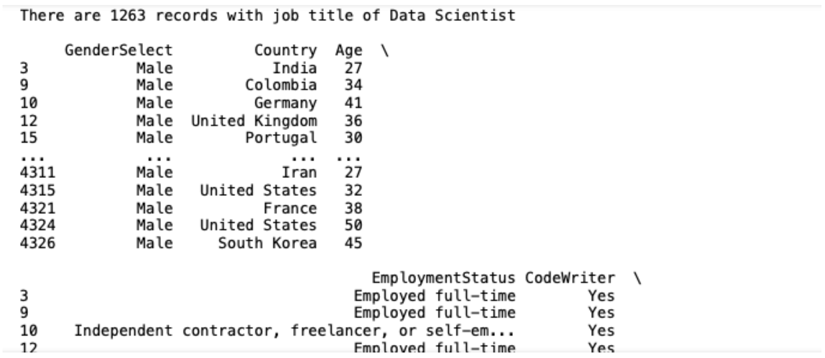
During this task, the data scientist was required to select users based on their job title as Data Scientists, store them in a variable df_demog_ds as well as count the number of data scientists.



```
#Effective Nested error handling (try-catch) added to prevent abnormal termination of program

try:
    #Get a count of how many records have selected Current Job Title as "Data Scientist"
    data_science_records_count = len(df_demog.loc[df_demog['CurrentJobTitleSelect'] == 'Data Scientist'])

    #If there are records (number of records counted above greater than 0, proceed to save them
    #into variable df_demog_ds, else print a no records message.
    if (data_science_records_count > 0):

        try:
            print ('There are ' + str(data_science_records_count) + ' records with job title of Data Scientist'
                + '\n ')
            df_demog_ds = df_demog.loc[df_demog['CurrentJobTitleSelect'] == 'Data Scientist']
            print (df_demog_ds)

        except Exception as e:
            logging.error(e)

    else:
        print ('There are no records of Job Title Data Scientists')

except Exception as e:
    logging.error(e)
```

Figure 3: Code Snippet for 1.0B

Based on the code snippet, the usage of effective error-handling (nested try-except block) prevents abnormal termination of the operation. The number of data scientists was calculated. Using a condition-based (if-else) statement to check if the data frame has any Data Scientists records (number of Data Scientists greater than 0), the data scientist records were saved into the variable df_demog_ds and df_demog_ds was displayed.



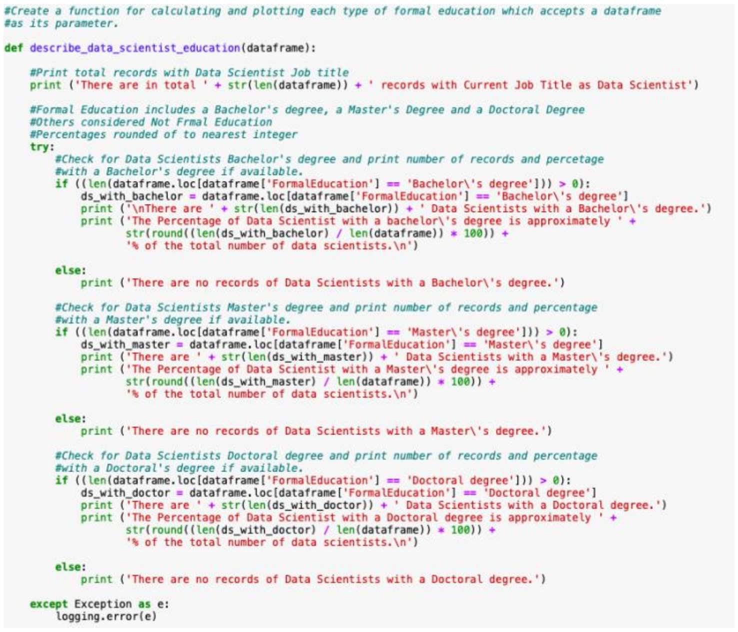
```
There are 1263 records with job title of Data Scientist

     GenderSelect         Country  Age  \
3            Male           India   27
9            Male        Colombia   34
10           Male         Germany   41
12           Male  United Kingdom   36
15           Male        Portugal   30
...           ...             ...  ...
4311         Male            Iran   27
4315         Male   United States   32
4321         Male          France   38
4324         Male   United States   50
4326         Male     South Korea   45

                                       EmploymentStatus CodeWriter  \
3                                    Employed full-time        Yes
9                                    Employed full-time        Yes
10    Independent contractor, freelancer, or self-em...        Yes
12                                   Employed full-time        Yes
```

Figure 4: Output of 1.0B

Based on the output above, it is found that 1263 records have a job title of Data Scientist out of 4327 records. Therefore, it can be concluded that 29.2% of the Kaggle users have a job title of Data Scientist. df_demog_ds is also displayed as shown partly in the above screenshot.

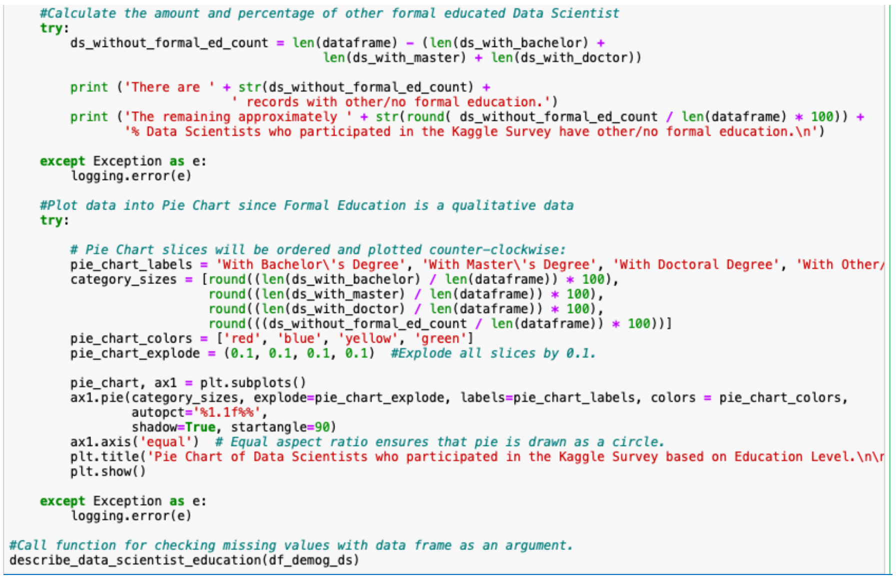
2.3. 1.1A: Data Scientist's Education

During this task, the data scientist was required to display the text output and visualisation of the number and percentage of data scientists. Pie Chart plot was selected as the most appropriate visualisation technique to plot a categorical variable based on its frequency.



```python
#Create a function for calculating and plotting each type of formal education which accepts a dataframe
#as its parameter.

def describe_data_scientist_education(dataframe):

    #Print total records with Data Scientist Job title
    print ('There are in total ' + str(len(dataframe)) + ' records with Current Job Title as Data Scientist')

    #Formal Education includes a Bachelor's degree, a Master's Degree and a Doctoral Degree
    #Others considered Not Frmal Education
    #Percentages rounded of to nearest integer
    try:
        #Check for Data Scientists Bachelor's degree and print number of records and percetage
        #with a Bachelor's degree if available.
        if ((len(dataframe.loc[dataframe['FormalEducation'] == 'Bachelor\'s degree'])) > 0):
            ds_with_bachelor = dataframe.loc[dataframe['FormalEducation'] == 'Bachelor\'s degree']
            print ('\nThere are ' + str(len(ds_with_bachelor)) + ' Data Scientists with a Bachelor\'s degree.')
            print ('The Percentage of Data Scientist with a bachelor\'s degree is approximately ' +
                  str(round((len(ds_with_bachelor) / len(dataframe)) * 100)) +
                  '% of the total number of data scientists.\n')

        else:
            print ('There are no records of Data Scientists with a Bachelor\'s degree.')

        #Check for Data Scientists Master's degree and print number of records and percentage
        #with a Master's degree if available.
        if ((len(dataframe.loc[dataframe['FormalEducation'] == 'Master\'s degree'])) > 0):
            ds_with_master = dataframe.loc[dataframe['FormalEducation'] == 'Master\'s degree']
            print ('There are ' + str(len(ds_with_master)) + ' Data Scientists with a Master\'s degree.')
            print ('The Percentage of Data Scientist with a Master\'s degree is approximately ' +
                  str(round((len(ds_with_master) / len(dataframe)) * 100)) +
                  '% of the total number of data scientists.\n')

        else:
            print ('There are no records of Data Scientists with a Master\'s degree.')

        #Check for Data Scientists Doctoral degree and print number of records and percentage
        #with a Doctoral's degree if available.
        if ((len(dataframe.loc[dataframe['FormalEducation'] == 'Doctoral degree'])) > 0):
            ds_with_doctor = dataframe.loc[dataframe['FormalEducation'] == 'Doctoral degree']
            print ('There are ' + str(len(ds_with_doctor)) + ' Data Scientists with a Doctoral degree.')
            print ('The Percentage of Data Scientist with a Doctoral degree is approximately ' +
                  str(round((len(ds_with_doctor) / len(dataframe)) * 100)) +
                  '% of the total number of data scientists.\n')

        else:
            print ('There are no records of Data Scientists with a Doctoral degree.')

    except Exception as e:
        logging.error(e)
```

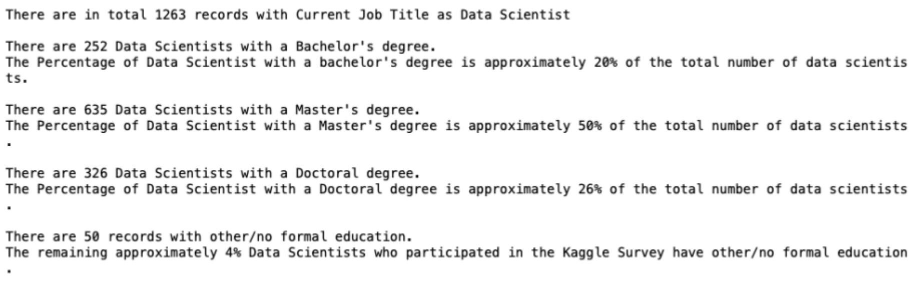
Figure 5: Code Snippet (1) for 1.1A



```
    #Calculate the amount and percentage of other formal educated Data Scientist
    try:
        ds_without_formal_ed_count = len(dataframe) - (len(ds_with_bachelor) +
                                     len(ds_with_master) + len(ds_with_doctor))

        print ('There are ' + str(ds_without_formal_ed_count) +
                           ' records with other/no formal education.')
        print ('The remaining approximately ' + str(round( ds_without_formal_ed_count / len(dataframe) * 100)) +
               '% Data Scientists who participated in the Kaggle Survey have other/no formal education.\n')

    except Exception as e:
        logging.error(e)

    #Plot data into Pie Chart since Formal Education is a qualitative data
    try:

        # Pie Chart slices will be ordered and plotted counter-clockwise:
        pie_chart_labels = 'With Bachelor\'s Degree', 'With Master\'s Degree', 'With Doctoral Degree', 'With Other/
        category_sizes = [round((len(ds_with_bachelor) / len(dataframe)) * 100),
                          round((len(ds_with_master) / len(dataframe)) * 100),
                          round((len(ds_with_doctor) / len(dataframe)) * 100),
                          round(((ds_without_formal_ed_count / len(dataframe)) * 100))]
        pie_chart_colors = ['red', 'blue', 'yellow', 'green']
        pie_chart_explode = (0.1, 0.1, 0.1, 0.1)  #Explode all slices by 0.1.

        pie_chart, ax1 = plt.subplots()
        ax1.pie(category_sizes, explode=pie_chart_explode, labels=pie_chart_labels, colors = pie_chart_colors,
                autopct='%1.1f%%',
                shadow=True, startangle=90)
        ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
        plt.title('Pie Chart of Data Scientists who participated in the Kaggle Survey based on Education Level.\n\r
        plt.show()

    except Exception as e:
        logging.error(e)

#Call function for checking missing values with data frame as an argument.
describe_data_scientist_education(df_demog_ds)
```

Figure 6: Code Snippet (2) for 1.1A

Based on the above code snippets, the data scientist created a function which accepts a data frame as parameter to calculate the number and percentage of data scientists based on education and the plot. The usage of effective error-handling (try-except block) prevents abnormal termination of the operation. Using a condition-based (if-else) statement to check if the data frame has any Data Scientists with bachelor's degree records (number of Data Scientists greater than 0), the number and percentage (rounded to nearest integer) of Data Scientists with a bachelor's degree was displayed. Similarly, the same procedure was carried out for master's and doctoral degrees. Furthermore, other education types (including diploma, etc.) were all classified under 'Other/No Formal Education' and the count and percentage was also displayed. Lastly, a pie-chart displaying the percentage frequency of each education type was plotted to aid visualisation.



```
There are in total 1263 records with Current Job Title as Data Scientist

There are 252 Data Scientists with a Bachelor's degree.
The Percentage of Data Scientist with a bachelor's degree is approximately 20% of the total number of data scientis
ts.

There are 635 Data Scientists with a Master's degree.
The Percentage of Data Scientist with a Master's degree is approximately 50% of the total number of data scientists
.

There are 326 Data Scientists with a Doctoral degree.
The Percentage of Data Scientist with a Doctoral degree is approximately 26% of the total number of data scientists
.

There are 50 records with other/no formal education.
The remaining approximately 4% Data Scientists who participated in the Kaggle Survey have other/no formal education
.
```

