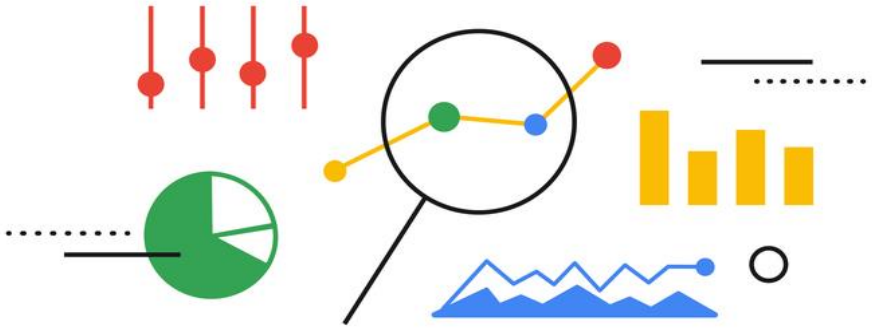
Title: "Data Analytics Project- [Public health awareness]"

Name :P GUNASEKARAN **REG NO :** 620521104008

Gmail: gunabarathi8877@gmail.com



Overview of the project

In this document, we present the findings and insights from our Data Analytics project, titled "[Public health awareness]." This project focuses on data cleaning, transformation, and statistical analysis to derive meaningful insights from raw data. Our objective is to provide a comprehensive framework for understanding the process, methodologies, and outcomes of this analytics endeavor.

Data collection

We begin by discussing the various data sources used for this project, the methods employed to acquire the data, and the preprocessing steps implemented to ensure data quality. This section provides a clear understanding of how we prepared the data for analysis.

Data can be aquired by various methods such as online surveying , interviews, online form data , data repos and scaping techniques we have used the data that is already available in the kaggle dataset

Data set : [https://www.kaggle.com/datasets/osmi/mental-health-in-tech-survey.](https://www.kaggle.com/datasets/osmi/mental-health-in-tech-survey)

Data cleaning

To analyze the whole dataset we first have to clean the data set we used kaggle to do this stuff so kaggle is compatible with various operations and I have some familiarity with kaggle

Code:

```
#imports necessary libraries to do basic things on the dataset
 import pandas as pd
 import numpy as np
 import seaborn as sns
 import matplotlib.pyplot as plt


 print('Successfully imported')
```

Read Dataset

```
#Reading data
  data = pd.read_csv('/kaggle/input/mental-health-in-tech-survey/survey.csv')
  data.head()
```

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	Often
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No	Rarely
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	Rarely
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No	Never

5 rows × 27 columns

```
#Check the dataset for missing data
  if data.isnull().sum().sum() == 0 :
    print ('There is no missing data in our dataset')
  else:
    print('There is {} missing data in our dataset
'.format(data.isnull().sum().sum()))
```

```
There is 1892 missing data in our dataset
```

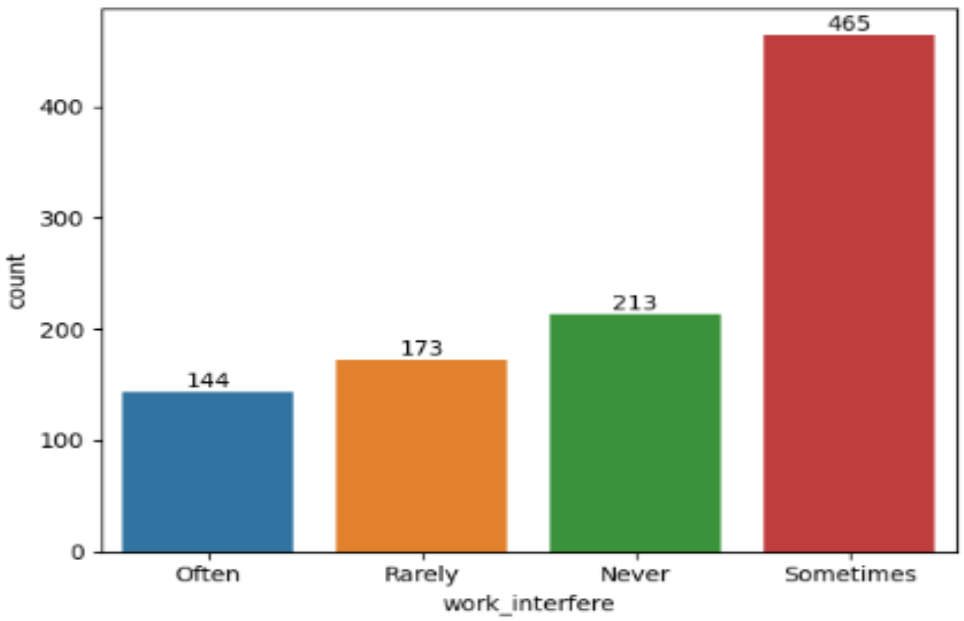
```
#Check our missing data from which columns and how many unique features
they have.
frame = pd.concat([data.isnull().sum(), data.nunique(), data.dtypes], axis
= 1, sort= False)
frame
```

