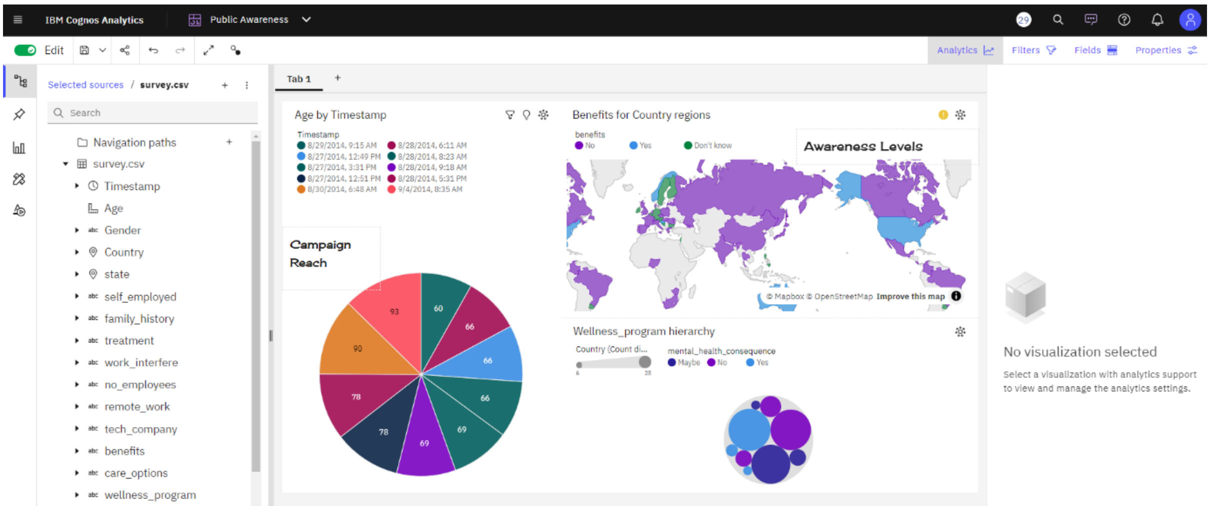
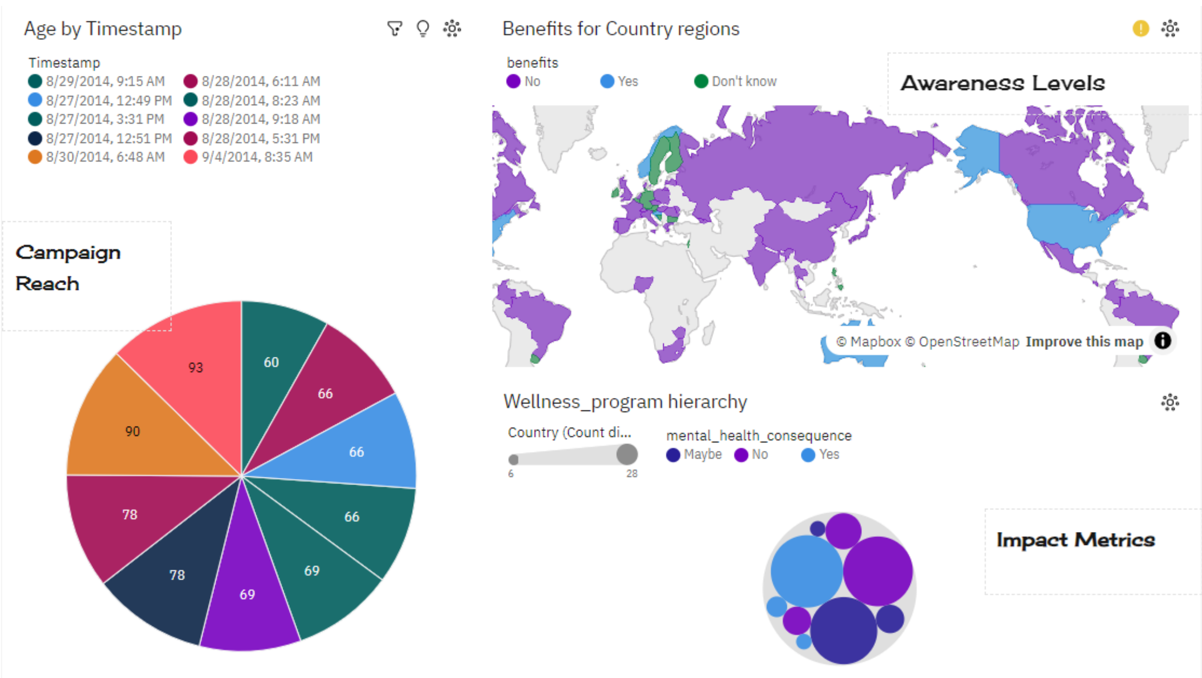
PHASE 4

