Phase 3: Development Part 1 Project Report

Public Health Awareness Campaign Analysis

Done By: Keerthi Varshini S
Date: 18/10/23

Table of Contents

1. Introduction

In this phase of the project, we embark on the development part to create a public health awareness campaign analysis. The foundation of this analysis is built upon the data collected from the source provided. The main focus is on using IBM Cognos for visualization to derive insights and patterns that will help us design an effective public health awareness campaign.

2. Objectives

The primary objectives of this project are as follows:

- Analyze mental health data within the tech industry.
- Identify key trends and patterns in mental health-related factors.
- Generate visualizations that convey these insights effectively.
- Formulate recommendations for the public health awareness campaign based on the analysis.

3. Data Collection

For this project, we collected data from the following source:

Dataset Link: https://www.kaggle.com/datasets/osmi/mental-health-in-tech-survey

The dataset we used is the "Mental Health in Tech Survey" dataset, which was obtained from Kaggle. It contains valuable information related to mental health within the technology sector.

4. Data Preprocessing

To ensure the quality and accuracy of the collected data, we performed extensive data cleaning and preprocessing in a Jupyter Notebook. The following are the key preprocessing steps:

Data Cleaning:

Data cleaning is an essential phase of data preparation aimed at ensuring the dataset's integrity and reliability. This process encompasses several key steps:

• **Handling Missing Values:** One of the primary data cleaning tasks involved addressing missing values within the dataset. We recognized that missing data can lead to skewed

analyses, and thus, we took meticulous care in handling them. For numerical attributes with missing values, we applied techniques like mean, median, or mode imputation, ensuring that the imputed values were coherent with the underlying data distribution. For categorical attributes, custom imputation methods were employed when suitable. These steps allowed us to mitigate the impact of missing data on our analysis.

• **Handling Duplicate Values: Ensuring Data Integrity**

Duplicate values in a dataset can introduce bias and inaccuracies. We systematically detected and treated duplicates by comparing records across attributes. Our approach involved considering whether to remove or retain duplicates based on their significance. A data verification step confirmed the effectiveness of our process. By addressing duplicate values, we improved data quality and eliminated potential sources of bias for more accurate analyses and machine learning models.

• **Data Format Consistency:** To maintain data quality, we addressed inconsistencies in data formatting. This included harmonizing date formats, standardizing the representation of categorical variables, and ensuring uniformity across the dataset.

The final dataset used for analysis is clean, consistent, and ready for visualization.

5. Visualization Using IBM Cognos

For the visualization part of the project, we utilized IBM Cognos. This platform allowed us to create various charts, graphs, and dashboards to convey the insights derived from the dataset. The visualizations include but are not limited to:

- Bar charts
- Pie charts
- Line charts
- Scatter plots
- Heatmaps

These visualizations help us understand trends and patterns related to mental health within the tech industry, such as the prevalence of mental health issues, their correlation with job factors, and geographical distribution.

6. Conclusion

In this phase of the project, we successfully initiated the development of the public health awareness campaign analysis. We defined our objectives, collected and preprocessed the dataset, and created compelling visualizations using IBM Cognos. The insights gathered from this analysis will serve as the foundation for our campaign's strategy.

In the upcoming phases, we will continue to build on this analysis, refine our findings, and formulate concrete recommendations for the public health awareness campaign.

IBM CONGO DASHBOARD LINK:

https://us3.ca.analytics.ibm.com/bi/?perspective=dashboard&pathRef=.my_folders%2FNew%2Bdashboard&action=view&mode=dashboard&subView=model0000018b393cc1c4_00000001

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("survey.csv")
data.head()
```

Out[3]:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_int
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	I
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	I
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

In [4]:
```
data.describe()
```

Out[4]:

	Age
count	1.259000e+03
mean	7.942815e+07
std	2.818299e+09
min	-1.726000e+03
25%	2.700000e+01
50%	3.100000e+01
75%	3.600000e+01
max	1.000000e+11

In [11]:
```
import pandas as pd

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

new_df = df.dropna()
```

In [12]:
```
import pandas as pd

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

df.dropna(inplace = True)
```

In [13]:
```
import pandas as pd

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

df.drop_duplicates(inplace = True)
```

In [14]:
```
df.to_csv('cleaned_survey.csv', index=False)
```

In [15]:
```
import pandas as pd

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

mean_age = df['Age'].mean()
df['Age'].fillna(mean_age, inplace=True)
```

In [17]:
```
import pandas as pd

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

df['comments'] = df['comments'].str.lower()
```

In [22]:
```
import pandas as pd
from scipy import stats

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Check the first few rows of the dataset
print(df.head())

# Check basic summary statistics
print(df.describe())

# Check for missing values
print(df.isnull().sum())

# Handle missing values by filling with mean or median (for a specific col
df['Age'].fillna(df['Age'].median(), inplace=True)

# Detect and handle outliers (for a specific numeric column like 'Age')
z_scores = stats.zscore(df['Age'])
df = df[(z_scores < 3)]

# Data type conversion (e.g., convert 'Timestamp' column to datetime)
df['Timestamp'] = pd.to_datetime(df['Timestamp'])

# Verify the changes
print(df.head())
```

```
             Timestamp  Age  Gender         Country state self_employed  \
0  2014-08-27 11:29:31   37  Female   United States    IL           NaN
1  2014-08-27 11:29:37   44       M   United States    IN           NaN
2  2014-08-27 11:29:44   32    Male          Canada   NaN           NaN
3  2014-08-27 11:29:46   31    Male  United Kingdom   NaN           NaN
4  2014-08-27 11:30:22   31    Male   United States    TX           NaN

  family_history treatment work_interfere    no_employees  ...  \
0             No       Yes          Often            6-25  ...
1             No        No         Rarely  More than 1000  ...
2             No        No         Rarely           6-25  ...
3            Yes       Yes          Often         26-100  ...
```

```
4             No        No          Never         100-500  ...

                leave mental_health_consequence phys_health_consequence  \
0       Somewhat easy                         No                       No
1          Don't know                      Maybe                       No
2  Somewhat difficult                         No                       No
3  Somewhat difficult                        Yes                      Yes
4          Don't know                         No                       No

      coworkers supervisor mental_health_interview phys_health_interview  \
0  Some of them        Yes                       No                 Maybe
1            No         No                       No                    No
2           Yes        Yes                      Yes                   Yes
3  Some of them         No                    Maybe                 Maybe
4  Some of them        Yes                      Yes                   Yes

  mental_vs_physical obs_consequence comments
0                Yes              No      NaN
1         Don't know              No      NaN
2                 No              No      NaN
3                 No             Yes      NaN
4         Don't know              No      NaN

[5 rows x 27 columns]
                Age
count  1.259000e+03
mean   7.942815e+07
std    2.818299e+09
min   -1.726000e+03
25%    2.700000e+01
50%    3.100000e+01
75%    3.600000e+01
max    1.000000e+11
Timestamp                      0
Age                            0
Gender                         0
Country                        0
state                        515
self_employed                 18
family_history                 0
treatment                      0
work_interfere               264
no_employees                   0
remote_work                    0
tech_company                   0
benefits                       0
care_options                   0
wellness_program               0
seek_help                      0
anonymity                      0
leave                          0
mental_health_consequence      0
phys_health_consequence        0
coworkers                      0
supervisor                     0
mental_health_interview        0
phys_health_interview          0
mental_vs_physical             0
obs_consequence                0
comments                    1095
dtype: int64
            Timestamp  Age  Gender         Country state self_employed  \
0 2014-08-27 11:29:31   37  Female   United States    IL           NaN
1 2014-08-27 11:29:37   44       M   United States    IN           NaN
```

```
2 2014-08-27 11:29:44   32    Male          Canada   NaN           NaN
3 2014-08-27 11:29:46   31    Male  United Kingdom   NaN           NaN
4 2014-08-27 11:30:22   31    Male   United States    TX           NaN

  family_history treatment work_interfere    no_employees  ...  \
0             No       Yes          Often            6-25  ...
1             No        No         Rarely  More than 1000  ...
2             No        No         Rarely            6-25  ...
3            Yes       Yes          Often          26-100  ...
4             No        No          Never         100-500  ...

               leave mental_health_consequence phys_health_consequence  \
0       Somewhat easy                       No                       No
1          Don't know                    Maybe                       No
2  Somewhat difficult                       No                       No
3  Somewhat difficult                      Yes                      Yes
4          Don't know                       No                       No

      coworkers supervisor mental_health_interview phys_health_interview  \
0  Some of them        Yes                      No                  Maybe
1            No         No                      No                     No
2           Yes        Yes                     Yes                    Yes
3  Some of them         No                   Maybe                  Maybe
4  Some of them        Yes                     Yes                    Yes

  mental_vs_physical obs_consequence comments
0                Yes              No      NaN
1         Don't know              No      NaN
2                 No              No      NaN
3                 No             Yes      NaN
4         Don't know              No      NaN
```

In [ ]:

In [41]:
```
import pandas as pd

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Standardize the Gender column
df['Gender'] = df['Gender'].str.lower()
df['Gender'] = df['Gender'].apply(lambda x: 'Male' if x in ['m', 'male','M

# Save the updated DataFrame back to the CSV file
df.to_csv('cleaned_survey.csv', index=False)
```

In [42]:
```
import pandas as pd

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in the Gender column
unique_genders = df['Gender'].unique()

# Print the unique values
for gender in unique_genders:
    print(gender)
```

```
Female
Male
maile
trans-female
```

cis female
cis male
woman
mal
male (cis)
queer/she/they
non-binary
femake
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
a little about you
malr
p
femail
cis man
ostensibly male, unsure what that really means

In [50]:

```
import pandas as pd

# Read the CSV file
df = pd.read_csv('cleaned_survey.csv')

# Define a mapping to standardize gender categories
gender_mapping = {
    'female': 'Female',
    'male': 'Male',
    'maile': 'Male',
    'trans-female': 'Trans Female',
    'cis female': 'Cis Female',
    'cis male': 'Cis Male',
    'woman': 'Female',
    'mal': 'Male',
    'male (cis)': 'Cis Male',
    'queer/she/they': 'Queer/She/They',
    'non-binary': 'Non-Binary',
    'femake': 'Female',
    'make': 'Male',
    'nah': 'Other',
    'all': 'Other',
    'enby': 'Non-Binary',
    'fluid': 'Fluid',
    'genderqueer': 'Genderqueer',
    'female ': 'Female',
    'androgyne': 'Androgyne',
    'agender': 'Agender',
    'cis-female/femme': 'Cis Female/Femme',
    'guy (-ish) ^_^': 'Other',
    'male leaning androgynous': 'Androgyne',
    'man': 'Male',
    'trans woman': 'Trans Female',
    'msle': 'Male',
    'neuter': 'Neuter',
    'female (trans)': 'Trans Female',
    'queer': 'Queer',
    'female (cis)': 'Cis Female',
    'mail': 'Male',
    'malr': 'Male',
    'p': 'Other',
    'femail': 'Female',
    'cis man': 'Cis Male',
    'ostensibly male, unsure what that really means': 'Other'
}

# Standardize the Gender column
df['Gender'] = df['Gender'].str.lower()
df['Gender'] = df['Gender'].apply(lambda x: gender_mapping.get(x, 'Other'))

# Save the updated DataFrame back to the CSV file
df.to_csv('cleaned_survey.csv', index=False)
```

In [54]:
```
import pandas as pd

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Filter rows with ages between 0 and 100
df = df[(df['Age'] >= 0) & (df['Age'] <= 100)]

# Reset the index to make it continuous
df.reset_index(drop=True, inplace=True)

# Save the updated DataFrame to a new CSV file
df.to_csv('cleaned_survey.csv', index=False)
```

In [ ]:
```
```

In [56]:
```
import pandas as pd

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in the Gender column
unique_country = df['Country'].unique()

# Print the unique values
for country in unique_country:
    print(country)
```

```
United States
Canada
United Kingdom
Bulgaria
France
Portugal
Netherlands
Switzerland
Poland
Australia
Germany
Russia
Mexico
Brazil
Slovenia
Costa Rica
Austria
Ireland
India
South Africa
Italy
Sweden
Colombia
Latvia
Romania
Belgium
New Zealand
Spain
Finland
Uruguay
Israel
Bosnia and Herzegovina
Hungary
Singapore
```

```
Japan
Nigeria
Croatia
Norway
Thailand
Denmark
Bahamas, The
Greece
Moldova
Georgia
China
Czech Republic
```

In [57]:
```python
import pandas as pd

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')


df['state'].fillna('Unknown', inplace=True)



# Save the updated DataFrame to a new CSV file
df.to_csv('cleaned_survey.csv', index=False)
```

In [59]:
```python
import pandas as pd

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in the Gender column
unique_self_employed = df['self_employed'].unique()

# Print the unique values
for self_employed in unique_self_employed:
    print(self_employed)
```

```
nan
Yes
No
```

In [61]:
```python
import pandas as pd

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')


df['self_employed'].fillna('Unknown', inplace=True)



# Save the updated DataFrame to a new CSV file
df.to_csv('cleaned_survey.csv', index=False)
```

In [62]:
```
import pandas as pd

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in the Gender column
unique_family_history = df['family_history'].unique()

# Print the unique values
for family_history in unique_family_history:
    print(family_history)
```

```
No
Yes
```

In [66]:
```
import pandas as pd

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in the Gender column
unique_treatment = df['treatment'].unique()

# Print the unique values
for treatment in unique_treatment:
    print(treatment)
```

```
Yes
No
```

In [5]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_work_interfere = df['work_interfere'].unique()

# Print the unique values
for work_interfere in unique_work_interfere:
    print(work_interfere)
```

```
Often
Rarely
Never
Sometimes
nan
```

In [6]:
```
import pandas as pd

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')


df['work_interfere'].fillna('Unknown', inplace=True)



# Save the updated DataFrame to a new CSV file
df.to_csv('cleaned_survey.csv', index=False)
```

In [8]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_no_employees = df['no_employees'].unique()

# Print the unique values
for no_employees in unique_no_employees:
    print(no_employees)
```

```
6-25
More than 1000
26-100
100-500
1-5
500-1000
```

In [9]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_remote_work = df['remote_work'].unique()

# Print the unique values
for remote_work in unique_remote_work:
    print(remote_work)
```

```
No
Yes
```

In [10]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_tech_company = df['tech_company'].unique()

# Print the unique values
for tech_company in unique_tech_company:
    print(tech_company)
```

```
Yes
No
```

In [12]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_benefits = df['benefits'].unique()

# Print the unique values
for benefits in unique_benefits:
    print(benefits)
```

```
Yes
Don't know
No
```

In [13]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_care_options = df['care_options'].unique()

# Print the unique values
for care_options in unique_care_options:
    print(care_options)
```

```
Not sure
No
Yes
```

In [14]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_wellness_program = df['wellness_program'].unique()

# Print the unique values
for wellness_program in unique_wellness_program:
    print(wellness_program)
```

```
No
Don't know
Yes
```

In [15]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_seek_help = df['seek_help'].unique()

# Print the unique values
for seek_help in unique_seek_help:
    print(seek_help)
```

```
Yes
Don't know
No
```

In [16]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_anonymity= df['anonymity'].unique()

# Print the unique values
for anonymity in unique_anonymity:
    print(anonymity)
```

```
Yes
Don't know
No
```

In [17]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_leave= df['leave'].unique()

# Print the unique values
for leave in unique_leave:
    print(leave)
```

```
Somewhat easy
Don't know
Somewhat difficult
Very difficult
Very easy
```

In [18]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_mental_health_consequence= df['mental_health_consequence'].unique()

# Print the unique values
for mental_health_consequence in unique_mental_health_consequence:
    print(mental_health_consequence)
```

```
No
Maybe
Yes
```

In [22]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_phys_health_consequence= df['phys_health_consequence'].unique()

# Print the unique values
for phys_health_consequence in unique_phys_health_consequence:
    print(phys_health_consequence)
```

```
No
Yes
Maybe
```

In [23]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_coworkers= df['coworkers'].unique()

# Print the unique values
for coworkers in unique_coworkers:
    print(coworkers)
```

```
Some of them
No
Yes
```

In [25]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_supervisor= df['supervisor'].unique()

# Print the unique values
for supervisor in unique_supervisor:
    print(supervisor)
```

```
Yes
No
Some of them
```

In [26]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_mental_health_interview= df['mental_health_interview'].unique()

# Print the unique values
for mental_health_interview in unique_mental_health_interview:
    print(mental_health_interview)
```

```
No
Yes
Maybe
```

In [29]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_phys_health_interview= df['phys_health_interview'].unique()

# Print the unique values
for phys_health_interview in unique_phys_health_interview:
    print(phys_health_interview)
```

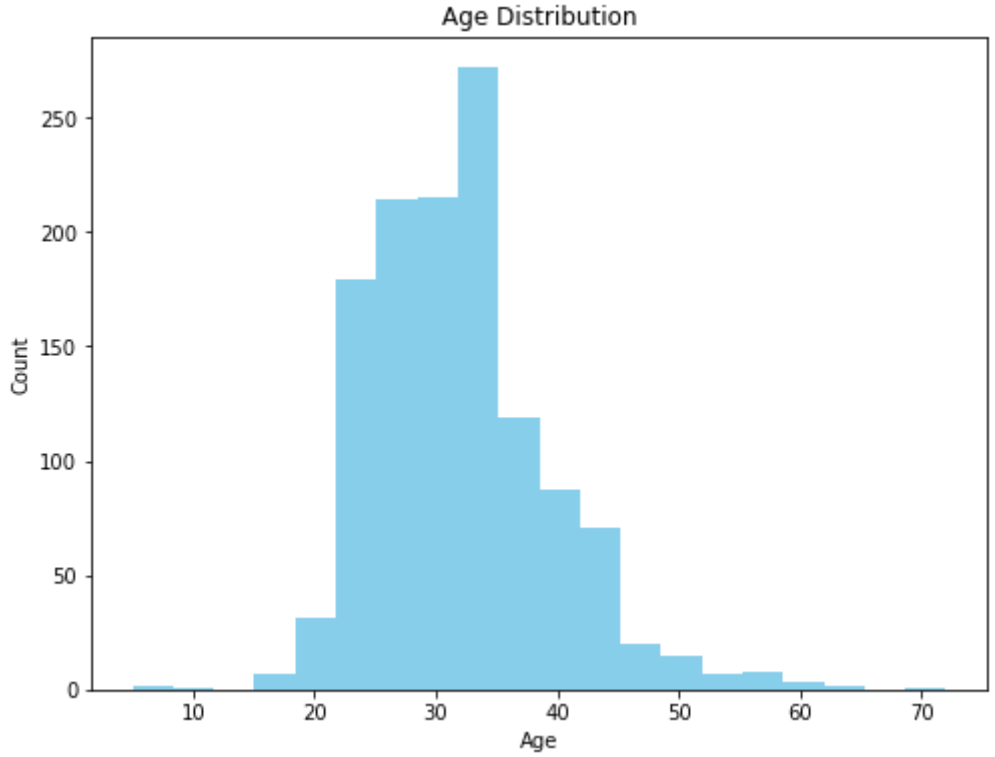
```
Maybe
No
Yes
```

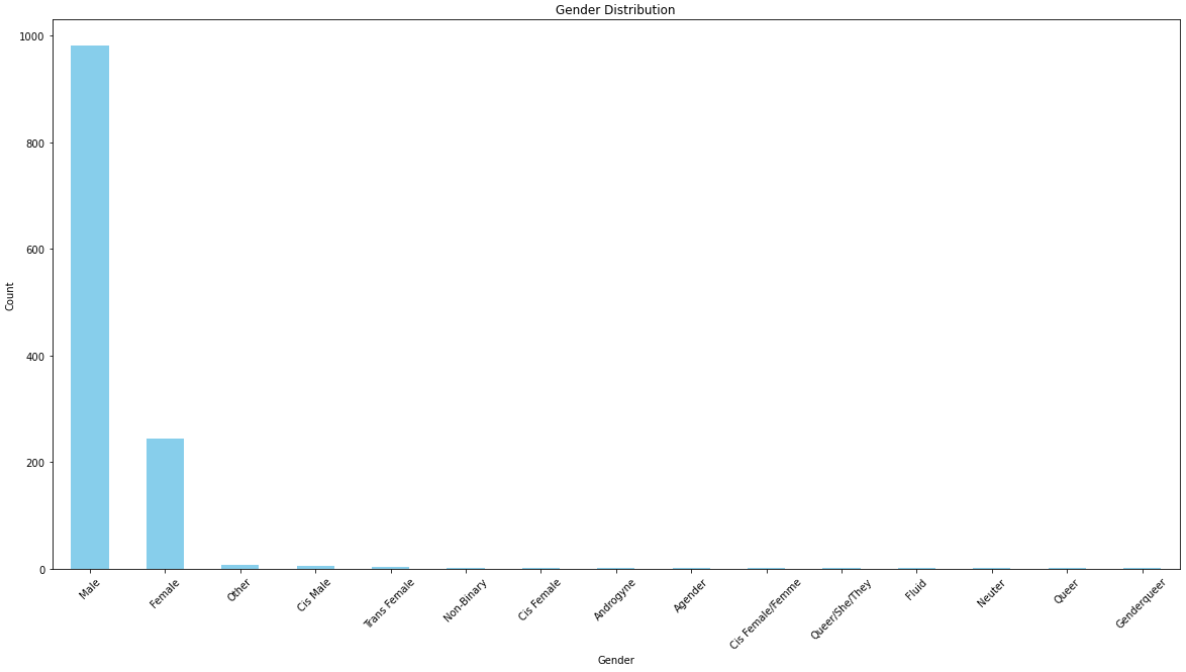
In [30]:
```
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Get the unique values in column
unique_obs_consequence= df['obs_consequence'].unique()

