PUBLIC HEALTH AWARENESS PROJECT

PROJECT DEFINITION:

The project focuses on evaluating the effectiveness of public health awareness campaigns by analyzing campaign data. The primary objective is to measure the impact of these campaigns in reaching the target audience and increasing awareness, providing insights for future strategies. The project involves defining specific analysis objectives, collecting campaign data, creating visualizations, and integrating code for enhanced data analysis.

DESIGN THINKING

ANALYSIS OBJECTIVES:

1. Audience Reach:

In the context of this public health awareness project, assessing the extent of campaign reach within the target audience, referred to as the "tragediennes," is a critical first step. This involves a comprehensive examination of how effectively the campaigns are penetrating the intended demographic. The evaluation goes beyond mere numbers and delves into the quality and depth of engagement.

2. Awareness Levels:

understanding the changes in awareness levels before and after the campaigns is a key component of the analysis. This objective is multifaceted. It involves conducting surveys and data collection both before and after the campaigns to gauge the shift in awareness within the tragedienne audience.

3. Campaign Impact:

Quantifying the overall impact on public health behavior and knowledge is at the core of this analysis objective. It involves looking beyond surface-level metrics and delving into the behavioral changes and knowledge enhancement resulting from the campaigns. The objective is to measure the tangible and intangible effects of the campaigns on the tragédienne population.

DATA C OLLECTION

1.Engagement Metrics:

These metrics delve into the audience's interaction with campaign content. By tracking social media interactions, website visits, and overall online engagement, we gain valuable insights into how our message is resonating. Monitoring clicks, likes, shares, and comments provides a real-time gauge of the audience's response and the virality of the content.

2.Audience Demographics:

To create targeted and relevant content, collecting demographic data is essential. This includes age, gender, location, and other pertinent factors. This information enables us to tailor the campaign to suit the preferences and characteristics of the specific audience segments we aim to reach.

3.Awareness Surveys:

Conducting pre-campaign and post-campaign surveys is vital for measuring the impact of the campaign. Pre-campaign surveys establish a baseline of audience awareness and perceptions, while post-campaign surveys allow us to assess the campaign's effectiveness in raising awareness and altering perceptions. The comparative analysis of these surveys offers valuable insights into the campaign's success.

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

4.Campaign Content Analysis:

Analyzing the campaign's content is critical for ensuring alignment with campaign goals. It involves a close examination of both the textual and visual elements. By evaluating how well the messages and visuals resonate with the campaign's objectives, we can refine our content strategy to maximize its impact and relevance.

Tools and Technologies

1. IBM Cognos:

IBM Cognos is a business intelligence and analytics platform that allows organizations to create interactive dashboards and reports.

- **Data Integration**: Begin by connecting Cognos to your data sources,whether they are databases, spreadsheets, or other data stores.
- **Dashboard Creation**: Cognos provides a user-friendly interface forbuilding interactive dashboards.

- **Data Exploration:** Cognos offers tools for exploring and visualizing data,making it easier to identify patterns and trends in your dataset.

2. Python or R:

Python and R are powerful programming languages used for advanced data analysis and statistical modeling.

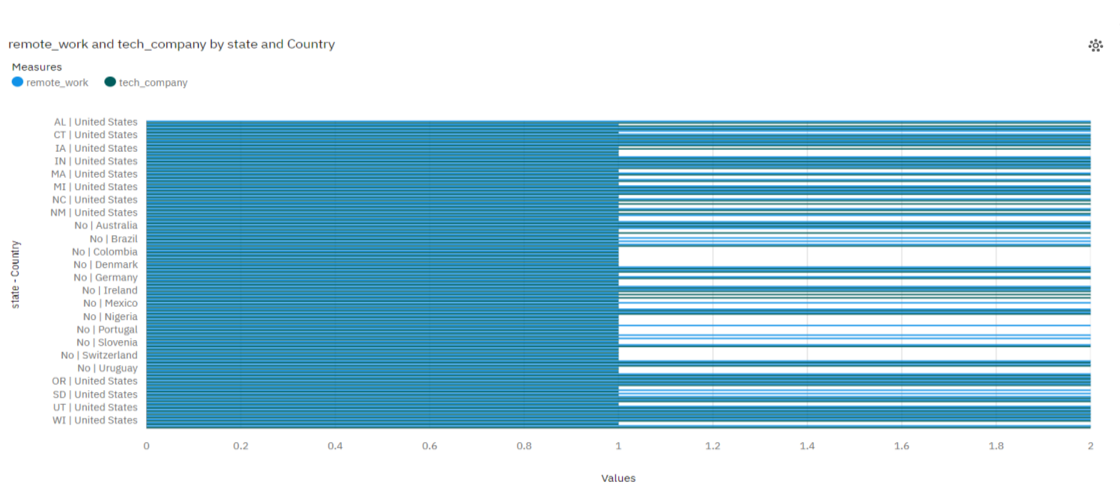
Using these tools and technologies effectively enhance your data analysisand project collaboration, making the entire process more efficient and

Productive.

