National College of Ireland

Configuration Manual

MSc Research Project
MSc In Data Analytics

Sonal Suryakant Puradkar
Student ID: X22130250

School of Computing
National College of Ireland

Supervisor: Arjun Chikkankod

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sonal Suryakant Puradkar
Student ID:	X22130250
Programme:	MSc In Data Analytics **Year:** 2023-2024.
Module:	MSc Research Project
Lecturer:	Arjun Chikkankod
Submission Due Date:	14 Dec 2023
Project Title:	Forecasting Global Mental Health Disorders : A Machine Learning Approach using Socioeconomic Indicators

Word Count:1137 **Page Count:**19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Sonal Suryakant Puradkar……

Date: 14 Dec 2023……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sonal Suryakant Puradkar
Student ID:X22130250

1 Introduction

The objective of the project is to find out if the countries where mental health disorders namely Anxiety Disorders, Bipolar Disorders, Eating Disorders, Schizophrenia and Depressive disorders will be highest can be predicted based on socio economic indicators and to see the forecasted values for 3 years. For this study we are taking country wise year wise data of the below socio economic indicators for the years 1960-2022:

Indicator name	Indicator Code
Adjusted net national income per capita (current US$)	NY.ADJ.NNTY.PC.CD
Consumer price index (2010 = 100)	FP.CPI.TOTL
Inflation, consumer prices (annual %)	FP.CPI.TOTL.ZG
Employment in industry (% of total employment) (modeled ILO estimate)	SL.IND.EMPL.ZS
Employment in services (% of total employment) (modeled ILO estimate)	SL.SRV.EMPL.ZS
Self-employed, total (% of total employment) (modeled ILO estimate)	SL.EMP.SELF.ZS
Proportion of people living below 50 percent of median income (%)	SI.DST.50MD
Unemployment with advanced education (% of total labor force with advanced education)	SL.UEM.ADVN.ZS
Unemployment, total (% of total labor force) (modeled ILO estimate)	SL.UEM.TOTL.ZS
New businesses registered (number)	IC.BUS.NREG
Multidimensional poverty index (scale 0-1)	SI.POV.MDIM.XQ

.
For mental health disorders we are taking country wise year wise Disability Adjusted Life Years (DALYs) data from 1990 -2019.

To further find out the workplace mental health affecting factors this study creates visualizations for the factors affecting workplace mental health.
Python is used for data pre-processing and data modelling and Power BI is used for visualization in this study.

2 Data Gathering

Socio economic indicators data has been downloaded from https://databank.worldbank.org/source/world-development-indicators and mental health DALY data has been downloaded from https://ourworldindata.org/mental-health . Workplace mental health data has been collected from survey on Kaggle

https://www.kaggle.com/code/aditimulye/mental-health-at-workplace/input. All the data has been gathered and kept into the root folder of the project.

3 Data Pre Processing

The data pre processing is done in python. It starts with reading data files from downloaded CSV files. In section 1 of the notebook CrispDM_Mental_Health_forecasting.ipynb all the libraries are imported and files are read into dataframes as shown below

1. Importing Libraries and Files

1.1 Importing Libraries

In [1]:
```
import requests
import pprint
import json
import pandas as pd
import os as os1
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required
for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

1.2 Importing Data Files and checking first 10 rows

In [2]:
```
metadata_df = pd.read_csv('Indicators_Metadata.csv')
Indicator_data_df = pd.read_csv('Indicators_Data.csv')
Mental_data_df = pd.read_csv('Mental_Health_Data.csv')
```

In [3]:
```
Indicator_data_df.head(10)
```

Out[3]:

	Country Name	Country Code	Series Name	Series Code	1960 [YR1960]	1961 [YR1961]	1962 [YR1962]	1963 [YR1963]	1964 [YR1964]	1965 [YR1965]	...	2013 [YR2013]	2014 [YR2014]	[YR
0	Afghanistan	AFG	Adjusted net national income per capita (curre...	NY.ADJ.NNTY.PC.CD	..	..	..	..	..	..	...	586.6136889	575.6807964	526.93
1	Afghanistan	AFG	Consumer price index (2010 = 100)	FP.CPI.TOTL	..	..	..	..	..	..	...	127.7952231	133.7683667	132.88
2	Afghanistan	AFG	Inflation, consumer prices (annual %)	FP.CPI.TOTL.ZG	..	..	..	..	..	..	...	7.385771784	4.673996035	-0.6617
3	Afghanistan	AFG	Employment in industry (% of total employment)...	SL.IND.EMPL.ZS	..	..	..	..	..	..	...	19.53853	22.04671	20.
4	Afghanistan	AFG	Employment in services (% of total employment)...	SL.SRV.EMPL.ZS	..	..	..	..	..	..	...	32.82068	33.15474	34.
5	Afghanistan	AFG	Self-employed, total (% of total employment) (...	SL.EMP.SELF.ZS	..	..	..	..	..	..	...	87.09895	86.01729	84.
			Proportion of											

:

In [4]:
```
metadata_df.head(10)
```

Out[4]:

	Series Name	Series Code
0	Adjusted net national income per capita (curre...	NY.ADJ.NNTY.PC.CD
1	Consumer price index (2010 = 100)	FP.CPI.TOTL
2	Inflation, consumer prices (annual %)	FP.CPI.TOTL.ZG
3	Employment in industry (% of total employment)...	SL.IND.EMPL.ZS
4	Employment in services (% of total employment)...	SL.SRV.EMPL.ZS
5	Self-employed, total (% of total employment) (...	SL.EMP.SELF.ZS
6	Proportion of people living below 50 percent o...	SI.DST.50MD
7	Unemployment with advanced education (% of tot...	SL.UEM.ADVN.ZS
8	Unemployment, total (% of total labor force) (...	SL.UEM.TOTL.ZS
9	New businesses registered (number)	IC.BUS.NREG

In [5]:
```
Mental_data_df.head(10)
```

Out[5]:

	Entity	Code	Year	DALYs from depressive disorders per 100,000 people in, both sexes aged age-standardized	DALYs from schizophrenia per 100,000 people in, both sexes aged age-standardized	DALYs from bipolar disorder per 100,000 people in, both sexes aged age-standardized	DALYs from eating disorders per 100,000 people in, both sexes aged age-standardized	DALYs from anxiety disorders per 100,000 people in, both sexes aged age-standardized
0	Afghanistan	AFG	1990	895.22565	138.24825	147.64412	26.471115	440.33000
1	Afghanistan	AFG	1991	893.88434	137.76122	147.56696	25.548681	439.47202
2	Afghanistan	AFG	1992	892.34973	137.08030	147.13086	24.637949	437.60718
3	Afghanistan	AFG	1993	891.51587	136.48602	146.78812	23.863169	436.69104
4	Afghanistan	AFG	1994	891.39160	136.18323	146.58481	23.189074	436.76800
5	Afghanistan	AFG	1995	891.21344	135.65398	146.63217	22.503244	436.69098
6	Afghanistan	AFG	1996	891.92096	135.16324	146.67957	21.827526	436.52713
7	Afghanistan	AFG	1997	893.02045	134.20360	146.54947	21.124723	436.42557
8	Afghanistan	AFG	1998	894.73680	133.56137	146.69704	20.412657	436.77020
9	Afghanistan	AFG	1999	896.13806	132.82457	146.76817	19.931143	437.12753

Data Pre Processing starts in section 2 in the notebook. The null values are represented as “..” in the downloaded CSVs , these values are replaced by nulls in the section 2.1. Also the data has years in columns and indicators in rows , so the data is unpivoted about years in section 2.2. Following this the year column is formatted to show numerical values in section 2.3 :

2.1 remove '.. (null values represented as '..') from indicators data

In [7]:
```
Indicator_data_df.replace('..', np.nan, inplace=True)
Indicator_data_df.fillna('', inplace=True)

```

In [8]:
```
# Chekcking indicators after removal