# Print the unique values
for obs_consequence in unique_obs_consequence:
    print(obs_consequence)
```

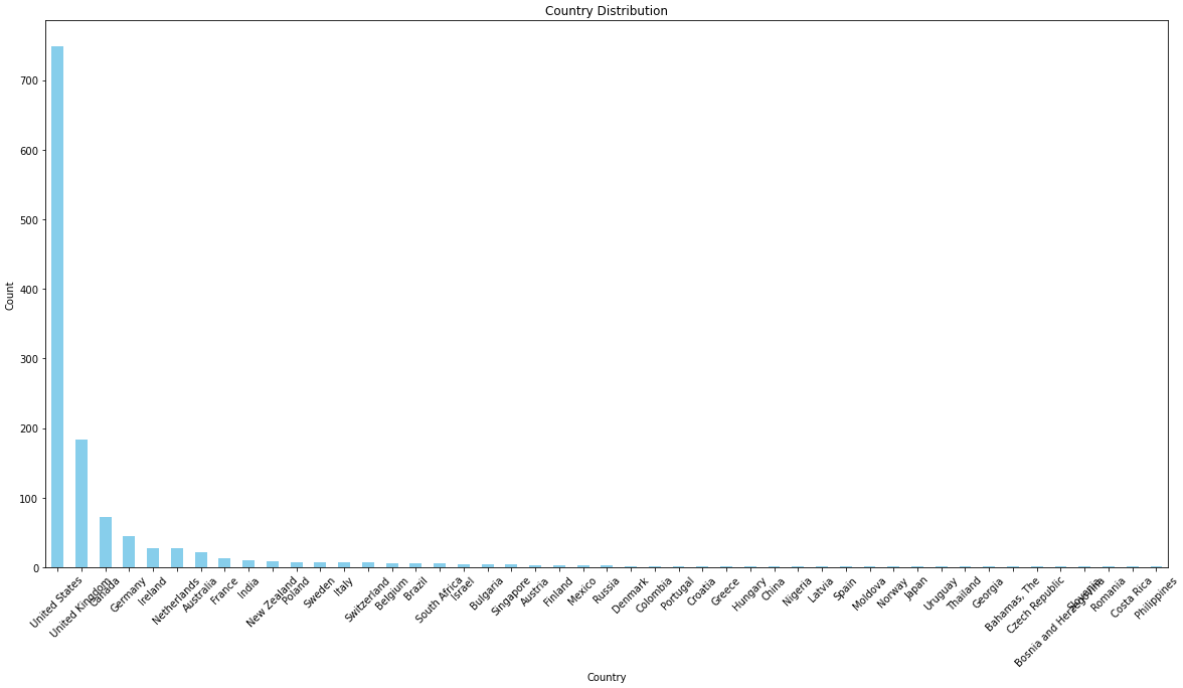
```
No
Yes
```

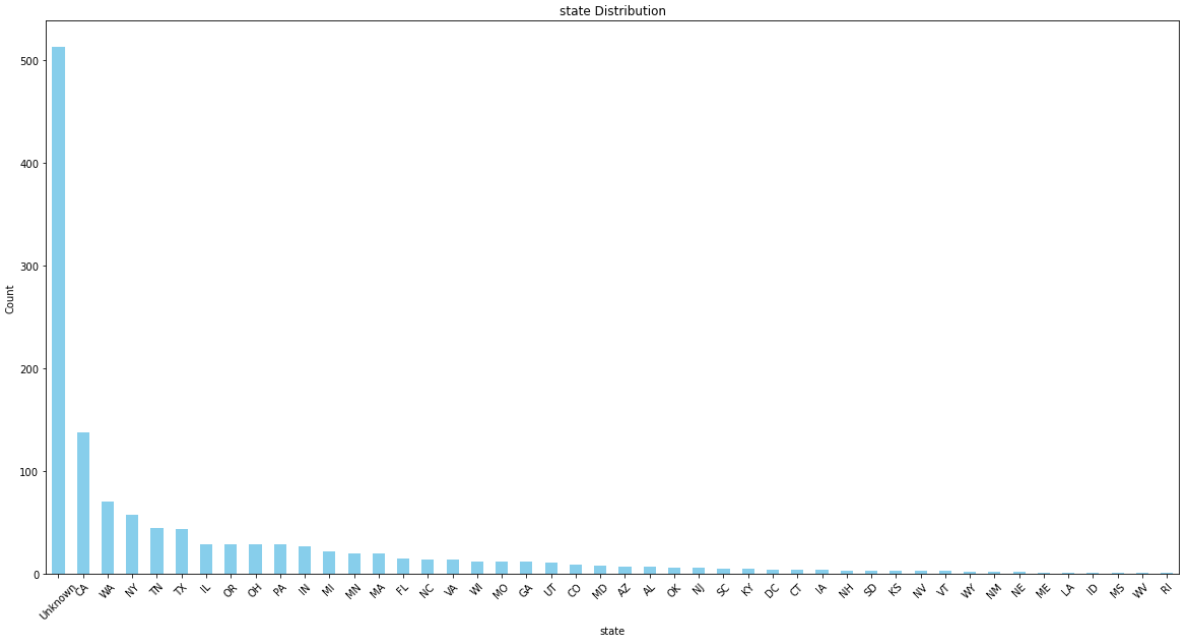
In [34]:
```
import pandas as pd

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')


df['comments'].fillna('Unknown', inplace=True)



# Save the updated DataFrame to a new CSV file
df.to_csv('cleaned_survey.csv', index=False)
```

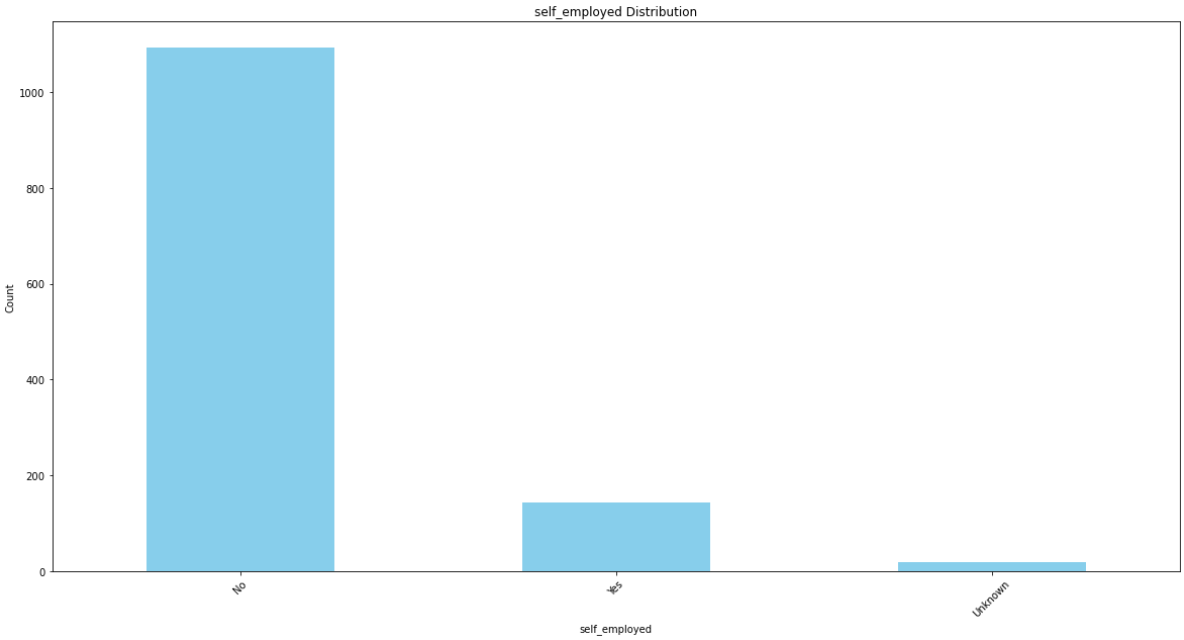
In [38]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Create visualizations for selected columns
columns_to_visualize = [
    'Age', 'Gender', 'Country', 'state', 'self_employed', 'family_history'
    'treatment', 'work_interfere', 'no_employees', 'remote_work',
    'tech_company', 'benefits', 'care_options', 'wellness_program', 'seek_
    'anonymity', 'leave', 'mental_health_consequence', 'phys_health_conseq
    'coworkers', 'supervisor', 'mental_health_interview', 'phys_health_int
    'mental_vs_physical', 'obs_consequence'
]

for column in columns_to_visualize:
    if column == 'Age':
        # Create a histogram for Age
        plt.figure(figsize=(8, 6))
        plt.hist(df[column], bins=20, color='skyblue')
        plt.xlabel(column)
        plt.ylabel('Count')
        plt.title(f'{column} Distribution')
        plt.show()
    elif column == 'Gender':
        # Create a bar chart for Gender
        gender_counts = df[column].value_counts()
        plt.figure(figsize=(20, 10))
        gender_counts.plot(kind='bar', color='skyblue')
        plt.xlabel(column)
        plt.ylabel('Count')
        plt.title(f'{column} Distribution')
        plt.xticks(rotation=45)
        plt.show()
    else:
        # Create a bar chart for other categorical columns
        category_counts = df[column].value_counts()
        plt.figure(figsize=(20, 10))
        category_counts.plot(kind='bar', color='skyblue')
        plt.xlabel(column)
        plt.ylabel('Count')
        plt.title(f'{column} Distribution')
        plt.xticks(rotation=45)
        plt.show()
```

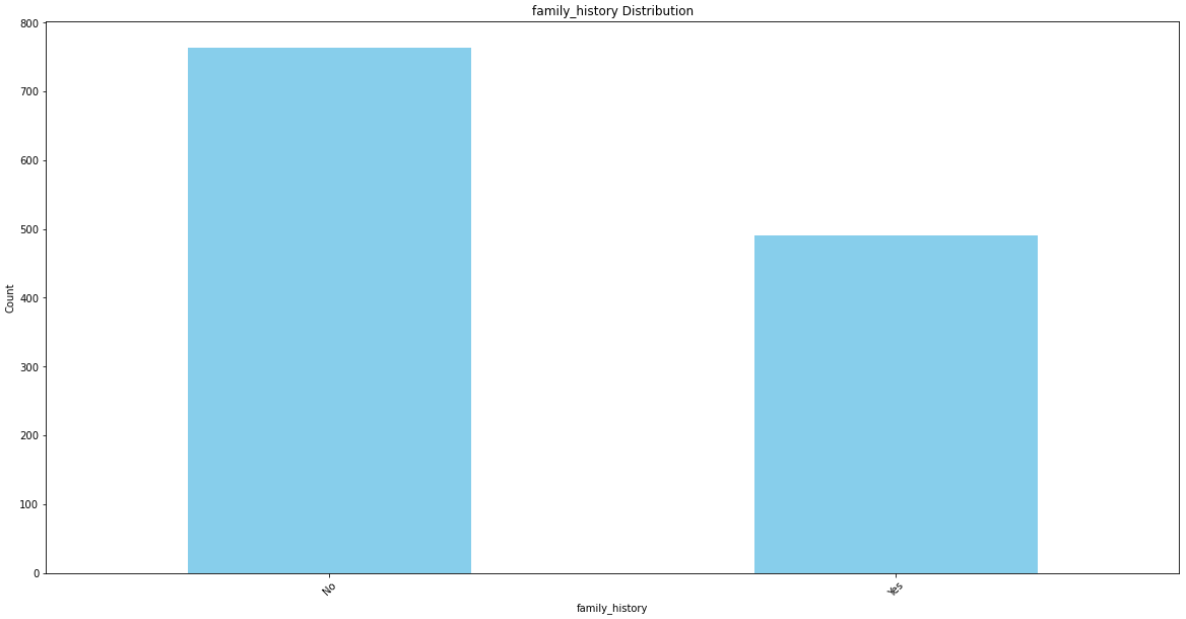

Age Distribution
Count
Age
0
50
100
150
200
250
10
20
30
40
50
60
70

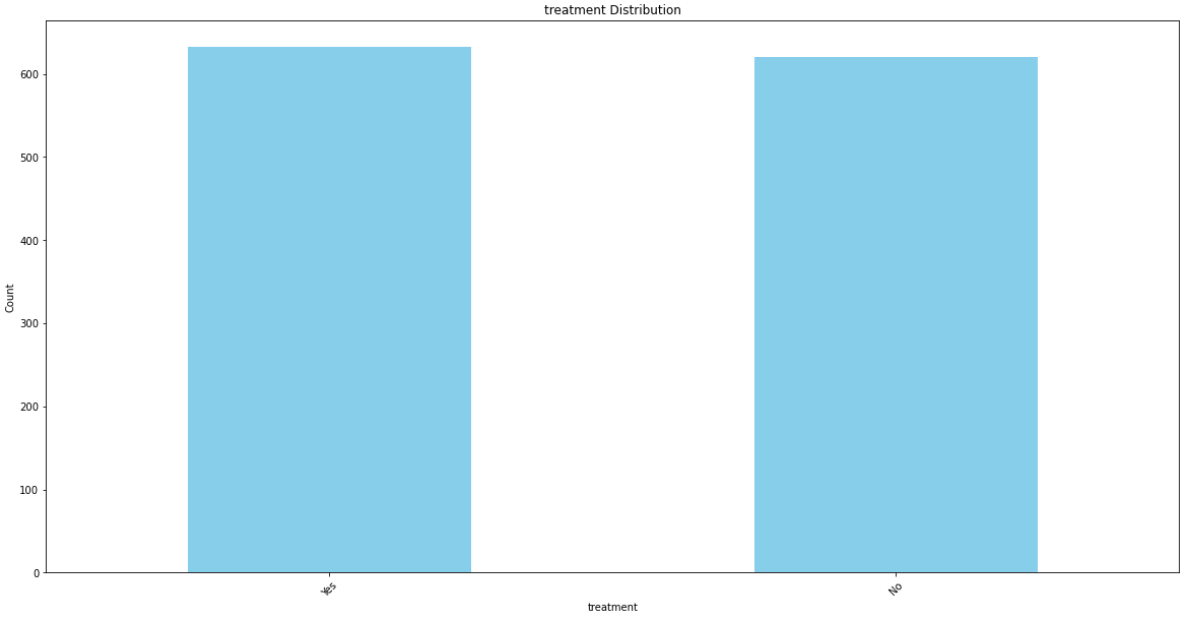

Gender Distribution
Count
Gender
0
200
400
600
800
1000
Male
Female
Other
Cis Male
Trans Female
Non-Binary
Cis Female
Androgyne
Agender
Cis Female/Femme
Queer/She/They
Fluid
Neuter
Queer
Genderqueer

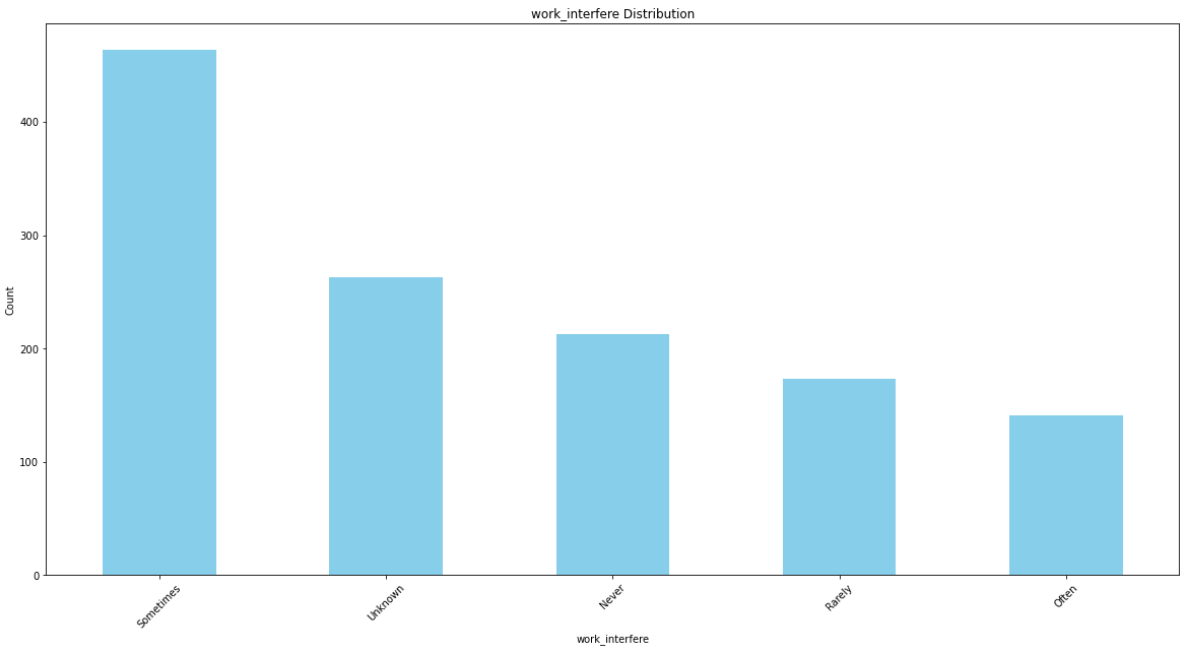

Country Distribution
Count
Country
United States
United Kingdom
Canada
Germany
Ireland
Netherlands
Australia
France
India
New Zealand
Poland
Sweden
Italy
Switzerland
Belgium
Brazil
South Africa
Israel
Bulgaria
Singapore
Austria
Finland
Mexico
Russia
Denmark
Colombia
Portugal
Croatia
Greece
Hungary
China
Nigeria
Latvia
Spain
Moldova
Norway
Japan
Uruguay
Thailand
Georgia
Bahamas, The
Czech Republic
Bosnia and Herzegovina
Slovenia
Romania
Costa Rica
Philippines

