

Public Health Awareness Project

Phase 3 Submission

In Phase 3 of the project, the loading and preprocessing the dataset has been done. Along with that, we have also started building the public health awareness campaign analysis using IBM Cognos for visualization.

We have defined the analysis objectives and collect campaign data from the source shared, processed and cleaned the collected data and ensured its quality and accuracy.

ANALYSIS OBJECTIVES

1. Audience Reach:

a. Assess the extent of campaign reach within the tragediennes:

- Identify the total number of individuals or entities reached by the campaign, such as the number of social media followers, email subscribers, or event attendees.

- Measure reach over time to understand whether the campaign reached its target audience continuously or experienced spikes during specific events or promotions.

b. Measure geographical and demographic coverage:

- Use demographic data to determine the age, gender, education level, income, and other characteristics of the audience reached.

- Analyze geographical data to assess where the audience is located, whether locally, nationally, or internationally.

- Identify any gaps in the demographic or geographic coverage to tailor future campaigns to reach underserved populations.

2. Awareness Levels:

a. Evaluate changes in awareness levels before and after the campaigns.

- Conduct surveys or collect data to measure the baseline level of awareness regarding the public health issue before launching the campaign.
- After the campaign, reevaluate awareness levels to measure the campaign's impact in terms of increased awareness.

b. Identify the most successful messages or topics:

- Analyze which campaign messages, slogans, or content were the most effective in increasing awareness.
- Determine which communication channels (e.g., social media, email newsletters, educational materials) were most effective in delivering these messages.

3. Campaign Impact:

a. Quantify the overall impact on public health behavior and knowledge:

- Use surveys, data collection, or interviews to assess behavioral changes, such as an increase in vaccination rates, adoption of healthier habits, or better understanding of a health issue.
- Measure changes in knowledge, such as increased awareness of symptoms, preventive measures, or available resources related to the public health issue.

b. Analyze correlations between engagement metrics and behavioral changes:

- Explore how campaign engagement metrics (e.g., social media likes, shares, comments, click-through rates) relate to actual behavioral changes or increases in knowledge.
- Investigate whether higher engagement with campaign content correlates with a greater impact on public health behavior and knowledge.

DATA PROCESSING AND CLEANING

Type your text

Code is uploaded in the github repository

(The following have been done in google collab using Python3)

In this process, the following have been performed:-

- 1) Importing the required python libraries such as pandas and numpy.
- 2) Loading the dataset into a dataframe. The dataset has been provided in the following link : <https://www.kaggle.com/datasets/osmi/mental-health-in-tech-survey>
- 3) Addressing missing values.
- 4) Replacing null values when necessary.
- 5) Normalising data which are inconsistent. For eg: male, Male ,M all refer to the same types. Making all of them as male has been done.

Google collab link for entire code :

<https://colab.research.google.com/drive/1TZwHowM1ktUcpnUmxIhXg37K3rVk1tdo?usp=sharing>

Visualization Using IBM Cognos

Visualization of the processed and cleaned data has been done using IBM Cognos.

IBM Cognos Link :

https://us3.ca.analytics.ibm.com/bi/?perspective=dashboard&pathRef=.my_folders%2Fdashboard&action=view&mode=dashboard

BOTH THE PYTHON CODE AND VISUALIZATION DOCUMENT HAVE BEEN INCLUDED BELOW.

▼ Import necessary libraries

```
#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')
```

Successfully imported

▼ Read Dataset

[+ Code](#)
[+ Text](#)

```
#Reading data
data = pd.read_csv('survey.csv')
data.head()
```

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No

5 rows × 27 columns

▼ Preprocessing and Cleaning dataset

```
#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

#Look at what is in the 'Work_interfere' column to choose a suitable method to fill nan values.

```
data['work_interfere'].unique()
```

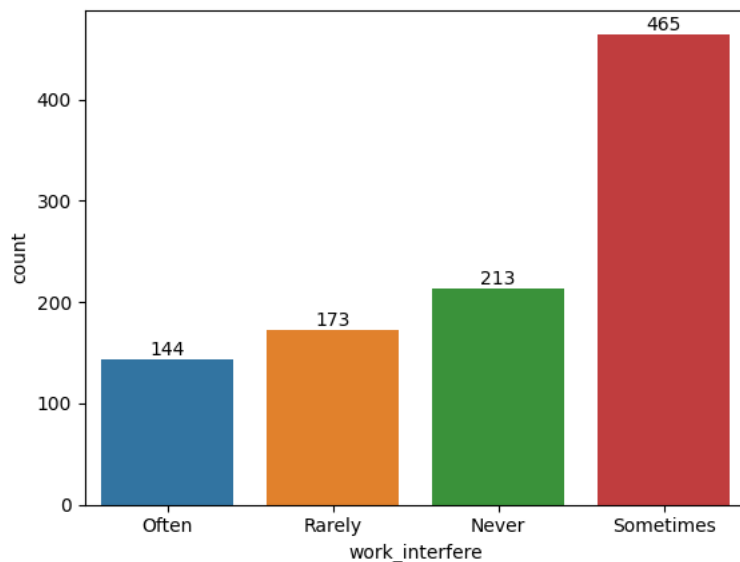
```
array(['Often', 'Rarely', 'Never', 'Sometimes', nan], dtype=object)
```

#Plot **work_interfere**

```
ax = sns.countplot(data = data , x = 'work_interfere');
```

#Add the value of each parameter 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)))
```