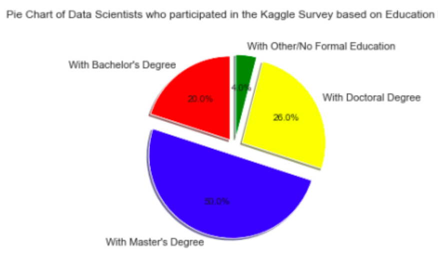
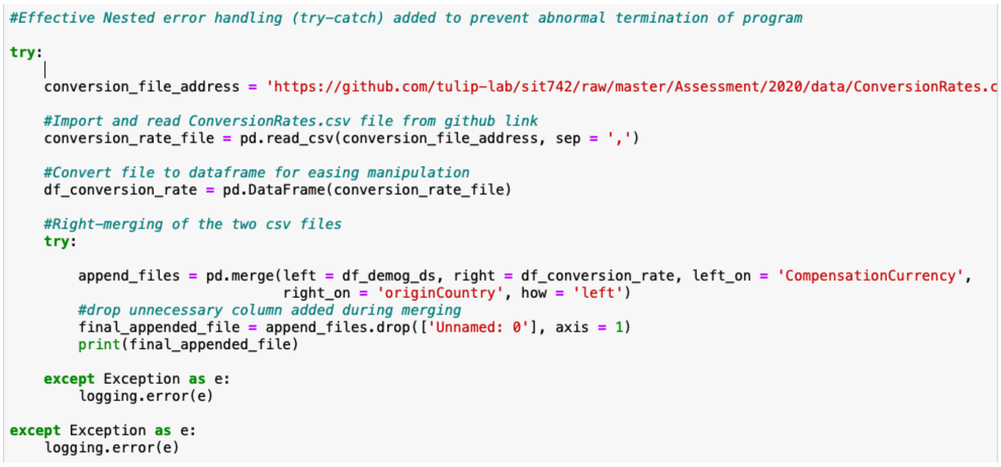


Figure 7: Output of 1.1A

Based on the above output, it can be seen that most of Kaggle's Data Scientists (50%) have a master's degree while 26% of them have a doctoral degree. A very small percentage of the data scientists had other or no formal education. With Master's degree and Doctoral degree amounting to 70%, it can be concluded that the job market for data scientists requires pretty high educational requirements.

2.4. 1.2A: Data Scientist's Salary

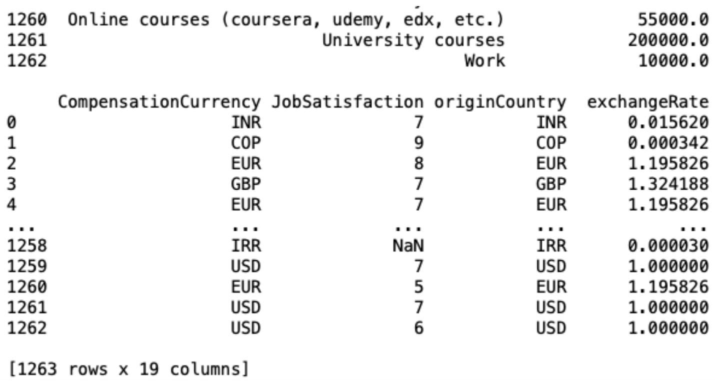
During this task, the data scientist was required to merge two datasets using country codes, MCQResponses.csv and ConversionRates.csv, to convert the salary into Australian Dollars and give the maximum and median salary in AUD.



```
#Effective Nested error handling (try-catch) added to prevent abnormal termination of program

try:
    |
    conversion_file_address = 'https://github.com/tulip-lab/sit742/raw/master/Assessment/2020/data/ConversionRates.c

    #Import and read ConversionRates.csv file from github link
    conversion_rate_file = pd.read_csv(conversion_file_address, sep = ',')

    #Convert file to dataframe for easing manipulation
    df_conversion_rate = pd.DataFrame(conversion_rate_file)

    #Right-merging of the two csv files
    try:

        append_files = pd.merge(left = df_demog_ds, right = df_conversion_rate, left_on = 'CompensationCurrency',
                                right_on = 'originCountry', how = 'left')
        #drop unnecessary column added during merging
        final_appended_file = append_files.drop(['Unnamed: 0'], axis = 1)
        print(final_appended_file)

    except Exception as e:
        logging.error(e)

except Exception as e:
    logging.error(e)
```

Figure 8: Code Snippet (1) for 1.2A

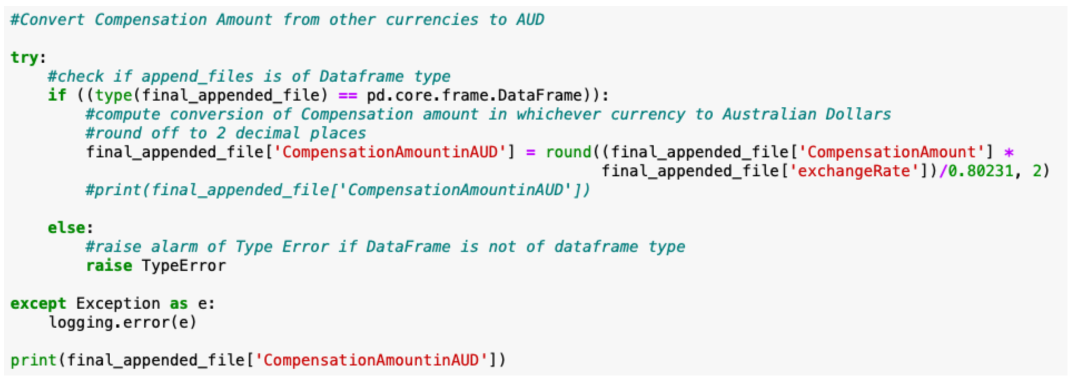
Based on the above code snippet, the usage of effective error-handling (nested try-except block) prevents abnormal termination of the operation. The ConversionRates.csv was uploaded read and converted into a Pandas data frame. The two datasets were then right merged using country codes in both datasets and the unnamed anomaly column created was discarded. The merged dataset was then displayed.



```
1260  Online courses (coursera, udemy, edx, etc.)            55000.0
1261                           University courses           200000.0
1262                                         Work            10000.0

     CompensationCurrency JobSatisfaction originCountry  exchangeRate
0                     INR               7           INR      0.015620
1                     COP               9           COP      0.000342
2                     EUR               8           EUR      1.195826
3                     GBP               7           GBP      1.324188
4                     EUR               7           EUR      1.195826
...                   ...             ...           ...           ...
1258                  IRR             NaN           IRR      0.000030
1259                  USD               7           USD      1.000000
1260                  EUR               5           EUR      1.195826
1261                  USD               7           USD      1.000000
1262                  USD               6           USD      1.000000

[1263 rows x 19 columns]
```

Figure 8: Output (1) for 1.2A

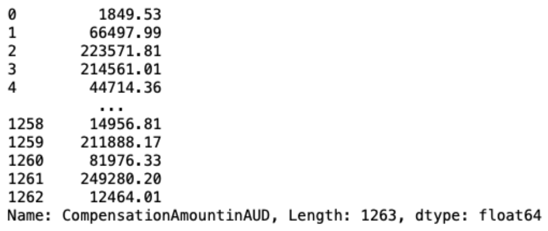
The output above shows the merged dataset columns.



```
#Convert Compensation Amount from other currencies to AUD

try:
    #check if append_files is of Dataframe type
    if ((type(final_appended_file) == pd.core.frame.DataFrame)):
        #compute conversion of Compensation amount in whichever currency to Australian Dollars
        #round off to 2 decimal places
        final_appended_file['CompensationAmountinAUD'] = round((final_appended_file['CompensationAmount'] *
                                                               final_appended_file['exchangeRate'])/0.80231, 2)
        #print(final_appended_file['CompensationAmountinAUD'])

    else:
        #raise alarm of Type Error if DataFrame is not of dataframe type
        raise TypeError

except Exception as e:
    logging.error(e)

print(final_appended_file['CompensationAmountinAUD'])
```

Figure 9: Code Snippet (2) for 1.2A

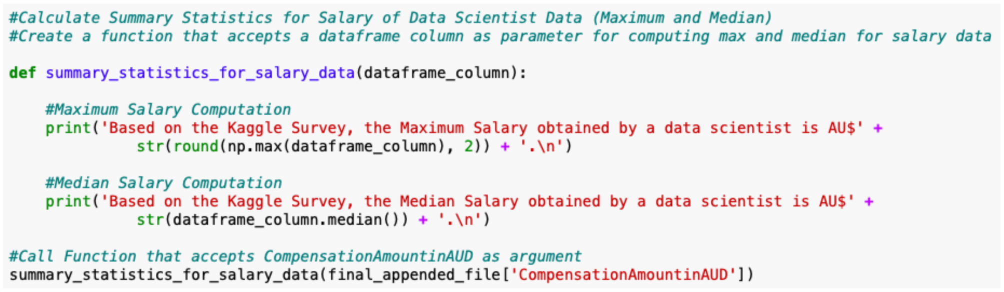
Based on the above code snippet, the usage of effective error-handling (try-except block) prevents abnormal termination of the operation. Using a condition-based (if-else) statement to check if the merged dataset is of Data Frame type, a new column 'CompensationAmountinAUD' was created to store the converted salary [(salary*exchange rate) * AUD rate] of each data scientist (rounded to 2 decimal places after AUD conversion) and the column 'CompensationAmountinAUD' was displayed.



```
0          1849.53
1         66497.99
2        223571.81
3        214561.01
4         44714.36
           ...
1258      14956.81
1259     211888.17
1260      81976.33
1261     249280.20
1262      12464.01
Name: CompensationAmountinAUD, Length: 1263, dtype: float64
```

Figure 10: Output (2) for 1.2A

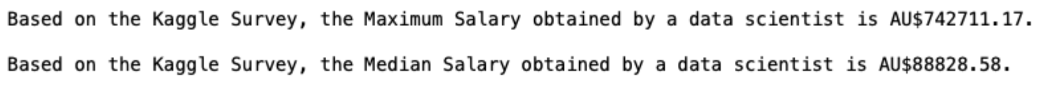
The output above shows the salary of all data scientists converted into Australian Dollars.



```
#Calculate Summary Statistics for Salary of Data Scientist Data (Maximum and Median)
#Create a function that accepts a dataframe column as parameter for computing max and median for salary data

def summary_statistics_for_salary_data(dataframe_column):

    #Maximum Salary Computation
    print('Based on the Kaggle Survey, the Maximum Salary obtained by a data scientist is AU$' +
          str(round(np.max(dataframe_column), 2)) + '.\n')

    #Median Salary Computation
    print('Based on the Kaggle Survey, the Median Salary obtained by a data scientist is AU$' +
          str(dataframe_column.median()) + '.\n')

#Call Function that accepts CompensationAmountinAUD as argument
summary_statistics_for_salary_data(final_appended_file['CompensationAmountinAUD'])
```

Figure 11: Code Snippet (3) for 1.2A

Based on the above code snippet, a function that accepts a data frame column as parameter was defined to calculate and display the maximum and the median salary of data scientists in Australian dollars and the function was called to compute so.



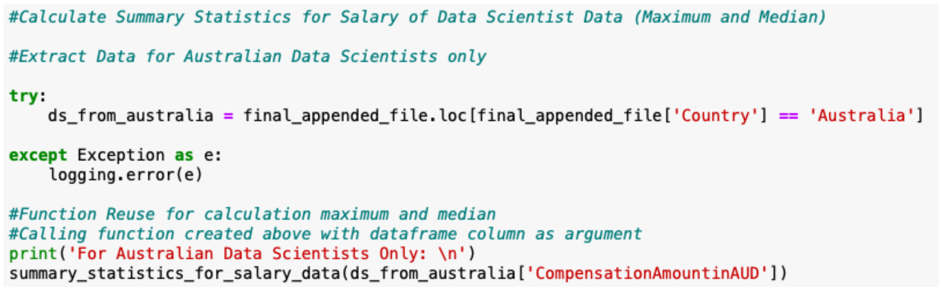
```
Based on the Kaggle Survey, the Maximum Salary obtained by a data scientist is AU$742711.17.

Based on the Kaggle Survey, the Median Salary obtained by a data scientist is AU$88828.58.
```

Figure 12: Output (3) for 1.2A

Based on the output, it can be concluded that the maximum and median salary around the globe is AU$74277.17 and AU$88828.58 respectively.