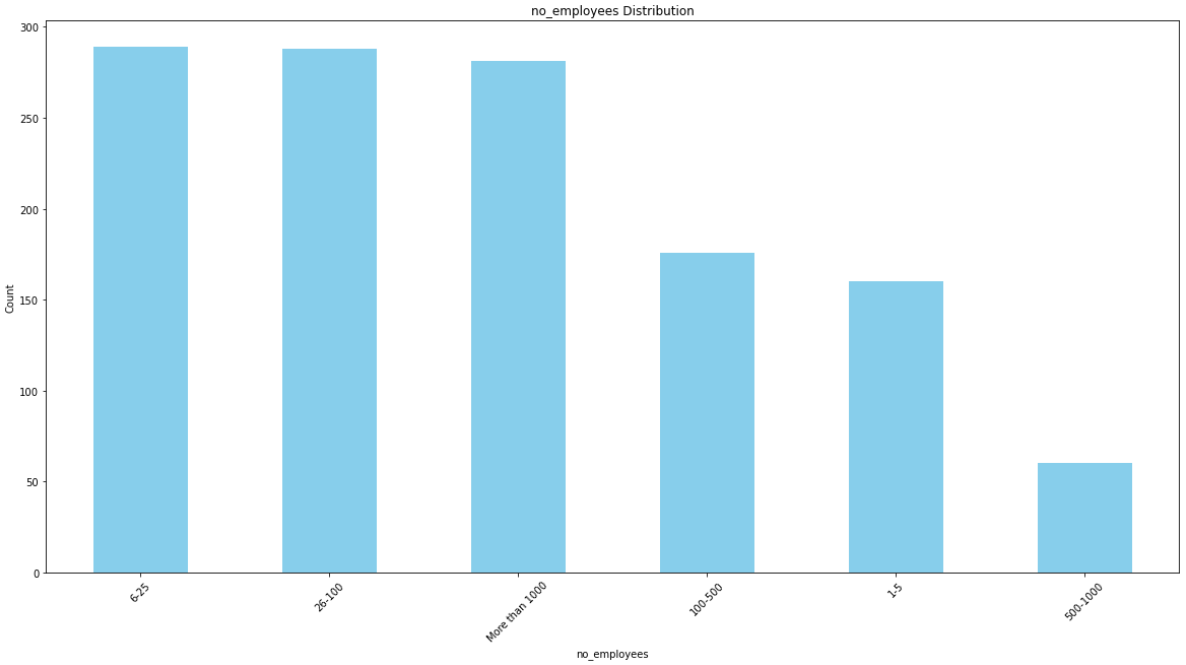

state Distribution
Count
state
Unknown
CA
WA
NY
TN
TX
IL
OR
OH
PA
IN
MI
MN
MA
FL
NC
VA
WI
MO
GA
UT
CO
MD
AZ
AL
OK
NJ
SC
KY
DC
CT
IA
NH
SD
KS
NV
VT
WY
NM
NE
ME
LA
ID
MS
WV
RI

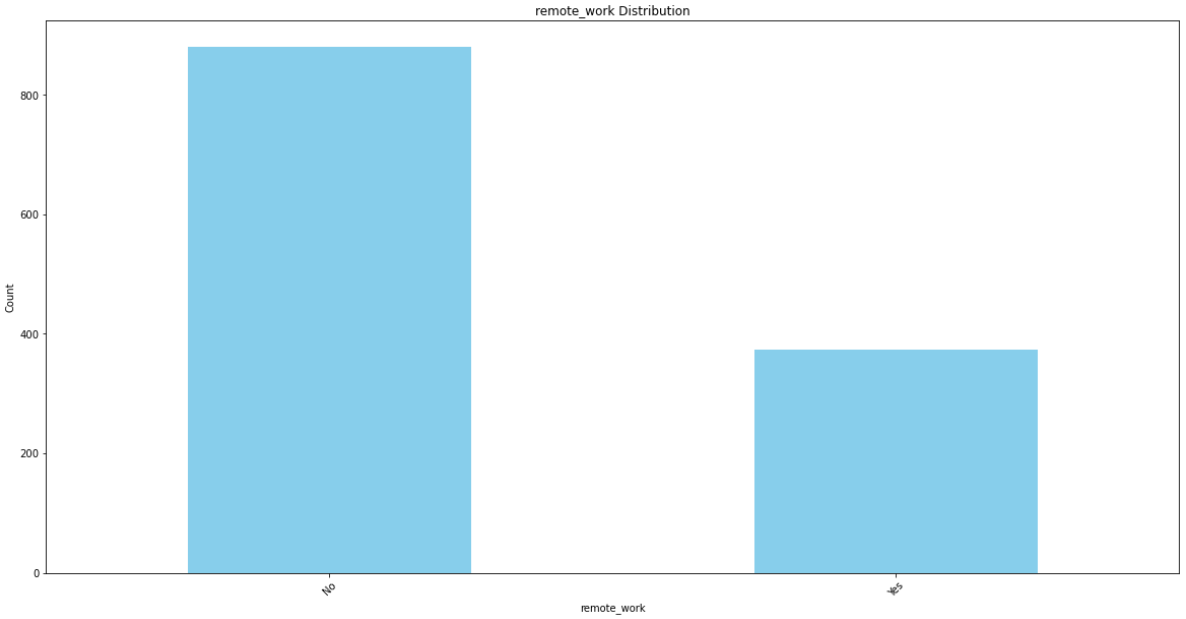

self_employed Distribution
Count
self_employed
No
Yes
Unknown

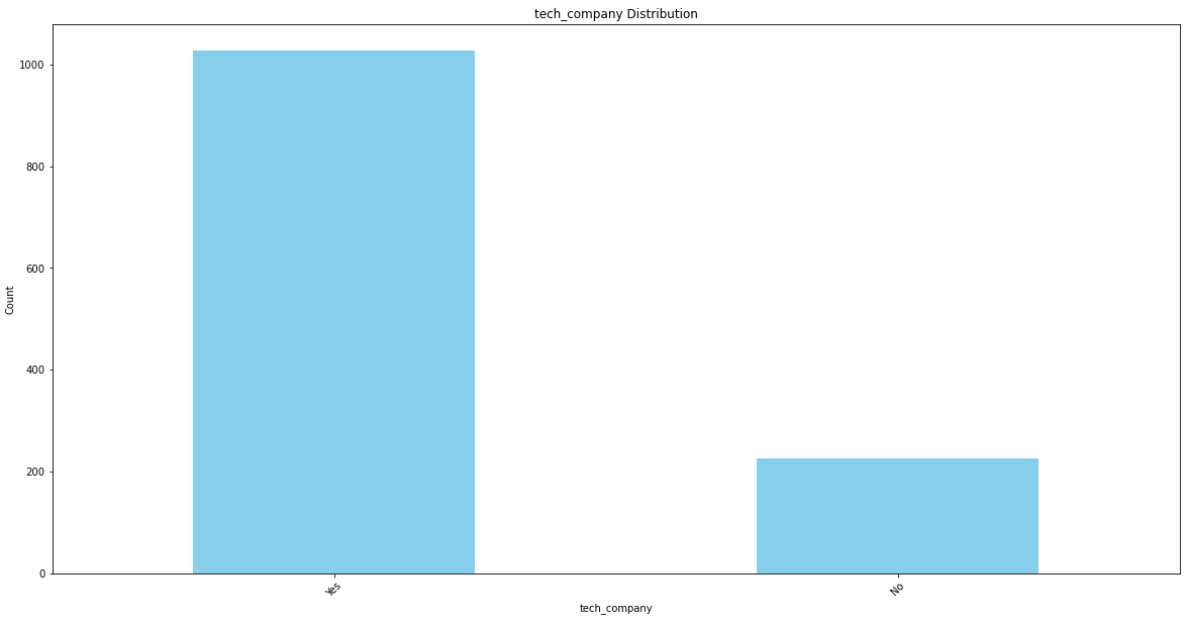

family_history Distribution
Count
family_history
No
Yes

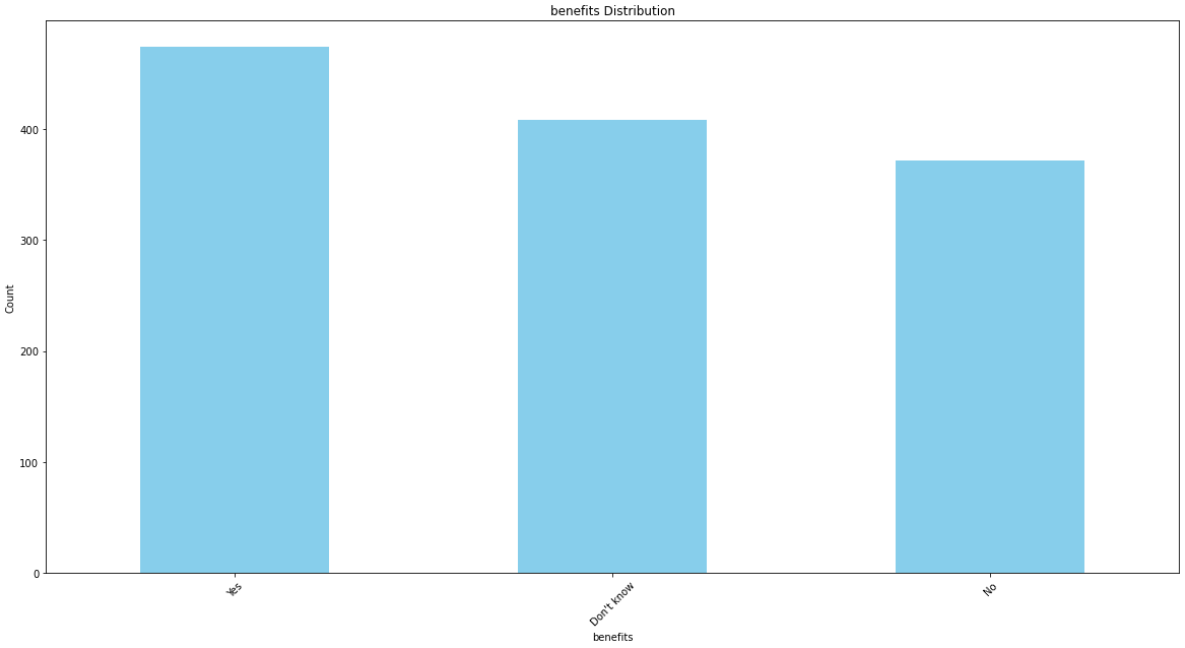

treatment Distribution
Count
treatment
Yes
No

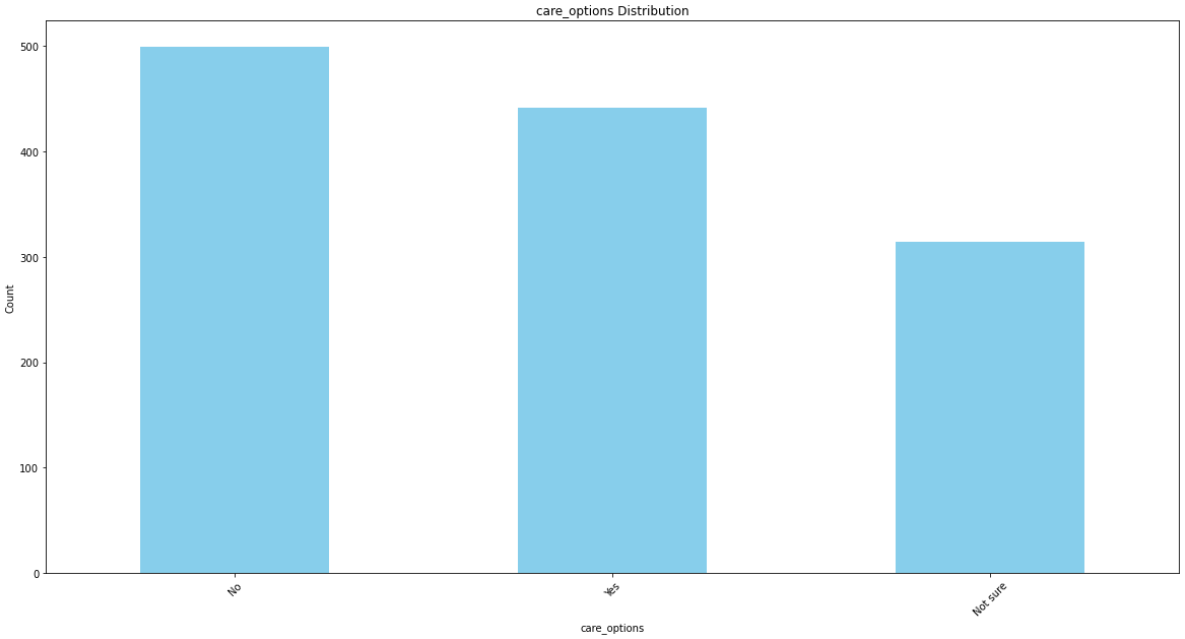

work_interfere Distribution
Count
work_interfere
Sometimes
Unknown
Never
Rarely
Often

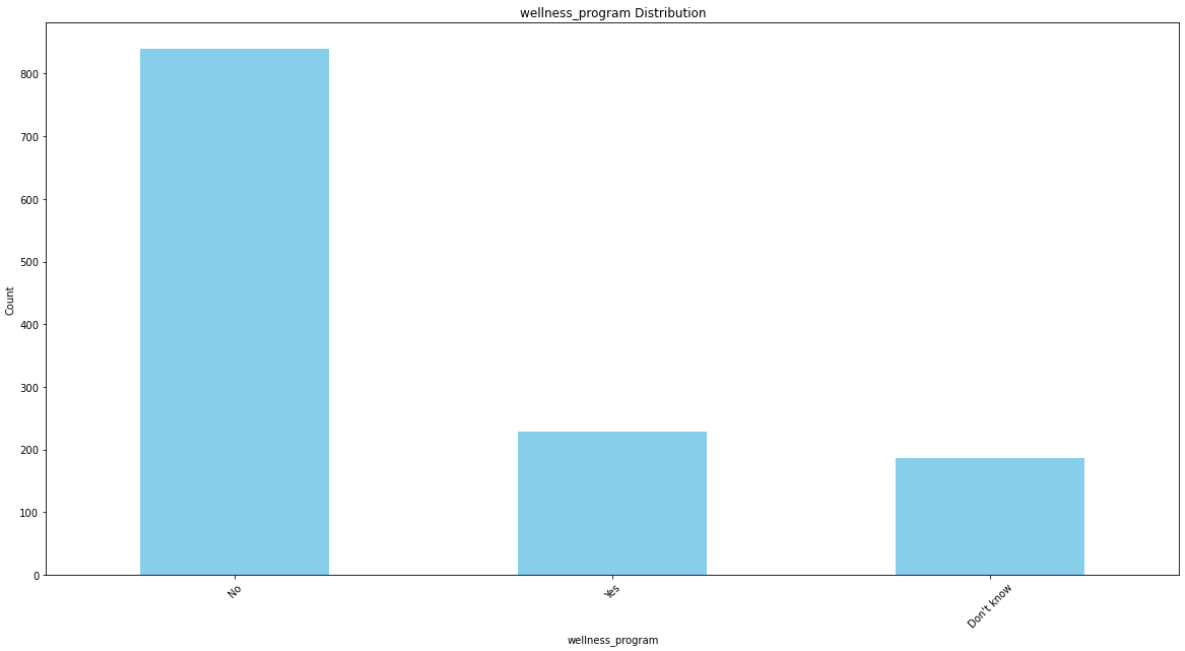

no_employees Distribution
Count
300
250
200
150
100
50
0
6-25
26-100
More than 1000
100-500
1-5
500-1000
no_employees