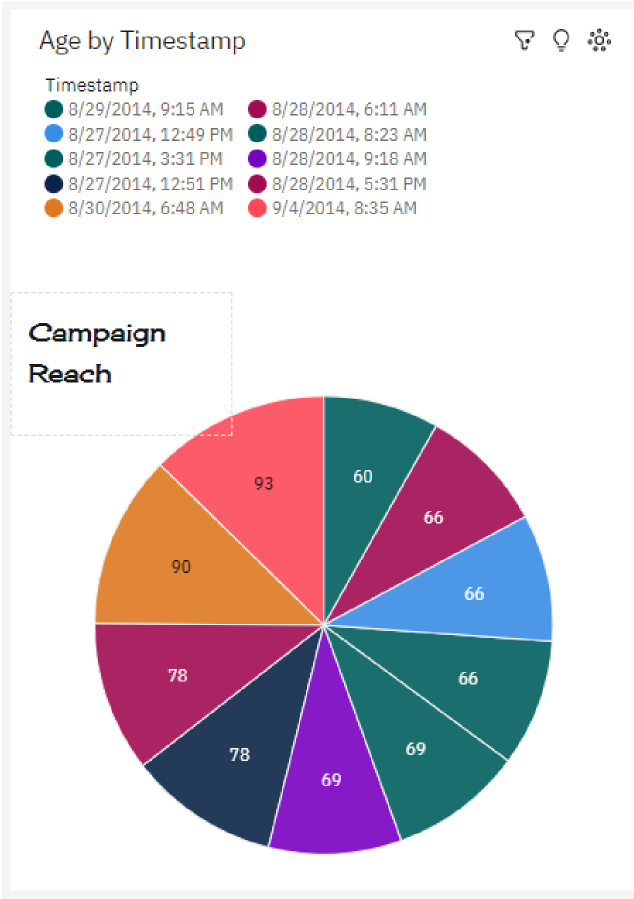
VISUALIZATION USING IBM COGNOS:



DASHBOARD:

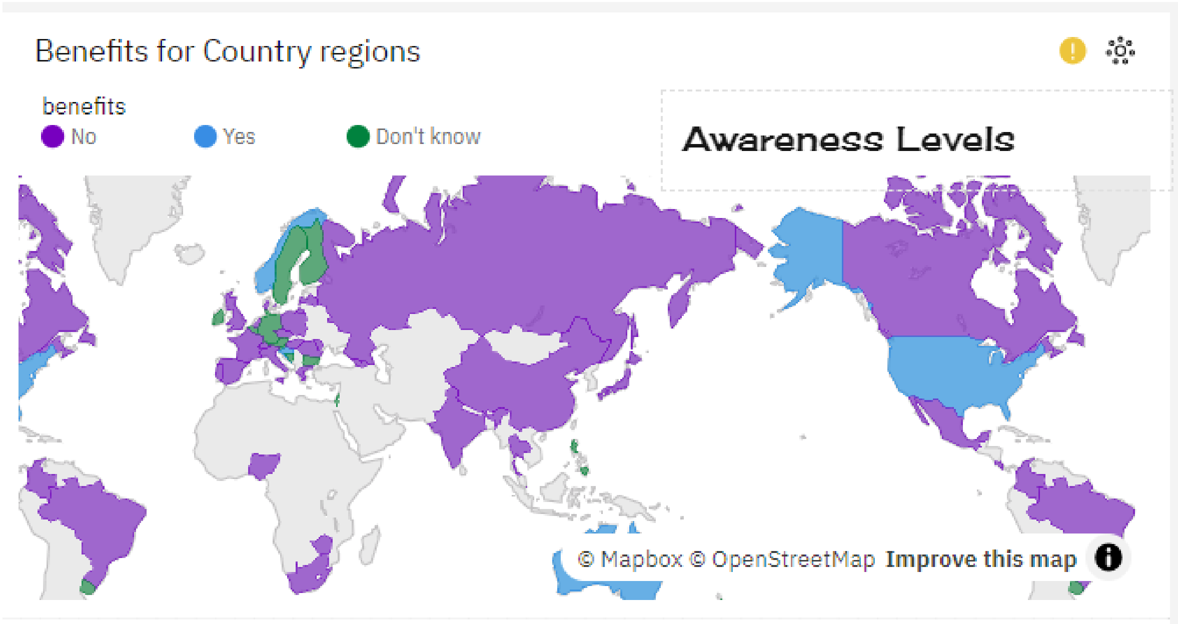


CAMPAIGN REACH:



