Phase 4: Data Exploration

Team Name: Deep Diver's

Team Members:

Vaishnavi Hemant Salaskar vsala2@unh.newhaven.edu

Vedant Chidgopkar vchid2@unh.newhaven.edu

Riddhi Joshi rjosh5@unh.newhaven.edu

Introduce your research question and selected data set

Today mental health is becoming a more common problem. However, evaluation of mental well-being is extremely important to understanding and providing therapeutic solutions. Diagnostics are complicated tasks and misdiagnosis can result in serious problems if a mental disorder is not properly detected. Can we recognize mental health issues accurately by using data mining techniques?

The data has been collected from Kaggle by Open Sourcing Mental Illness, LTD. Survey data about mental health attitudes are included in this dataset. Which then has been analyzed and pre-processed. The data contains different labels such as age, gender, country, self-employee, family history, work interference, seek help, etc. For better prediction, we have label encoded the data

List of Exploration Techniques

- Count
- Count Plot
- Cross Tab
- Factor Plot
- Histogram
- Cat Plot
- Box Plot
- Dis Plot
- Pair Plot
- Scatter Plot

Description of data explorations

We have done the Data Exploration in following ways,

1. **Data Understanding.**

We have used Jupyter Notebook a Python framework for data exploration. In that we have used libraries like pandas, seaborn, NumPy and matplotlib for data visualization and cleaning. The first step was to upload the .CVS file and view that in Jupyter.

In [570]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Understanding the Data

In [571]:
```
data = pd.read_csv('survey.csv')
```

In [572]:
```
data.head()
```

Out[572]:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_health_consequence
0	8/27/2014 11:29	37	Female	United States	IL	NaN	No	Yes	Often	25-Jun	...	Somewhat easy	No
1	8/27/2014 11:29	44	M	United States	IN	NaN	No	No	Rarely	More than 1000	...	Don't know	Maybe
2	8/27/2014 11:29	32	Male	Canada	NaN	NaN	No	No	Rarely	25-Jun	...	Somewhat difficult	No
3	8/27/2014 11:29	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100	...	Somewhat difficult	Yes
4	8/27/2014 11:30	31	Male	United States	TX	NaN	No	No	Never	100-500	...	Don't know	No

5 rows × 27 columns

Before cleaning the data there were 27 columns and 1259 rows. With null values and missing values.

In [575]:
```
data.nunique()
```

Out[575]:
```
Timestamp                    884
Age                           53
Gender                        49
Country                       48
state                         45
self_employed                  2
family_history                 2
treatment                      2
work_interfere                 4
no_employees                   6
remote_work                    2
tech_company                   2
benefits                       3
care_options                   3
wellness_program               3
seek_help                      3
anonymity                      3
leave                          5
mental_health_consequence      3
phys_health_consequence        3
coworkers                      3
supervisor                     3
mental_health_interview        3
phys_health_interview          3
mental_vs_physical             3
obs_consequence                2
comments                     160
dtype: int64
```

nunique() function return number of unique elements in the object. It returns a scalar value which is the count of all the unique values in the Index. By default the NaN values are not included in the count.