VISUALIZATION STRATEGY

Dashboard Design:

★ Create interactive dashboards to present key metrics.
★ Use charts, graphs, and maps for effective communication.

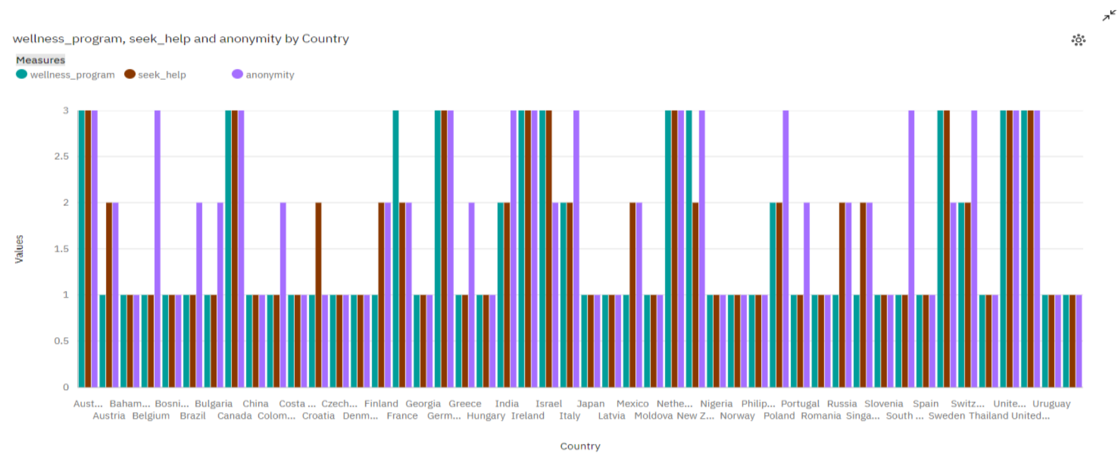


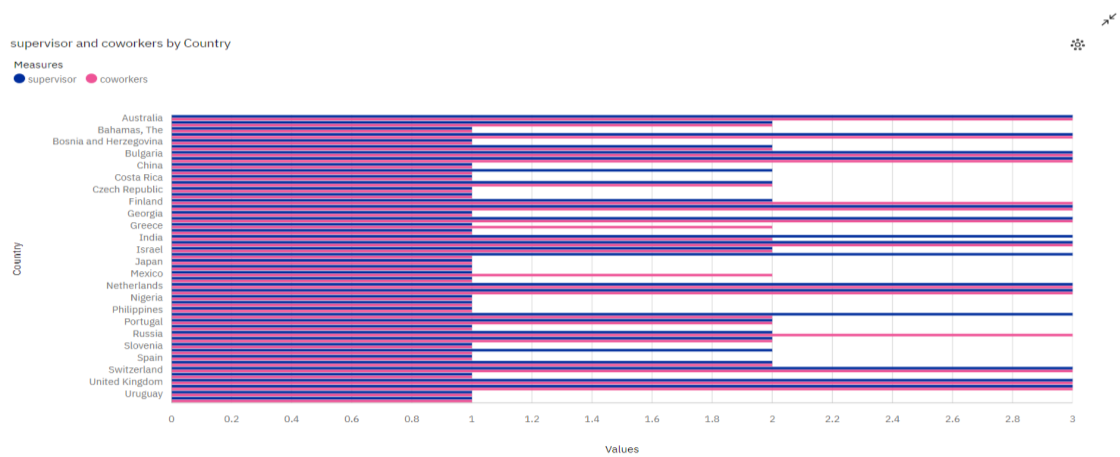
2. Temporal Analysis:

★ Develop visualizations showing the temporal evolution of

★ campaign impact.

★ Use time-series charts to illustrate changes in awareness levels.