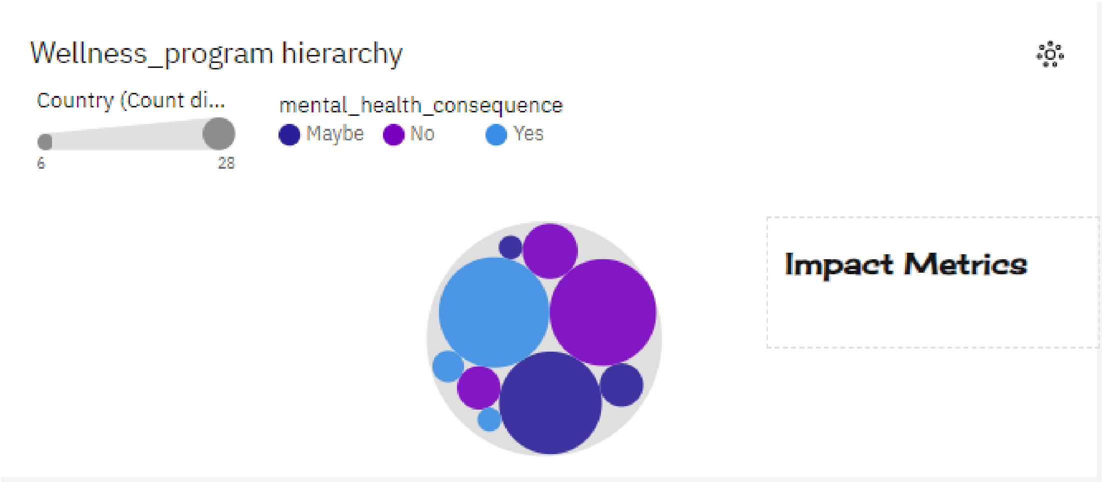
- State NA has the highest total Age due to Timestamp 2014-09-04T08:35:49.
- Over all values of Timestamp, the sum of Age is 735.
- Age ranges from 60, when Timestamp is 2014-08-29T09:15:52, to 93, when Timestamp is 2014-09-04T08:35:49.
- For Age, the most significant values of Timestamp are 2014-09-04T08:35:49 and 2014-08-30T06:48:28, whose respective Age values add up to 183, or 24.9 % of the total.
- Objectives of Campaign Reach Visualization are to:
  - ✓ Track the campaign's progress over time.
  - ✓ Identify peak engagement periods.
  - ✓ Evaluate the effectiveness of outreach efforts.
  - ✓ Compare campaign performance across different segments.
  - ✓ Assess the impact of external factors on the campaign's reach.

AWARENESS LEVELS:



- No benefits accounted for 100% of Zimbabwe Age compared to 15% for United States.
- Benefits No has the highest Age at 300,000,030,042, out of which Country Zimbabwe contributed the most at 299,999,999,997.
- Country Zimbabwe has the highest total Age due to benefits No.
- The total number of results for benefits, across all countries, is nearly four thousand.
- Objectives of Awareness Level Visualization are to:
    - ✓ Understand awareness levels across different regions.
    - ✓ Allocate resources effectively based on awareness levels.
    - ✓ Tailor campaigns to specific demographics.
    - ✓ Compare awareness levels between regions.
    - ✓ Evaluate the impact of awareness campaigns.

IMPACT METRICS:



- Wellness_program No has the highest Country due to mental_health_consequence Yes.
- Wellness_program No has the highest values of both Country and Age.
- Mental_health_consequence No has the highest Country at 52, out of which wellness_program No contributed the most at 27.
- Mental_health_consequence No has the highest Count distinct Country but is ranked #3 in Total Age.
- Mental_health_consequence Yes has the highest Total Age but is ranked #1 in Count distinct Country.
- No is the most frequently occurring category of wellness_program with a count of 2526 items with Country values (66.9 % of the total).
- The overall number of results for Country is nearly four thousand.
- Objectives of Impact Metrics Visualization are to:
  - ✓ Assess the campaign's effectiveness.
  - ✓ Identify areas for improvement.
  - ✓ Track changes over time.
  - ✓ Compare responses across different groups.
  - ✓ Measure overall organizational well-being.

DATA ANALYSIS:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.api as sm
import seaborn as sns
data = pd.read_csv('/content/survey.csv')
data.head()
```

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

5 rows × 27 columns

```
# Calculate the engagement rate
total_responses = len(data)
received_treatment = len(data[data['treatment'] == 'Yes'])
engagement_rate = (received_treatment / total_responses) * 100
print(f"Engagement Rate: {engagement_rate:.2f}%")
```