2.5. 1.2B: Australian Data Scientist's Salary

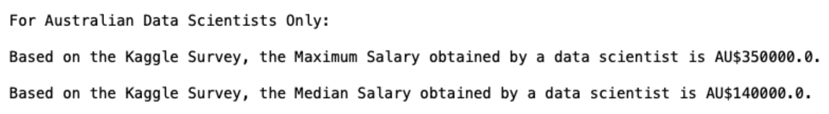
During this task, the data scientist was required to calculate the maximum and median salaries of Australian Data Scientists as well as plot a box and whisker plot for the salary of Australian Data Scientists.



```
#Calculate Summary Statistics for Salary of Data Scientist Data (Maximum and Median)

#Extract Data for Australian Data Scientists only

try:
    ds_from_australia = final_appended_file.loc[final_appended_file['Country'] == 'Australia']

except Exception as e:
    logging.error(e)

#Function Reuse for calculation maximum and median
#Calling function created above with dataframe column as argument
print('For Australian Data Scientists Only: \n')
summary_statistics_for_salary_data(ds_from_australia['CompensationAmountinAUD'])
```

Figure 13: Code Snippet (1) for 1.2B

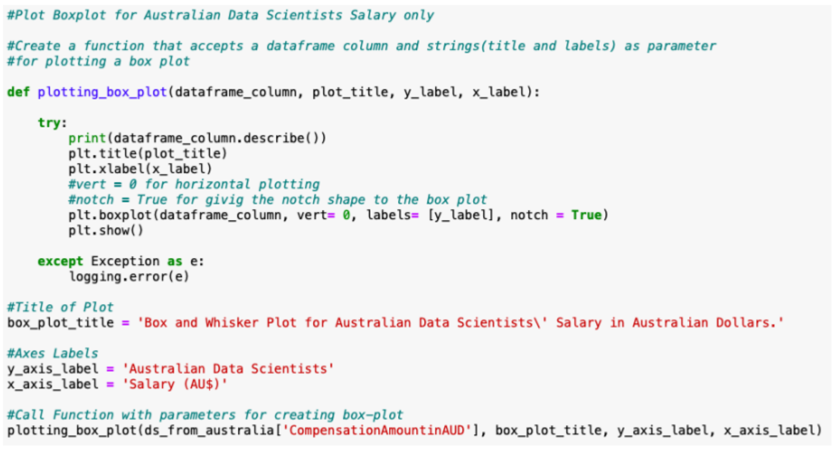
Based on the above code snippet, the usage of effective error-handling (try-except block) prevents abnormal termination of the operation. The information for Australian data scientists only was extracted and saved into variable ds_from_australia. The function defined in 1.2A was reused to compute the maximum and median salary of Australian Data Scientists in Australian Dollars.



```
For Australian Data Scientists Only:

Based on the Kaggle Survey, the Maximum Salary obtained by a data scientist is AU$350000.0.

Based on the Kaggle Survey, the Median Salary obtained by a data scientist is AU$140000.0.
```

Figure 14: Output (1) for 1.2B

Based on the output above, it can be concluded that the maximum and median salary in Australia is AU$350,000.00 and AU$140000.00 respectively.



```
#Plot Boxplot for Australian Data Scientists Salary only

#Create a function that accepts a dataframe column and strings(title and labels) as parameter
#for plotting a box plot

def plotting_box_plot(dataframe_column, plot_title, y_label, x_label):

    try:
        print(dataframe_column.describe())
        plt.title(plot_title)
        plt.xlabel(x_label)
        #vert = 0 for horizontal plotting
        #notch = True for givig the notch shape to the box plot
        plt.boxplot(dataframe_column, vert= 0, labels= [y_label], notch = True)
        plt.show()

    except Exception as e:
        logging.error(e)

#Title of Plot
box_plot_title = 'Box and Whisker Plot for Australian Data Scientists\' Salary in Australian Dollars.'

#Axes Labels
y_axis_label = 'Australian Data Scientists'
x_axis_label = 'Salary (AU$)'

#Call Function with parameters for creating box-plot
plotting_box_plot(ds_from_australia['CompensationAmountinAUD'], box_plot_title, y_axis_label, x_axis_label)
```

Figure 15: Code Snippet (2) for 1.2B

Based on the code snippet, a function that accepts several parameters for computing the summary measures and creating a single box and whisker plot was defined. The usage of effective error-handling (try-except block) prevents abnormal termination of the operation. The box plot title and axes titles were then stored in a variable to be parsed as arguments when calling the function.

```
count        29.000000
mean     149341.250690
std       83700.147257
min          90.000000
25%      120000.000000
50%      140000.000000
75%      180000.000000
max      350000.000000
Name: CompensationAmountinAUD, dtype: float64
```

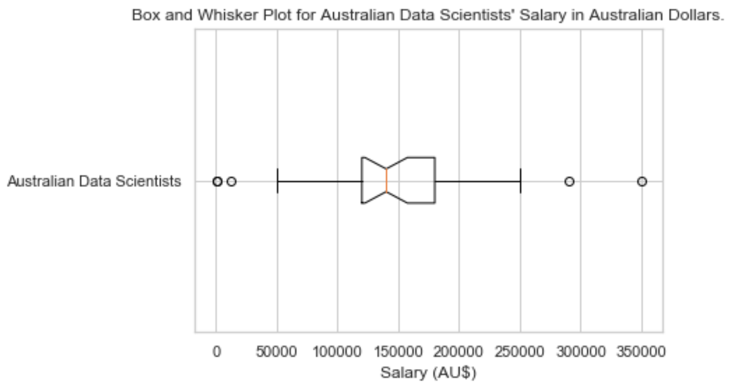
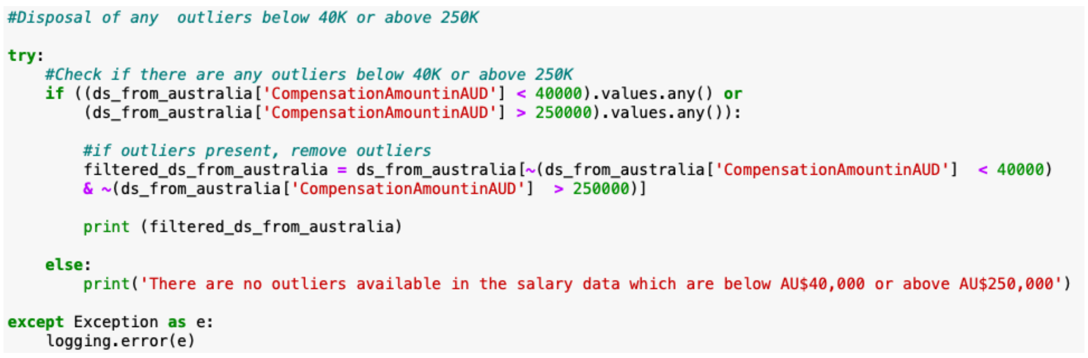


Figure 16: Output (2) for 1.2B

Based on the above output, it can be seen that there are some outliers in the 29 Australian Data Scientist Salary Data. Furthermore, based on the Kaggle survey, 75% of data scientists in Australia received lower than AU$180,000 and the middle 50% received a salary ranging from AU$120,000 to AU$180,000. The minimum salary of a data scientist in Australia starts at $90,000. Due to its resistance to outliers, median salary is better than mean salary to quantify the average salary of data scientists.

2.6. 1.2C: Filtered Australian Data Scientist's Salary

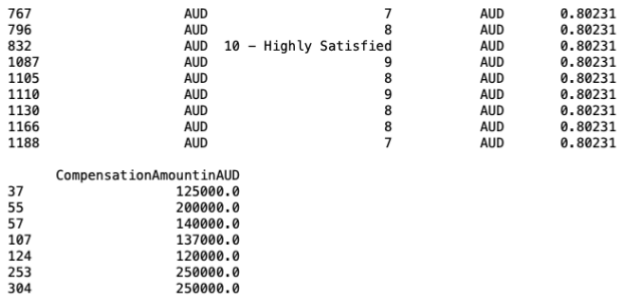
During this task, the data scientist was required to filter and remove any salary data outliers below AU$40,000 and above AU$250,00, then calculate the new maximum and median salaries of Australian Data Scientists as well as plot a box and whisker plot for the new salary of Australian Data Scientists.



```
#Disposal of any  outliers below 40K or above 250K

try:
    #Check if there are any outliers below 40K or above 250K
    if ((ds_from_australia['CompensationAmountinAUD'] < 40000).values.any() or
        (ds_from_australia['CompensationAmountinAUD'] > 250000).values.any()):

        #if outliers present, remove outliers
        filtered_ds_from_australia = ds_from_australia[~(ds_from_australia['CompensationAmountinAUD']  < 40000)
        & ~(ds_from_australia['CompensationAmountinAUD']  > 250000)]

        print (filtered_ds_from_australia)

    else:
        print('There are no outliers available in the salary data which are below AU$40,000 or above AU$250,000')

except Exception as e:
    logging.error(e)
```

Figure 17: Code Snippet (1) for 1.2C

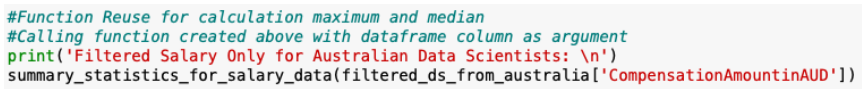
Based on the above code snippet, the usage of effective error-handling (try-except block) prevents abnormal termination of the operation. Using a condition-based (if-else) statement to check if there are any outliers below AU$40,000 and AU$250,000, the outliers were discarded, and the new outlier-free salary data was stored in variable filtered_ds_from_australia and then displayed.



```
767                    AUD                   7          AUD      0.80231
796                    AUD                   8          AUD      0.80231
832                    AUD  10 - Highly Satisfied       AUD      0.80231
1087                   AUD                   9          AUD      0.80231
1105                   AUD                   8          AUD      0.80231
1110                   AUD                   9          AUD      0.80231
1130                   AUD                   8          AUD      0.80231
1166                   AUD                   8          AUD      0.80231
1188                   AUD                   7          AUD      0.80231

      CompensationAmountinAUD
37                   125000.0
55                   200000.0
57                   140000.0
107                  137000.0
124                  120000.0
253                  250000.0
304                  250000.0
```

Figure 18: Output (1) for 1.2C

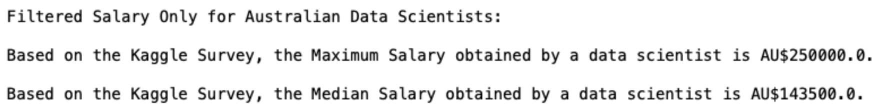
The output above shows some information of Australian Data Scientists without salary outliers.



```
#Function Reuse for calculation maximum and median
#Calling function created above with dataframe column as argument
print('Filtered Salary Only for Australian Data Scientists: \n')
summary_statistics_for_salary_data(filtered_ds_from_australia['CompensationAmountinAUD'])
```

Figure 19: Code Snippet (2) for 1.2C

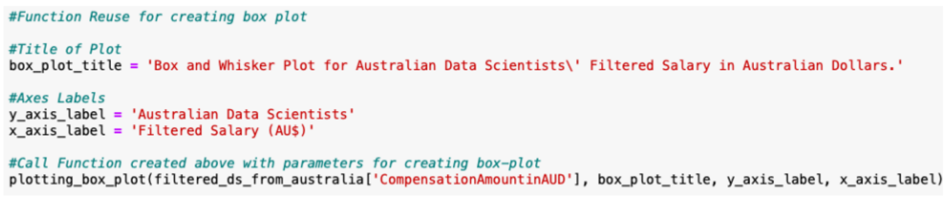
Based on the above code snippet, the function defined in 1.2A was reused to compute the new maximum and median salary without outliers of Australian Data Scientists in Australian Dollars.



```
Filtered Salary Only for Australian Data Scientists:

Based on the Kaggle Survey, the Maximum Salary obtained by a data scientist is AU$250000.0.

Based on the Kaggle Survey, the Median Salary obtained by a data scientist is AU$143500.0.
```

Figure 20: Output (2) for 1.2C

Based on the output above, it can be concluded that the new maximum and median salary without outliers in Australia is AU$250,000.00 and AU$143500.00 respectively.



```
#Function Reuse for creating box plot

#Title of Plot
box_plot_title = 'Box and Whisker Plot for Australian Data Scientists\' Filtered Salary in Australian Dollars.'

#Axes Labels
y_axis_label = 'Australian Data Scientists'
x_axis_label = 'Filtered Salary (AU$)'

#Call Function created above with parameters for creating box-plot
plotting_box_plot(filtered_ds_from_australia['CompensationAmountinAUD'], box_plot_title, y_axis_label, x_axis_label)
```

Figure 21: Code Snippet (3) for 1.2C

Based on the above code snippet, the function defined in 1.2B was reused to plot the new box and whisker plot of salary without outliers of Australian Data Scientists in Australian Dollars. The box plot title and axes titles were then stored in a variable to be parsed as arguments when calling the function.

```
count        24.000000
mean     153291.666667
std       56045.384689
min       50000.000000
25%      123750.000000
50%      143500.000000
75%      172500.000000
max      250000.000000
Name: CompensationAmountinAUD, dtype: float64
```

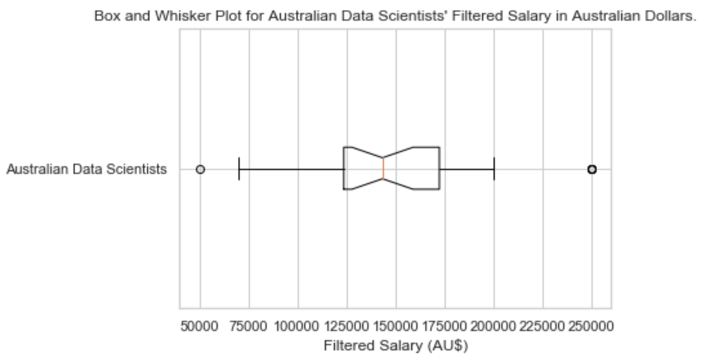
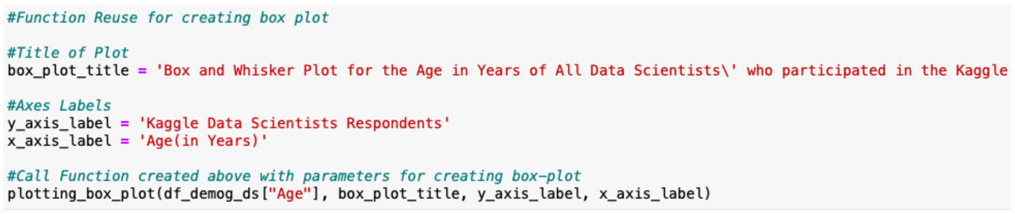


Figure 22: Output (3) for 1.2C

Based on the above output, it can be concluded that there are some new outliers in the 24 filtered Australian Data Scientist Salary Data. Furthermore, based on the new filtered salary data, 75% of data scientists in Australia received lower than AU$172,500 and the middle 50% received a salary ranging from AU$123,750 to AU$172,500. The new minimum filtered salary of a data scientist in Australia starts at $50,000.

