SIT742 ASSIGNMENT 1

RAJESHKUMAR RAMPRASAD MOURYA
rmourya@deakin.edu.au
218615876

Data Exploration : Part 1

Task 1.0.A Columns and their #Records

Source code:

```
# Check columns with NAs
df_demog.isna().any() # isna() provide detailed values stating where
                       value is NA or not
                      # isna().any() provide column which contains at
                       least one NA entry
                      # True signifies presence of null value
```

Output :

```
GenderSelect                    False
Country                         False
Age                             False
EmploymentStatus                False
CodeWriter                      False
CurrentJobTitleSelect           False
TitleFit                         True
CurrentEmployerType              True
MLToolNextYearSelect             True
MLMethodNextYearSelect           True
LanguageRecommendationSelect     True
FormalEducation                 False
MajorSelect                      True
FirstTrainingSelect              True
CompensationAmount              False
CompensationCurrency            False
JobSatisfaction                  True
dtype: bool
```

Source code:

```
# Count function displays number of non null values in columns of dataf
rame

df_demog.count()
```

Output:

```
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
```

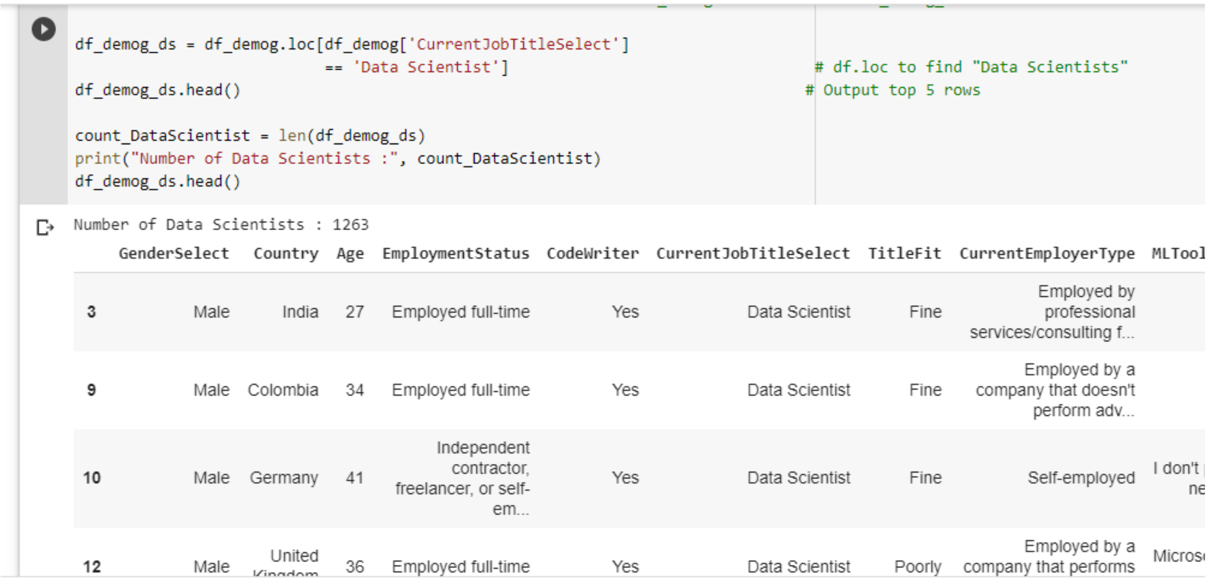
Task 1.0.B #Data Scientists

Source code:

```
# Select Data Scientist from column CurrentJobTitleSelect in df_demog a
nd store into df_demog_ds

df_demog_ds = df_demog.loc[df_demog['CurrentJobTitleSelect']
                           == 'Data Scientist']
         # df.loc to find "Data Scientists"
df_demog_ds.head()
         # Output top 5 rows

count_DataScientist = len(df_demog_ds)
print("Number of Data Scientists :", count_DataScientist)
```

Output

```
Number of Data Scientists : 1263
```



```
df_demog_ds = df_demog.loc[df_demog['CurrentJobTitleSelect']
                           == 'Data Scientist']                        # df.loc to find "Data Scientists"
df_demog_ds.head()                                                    # Output top 5 rows

count_DataScientist = len(df_demog_ds)
print("Number of Data Scientists :", count_DataScientist)
df_demog_ds.head()
```

```
Number of Data Scientists : 1263
```

	GenderSelect	Country	Age	EmploymentStatus	CodeWriter	CurrentJobTitleSelect	TitleFit	CurrentEmployerType	MLTool
3	Male	India	27	Employed full-time	Yes	Data Scientist	Fine	Employed by professional services/consulting f...	
9	Male	Colombia	34	Employed full-time	Yes	Data Scientist	Fine	Employed by a company that doesn't perform adv...	
10	Male	Germany	41	Independent contractor, freelancer, or self-em...	Yes	Data Scientist	Fine	Self-employed	I don't ne
12	Male	United Kingdom	36	Employed full-time	Yes	Data Scientist	Poorly	Employed by a company that performs	Micros