Indicator_data_df.head()
```

Out[8]:

	Country Name	Country Code	Series Name	Series Code	1960 [YR1960]	1961 [YR1961]	1962 [YR1962]	1963 [YR1963]	1964 [YR1964]	1965 [YR1965]	...	2013 [YR2013]	2014 [YR2014]	[YR
0	Afghanistan	AFG	Adjusted net national income per capita (curre...	NY.ADJ.NNTY.PC.CD							...	586.6136889	575.6807964	526.93
1	Afghanistan	AFG	Consumer price index (2010 = 100)	FP.CPI.TOTL							...	127.7952231	133.7683667	132.883
2	Afghanistan	AFG	Inflation, consumer prices (annual %)	FP.CPI.TOTL.ZG							...	7.385771784	4.673996035	-0.66170
3	Afghanistan	AFG	Employment in industry (% of total employment)...	SL.IND.EMPL.ZS							...	19.53853	22.04671	20.6
4	Afghanistan	AFG	Employment in services (% of total employment)...	SL.SRV.EMPL.ZS							...	32.82068	33.15474	34.6

5 rows × 67 columns

2.2 Unpivot Indicators data about years and Pivot about values

In [9]:
```
#Getting List of Columns

Indicator_data_df_columns  = Indicator_data_df.columns
Indicator_data_columns_list = Indicator_data_df_columns.tolist()
pprint.pprint(Indicator_data_columns_list)


```

2.3 Format year column , convert value to numeric column

In [11]:
```
# Formatting Year Column

Indicators_df['year'] = Indicators_df['year'].str[:4]
Indicators_df.head()

```

Out[11]:

	Country Name	Country Code	Series Code	year	value
0	Afghanistan	AFG	NY.ADJ.NNTY.PC.CD	1960	
1	Afghanistan	AFG	FP.CPI.TOTL	1960	
2	Afghanistan	AFG	FP.CPI.TOTL.ZG	1960	
3	Afghanistan	AFG	SL.IND.EMPL.ZS	1960	
4	Afghanistan	AFG	SL.SRV.EMPL.ZS	1960	

In [12]:
```
# Typecasting Value column to numeric values
Indicators_df['value'] = pd.to_numeric(Indicators_df['value'])
```

The column Year and Country code are renamed in section 2.4 to make it consistent in mental health disorders dataset and indicators dataset. This is followed up by joining the two dataframes in section 2.5

2.4 Renaming colunmnames to make it consistent across all dataframes

In [13]:
```
Indicators_df = Indicators_df.rename(columns={'year': 'Year', 'Country Code' : 'Code'})
Indicators_df['value'].replace('', np.nan, inplace=True)
Indicators_df.head()
```

Out[13]:

	Country Name	Code	Series Code	Year	value
0	Afghanistan	AFG	NY.ADJ.NNTY.PC.CD	1960	NaN
1	Afghanistan	AFG	FP.CPI.TOTL	1960	NaN
2	Afghanistan	AFG	FP.CPI.TOTL.ZG	1960	NaN
3	Afghanistan	AFG	SL.IND.EMPL.ZS	1960	NaN
4	Afghanistan	AFG	SL.SRV.EMPL.ZS	1960	NaN

In [14]:
```
Indicators_df['value'] = Indicators_df['value'].apply(lambda x: float(x) if x else float('nan'))
column_types = Indicators_df.dtypes

print(column_types)

```

```
Country Name     object
Code             object
Series Code      object
Year             object
value           float64
dtype: object
```

2.5 Join Indicators and mental health dataframes based on country code and year

In [15]:
```
Indicators_df['Year'] = Indicators_df['Year'].astype(str)
Mental_data_df['Year'] = Mental_data_df['Year'].astype(str)
data_df = pd.merge(Indicators_df, Mental_data_df, on=['Code', 'Year'], how='inner')
```

In [16]:
```
data_df.head()
```

Out[16]:

	Country Name	Code	Series Code	Year	value	Entity	DALYs from depressive disorders per 100,000 people in, both sexes aged age-standardized	DALYs from schizophrenia per 100,000 people in, both sexes aged age-standardized	DALYs from bipolar disorder per 100,000 people in, both sexes aged age-standardized	DALYs from eating disorders per 100,000 people in, both sexes aged age-standardized	DALYs from anxiety disorders per 100,000 people in, both sexes aged age-standardized
0	Afghanistan	AFG	NY.ADJ.NNTY.PC.CD	1990	NaN	Afghanistan	895.22565	138.24825	147.64412	26.471115	440.33
1	Afghanistan	AFG	FP.CPI.TOTL	1990	NaN	Afghanistan	895.22565	138.24825	147.64412	26.471115	440.33
2	Afghanistan	AFG	FP.CPI.TOTL.ZG	1990	NaN	Afghanistan	895.22565	138.24825	147.64412	26.471115	440.33
3	Afghanistan	AFG	SL.IND.EMPL.ZS	1990	NaN	Afghanistan	895.22565	138.24825	147.64412	26.471115	440.33
4	Afghanistan	AFG	SL.SRV.EMPL.ZS	1990	NaN	Afghanistan	895.22565	138.24825	147.64412	26.471115	440.33

In [17]:
```
data_df.info()
```

In section 2.6 the disorders DALYs are renamed to a shorter name for ease of understanding :

2.6 Renaming the disorder features to shorter names

In [18]:
```
data_df.columns

```

Out[18]:
```
Index(['Country Name', 'Code', 'Series Code', 'Year', 'value', 'Entity',
       'DALYs from depressive disorders per 100,000 people in, both sexes aged age-standardized',
       'DALYs from schizophrenia per 100,000 people in, both sexes aged age-standardized',
       'DALYs from bipolar disorder per 100,000 people in, both sexes aged age-standardized',
       'DALYs from eating disorders per 100,000 people in, both sexes aged age-standardized',
       'DALYs from anxiety disorders per 100,000 people in, both sexes aged age-standardized'],
      dtype='object')
```

In [19]:
```
data_df = data_df.rename(columns={
    'DALYs from depressive disorders per 100,000 people in, both sexes aged age-standardized': 'Depressive_Disorders',
    'DALYs from schizophrenia per 100,000 people in, both sexes aged age-standardized': 'Schizophrenia',
    'DALYs from bipolar disorder per 100,000 people in, both sexes aged age-standardized': 'Bipolar_Disorders',
    'DALYs from eating disorders per 100,000 people in, both sexes aged age-standardized': 'Eating_Disorders',
    'DALYs from anxiety disorders per 100,000 people in, both sexes aged age-standardized': 'Anxiety_Disorders'