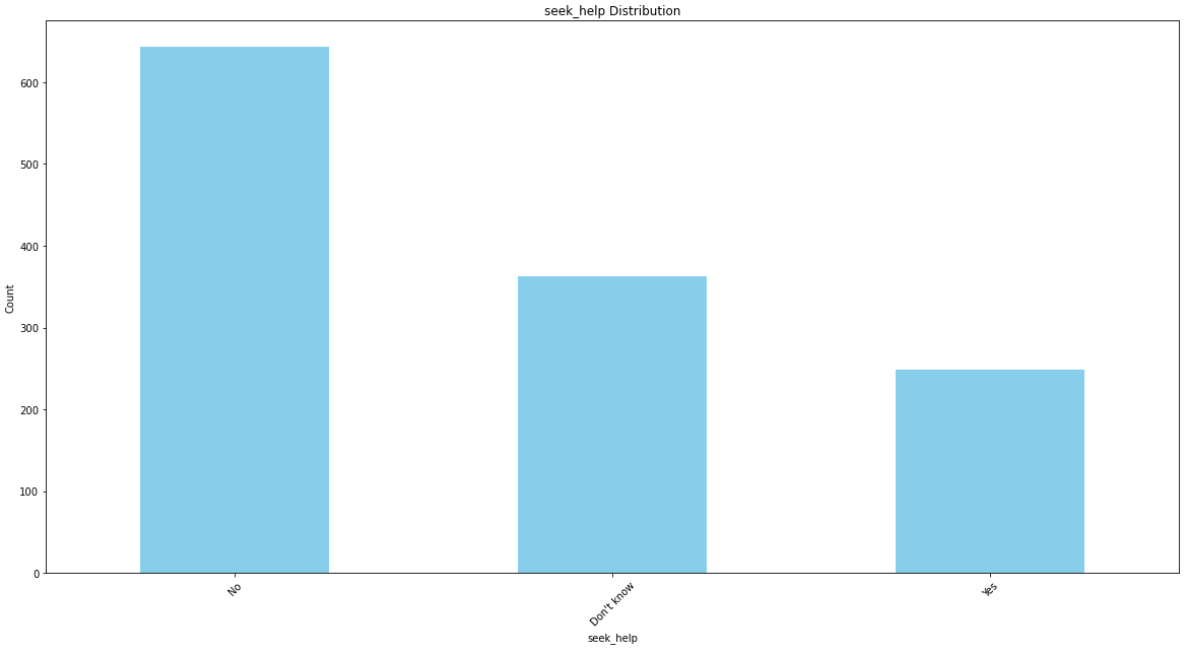

remote_work Distribution
Count
800
600
400
200
0
No
Yes
remote_work

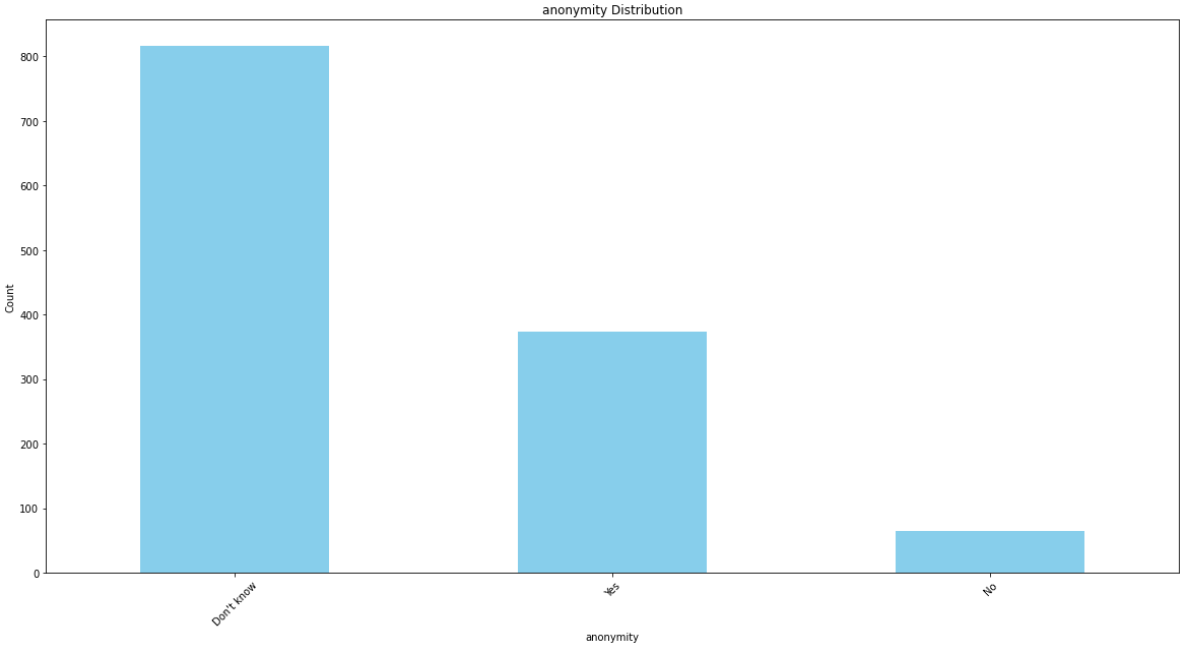

tech_company Distribution
Count
1000
800
600
400
200
0
Yes
No
tech_company

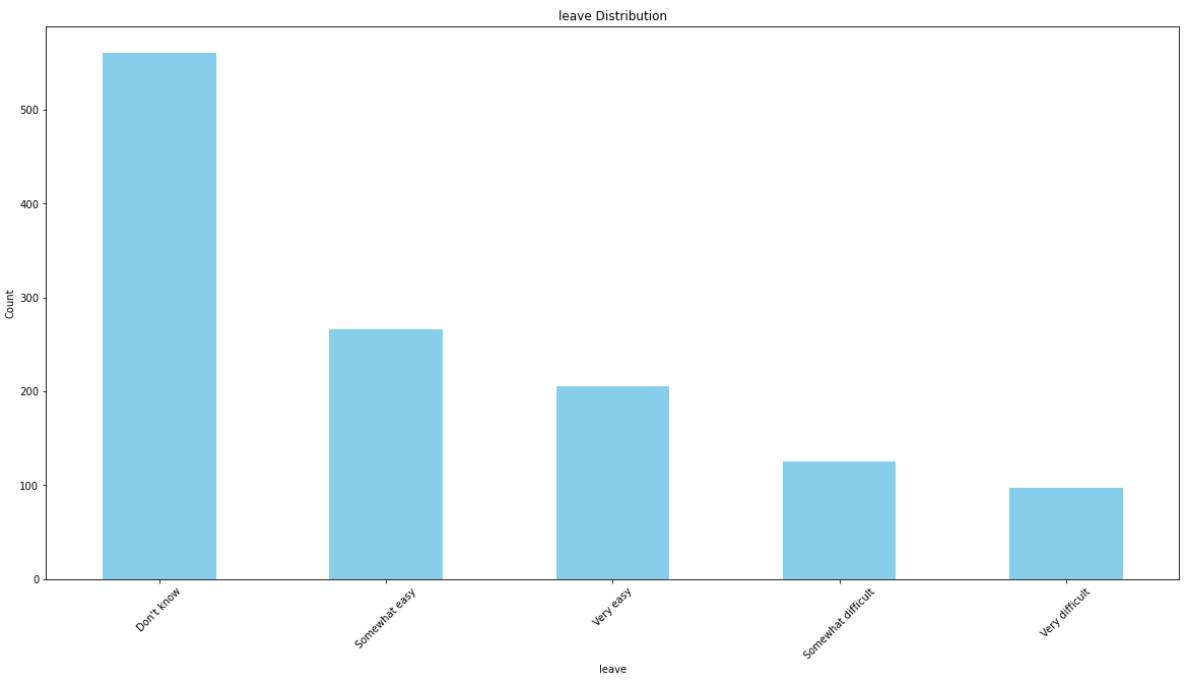

benefits Distribution
Count
0
100
200
300
400
Yes
Don't know
No
benefits

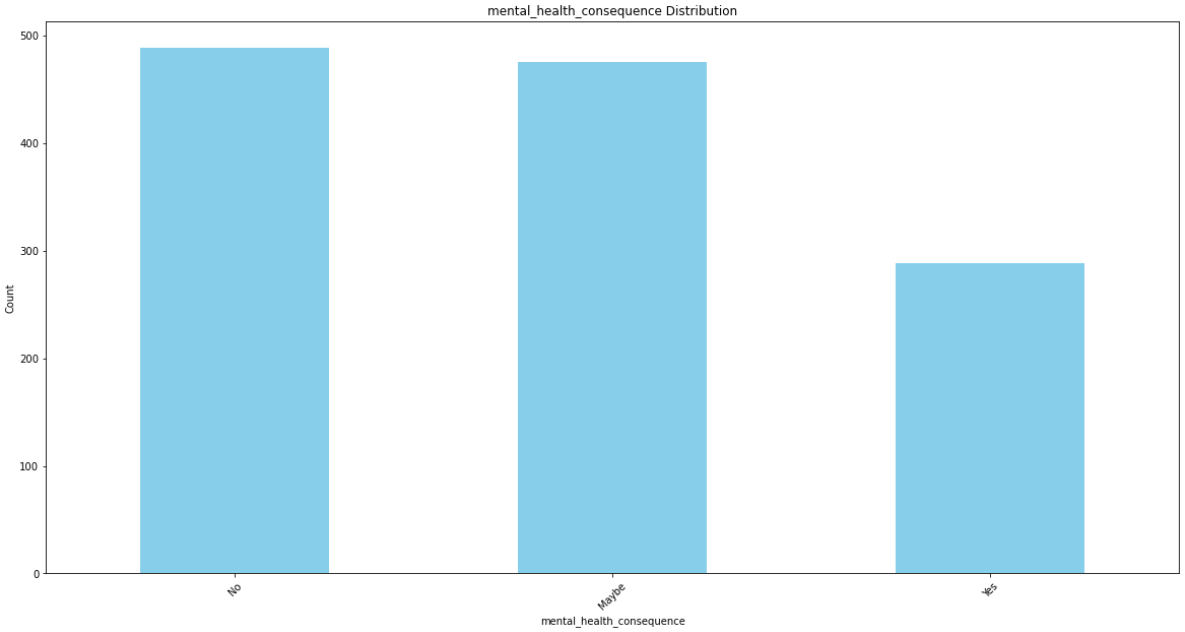

care_options Distribution
Count
0
100
200
300
400
500
No
Yes
Not sure
care_options

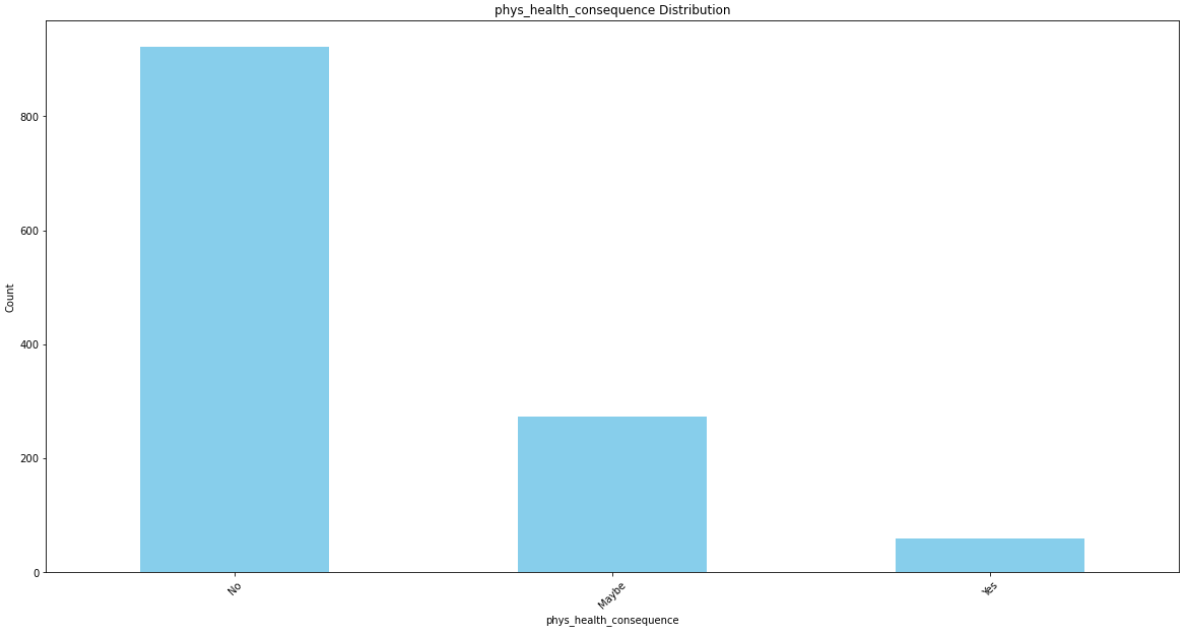

wellness_program Distribution
Count
0
100
200
300
400
500
600
700
800
No
Yes
Don't know
wellness_program

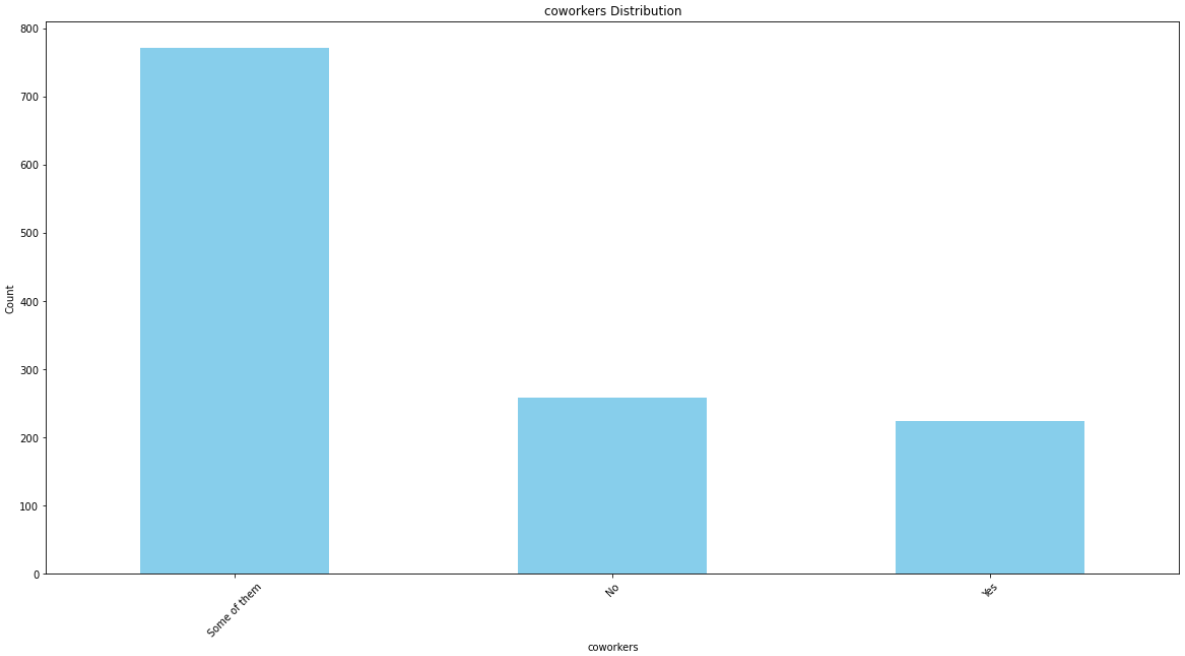

seek_help Distribution
Count
0
100
200
300
400
500
600
No
Don't know
Yes
seek_help

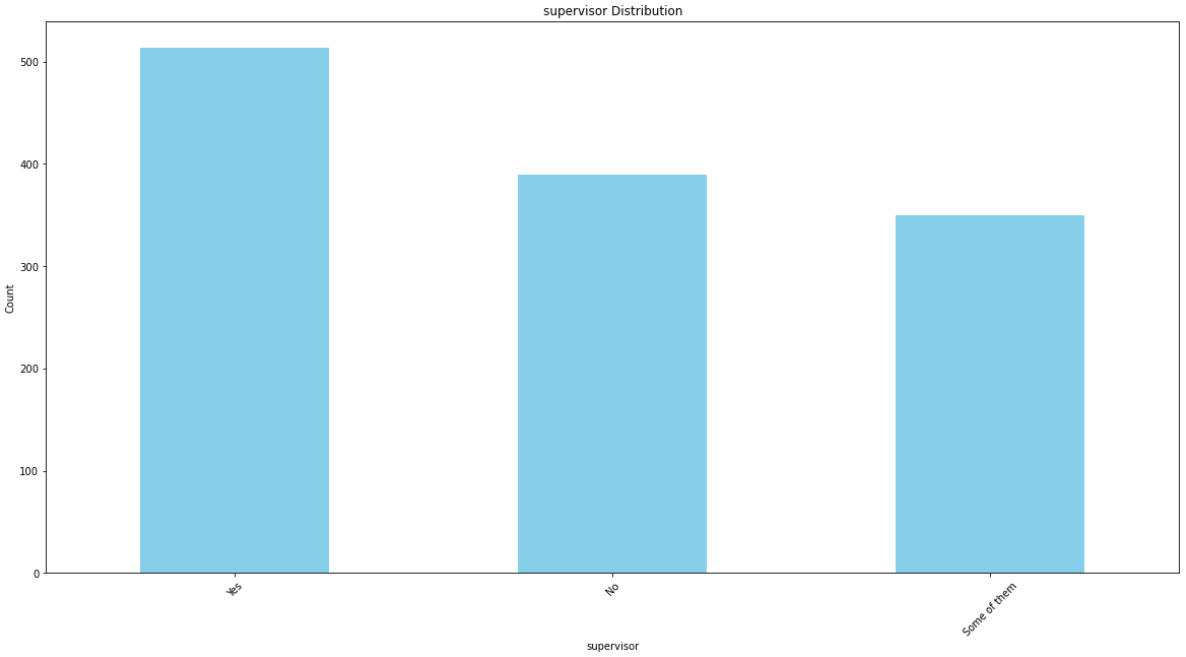

anonymity Distribution
Count
0
100
200
300
400
500
600
700
800
Don't know
Yes
No
anonymity

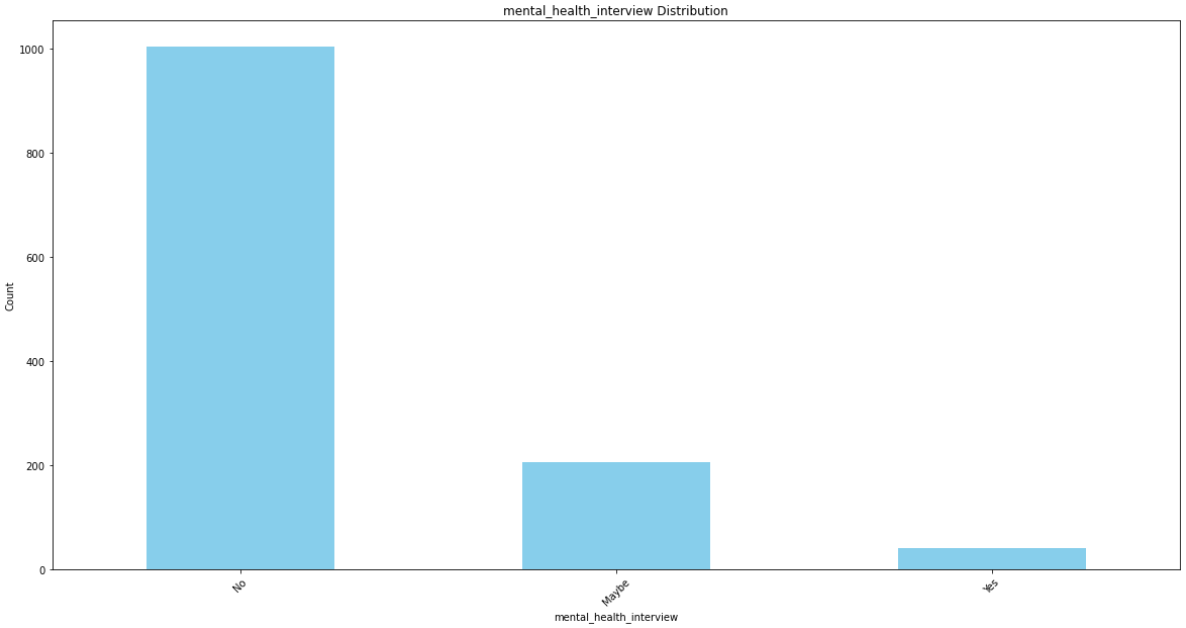

leave Distribution
Count
0
100
200
300
400
500
Don't know
Somewhat easy
Very easy
Somewhat difficult
Very difficult
leave

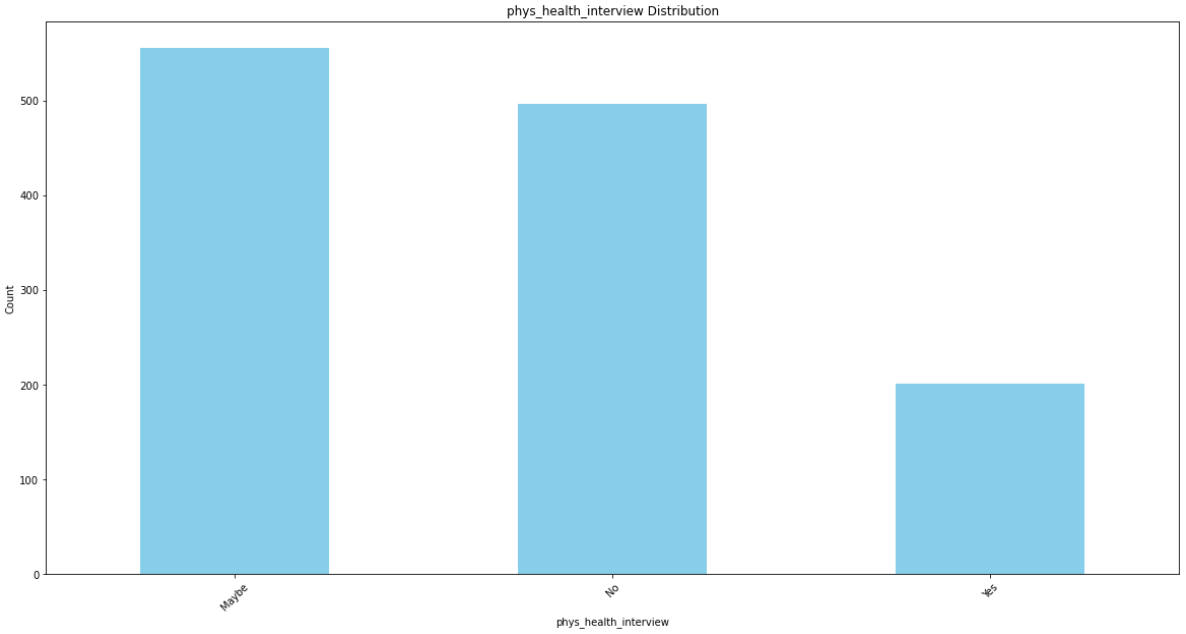

mental_health_consequence Distribution
Count
0
100
200
300
400
500
No
Maybe
Yes
mental_health_consequence

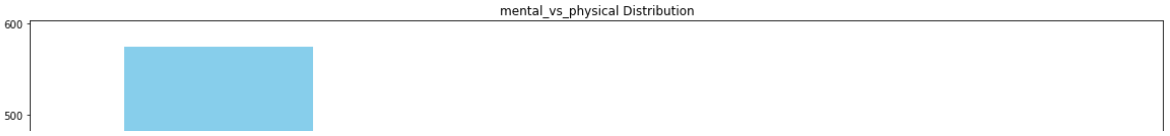

phys_health_consequence Distribution
Count
0
200
400
600
800
No
Maybe
Yes
phys_health_consequence


coworkers Distribution
Count
0
100
200
300
400
500
600
700
800
Some of them
No
Yes
coworkers