```
Engagement Rate: 50.60%
```

```
data['Age'] = data['Age'].astype(int)
data['Gender'] = data['Country'].astype(str)
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
data['Gender'] = label_encoder.fit_transform(data['Gender'])
data['work_interfere'] = label_encoder.fit_transform(data['work_interfere'])
```

#**demographic analysis**

import pandas as pd

demographic_attributes = ['Age', 'Gender', 'Country']

demographic_stats = data.groupby(demographic_attributes).agg({

'Gender': ['mean', 'median'],

'benefits': 'count'

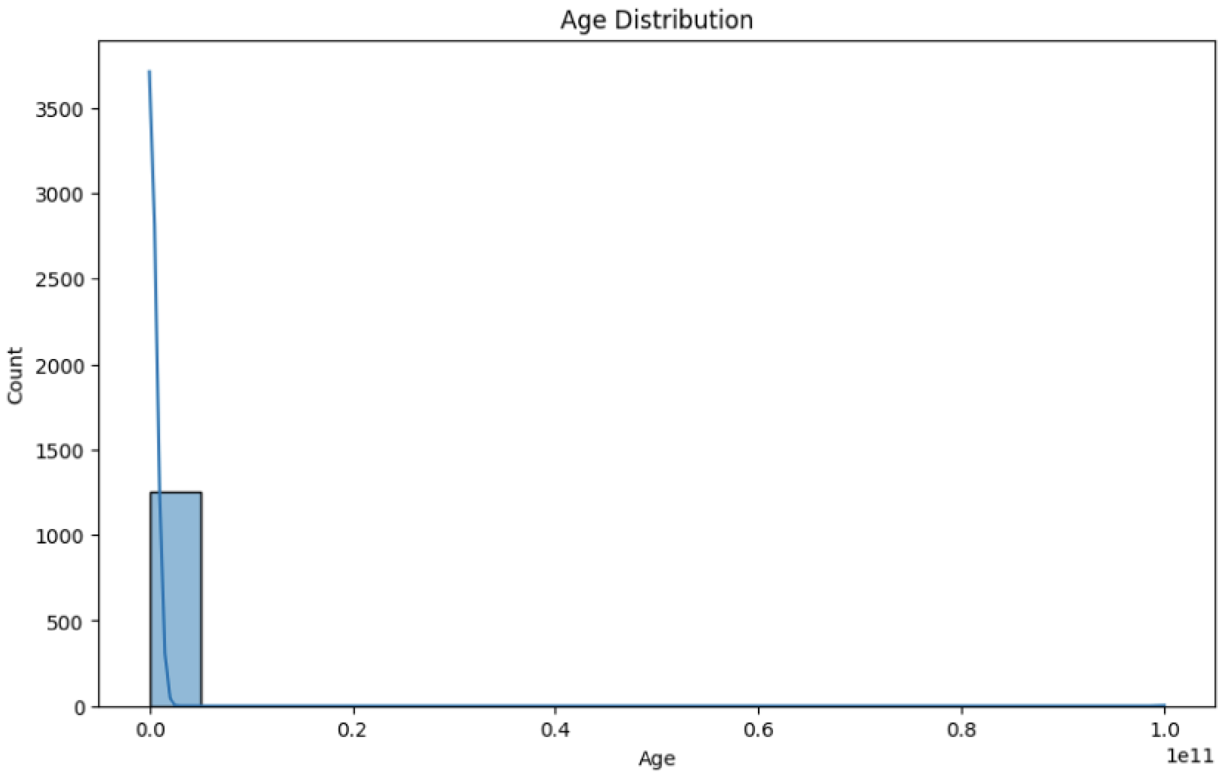
}).reset_index()

demographic_stats.columns = ['_'.join(col).strip() for col in demographic_stats.columns.values]

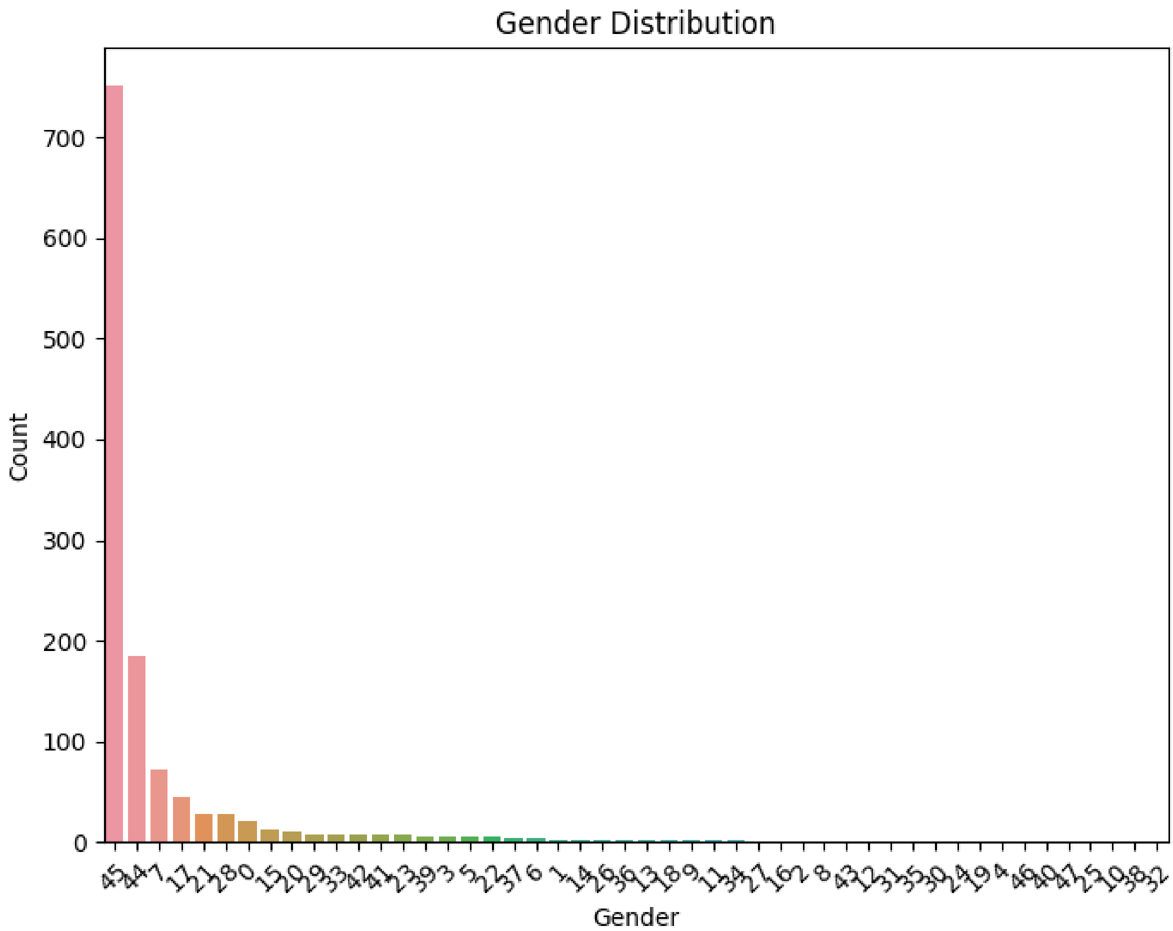
print(demographic_stats)