})
data_df = data_df.drop(columns=['Entity'])



```

In [20]:
```
data_df.head()
#data_df.to_csv('data.csv')
```

Out[20]:

	Country Name	Code	Series Code	Year	value	Depressive_Disorders	Schizophrenia	Bipolar_Disorders	Eating_Disorders	Anxiety_Disorders
0	Afghanistan	AFG	NY.ADJ.NNTY.PC.CD	1990	NaN	895.22565	138.24825	147.64412	26.471115	440.33
1	Afghanistan	AFG	FP.CPI.TOTL	1990	NaN	895.22565	138.24825	147.64412	26.471115	440.33
2	Afghanistan	AFG	FP.CPI.TOTL.ZG	1990	NaN	895.22565	138.24825	147.64412	26.471115	440.33
3	Afghanistan	AFG	SL.IND.EMPL.ZS	1990	NaN	895.22565	138.24825	147.64412	26.471115	440.33
4	Afghanistan	AFG	SL.SRV.EMPL.ZS	1990	NaN	895.22565	138.24825	147.64412	26.471115	440.33

In section 2.7 the data is unpivoted about indicators so they show up in row and can be used in prediction. These indicator codes in the column series code are then renamed for ease of understanding

2.7 Pivoting about indicators and renaming indicator columns

In [21]:
```
pivoted_df = data_df.pivot(index=['Country Name','Code','Year','Depressive_Disorders','Schizophrenia','Bipolar_Disorders','E
```

In [22]:
```
pivoted_df.reset_index(inplace=True)
```

In [23]:
```
pivoted_df.columns
```

Out[23]:
```
Index(['Country Name', 'Code', 'Year', 'Depressive_Disorders', 'Schizophrenia',
       'Bipolar_Disorders', 'Eating_Disorders', 'Anxiety_Disorders',
       'FP.CPI.TOTL', 'FP.CPI.TOTL.ZG', 'IC.BUS.NREG', 'NY.ADJ.NNTY.PC.CD',
       'SI.DST.50MD', 'SI.POV.MDIM.XQ', 'SL.EMP.SELF.ZS', 'SL.IND.EMPL.ZS',
       'SL.SRV.EMPL.ZS', 'SL.UEM.ADVN.ZS', 'SL.UEM.TOTL.ZS'],
      dtype='object', name='Series Code')
```

In [24]:
```
pivoted_df.head()
```

Out[24]:

Series Code	Country Name	Code	Year	Depressive_Disorders	Schizophrenia	Bipolar_Disorders	Eating_Disorders	Anxiety_Disorders	FP.CPI.TOTL	FP.CPI.TOTL.ZG	IC.
0	Afghanistan	AFG	1990	895.22565	138.24825	147.64412	26.471115	440.33000	NaN	NaN	
1	Afghanistan	AFG	1991	893.88434	137.76122	147.56696	25.548681	439.47202	NaN	NaN	
2	Afghanistan	AFG	1992	892.34973	137.08030	147.13086	24.637949	437.60718	NaN	NaN	
3	Afghanistan	AFG	1993	891.51587	136.48602	146.78812	23.863169	436.69104	NaN	NaN	
4	Afghanistan	AFG	1994	891.39160	136.18323	146.58481	23.189074	436.76800	NaN	NaN	

In [25]:
```

#renaming column indicator columns

#a = FP.CPI.TOTL
#b = FP.CPI.TOTL.ZG
#c = IC.BUS.NREG
#d = SI.DST.50MD
#e = SL.UEM.ADVN.ZS
#f = SL.EMP.SELF.ZS
#g =  SL.UEM.TOTL.ZS
#h = NY.ADJ.NNTY.PC.CD
#i = SI.POV.MDIM.XQ
#j = SL.SRV.EMPL.ZS
#k = SL.IND.EMPL.ZS


pivoted_df = pivoted_df.rename(columns={'FP.CPI.TOTL': 'A', 'FP.CPI.TOTL.ZG' : 'B' , 'IC.BUS.NREG' : 'C', 'SI.DST.50MD' : 'D


```

There are missing values for indicators and indicators with more than 60 percent data missing are dropped from this analysis. Missing values are handled in section 2.8.

2.8 Handling Missing values

2.8.1 Checking distribution of missing/null values and visualizing them

In [26]:
```
# Counting blank values in all indicator columns

blank_counts = pivoted_df.groupby(['Country Name', 'Code','Year', 'Depressive_Disorders','Schizophrenia','Bipolar_Disorders'
blank_counts.head()

#blank_counts.to_csv('blanks.csv')
```

Out[26]:

								Series Code	Country Name	Code	Year	Depressive_Disorc
Country Name	Code	Year	Depressive_Disorders	Schizophrenia	Bipolar_Disorders	Eating_Disorders	Anxiety_Disorders					
Afghanistan	AFG	1990	895.22565	138.24825	147.64412	26.471115	440.33000	0	0	0		
		1991	893.88434	137.76122	147.56696	25.548681	439.47202	0	0	0		
		1992	892.34973	137.08030	147.13086	24.637949	437.60718	0	0	0		
		1993	891.51587	136.48602	146.78812	23.863169	436.69104	0	0	0		
		1994	891.39160	136.18323	146.58481	23.189074	436.76800	0	0	0		

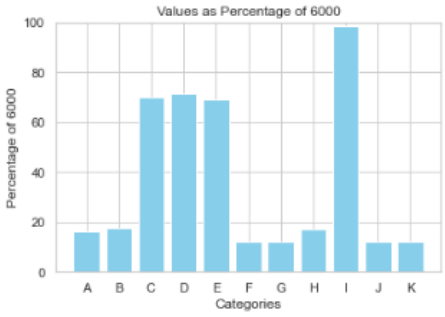
In [27]:
```
# Counting total blank values in each column

cols = ['A','B','C','D','E','F','G','H','I','J','K']
blanks_agg_df = blank_counts[cols].sum()
print(blanks_agg_df)
```

```
Series Code
A     984
B    1044
C    4214
D    4293
E    4141
F     722
G     722
H    1036
I    5917
J     722
K     722
dtype: int64
```

```
percentage_distribution = blank_counts[cols].mean() * 100

print (percentage_distribution)
```

```
Series Code
A    16.400000
B    17.400000
C    70.233333
D    71.550000
E    69.016667
F    12.033333
G    12.033333
H    17.266667
I    98.616667
J    12.033333
K    12.033333
dtype: float64
```

[29]:
```
series = pd.Series(blanks_agg_df)
# Calculating percentage out of 6000 which is total number of records
percentages = series / 6000 * 100

# Plot the percentages
plt.bar(percentages.index, percentages.values, color='skyblue')
plt.xlabel('Categories')
plt.ylabel('Percentage of 6000')
plt.title('Values as Percentage of 6000')
plt.ylim(0, 100)  # Set the y-axis limit to 0-100 for percentage representation