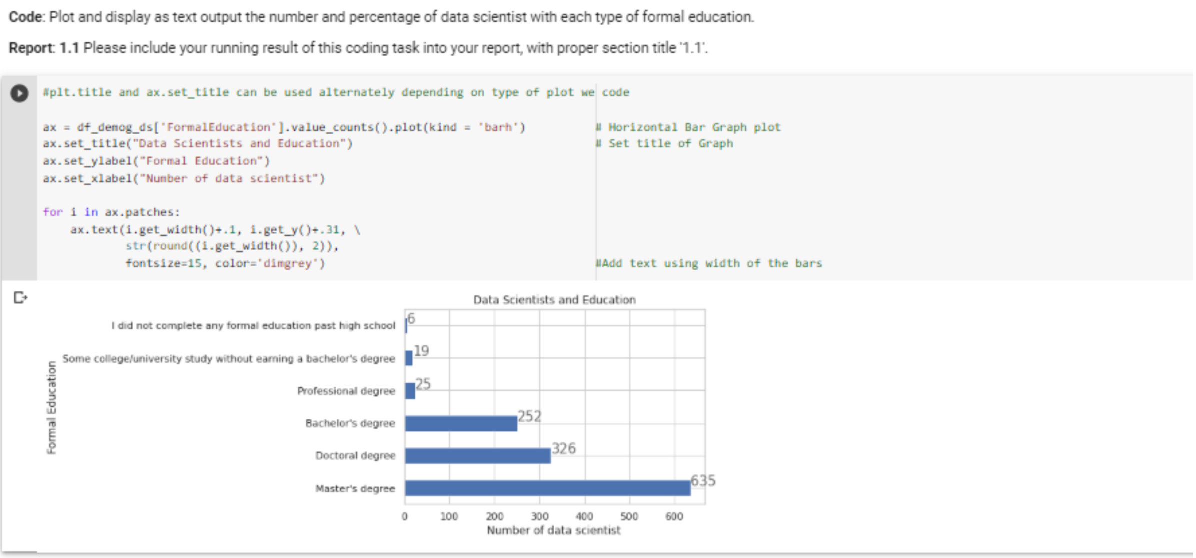
Task 1.1 Plot and Info on DS's Education

Source code:

```
#plt.title and ax.set_title can be used alternately depending on type o
f plot we code
```

```
ax = df_demog_ds['FormalEducation'].value_counts().plot(kind = 'barh')
         # Horizontal Bar Graph plot
ax.set_title("Data Scientists and Education")
         # Set title of Graph
ax.set_ylabel("Formal Education")
ax.set_xlabel("Number of data scientist")

for i in ax.patches:
    ax.text(i.get_width()+.1, i.get_y()+.31, \
            str(round((i.get_width()), 2)),
            fontsize=15, color='dimgrey')
         #Add text using width of the bars
```

Output:



Code: Plot and display as text output the number and percentage of data scientist with each type of formal education.

Report: **1.1** Please include your running result of this coding task into your report, with proper section title '1.1'.

```
#plt.title and ax.set_title can be used alternately depending on type of plot we code

ax = df_demog_ds['FormalEducation'].value_counts().plot(kind = 'barh')          # Horizontal Bar Graph plot
ax.set_title("Data Scientists and Education")                                   # Set title of Graph
ax.set_ylabel("Formal Education")
ax.set_xlabel("Number of data scientist")

for i in ax.patches:
    ax.text(i.get_width()+.1, i.get_y()+.31, \
            str(round((i.get_width()), 2)),
            fontsize=15, color='dimgrey')                                        #Add text using width of the bars
```

Data Scientists and Education
I did not complete any formal education past high school
Some college/university study without earning a bachelor's degree
Professional degree
Bachelor's degree
Doctoral degree
Master's degree
6
19
25
252
326
635
Formal Education
0
100
200
300
400
500
600
Number of data scientist

Task 1.2.A Max/Median Salary in AUD

Source code:

```
medianSalary = result['SalaryAUD'].median()
         # Default function to find median of series
```

Output:

```
Median salary of Data Scientists (in AUD) is :  88829
```

Source Code:

```
print("Median salary of Data Scientists (in AUD) is : ",
      round(medianSalary))
         # Rounded Median Salary readability
```

Output:

```
Maximum salary of Data Scientists (in AUD) is :  742711
```

Task 1.2.B Max/Median Salary for Australians and Boxplot

Source code:

```
AusSalary = result.loc[result['Country'] == 'Australia']
         # Assign Australia's data of DS

medianAus = AusSalary['SalaryAUD'].median()
         # Calculate Median
print("Median salary of Australian Data Scientists (in AUD) is : ",
      round(medianAus))

maxAus = AusSalary['SalaryAUD'].max()
         # Calculate Max
print("Maximum salary of Australian Data Scientists (in AUD) is : ",
      round(maxAus))
         # Results rounded for readability
```

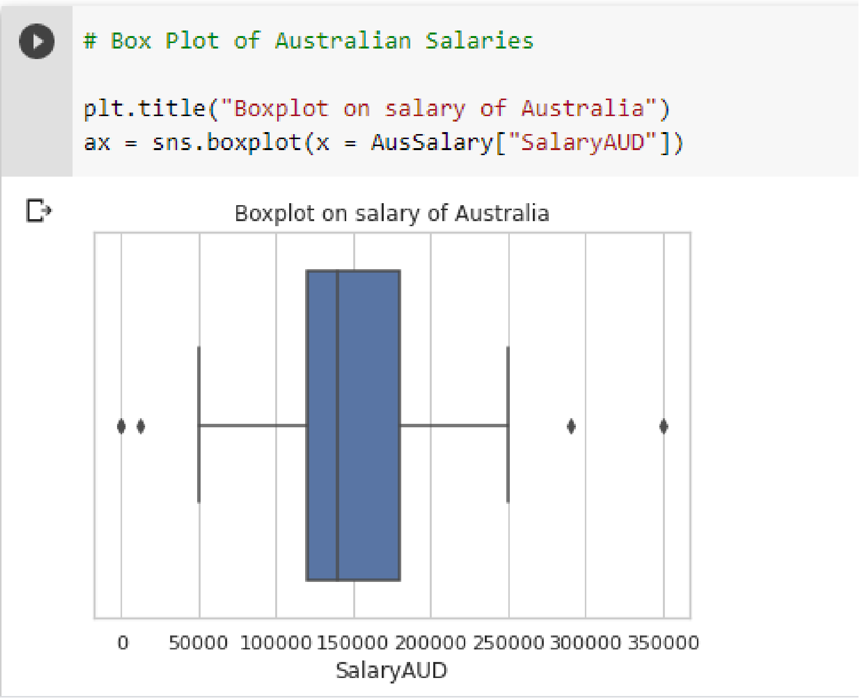
Output:

```
Median salary of Australian Data Scientists (in AUD) is :  140000

Maximum salary of Australian Data Scientists (in AUD) is :  350000
```