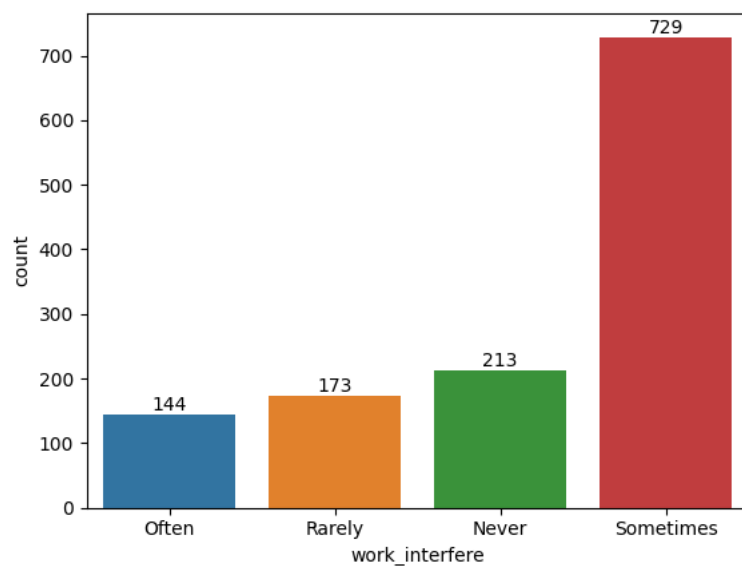
```
data['self_employed'] = np.ravel(SimpleImputer(strategy = 'most_frequent').fit_transform(data['self_employed'].values.reshape(-1,1)))
```

```
data.head()
```

	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	no_employees	remote_work	tech_company	...	anon
0	37	Female	United States	No	No	Yes	Often	6-25	No	Yes	...	
1	44	M	United States	No	No	No	Rarely	More than 1000	No	No	...	Don'
2	32	Male	Canada	No	No	No	Rarely	6-25	No	Yes	...	Don'
3	31	Male	United Kingdom	No	Yes	Yes	Often	26-100	No	Yes	...	
4	31	Male	United States	No	No	No	Never	100-500	Yes	Yes	...	Don'

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)
```

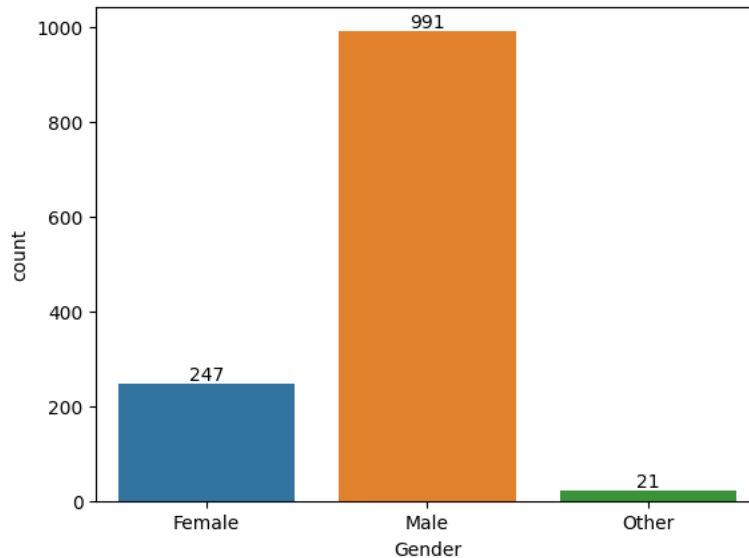
```
print(data['Gender'].unique())
```

```
['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('There 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()
```

```
array([ 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,
        9999999999, 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.
```

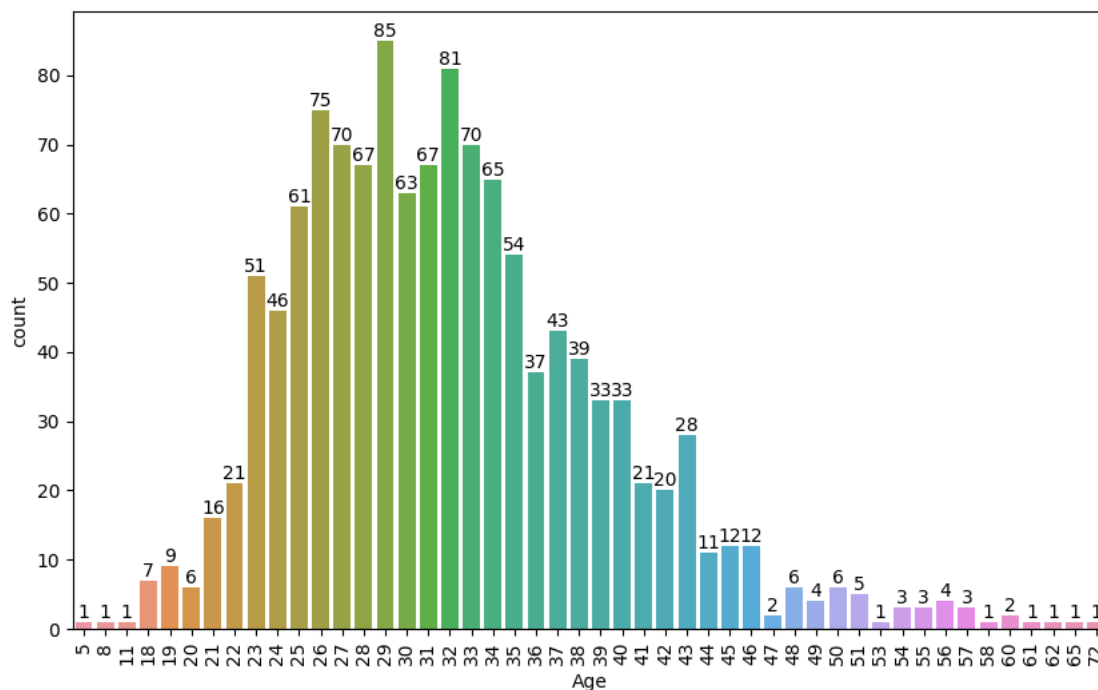
```
data.drop(data[data['Age']<0].index, inplace = True)
```