# Show the plot
plt.show()
```



2.8.2 Dropping indicators where more than 60 percent data is null

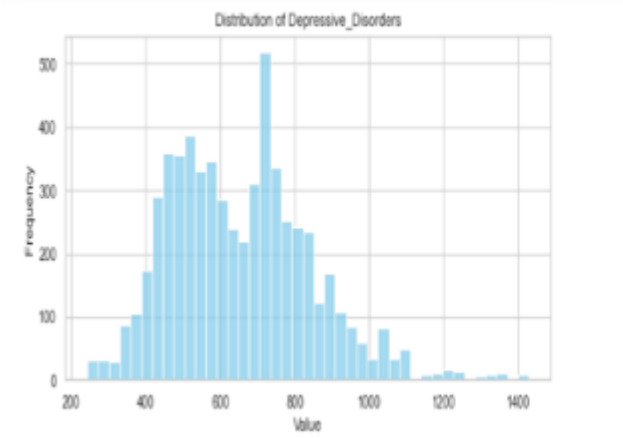
[30]:
```
pivoted_df = pivoted_df[['Country Name', 'Code','Year', 'Depressive_Disorders','Schizophrenia','Bipolar_Disorders','Eating_D
```

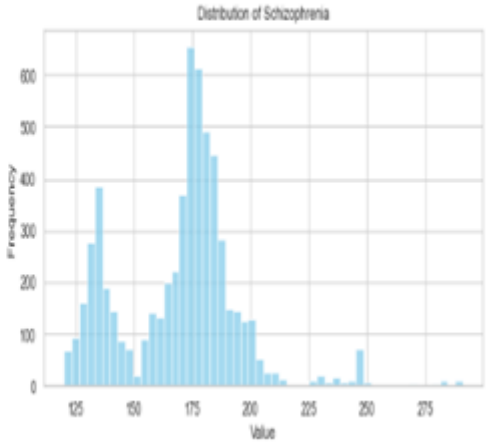
For the columns remaining in the analysis, missing values are replaced by mean values in section 2.8.2.

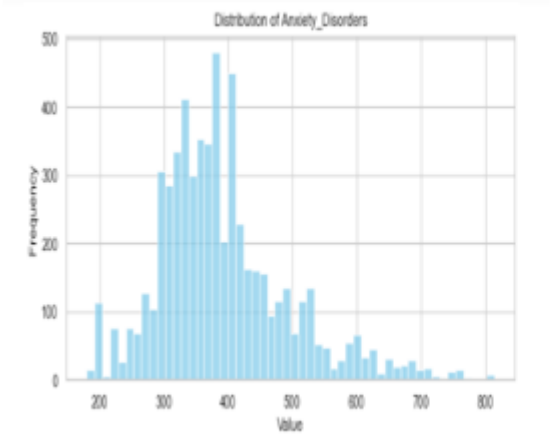
2.8.3 Replacing missing values with mean()

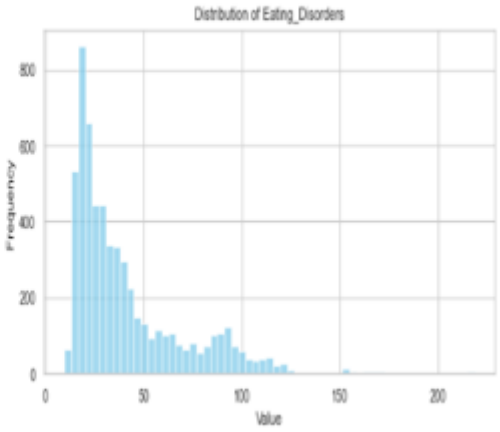
```
In [32]:
1 numerical_columns = ['A', 'B', 'F' ,'G' , 'H' , 'J' , 'K']
2 pivoted_df[numerical_columns] = pivoted_df[numerical_columns].fillna(pivoted_df[numerical_columns].mean())
```

This is followed up with exploratory data anlaysis using seaborn and feature scaling. In EDA distribution of all the columns is explored and correlation matrix is prepared for indicators and mental health disorders to find out any association.

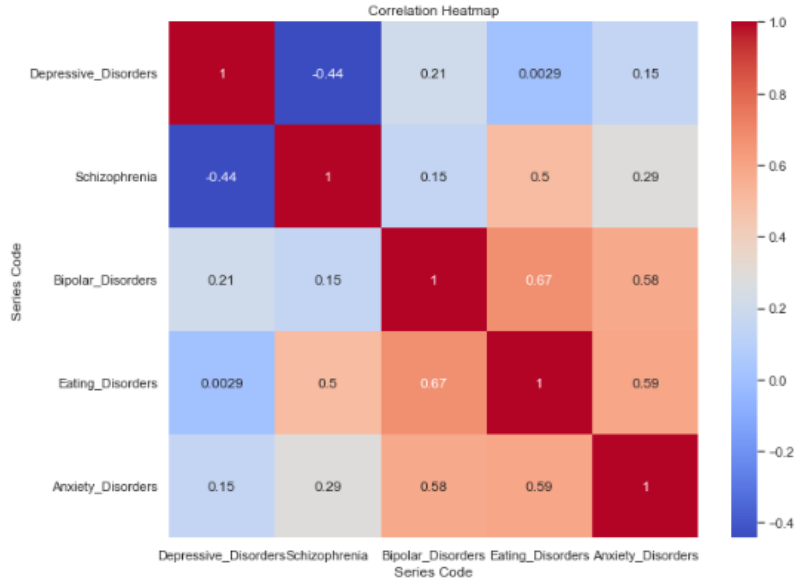




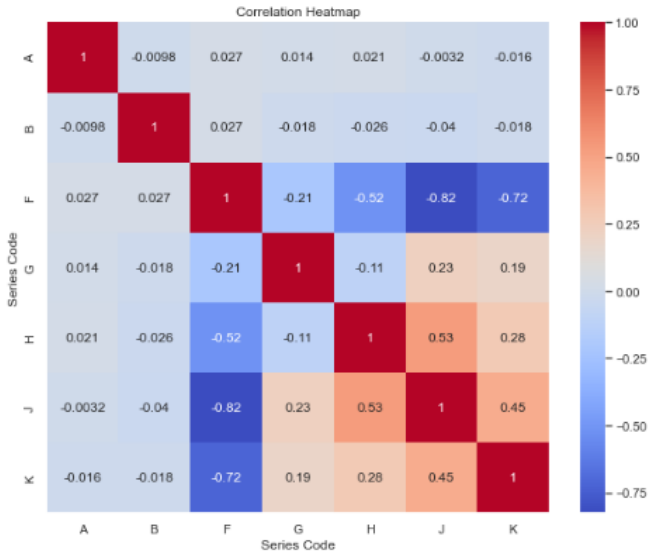




In [40]:
```
#correlation matrix

correlation_matrix = target_v_df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