In [576]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Timestamp                  1259 non-null   object
 1   Age                        1259 non-null   int64
 2   Gender                     1259 non-null   object
 3   Country                    1259 non-null   object
 4   state                      744 non-null    object
 5   self_employed              1241 non-null   object
 6   family_history             1259 non-null   object
 7   treatment                  1259 non-null   object
 8   work_interfere             995 non-null    object
 9   no_employees               1259 non-null   object
 10  remote_work                1259 non-null   object
 11  tech_company               1259 non-null   object
 12  benefits                   1259 non-null   object
 13  care_options               1259 non-null   object
 14  wellness_program           1259 non-null   object
 15  seek_help                  1259 non-null   object
 16  anonymity                  1259 non-null   object
 17  leave                      1259 non-null   object
 18  mental_health_consequence  1259 non-null   object
 19  phys_health_consequence    1259 non-null   object
 20  coworkers                  1259 non-null   object
 21  supervisor                 1259 non-null   object
 22  mental_health_interview    1259 non-null   object
 23  phys_health_interview      1259 non-null   object
 24  mental_vs_physical         1259 non-null   object
 25  obs_consequence            1259 non-null   object
 26  comments                   164 non-null    object
dtypes: int64(1), object(26)
memory usage: 265.7+ KB
```

Also, the mean, median of data was not countable because of unstructured data.

In [577]:
```
data.describe()
```

Out[577]:

	Age
count	1.259000e+03
mean	7.942815e+07
std	2.818299e+09
min	-1.726000e+03
25%	2.700000e+01
50%	3.100000e+01
75%	3.600000e+01
max	1.000000e+11

In [578]:
```
type(data)
```

Out[578]:
```
pandas.core.frame.DataFrame
```

In this step we gained more knowledge about data including count number of row and columns, missing values, null values using mentioned commands. Depending upon these we come to the conclusion that data cleaning is necessary with available data to get accurate result.

2. Data Cleaning

In this step of data exploration cleaning of data is carried out. We started with removing unnecessary data columns. In order, to make data less noisy. As shown below.

In [580]:
```
data = data.drop(['state'], axis=1)
```

In [581]:
```
data.head()
```

Out[581]:

	Timestamp	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	no_employees	remote_work	...	leave	mental_health_conse
0	8/27/2014 11:29	37	Female	United States	NaN	No	Yes	Often	25-Jun	No	...	Somewhat easy	
1	8/27/2014 11:29	44	M	United States	NaN	No	No	Rarely	More than 1000	No	...	Don't know	
2	8/27/2014 11:29	32	Male	Canada	NaN	No	No	Rarely	25-Jun	No	...	Somewhat difficult	
3	8/27/2014 11:29	31	Male	United Kingdom	NaN	Yes	Yes	Often	26-100	No	...	Somewhat difficult	
4	8/27/2014 11:30	31	Male	United States	NaN	No	No	Never	100-500	Yes	...	Don't know	

5 rows × 26 columns

In [582]:
```
data = data.drop(['no_employees'], axis=1)
data = data.drop(['tech_company'], axis=1)
data = data.drop(['benefits'], axis=1)
data = data.drop(['care_options'], axis=1)
data = data.drop(['wellness_program'], axis=1)
data = data.drop(['anonymity'], axis=1)
data = data.drop(['phys_health_consequence'], axis=1)
data = data.drop(['supervisor'], axis=1)
data = data.drop(['mental_health_interview'], axis=1)
data = data.drop(['phys_health_interview'], axis=1)
data = data.drop(['mental_vs_physical'], axis=1)
data = data.drop(['obs_consequence'], axis=1)
data = data.drop(['comments'], axis=1)
```


Moreover, to handle null value forward fill function is used.

In [586]:
```
data = data.fillna(method = "ffill")
```

In [587]:
```
data
```

Out[587]:

	Timestamp	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	remote_work	seek_help	leave	mental_health_consequer
0	8/27/2014 11:29	37	Female	United States	NaN	No	Yes	Often	No	Yes	Somewhat easy	
1	8/27/2014 11:29	44	M	United States	NaN	No	No	Rarely	No	Don't know	Don't know	Ma
2	8/27/2014 11:29	32	Male	Canada	NaN	No	No	Rarely	No	No	Somewhat difficult	
3	8/27/2014 11:29	31	Male	United Kingdom	NaN	Yes	Yes	Often	No	No	Somewhat difficult	
4	8/27/2014 11:30	31	Male	United States	NaN	No	No	Never	Yes	Don't know	Don't know	
...	...	...	...	...	...	...	...	...	...	...	...	
1254	9/12/2015 11:17	26	male	United Kingdom	No	No	Yes	Rarely	No	No	Somewhat easy	
1255	9/26/2015 1:07	32	Male	United States	No	Yes	Yes	Often	Yes	No	Somewhat difficult	
1256	11/7/2015 12:36	34	male	United States	No	Yes	Yes	Sometimes	No	No	Somewhat difficult	
1257	11/30/2015 21:25	46	f	United States	No	No	No	Sometimes	Yes	No	Don't know	
1258	2/1/2016 23:04	25	Male	United States	No	Yes	Yes	Sometimes	No	No	Don't know	Ma

1259 rows × 13 columns

There were multiple values for same type of value we converted that into single value usimg replace function. The execution is as showed below.