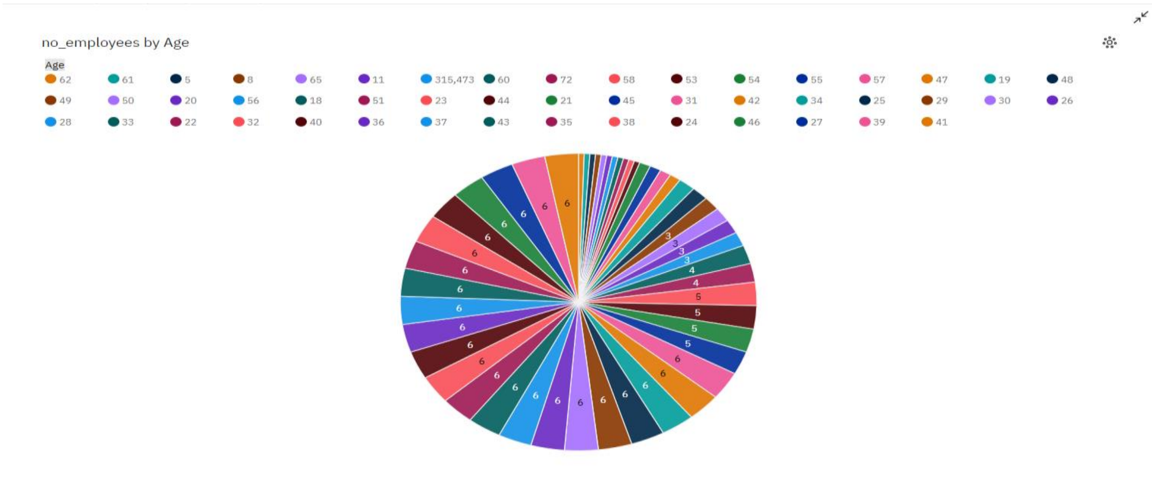
3.Demographic Breakdown:

★ Include visualizations breaking down campaign impact by

★ demographic groups.

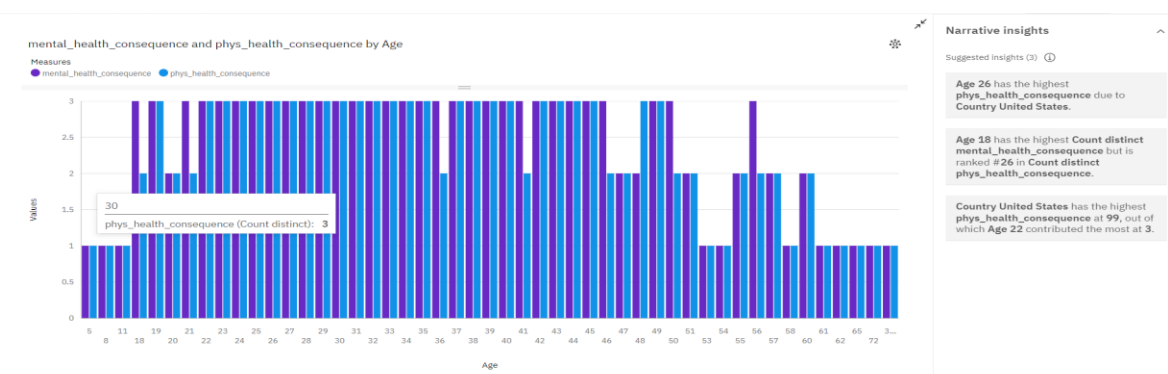
★ Utilize pie charts or bar graphs to highlight variations among

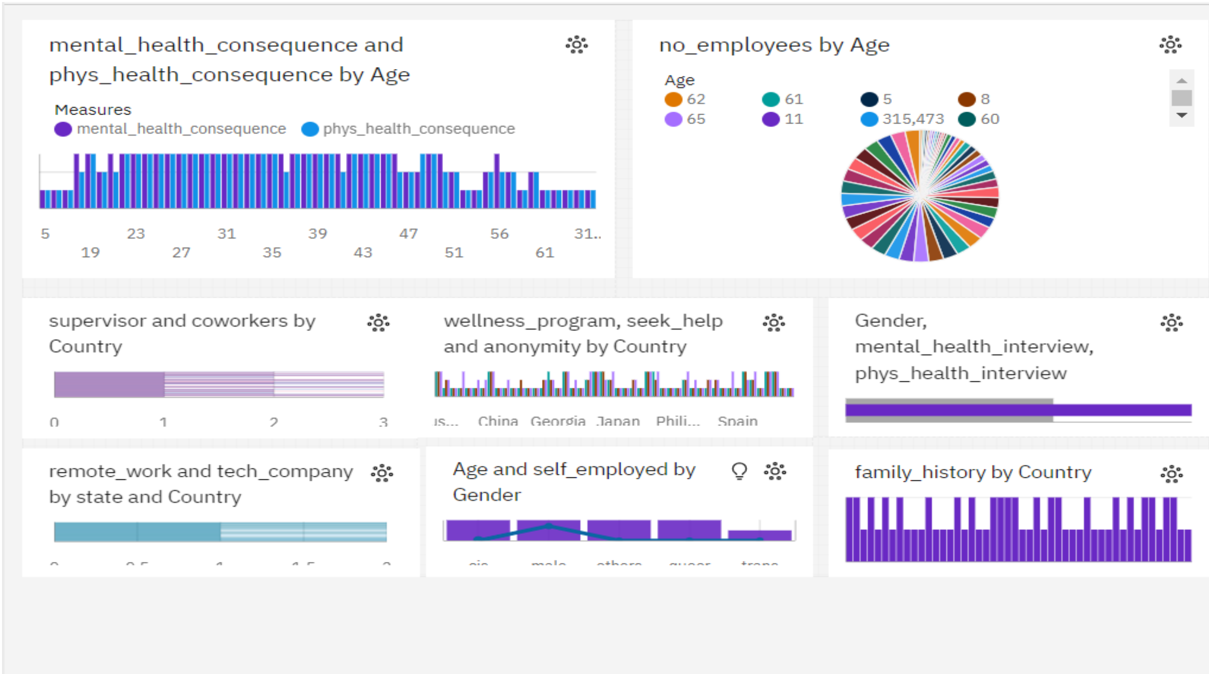
★ segments.