Source code:

```
# Box Plot of Australian Salaries

plt.title("Boxplot on salary of Australia")
ax = sns.boxplot(x = AusSalary["SalaryAUD"])
```

Output:



```
# Box Plot of Australian Salaries

plt.title("Boxplot on salary of Australia")
ax = sns.boxplot(x = AusSalary["SalaryAUD"])
```

Task 1.2.C Max/Median Salary for Filtered Australians and Boxplot

Source code:

```
condition1 = AusSalary['SalaryAUD'] > 40000
         # Conditions as specified above
condition2 = AusSalary['SalaryAUD'] < 250000

newSalary = AusSalary[condition1 & condition2]
         # Assigned new salaries to variable

newMedian = newSalary['SalaryAUD'].median()
         # Calculate Median
print("New Median salary of Australian Data Scientists (in AUD) is : "
      , round(newMedian))

newMax = newSalary['SalaryAUD'].max()
         # Calculate Mean
print("New Maximum salary of Australian Data Scientists (in AUD) is : "
      , round(newMax))
         # Results rounded off for readability
```

Output:

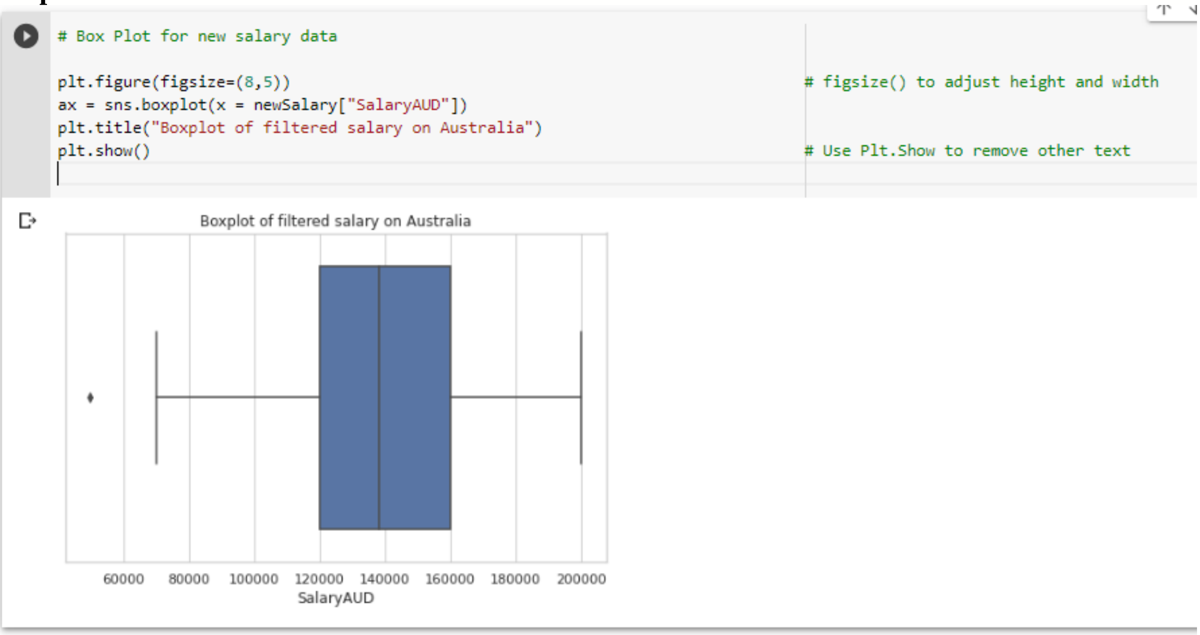
```
New Median salary of Australian Data Scientists (in AUD) is :  138000
New Maximum salary of Australian Data Scientists (in AUD) is :  200000
```

Source Code:

```
# Box Plot for new salary data
```

```
plt.figure(figsize=(8,5))
         # figsize() to adjust height and width
ax = sns.boxplot(x = newSalary["SalaryAUD"])
plt.title("Boxplot of filtered salary on Australia")
plt.show()
         # Use Plt.Show to remove other text
```

Output:



```
# Box Plot for new salary data

plt.figure(figsize=(8,5))                                      # figsize() to adjust height and width
ax = sns.boxplot(x = newSalary["SalaryAUD"])
plt.title("Boxplot of filtered salary on Australia")
plt.show()                                                     # Use Plt.Show to remove other text
```

Task 1.3.A Mean/Median Age and Age Range Counts

1. **Five Summary descriptive stats along with mean**

Source code:

```
# Calculate five number summary


quartiles = np.percentile(df_demog_ds['Age'], [25,50,75])
         # Calculate 25%, 50% and 75% quarters

min_age = df_demog_ds['Age'].min()
max_age = df_demog_ds['Age'].max()
mean_age = round(df_demog_ds['Age'].mean())
         # Mean Rounded off

print("**** Five Point summary ****")
print("Minimum Age : ", min_age)
print("Mean Age : ", mean_age)
print(" Q1 (25%) : ", quartiles[0])
print(" Q2 (50%) : ", quartiles[1])
print(" Q3 (75%) : ", quartiles[2])
print("Maximum Age : ", max_age)
```

Output:

```
**** Five Point summary ****
Minimum Age :  16
Mean Age :  34
 Q1 (25%) :  27.0
 Q2 (50%) :  32.0
 Q3 (75%) :  37.5
Maximum Age :  75
```