```
            Age_  Gender_         Country_  Gender_mean  Gender_median  \
0          -1726       44   United Kingdom         44.0           44.0   
1            -29       45    United States         45.0           45.0   
2             -1       45    United States         45.0           45.0   
3              5       45    United States         45.0           45.0   
4              8        2     Bahamas, The          2.0            2.0   
..           ...      ...              ...          ...            ...   
268           62       45    United States         45.0           45.0   
269           65       45    United States         45.0           45.0   
270           72       45    United States         45.0           45.0   
271          329       45    United States         45.0           45.0   
272  99999999999       47         Zimbabwe         47.0           47.0   

     benefits_count  
0                 1  
1                 1  
2                 1  
3                 1  
4                 1  
..              ...  
268               1  
269               1  
270               1  
271               1  
272               1  

[273 rows x 6 columns]
```

```
plt.figure(figsize=(10, 6))
sns.histplot(data['Age'], kde=True, bins=20)
plt.title("Age Distribution")
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()
```

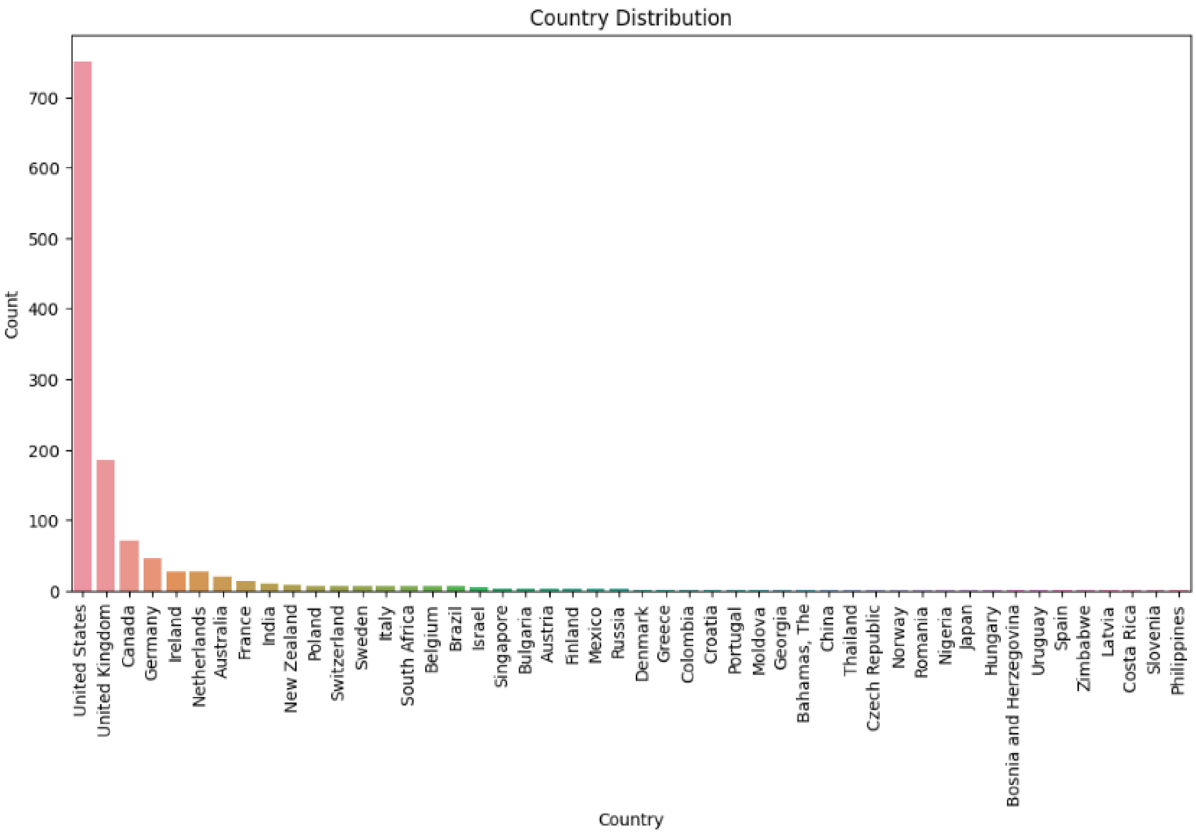


```
# Gender Distribution
plt.figure(figsize=(8, 6))
sns.countplot(x='Gender', data=data, order=data['Gender'].value_counts().index)
plt.title("Gender Distribution")
plt.xlabel("Gender")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