4.Comparison Visuals:

- ★ visualizations facilitating easy comparison between
- ★ different campaigns.
- ★ Use side-by-side charts to showcase relative effectiveness.





CODE INTEGRATION

1.Data Cleaning:

♦ Implement code for cleaning and preprocessing raw data.

♦ Address missing values, outliers, and data anomalies.

2.Transformational Analysis:

♦ Use code for complex data transformations and feature engineering.

♦ Implement algorithms for deriving additional insights.

3. Statistical Analysis:

♦ Apply statistical tests and analyses using code to identify patterns.

♦ Perform hypothesis testing to validate observed effects.

4.Automation:

♦ Integrate code for automating repetitive tasks in data processing.

♦ Ensure scalability for future campaigns and analyses.

```
df = pd.read_csv('/kaggle/input/telco-customer-churn/WA_Fn-UseC_-Telco-Customer-Churn.csv')
df.head(5)
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No

5 rows × 21 columns

```
print_unique_values(df2)
```

```
gender ['Female' 'Male']
SeniorCitizen [0 1]
Partner ['Yes' 'No']
Dependents ['No' 'Yes']
tenure [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]
PhoneService ['No' 'Yes']
MultipleLines ['No' 'Yes']
OnlineSecurity ['No' 'Yes']
OnlineBackup ['Yes' 'No']
DeviceProtection ['No' 'Yes']
TechSupport ['No' 'Yes']
StreamingTV ['No' 'Yes']
StreamingMovies ['No' 'Yes']
PaperlessBilling ['Yes' 'No']
MonthlyCharges [29.85 56.95 53.85 ... 63.1  44.2  78.7 ]
TotalCharges [  29.85 1889.5   108.15 ...  346.45  306.6  6844.5 ]
Churn ['No' 'Yes']
InternetService_DSL [ True False]
InternetService_Fiber optic [False  True]
InternetService_No [False  True]
Contract_Month-to-month [ True False]
Contract_One year [False  True]
Contract_Two year [False  True]
PaymentMethod_Bank transfer (automatic) [False  True]
```

In [1]:
```python
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt


print('Successfully imported')
```

```
Successfully imported
```

In [3]:
```python
data = pd.read_csv('survey (2).csv')
data.head()
```

Out[3]:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_emplo
0	27-08-2014 11:29	37	male	United States	IL	No	No	Yes	Often	Ju
1	27-08-2014 11:29	44	male	United States	IN	No	No	No	Rarely	More
2	27-08-2014 11:29	32	male	Canada	No	No	No	No	Rarely	Ju
3	27-08-2014 11:29	31	male	United Kingdom	No	No	Yes	Yes	Often	26
4	27-08-2014 11:30	31	male	United States	TX	No	No	No	Never	100

5 rows × 27 columns

In [4]:
```python
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 no missing data in our dataset
```

In [5]:
```python
frame = pd.concat([data.isnull().sum(), data.nunique(), data.dtypes], axis = 1, sort= False)
frame
```

Out[5]:

	0	1	2
Timestamp	0	884	object
Age	0	49	int64
Gender	0	5	object
Country	0	48	object
state	0	46	object
self_employed	0	2	object
family_history	0	2	object
treatment	0	2	object
work_interfere	0	5	object
no_employees	0	6	object
remote_work	0	2	object

	0	1	2
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
obs_consequence	0	2	object
comments	0	161	object

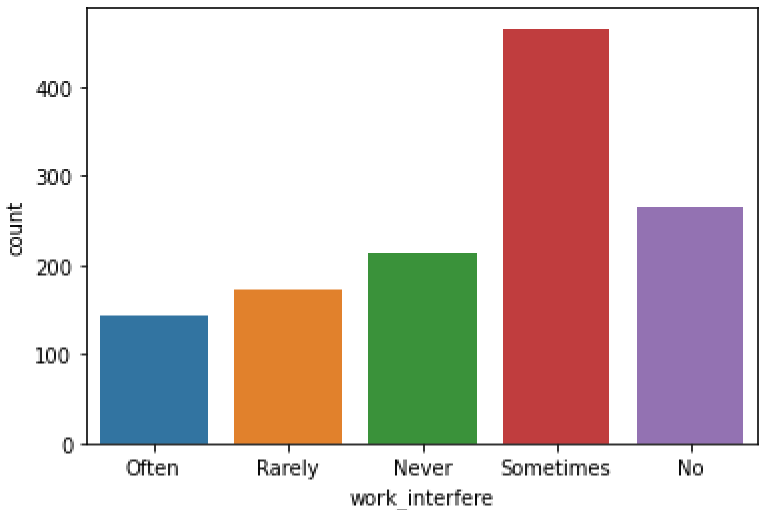
In [6]:
```
data['work_interfere'].unique()
```

Out[6]:
```
array(['Often', 'Rarely', 'Never', 'Sometimes', 'No'], dtype=object)
```

In [7]:
```
#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]);
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-7-94978d9b0cf1> in <module>
      2 ax = sns.countplot(data = data , x = 'work_interfere');
      3 #Add the value of each parametr on the Plot
----> 4 ax.bar_label(ax.containers[0]);

AttributeError: 'AxesSubplot' object has no attribute 'bar_label'
```



In [8]:

```
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[
data['self_employed'] = np.ravel(SimpleImputer(strategy = 'most_frequent').fit_transform(data['

data.head()
```

Out[8]:

	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	no_employees	remote_work
0	37	male	United States	No	No	Yes	Often	Jun-25	No
1	44	male	United States	No	No	No	Rarely	More than 1000	No
2	32	male	Canada	No	No	No	Rarely	Jun-25	No
3	31	male	United Kingdom	No	Yes	Yes	Often	26-100	No
4	31	male	United States	No	No	No	Never	100-500	Yes

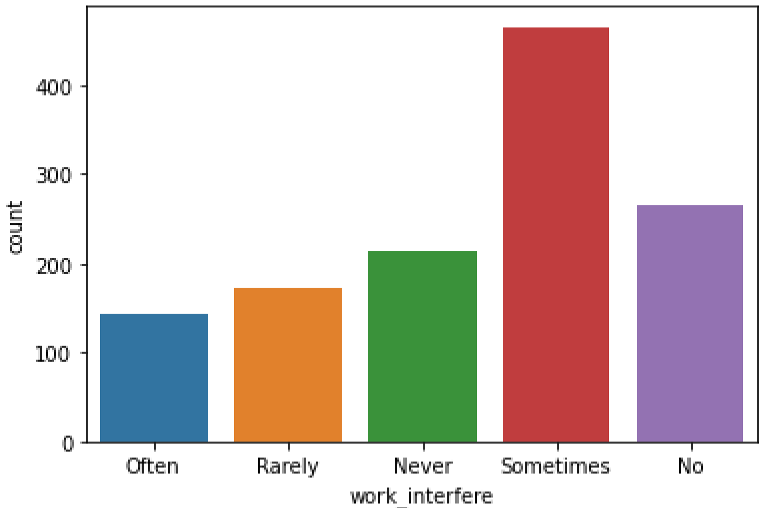
5 rows × 24 columns

In [9]:

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

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-9-43679f127d52> in <module>
      1 ax = sns.countplot(data=data, x='work_interfere');
----> 2 ax.bar_label(ax.containers[0]);

AttributeError: 'AxesSubplot' object has no attribute 'bar_label'
```



In [10]:

```
#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())
```

```
['male' 'trans' 'cis' 'queer' 'others']

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

number of unique Gender in our dataset is : 5
```

In [11]:
```
#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
```

In [12]:
```
data['Age'].unique()
```

Out[12]:
```
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, 315473,     43,     56,     60,
           54,     55,     48,     20,     57,     58,     47,     62,
           51,     65,     49,      5,     53,     61,      8,     11,
           72])
```