2. **What is the mean age of all data scientists?**

Source code:

```
# Code to Calculate Mean Age

AgeMean = df_demog_ds['Age'].mean()
print("Mean Age is :" ,AgeMean)
```

Output:

```
Mean Age is : 33.72050673000792
```

3. **What is the median age of all data scientists?**

Source code:

```
# Code to calculate Median Age

AgeMedian = df_demog_ds['Age'].median()
print("Median Age is :" ,AgeMedian)
```

Output:

```
Median Age is : 32.0
```

4. **how many data scientists aged between 24 and 60**

Source code:

```
# Your code: How many data scientsits aged between 24 and 60

lessthan = df_demog_ds['Age'] > 23
morethan = df_demog_ds['Age'] < 61

ageCount = df_demog_ds[lessthan & morethan]
x = len(ageCount)

print("Number of data scientists between age 24 and 60 : ", x)
```

Output:

```
Number of data scientists between age 24 and 60 :  1188
```

5. **how many respondents were under 18?**

Source code:

```
# Your Code: how many respondents under 18?

lessthan18 = df_demog_ds['Age'] < 18

agebelow = df_demog_ds[lessthan18]
#below18 = agebelow['Age'].count()
below18 = len(agebelow)
print("Number of data scietist under 18: ", below18)
```

Output:

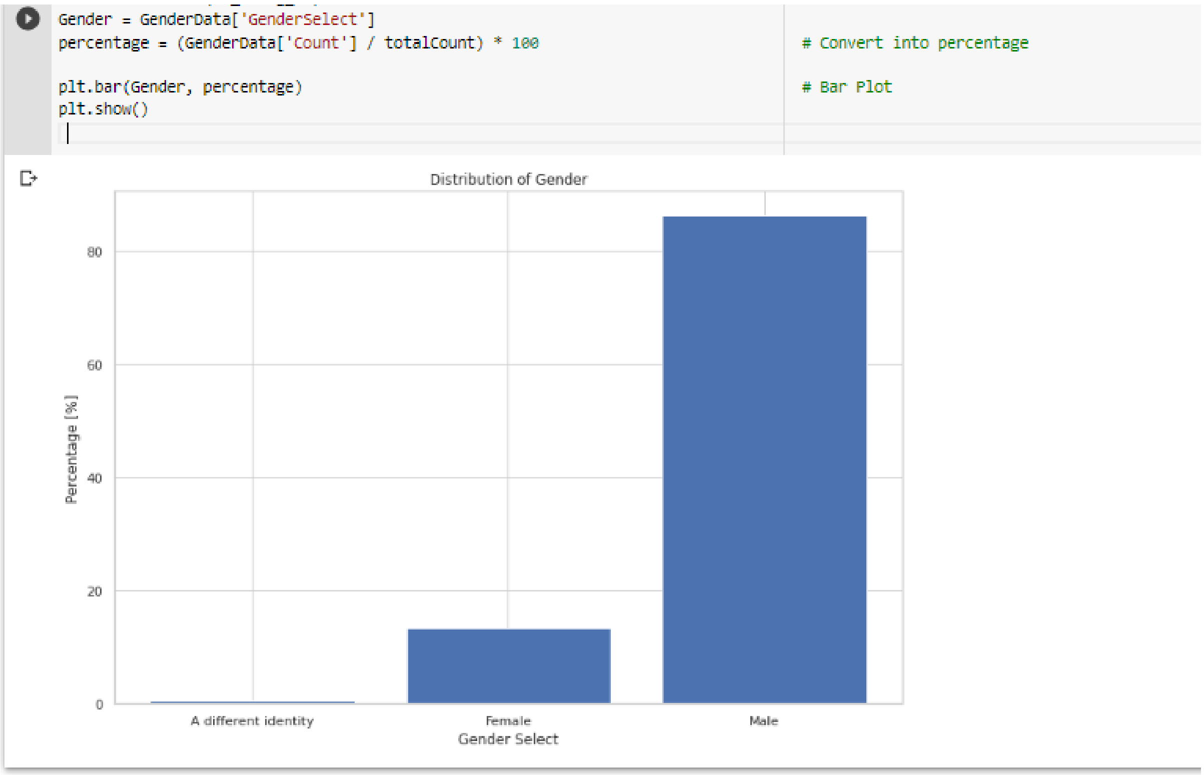
```
Number of data scietist under 18:  1
```

Task 1.3.B Barchart for Gender

Source code:

```
plt.figure(figsize=(12,8))
plt.title('Distribution of Gender')

plt.xlabel('Gender Select')

plt.ylabel('Percentage [%]')

GenderData = df_demog_ds.groupby("GenderSelect").size().reset_index(nam
e='Count') # Use GroupBy on size/count to find

           # Count of each gender
totalCount = len(df_demog_ds)
           # total count
Gender = GenderData['GenderSelect']
percentage = (GenderData['Count'] / totalCount) * 100
           # Convert into percentage

plt.bar(Gender, percentage)
           # Bar Plot
plt.show()
```

Output:



```
Gender = GenderData['GenderSelect']
percentage = (GenderData['Count'] / totalCount) * 100        # Convert into percentage

plt.bar(Gender, percentage)                                   # Bar Plot
plt.show()
```