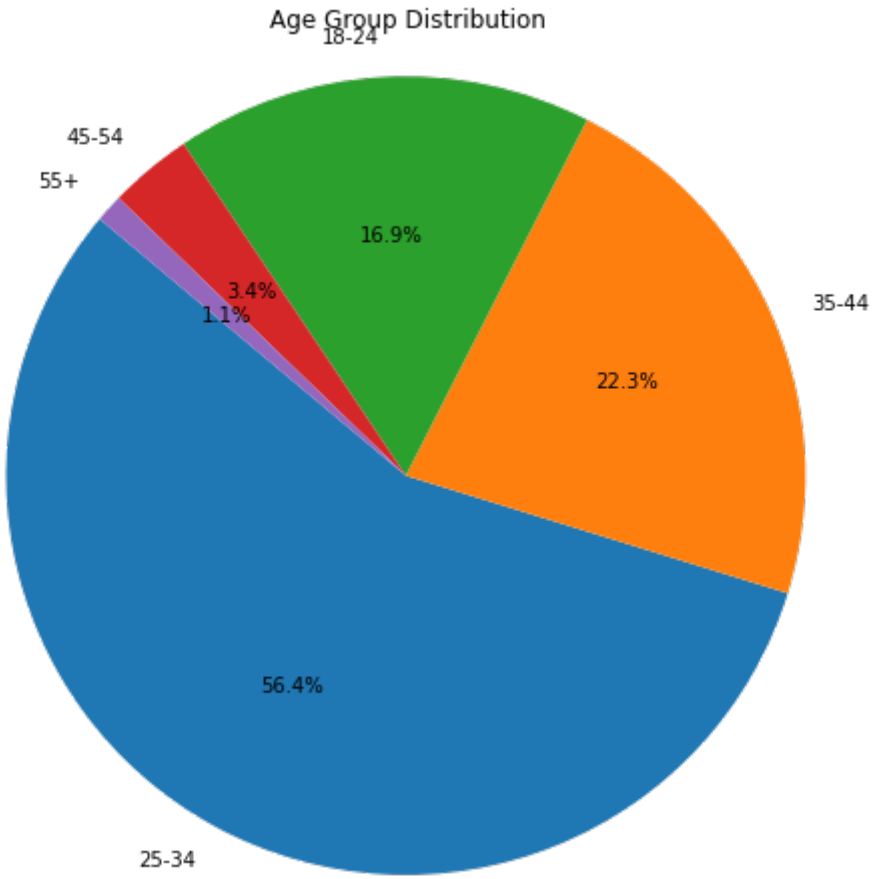

supervisor Distribution
Count
0
100
200
300
400
500
Yes
No
Some of them
supervisor


mental_health_interview Distribution
Count
0
200
400
600
800
1000
No
Maybe
Yes
mental_health_interview


phys_health_interview Distribution
Count
0
100
200
300
400
500
Maybe
No
Yes
phys_health_interview

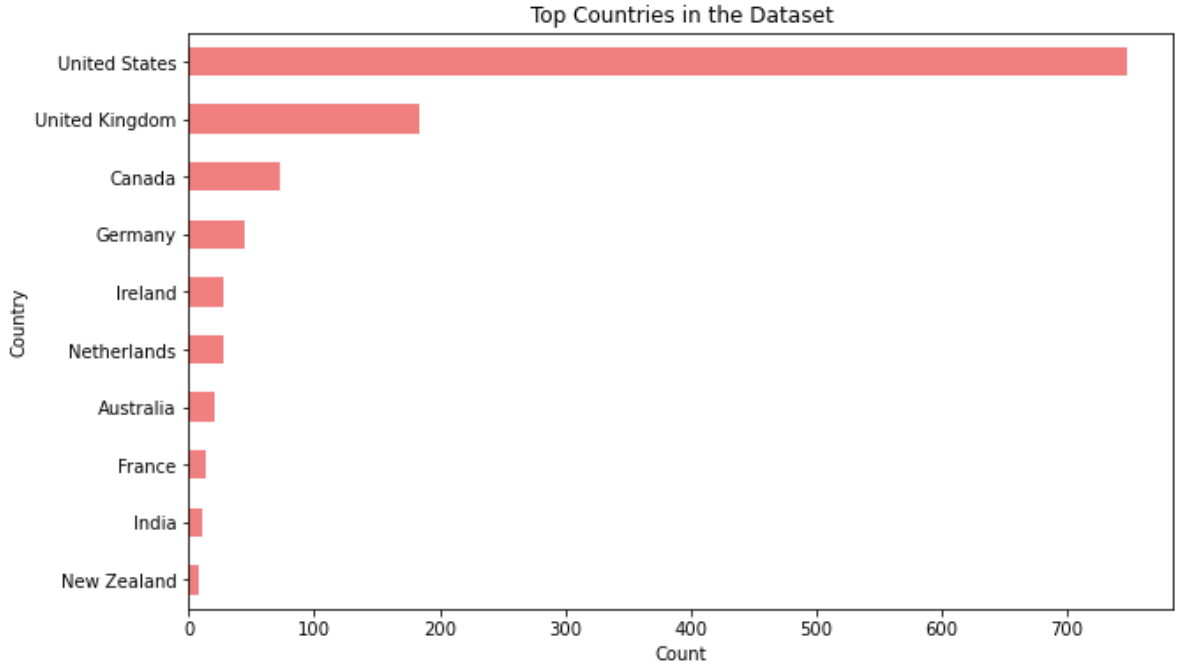


In [41]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Define age groups
bins = [18, 25, 35, 45, 55, 120]
labels = ['18-24', '25-34', '35-44', '45-54', '55+']

# Create a new column 'AgeGroup' based on the age categories
df['AgeGroup'] = pd.cut(df['Age'], bins=bins, labels=labels)

# Calculate the distribution of age groups
age_group_counts = df['AgeGroup'].value_counts()

# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(age_group_counts, labels=age_group_counts.index, autopct='%1.1f%%'
plt.title('Age Group Distribution')
plt.axis('equal')  # Equal aspect ratio ensures that the pie chart is circ
plt.show()
```

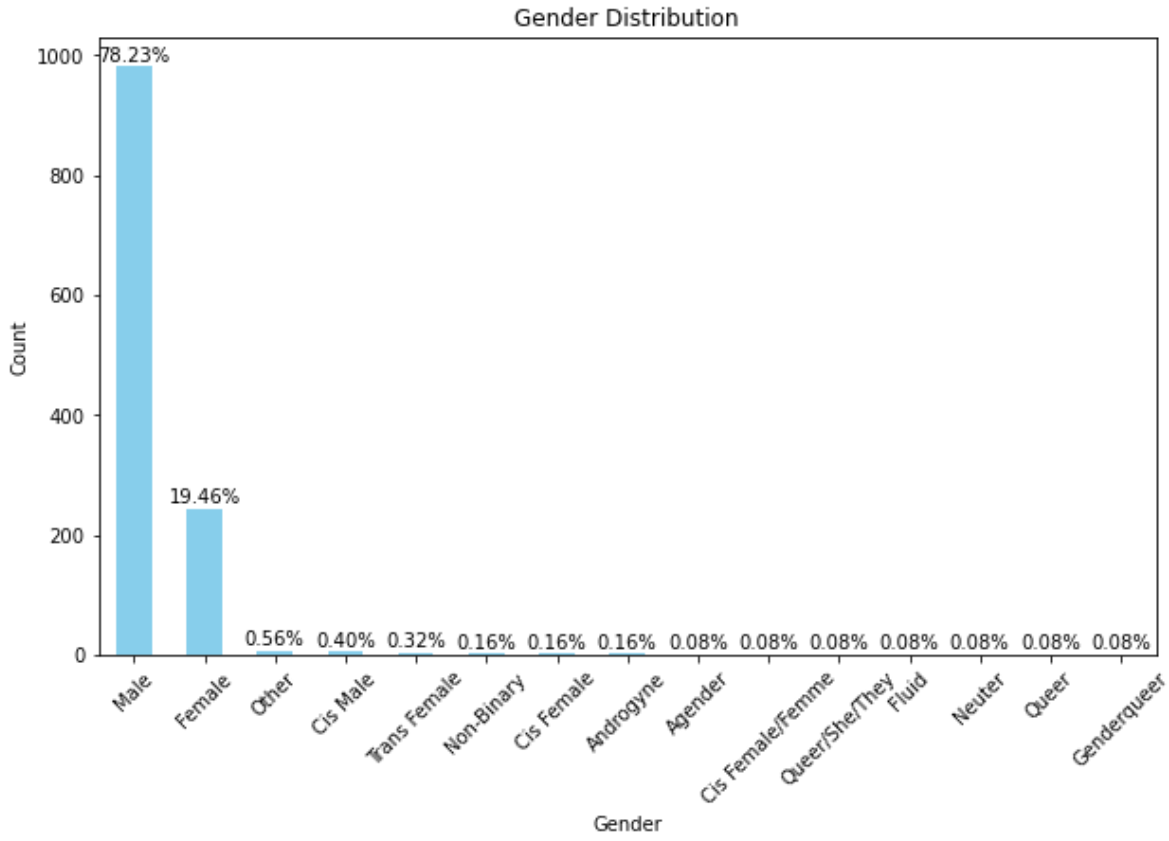


In [42]:

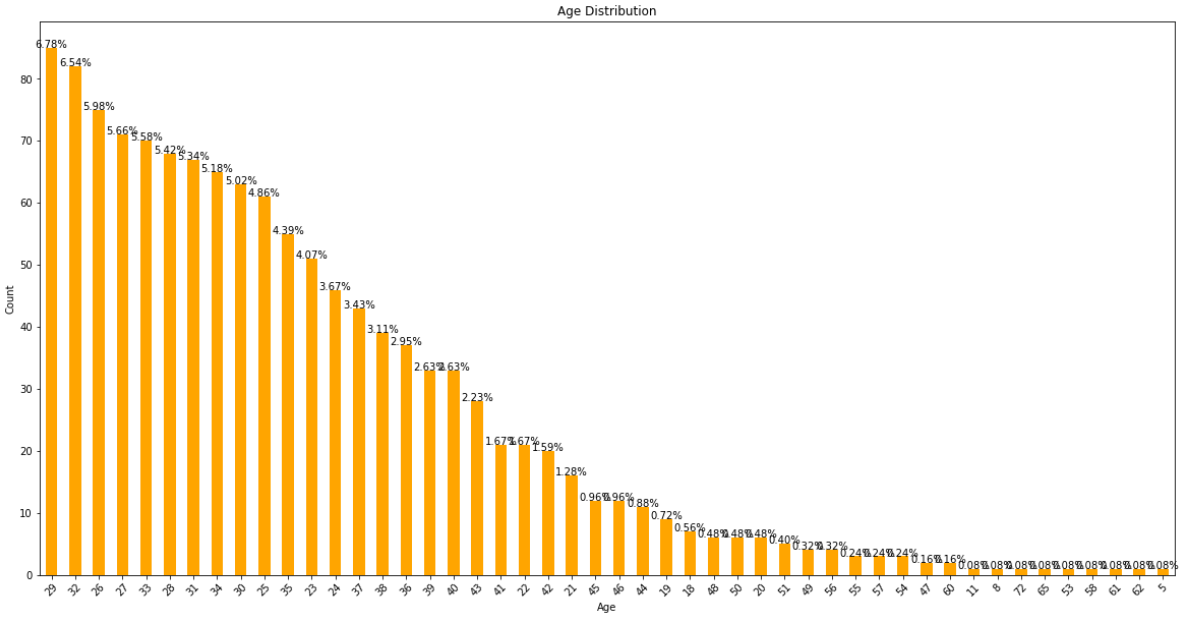
```
import pandas as pd
import matplotlib.pyplot as plt

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')


# Create horizontal bar chart for Country
country_counts = df['Country'].value_counts()
top_n = 10  # Choose how many countries to display
top_countries = country_counts.head(top_n)
plt.figure(figsize=(10, 6))
top_countries.plot(kind='barh', color='lightcoral')
plt.xlabel('Count')
plt.ylabel('Country')
plt.title('Top Countries in the Dataset')
plt.gca().invert_yaxis()
plt.show()
```

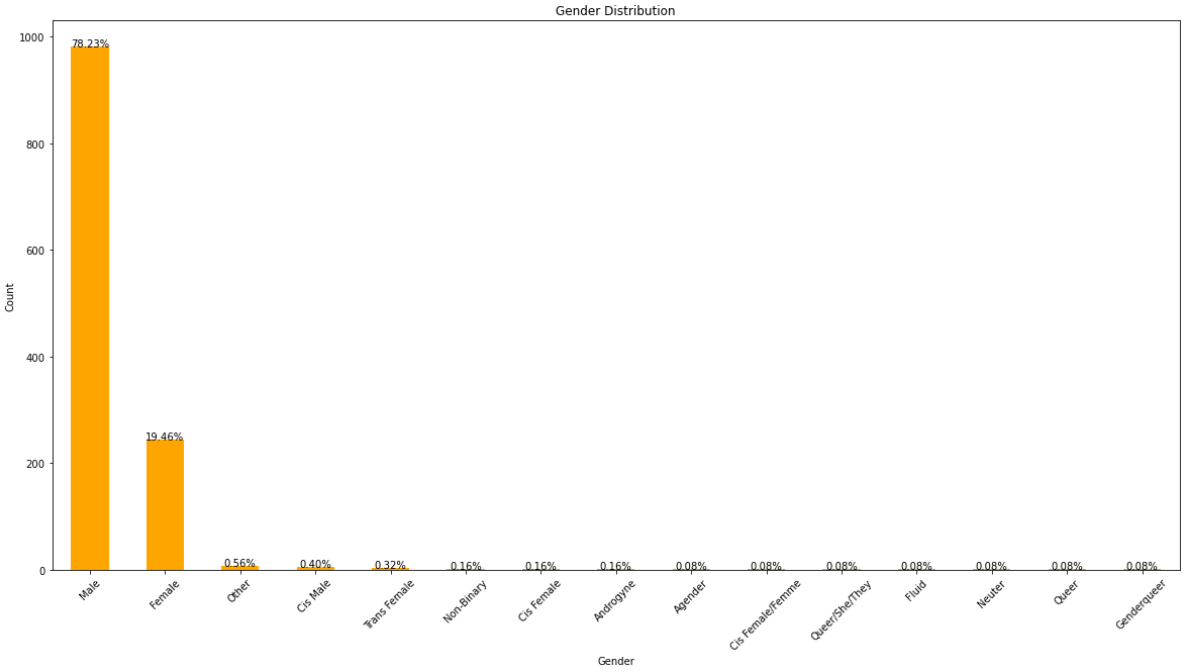


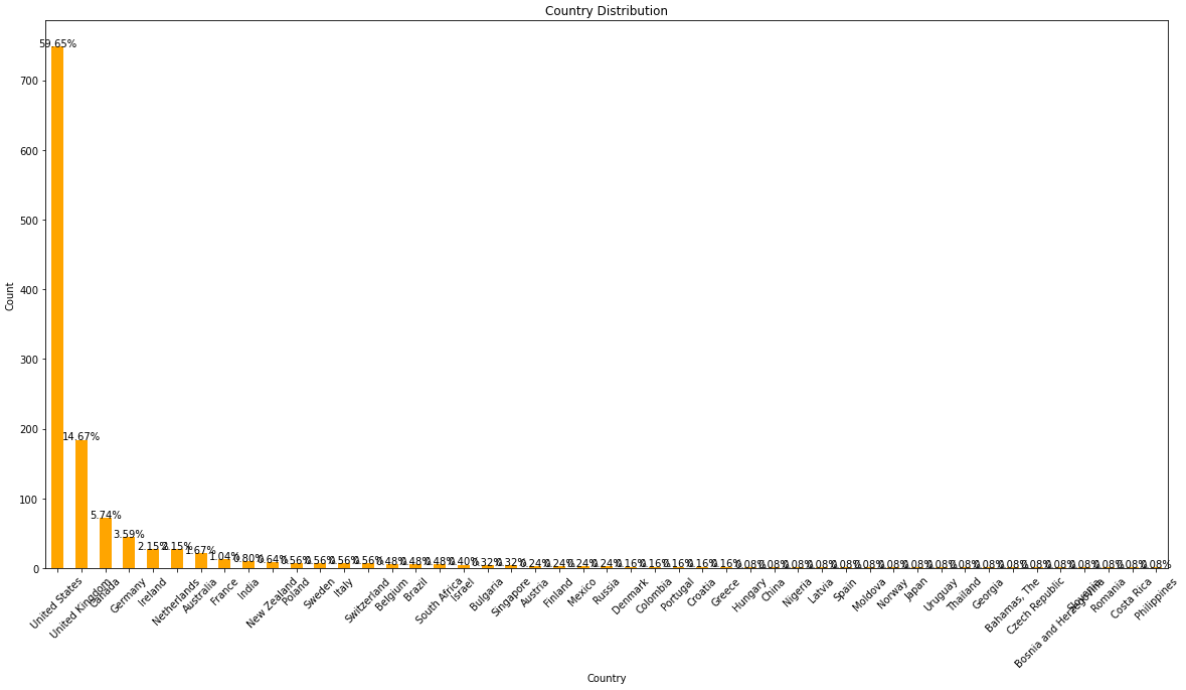
In [43]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Create a bar chart for Gender distribution with percentage labels
gender_counts = df['Gender'].value_counts()
total_count = len(df)
percentage_values = (gender_counts / total_count) * 100

plt.figure(figsize=(10, 6))
bars = gender_counts.plot(kind='bar', color='skyblue')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Gender Distribution')
plt.xticks(rotation=45)

# Add percentage labels on top of each bar
for i, count in enumerate(gender_counts):
    plt.text(i, count + 10, f'{percentage_values[i]:.2f}%', ha='center')

plt.show()
```

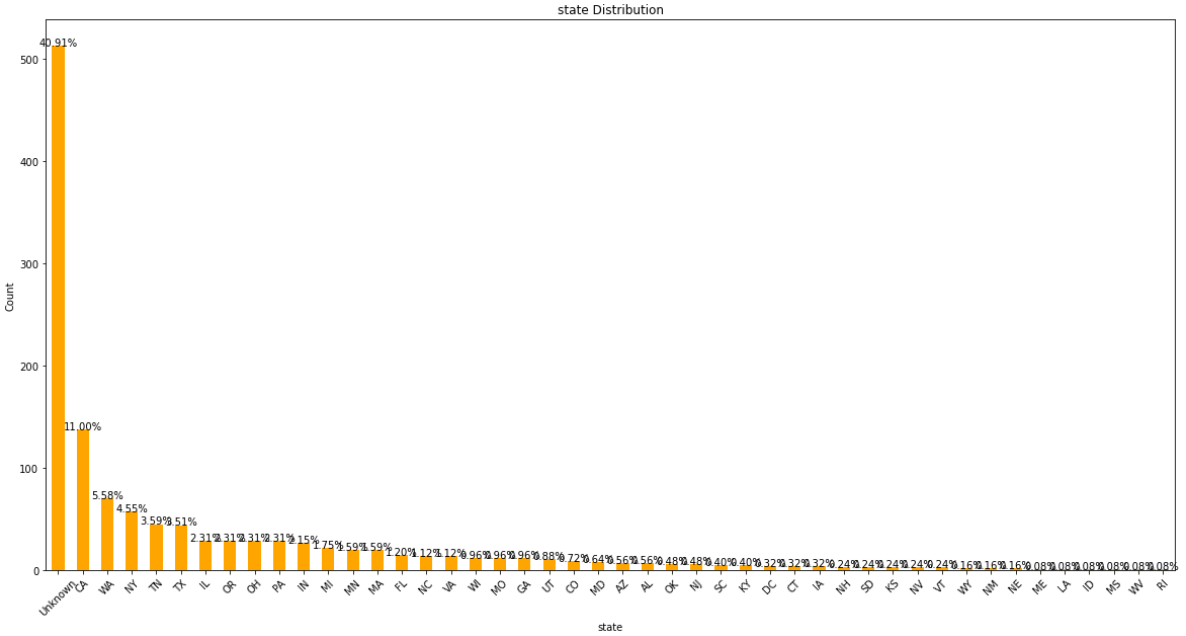


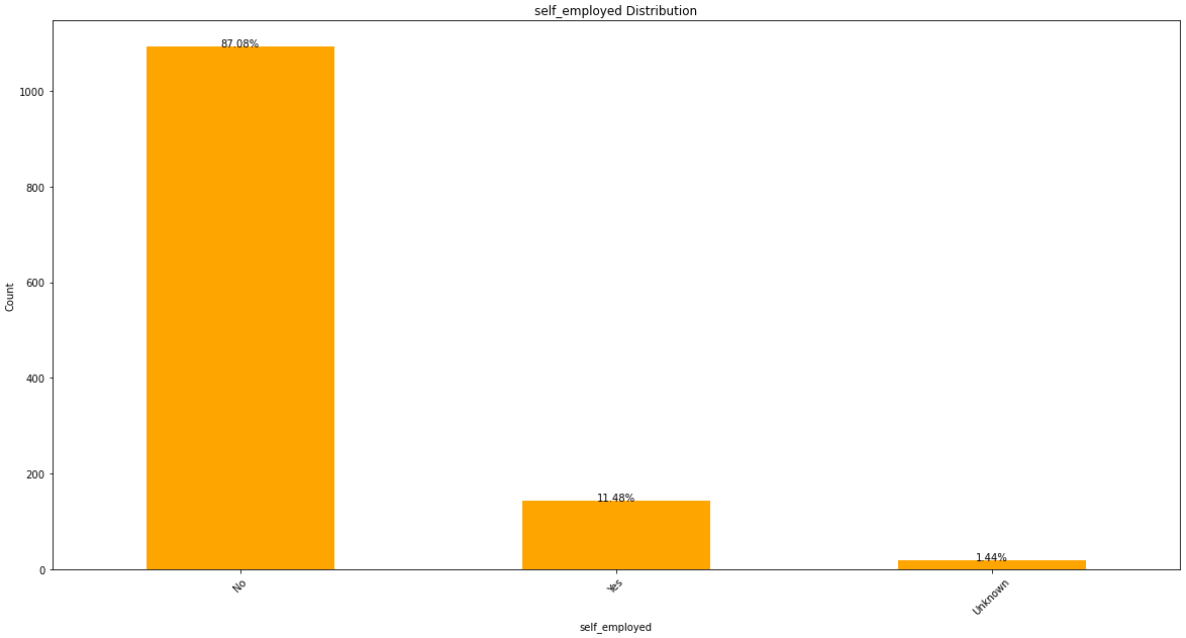
In [48]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# Define a list of columns to visualize with percentages
columns_to_visualize = [
    'Age', 'Gender', 'Country', 'state', 'self_employed', 'family_history'
    'treatment', 'work_interfere', 'no_employees', 'remote_work',
    'tech_company', 'benefits', 'care_options', 'wellness_program', 'seek_
    'anonymity', 'leave', 'mental_health_consequence', 'phys_health_conseq
    'coworkers', 'supervisor', 'mental_health_interview', 'phys_health_int
    'mental_vs_physical', 'obs_consequence'
]