2.7. 1.3A: Data Scientist's Age

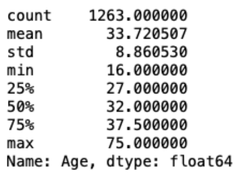
During this task, the data scientist was required to plot a box and whisker plot for age, calculate a five-number summary of the age of data scientists, calculate the number of data scientists of working age as well as find the number of respondents below 18 years old.



```
#Function Reuse for creating box plot

#Title of Plot
box_plot_title = 'Box and Whisker Plot for the Age in Years of All Data Scientists\' who participated in the Kaggle

#Axes Labels
y_axis_label = 'Kaggle Data Scientists Respondents'
x_axis_label = 'Age(in Years)'

#Call Function created above with parameters for creating box-plot
plotting_box_plot(df_demog_ds["Age"], box_plot_title, y_axis_label, x_axis_label)
```

Figure 23: Code Snippet (1) for 1.3A

Based on the above code snippet, the function defined in 1.2B was reused to plot the box and whisker plot of age of all Kaggle Survey Data Scientists. The box plot title and axes titles were then stored in a variable to be parsed as arguments when calling the function.



```
count    1263.000000
mean       33.720507
std         8.860530
min        16.000000
25%        27.000000
50%        32.000000
75%        37.500000
max        75.000000
Name: Age, dtype: float64
```

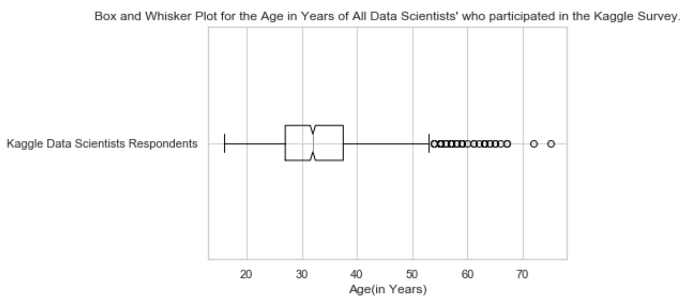
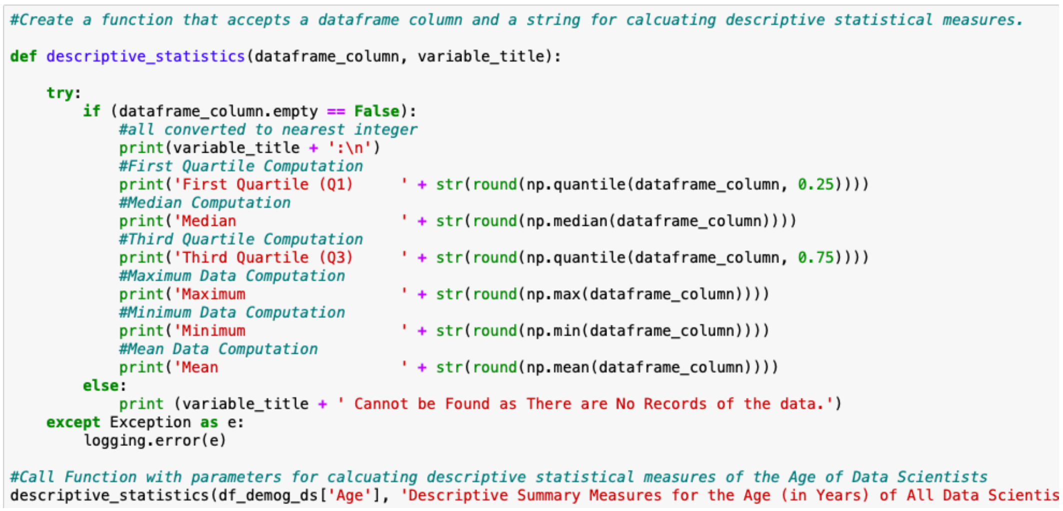


Figure 24: Output (1) for 1.3A

Based on the above output, it can be concluded that there are several outliers in the 1263 Data Scientist age. Furthermore, based on the Kaggle Survey data, approximately 75% of data scientists are aged below 38 years and the middle 50% are aged between 27 years and 38 years. The maximum and minimum age of data scientists who participated in the Kaggle Survey are 75 years old and 16 years old respectively.



```
#Create a function that accepts a dataframe column and a string for calcuating descriptive statistical measures.

def descriptive_statistics(dataframe_column, variable_title):

    try:
        if (dataframe_column.empty == False):
            #all converted to nearest integer
            print(variable_title + ':\n')
            #First Quartile Computation
            print('First Quartile (Q1)      ' + str(round(np.quantile(dataframe_column, 0.25))))
            #Median Computation
            print('Median                   ' + str(round(np.median(dataframe_column))))
            #Third Quartile Computation
            print('Third Quartile (Q3)      ' + str(round(np.quantile(dataframe_column, 0.75))))
            #Maximum Data Computation
            print('Maximum                  ' + str(round(np.max(dataframe_column))))
            #Minimum Data Computation
            print('Minimum                  ' + str(round(np.min(dataframe_column))))
            #Mean Data Computation
            print('Mean                     ' + str(round(np.mean(dataframe_column))))
        else:
            print (variable_title + ' Cannot be Found as There are No Records of the data.')
    except Exception as e:
        logging.error(e)

#Call Function with parameters for calcuating descriptive statistical measures of the Age of Data Scientists
descriptive_statistics(df_demog_ds['Age'], 'Descriptive Summary Measures for the Age (in Years) of All Data Scientis
```

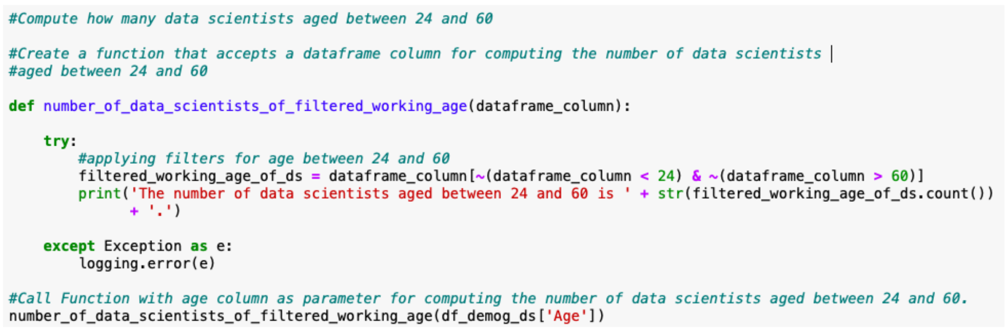
Figure 25: Code Snippet (2) for 1.3A

Based on the code snippet, a function that accepts two parameters is defined to compute the summary statistics (lower quartile, median, upper quartile, maximum, minimum and mean) of any numerical data using NumPy. Using a condition-based (if-else statement) to check if the parameter entered is not an empty list, column, series, etc., the summary statistics are computed and printed when the function is called.

```
Descriptive Summary Measures for the Age (in Years) of All Data Scientists:

First Quartile (Q1)     27.0
Median                  32.0
Third Quartile (Q3)     38.0
Maximum                 75
Minimum                 16
Mean                    34
```

Figure 26: Output (2) for 1.3A

Based on the above output, it can be concluded that approximately 75% of data scientists are aged below 38 years and the middle 50% are aged between 27 years and 38 years. The maximum and minimum age of data scientists who participated in the Kaggle Survey are 75 years old and 16 years old respectively. Furthermore, the mean and median age of Data Scientists who participated in the Kaggle Survey are 34 years and 32 years respectively. Due to its resistance to outliers, median age is better than mean age to quantify the average age of data scientists.



```
#Compute how many data scientists aged between 24 and 60

#Create a function that accepts a dataframe column for computing the number of data scientists
#aged between 24 and 60

def number_of_data_scientists_of_filtered_working_age(dataframe_column):

    try:
        #applying filters for age between 24 and 60
        filtered_working_age_of_ds = dataframe_column[~(dataframe_column < 24) & ~(dataframe_column > 60)]
        print('The number of data scientists aged between 24 and 60 is ' + str(filtered_working_age_of_ds.count())
              + '.')

    except Exception as e:
        logging.error(e)

#Call Function with age column as parameter for computing the number of data scientists aged between 24 and 60.
number_of_data_scientists_of_filtered_working_age(df_demog_ds['Age'])
```

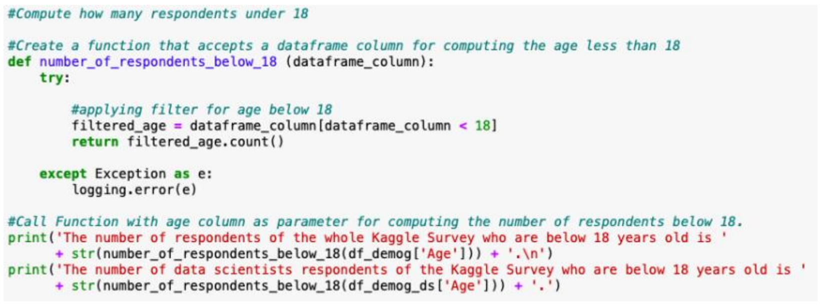
Figure 27: Code Snippet (3) for 1.3A

Based on the above code snippet, a function which accepts age values as parameter was defined to compute the number of data scientists of working age. the usage of effective error-handling (try-except block) prevents abnormal termination of the operation. The usage of effective error-handling (try-except block) prevents abnormal termination of the operation. The filter is applied to the age data and the number of data scientists of working age is printed.

```
The number of data scientists aged between 24 and 60 is 1188.
```

Figure 28: Output (3) for 1.3A

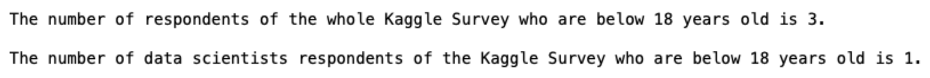
Based on the output, 1188 data scientists are of working age out of 1263 data scientists who participated in the Kaggle Survey. Therefore, it can be concluded that the majority of data scientists (94.1%) are of working age between 24 years and 60 years.



```
#Compute how many respondents under 18

#Create a function that accepts a dataframe column for computing the age less than 18
def number_of_respondents_below_18 (dataframe_column):
    try:

        #applying filter for age below 18
        filtered_age = dataframe_column[dataframe_column < 18]
        return filtered_age.count()

    except Exception as e:
        logging.error(e)

#Call Function with age column as parameter for computing the number of respondents below 18.
print('The number of respondents of the whole Kaggle Survey who are below 18 years old is '
      + str(number_of_respondents_below_18(df_demog['Age'])) + '.\n')
print('The number of data scientists respondents of the Kaggle Survey who are below 18 years old is '
      + str(number_of_respondents_below_18(df_demog_ds['Age'])) + '.')
```

Figure 29: Code Snippet (4) for 1.3A

Based on the above code snippet, a function accepting values of age as argument was defined to compute the number of values below 18. After calling the functions twice, the number of respondents in the Kaggle Survey with age less than 18 and the number of data scientists with age below 18 is computed and displayed.