```
data.drop(data[data['Age']>99].index, inplace = True)
```

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

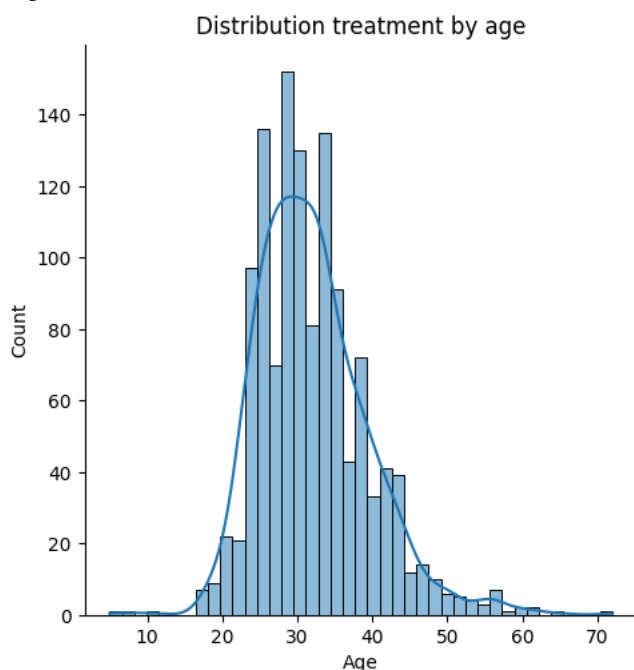
```
[37 44 32 31 33 35 39 42 23 29 36 27 46 41 34 30 40 38 50 24 18 28 26 22
 19 25 45 21 43 56 60 54 55 48 20 57 58 47 62 51 65 49  5 53 61  8 11 72]
```

```
#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);
```

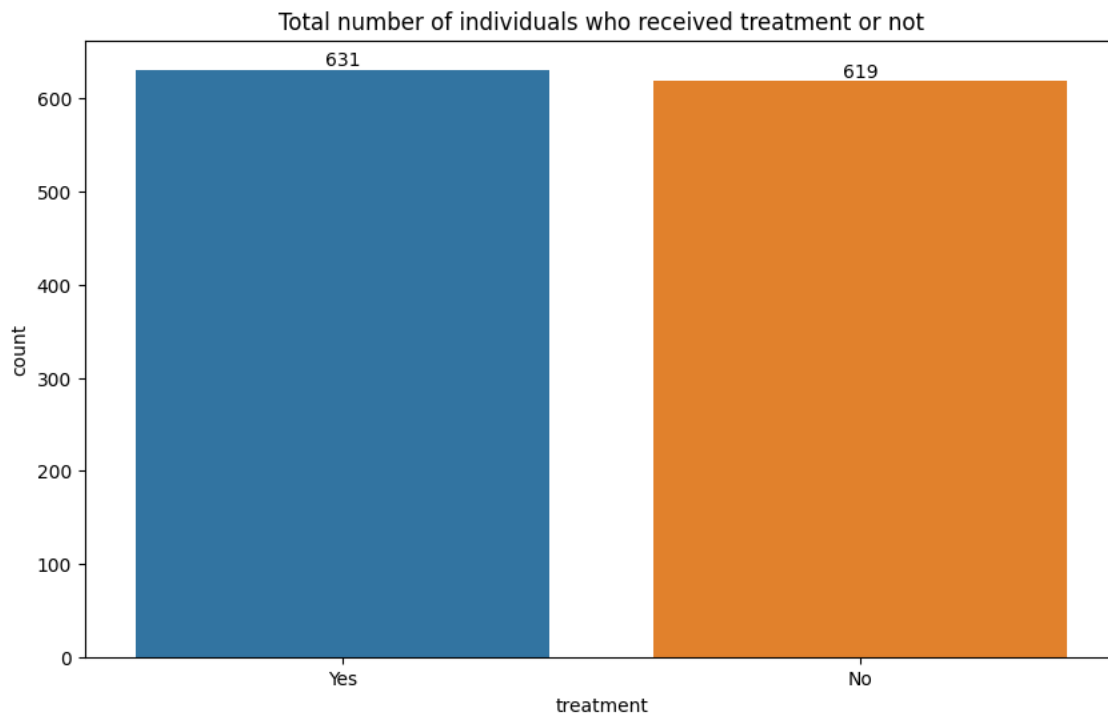


```
#In this plot moreover on Age distribution we can see treatment distribution by age
plt.figure(figsize=(10, 6));
sns.displot(data['Age'], kde = 'treatment');
plt.title('Distribution treatment by age');
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
<Figure size 1000x600 with 0 Axes>
```



```
#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');
```

```
#Check Dtypes
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1250 entries, 0 to 1258
Data columns (total 24 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Age                                    1250 non-null   int64
1   Gender                                1250 non-null   object
2   Country                               1250 non-null   object
3   self_employed                         1250 non-null   object
4   family_history                        1250 non-null   object
5   treatment                             1250 non-null   object
6   work_interfere                        1250 non-null   object
7   no_employees                          1250 non-null   object
8   remote_work                           1250 non-null   object
9   tech_company                          1250 non-null   object
10  benefits                              1250 non-null   object
11  care_options                          1250 non-null   object
12  wellness_program                      1250 non-null   object
13  seek_help                             1250 non-null   object
14  anonymity                             1250 non-null   object
15  leave                                 1250 non-null   object
16  mental_health_consequence             1250 non-null   object
17  phys_health_consequence               1250 non-null   object
18  coworkers                             1250 non-null   object
19  supervisor                            1250 non-null   object
20  mental_health_interview               1250 non-null   object
21  phys_health_interview                 1250 non-null   object
22  mental_vs_physical                    1250 non-null   object
23  obs_consequence                       1250 non-null   object
dtypes: int64(1), object(23)
memory usage: 244.1+ KB
```