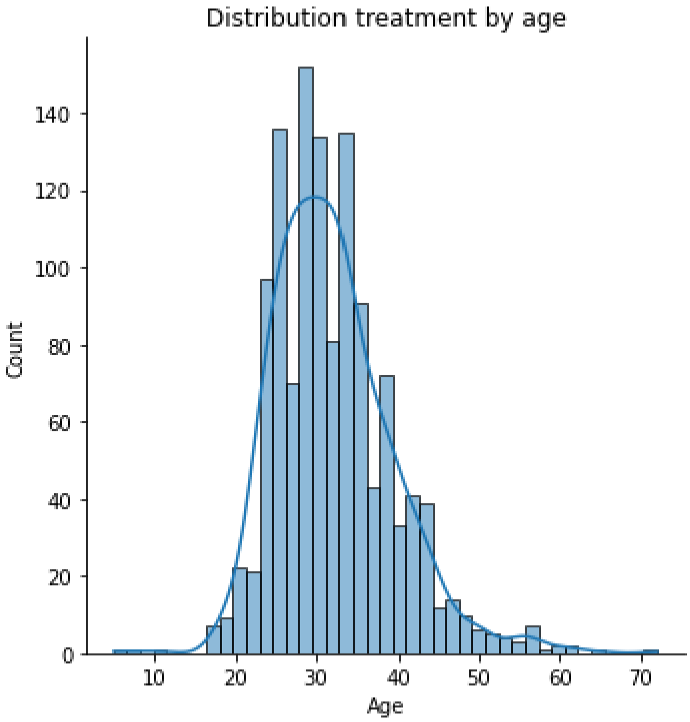
In [13]:
```
#We had a lot of nonsense answers in the Age column too
#This filtering will drop entries exceeding 100 years and those indicating negative values.
data.drop(data[data['Age']<0].index, inplace = True)
data.drop(data[data['Age']>99].index, inplace = True)

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]
```

In [14]:
```
#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');
```

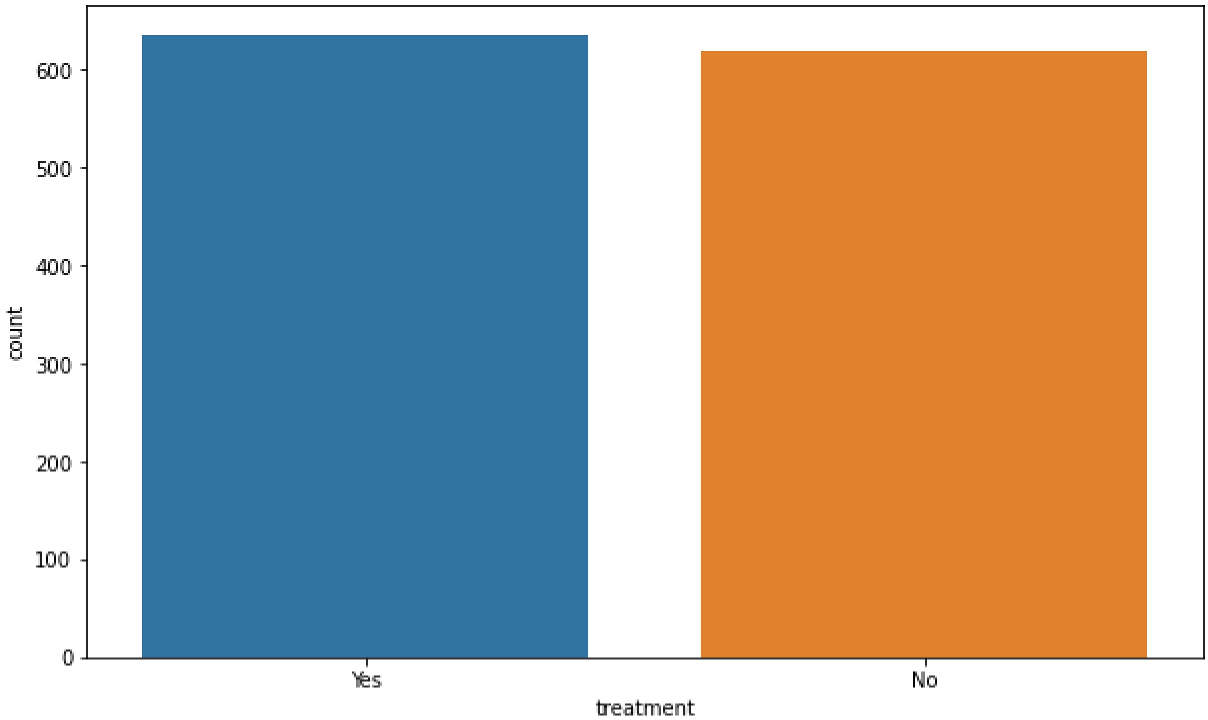
```
<Figure size 720x432 with 0 Axes>
```