	0	1	2
Timestamp	0	1246	object
Age	0	53	int64
Gender	0	49	object
Country	0	48	object
state	515	45	object
self_employed	18	2	object
family_history	0	2	object
treatment	0	2	object
work_interfere	264	4	object
no_employees	0	6	object
remote_work	0	2	object
tech_company	0	2	object
benefits	0	3	object
care_options	0	3	object
wellness_program	0	3	object
seek_help	0	3	object
anonymity	0	3	object
leave	0	5	object
mental_health_consequence	0	3	object
phys_health_consequence	0	3	object
coworkers	0	3	object
supervisor	0	3	object
mental_health_interview	0	3	object
phys_health_interview	0	3	object
mental_vs_physical	0	3	object

```
#Look at what is in the 'Work_interfere' column to choose a suitable method to fill nan values.
    data['work_interfere'].unique()
```

```
array(['Often', 'Rarely', 'Never', 'Sometimes', nan], dtype=object)
```

```
#Plot **work_interfere**
ax = sns.countplot(data = data , x = 'work_interfere');
#Add the value of each parametr on the Plot
ax.bar_label(ax.containers[0]);
```

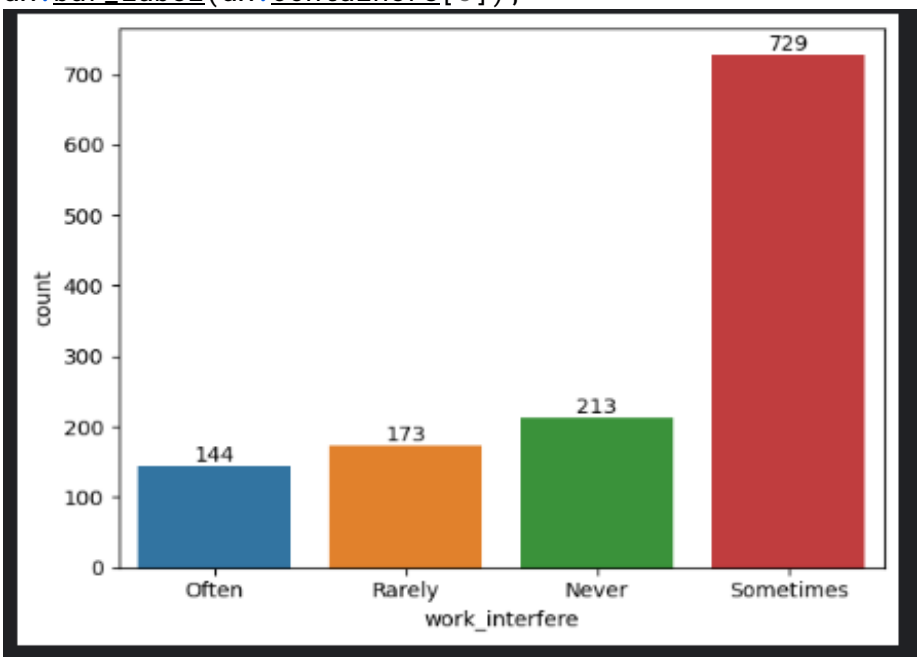



```
from sklearn.impute import SimpleImputer
import numpy as np
columns_to_drop = ['state', 'comments', 'Timestamp']
for column in columns_to_drop:
    if column in data.columns:
        data = data.drop(columns=[column])


# Fill in missing values in work_interfere column
data['work_interfere'] = np.ravel(SimpleImputer(strategy =
'most_frequent').fit_transform(data['work_interfere'].values.reshape(-
1,1)))
data['self_employed'] = np.ravel(SimpleImputer(strategy =
'most_frequent').fit_transform(data['self_employed'].values.reshape(-
1,1)))

data.head()
```