```
In [589]: data['Gender'].nunique()
Out[589]: 49

In [590]: print(data['Gender'].unique())

['Female' 'M' 'Male' 'male' 'female' 'm' 'Male-ish' 'maile' 'Trans-female'
 'Cis Female' 'F' 'something kinda male?' 'Cis Male' 'Woman' 'f' 'Mal'
 'Male (CIS)' 'queer/she/they' 'non-binary' 'Femake' 'woman' 'Make' 'Nah'
 'All' 'Enby' 'fluid' 'Genderqueer' 'Female ' 'Androgyne' 'Agender'
 'cis-female/femme' 'Guy (-ish) ^_^' 'male leaning androgynous' 'Male '
 'Man' 'Trans woman' 'msle' 'Neuter' 'Female (trans)' 'queer'
 'Female (cis)' 'Mail' 'cis male' 'A little about you' 'Malr' 'p' 'femail'
 'Cis Man' 'ostensibly male, unsure what that really means']

In [591]: data['Gender']=data['Gender'].replace(['female', 'Trans-female', 'Cis Female', 'F', 'Woman', 'f', 'queer/she/they', 'Femake', 'wo

In [592]: data['Gender'].nunique()
Out[592]: 33

In [593]: Make', 'Guy (-ish) ^_^', 'male leaning androgynous', 'Male ', 'Male','Man','msle', 'Mail','cis male', 'Malr', 'Cis Man'], 'Male')

In [594]: data['Gender'].nunique()
Out[594]: 14

In [595]: print(data['Gender'].unique())

['Female' 'Male' 'non-binary' 'Nah' 'All' 'Enby' 'fluid' 'Genderqueer'
 'Androgyne' 'Agender' 'Neuter' 'A little about you' 'p'
 'ostensibly male, unsure what that really means']
```



```
In [608]: data = data[data.Gender != 'Nah']

In [609]: print(data['Gender'].unique())

['Male' 'Female' 'non-binary' 'All' 'Enby' 'fluid' 'Genderqueer'
 'Androgyne' 'Agender' 'Neuter' 'A little about you' 'p'
 'ostensibly male, unsure what that really means']

In [610]: data = data[data.Gender != 'non-binary']

In [611]: print(data['Gender'].unique())

['Male' 'Female' 'All' 'Enby' 'fluid' 'Genderqueer' 'Androgyne' 'Agender'
 'Neuter' 'A little about you' 'p'
 'ostensibly male, unsure what that really means']

In [612]: values = ['All','Enby','fluid','Genderqueer','Androgyne','Agender','Neuter','A little about you','p'
          ,'ostensibly male, unsure what that really means']

In [613]: data = data[data.Gender.isin(values) == False]

In [614]: print(data['Gender'].unique())

['Male' 'Female']
```

There were some consecutive null values in self_employed column which were dropped using dropna function in pandas.

```
In [600]: data=data.dropna()

In [601]: data
Out[601]:
```

	Timestamp	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	remote_work	seek_help	leave	mental_health_consequer
18	8/27/2014 11:34	46	Male	United States	Yes	Yes	No	Sometimes	Yes	Don't know	Very easy	
19	8/27/2014 11:35	36	Male	France	Yes	Yes	No	Sometimes	Yes	No	Somewhat easy	
20	8/27/2014 11:35	29	Male	United States	No	Yes	Yes	Sometimes	No	No	Somewhat difficult	Ma
21	8/27/2014 11:35	31	Male	United States	Yes	No	No	Never	Yes	No	Somewhat difficult	
22	8/27/2014 11:35	46	Male	United States	No	No	Yes	Often	Yes	No	Don't know	Ma
...	...	...	...	...	...	...	...	...	...	...	...	
1254	9/12/2015 11:17	26	Male	United Kingdom	No	No	Yes	Rarely	No	No	Somewhat easy	
1255	9/26/2015 1:07	32	Male	United States	No	Yes	Yes	Often	Yes	No	Somewhat difficult	
1256	11/7/2015 12:36	34	Male	United States	No	Yes	Yes	Sometimes	No	No	Somewhat difficult	
1257	11/30/2015 21:25	46	Female	United States	No	No	No	Sometimes	Yes	No	Don't know	
1258	2/1/2016 23:04	25	Male	United States	No	Yes	Yes	Sometimes	No	No	Don't know	Ma

1241 rows × 13 columns

In Age column the range of age was out of bounds to fix that we applied drop method in different form. The operation is shown in below figure,