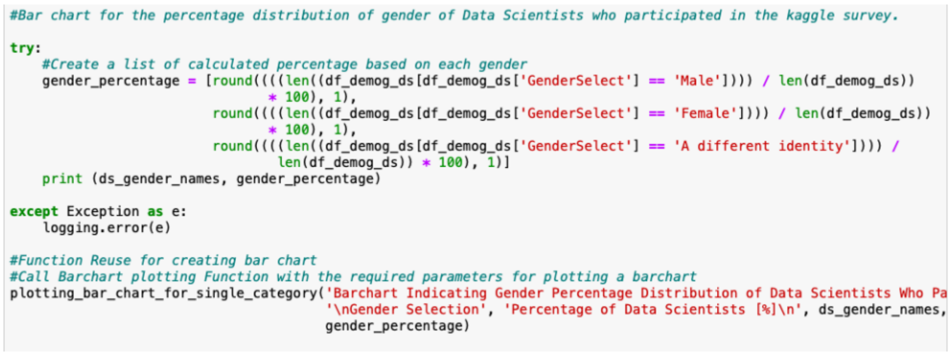
```
The number of respondents of the whole Kaggle Survey who are below 18 years old is 3.

The number of data scientists respondents of the Kaggle Survey who are below 18 years old is 1.
```

Figure 30: Output (4) for 1.3A

Based on the output above, 3 respondents of the whole Kaggle Survey are aged below 18 while the number of data scientists aged below 18 is 1. Therefore, it can be concluded that there are very few respondents below the working age.

2.8. 1.3B: Data Scientist's Gender

During this task, the data scientist is required to plot a bar chart to show the percentage distribution of males, females and other genders.



```
#Bar chart for the percentage distribution of gender of Data Scientists who participated in the kaggle survey.

try:
    #Create a list of calculated percentage based on each gender
    gender_percentage = [round((((len((df_demog_ds[df_demog_ds['GenderSelect'] == 'Male']))) / len(df_demog_ds))
                               * 100), 1),
                         round((((len((df_demog_ds[df_demog_ds['GenderSelect'] == 'Female']))) / len(df_demog_ds))
                               * 100), 1),
                         round((((len((df_demog_ds[df_demog_ds['GenderSelect'] == 'A different identity']))) /
                                len(df_demog_ds)) * 100), 1)]
    print (ds_gender_names, gender_percentage)

except Exception as e:
    logging.error(e)

#Function Reuse for creating bar chart
#Call Barchart plotting Function with the required parameters for plotting a barchart
plotting_bar_chart_for_single_category('Barchart Indicating Gender Percentage Distribution of Data Scientists Who Pa
                                       '\nGender Selection', 'Percentage of Data Scientists [%]\n', ds_gender_names,
                                       gender_percentage)
```

Figure 31: Code Snippet for 1.3B

Based on the above code snippet, the usage of effective error-handling (try-except block) prevents abnormal termination of the operation. A list consisting of the percentage of each gender is created. The variable ds_gender_names which is already created and filled before is reused. The percentage gender and related category is then printed. Furthermore, the function for plotting a single bar chart that has been previously defined in 1.3.2 (please refer to script) is re-utilised by parsing all the required arguments.

```
['Male', 'Female', 'Other identities'] [86.2, 13.3, 0.5]
```

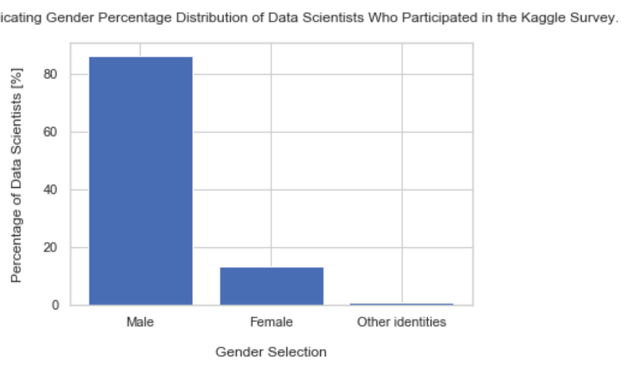
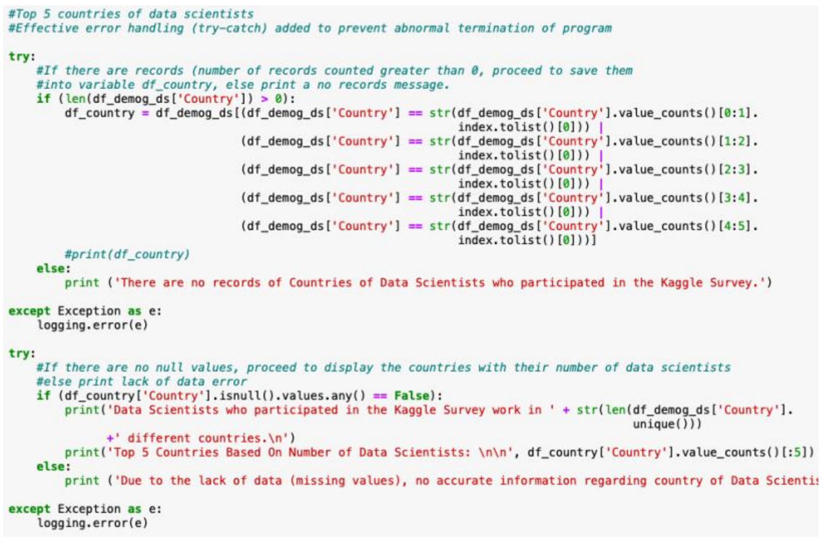


Figure 32: Output for 1.3B

Based on the above output, it can be seen that male data scientists are much higher in proportion (86.2%) as compared to other genders in the Kaggle Survey. Inferentially, it can be concluded that the data science job market consists of more males as compared to other genders. Other identities (e.g. LGBTQ+) are seen as the least in proportion of the whole data science market. Female data scientists are approximately 6 times less likely than male data scientists.

2.9. 1.3C: Data Scientist's Top 5 Countries

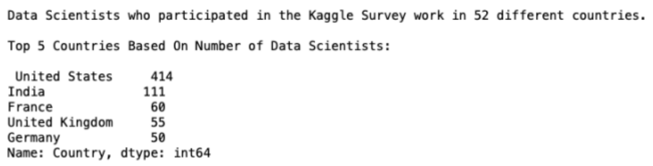
During this task, the data scientist was required to find the top five countries with the greatest number of data scientists and their corresponding frequency as well as visualise the data using the most appropriate technique.



```
#Top 5 countries of data scientists
#Effective error handling (try-catch) added to prevent abnormal termination of program

try:
    #If there are records (number of records counted greater than 0, proceed to save them
    #into variable df_country, else print a no records message.
    if (len(df_demog_ds['Country']) > 0):
        df_country = df_demog_ds[(df_demog_ds['Country'] == str(df_demog_ds['Country'].value_counts()[0:1].
                                                                 index.tolist()[0])) |
                                 (df_demog_ds['Country'] == str(df_demog_ds['Country'].value_counts()[1:2].
                                                                 index.tolist()[0])) |
                                 (df_demog_ds['Country'] == str(df_demog_ds['Country'].value_counts()[2:3].
                                                                 index.tolist()[0])) |
                                 (df_demog_ds['Country'] == str(df_demog_ds['Country'].value_counts()[3:4].
                                                                 index.tolist()[0])) |
                                 (df_demog_ds['Country'] == str(df_demog_ds['Country'].value_counts()[4:5].
                                                                 index.tolist()[0]))]

        #print(df_country)
    else:
        print ('There are no records of Countries of Data Scientists who participated in the Kaggle Survey.')

except Exception as e:
    logging.error(e)

try:
    #If there are no null values, proceed to display the countries with their number of data scientists
    #else print lack of data error
    if (df_country['Country'].isnull().values.any() == False):
        print('Data Scientists who participated in the Kaggle Survey work in ' + str(len(df_demog_ds['Country'].
                                                                                         unique()))

              +' different countries.\n')
        print('Top 5 Countries Based On Number of Data Scientists: \n\n', df_country['Country'].value_counts()[:5])
    else:
        print ('Due to the lack of data (missing values), no accurate information regarding country of Data Scientis

except Exception as e:
    logging.error(e)
```

Figure 33: Code Snippet (1) for 1.3C

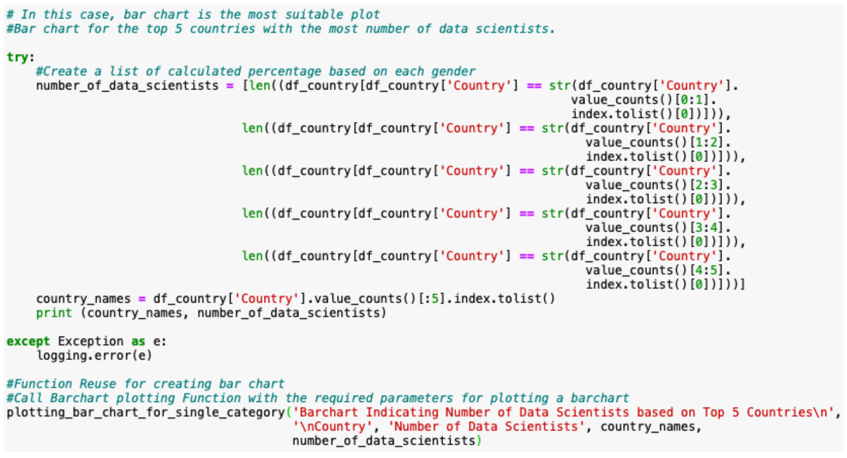
Based on the code snippet, the usage of effective error-handling (try-except block) prevents abnormal termination of the operation. Using a condition-based (if-else) statement to check if there are any records of country, the information of the top 5 countries is saved in variable df_country. Furthermore, if there are no null values in the 'Country' column of the top 5 countries for data scientists (df_column), the number of countries from which data scientists participated will be printed followed by the top five countries of data scientists and their frequency.



```
Data Scientists who participated in the Kaggle Survey work in 52 different countries.

Top 5 Countries Based On Number of Data Scientists:

 United States     414
India             111
France             60
United Kingdom     55
Germany            50
Name: Country, dtype: int64
```

Figure 34: Output (1) for 1.3C

Based on the above output, it can be seen that US has the highest number of data scientists based on the Kaggle Survey. Therefore, it can be concluded that US has the highest percentage of data scientists (32.8%), followed by India (8.8%), France (4.8%), UK (4.4%) and Germany (4.0%). From the proportion of data scientists based on country, it can be seen that US has approximately 4 times the number of data scientists as compared to India while the European countries (France, UK and Germany) have fairly similar proportions of Data Scientists. Therefore, it can be inferred that US has the largest data science community.



```
# In this case, bar chart is the most suitable plot
#Bar chart for the top 5 countries with the most number of data scientists.

try:
    #Create a list of calculated percentage based on each gender
    number_of_data_scientists = [len((df_country[df_country['Country'] == str(df_country['Country'].
                                                                          value_counts()[0:1].
                                                                          index.tolist()[0])])),
                                 len((df_country[df_country['Country'] == str(df_country['Country'].
                                                                          value_counts()[1:2].
                                                                          index.tolist()[0])])),
                                 len((df_country[df_country['Country'] == str(df_country['Country'].
                                                                          value_counts()[2:3].
                                                                          index.tolist()[0])])),
                                 len((df_country[df_country['Country'] == str(df_country['Country'].
                                                                          value_counts()[3:4].
                                                                          index.tolist()[0])])),
                                 len((df_country[df_country['Country'] == str(df_country['Country'].
                                                                          value_counts()[4:5].
                                                                          index.tolist()[0])]))]
    country_names = df_country['Country'].value_counts()[:5].index.tolist()
    print (country_names, number_of_data_scientists)

except Exception as e:
    logging.error(e)

#Function Reuse for creating bar chart
#Call Barchart plotting Function with the required parameters for plotting a barchart
plotting_bar_chart_for_single_category('Barchart Indicating Number of Data Scientists based on Top 5 Countries\n',
                                       '\nCountry', 'Number of Data Scientists', country_names,
                                       number_of_data_scientists)
```

Figure 35: Code Snippet (2) for 1.3C

In the case of plotting categorical data based on the frequency/count, a bar chart is the most suitable visualisation technique to be used. Based on the above code snippet, the usage of effective error-handling (try-except block) prevents abnormal termination of the operation. A list was initialised and filled with the number of data scientist of each of the top five countries and the top five country names were computed automatically, both without manually inputting the name of the countries. The country names and their corresponding frequency were then printed. Furthermore, the function for plotting a single bar chart that has been previously defined in 1.3.2 (please refer to script) is re-utilised by parsing all the required arguments.

```
['United States', 'India', 'France', 'United Kingdom', 'Germany'] [414, 111, 60, 55, 50]
```

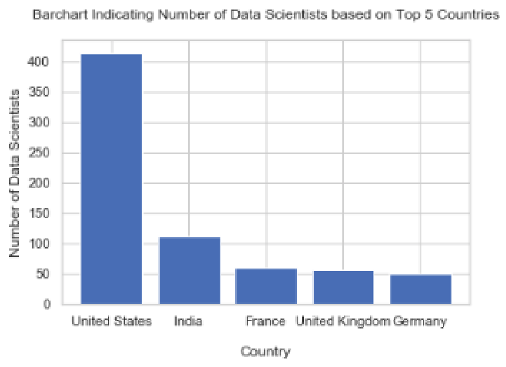
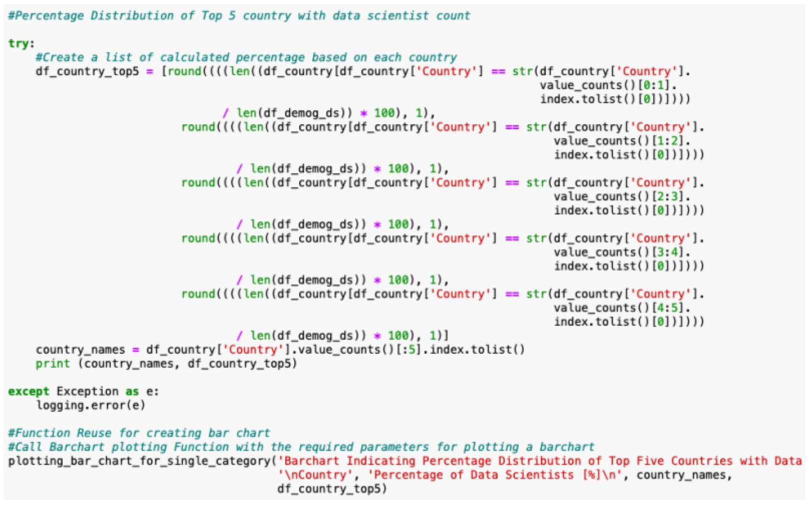


Figure 35: Output (2) for 1.3C

Based on the above output, it can be concluded that US has the largest data science community with 414 data scientists while India, being on the second position, has a smaller data scientist population of 4 times less than that of US. The European Countries (France, UK and Germany) has fairly similar number of data scientists with only very minor differences of 5 data scientists.