```
#Use LabelEncoder to change the Dtypes to 'int'
```

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
#Make the dataset include all the columns we need to change their dtypes
```

```
columns_to_encode = ['Gender', 'Country', 'self_employed', 'family_history', 'treatment', 'work_interfere', 'no_employees',
                     'remote_work', 'tech_company', 'benefits', 'care_options', 'wellness_program',
                     'seek_help', 'anonymity', 'leave', 'mental_health_consequence', 'phys_health_interview',
                     'coworkers', 'supervisor', 'mental_health_interview', 'phys_health_interview',
                     'mental_vs_physical', 'obs_consequence']
```

```
#Write a Loop for fitting LabelEncoder on columns_to_encode
```

```
for columns in columns_to_encode:
```

```
    data[columns] = le.fit_transform(data[columns])
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1250 entries, 0 to 1258
```

```
Data columns (total 24 columns):
#   Column                               Non-Null Count  Dtype
---  -
0   Age                                   1250 non-null   int64
1   Gender                               1250 non-null   int64
2   Country                              1250 non-null   int64
3   self_employed                        1250 non-null   int64
4   family_history                       1250 non-null   int64
5   treatment                            1250 non-null   int64
6   work_interfere                       1250 non-null   int64
7   no_employees                        1250 non-null   int64
8   remote_work                          1250 non-null   int64
9   tech_company                        1250 non-null   int64
10  benefits                             1250 non-null   int64
11  care_options                         1250 non-null   int64
12  wellness_program                    1250 non-null   int64
13  seek_help                           1250 non-null   int64
14  anonymity                            1250 non-null   int64
15  leave                                1250 non-null   int64
16  mental_health_consequence           1250 non-null   int64
17  phys_health_consequence              1250 non-null   int64
18  coworkers                            1250 non-null   int64
19  supervisor                           1250 non-null   int64
20  mental_health_interview              1250 non-null   int64
21  phys_health_interview                1250 non-null   int64
22  mental_vs_physical                  1250 non-null   int64
23  obs_consequence                     1250 non-null   int64
dtypes: int64(24)
memory usage: 244.1 KB

#Let's check Standard deviation
data.describe()
```

	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	no_employees	remote_work	tech_company
count	1250.00000	1250.00000	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000
mean	32.02400	0.81760	37.792800	0.114400	0.390400	0.504800	2.128000	2.786400	0.298400	0.298400
std	7.38408	0.42388	13.334981	0.318424	0.488035	0.500177	1.165806	1.738733	0.457739	0.457739
min	5.00000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	27.00000	1.00000	42.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000
50%	31.00000	1.00000	45.000000	0.000000	0.000000	1.000000	3.000000	3.000000	0.000000	0.000000
75%	36.00000	1.00000	45.000000	0.000000	1.000000	1.000000	3.000000	4.000000	1.000000	1.000000
max	72.00000	2.00000	46.000000	1.000000	1.000000	1.000000	3.000000	5.000000	1.000000	1.000000

8 rows × 24 columns

```
from sklearn.preprocessing import MaxAbsScaler, StandardScaler

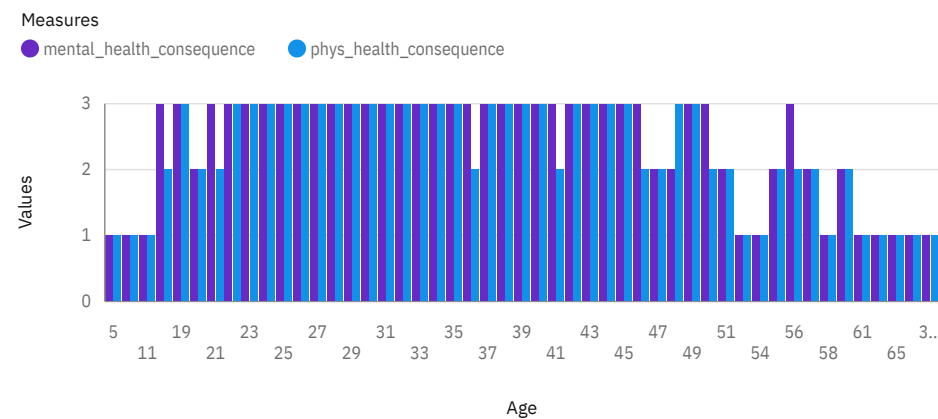
data['Age'] = MaxAbsScaler().fit_transform(data[['Age']])
data['Country'] = StandardScaler().fit_transform(data[['Country']])
data['work_interfere'] = StandardScaler().fit_transform(data[['work_interfere']])
data['no_employees'] = StandardScaler().fit_transform(data[['no_employees']])
data['leave'] = StandardScaler().fit_transform(data[['leave']])

data.describe()
```

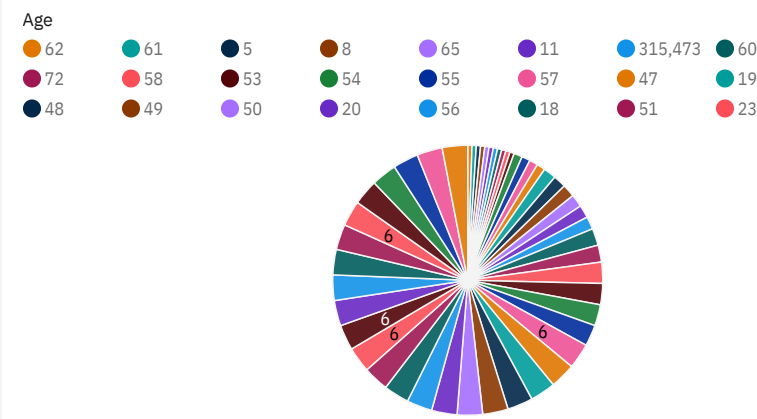
	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	no_employees	remote_work	tech_company
count	1250.000000	1250.00000	1.250000e+03	1250.000000	1250.000000	1250.000000	1.250000e+03	1.250000e+03	1250.000000	1250.000000
mean	0.444778	0.81760	3.979039e-17	0.114400	0.390400	0.504800	-1.193712e-16	-1.705303e-17	0.298400	0.298400
std	0.102557	0.42388	1.000400e+00	0.318424	0.488035	0.500177	1.000400e+00	1.000400e+00	0.457739	0.457739
min	0.069444	0.00000	-2.835244e+00	0.000000	0.000000	0.000000	-1.826077e+00	-1.603187e+00	0.000000	0.000000
25%	0.375000	1.00000	3.156273e-01	0.000000	0.000000	0.000000	-9.679583e-01	-1.027826e+00	0.000000	0.000000
50%	0.430556	1.00000	5.406895e-01	0.000000	0.000000	1.000000	7.482798e-01	1.228972e-01	0.000000	0.000000
75%	0.500000	1.00000	5.406895e-01	0.000000	1.000000	1.000000	7.482798e-01	6.982587e-01	1.000000	1.000000
max	1.000000	2.00000	6.157103e-01	1.000000	1.000000	1.000000	7.482798e-01	1.273620e+00	1.000000	1.000000