	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	no_employees	remote_work	tech_company	...	anonymity	leave	mental_health_consequence	phys_health_co
0	37	Female	United States	No	No	Yes	Often	6-25	No	Yes	...	Yes	Somewhat easy	No	
1	44	M	United States	No	No	No	Rarely	More than 1000	No	No	...	Don't know	Don't know	Maybe	
2	32	Male	Canada	No	No	No	Rarely	6-25	No	Yes	...	Don't know	Somewhat difficult	No	
3	31	Male	United Kingdom	No	Yes	Yes	Often	26-100	No	Yes	...	No	Somewhat difficult	Yes	
4	31	Male	United States	No	No	No	Never	100-500	Yes	Yes	...	Don't know	Don't know	No	

5 rows × 24 columns

```
ax = sns.countplot(data=data, x='work_interfere');
ax.bar_label(ax.containers[0]);
```




```
 #Check unique data in gender columns
     print(data['Gender'].unique())
     print('')
     print('-'*75)
     print('')
  #Check number of unique data too.
     print('number of unique Gender in our dataset is :',
data['Gender'].nunique())
```

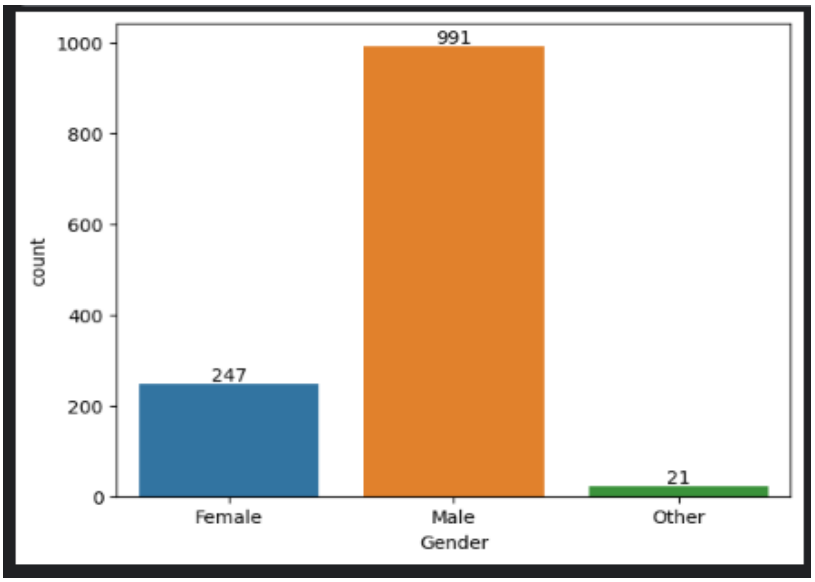
```
['Female' 'M' 'Male' 'male' 'female' 'm' 'Male-ish' 'maile' 'Trans-female'
 'Cis Female' 'F' 'something kinda male?' 'Cis Male' 'Woman' 'f' 'Mal'
 'Male (CIS)' 'queer/she/they' 'non-binary' 'Femake' 'woman' 'Make' 'Nah'
 'All' 'Enby' 'fluid' 'Genderqueer' 'Female ' 'Androgyne' 'Agender'
 'cis-female/femme' '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']

---------------------------------------------------------------------------

number of unique Gender in our dataset is : 49
```

```
#Gender data contains dictation problems, nonsense answers, and too unique
Genders.
#_So Let's clean it and organize it into Male, Female, and other categories

data['Gender'].replace(['Male ', 'male', 'M', 'm', 'Male', 'Cis Male',
                        'Man', 'cis male', 'Mail', 'Male-ish', 'Male (CIS)',
                         'Cis Man', 'msle', 'Malr', 'Mal', 'maile', 'Make',],
'Male', inplace = True)

data['Gender'].replace(['Female ', 'female', 'F', 'f', 'Woman', 'Female',
```

```
                      'femail', 'Cis Female', 'cis-female/femme', 'Femake',
'Female (cis)',
                      'woman',], 'Female', inplace = True)

data["Gender"].replace(['Female (trans)', 'queer/she/they', 'non-binary',
                      'fluid', 'queer', 'Androgyne', 'Trans-female', 'male
leaning androgynous',
                       'Agender', 'A little about you', 'Nah', 'All',
                       'ostensibly male, unsure what that really means',
                       'Genderqueer', 'Enby', 'p', 'Neuter', 'something
kinda male?',
                       'Guy (-ish) ^_^', 'Trans woman',], 'Other', inplace
= True)

print(data['Gender'].unique())
```

```
['Female' 'Male' 'Other']
```

```
#Plot Genders column after cleaning and new categorizing
ax = sns.countplot(data=data, x='Gender');
ax.bar_label(ax.containers[0]);
```




```
#Our data is clean now ? let's see.
if data.isnull().sum().sum() == 0:
    print('There is no missing data')
else:
    print('There is {} missing data'.format(data.isnull().sum().sum()))
```

```
There is no missing data
```

```
#Let's check duplicated data.
if data.duplicated().sum() == 0:
    print('There is no duplicated data:')
else:
    print('Tehre is {} duplicated data:'.format(data.duplicated().sum()))
    #If there is duplicated data drop it.
    data.drop_duplicates(inplace=True)

print('-'*50)
print(data.duplicated().sum())
```

```
Tehre is 4 duplicated data:
--------------------------------------------------
0
```