# Customize visualizations for each column
for column in columns_to_visualize:
    if column in ['Age', 'Gender', 'Country','state', 'self_employed', 'fa
    'treatment', 'work_interfere', 'no_employees', 'remote_work',
    'tech_company', 'benefits', 'care_options', 'wellness_program', 'seek_
    'anonymity', 'leave', 'mental_health_consequence', 'phys_health_conseq
    'coworkers', 'supervisor', 'mental_health_interview', 'phys_health_int
    'mental_vs_physical', 'obs_consequence']:
        # Create a bar chart for selected columns
        column_counts = df[column].value_counts()
        plt.figure(figsize=(20, 10))
        ax = column_counts.plot(kind='bar', color='orange')
        plt.xlabel(column)
        plt.ylabel('Count')
        plt.title(f'{column} Distribution')
        plt.xticks(rotation=45)

        # Add percentage labels to the bars
        total_count = len(df[column])
        for p in ax.patches:
            percentage = f"{100 * p.get_height() / total_count:.2f}%"
            ax.annotate(percentage, (p.get_x() + p.get_width() / 2, p.get_

        plt.show()
```

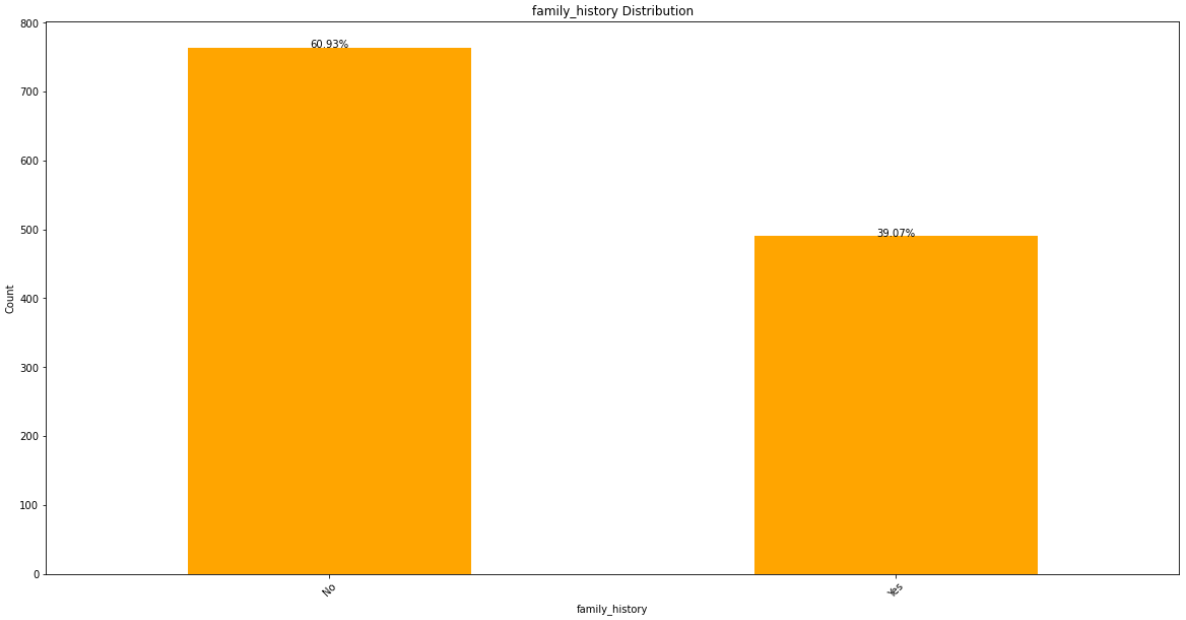


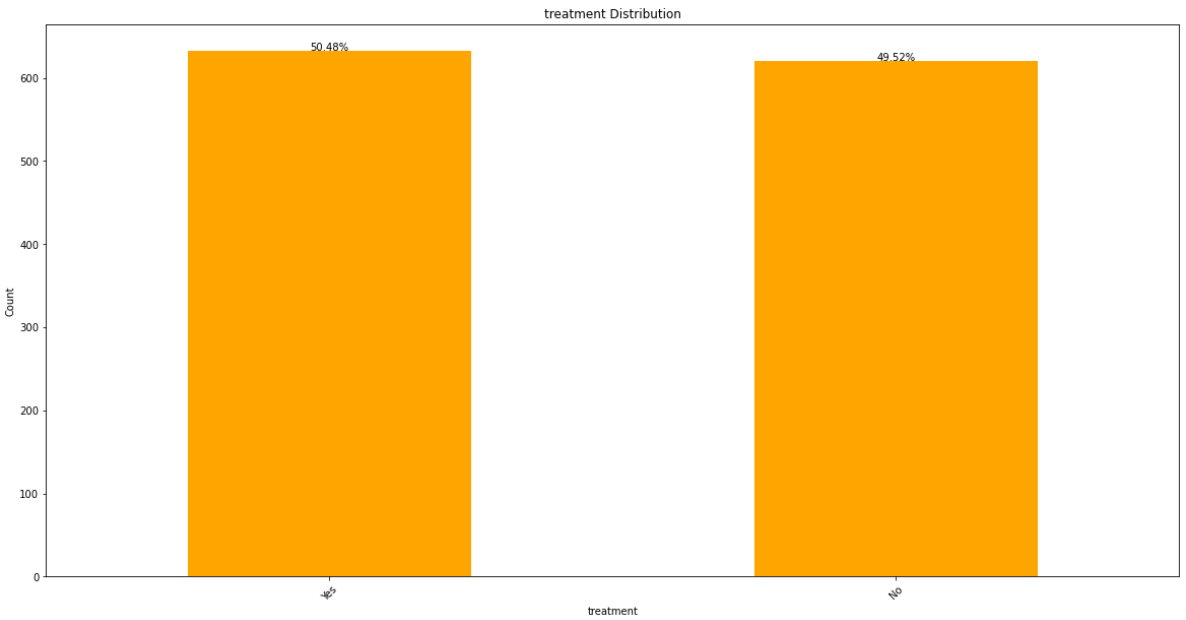

Gender Distribution
Count
Gender
Male
Female
Other
Cis Male
Trans Female
Non-Binary
Cis Female
Androgyne
Agender
Cis Female/Femme
Queer/She/They
Fluid
Neuter
Queer
Genderqueer
78.23%
19.46%
0.56%
0.40%
0.32%
0.16%
0.16%
0.16%
0.08%
0.08%
0.08%
0.08%
0.08%
0.08%
0.08%

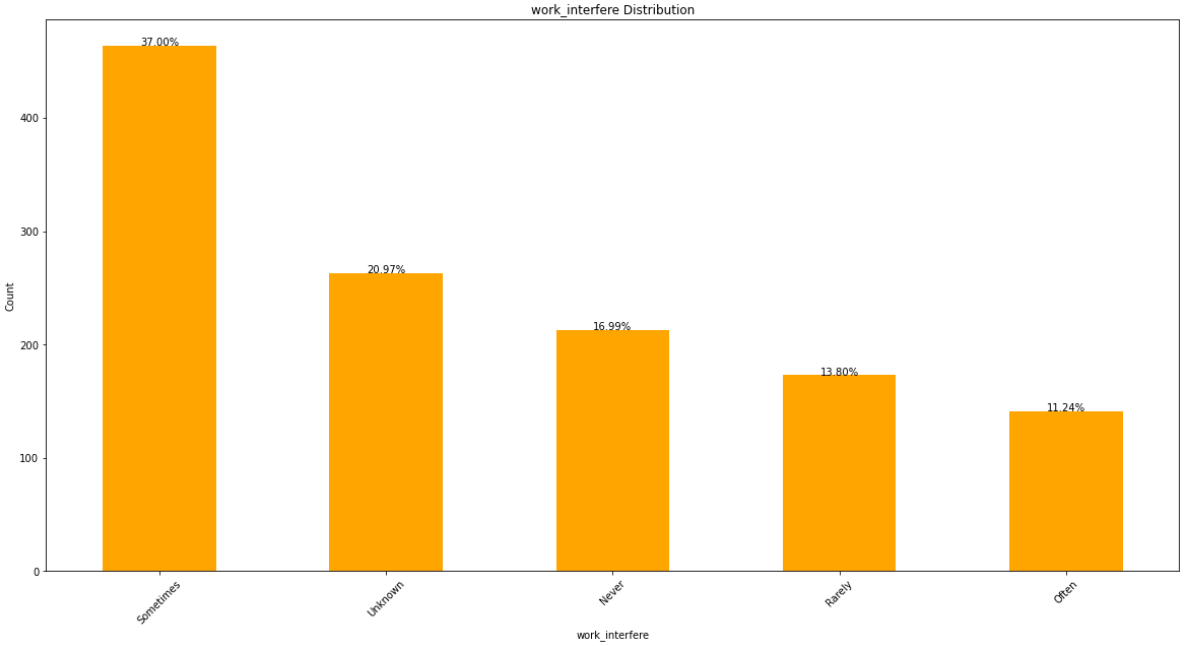

Country Distribution
Count
Country
United States
United Kingdom
Canada
Germany
Ireland
Netherlands
Australia
France
India
New Zealand
Poland
Sweden
Italy
Switzerland
Belgium
Brazil
South Africa
Israel
Bulgaria
Singapore
Austria
Finland
Mexico
Russia
Denmark
Colombia
Portugal
Croatia
Greece
Hungary
China
Nigeria
Latvia
Spain
Moldova
Norway
Japan
Uruguay
Thailand
Georgia
Bahamas, The
Czech Republic
Bosnia and Herzegovina
Slovenia
Romania
Costa Rica
Philippines
59.65%
14.67%
5.74%
3.59%

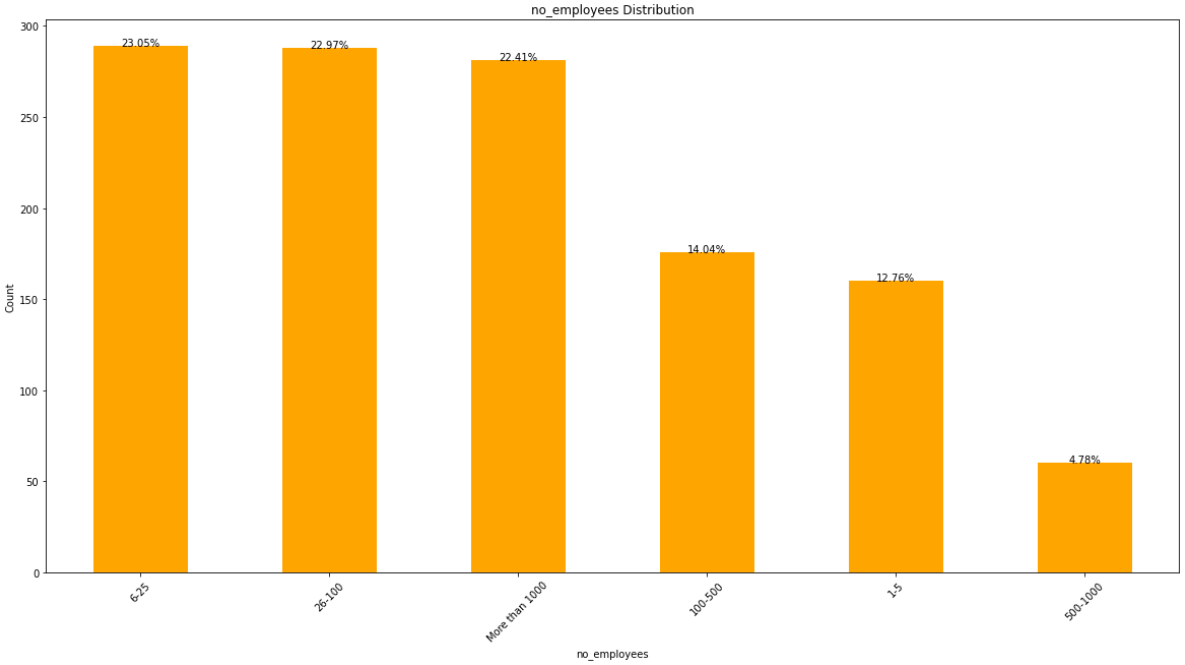

state Distribution
Count
state
Unknown
CA
WA
NY
TN
TX
IL
OR
OH
PA
IN
MI
MN
MA
FL
NC
VA
WI
MO
GA
UT
CO
MD
AZ
AL
OK
NJ
SC
KY
DC