Country Distribution

```
plt.figure(figsize=(12, 6))
sns.countplot(x='Country', data=data, order=data['Country'].value_counts().index)
plt.title("Country Distribution")
plt.xlabel("Country")
plt.ylabel("Count")
plt.xticks(rotation=90)
plt.show()
```



STATISTICAL TEST

```
import scipy.stats as stats

treatment_group = data[data['treatment'] == 'Yes']['Age']

no_treatment_group = data[data['treatment'] == 'No']['Age']

t_stat, p_value = stats.ttest_ind(treatment_group, no_treatment_group, equal_var=False)

print(f'T-Stat: {t_stat}')

print(f'P-Value: {p_value}')

if p_value < 0.05:

    print('There is a significant age difference between the treatment and no treatment groups.')

else:

    print('There is no significant age difference between the groups.')
```

```
T-Stat: 0.9999999927079095
P-Value: 0.31769081891318196
There is no significant age difference between the groups.
```

#chi-square test

```
from scipy.stats import chi2_contingency

contingency_table = pd.crosstab(data['Gender'], data['treatment'])

chi2, p, _, _ = chi2_contingency(contingency_table)

print(f'Chi-squared statistic: {chi2}')

print(f'P-Value: {p}')

if p < 0.05:

    print('Gender and treatment are not independent; there is an association.')

else:

    print('Gender and treatment are independent; there is no significant association.')
```

```
Chi-squared statistic: 69.79950917591624
P-Value: 0.01705720173000851
Gender and treatment are not independent; there is an association.
```

#correlation analysis

import scipy.stats as stats

data['treatment_binary'] = (data['treatment'] == 'Yes').astype(int)

correlation, p_value = stats.pointbiserialr(data['Age'], data['treatment_binary'])

print(f"Point-biserial Correlation: {correlation}")

print(f"P-Value: {p_value}")

```
Point-biserial Correlation: 0.027860259191070148
P-Value: 0.3232709949993724
```

#ANOVA(Analysis of Variance)

import statsmodels.api as sm

from statsmodels.formula.api import ols

model = ols('Age ~ Country', data=data).fit()

anova_table = sm.stats.anova_lm(model, typ=2)

print(anova_table)

```
                sum_sq      df             F  PR(>F)
Country   9.992057e+21    47.0  7.981948e+16     0.0
Residual  3.225465e+06  1211.0           NaN     NaN
```

#Regression analysis

import statsmodels.api as sm

X = data[['Gender', 'Country']]

X = pd.get_dummies(X, columns=['Gender', 'Country'], drop_first=True)

y = data['Age']

X = sm.add_constant(X)

model = sm.OLS(y, X).fit()

print(model.summary())