2.10. 1.3D: Data Scientist's Top 5 Countries Percentage Visualization

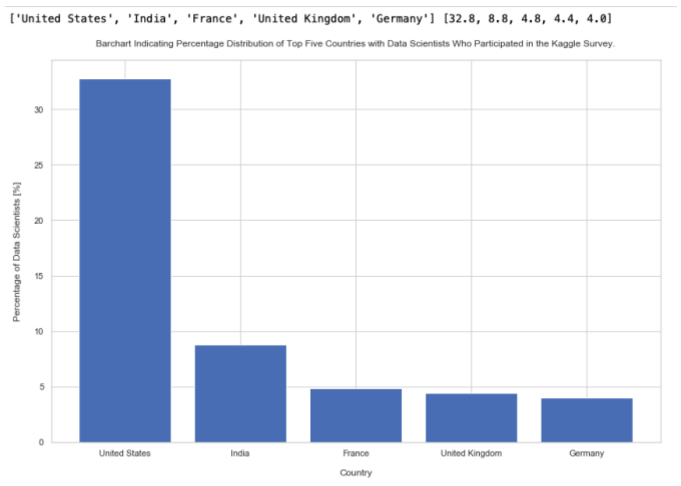
During this task, the data scientist is required to plot a bar chart to visually represent the percentage of data scientists in the top five countries based on the overall number of data scientists in all countries.



```
#Percentage Distribution of Top 5 country with data scientist count

try:
    #Create a list of calculated percentage based on each country
    df_country_top5 = [round((((len((df_country[df_country['Country'] == str(df_country['Country'].
                                                                          value_counts()[0:1].
                                                                          index.tolist()[0])]))))
                           / len(df_demog_ds)) * 100), 1),
                        round((((len((df_country[df_country['Country'] == str(df_country['Country'].
                                                                          value_counts()[1:2].
                                                                          index.tolist()[0])]))))
                            / len(df_demog_ds)) * 100), 1),
                        round((((len((df_country[df_country['Country'] == str(df_country['Country'].
                                                                          value_counts()[2:3].
                                                                          index.tolist()[0])]))))
                            / len(df_demog_ds)) * 100), 1),
                        round((((len((df_country[df_country['Country'] == str(df_country['Country'].
                                                                          value_counts()[3:4].
                                                                          index.tolist()[0])]))))
                            / len(df_demog_ds)) * 100), 1),
                        round((((len((df_country[df_country['Country'] == str(df_country['Country'].
                                                                          value_counts()[4:5].
                                                                          index.tolist()[0])]))))
                            / len(df_demog_ds)) * 100), 1)]
    country_names = df_country['Country'].value_counts()[:5].index.tolist()
    print (country_names, df_country_top5)

except Exception as e:
    logging.error(e)

#Function Reuse for creating bar chart
#Call Barchart plotting Function with the required parameters for plotting a barchart
plotting_bar_chart_for_single_category('Barchart Indicating Percentage Distribution of Top Five Countries with Data
                                       '\nCountry', 'Percentage of Data Scientists [%]\n', country_names,
                                       df_country_top5)
```

Figure 36: Code Snippet for 1.3D

Based on the code snippet, the usage of effective error-handling (try-except block) prevents abnormal termination of the operation. A list was initialised and filled with the percentage proportion of data scientist of each of the top five countries and the top five country names were computed automatically, both without manually inputting the name of the countries. The country names and their corresponding percentage proportion were then printed. Furthermore, the function for plotting a single bar chart that has been previously defined in 1.3.2 (please refer to script) is re-utilised by parsing all the required arguments.



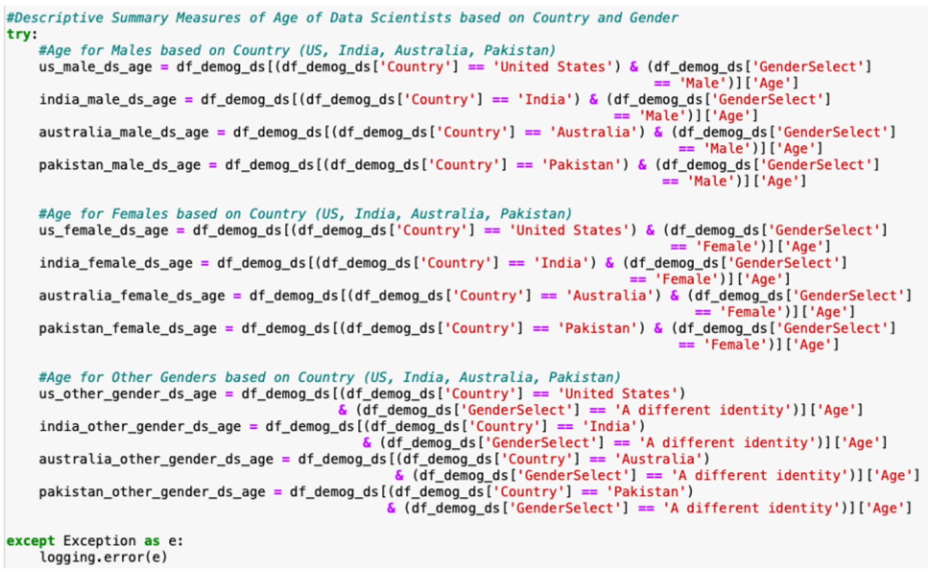
```
['United States', 'India', 'France', 'United Kingdom', 'Germany'] [32.8, 8.8, 4.8, 4.4, 4.0]
```

Figure 37: Output for 1.3D

Based on the above output, it can be concluded that US has the largest data science community with 32.8% while India, being on the second position, has a smaller data scientist population of 4 times less than that of US. The European Countries (France, UK and Germany) has fairly similar proportions with only very minor differences. Furthermore, US has approximately 11% of data scientists as compared to India, France, UK and Germany combined.

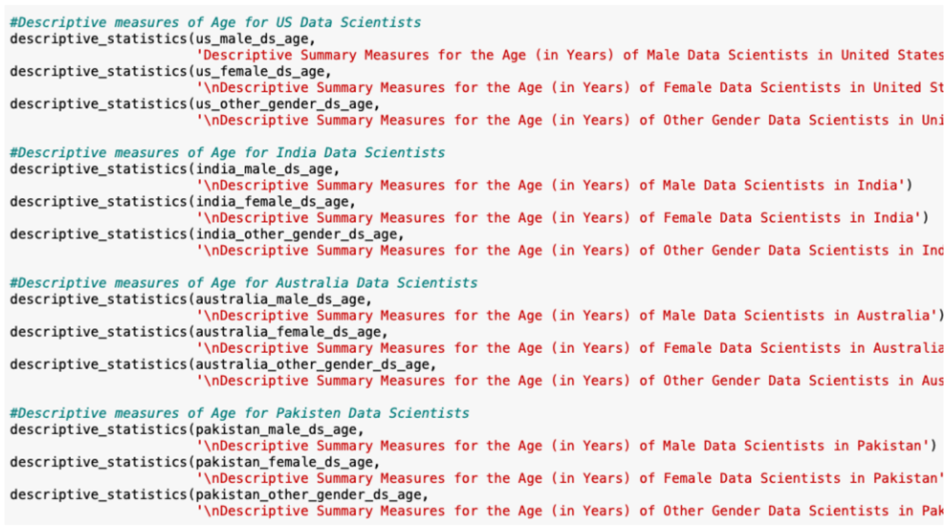
2.11. 1.3E: Data Scientist's Age Summary based on Country

During this task, the data scientist was required to compute the mean and median age of data scientists based on each gender category for the United States, India, Australia and Pakistan.



```python
#Descriptive Summary Measures of Age of Data Scientists based on Country and Gender
try:
    #Age for Males based on Country (US, India, Australia, Pakistan)
    us_male_ds_age = df_demog_ds[(df_demog_ds['Country'] == 'United States') & (df_demog_ds['GenderSelect']
                                                                                 == 'Male')]['Age']
    india_male_ds_age = df_demog_ds[(df_demog_ds['Country'] == 'India') & (df_demog_ds['GenderSelect']
                                                                            == 'Male')]['Age']
    australia_male_ds_age = df_demog_ds[(df_demog_ds['Country'] == 'Australia') & (df_demog_ds['GenderSelect']
                                                                                    == 'Male')]['Age']
    pakistan_male_ds_age = df_demog_ds[(df_demog_ds['Country'] == 'Pakistan') & (df_demog_ds['GenderSelect']
                                                                                  == 'Male')]['Age']

    #Age for Females based on Country (US, India, Australia, Pakistan)
    us_female_ds_age = df_demog_ds[(df_demog_ds['Country'] == 'United States') & (df_demog_ds['GenderSelect']
                                                                                   == 'Female')]['Age']
    india_female_ds_age = df_demog_ds[(df_demog_ds['Country'] == 'India') & (df_demog_ds['GenderSelect']
                                                                              == 'Female')]['Age']
    australia_female_ds_age = df_demog_ds[(df_demog_ds['Country'] == 'Australia') & (df_demog_ds['GenderSelect']
                                                                                      == 'Female')]['Age']
    pakistan_female_ds_age = df_demog_ds[(df_demog_ds['Country'] == 'Pakistan') & (df_demog_ds['GenderSelect']
                                                                                    == 'Female')]['Age']

    #Age for Other Genders based on Country (US, India, Australia, Pakistan)
    us_other_gender_ds_age = df_demog_ds[(df_demog_ds['Country'] == 'United States')
                                         & (df_demog_ds['GenderSelect'] == 'A different identity')]['Age']
    india_other_gender_ds_age = df_demog_ds[(df_demog_ds['Country'] == 'India')
                                            & (df_demog_ds['GenderSelect'] == 'A different identity')]['Age']
    australia_other_gender_ds_age = df_demog_ds[(df_demog_ds['Country'] == 'Australia')
                                                & (df_demog_ds['GenderSelect'] == 'A different identity')]['Age']
    pakistan_other_gender_ds_age = df_demog_ds[(df_demog_ds['Country'] == 'Pakistan')
                                               & (df_demog_ds['GenderSelect'] == 'A different identity')]['Age']

except Exception as e:
    logging.error(e)
```