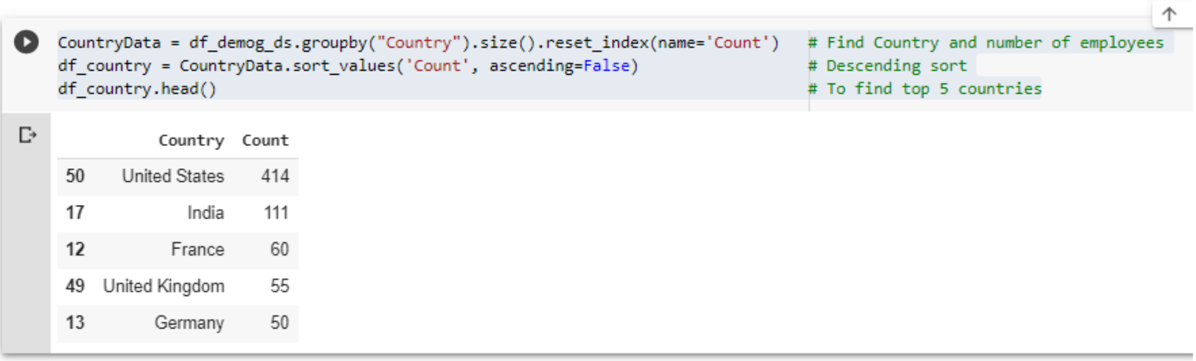
Task 1.3.C Find the top 5 countries of data scientists.

Source code:

```
CountryData = df_demog_ds.groupby("Country").size().reset_index(name='Count')    # Find Country and number of employees
df_country = CountryData.sort_values('Count', ascending=False)
         # Descending sort
df_country.head()
         # To find top 5 countries
```

Output:

Report: **1.3.C** In your report's section '1.3.C', answer what are those top 5 countries and their corresponding number of data scienists



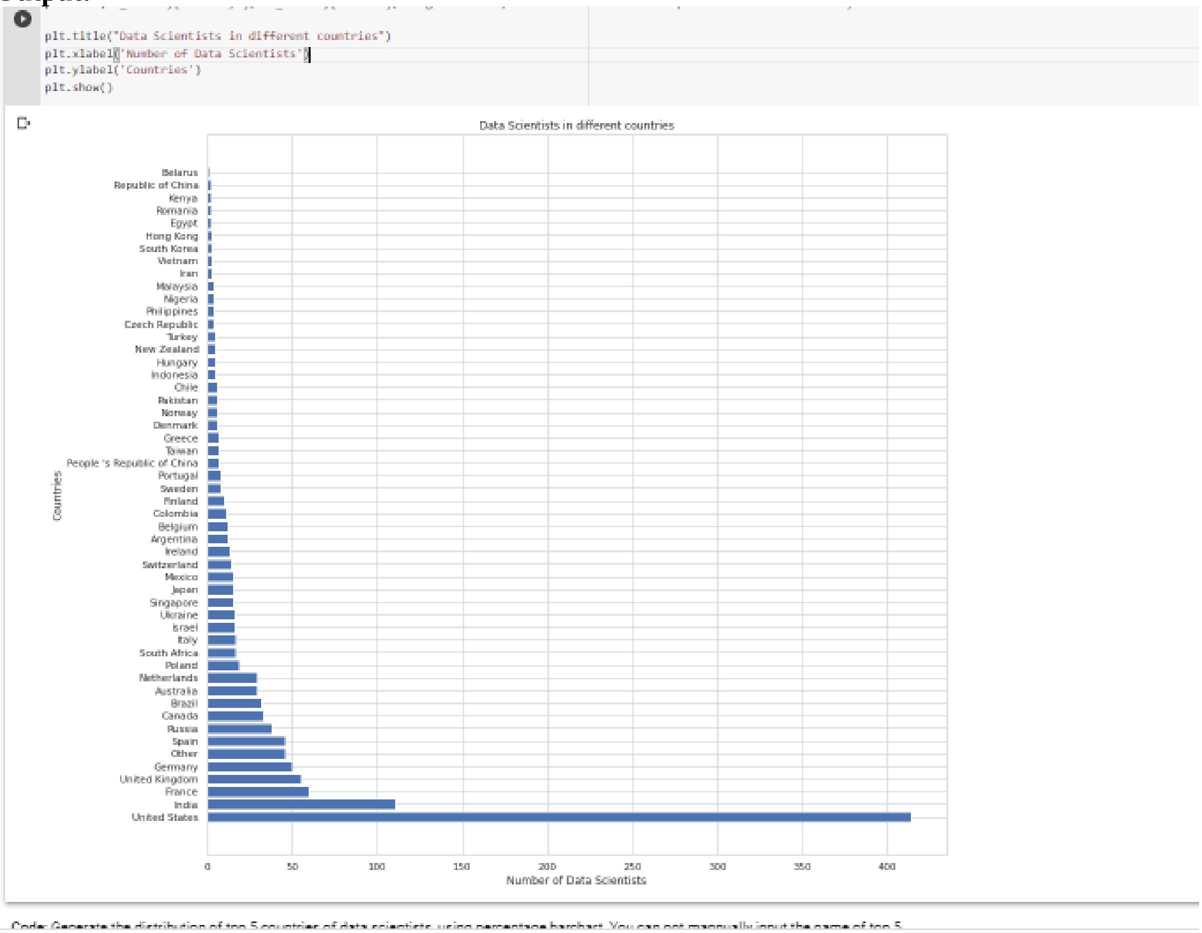
```
CountryData = df_demog_ds.groupby("Country").size().reset_index(name='Count')   # Find Country and number of employees
df_country = CountryData.sort_values('Count', ascending=False)                   # Descending sort
df_country.head()                                                                 # To find top 5 countries
```

	Country	Count
50	United States	414
17	India	111
12	France	60
49	United Kingdom	55
13	Germany	50

Source code:

```
# Show only one suitable plot of country: either Bar plot, count plot,
or Boxplot (possible or not?)

plt.figure(figsize=(15,15))
plt.barh(df_country['Country'], df_country['Count'], align='center')
         # Horizontal plot for better visibility

plt.title("Data Scientists in different countries")
plt.xlabel('Number of Data Scientists')
plt.ylabel('Countries')
plt.show()
```