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    Age   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 7.982e+16
Date:                Wed, 01 Nov 2023   Prob (F-statistic):               0.00
Time:                        10:01:47   Log-Likelihood:                -6727.1
No. Observations:                1259   AIC:                         1.355e+04
Df Residuals:                    1211   BIC:                         1.380e+04
Df Model:                          47
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const             29.0000     11.262      2.575      0.010       6.905      51.095
Gender_1          -1.1667     15.927     -0.073      0.942     -32.414      30.081
Gender_2         -10.5000     26.412     -0.398      0.691     -62.318      41.318
Gender_3           0.2500     11.945      0.021      0.983     -23.185      23.685
Gender_4          -2.0000     26.412     -0.076      0.940     -53.818      49.818
Gender_5          -0.8333     11.945     -0.070      0.944     -24.269      22.602
Gender_6          -0.3750     14.077     -0.027      0.979     -27.994      27.244
Gender_7           0.1597      6.400      0.025      0.980     -12.396      12.715
Gender_8           5.5000     26.412      0.208      0.835     -46.318      57.318
Gender_9          -1.0000     19.096     -0.052      0.958     -38.464      36.464
Gender_10          4.5000     26.412      0.170      0.865     -47.318      56.318
Gender_11          4.5000     19.096      0.236      0.814     -32.964      41.964
Gender_12          2.0000     26.412      0.076      0.940     -49.818      53.818
Gender_13          3.0000     19.096      0.157      0.875     -34.464      40.464
Gender_14          0.1667     15.927      0.010      0.992     -31.081      31.414
Gender_15          1.2692      9.107      0.139      0.889     -16.597      19.136
Gender_16         -4.5000     26.412     -0.170      0.865     -56.318      47.318
Gender_17          0.7111      6.819      0.104      0.917     -12.668      14.090
Gender_18          3.7500     19.096      0.196      0.844     -33.714      41.214
Gender_19         -1.0000     26.412     -0.038      0.970     -52.818      50.818
Gender_20         -2.2500      9.914     -0.227      0.821     -21.701      17.201
Gender_21          1.4629      7.508      0.195      0.846     -13.267      16.193
Gender_22         -2.0000     12.841     -0.156      0.876     -27.192      23.192
Gender_23          2.2143     11.262      0.197      0.844     -19.881      24.309
Gender_24         10.0000     26.412      0.379      0.705     -41.818      61.818
```

#Hyp testing (Z-TEST)

from statsmodels.stats.proportion import proportions_ztest