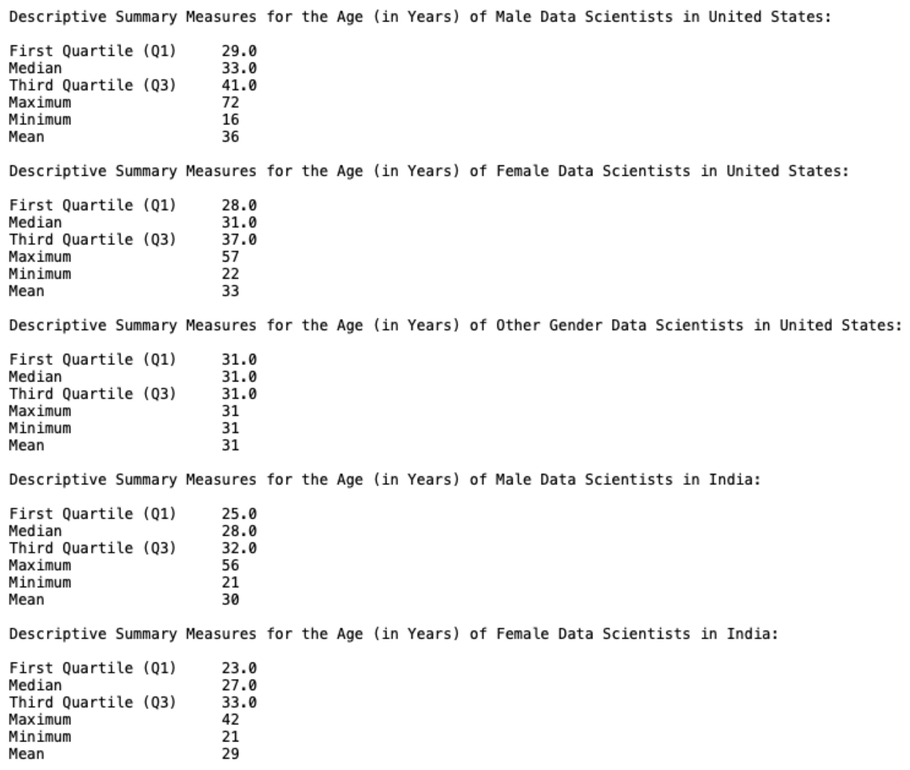
Figure 38: Code Snippet (1) for 1.3E



```python
#Descriptive measures of Age for US Data Scientists
descriptive_statistics(us_male_ds_age,
                       'Descriptive Summary Measures for the Age (in Years) of Male Data Scientists in United States
descriptive_statistics(us_female_ds_age,
                       '\nDescriptive Summary Measures for the Age (in Years) of Female Data Scientists in United St
descriptive_statistics(us_other_gender_ds_age,
                       '\nDescriptive Summary Measures for the Age (in Years) of Other Gender Data Scientists in Uni

#Descriptive measures of Age for India Data Scientists
descriptive_statistics(india_male_ds_age,
                       '\nDescriptive Summary Measures for the Age (in Years) of Male Data Scientists in India')
descriptive_statistics(india_female_ds_age,
                       '\nDescriptive Summary Measures for the Age (in Years) of Female Data Scientists in India')
descriptive_statistics(india_other_gender_ds_age,
                       '\nDescriptive Summary Measures for the Age (in Years) of Other Gender Data Scientists in Ind

#Descriptive measures of Age for Australia Data Scientists
descriptive_statistics(australia_male_ds_age,
                       '\nDescriptive Summary Measures for the Age (in Years) of Male Data Scientists in Australia')
descriptive_statistics(australia_female_ds_age,
                       '\nDescriptive Summary Measures for the Age (in Years) of Female Data Scientists in Australia
descriptive_statistics(australia_other_gender_ds_age,
                       '\nDescriptive Summary Measures for the Age (in Years) of Other Gender Data Scientists in Aus

#Descriptive measures of Age for Pakisten Data Scientists
descriptive_statistics(pakistan_male_ds_age,
                       '\nDescriptive Summary Measures for the Age (in Years) of Male Data Scientists in Pakistan')
descriptive_statistics(pakistan_female_ds_age,
                       '\nDescriptive Summary Measures for the Age (in Years) of Female Data Scientists in Pakistan'
descriptive_statistics(pakistan_other_gender_ds_age,
                       '\nDescriptive Summary Measures for the Age (in Years) of Other Gender Data Scientists in Pak
```

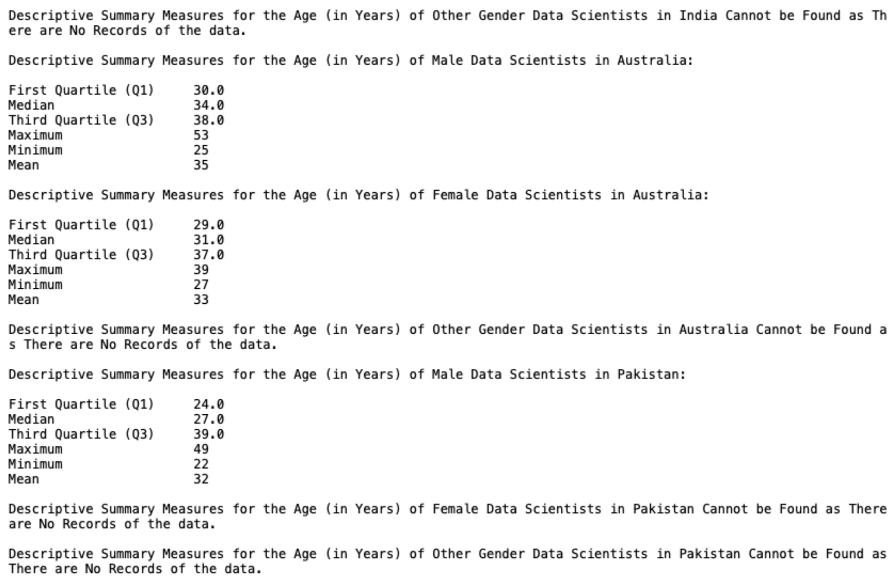
Figure 39: Code Snippet (2) for 1.3E

Based on the above code snippets, the usage of effective error-handling (try-except block) prevents abnormal termination of the operation. Furthermore, the values for the age of the male data scientists in US is extracted and saved in the variable us_male_ds_age. Similarly, the age of all genders of data scientists in US, India, Australia and Pakistan are extracted and saved in their respective variables as shown in the snippets. The function which has been previously defined in 1.3A for calculating the descriptive statistics is re-utilized as required for the descriptive statistical summaries of all the gender categories in the four aforementioned countries.



```
Descriptive Summary Measures for the Age (in Years) of Male Data Scientists in United States:

First Quartile (Q1)      29.0
Median                   33.0
Third Quartile (Q3)      41.0
Maximum                  72
Minimum                  16
Mean                     36

Descriptive Summary Measures for the Age (in Years) of Female Data Scientists in United States:

First Quartile (Q1)      28.0
Median                   31.0
Third Quartile (Q3)      37.0
Maximum                  57
Minimum                  22
Mean                     33

Descriptive Summary Measures for the Age (in Years) of Other Gender Data Scientists in United States:

First Quartile (Q1)      31.0
Median                   31.0
Third Quartile (Q3)      31.0
Maximum                  31
Minimum                  31
Mean                     31

Descriptive Summary Measures for the Age (in Years) of Male Data Scientists in India:

First Quartile (Q1)      25.0
Median                   28.0
Third Quartile (Q3)      32.0
Maximum                  56
Minimum                  21
Mean                     30

Descriptive Summary Measures for the Age (in Years) of Female Data Scientists in India:

First Quartile (Q1)      23.0
Median                   27.0
Third Quartile (Q3)      33.0
Maximum                  42
Minimum                  21
Mean                     29
```

Figure 40: Output (1) for 1.3E



```
Descriptive Summary Measures for the Age (in Years) of Other Gender Data Scientists in India Cannot be Found as Th
ere are No Records of the data.

Descriptive Summary Measures for the Age (in Years) of Male Data Scientists in Australia:

First Quartile (Q1)      30.0
Median                   34.0
Third Quartile (Q3)      38.0
Maximum                  53
Minimum                  25
Mean                     35

Descriptive Summary Measures for the Age (in Years) of Female Data Scientists in Australia:

First Quartile (Q1)      29.0
Median                   31.0
Third Quartile (Q3)      37.0
Maximum                  39
Minimum                  27
Mean                     33

Descriptive Summary Measures for the Age (in Years) of Other Gender Data Scientists in Australia Cannot be Found a
s There are No Records of the data.

Descriptive Summary Measures for the Age (in Years) of Male Data Scientists in Pakistan:

First Quartile (Q1)      24.0
Median                   27.0
Third Quartile (Q3)      39.0
Maximum                  49
Minimum                  22
Mean                     32

Descriptive Summary Measures for the Age (in Years) of Female Data Scientists in Pakistan Cannot be Found as There
are No Records of the data.

Descriptive Summary Measures for the Age (in Years) of Other Gender Data Scientists in Pakistan Cannot be Found as
There are No Records of the data.
```

Figure 41: Output (2) for 1.3E

Based on the above output analysis of the Kaggle Survey Data, in US, male data scientists have a higher mean (36) and median age (33) as compared to Female and Other gender data scientists. Other gender data scientists in the US have the lowest mean age. Interestingly, the median age of female and other gender data scientists in the US are similar. In India, male data scientists have a higher mean (30) and median age (28) as compared to Female data scientists. There were no records of other gender data scientists in India and no descriptive summary measures were able to be calculated for this gender category. Similarly, in Australia, male data scientists have a higher mean (35) and median age (34) as compared to Female data scientists. There were no records of other gender data scientists in Australia and no descriptive summary measures were able to be calculated for this gender category. Lastly, Pakistan has only male data scientists with a mean and median age of 32 and 27 respectively. There are no records of female and other gender data scientists in Pakistan and no descriptive summary measures were able to be calculated for those gender categories.

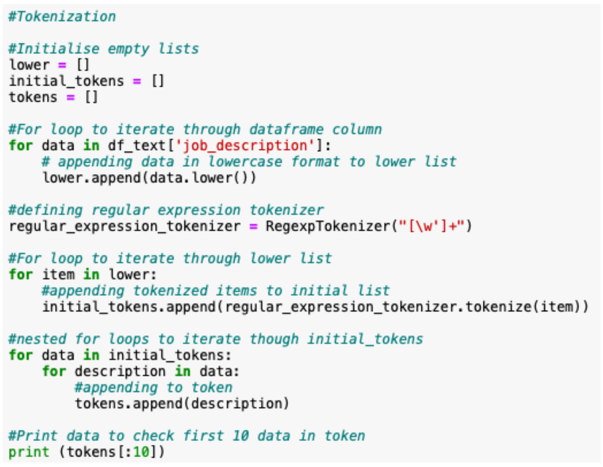
Overall, it can be concluded that male data scientists in the US have the highest mean and median age as compared to male data scientists of other countries. Interestingly, both female data scientists in Australia and US have similar highest mean and median age as compared to those of other countries. The only country with other gender data scientists out of the four countries is US. In Pakistan, the data science job market seems fully male dominated.

3.0. PART 2: TEXT ANALYTICS

This section deals with the text analysis of the job postings in the aim of producing some insights about data science job advertisement data.

3.1. 2.1A: Tokenization

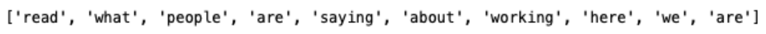
During this task, the data scientist is required to extract the tokens from the 'job_description' column from the JobPosting.csv file and append the extracted tokens to the tokens list.



```
#Tokenization

#Initialise empty lists
lower = []
initial_tokens = []
tokens = []

#For loop to iterate through dataframe column
for data in df_text['job_description']:
    # appending data in lowercase format to lower list
    lower.append(data.lower())

#defining regular expression tokenizer
regular_expression_tokenizer = RegexpTokenizer("[\w']+")

#For loop to iterate through lower list
for item in lower:
    #appending tokenized items to initial list
    initial_tokens.append(regular_expression_tokenizer.tokenize(item))

#nested for loops to iterate though initial_tokens
for data in initial_tokens:
    for description in data:
        #appending to token
        tokens.append(description)

#Print data to check first 10 data in token
print (tokens[:10])
```

Figure 42: Code Snippet for 2.1A

Based on the above code snippet, the empty lists are initialised. Firstly, a for loop is utilized to iterate through the data frame column and all the data are converted to lower case letters and appended to the lower list. A regular expression tokenizer is defined as sentence and word tokenizer are not suitable as a regular expression tokenizer allows for custom word tokenization. The tokenized items are appended to an initial token which is then looped through using nested for loops to get each token and append them to the tokens list. The first 10 data in the tokens is displayed for testing purposes.



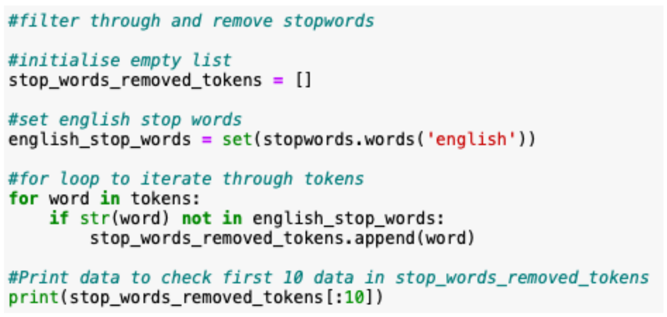
```
['read', 'what', 'people', 'are', 'saying', 'about', 'working', 'here', 'we', 'are']
```

Figure 43: Output for 2.1A

The output above shows the first 10 data in the tokens list.