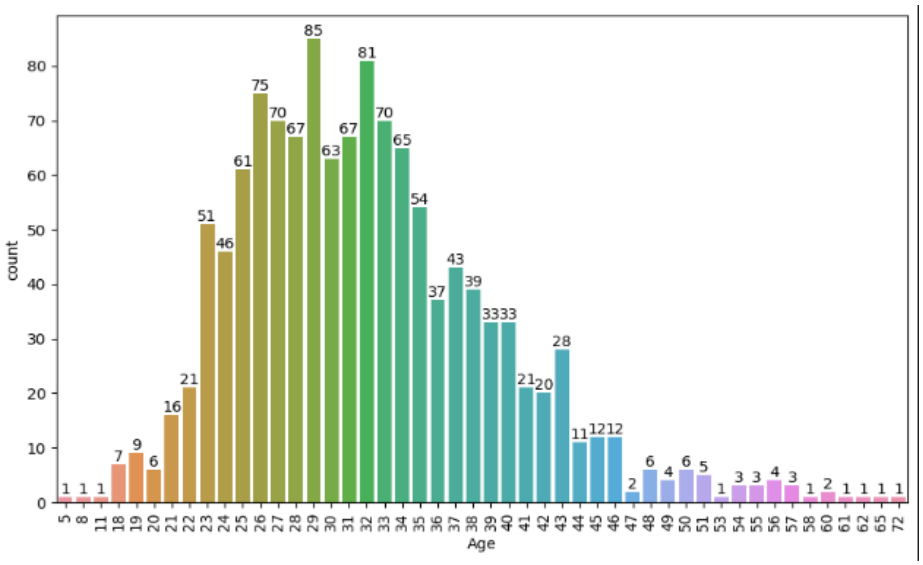
```
#Look unique data in Age column
data['Age'].unique()
```

```
[       37,        44,        32,        31,        33,
        35,        39,        42,        23,        29,
        36,        27,        46,        41,        34,
        30,        40,        38,        50,        24,
        18,        28,        26,        22,        19,
        25,        45,        21,       -29,        43,
        56,        60,        54,       329,        55,
 99999999999,        48,        20,        57,        58,
        47,        62,        51,        65,        49,
     -1726,         5,        53,        61,         8,
        11,        -1,        72])
```