```
In [616]: print(data['Age'].unique())

[   46    36    29    31    41    33    35    34    37    32    30    42
    40    27    38    50    24    18    28    26    22    44    23    19
    25    39    45    21   -29    43    56    60    54   329    55    48
    20    57    58    47    62    51    65    49 -1726     5    53    61
    11    72]

In [617]: values = [-29, 329, -1726, 5, 72, 11]

In [618]: data = data[data.Age.isin(values) == False]

In [619]: print(data['Age'].unique())

[46 36 29 31 41 33 35 34 37 32 30 42 40 27 38 50 24 18 28 26 22 44 23 19
 25 39 45 21 43 56 60 54 55 48 20 57 58 47 62 51 65 49 53 61]

In [620]: data
Out[620]:
```

	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	remote_work	seek_help	mental_health_consequence	coworkers
18	46	Male	United States	Yes	Yes	No	Sometimes	Yes	Don't know	No	Yes
19	36	Male	France	Yes	Yes	No	Sometimes	Yes	No	No	Some of them
20	29	Male	United States	No	Yes	Yes	Sometimes	No	No	Maybe	Some of them
21	31	Male	United States	Yes	No	No	Never	Yes	No	No	Some of them
22	46	Male	United States	No	No	Yes	Often	Yes	No	Maybe	Some of them
...	...	...	...	...	...	...	...	...	...	...	...
1254	26	Male	United Kingdom	No	No	Yes	Rarely	No	No	No	Some of them
1255	32	Male	United States	No	Yes	Yes	Often	Yes	No	No	Some of them
1256	34	Male	United States	No	Yes	Yes	Sometimes	No	No	Yes	No
1257	46	Female	United States	No	No	No	Sometimes	Yes	No	Yes	No
1258	25	Male	United States	No	Yes	Yes	Sometimes	No	No	Maybe	Some of them

1223 rows × 11 columns

In this way we cleaned the rest of the data to make it more accurate and useful for further use.

3. Data visualization:

Following the cleaning, the next stage is to analyse and visualize the variables, which can be done by establishing a relationship between them. Here we have tried show relationship between different variable in our data.

Relation Between the Variables

```
In [621]: data['Country'].value_counts()

Out[621]: United States             731
          United Kingdom            177
          Canada                     68
          Germany                    44
          Netherlands                27
          Ireland                    27
          Australia                  21
          France                     13
          India                      10
          New Zealand                 8
          Poland                      7
          Switzerland                 7
          Sweden                      7
          Italy                       7
          South Africa                6
          Belgium                     6
          Brazil                      6
          Israel                      5
          Singapore                   4
          Austria                     3
          Bulgaria                    3
          Mexico                      3
          Russia                      3
          Finland                     3
          Denmark                     2
          Greece                      2
          Colombia                    2
          Portugal                    2
          Croatia                     2
          Moldova                     1
          Georgia                     1
          China                       1
          Thailand                    1
          Czech Republic              1
          Norway                      1
          Latvia                      1
          Nigeria                     1
          Japan                       1
          Hungary                     1
          Bosnia and Herzegovina      1
          Uruguay                     1
          Spain                       1
          Romania                     1
          Costa Rica                  1
          Slovenia                    1
          Philippines                 1
          Name: Country, dtype: int64
```

In the below given we are trying to show count of variables.

```
In [624]: data[['self_employed','work_interfere']].value_counts()

Out[624]: self_employed  work_interfere
          No             Sometimes         511
                         Never             237
                         Rarely            188
                         Often             146
          Yes            Sometimes          68
                         Often              29
                         Never              23
                         Rarely             21
          dtype: int64
```

Using describe we can get mean, standard deviation and min, max value for age variable.

In [623]:
```
data.describe()
```

Out[623]:

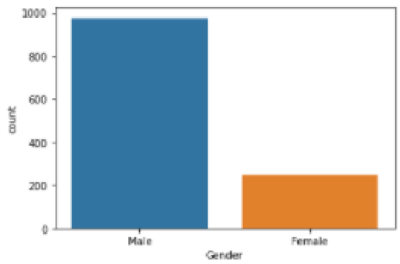
	Age
count	1223.000000
mean	32.052330
std	7.226896
min	18.000000
25%	27.000000
50%	31.000000
75%	36.000000
max	65.000000

To know number of count for male and female we have done plotting by count plot.

In [625]:
```
sns.countplot(data['Gender'])
```