Output:



```
plt.title("Data Scientists in different countries")
plt.xlabel('Number of Data Scientists')
plt.ylabel('Countries')
plt.show()
```

Code: Generate the distribution of top 5 countries of data scientists, using percentage barchart. You can not manually input the name of top 5

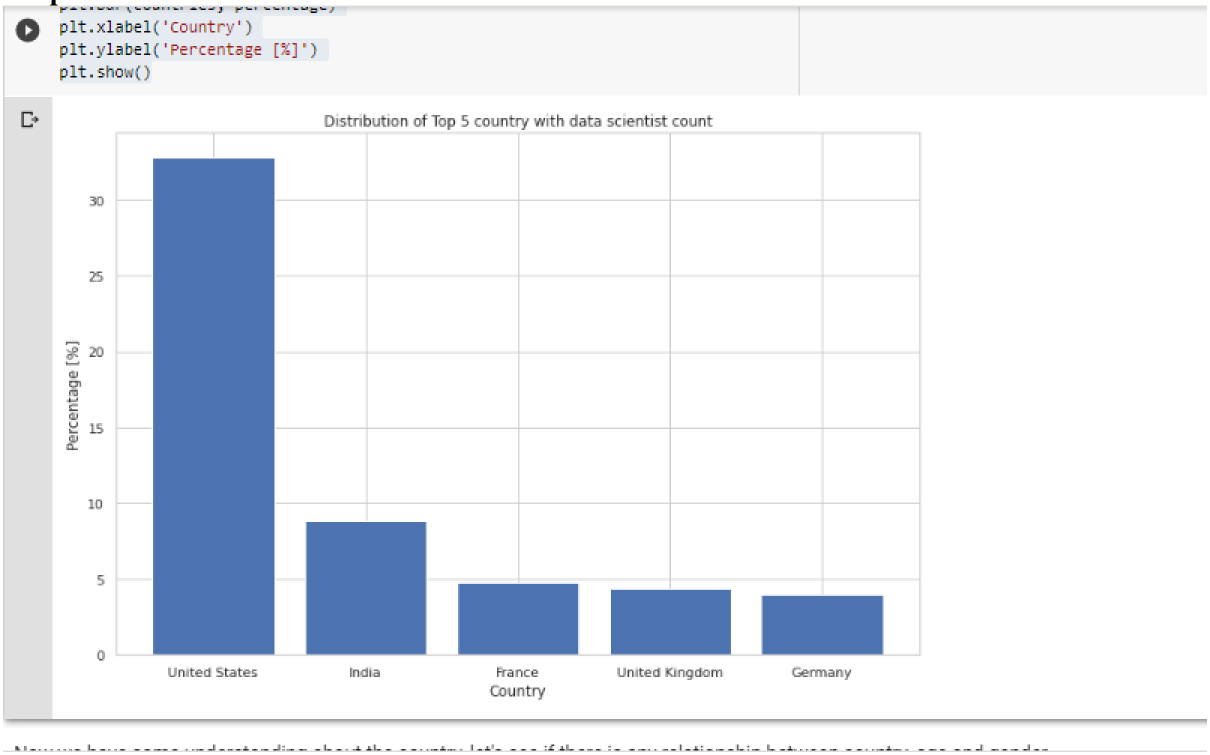
Task 1.3.D Barchart of Top 5

Source code:

```
#percentage
plt.figure(figsize=(12,8))
plt.title('Distribution of Top 5 country with data scientist count')
```

```
df_country_top5 = df_country.head()
         # Head() function to assign top 5 values

countries = df_country_top5['Country']
         # X values
percentage = (df_country_top5['Count'] / totalCount) * 100
         # Y values

plt.bar(countries, percentage)
plt.xlabel('Country')
plt.ylabel('Percentage [%]')
plt.show()
```

Output:



```
plt.bar(countries, percentage)
plt.xlabel('Country')
plt.ylabel('Percentage [%]')
plt.show()
```

Distribution of Top 5 country with data scientist count

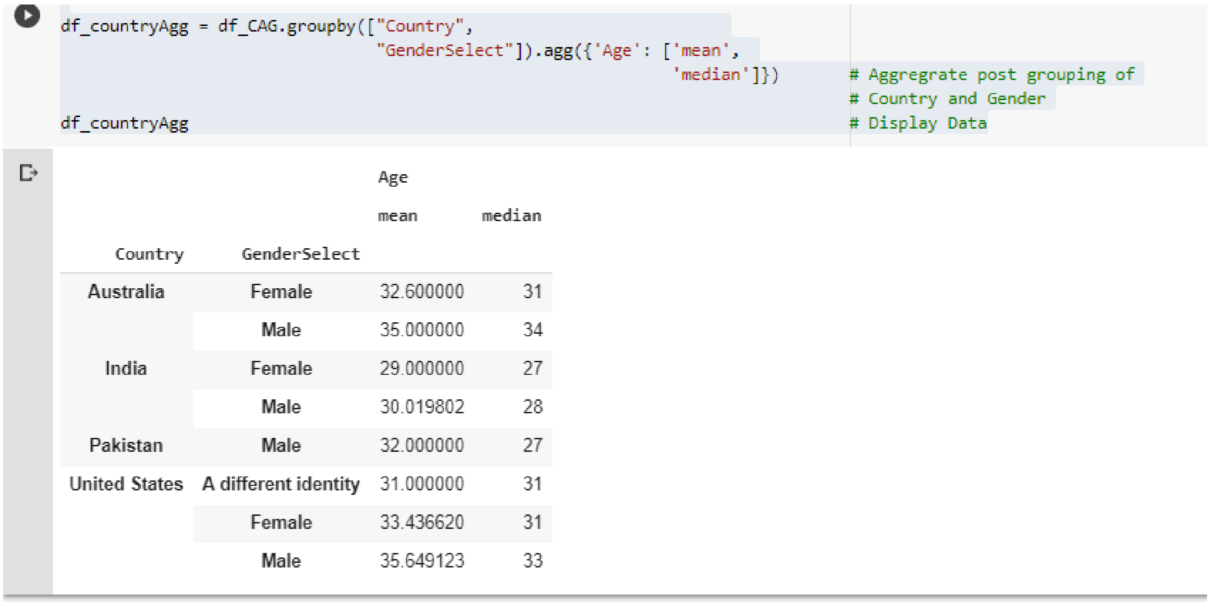
Now we have some understanding about the country, let's see if there is any relationship between country, age and gender.