8 rows × 24 columns

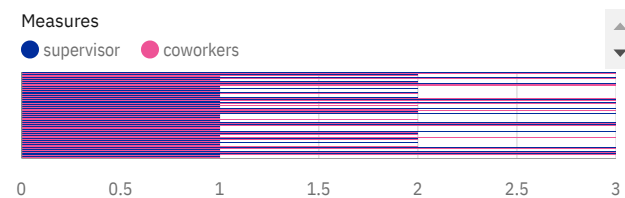
mental_health_consequence and phys_health_consequence by Age



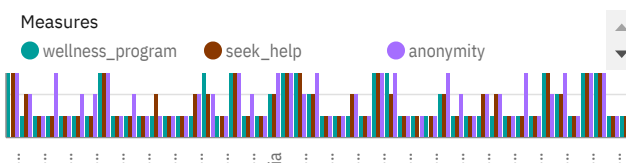
no_employees by Age



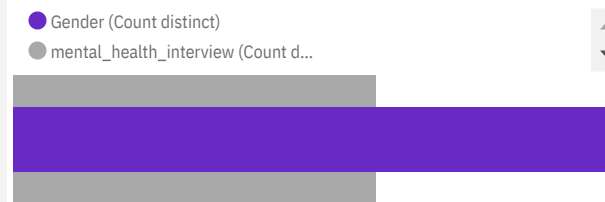
supervisor and coworkers by Country



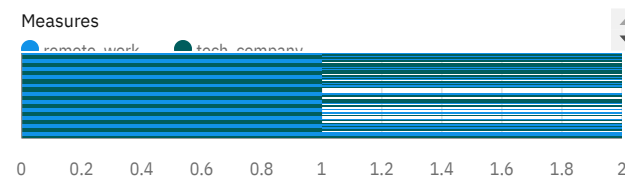
wellness_program, seek_help and anonymity by Country



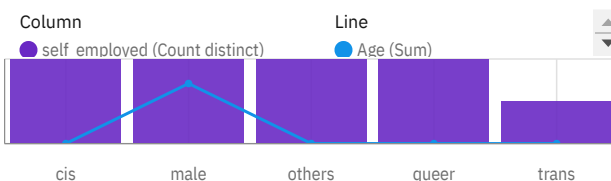
Gender, mental_health_interview, phys_health_interview



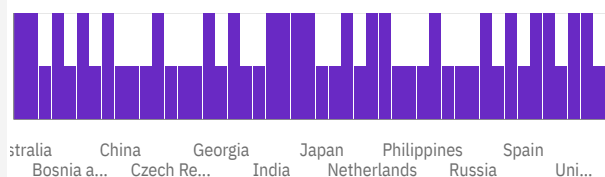
remote_work and tech_company by state and Country