In [15]:
```
#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');
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-15-8e3cfe8f45f6> in <module>
      2 plt.figure(figsize = (10,6));
      3 treat = sns.countplot(data = data,  x = 'treatment');
----> 4 treat.bar_label(treat.containers[0]);
      5 plt.title('Total number of individuals who received treatment or not');

AttributeError: 'AxesSubplot' object has no attribute 'bar_label'
```



In [16]:
```
#Check Dtypes
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1254 entries, 0 to 1258
Data columns (total 24 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Age                        1254 non-null   int64 
 1   Gender                     1254 non-null   object
 2   Country                    1254 non-null   object
 3   self_employed              1254 non-null   object
 4   family_history             1254 non-null   object
 5   treatment                  1254 non-null   object
 6   work_interfere             1254 non-null   object
 7   no_employees               1254 non-null   object
 8   remote_work                1254 non-null   object
 9   tech_company               1254 non-null   object
 10  benefits                   1254 non-null   object
 11  care_options               1254 non-null   object
 12  wellness_program           1254 non-null   object
 13  seek_help                  1254 non-null   object
 14  anonymity                  1254 non-null   object
 15  leave                      1254 non-null   object
 16  mental_health_consequence  1254 non-null   object
 17  phys_health_consequence    1254 non-null   object
 18  coworkers                  1254 non-null   object
 19  supervisor                 1254 non-null   object
 20  mental_health_interview    1254 non-null   object
 21  phys_health_interview      1254 non-null   object
 22  mental_vs_physical         1254 non-null   object
 23  obs_consequence            1254 non-null   object
dtypes: int64(1), object(23)
memory usage: 284.9+ KB
```

In [17]:

```
#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_
                     'remote_work', 'tech_company','benefits','care_options', 'wellness
                     'seek_help', 'anonymity', 'leave', 'mental_health_consequence', 'p
                     'coworkers', 'supervisor', 'mental_health_interview','phys_health_
                     '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'>
Int64Index: 1254 entries, 0 to 1258
Data columns (total 24 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Age                        1254 non-null   int64
 1   Gender                     1254 non-null   int64
 2   Country                    1254 non-null   int64
 3   self_employed              1254 non-null   int64
 4   family_history             1254 non-null   int64
 5   treatment                  1254 non-null   int64
 6   work_interfere             1254 non-null   int64
 7   no_employees               1254 non-null   int64
 8   remote_work                1254 non-null   int64
 9   tech_company               1254 non-null   int64
 10  benefits                   1254 non-null   int64
 11  care_options               1254 non-null   int64
 12  wellness_program           1254 non-null   int64
 13  seek_help                  1254 non-null   int64
 14  anonymity                  1254 non-null   int64
 15  leave                      1254 non-null   int64
 16  mental_health_consequence  1254 non-null   int64
 17  phys_health_consequence    1254 non-null   int64
```

```
 18  coworkers                  1254 non-null   int64
 19  supervisor                 1254 non-null   int64
 20  mental_health_interview    1254 non-null   int64
 21  phys_health_interview      1254 non-null   int64
 22  mental_vs_physical         1254 non-null   int64
 23  obs_consequence            1254 non-null   int64
dtypes: int64(24)
memory usage: 284.9 KB
```

In [18]:
```
#Let's check Standard deviation
data.describe()
```

Out[18]:

	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	no_emp
count	1254.000000	1254.000000	1254.000000	1254.000000	1254.000000	1254.000000	1254.000000	1254.
mean	32.020734	1.012759	37.816587	0.115630	0.390750	0.506380	2.329346	2.
std	7.372511	0.191178	13.320466	0.319908	0.488113	0.500159	1.549652	1.
min	5.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
25%	27.000000	1.000000	42.000000	0.000000	0.000000	0.000000	1.000000	1.
50%	31.000000	1.000000	45.000000	0.000000	0.000000	1.000000	3.000000	3.
75%	36.000000	1.000000	45.000000	0.000000	1.000000	1.000000	4.000000	4.
max	72.000000	4.000000	47.000000	1.000000	1.000000	1.000000	4.000000	5.