count_treatment = data[data['treatment'] == 'Yes'].shape[0]

count_no_treatment = data[data['treatment'] == 'No'].shape[0]

stat, p_value = proportions_ztest([count_treatment, count_no_treatment], [data.shape[0], data.shape[0]])

print(f'Z-statistic: {stat}')

print(f'P-value: {p_value}')

ADVANCED DATA ANALYSIS

from sklearn.cluster import KMeans

data_for_clustering = data[['Age', 'treatment_binary']] # Assuming 'treatment_binary' is converted as described earlier

kmeans = KMeans(n_clusters=3, random_state=0)

data['Cluster'] = kmeans.fit_predict(data_for_clustering)

 # Visualize the clusters

plt.figure(figsize=(8, 6))

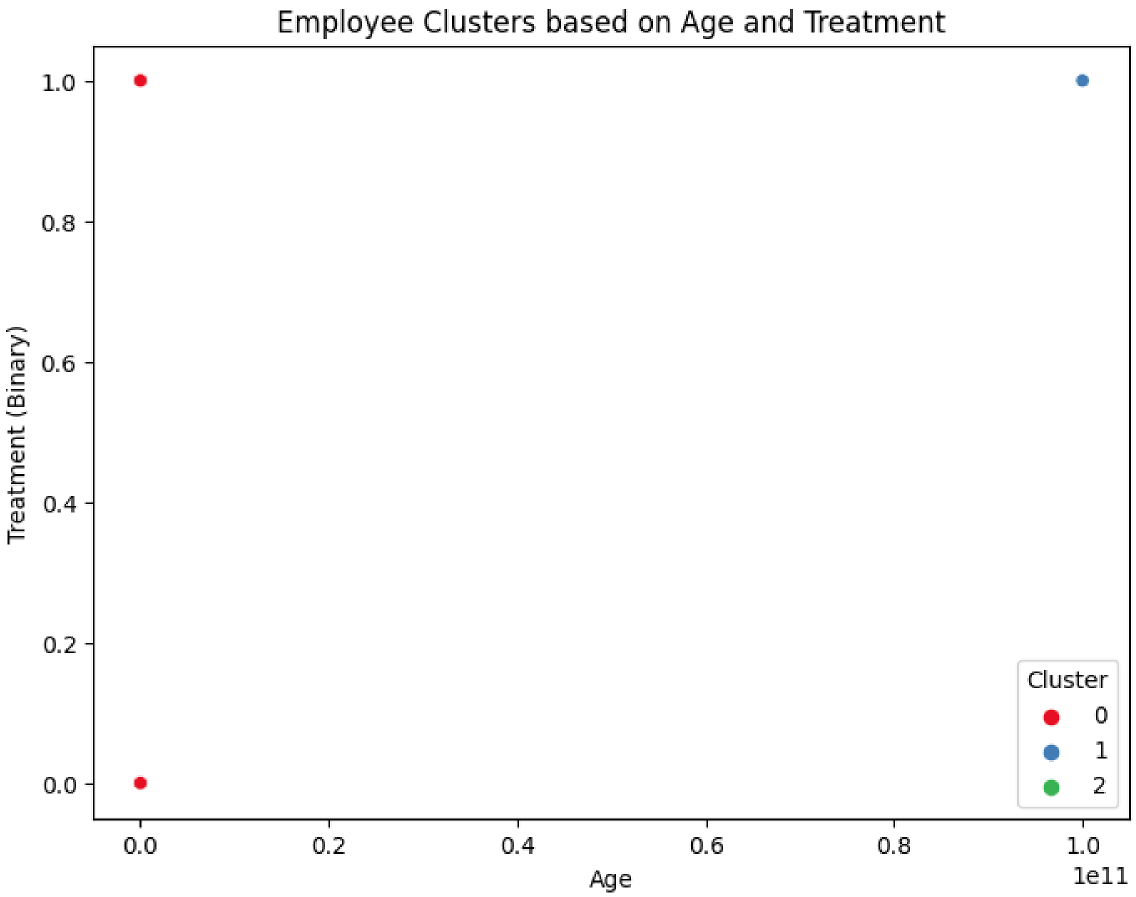
sns.scatterplot(data=data, x='Age', y='treatment_binary', hue='Cluster', palette='Set1')

plt.title('Employee Clusters based on Age and Treatment')

plt.xlabel('Age')

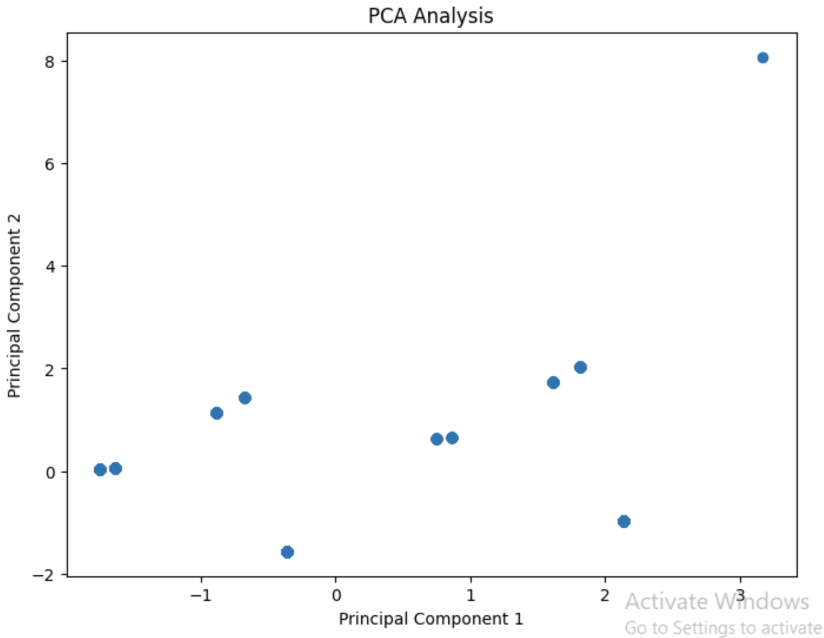
plt.ylabel('Treatment (Binary)')

plt.show()



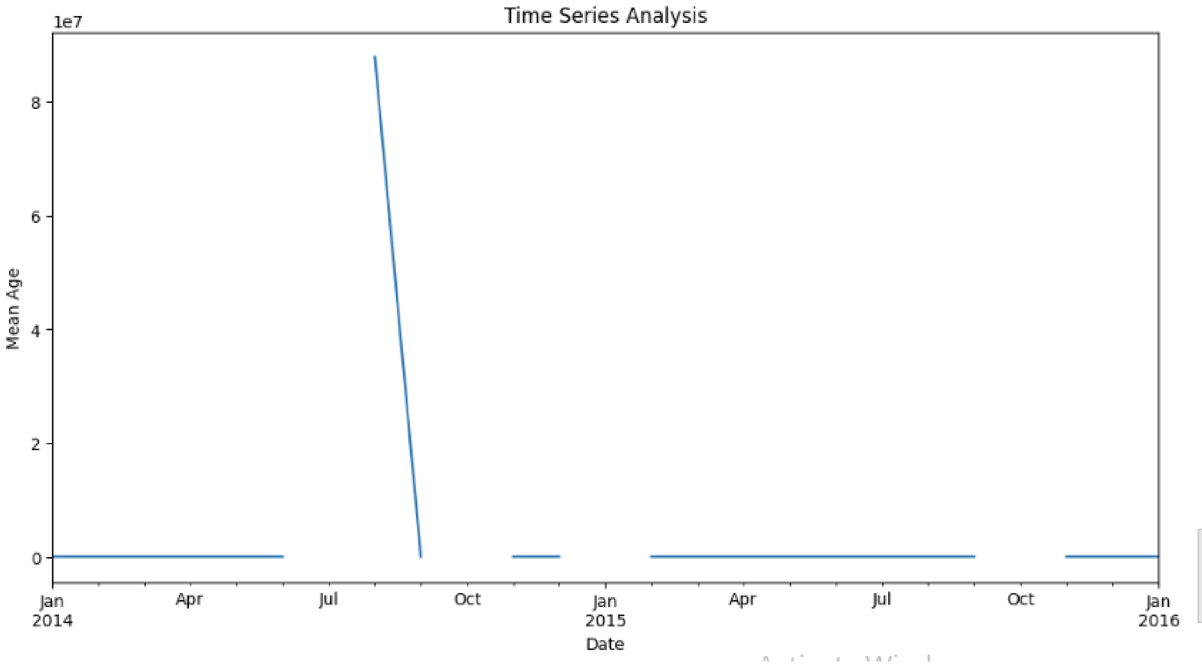
#Principal Component analysis

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
columns_to_impute = ['family_history', 'work_interfere']
imputer = SimpleImputer(strategy='most_frequent')
data[columns_to_impute] = imputer.fit_transform(data[columns_to_impute])
features = data[['Age', 'family_history', 'work_interfere']]
features_encoded = pd.get_dummies(features, columns=['family_history', 'work_interfere'])
features_std = (features_encoded - features_encoded.mean()) / features_encoded.std()
pca = PCA(n_components=2)
principal_components = pca.fit_transform(features_std)
plt.figure(figsize=(8, 6))
plt.scatter(principal_components[:, 0], principal_components[:, 1])
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Analysis')
plt.show()
```



#Time Series

```
data['Timestamp'] = pd.to_datetime(data['Timestamp'])
data.set_index('Timestamp', inplace=True)
monthly_mean_age = data['Age'].resample('M').mean()
plt.figure(figsize=(12, 6))
monthly_mean_age.plot()
plt.xlabel('Date')
plt.ylabel('Mean Age')
plt.title('Time Series Analysis')
plt.show()
```



#Logistic Regression for Binary Classification

```
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, confusion_matri

from sklearn.model_selection import train_test_split

X = pd.get_dummies(data[['Age', 'family_history', 'work_interfere']],
columns=['family_history', 'work_interfere'])

y = data['treatment_binary']

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

model = LogisticRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)

print(f'Accuracy: {accuracy}')

print('Confusion Matrix:')

print(conf_matrix)
```

```
Accuracy: 0.8015873015873016
Confusion Matrix:
[[ 88  41]
 [  9 114]]
```