Age and self_employed by Gender



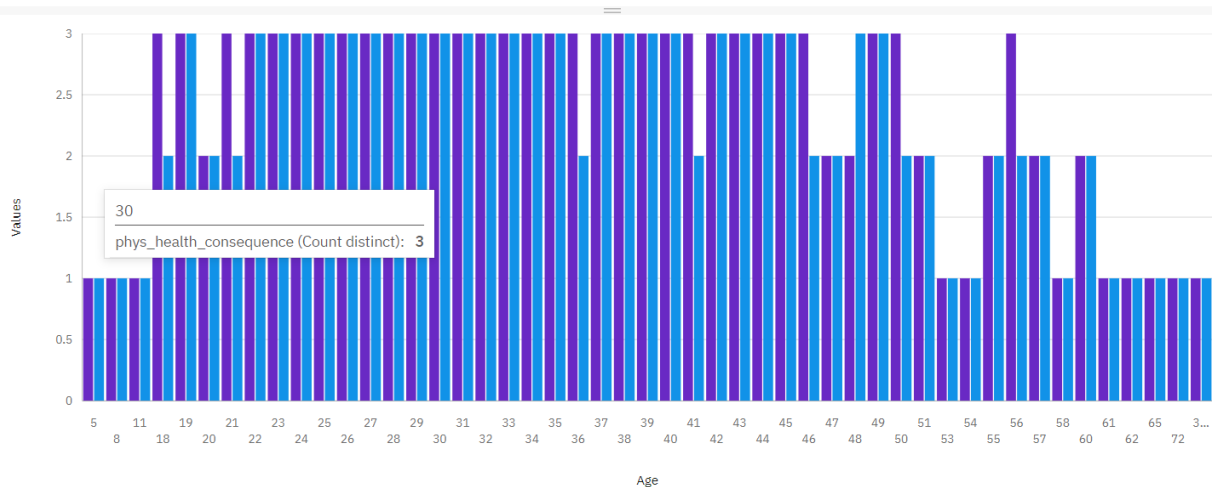
family_history by Country



mental_health_consequence and phys_health_consequence by Age

Measures

● mental_health_consequence ● phys_health_consequence



Narrative insights

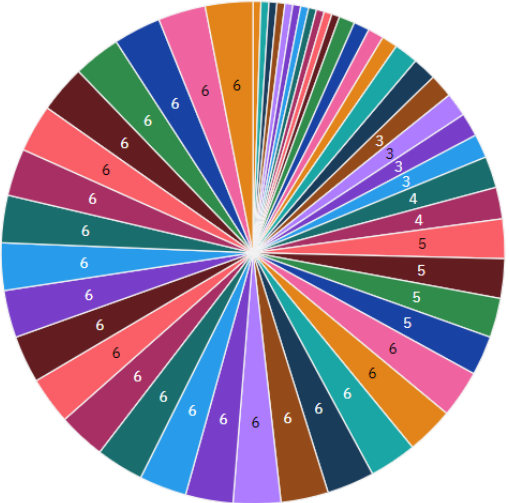
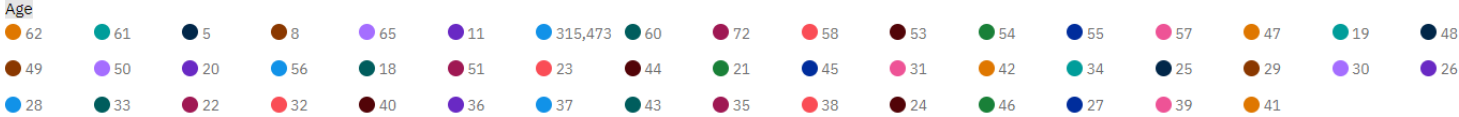
Suggested insights (3) ⓘ

Age 26 has the highest **phys_health_consequence** due to **Country United States**.

Age 18 has the highest **Count distinct mental_health_consequence** but is ranked #26 in **Count distinct phys_health_consequence**.

Country United States has the highest **phys_health_consequence** at 99, out of which **Age 22** contributed the most at 3.