```
C:\Users\VC\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[625]: <AxesSubplot:xlabel='Gender', ylabel='count'>



In [626]:
```
data['Gender'].value_counts()
```

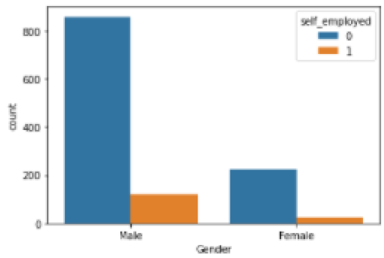
Out[626]:
```
Male      976
Female    247
Name: Gender, dtype: int64
```

Count plot for plotting the count of self-employed male and female.

In [649]:
```
sns.countplot('Gender', hue="self_employed", data=data)
```

```
C:\Users\VC\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[649]: <AxesSubplot:xlabel='Gender', ylabel='count'>



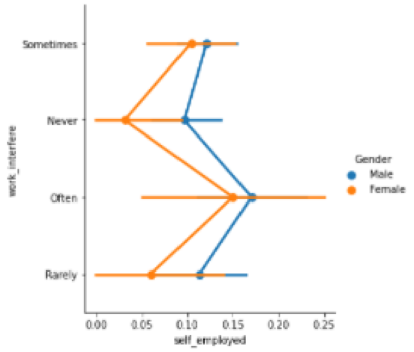
In [650]:
```
pd.crosstab(data['Gender'], data['self_employed'], margins=True)
```

Out[650]:

self_employed	0	1	All
Gender			
Female	224	23	247
Male	858	118	976
All	1082	141	1223

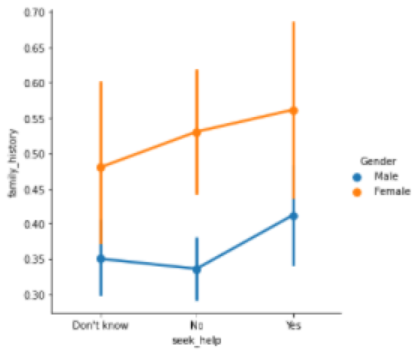
We have used factorplot to find relation between work_interfere and gender data.

```
In [663]: sns.factorplot('self_employed', 'work_interfere', hue='Gender', data=data)

C:\Users\VC\anaconda3\lib\site-packages\seaborn\categorical.py:3717: UserWarning: The `factorplot` function has been renamed to
`catplot`. The original name will be removed in a future release. Please update your code. Note that the default `kind` in `fac
torplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\Users\VC\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args:
x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(

Out[663]: <seaborn.axisgrid.FacetGrid at 0x1c877ee3f40>
```



Factorplot to show relation between family_history and seek_help to find out connetcion for the mental health history in family and treatment.

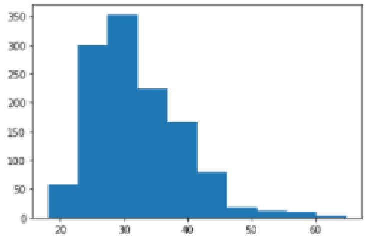
```
In [666]: sns.factorplot('seek_help', 'family_history', hue='Gender', data=data)

C:\Users\VC\anaconda3\lib\site-packages\seaborn\categorical.py:3717: UserWarning: The `factorplot` function has been renamed to
`catplot`. The original name will be removed in a future release. Please update your code. Note that the default `kind` in `fac
torplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\Users\VC\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args:
x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(

Out[666]: <seaborn.axisgrid.FacetGrid at 0x1c876c7a4f0>
```



Ploting of age using histogram to visulize the range of age in taken dataset.