CT
IA
NH
SD
KS
NV
VT
WY
NM
NE
ME
LA
ID
MS
WV
RI
40.91%
11.00%
5.58%
4.55%
3.59%
3.51%

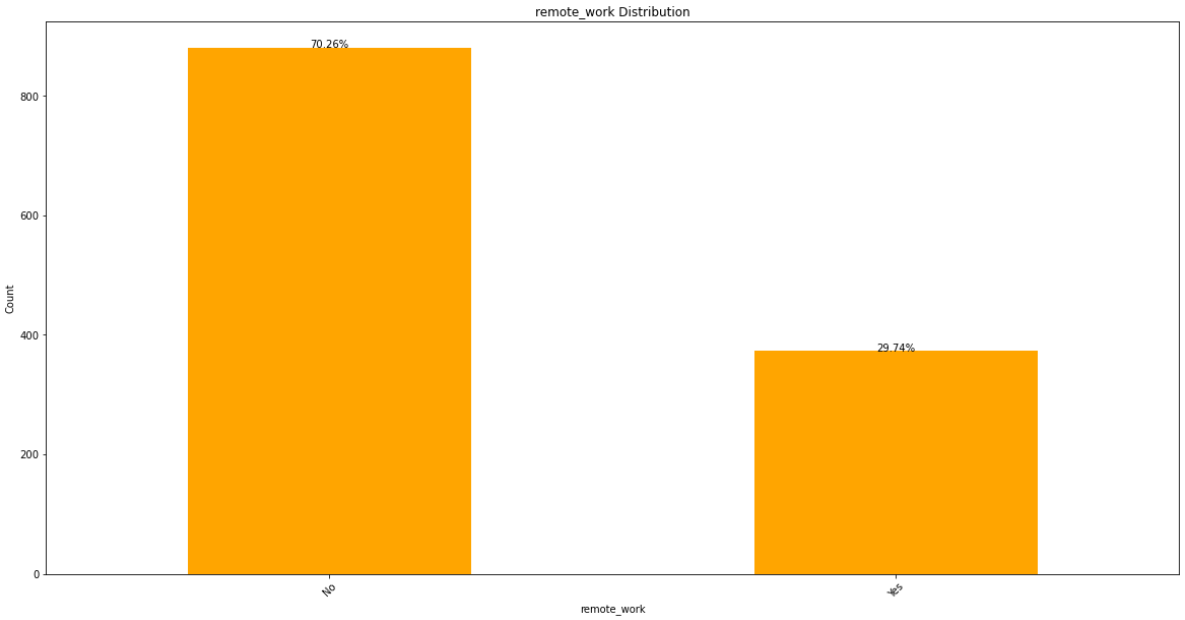

self_employed Distribution
87.08%
11.48%
1.44%
Count
0
200
400
600
800
1000
No
Yes
Unknown
self_employed

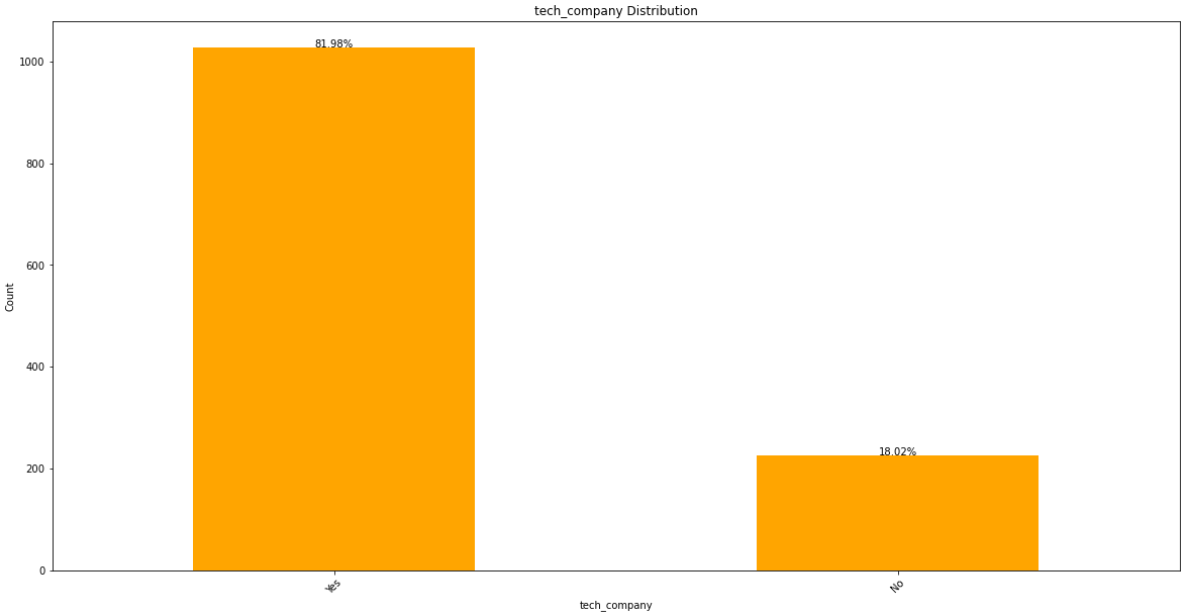

family_history Distribution
60.93%
39.07%
Count
0
100
200
300
400
500
600
700
800
No
Yes
family_history

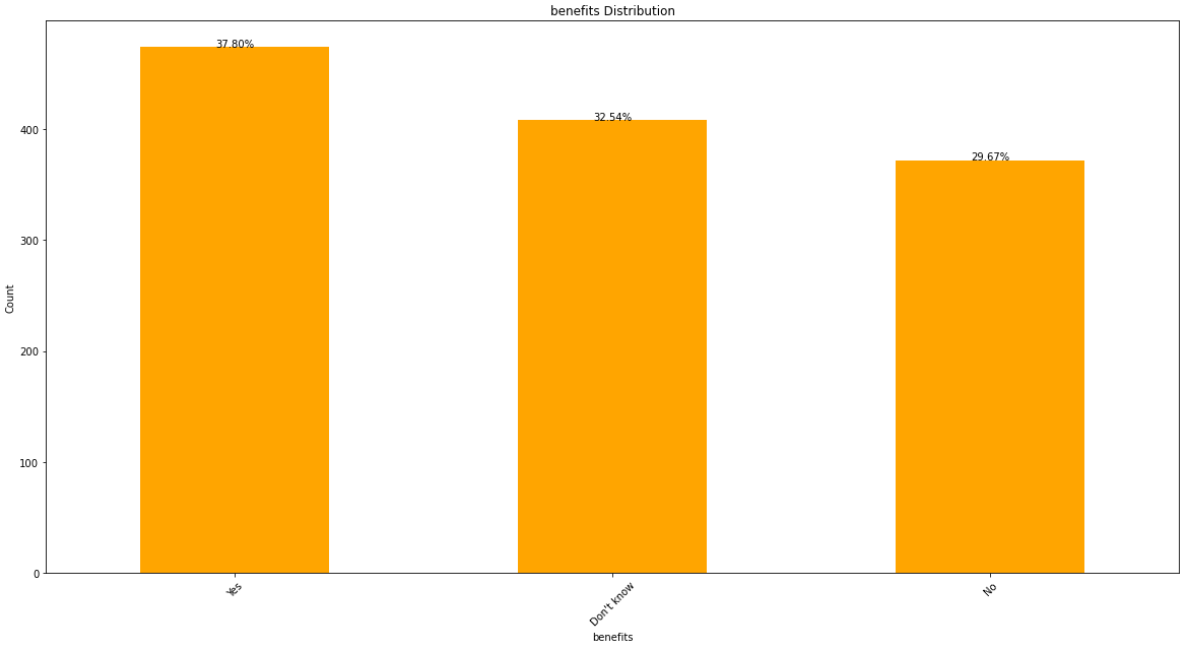

treatment Distribution
50.48%
49.52%
Count
0
100
200
300
400
500
600
Yes
No
treatment

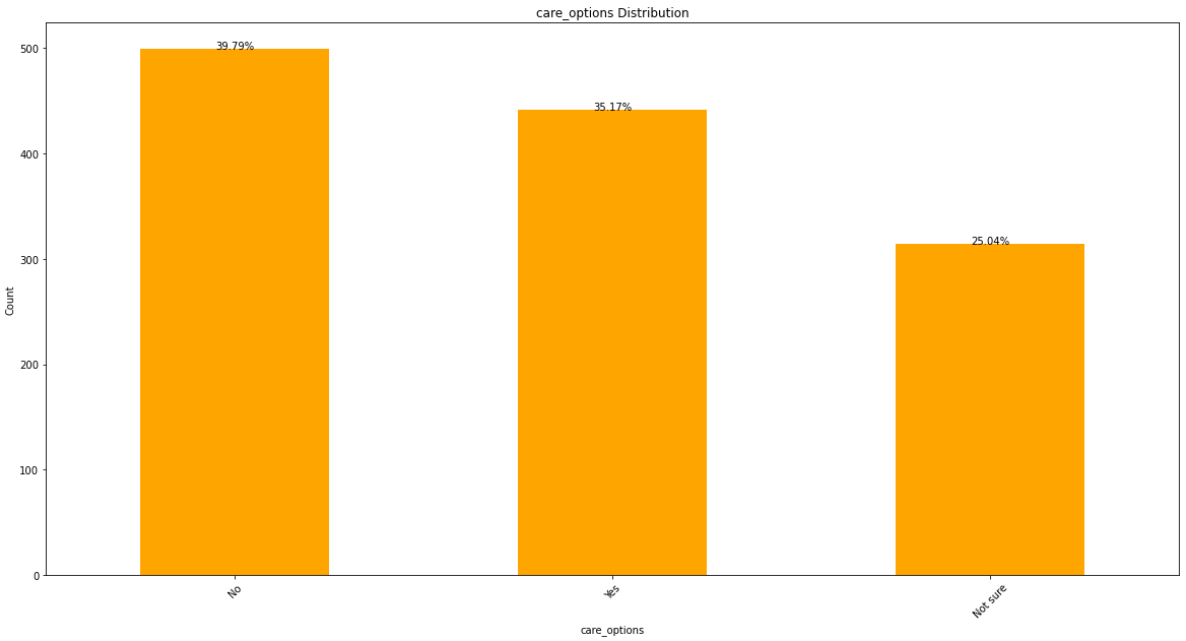

work_interfere Distribution
37.00%
20.97%
16.99%
13.80%
11.24%
Count
0
100
200
300
400
Sometimes
Unknown
Never
Rarely
Often
work_interfere

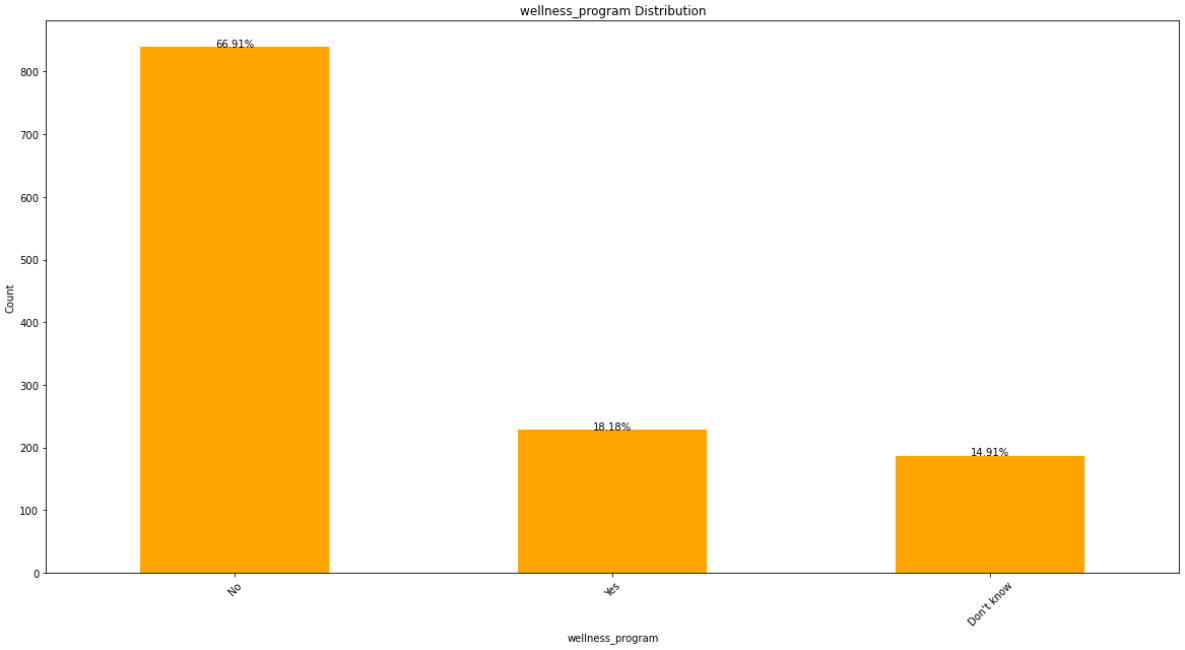

no_employees Distribution
23.05%
22.97%
22.41%
14.04%
12.76%
4.78%
Count
0
50
100
150
200
250
300
6-25
26-100
More than 1000
100-500
1-5
500-1000
no_employees

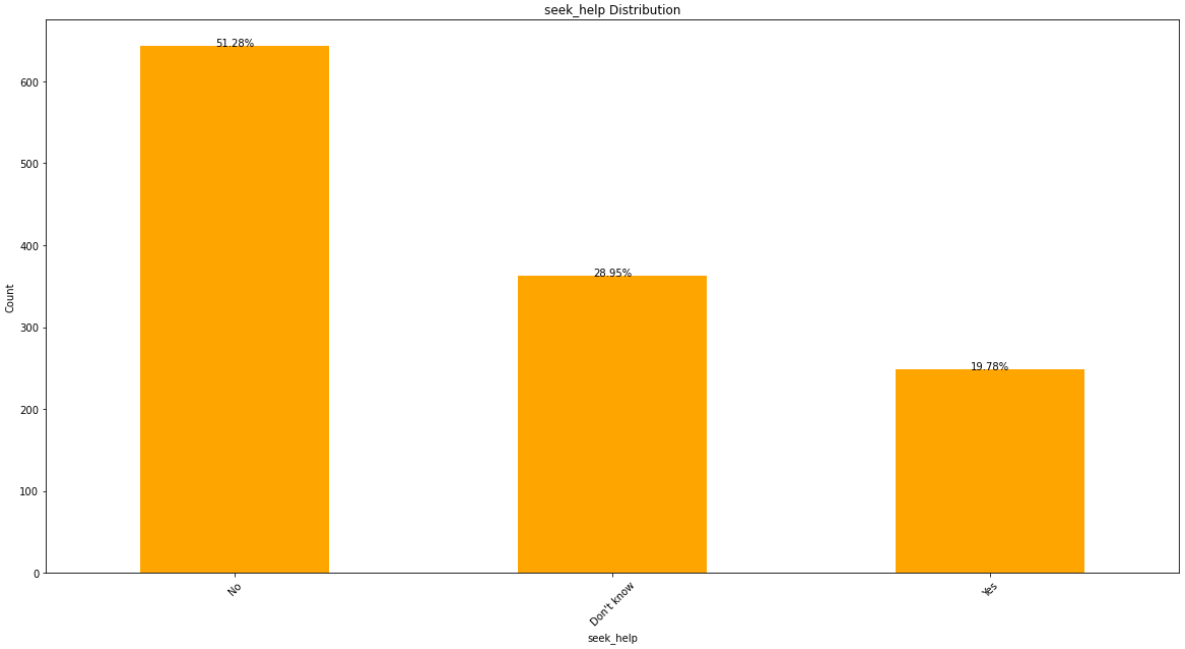

remote_work Distribution
70.26%
29.74%
Count
0
200
400
600
800
No
Yes
remote_work

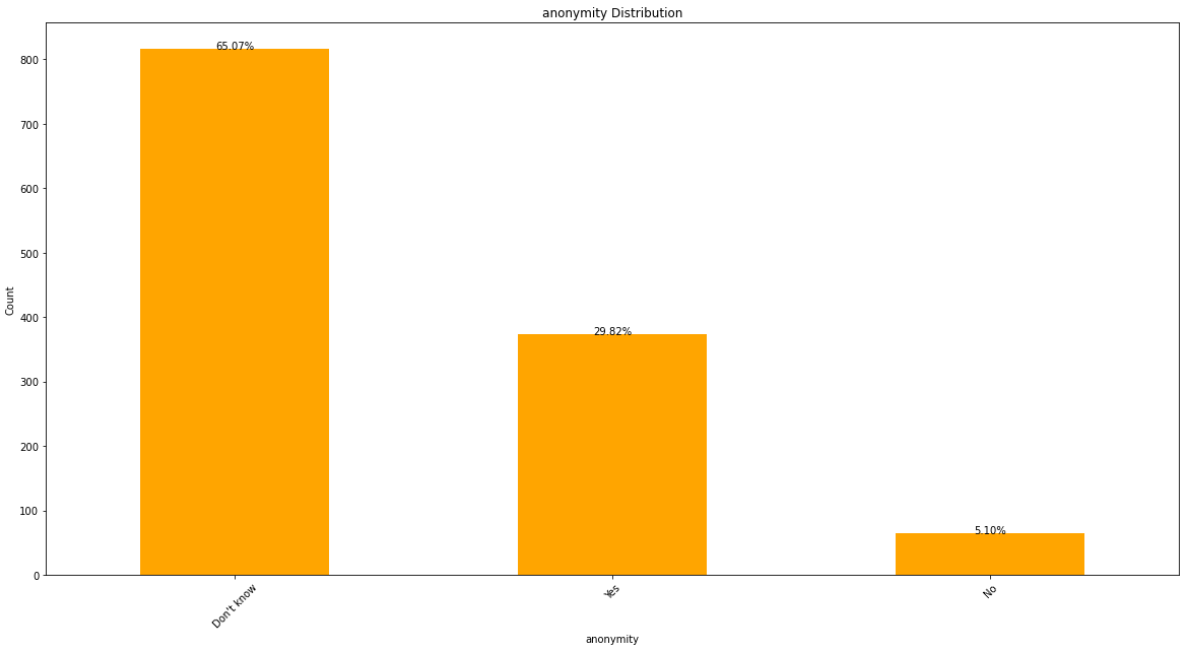

tech_company Distribution
81.98%
18.02%
Count
tech_company
Yes
No

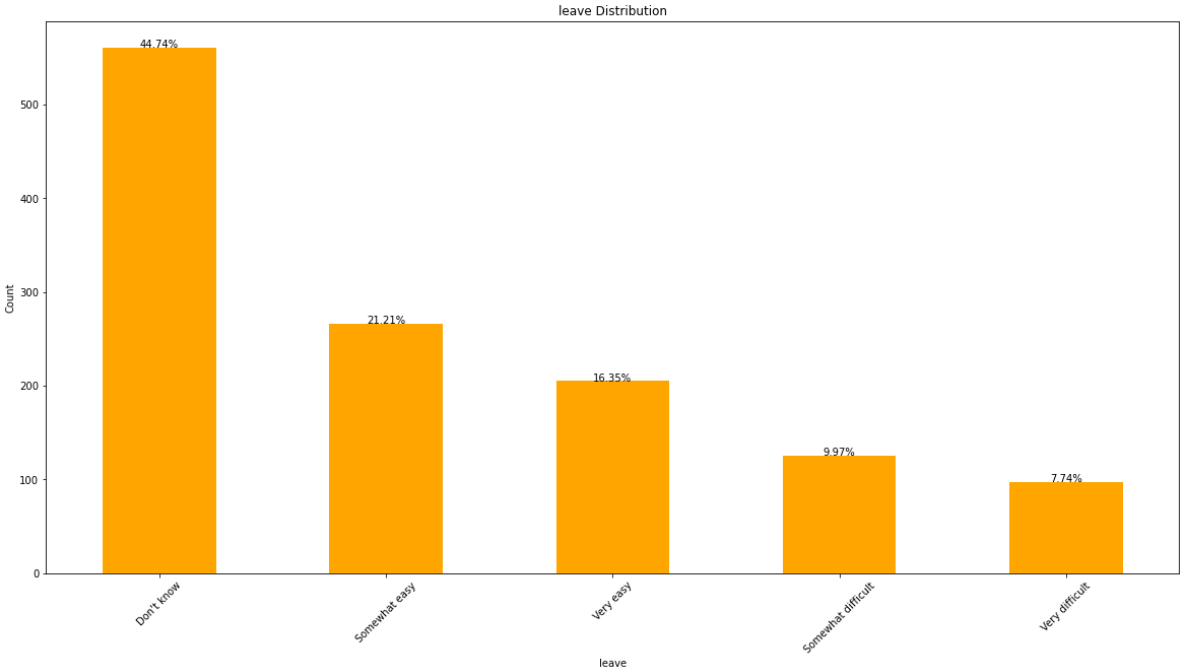

benefits Distribution
37.80%
32.54%
29.67%
Count
benefits
Yes
Don't know
No

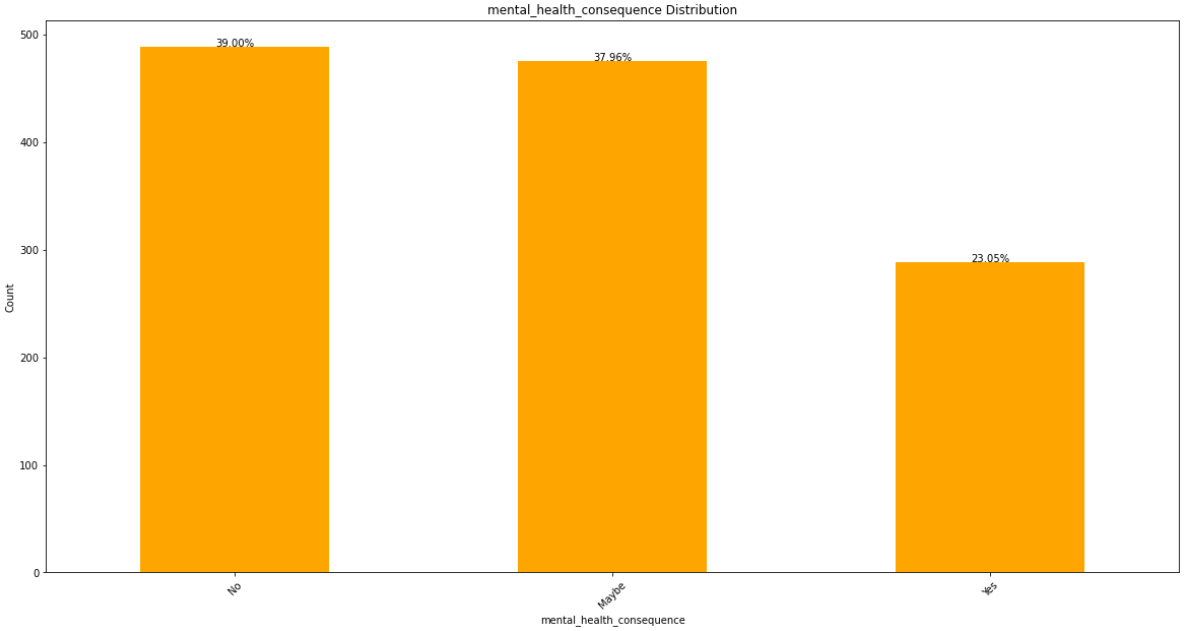

care_options Distribution
39.79%
35.17%
25.04%
Count
care_options
No
Yes
Not sure

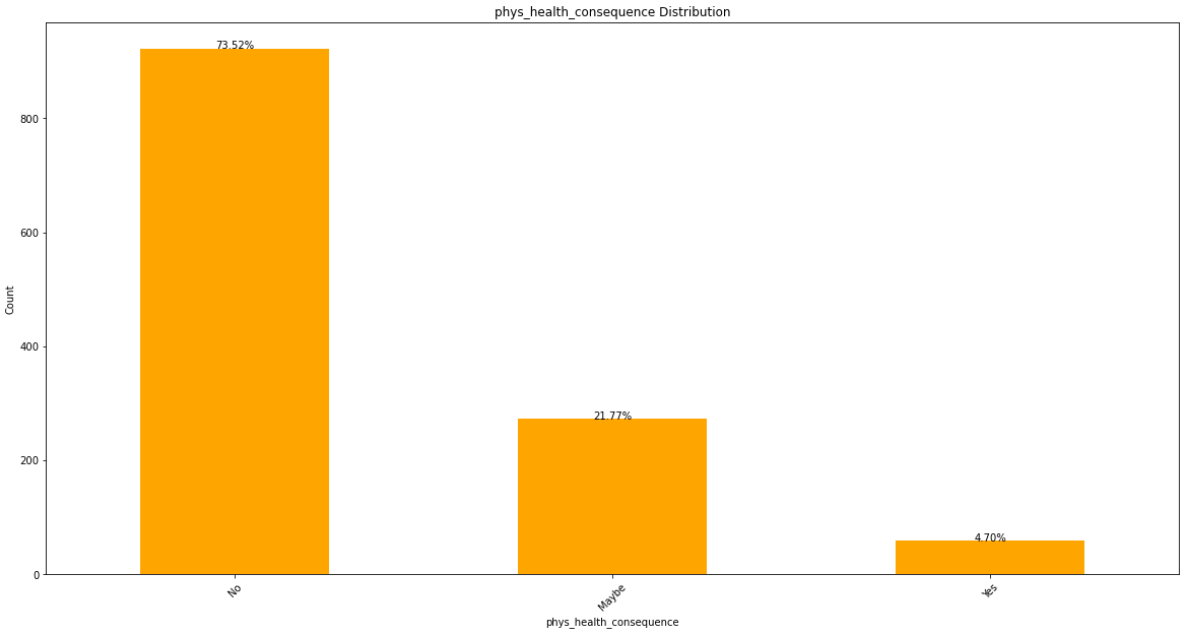

wellness_program Distribution
66.91%
18.18%
14.91%
Count
0
100
200
300
400
500
600
700
800
No
Yes
Don't know
wellness_program

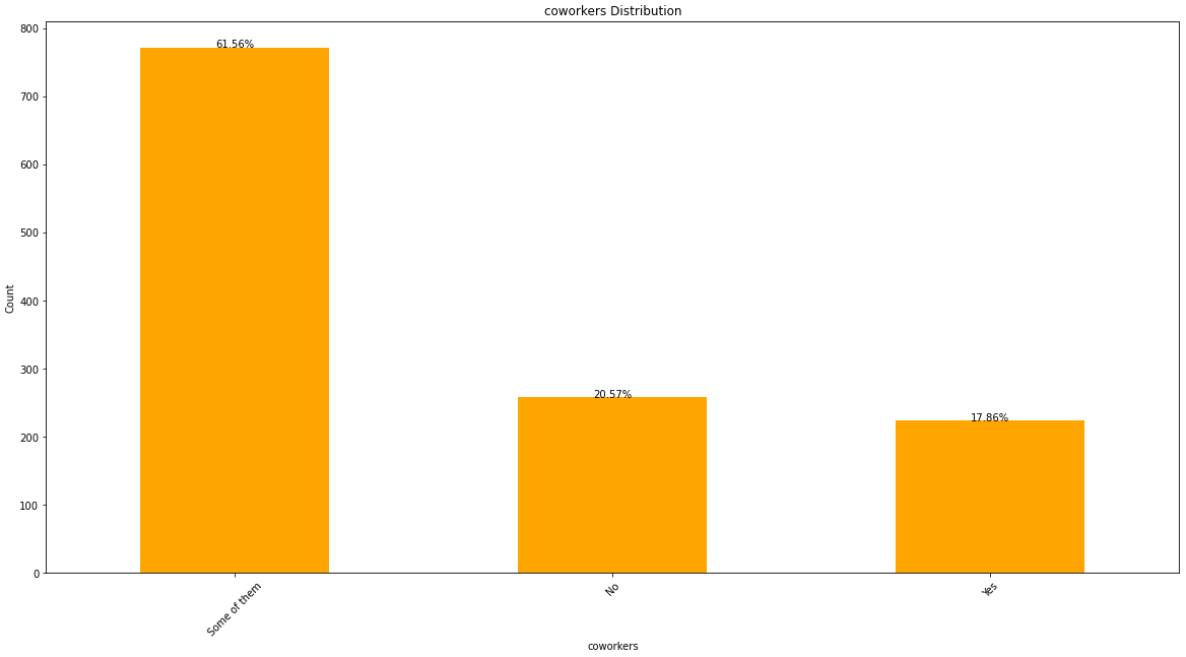

seek_help Distribution
51.28%
28.95%
19.78%
Count
0
100
200
300
400
500
600
No
Don't know
Yes
seek_help

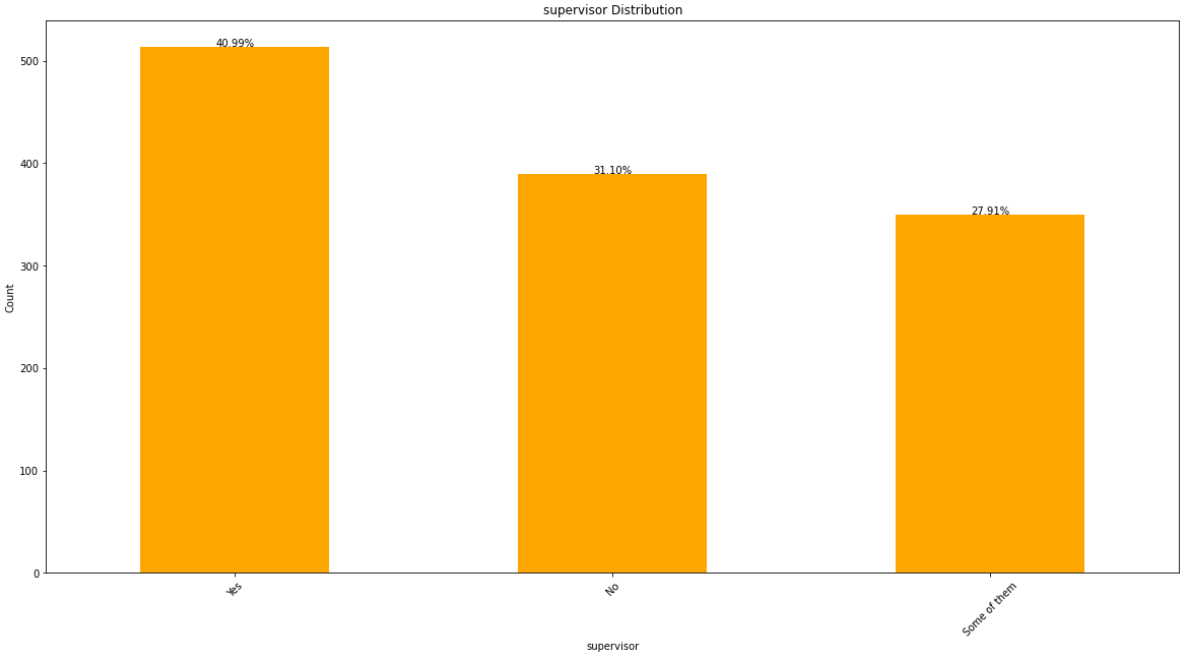

anonymity Distribution
65.07%
29.82%
5.10%
Count
0
100
200
300
400
500
600
700
800
Don't know
Yes
No
anonymity

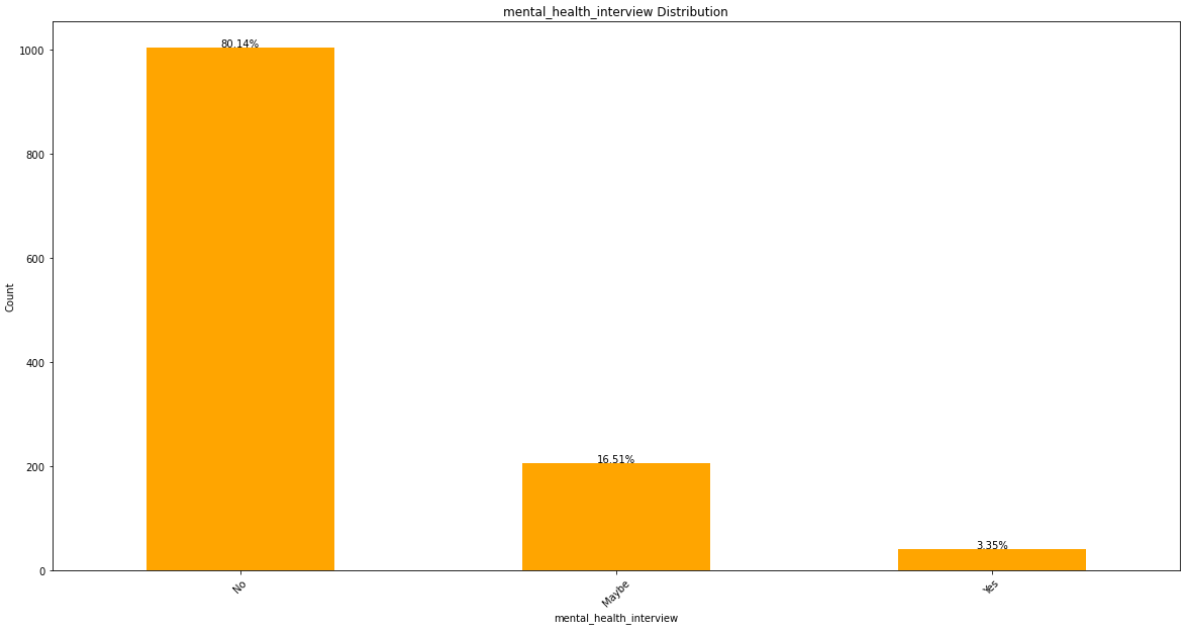

leave Distribution
Count
leave
Don't know
Somewhat easy
Very easy
Somewhat difficult
Very difficult
44.74%
21.21%
16.35%
9.97%
7.74%

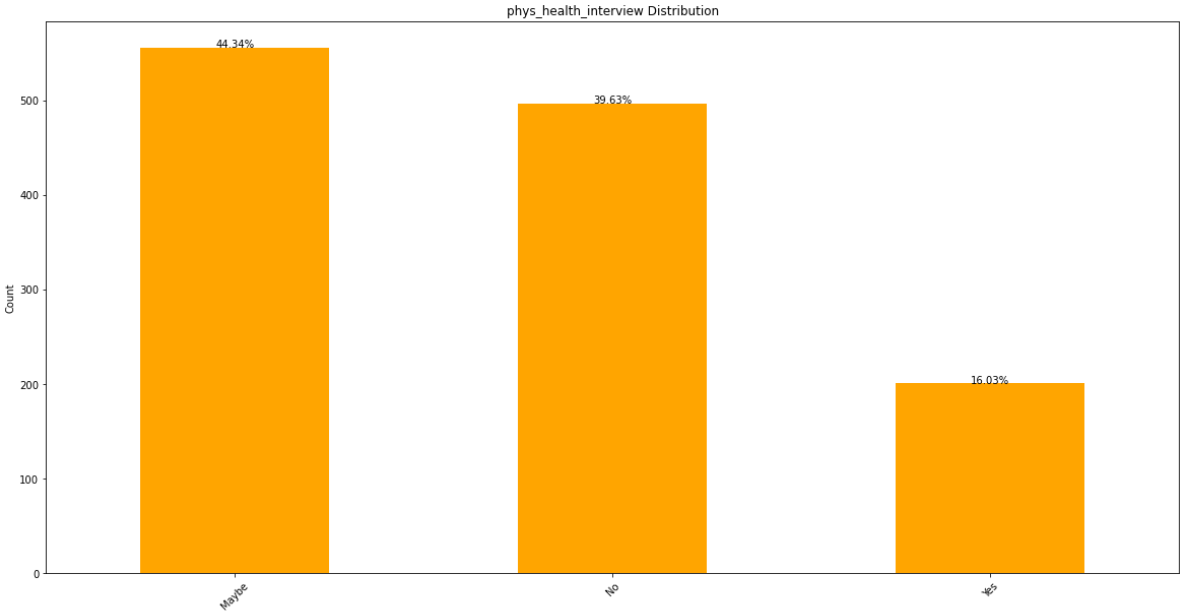

mental_health_consequence Distribution
Count
mental_health_consequence
No
Maybe
Yes
39.00%
37.96%