8 rows × 24 columns

In [ ]:
```
Split data to train and test
```

In [19]:
```
from sklearn.model_selection import train_test_split

#I wanna work on 'treatment' column.
X = data.drop(columns = ['treatment'])
y = data['treatment']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

print(X_train.shape, y_train.shape)
print('-'*30)
print(X_test.shape, y_test.shape)
print('_'*30)
```

```
(940, 23) (940,)
------------------------------
(314, 23) (314,)
______________________________
```

In [20]:
```
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier as RFC
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.tree import DecisionTreeClassifier as DT
```

In [ ]:
```
Random forest Classifier
```

In [ ]:
```
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier as RFC
```

```
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.tree import DecisionTreeClassifier as DT
```

In [24]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier as RFC
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score

# Define the steps for the pipeline
steps_rfc = [('Scaler', StandardScaler()), ('clf', RFC(n_estimators=40))]

# Create the pipeline
clf_rfc = Pipeline(steps=steps_rfc)

# Fit the pipeline to your training data
clf_rfc.fit(X_train, y_train)

# Make predictions on the test data
y_pred_rfc = clf_rfc.predict(X_test)

# Calculate and print the accuracy
print('RFC accuracy: ', accuracy_score(y_true=y_test, y_pred=y_pred_rfc) * 100)
```

```
RFC accuracy:  83.43949044585987
```

In [ ]:

```
K Nearest Neighbor
```

In [25]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier as RFC
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
steps_knn = [('Scaler', StandardScaler()),
             ('clf', KNN(n_neighbors = 5))]



clf_knn = Pipeline(steps=steps_knn)


clf_knn.fit(X_train, y_train)

y_pred_knn = clf_knn.predict(X_test)
print('KNN accuracy :', accuracy_score(y_true=y_test, y_pred=y_pred_knn)*100)
```

```
KNN accuracy : 74.84076433121018
```

In [ ]:

```
Support vector Classifier
```

In [26]:

```
steps_svc = [('Scaler', StandardScaler()),
             ('clf', SVC())]



clf_svc = Pipeline(steps=steps_svc)


clf_svc.fit(X_train, y_train)

y_pred_svc = clf_svc.predict(X_test)
print('SVC accuracy :', accuracy_score(y_true=y_test, y_pred=y_pred_svc)*100)
```

```
SVC accuracy : 79.61783439490446
```

In [ ]:

```
Decision Tree
```

In [27]:

```
steps_dt = [('Scaler', StandardScaler()),
             ('clf', DT())]



clf_dt = Pipeline(steps=steps_dt)


clf_dt.fit(X_train, y_train)

y_pred_dt = clf_dt.predict(X_test)
print('DT accuracy :', accuracy_score(y_true=y_test, y_pred=y_pred_dt)*100)
```

```
DT accuracy : 75.15923566878982
```

In [ ]:

```
Tuning
```

In [28]:

```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
```

In [29]:

```
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}
```

In [31]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [32]:

```
rf_classifier = RandomForestClassifier()
```

In [33]:

```
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}
```

In [34]:

```
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, n_jobs=-1, ver
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 324 candidates, totalling 1620 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 32 concurrent workers.
[Parallel(n_jobs=-1)]: Done  98 tasks      | elapsed:    5.2s
[Parallel(n_jobs=-1)]: Done 301 tasks      | elapsed:   10.7s
[Parallel(n_jobs=-1)]: Done 584 tasks      | elapsed:   18.3s
[Parallel(n_jobs=-1)]: Done 949 tasks      | elapsed:   28.2s
[Parallel(n_jobs=-1)]: Done 1394 tasks      | elapsed:   40.2s
[Parallel(n_jobs=-1)]: Done 1620 out of 1620 | elapsed:   46.3s finished
```

Out[34]:

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'max_depth': [None, 10, 20, 30],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [100, 200, 300]},
             verbose=2)
```

In [37]:

```
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)
```

```
Best Hyperparameters: {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_s
amples_split': 10, 'n_estimators': 100}
```

In [36]:
```python
best_rf_classifier = RandomForestClassifier(**best_params)
best_rf_classifier.fit(X_train, y_train)
```

Out[36]:
```
RandomForestClassifier(max_features='log2', min_samples_leaf=2,
                       min_samples_split=10)
```

In [39]:
```python
y_pred = best_rf_classifier.predict(X_test)
```

In [40]:
```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Accuracy: 0.8446215139442231
Precision: 0.8092105263157895
Recall: 0.924812030075188
F1 Score: 0.863157894736842
```

In [ ]:

CONCLUSION:

In conclusion, this approach aims to provide a comprehensive understanding of public health awareness campaign effectiveness. By combining analysis objectives, data collection, visualization planning, and code integration, the project seeks to deliver actionable insights for guiding future campaigns. The combination of quantitative metrics and qualitative assessments will contribute to a holistic evaluation of campaign success and areas for improvement.

TEAM MEMBERS:

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