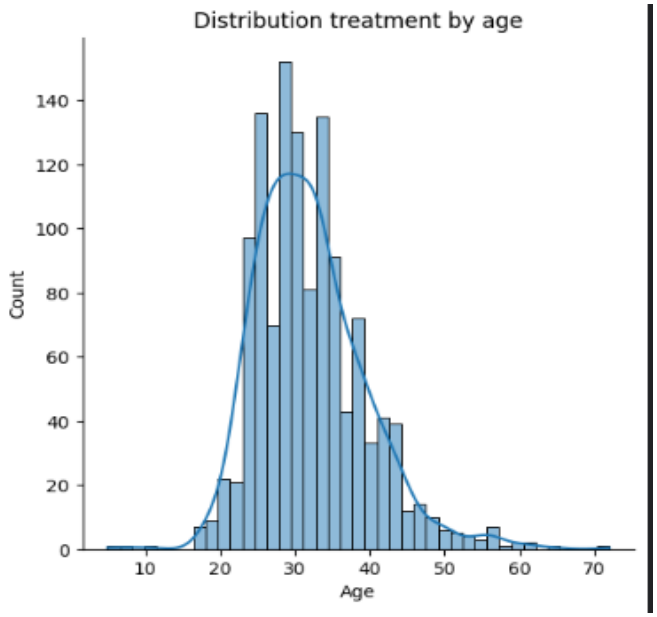
```
#We had a lot of nonsense answers in the Age column too
#This filtering will drop entries exceeding 100 years and those indicating
negative values.
data.drop(data[data['Age']<0].index, inplace = True)
data.drop(data[data['Age']>99].index, inplace = True)

print(data['Age'].unique())
```

```
[37 44 32 31 33 35 39 42 23 29 36 27 46 41 34 30 40 38 50 24 18 28 26 22
 19 25 45 21 43 56 60 54 55 48 20 57 58 47 62 51 65 49  5 53 61  8 11 72]
```

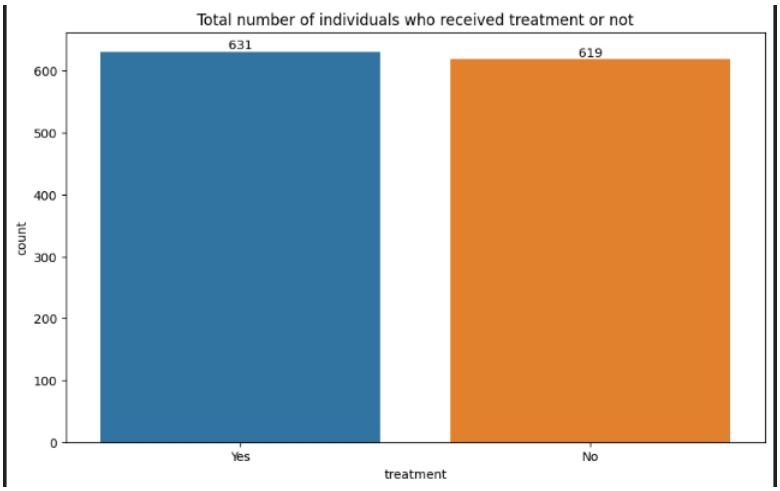
```
#Let's see the Age distribution in this dataset.
plt.figure(figsize = (10,6))
age_range_plot = sns.countplot(data = data, x = 'Age');
age_range_plot.bar_label(age_range_plot.containers[0]);
plt.xticks(rotation=90);
```




```
#In this plot moreover on Age distribution we can see treatment
distribution by age
plt.figure(figsize=(10, 6));
sns.displot(data['Age'], kde = 'treatment');
plt.title('Distribution treatment by age');
```




```
#In this plot We can see Total number of individuals who received treatment
or not.
plt.figure(figsize = (10,6));
treat = sns.countplot(data = data,  x = 'treatment');
treat.bar_label(treat.containers[0]);
plt.title('Total number of individuals who received treatment or not');
```




```
#Let's check Basic statistical info
data.describe()
```

	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	no_employees	remote_work	tech_company	...	anonymity	leave	mental_health_consei
count	1250.00000	1250.00000	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000	...	1250.000000	1250.000000	1250.
mean	32.02400	0.81760	37.792800	0.114400	0.390400	0.504800	2.128000	2.786400	0.298400	0.820000	...	0.648000	1.410400	0.
std	7.38408	0.42388	13.334981	0.318424	0.488035	0.500177	1.165806	1.738733	0.457739	0.384341	...	0.909482	1.509634	0.
min	5.00000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.
25%	27.00000	1.00000	42.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	...	0.000000	0.000000	0.
50%	31.00000	1.00000	45.000000	0.000000	0.000000	1.000000	3.000000	3.000000	0.000000	1.000000	...	0.000000	1.000000	1.
75%	36.00000	1.00000	45.000000	0.000000	1.000000	1.000000	3.000000	4.000000	1.000000	1.000000	...	2.000000	2.000000	1.
max	72.00000	2.00000	46.000000	1.000000	1.000000	1.000000	3.000000	5.000000	1.000000	1.000000	...	2.000000	4.000000	2.