Task 1.3.E Gender Age Info for 4 Countries

Source code:

```
df_CAG = df_demog_ds.loc[ df_demog_ds['Country']
                                      .isin(['United States', 'India',
                                             'Australia', 'Pakistan'])]
         # Filter required data

df_countryAgg = df_CAG.groupby(["Country",
                                 "GenderSelect"]).agg({'Age': ['mean',
                                                                'median']
})        # Aggregrate post grouping of

          # Country and Gender
df_countryAgg
          # Display Data
```

Output:



```
df_countryAgg = df_CAG.groupby(["Country",
                                "GenderSelect"]).agg({'Age': ['mean',
                                                               'median']})       # Aggregrate post grouping of
                                                                                 # Country and Gender
df_countryAgg                                                                    # Display Data
```

		Age	
		mean	median
Country	GenderSelect		
Australia	Female	32.600000	31
	Male	35.000000	34
India	Female	29.000000	27
	Male	30.019802	28
Pakistan	Male	32.000000	27
United States	A different identity	31.000000	31
	Female	33.436620	31
	Male	35.649123	33

Data Exploration : Part 2

Task 2.1.A Token Extraction Code

Source code:

```
lower = []
for item in df_text['job_description']:
    lower.append(item.lower())
         # lowercase description to 'list'
Rawdata = ' '.join([str(elem) for elem in lower])
         # Convert list to string
tokens = []

stop_words = set(stopwords.words('english'))
         # Load English stopwords
tokenizer = RegexpTokenizer(r"\w+(?:[-
']\w+)?")                                  # Use Regular expressions

def tokenizeLowerData (lower):
         # Custom Tokenizer Function
    tokens = tokenizer.tokenize(lower)
         # Tokenized data
    filtered_tokens = [token for token in tokens if token not in stop_w
ords]    # Exclude stop_words
    return filtered_tokens
```

Task 2.1.B Word List > 6000

Source code:

```
# Your Code
# find top common words with document frequencies > 6000
# you may use function FreqDist() and sort()


frequent_words = FreqDist(sorted(tokenizeLowerData(Rawdata)))
         # Find frequency of distinct words

freq6000 = []

for word, frequency in sorted(frequent_words.most_common()):
    if(frequency > 6000) :
      freq6000.append((word, frequency))
         # append words in list if freq > 6000
freq6000
         # Show list with words of freq > 6000
```

Output:

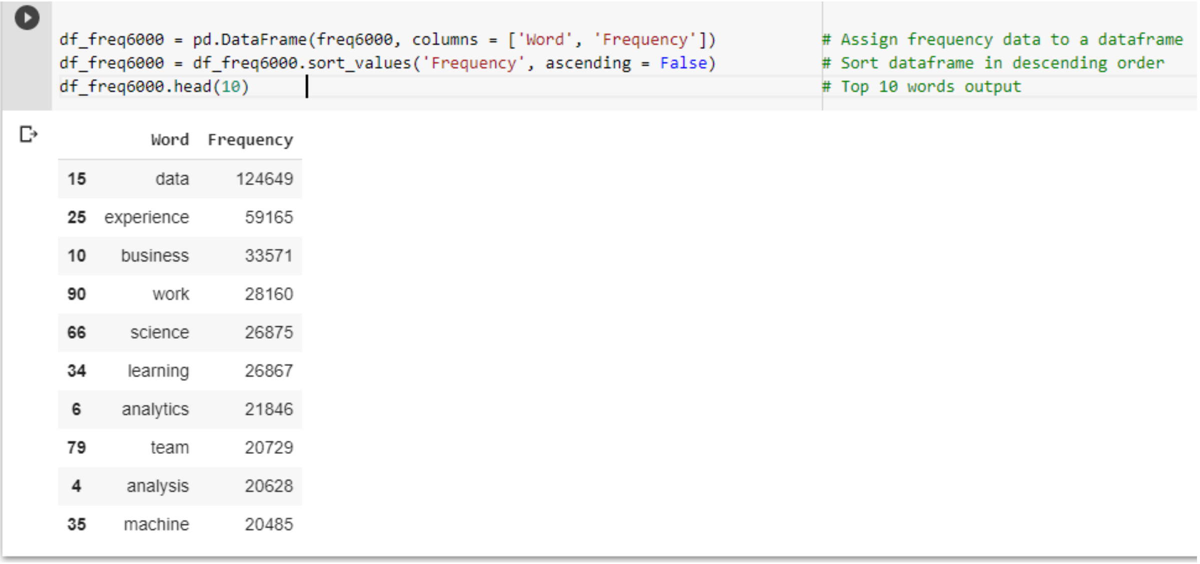
```
[('ability', 15686),
 ('across', 7189),
 ('advanced', 10627),
 ('algorithms', 9070),
 ('analysis', 20628),
 ('analytical', 8872),
 ('analytics', 21846),
 ('apply', 6203),
 ('big', 6626),
 ('build', 8212),
 ('business', 33571),
 ('company', 8999),
 ('complex', 8938),
 ('computer', 9676),
 ('customer', 6852),
 ('data', 124649),
 ('degree', 11338),
 ('design', 8759),
 ('develop', 11548),
 ('development', 12751),
 ('e', 6808),
 ('employment', 6696),
 ('engineering', 10141),
 ('environment', 8551),
 ('etc', 8308),
 ('experience', 59165),
 ('field', 7453),
 ('help', 7716),
 ('including', 10842),
 ('information', 11852),
 ('insights', 8911),
 ('job', 12292),
 ('knowledge', 13232),
 ('large', 7548),
 ('learning', 26867),
 ('machine', 20485),
 ('management', 9949),
 ('methods', 7110),
 ('modeling', 11045),
 ('models', 16559),
 ('must', 6196),
 ('new', 12688),
 ('one', 6038),
 ('opportunities', 6064),
 ('opportunity', 9432),
 ('people', 7561),
 ('position', 9341),
 ('predictive', 8202),
 ('preferred', 8005),
 ('problems', 9193),
 ('product', 8096),
 ('products', 6900),
 ('programming', 6649),
 ('projects', 7766),
 ('provide', 7169),
 ('python', 11955),
 ('qualifications', 8274),
 ('quantitative', 6490),
 ('r', 9336),
 ('related', 9236),
 ('required', 11028),
 ('requirements', 8057),
 ('research', 12208),
 ('responsibilities', 6995),
 ('results', 6354),
 ('role', 7287),
 ('science', 26875),
 ('scientist', 16364),
 ('services', 7849),
 ('skills', 19819),
 ('software', 8367),
 ('solutions', 15122),
 ('sql', 8145),
 ('statistical', 14657),
 ('statistics', 10254),
 ('status', 8348),
 ('strong', 11316),
 ('support', 9412),
 ('systems', 8475),
 ('team', 20729),
 ('teams', 7882),
 ('technical', 10683),
 ('techniques', 11555),
 ('technology', 8437),
 ('time', 6592),
 ('tools', 12777),
 ('understanding', 6739),
 ('us', 6999),
 ('use', 7574),
 ('using', 12635),
 ('work', 28160),
 ('working', 13382),
 ('years', 16235)]
```