In [43]:
```
#correlation matrix

correlation_matrix = indicator_v_df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



It is found that the disorders bipolar disorders and anxiety disorders are correlated according to available data, hence bipolar disorder are dropped. Similarly indicators J , K and F are correlated so the indicators J and K are dropped.

In [41]:
```
#Bipolar and anxiery disorder seem correlated hence replacing bipolar

target_v_df = df[['Country Name','Code','Year','Depressive_Disorders','Schizophrenia','Eating_Disorders','Anxiety_Disorders'
```

In [44]:
```
# dropping J and K as F , J and K are correlated and creating target dataframes for all the disorders


df_ad = df[['Country Name','Code','Year','A','B','F','G','H','Anxiety_Disorders']]

df_dd = df[['Country Name','Code','Year','A','B','F','G','H','Depressive_Disorders']]
df_s  = df[['Country Name','Code','Year','A','B','F','G','H','Schizophrenia']]
df_ed = df[['Country Name','Code','Year','A','B','F','G','H','Eating_Disorders']]
```

This is followed up by creating separate dataframes for model implementaion for all the disorders as seen in the picture above.

4 Modelling Evaluation and Visualizations

All the disorders are modelled using naiive forecasting and exponential smoothing model and evaluated against mean squared error, mean absolute error , root mean squared error ,R squared and Adjusted R squared values. Following this forecasted values are generated in separate CSVs to be used for data visualizations using power bi. Section 3.1 has modelling and forecasting for anxiety disorders.

3.1 Modelling for Anxiety Disorders

In [53]:
```
data = df_ad
```

In [54]:
```
#renaming anxiety disorder to t and exporting to csv
dataad = data.rename(columns={'Anxiety_Disorders': 'T'})

dataad.to_csv('addata.csv')
```

In [55]:
```
dataad.head()
```

Out[55]:

	Series Code	Country Name	Code	Year	A	B	F	G	H	T
0		Afghanistan	AFG	1990	6.410690e-16	1.504054e-16	2.443652e-15	1.862180e-15	7.866365e-17	440.33000
1		Afghanistan	AFG	1991	6.410690e-16	1.504054e-16	1.854854e+00	-8.010559e-03	7.866365e-17	439.47202
2		Afghanistan	AFG	1992	6.410690e-16	1.504054e-16	1.853048e+00	2.012347e-04	7.866365e-17	437.60718
3		Afghanistan	AFG	1993	6.410690e-16	1.504054e-16	1.850257e+00	-7.661121e-03	7.866365e-17	436.69104
4		Afghanistan	AFG	1994	6.410690e-16	1.504054e-16	1.848748e+00	-9.757749e-03	7.866365e-17	436.76800

In [56]:
```
#pip install pmdarima
```

3.1.1 Naiive Forecasting

In [57]:
```
import pandas as pd
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Load data
df = pd.read_csv('addata.csv')

# Prepare data
df = df.set_index('Year')

# Split data into train and test
train = df[:-12]
test = df[-12:]

# Naive forecast
predictions = train['T'][-12:].values

# Calculate MSE
mse = mean_squared_error(test['T'].values, predictions)
mae = mean_absolute_error(test['T'].values, predictions)
r2 = r2_score(test['T'].values, predictions)



print('Naive MSE:', mse)
print('Naive MAE:', mae)
print('R-squared (R²):', r2)

n = len(test)
k = 1  # number of predictors (in this case, 1 for the naive forecast)
```

3.1.2 Exponential Smoothing Model

In [58]:

```
import matplotlib.pyplot as plt
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn.metrics import mean_squared_error
from math import sqrt

df = pd.read_csv('addata.csv')

# Specify the number of steps to forecast
forecast_steps = 3

# Fit the Exponential Smoothing model
model = ExponentialSmoothing(df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
model_fit = model.fit()

# Make forecasts
forecast_values = model_fit.forecast(steps=forecast_steps)

# Evaluate the model using Mean Squared Error (MSE)
y_test = df['T'][-forecast_steps:]
mse = mean_squared_error(y_test, forecast_values)
rmse = sqrt(mse)
print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")

# Calculate R-squared
y_mean = df['T'].mean()
sst = ((df['T'] - y_mean)**2).sum()
ssr = ((y_test - forecast_values)**2).sum()
r2 = 1 - (ssr / sst)
print(f"R-squared (R²): {r2}")

# Calculate Adjusted R-squared
n = len(y_test)
k = 3  # number of predictors (seasonal, trend, and damping terms)
adjusted_r2 = 1 - ((1 - r2) * (n - 1) / (n - k - 1))
print(f"Adjusted R-squared: {adjusted_r2}")

```

```
C:\Users\KOMALP~1\AppData\Local\Temp/ipykernel_11200/1313939485.py:12: FutureWarning: the 'damped'' keyword is deprecated, use
'damped_trend' instead
  model = ExponentialSmoothing(df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.py:427: FutureWarning: After 0.13 initialization m
ust be handled at model creation
  warnings.warn(
```

```
Mean Squared Error: 16.58262528429255
Root Mean Squared Error: 4.07217697114118
R-squared (R²): 1.0
Adjusted R-squared: 1.0
```

3.1.3 Forecasted Values

In [59]:

```
import pandas as pd
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn.metrics import mean_squared_error
from math import sqrt

# Print column names to identify the correct ones
print(df.columns)

# Specify the column name for the year
year_column = 'Year'

# Specifying the number of steps to forecast (3 years)
forecast_steps = 3

# Create an empty dataframe to store the forecasted values
forecast_df = pd.DataFrame()

# Loop through each unique country in the dataframe
for country in df['Country Name'].unique():
    # Filter data for the current country
    country_df = df[df['Country Name'] == country]

    # Fit the Exponential Smoothing model
    model = ExponentialSmoothing(country_df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
    model_fit = model.fit()

    # Make forecasts for the next 3 years
    forecast_values = model_fit.forecast(steps=forecast_steps)

    # Create a dataframe to store the forecast values with the corresponding country, code, and year
    country_forecast_df = pd.DataFrame({'Country Name': [country] * forecast_steps,
                                        'Code': [country_df['Code'].iloc[0]] * forecast_steps,
                                        year_column: list(range(country_df[year_column].max() + 1, country_df[year_column].m
                                        'Forecast': forecast_values})

    # Concatenate the current country's forecast dataframe to the overall forecast dataframe
    forecast_df = pd.concat([forecast_df, country_forecast_df], ignore_index=True)

# Print the forecasted values dataframe
print(forecast_df)

forecast_df.to_csv('forecasted_values_anxiety_disorders.csv', index=False)

```

```
Index(['Unnamed: 0', 'Country Name', 'Code', 'Year', 'A', 'B', 'F', 'G', 'H',
       'T'],
      dtype='object')
```

```
C:\Users\KOMALP~1\AppData\Local\Temp/ipykernel_11200/3379567552.py:24: FutureWarning: the 'damped'' keyword is deprecated, u
```

Section 3.2 has modelling and forecasting for depressive disorders.

3.2 Modelling for Depressive disorders

3.2.1 Naive Forecasting

[60]:
```
data = df_dd
```

[61]:
```
#renaming anxiety disorder to t and exporting to csv
datadd = data.rename(columns={'Depressive_Disorders': 'T'})

datadd.to_csv('dddata.csv')


```

[62]:
```
import pandas as pd
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split