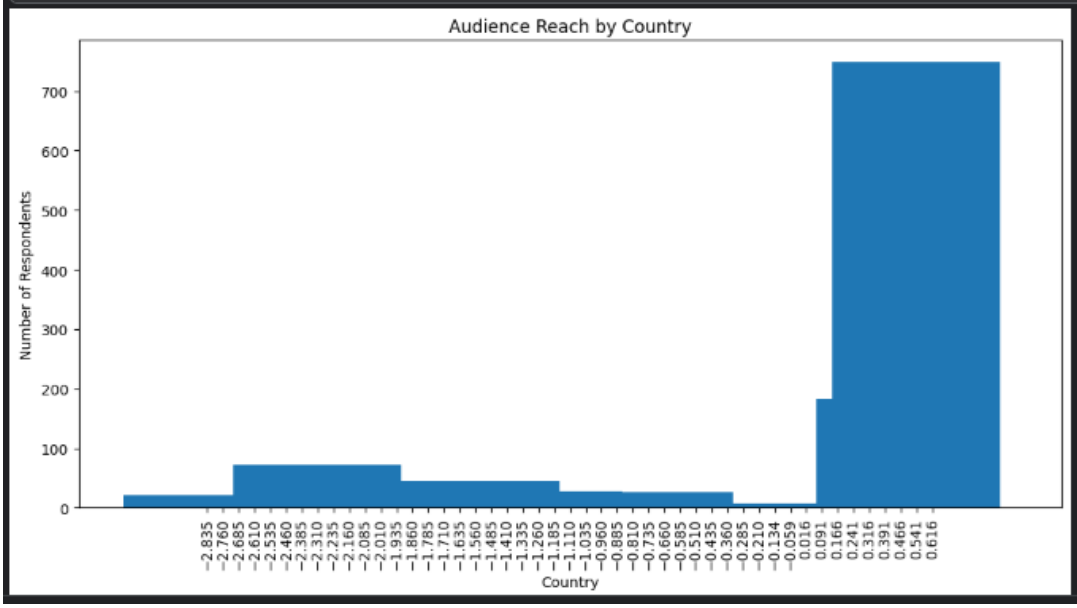
So far we have finished the data cleaning and preprocessing phase now let’s jump into our main agenda in this analysis

Audience reach

To measure audience reach, we are going to analyze the distribution of respondents by country and state. Visualize this using maps or bar charts.