no_employees by Age

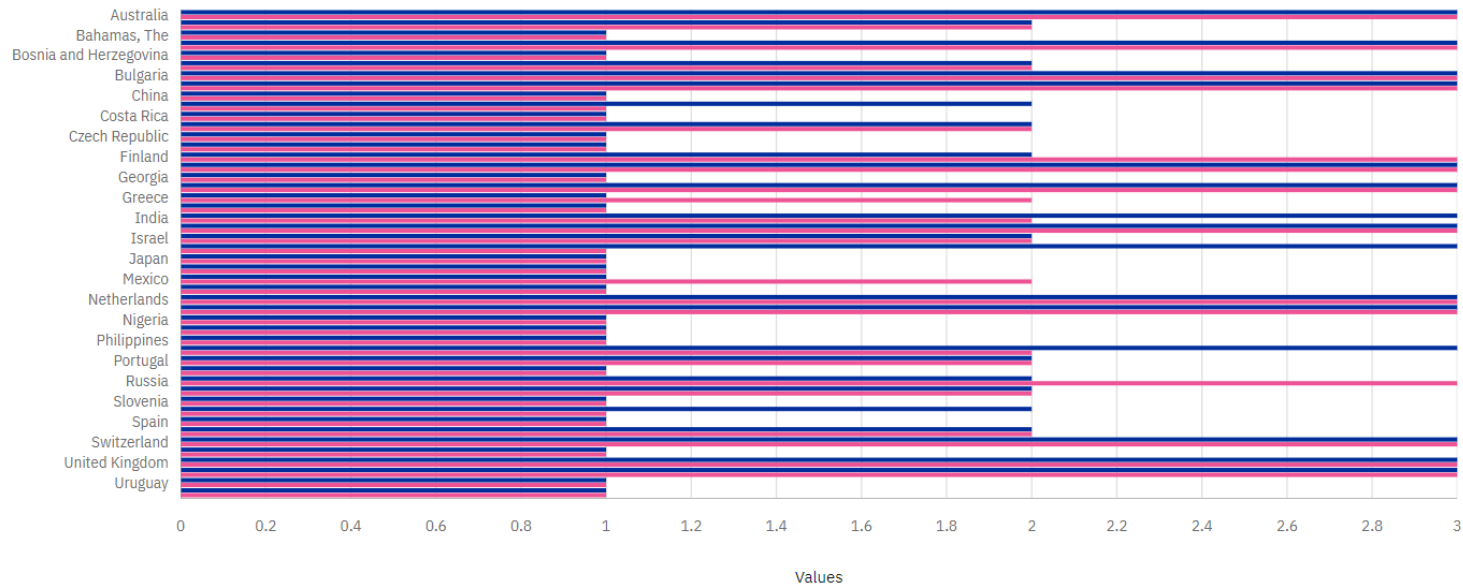


supervisor and coworkers by Country

Measures

● supervisor ● coworkers

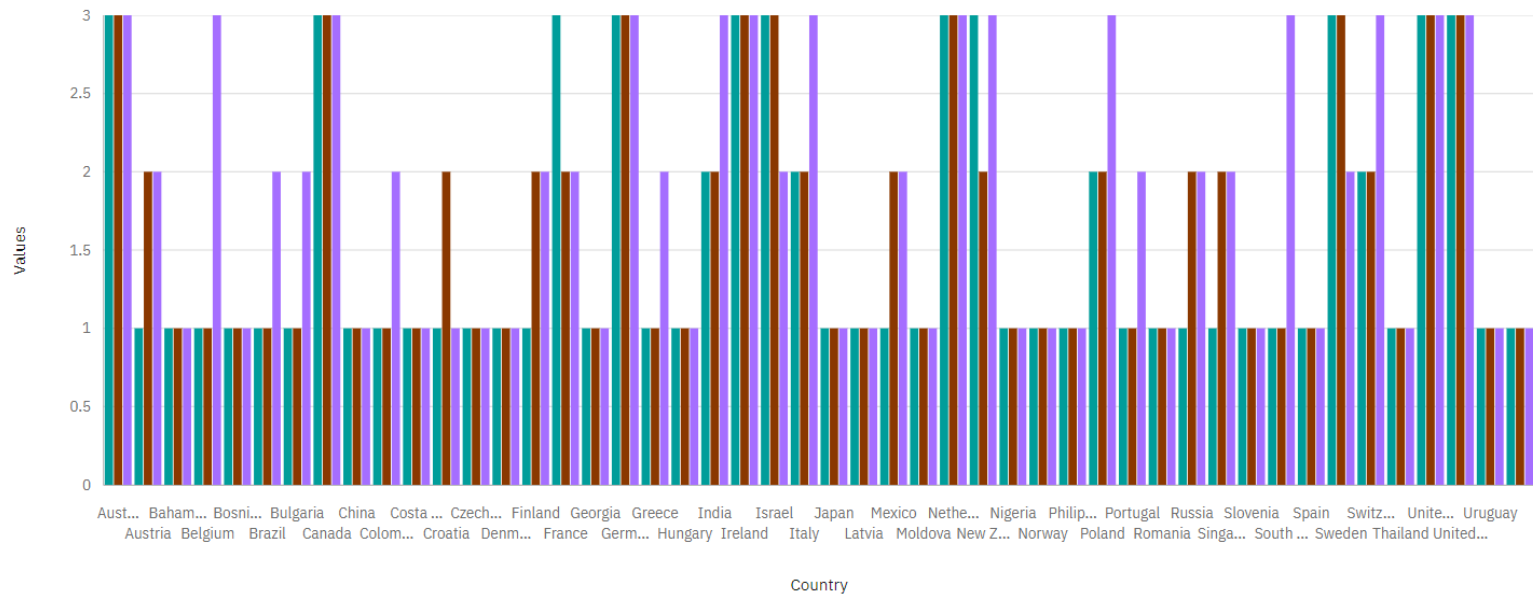
Country



wellness_program, seek_help and anonymity by Country

Measures

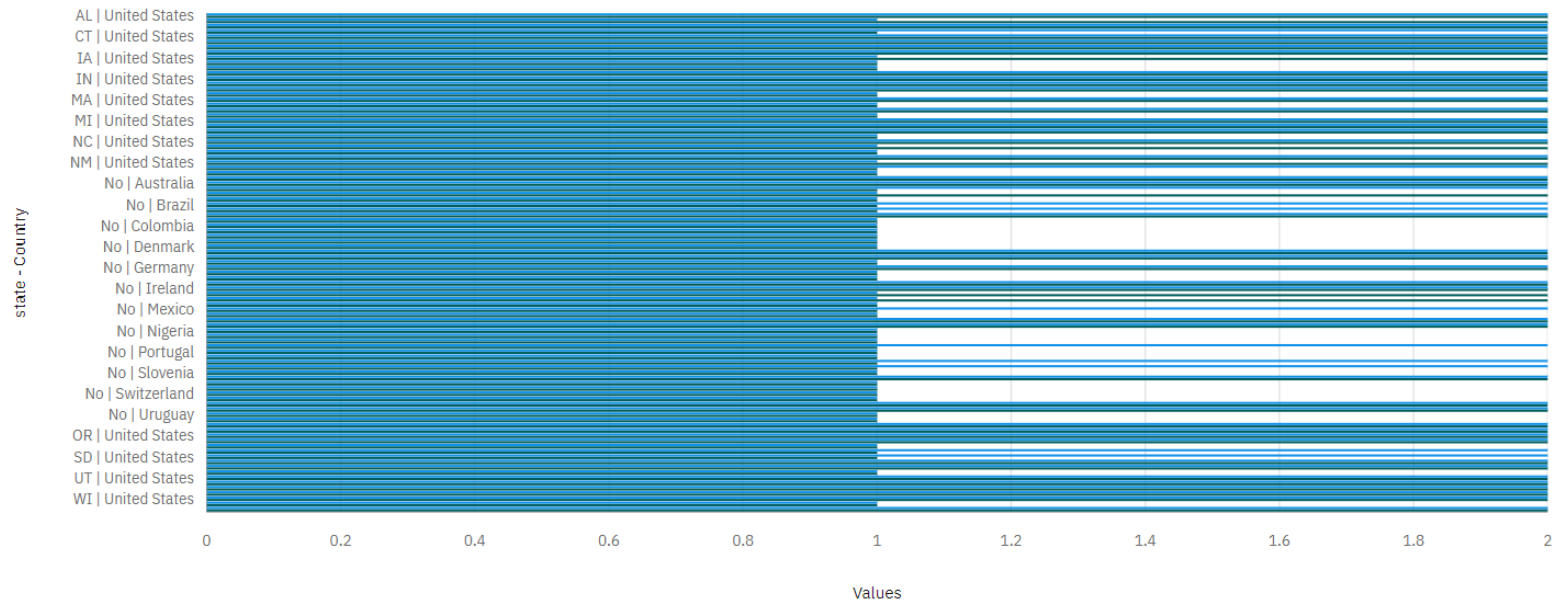
wellness_program seek_help anonymity



remote_work and tech_company by state and Country