23.05%

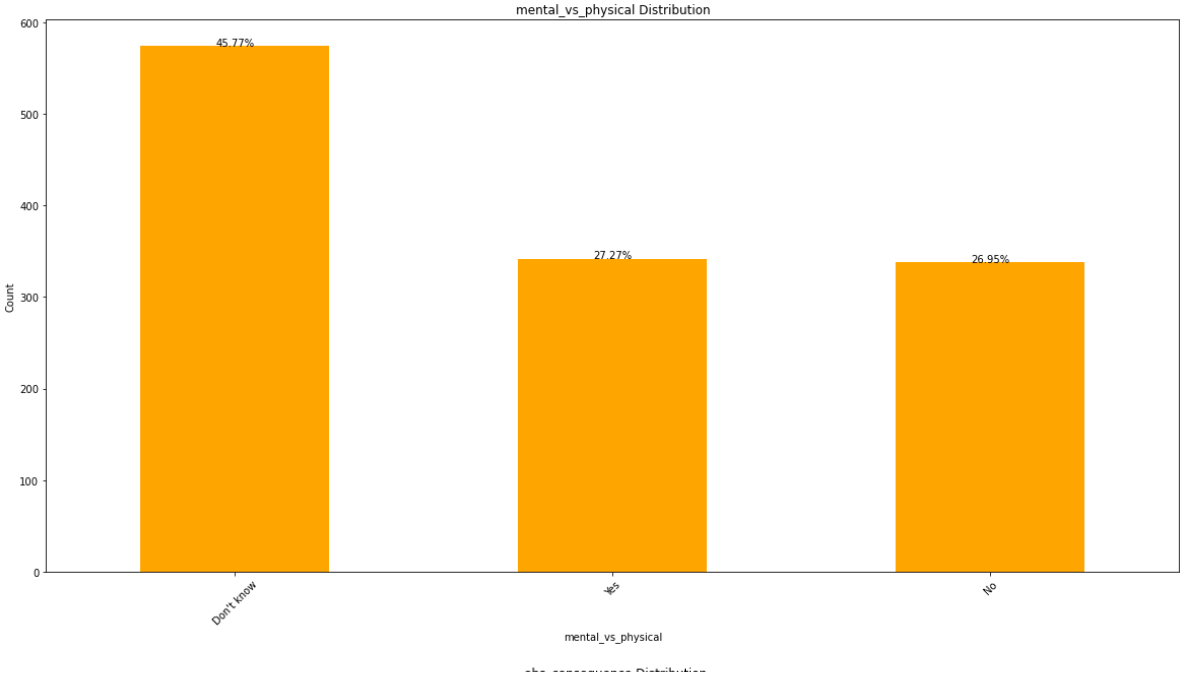

phys_health_consequence Distribution
Count
phys_health_consequence
No
Maybe
Yes
73.52%
21.77%
4.70%


coworkers Distribution
Count
61.56%
20.57%
17.86%
Some of them
No
Yes
coworkers


supervisor Distribution
Count
40.99%
31.10%
27.91%
Yes
No
Some of them
supervisor


mental_health_interview Distribution
Count
80.14%
16.51%
3.35%
No
Maybe
Yes
mental_health_interview

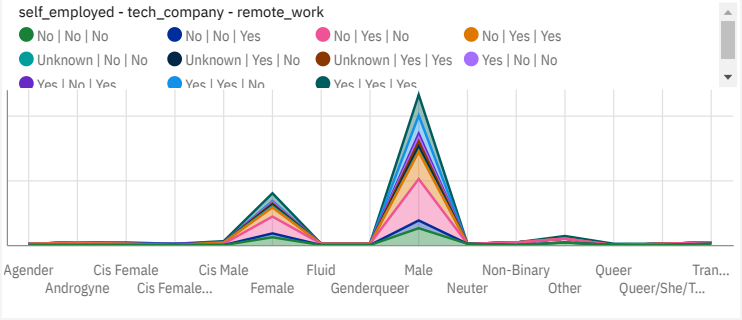

phys_health_interview Distribution
44.34%
39.63%
16.03%
Count
0
100
200
300
400
500
Maybe
No
Yes
phys_health_interview


mental_vs_physical Distribution
45.77%
27.27%
26.95%
Count
0
100
200
300
400
500
600
Don't know
Yes
No
mental_vs_physical

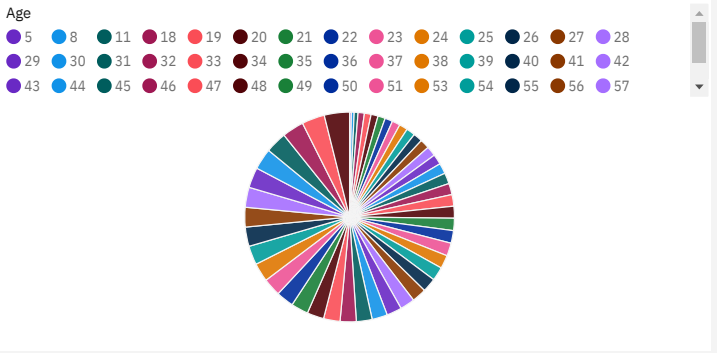
In [ ]:

Tab 1

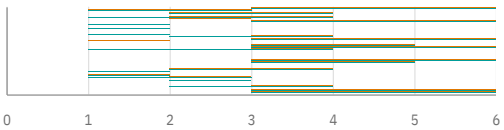
Country by Gender colored by self_employed, tech_company and remote_work



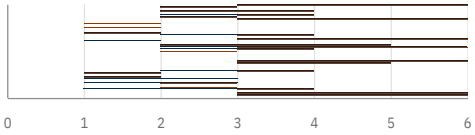
Age by Age



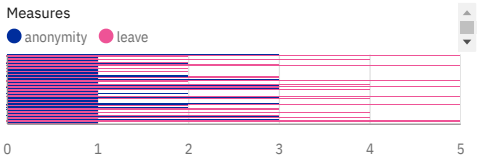
mental_health_consequence and phys_health_consequence by Country



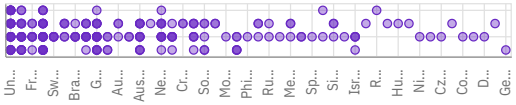
mental_health_interview and phys_health_interview by Country



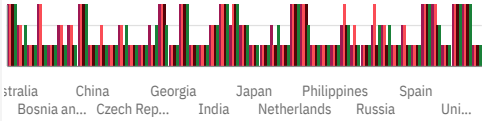
anonymity and leave by Country



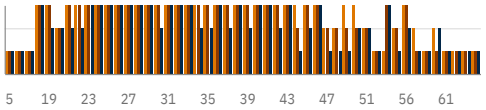
Country by tech_companyremote_work with points for no_employees



benefits, care_options, wellness_program and seek_help by Country

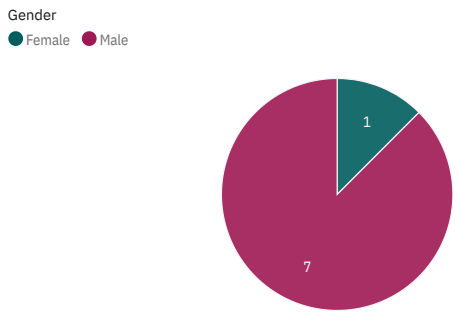


mental_health_consequence, phys_health_interview and mental_health_interview by Age

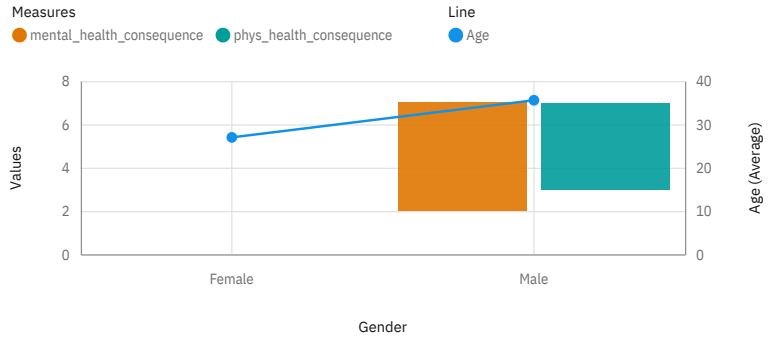


Tab 2

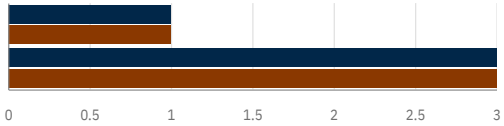
Gender by Gender



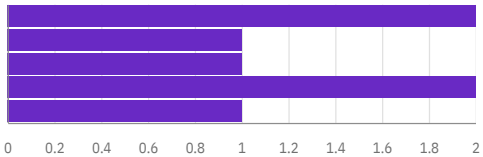
Age and mental_health_consequence and phys_health_consequence by Gender



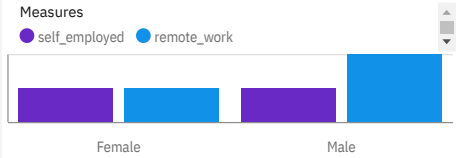
mental_health_interview and phys_health_interview by Gender



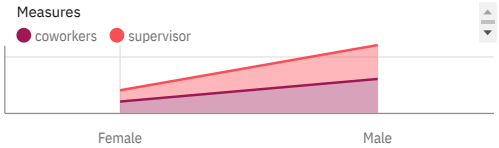
no_employees by Age



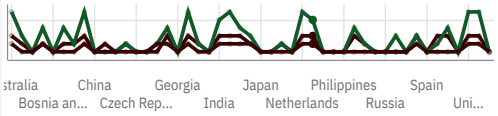
self_employed and remote_work by Gender



coworkers and supervisor by Gender



tech_company, no_employees and mental_vs_physical by Country



Gender, Age and mental_vs_physical by Country