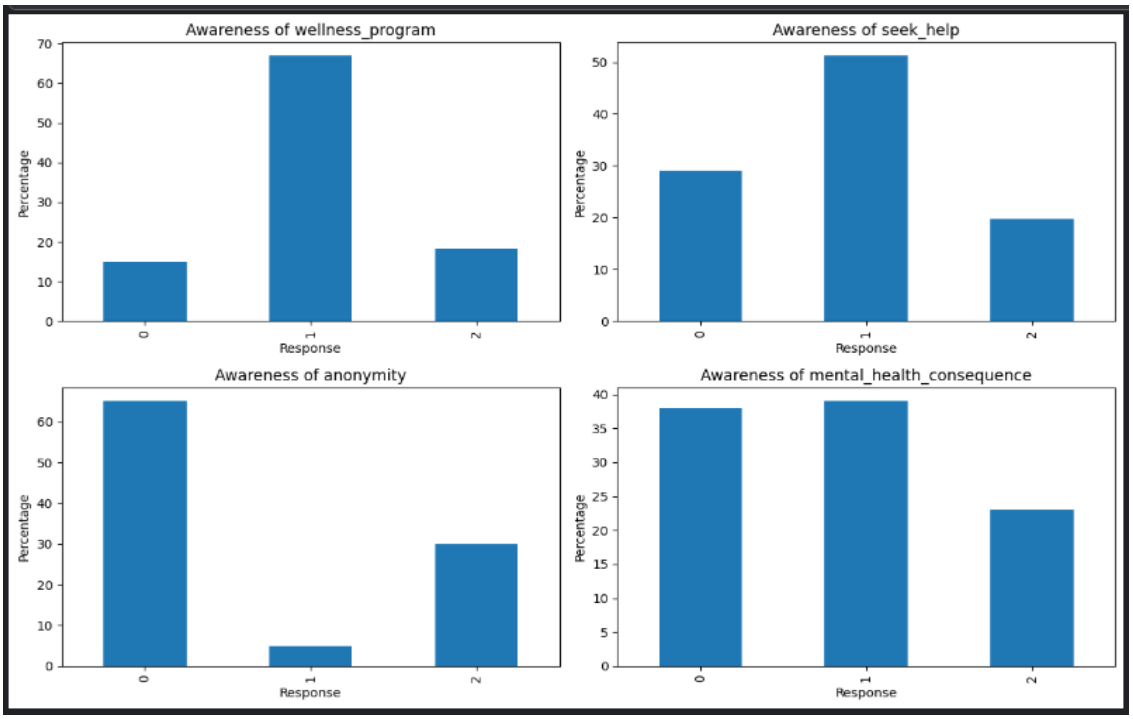
Code:

```
# Count the number of respondents per country
country_counts = data['Country'].value_counts()

# Create a bar chart with country names on the x-axis
plt.figure(figsize=(12, 6))
plt.bar(country_counts.index, country_counts.values)
plt.xlabel("Country")
plt.ylabel("Number of Respondents")
plt.title("Audience Reach by Country")

# Set the x-tick labels to be country names
plt.xticks(country_counts.index, rotation=90)

plt.show()
```

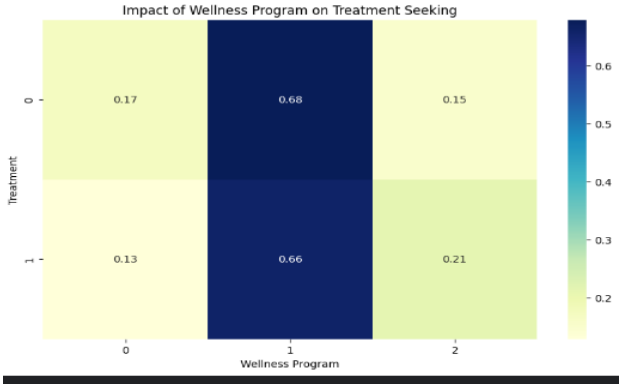



```
awareness_columns = ["wellness_program", "seek_help", "anonymity", "mental_health_consequence"]

awareness_counts = {}
total_respondents = len(data)

for column in awareness_columns:
    counts = data[column].value_counts()
    awareness_counts[column] = counts / total_respondents * 100

awareness_df = pd.DataFrame(awareness_counts)
```

```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 8))

for i, column in enumerate(awareness_columns):
    ax = axes[i // 2, i % 2]
    awareness_df[column].plot(kind="bar", ax=ax)
    ax.set_title(f"Awareness of {column}")
    ax.set_ylabel("Percentage")
    ax.set_xlabel("Response")

plt.tight_layout()
plt.show()
```




```python
# Create a subset of relevant columns
columns_of_interest = ["wellness_program", "seek_help", "anonymity", "treatment"]
subset_data = data[columns_of_interest]

# Replace missing values (if any) with a default value (e.g., "Not Specified")
subset_data.fillna("Not Specified", inplace=True)

# Create a contingency table to calculate percentages
contingency_table = pd.crosstab(subset_data["treatment"], subset_data["wellness_program"], normalize="index")

# Visualize the impact of wellness_program on treatment-seeking
plt.figure(figsize=(10, 6))
sns.heatmap(contingency_table, annot=True, cmap="YlGnBu")
plt.title("Impact of Wellness Program on Treatment Seeking")
plt.xlabel("Wellness Program")
plt.ylabel("Treatment")
plt.show()
```



Conclusion

In conclusion, we provide a comprehensive recap of the project's objectives and the accomplishments achieved. We discuss lessons learned, challenges encountered, and the overall significance of the project in the context of data analytics.