Measures

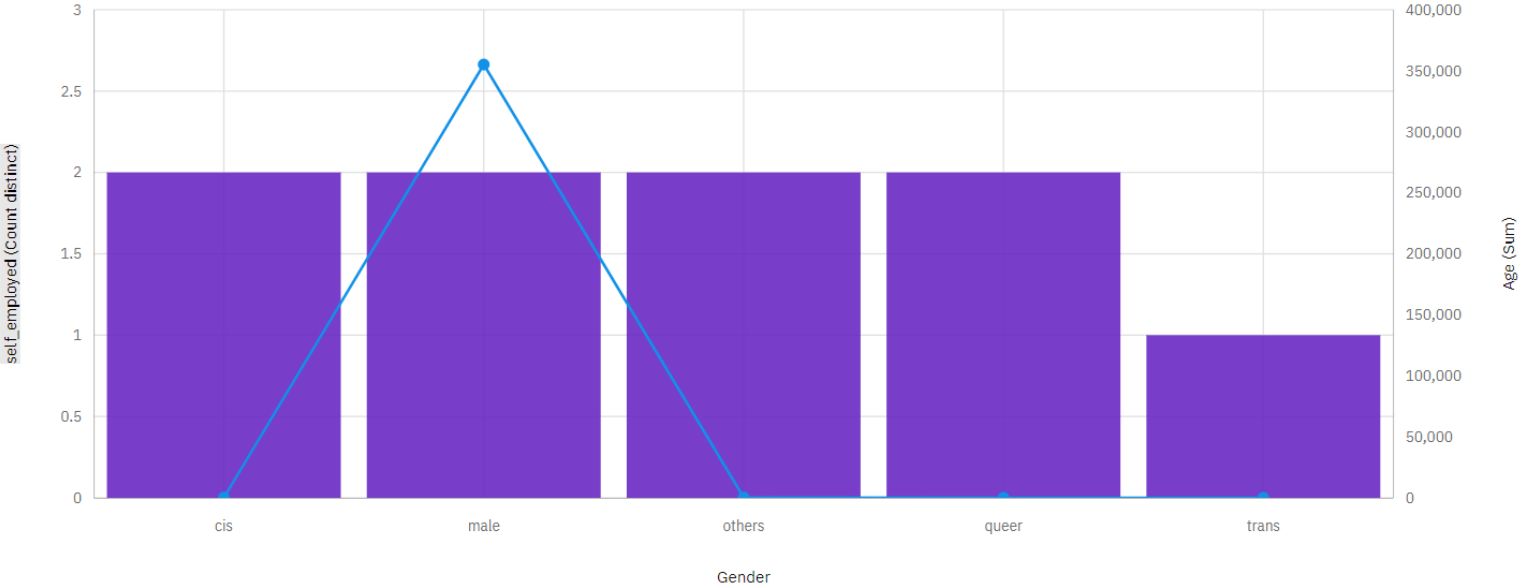
remote_work tech_company



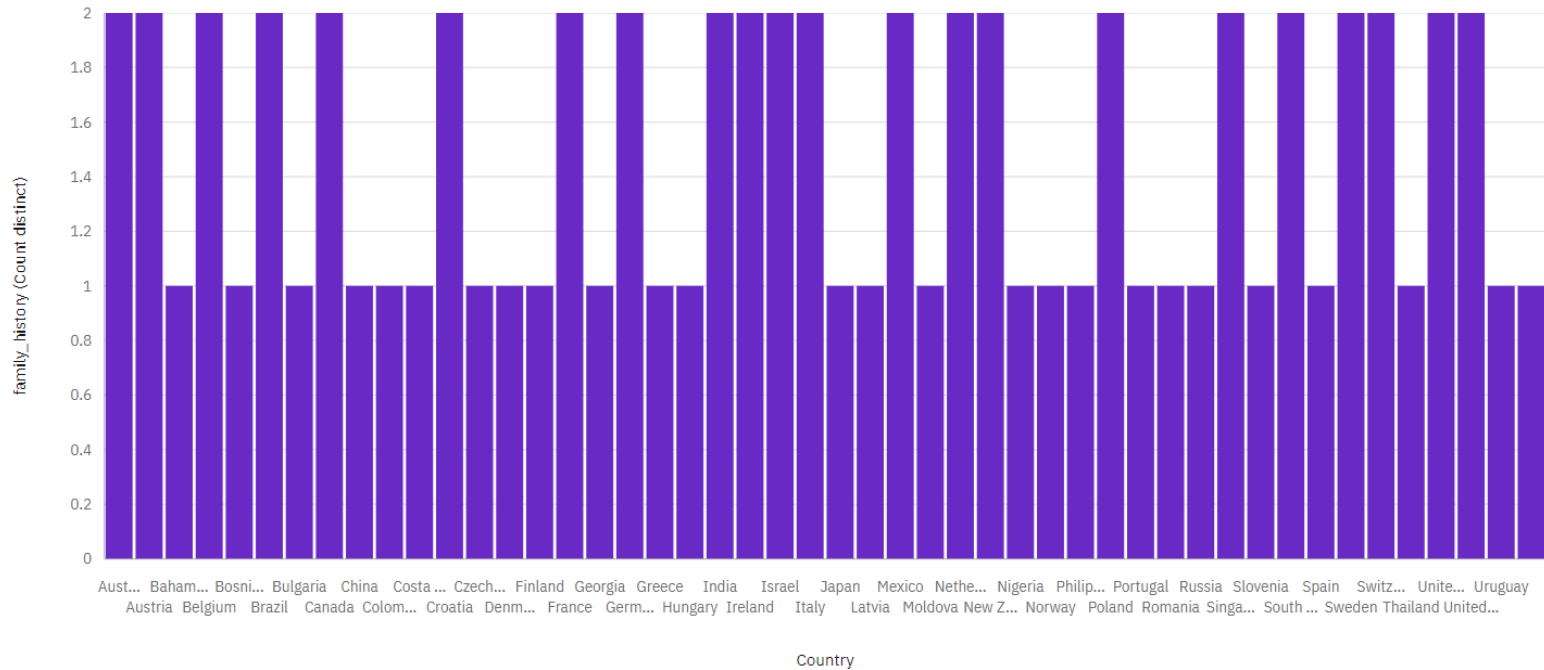
Age and self_employed by Gender

Column Line

● self_employed (Count distinct) ● Age (Sum)



family_history by Country



TEAM MEMBERS:

1. RUKSHANA J - 2021115086
2. SWATHI K - 2021115116
3. THIRUMURUGAN K - 2021115117
4. VASUMATHY V - 2021115118
5. KAVIRAM R -2021115332