# Load data
df = pd.read_csv('dddata.csv')

# Prepare data
df = df.set_index('Year')

# Split data into train and test (70% train, 30% test)
train, test = train_test_split(df, test_size=0.3, shuffle=False)

# Naive forecast
predictions = train['T'][-len(test):].values

# Calculate MSE
mse = mean_squared_error(test['T'].values, predictions)
mae = mean_absolute_error(test['T'].values, predictions)
r2 = r2_score(test['T'].values, predictions)

print('Naive MSE:', mse)
print('Naive MAE:', mae)
print('R-squared (R²):', r2)

n = len(test)
k = 1  # number of predictors (in this case, 1 for the naive forecast)
adjusted_r2 = 1 - ((1 - r2) * (n - 1) / (n - k - 1))

print('Adjusted R-squared:', adjusted_r2)
```

```
Naive MSE: 66881.41718150298
Naive MAE: 200.21860311666666
R-squared (R²): -0.6408914186754229
Adjusted R-squared: -0.6418040390417608
```

3.2.2 Exponential Smoothing Model

In [63]:
```
import matplotlib.pyplot as plt
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn.metrics import mean_squared_error
from math import sqrt

df = pd.read_csv('dddata.csv')

# Specify the number of steps to forecast
forecast_steps = 3

# Fit the Exponential Smoothing model
model = ExponentialSmoothing(df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
model_fit = model.fit()

# Make forecasts
forecast_values = model_fit.forecast(steps=forecast_steps)

# Evaluate the model using Mean Squared Error (MSE)
y_test = df['T'][-forecast_steps:]
mse = mean_squared_error(y_test, forecast_values)
rmse = sqrt(mse)
print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")

# Calculate R-squared
y_mean = df['T'].mean()
sst = ((df['T'] - y_mean)**2).sum()
ssr = ((y_test - forecast_values)**2).sum()
r2 = 1 - (ssr / sst)
print(f"R-squared (R²): {r2}")

# Calculate Adjusted R-squared
n = len(y_test)
k = 3  # number of predictors (seasonal, trend, and damping terms)
adjusted_r2 = 1 - ((1 - r2) * (n - 1) / (n - k - 1))
print(f"Adjusted R-squared: {adjusted_r2}")

```

```
C:\Users\KOMALP~1\AppData\Local\Temp/ipykernel_11200/595276686.py:12: FutureWarning: the 'damped'' keyword is deprecated, use 'damped_trend' instead
  model = ExponentialSmoothing(df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.py:427: FutureWarning: After 0.13 initialization must be handled at model creation
  warnings.warn(
```

```
Mean Squared Error: 12.898036252732785
Root Mean Squared Error: 3.5913836125834266
R-squared (R²): 1.0
Adjusted R-squared: 1.0
```

3.2.3 Forecasted Values

In [64]:

```
import pandas as pd
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn.metrics import mean_squared_error
from math import sqrt

# Print column names to identify the correct ones
print(df.columns)

# Specify the column name for the year
year_column = 'Year'

# Specifying the number of steps to forecast (3 years)
forecast_steps = 3

# Create an empty dataframe to store the forecasted values
forecast_df = pd.DataFrame()

# Loop through each unique country in the dataframe
for country in df['Country Name'].unique():
    # Filter data for the current country
    country_df = df[df['Country Name'] == country]

    # Fit the Exponential Smoothing model
    model = ExponentialSmoothing(country_df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
    model_fit = model.fit()

    # Make forecasts for the next 3 years
    forecast_values = model_fit.forecast(steps=forecast_steps)

    # Create a dataframe to store the forecast values with the corresponding country, code, and year
    country_forecast_df = pd.DataFrame({'Country Name': [country] * forecast_steps,
                                        'Code': [country_df['Code'].iloc[0]] * forecast_steps,
                                        year_column: list(range(country_df[year_column].max() + 1, country_df[year_column].m
                                        'Forecast': forecast_values})

    # Concatenate the current country's forecast dataframe to the overall forecast dataframe
    forecast_df = pd.concat([forecast_df, country_forecast_df], ignore_index=True)

# Print the forecasted values dataframe
print(forecast_df)

forecast_df.to_csv('forecasted_values_dipressive_disorders.csv', index=False)
```

```
Index(['Unnamed: 0', 'Country Name', 'Code', 'Year', 'A', 'B', 'F', 'G', 'H',
       'T'],
      dtype='object')
C:\Users\KOMALP~1\AppData\Local\Temp/ipykernel_11200/923382177.py:24: FutureWarning: the 'damped'' keyword is deprecated, us
e 'damped_trend' instead
  model = ExponentialSmoothing(country_df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.py:427: FutureWarning: After 0.13 initializatio
```

Section 3.3 has modelling evaluation and forecasting for Schizophrenia :

3.3 Modelling for Schizophrenia

In [65]:

```
#Running standard scaler for all indicator columns with target variable as anxiety disorders

from sklearn.preprocessing import StandardScaler

data = df_s

columns_to_scale = ['A', 'B','F','G','H']

# Create a StandardScaler
scaler = StandardScaler()


# Fit and transform the numerical columns
```

In [66]:

```
#renaming depressive disorder to t and exporting to csv
datas = data.rename(columns={'Schizophrenia': 'T'})

datas.to_csv('sdata.csv')
```

In [67]:

```
datas.head()
```

Out[67]:

Series Code	Country Name	Code	Year	A	B	F	G	H	T
0	Afghanistan	AFG	1990	91.735775	27.1985	44.252894	8.166848	7903.252158	138.24825
1	Afghanistan	AFG	1991	91.735775	27.1985	92.793270	8.121000	7903.252158	137.76122
2	Afghanistan	AFG	1992	91.735775	27.1985	92.745990	8.168000	7903.252158	137.08030
3	Afghanistan	AFG	1993	91.735775	27.1985	92.672950	8.123000	7903.252158	136.48602
4	Afghanistan	AFG	1994	91.735775	27.1985	92.633470	8.111000	7903.252158	136.18323

3.3.1 Naïve Forecasting

In [68]:

```
import pandas as pd
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Load data
df = pd.read_csv('sdata.csv')

# Prepare data
df = df.set_index('Year')

# Split data into train and test
train = df[:-12]
test = df[-12:]

# Naive forecast
predictions = train['T'][-12:].values

# Calculate MSE
mse = mean_squared_error(test['T'].values, predictions)
mae = mean_absolute_error(test['T'].values, predictions)
r2 = r2_score(test['T'].values, predictions)
```

3.3.2 Exponential Smoothing Model

[69]:
```
import matplotlib.pyplot as plt
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn.metrics import mean_squared_error , r2_score
from math import sqrt

# Load data
df = pd.read_csv('sdata.csv')

# Specify the number of steps to forecast
forecast_steps = 3

# Fit the Exponential Smoothing model
model = ExponentialSmoothing(df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
model_fit = model.fit()

# Make forecasts
forecast_values = model_fit.forecast(steps=forecast_steps)

# Evaluate the model using Mean Squared Error (MSE)
y_test = df['T'][-forecast_steps:]
mse = mean_squared_error(y_test, forecast_values)
rmse = sqrt(mse)
print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")


r2 = r2_score(test['T'].values, predictions)
#print('R-squared (R²):', r2)

# Calculate R-squared
y_mean = df['T'].mean()
sst = ((df['T'] - y_mean)**2).sum()
ssr = ((y_test - forecast_values)**2).sum()
r2 = 1 - (ssr / sst)
print(f"R-squared (R²): {r2}")