```
In [668]: plt.hist(data['Age'])
```

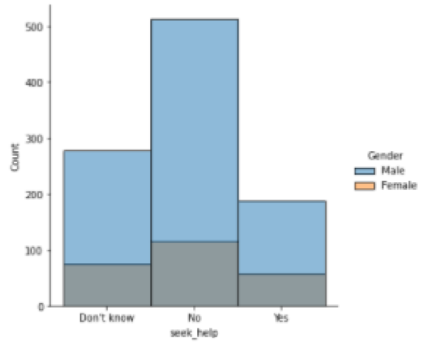
```
Out[668]: (array([ 58., 299., 353., 224., 166.,  80.,  18.,  12.,  10.,   3.]),
           array([18. , 22.7, 27.4, 32.1, 36.8, 41.5, 46.2, 50.9, 55.6, 60.3, 65. ]),
           <BarContainer object of 10 artists>)
```



Now ploting data to show relation between in seek_help and Gender using displot.In plotting we can clearly find the count of female and male have seek help.

```
In [670]: sns.displot(data, x="seek_help", hue="Gender")
```

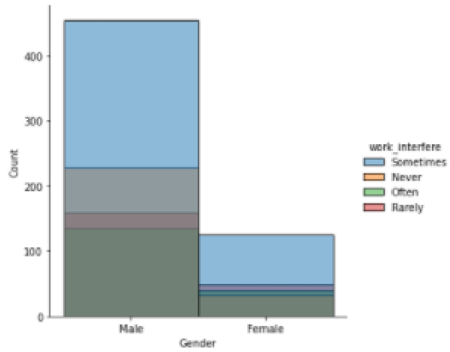
```
Out[670]: <seaborn.axisgrid.FacetGrid at 0x1c876bf6bb0>
```



Again ploting to describe the relation between work_interfere and Gender. Displot shows the ration of the work_interfere for Gender.

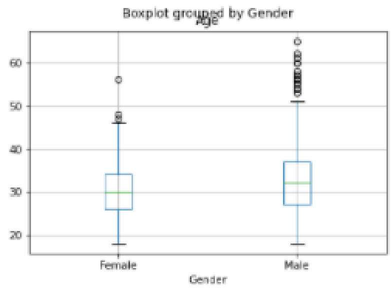
```
In [671]: sns.displot(data, x="Gender", hue="work_interfere")
```