3.2. 2.1B: List words with frequency more than 6000

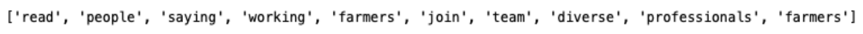
During this task, the data scientist was required to filter through the tokens list, remove the stop words and list the words with frequency higher than 6000.



```
#filter through and remove stopwords

#initialise empty list
stop_words_removed_tokens = []

#set english stop words
english_stop_words = set(stopwords.words('english'))

#for loop to iterate through tokens
for word in tokens:
    if str(word) not in english_stop_words:
        stop_words_removed_tokens.append(word)

#Print data to check first 10 data in stop_words_removed_tokens
print(stop_words_removed_tokens[:10])
```

Figure 44: Code Snippet (1) for 2.1B

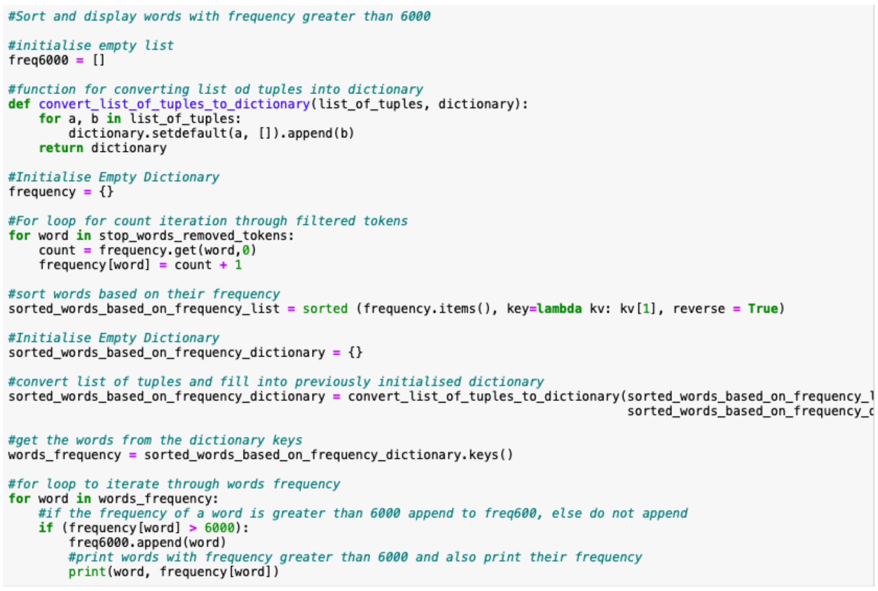
Based on the above code snippet, the empty list is initialised. The stop words in this scenario are set to those of English language and saved in a variable, english_stop_words, of set data type. A for loop is used to iterate through the tokens list. Using a condition-based (if) statement to check if each word in a token is not in the set english_stop_words variable. The words, which are not in the stop words list, are then appended to the stop_words_removed_tokens list and the first 10 data of the stop_words_removed_tokens list is then displayed for test purposes.



```
['read', 'people', 'saying', 'working', 'farmers', 'join', 'team', 'diverse', 'professionals', 'farmers']
```

Figure 45: Output (1) for 2.1B

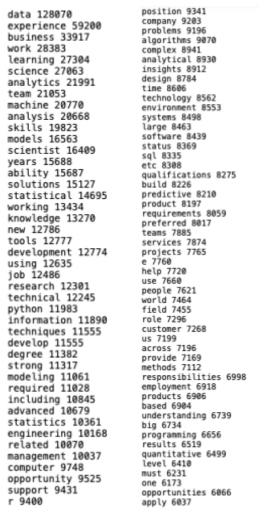
Based on the above output, it can be seen that the stop words have been discarded.



```
#Sort and display words with frequency greater than 6000

#initialise empty list
freq6000 = []

#function for converting list od tuples into dictionary
def convert_list_of_tuples_to_dictionary(list_of_tuples, dictionary):
    for a, b in list_of_tuples:
        dictionary.setdefault(a, []).append(b)
    return dictionary

#Initialise Empty Dictionary
frequency = {}

#For loop for count iteration through filtered tokens
for word in stop_words_removed_tokens:
    count = frequency.get(word,0)
    frequency[word] = count + 1

#sort words based on their frequency
sorted_words_based_on_frequency_list = sorted (frequency.items(), key=lambda kv: kv[1], reverse = True)

#Initialise Empty Dictionary
sorted_words_based_on_frequency_dictionary = {}

#convert list of tuples and fill into previously initialised dictionary
sorted_words_based_on_frequency_dictionary = convert_list_of_tuples_to_dictionary(sorted_words_based_on_frequency_l
                                                                                  sorted_words_based_on_frequency_d

#get the words from the dictionary keys
words_frequency = sorted_words_based_on_frequency_dictionary.keys()

#for loop to iterate through words frequency
for word in words_frequency:
    #if the frequency of a word is greater than 6000 append to freq600, else do not append
    if (frequency[word] > 6000):
        freq6000.append(word)
        #print words with frequency greater than 6000 and also print their frequency
        print(word, frequency[word])
```

Figure 46: Code Snippet (2) for 2.1B

Based on the above code snippet, the empty list and dictionary are initialised. A function was defined to convert a list of tuples into a dictionary by accepting two parameters and looping through using a for loop to convert the list to a dictionary. A for loop was used to iterate through stop_words_removed_tokens list and count the frequency of occurrence of each word. The words and their frequency of occurrence are then arranged in descending frequency order. The function for converting a list of tuples into a dictionary is called. The keys of the dictionary created are the words in descending frequency of occurrence order. A for loop is used to iterate through word_frequency and using a condition-based (if) statement to check the words with frequency greater than 6000. Only words with frequency greater than 6000 are then appended to freq6000. The words with frequency greater than 6000 and their exact frequency of occurrence are then displayed.



```
data 128070
experience 59200
business 33917
work 28383
learning 27304
science 27063
analytics 21991
team 21053
machine 20770
analysis 20668
skills 19823
models 16563
scientist 16409
years 15688
ability 15687
solutions 15127
statistical 14695
working 13434
knowledge 13270
new 12786
tools 12777
development 12774
using 12635
job 12486
research 12301
technical 12245
python 11983
information 11890
techniques 11555
develop 11555
degree 11382
strong 11317
modeling 11061
required 11028
including 10845
advanced 10679
statistics 10361
engineering 10168
related 10070
management 10037
computer 9748
opportunity 9525
support 9431
r 9400
```

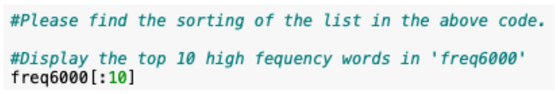
```
position 9341
company 9203
problems 9196
algorithms 9070
complex 8941
analytical 8930
insights 8912
design 8784
time 8606
technology 8562
environment 8553
systems 8498
large 8463
software 8439
status 8369
sql 8335
etc 8308
qualifications 8275
build 8226
predictive 8210
product 8197
requirements 8059
preferred 8017
teams 7885
services 7874
projects 7765
e 7760
help 7720
use 7660
people 7621
world 7464
field 7455
role 7296
customer 7268
us 7199
across 7196
provide 7169
methods 7112
responsibilities 6998
employment 6918
products 6906
based 6904
understanding 6739
big 6734
programming 6656
results 6519
quantitative 6499
level 6410
must 6231
one 6173
opportunities 6066
apply 6037
```

Figure 47: Output (2) for 2.1B

The output shows the words with frequency greater than 6000. There are 96 words with frequency of occurrence greater than 6000. Based on the output, it can be seen that the word ‘data’ is of highest occurrence (128070 times). Furthermore, it can be concluded that data analysts with Python (11983 times) skills are more sought after followed by R programming (9400 times) in the US. With 15688 occurences of the word ‘years’, it can be deduced that more than one year of experience is being desired for data scientist related jobs in the US. The high occurrence of the word ‘degree’ (11382 times) can also infer that the minimum educational requirement for a data scientist job is a bachelor’s degree or higher.

3.3. 2.1C: Sort and List top 10 words with highest frequency

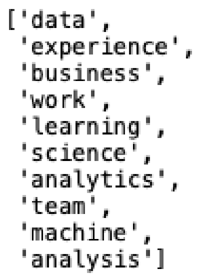
During this task, the data scientist was required to sort the variable freq6000 values and display the top 10 words with the highest frequency in freq6000.



```
#Please find the sorting of the list in the above code.

#Display the top 10 high fequency words in 'freq6000'
freq6000[:10]
```

Figure 48: Code Snippet for 2.1C

Based on the above code screenshot, the sorting part was already performed by the data scientist in 2.1C. Please refer to 2.1C for the sorting code. The top 10 words with the most common occurrence in freq6000 are displayed.



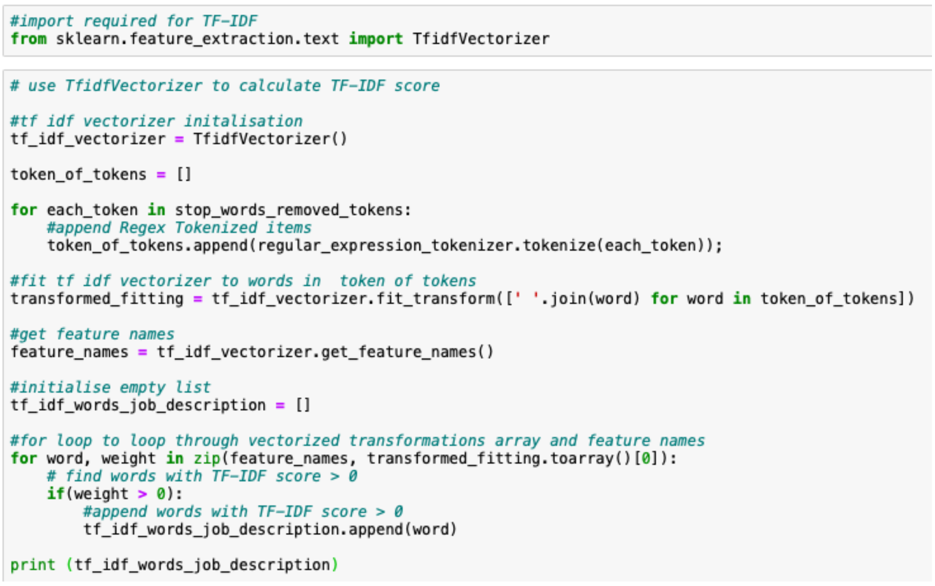
```
['data',
 'experience',
 'business',
 'work',
 'learning',
 'science',
 'analytics',
 'team',
 'machine',
 'analysis']
```

Figure 49: Output for 2.1C

The above output displays the top 10 words with the most common occurrence in freq6000. It can be seen that ‘data’ is the most occurring word while ‘analysis’ is the 10th most occurring word. Based on the output above, it can be inferenced that machine learning is one of the most sought skills of a data scientist in the US as the word ‘learning’ is the 5th most occurring word and ‘machine’ is the 9th one. As the word experience is the 2nd most occurring word, it can be inferred that at least one year of experience is required to apply for those jobs.

3.4. 2.1D: Self-defined Text Analysis Task

During this task, the data scientist was required to implement one more analytics task of own preference and report the discovery of the analysis.



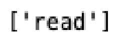
```
#import required for TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# use TfidfVectorizer to calculate TF-IDF score

#tf idf vectorizer initalisation
tf_idf_vectorizer = TfidfVectorizer()

token_of_tokens = []

for each_token in stop_words_removed_tokens:
    #append Regex Tokenized items
    token_of_tokens.append(regular_expression_tokenizer.tokenize(each_token));

#fit tf idf vectorizer to words in  token of tokens
transformed_fitting = tf_idf_vectorizer.fit_transform([' '.join(word) for word in token_of_tokens])

#get feature names
feature_names = tf_idf_vectorizer.get_feature_names()

#initialise empty list
tf_idf_words_job_description = []

#for loop to loop through vectorized transformations array and feature names
for word, weight in zip(feature_names, transformed_fitting.toarray()[0]):
    # find words with TF-IDF score > 0
    if(weight > 0):
        #append words with TF-IDF score > 0
        tf_idf_words_job_description.append(word)

print (tf_idf_words_job_description)
```

Figure 49: Code Snippet for 2.1D

For this task, the data scientist chose to perform feature extraction using Term Frequency – Inverse Document Frequency (TF-IDF) matching score technique. Feature extraction is necessary to get important insights about the job descriptions. TF-IDF allows for the quantification of words to reveal their importance in the job descriptions gathered. (Scott, 2019)

Based on the above code, the library skit-learn and the TF-IDF vector are imported. The vector is then initialised. An empty list is also initialised. A for loop was used to iterate through the filtered tokens (stop words removed before) and to tokenize each token. A fitted transformation is applied to the vectorized. A for loop followed by a condition-based (if) statement is used to append words with weight greater than 0.0 in the list. The list of weights greater than 0 is then displayed.



```
['read']
```

Figure 50: Output for 2.1D

The output above shows the list of words of significance weight greater than 0. Based on the output above, it can be concluded that the word ‘read’ is of highest significance from the job description data.

4.0. CONCLUSION

During this assessment, the data scientist was familiarised with hands-on experience of basic data analysis, exploratory data analysis, visualisation techniques and plotting as well as advanced text analytics using Python. After the analysis, some few interesting insights were revealed both about the data science job market. Some of them include:

- Nearly 70% of data scientists have a postgraduate qualification (Masters or Doctoral) which infers that in the future, most data science jobs would include a minimum requirement of a postgraduate qualification;
- The global average salary of a data scientist varries between AU$88,000 to AU$90,000;
- Australian Data Scientists are comparatively highly paid as compared to those of other countries with an average yearly income of $143500;
- The global average age of a data scientist is 32 years;
- The data science market is mostly male dominated in all countries with Pakistan being a country with full male dominance over the market;
- The LGBTQ+ category is the least participating in the data science job industry with most of them working in USA;
- USA has the largest data science community in the world followed by India;
- Most data scientist jobs in USA require a minimum of 1 year of experience to several years of experience;
- Python, R and SQL are the most sought after skills of data scientists.

5.0. REFERENCES

Scott, W., 2019. *TF-IDF from scratch in python on real world dataset.*. [Online] Available at: https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089
[Accessed 15 April 2020].