# Calculate Adjusted R-squared
n = len(y_test)
k = 3  # number of predictors (seasonal, trend, and damping terms)
adjusted_r2 = 1 - ((1 - r2) * (n - 1) / (n - k - 1))
print(f"Adjusted R-squared: {adjusted_r2}")

```

```
C:\Users\KOMALP~1\AppData\Local\Temp/ipykernel_11200/3490289368.py:13: FutureWarning: the 'damped'' keyword is deprecated, use 'damped_trend' instead
  model = ExponentialSmoothing(df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.py:427: FutureWarning: After 0.13 initialization must be handled at model creation
  warnings.warn(
```

```
Mean Squared Error: 0.19596351721161276
Root Mean Squared Error: 0.4426776673965073
```

3.3.3 Forecasted Values

In [70]:
```
import pandas as pd
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn.metrics import mean_squared_error
from math import sqrt

# Print column names to identify the correct ones
print(df.columns)

# Specify the column name for the year
year_column = 'Year'

# Specifying the number of steps to forecast (3 years)
forecast_steps = 3

# Create an empty dataframe to store the forecasted values
forecast_df = pd.DataFrame()

# Loop through each unique country in the dataframe
for country in df['Country Name'].unique():
    # Filter data for the current country
    country_df = df[df['Country Name'] == country]

    # Fit the Exponential Smoothing model
    model = ExponentialSmoothing(country_df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
    model_fit = model.fit()

    # Make forecasts for the next 3 years
    forecast_values = model_fit.forecast(steps=forecast_steps)

    # Create a dataframe to store the forecast values with the corresponding country, code, and year
    country_forecast_df = pd.DataFrame({'Country Name': [country] * forecast_steps,
                                        'Code': [country_df['Code'].iloc[0]] * forecast_steps,
                                        year_column: list(range(country_df[year_column].max() + 1, country_df[year_column].m
                                        'Forecast': forecast_values})

    # Concatenate the current country's forecast dataframe to the overall forecast dataframe
    forecast_df = pd.concat([forecast_df, country_forecast_df], ignore_index=True)

# Print the forecasted values dataframe
print(forecast_df)

forecast_df.to_csv('forecasted_values_schizophrenia.csv', index=False)

```

Section 3.4 has modelling evaluation and forecasting for eating disorders :

3.4 Modelling for Eating Disorders

n [71]:
```
#Running standard scaler for all indicator columns with target variable as anxiety disorders

from sklearn.preprocessing import StandardScaler

data = df_ed

columns_to_scale = ['A', 'B','F','G','H']

# Create a StandardScaler
scaler = StandardScaler()


# Fit and transform the numerical columns
```

n [72]:
```
#renaming depressive disorder to t and exporting to csv
dataed = data.rename(columns={'Eating_Disorders': 'T'})

dataed.to_csv('eddata.csv')
```

3.4.1 Naiive Forecasting

n [73]:
```
import pandas as pd
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Load data
df = pd.read_csv('eddata.csv')
df.head()
```

ut[73]:

	Unnamed: 0	Country Name	Code	Year	A	B	F	G	H	T
0	0	Afghanistan	AFG	1990	91.735775	27.1965	44.252894	8.166848	7903.252158	26.471115
1	1	Afghanistan	AFG	1991	91.735775	27.1965	92.793270	8.121000	7903.252158	25.548681
2	2	Afghanistan	AFG	1992	91.735775	27.1965	92.745990	8.168000	7903.252158	24.637949
3	3	Afghanistan	AFG	1993	91.735775	27.1965	92.672950	8.123000	7903.252158	23.863169
4	4	Afghanistan	AFG	1994	91.735775	27.1965	92.633470	8.111000	7903.252158	23.189074

n [74]:
```

# Prepare data
df = df.set_index('Year')

# Split data into train and test
train = df[:-12]
test = df[-12:]

# Naive forecast
predictions = train['T'][-12:].values

# Calculate MSE
mse = mean_squared_error(test['T'].values, predictions)
mae = mean_absolute_error(test['T'].values, predictions)
r2 = r2_score(test['T'].values, predictions)

```

3.4.2 Exponential Smoothing Model

In [75]:
```
import matplotlib.pyplot as plt
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn.metrics import mean_squared_error
from math import sqrt

# Specifying the number of steps to forecast
forecast_steps = 3

df = pd.read_csv('eddata.csv')

# Fit the Exponential Smoothing model
model = ExponentialSmoothing(df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
model_fit = model.fit()

# Make forecasts
forecast_values = model_fit.forecast(steps=forecast_steps)

# Evaluate the model using Mean Squared Error (MSE)
y_test = df['T'][-forecast_steps:]
mse = mean_squared_error(y_test, forecast_values)
rmse = sqrt(mse)
print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")

# Evaluate the model by calculating R-squared
y_mean = df['T'].mean()
sst = ((df['T'] - y_mean)**2).sum()
ssr = ((y_test - forecast_values)**2).sum()
r2 = 1 - (ssr / sst)
print(f"R-squared (R²): {r2}")

# Evaluate the model by calculating Adjusted R-squared
n = len(y_test)
k = 3  # number of predictors (seasonal, trend, and damping terms)
adjusted_r2 = 1 - ((1 - r2) * (n - 1) / (n - k - 1))
print(f"Adjusted R-squared: {adjusted_r2}")


```

```
C:\Users\KOMALP~1\AppData\Local\Temp/ipykernel_11200/2371666862.py:12: FutureWarning: the 'damped'' keyword is deprecated, use 'damped_trend' instead
  model = ExponentialSmoothing(df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.py:427: FutureWarning: After 0.13 initialization must be handled at model creation
  warnings.warn(
Mean Squared Error: 0.026787367780580437
Root Mean Squared Error: 0.1636846910929556
R-squared (R²): 1.0
Adjusted R-squared: 1.0
```

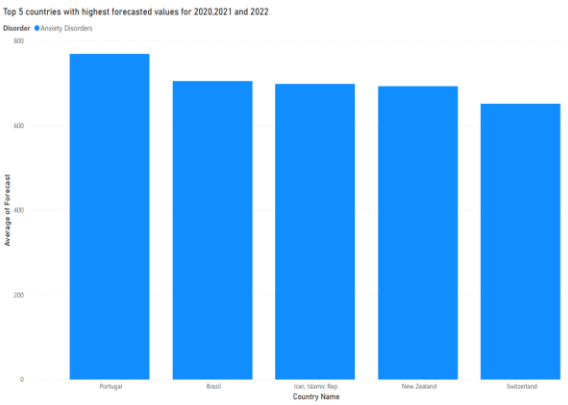
3.4.3 Forecasted Values

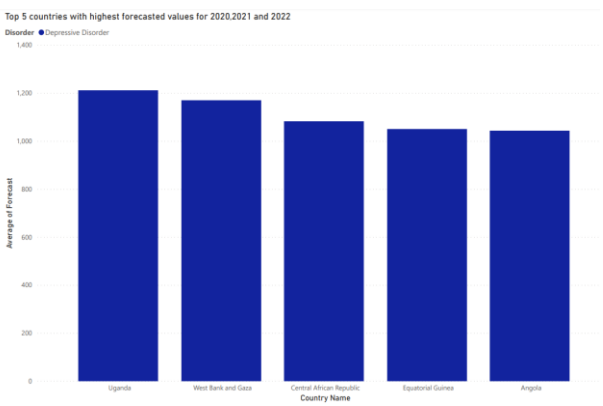
In [76]:

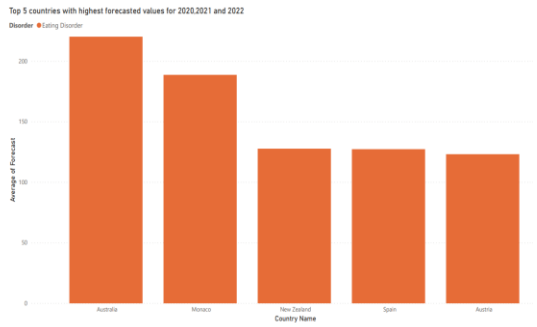
```
import pandas as pd
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn.metrics import mean_squared_error
from math import sqrt