```
Out[671]: <seaborn.axisgrid.FacetGrid at 0x1c876b969a0>
```



Boxplot to plot the count of the female and male the Gender and find relation between age.

In [682]:
```python
boxplot = data.boxplot(by='Gender',column=['Age'])
```



Here we have converted catogerial data values in binary values.In data forSelf emplyeed is given binary value 1 for No 0.

Converting Catogoricl Data to Binary Data

In [632]:
```python
dummy = pd.get_dummies(data['self_employed'])
```

In [633]:
```python
dummy.head()
```

Out[633]:

	No	Yes
18	0	1
19	0	1
20	1	0
21	0	1
22	1	0

In [634]:
```python
data= pd.concat((data, dummy), axis=1)
```

In [635]:
```python
data.head()
```

Out[635]:

	Age	Gender	Country	self employed	family history	treatment	work interfere	remote work	seek help	mental health consequence	coworkers	No	Yes
18	46	Male	United States	Yes	Yes	No	Sometimes	Yes	Don't know	No	Yes	0	1
19	36	Male	France	Yes	Yes	No	Sometimes	Yes	No	No	Some of them	0	1
20	29	Male	United States	No	Yes	Yes	Sometimes	No	No	Maybe	Some of them	1	0
21	31	Male	United States	Yes	No	No	Never	Yes	No	No	Some of them	0	1
22	46	Male	United States	No	No	Yes	Often	Yes	No	Maybe	Some of them	1	0

In [636]:
```python
data = data.drop(['self_employed'], axis=1)
data = data.drop(['No'], axis=1)
```

In [637]:
```python
data=data.rename(columns={'Yes': 'self_employed'})
```

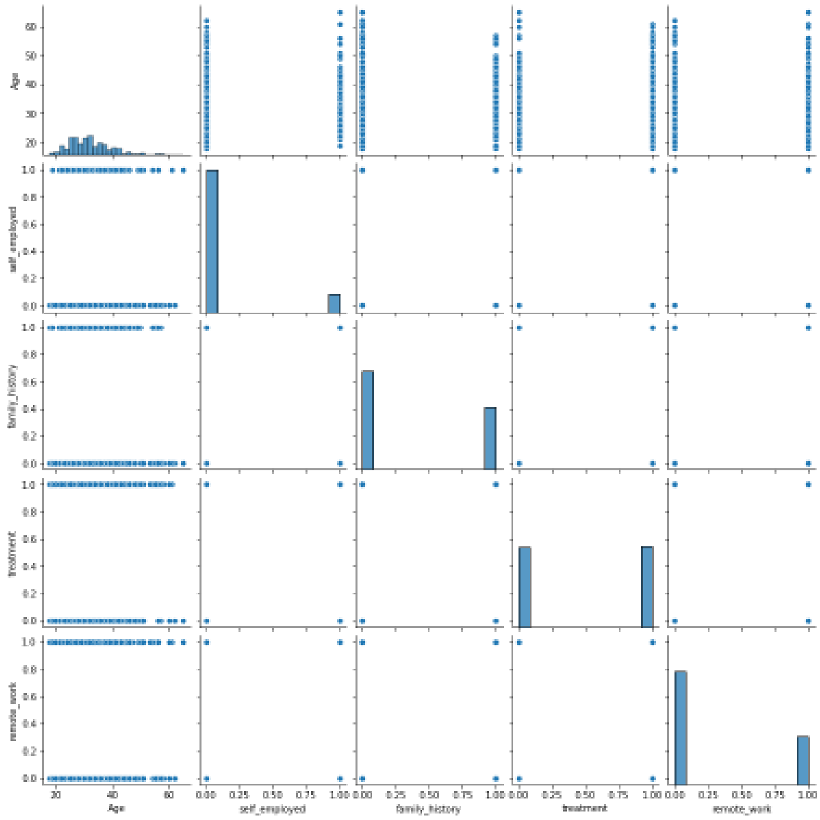
In [638]:
```python
data
```

Out[638]:

	Age	Gender	Country	family history	treatment	work interfere	remote work	seek help	mental health consequence	coworkers	self employed
18	46	Male	United States	Yes	No	Sometimes	Yes	Don't know	No	Yes	1
19	36	Male	France	Yes	No	Sometimes	Yes	No	No	Some of them	1
20	29	Male	United States	Yes	Yes	Sometimes	No	No	Maybe	Some of them	0
21	31	Male	United States	No	No	Never	Yes	No	No	Some of them	1
22	46	Male	United States	No	Yes	Often	Yes	No	Maybe	Some of them	0
...	...	...	...	...	...	...	...	...	...	...	...
1254	26	Male	United Kingdom	No	Yes	Rarely	No	No	No	Some of them	0
1258	32	Male	United States	Yes	Yes	Often	Yes	No	No	Some of them	0

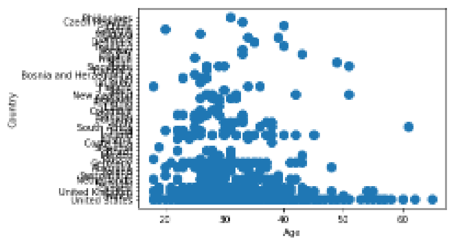
The pair plot, on the other hand, considers two variables, which can be continuous category or Boolean, and is a collection of plots for the variables in the dataset.

```
In [689]: sns.pairplot(data)

Out[689]: <seaborn.axisgrid.PairGrid at 0x1c87e42c048>
```



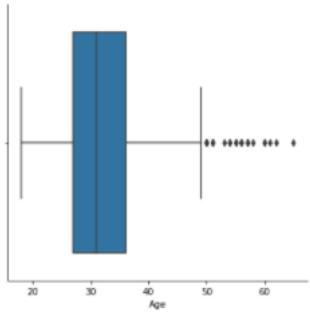
Next, plot data on the relationship between different numeric variables such as age column and one category data variable like country column. This goal can be achieved with the help of the scatter plot.

```
In [707]: data.plot.scatter(x='Age',y='Country', s=100);
```



The categorical plot, which visualizes the distribution of the variable throughout the dataset, was the final plot used to shows the data.

```
In [688]: sns.catplot(x='Age', kind='box', data=data)
Out[688]: <seaborn.axisgrid.FacetGrid at 0x1c87e657bb0>
```



GitHub repository link :

https://github.com/vsala2/DataMining