Task 2.1.C Top 10 Words

Source code:

```
# Your Code to sort and display the top 10 high fequency words in 'freq
6000'


df_freq6000 = pd.DataFrame(freq6000, columns = ['Word', 'Frequency'])
         # Assign frequency data to a dataframe
df_freq6000 = df_freq6000.sort_values('Frequency', ascending = False)
         # Sort dataframe in descending order
df_freq6000.head(10)
         # Top 10 words output
```

Output:



```
df_freq6000 = pd.DataFrame(freq6000, columns = ['Word', 'Frequency'])    # Assign frequency data to a dataframe
df_freq6000 = df_freq6000.sort_values('Frequency', ascending = False)     # Sort dataframe in descending order
df_freq6000.head(10)                                                      # Top 10 words output
```

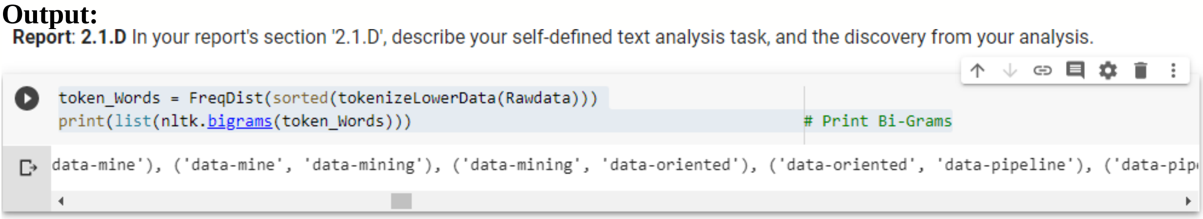
	Word	Frequency
15	data	124649
25	experience	59165
10	business	33571
90	work	28160
66	science	26875
34	learning	26867
6	analytics	21846
79	team	20729
4	analysis	20628
35	machine	20485

Task 2.1.D Text Analysis Code with Comments

Source code:

```
token_Words = FreqDist(sorted(tokenizeLowerData(Rawdata)))
print(list(nltk.bigrams(token_Words)))
         # Print Bi-Grams
```

Output:



Report: **2.1.D** In your report's section '2.1.D', describe your self-defined text analysis task, and the discovery from your analysis.

```
token_Words = FreqDist(sorted(tokenizeLowerData(Rawdata)))
print(list(nltk.bigrams(token_Words)))                          # Print Bi-Grams
```

```
data-mine'), ('data-mine', 'data-mining'), ('data-mining', 'data-oriented'), ('data-oriented', 'data-pipeline'), ('data-pip
```

Usability:

This result can be used in statistical findings on the frequency of such pairs in each text. That will corelate to the general sentiment of the descriptions present in the body of the text.

References

Pandas.pydata.org, 2020, *Pandas - Python Data Analysis Library*, retrieved 18 April 2020, <https://pandas.pydata.org/>

Matplotlib.org, 2020, *Matplotlib.Pyplot — Matplotlib 3.1.2 Documentation*, retrieved 17 April 2020, <https://matplotlib.org/3.1.1/api/pyplot_summary.html>

Docs.python.or,. 2020, *Tokenize — Tokenizer For Python Source — Python 3.8.2 Documentation*, retrieved 18 April 2020 , <https://docs.python.org/3/library/tokenize.html#tokenize.tokenize>

Nltk.org, 2020, *Nltk.Tokenize.Regexp — NLTK 3.5 Documentation*, retrieved 18 April 2020, <https://www.nltk.org/_modules/nltk/tokenize/regexp.html>

Nltk.org, 2020, *Nltk Package — NLTK 3.5 Documentation*, retrieved 18 April 2020, <https://www.nltk.org/api/nltk.html?highlight=freqdist>