# Print column names to identify the correct ones
print(df.columns)

# Specify the column name for the year
year_column = 'Year'

# Specifying the number of steps to forecast (3 years)
forecast_steps = 3

# Create an empty dataframe to store the forecasted values
forecast_df = pd.DataFrame()

# Loop through each unique country in the dataframe
for country in df['Country Name'].unique():
    # Filter data for the current country
    country_df = df[df['Country Name'] == country]

    # Fit the Exponential Smoothing model
    model = ExponentialSmoothing(country_df['T'], seasonal='add', seasonal_periods=4, trend='add', damped=True)
    model_fit = model.fit()

    # Make forecasts for the next 3 years
    forecast_values = model_fit.forecast(steps=forecast_steps)

    # Create a dataframe to store the forecast values with the corresponding country, code, and year
    country_forecast_df = pd.DataFrame({'Country Name': [country] * forecast_steps,
                                        'Code': [country_df['Code'].iloc[0]] * forecast_steps,
                                        year_column: list(range(country_df[year_column].max() + 1, country_df[year_column].m
                                        'Forecast': forecast_values})

    # Concatenate the current country's forecast dataframe to the overall forecast dataframe
    forecast_df = pd.concat([forecast_df, country_forecast_df], ignore_index=True)

# Print the forecasted values dataframe
print(forecast_df)

forecast_df.to_csv('forecasted_values_eating_disorders.csv', index=False)
```

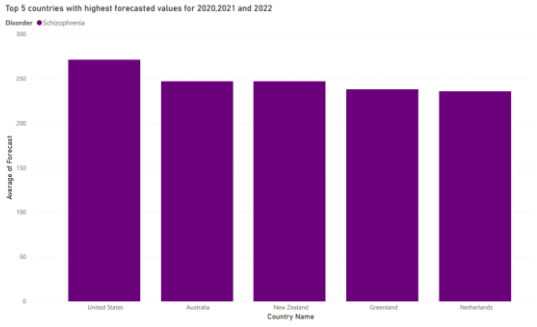
```
Index(['Unnamed: 0', 'Country Name', 'Code', 'Year', 'A', 'B', 'F', 'G', 'H',
       'T'],
```

Forecasted CSVs are used to create visualizations in power bi. Below reports show the top 5 countries forecasted to have highest number of cases of these mental health disorders. For this analysis average value of forecast for 2020, 2021 and 2022 is taken and plotted :

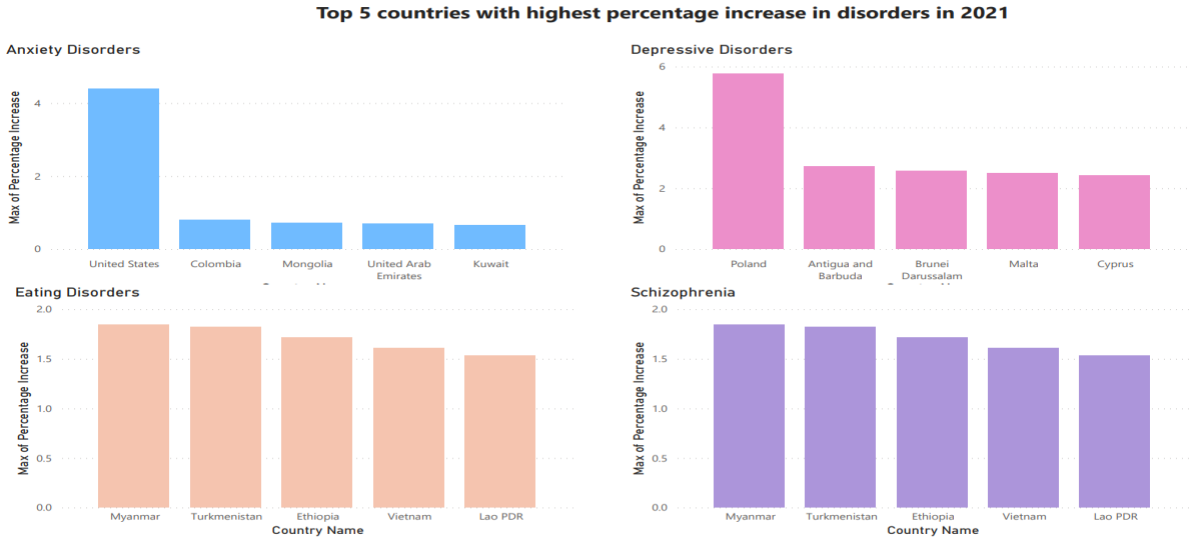








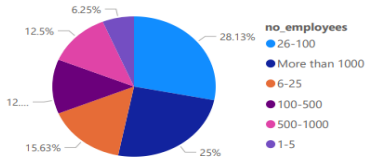
Below dashboard shows countries where percentage increase over these three years will be highest. For this visualization a new metric for percentage increase is calculated and maximum of the value is taken , the dashboard is then filtered for showing top 5 values in power bi.



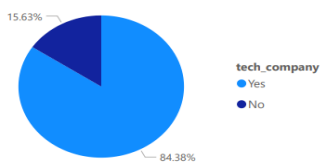
To further drill down to workplace related mental health factors, the downloaded dataset is explored against all factors and pie charts are created in power bi to show the distribution of mental health disorders according to these factors.

Percentage distribution of various factors affecting workplace mental well being

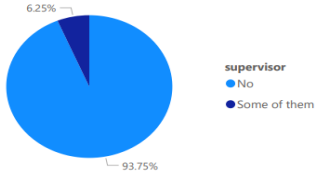
Number of Employees



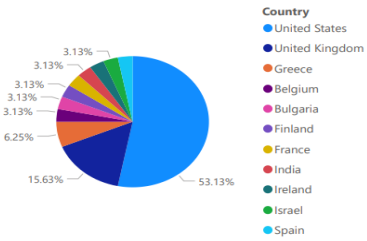
Tech Company



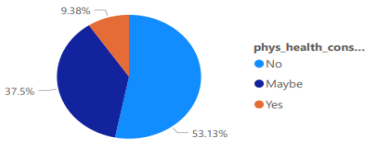
Supervisor



Country



Physical Health Condition



State Wise Distribution in US