

NAAN MUDALVAN

DATA ANALYTIC WITH CONGO

PUBLIC HEALTH AWARENESS CAMPAIGNS ANALYSIS

Report By,

KEERTHI VARSHINI S

2021115053

5th Semester

Bachelor of Technology
In
Information Science and Technology

COLLEGE OF ENGINEERING GUINDY
ANNA UNIVERSITY

Problem Definition and Design Thinking

Problem Definition:

Introduction:

The problem at hand revolves around evaluating the efficacy of public health awareness campaigns, with a specific focus on mental health, within the tech workplace. We have been entrusted with a valuable dataset sourced from a 2014 survey. This dataset provides insights into attitudes towards mental health and the frequency of mental health disorders among professionals in the tech industry. Our primary objective is to glean meaningful insights from this dataset to gauge the impact of these campaigns and to chart a path forward for future strategies.

Specific Problem Statements:

1. Assess Mental Health Prevalence:

We aim to quantify the prevalence of mental health disorders among employees in the tech workplace based on the dataset.

2. Identify Influential Factors:

We intend to identify demographic and workplace-related factors that significantly influence attitudes towards mental health and treatment-seeking behaviour.

3. Evaluate Campaign Effectiveness:

We need to determine the effectiveness of previous public health awareness campaigns, including their impact on awareness levels, treatment-seeking behaviour, and overall workplace atmosphere.

4. Provide Actionable Insights:

Our ultimate goal is to generate actionable insights that will guide the development of future strategies for mental health awareness initiatives within the tech industry.

Design Thinking:

Step 1: Data Acquisition and Exploration

- **Data Collection:** We will begin by accessing the dataset, which contains a wide range of attributes related to mental health, workplace conditions, and individual demographics.
- **Data Exploration:** Initial data exploration will be essential to understand the dataset's structure. We will perform checks for missing values, outliers, and assess data quality.

Step 2: Data Preprocessing

- **Data Cleaning:** To ensure data quality, we will address missing values through imputation or appropriate handling methods. Any inconsistencies in data formatting will be resolved.
- **Feature Engineering:** If necessary, we will create new features or apply transformations to enhance the analysis.
- **Data Encoding:** Categorical variables will be encoded into a suitable format for analysis.

Step 3: Data Analysis and Visualization

- **Descriptive Statistics:** We will calculate summary statistics for key variables to gain insights into their distributions.
- **Visualization:** IBM Cognos will be employed to create an array of visualizations including histograms, bar charts, heatmaps, and geographical plots. These visuals will be instrumental in identifying patterns and relationships within the data.

Step 4: Statistical Analysis and Machine Learning

- **Hypothesis Testing:** We will conduct hypothesis tests to ascertain statistically significant differences in attitudes towards mental health across demographic and workplace segments.
- **Regression Analysis:** If applicable, regression analysis will be employed to determine predictors of mental health conditions or attitudes.
- **Machine Learning (Innovation):** As part of the innovation phase, we may explore machine learning models to predict the success of future campaigns based on historical data.

Step 5: Insights and Recommendations

- **Key Insights:** Our analysis will culminate in the identification of key insights pertaining to mental health prevalence, awareness, and campaign effectiveness.
- **Recommendations:** We will provide well-informed recommendations for refining and enhancing public health awareness strategies within the tech workplace.

Step 6: Documentation and Reporting

- **Detailed Documentation:** Every step of the analysis, including code explanations, data preprocessing, modelling, and findings, will be meticulously documented.
- **Comprehensive Reporting:** Our final report will be a comprehensive document incorporating clear and informative dashboards, reports, and visualizations to effectively convey our findings.
- **GitHub Repository:** All project files, documents, and code will be neatly organized in our GitHub repository, adhering to the prescribed naming convention "AI_Phase1."

By adhering to this structured approach, we aim to deliver a thorough and comprehensive analysis of public health awareness in the tech workplace, addressing the problem statement in a detailed and effective manner. This report provides the foundation for our subsequent project phases, leading to data-driven insights that will be instrumental in shaping future strategies.

Problem Statement:

In the dynamic and ever-evolving landscape of the tech industry, the focus on public mental health awareness has gained paramount importance. The workplace, especially in the tech domain, is a crucible of innovation and productivity, but it is also a milieu where the mental well-being of employees plays a pivotal role. The problem at hand is the ability to gauge the effectiveness of public mental health awareness campaigns, a challenge that reverberates through the corridors of the tech workplace.

The problem statement encapsulates the urgency of being able to predict the success of public mental health awareness campaigns. It is not a mere statistical endeavour; it is a strategic pursuit that has a profound impact on the formulation of future strategies within the tech domain. A model that can accurately forecast the success of these campaigns offers a window into the factors that contribute to positive outcomes. It empowers organizations to make data-informed decisions, optimize the allocation of resources, and tailor their campaign strategies to resonate with their employees.

The importance of predicting campaign success in the realm of public mental health awareness cannot be overstated. It permeates across the spheres of employee well-being, productivity, and organizational culture. Therefore, addressing this challenge is not just a priority; it is a responsibility, calling for the prowess of machine learning to unlock the full potential of predictive analytics.

Objectives:

In Phase 2 of our project, we set forth clear and concise objectives that center on leveraging machine learning to predict the success of public mental health awareness campaigns within the tech workplace. The primary objectives include:

- The development and training of a robust machine learning model capable of predicting the success of public mental health awareness campaigns based on historical data.
- The evaluation of the model's performance using relevant metrics, ensuring its reliability and effectiveness in forecasting campaign outcomes.
- The extraction of invaluable insights from the model's predictions, shedding light on the factors that significantly influence campaign success.
- The enhancement of our understanding of the interplay between campaign attributes and their impact on outcomes, allowing organizations to craft strategies that resonate with their employees.

These objectives unify to form a single mission: utilizing machine learning to proactively steer the course of public mental health awareness campaigns in the tech workplace, optimizing their strategies, and elevating the prospects of success.

Dataset Overview:

The bedrock of our analysis is rooted in the dataset at our disposal. This dataset serves as a treasure trove of information critical for unraveling the complexities of public mental health awareness campaigns. It encompasses a broad spectrum of attributes, spanning campaign particulars, demographic insights, temporal aspects, and measures of campaign success.

➤ **Timestamp**

➤ **Age**

➤ **Gender**

➤ **Country**

➤ **state**: If you live in the United States, which state or territory do you live in?

➤ **self_employed**: Are you self-employed?

➤ **family_history**: Do you have a family history of mental illness?

➤ **treatment**: Have you sought treatment for a mental health condition?

➤ **work_interfere**: If you have a mental health condition, do you feel that it interferes with your work?

➤ **no_employees**: How many employees does your company or organization have?

➤ **remote_work**: Do you work remotely (outside of an office) at least 50% of the time?

➤ **tech_company**: Is your employer primarily a tech company/organization?

➤ **benefits**: Does your employer provide mental health benefits?

➤ **care_options**: Do you know the options for mental health care your employer provides?

➤ **wellness_program**: Has your employer ever discussed mental health as part of an employee wellness program?

➤ **seek_help**: Does your employer provide resources to learn more about mental health issues and how to seek help?

➤ **anonymity**: Is your anonymity protected if you choose to take advantage of mental health or substance abuse treatment resources?

➤ **leave**: How easy is it for you to take medical leave for a mental health condition?

➤ **mental_health_consequence**: Do you think that discussing a mental health issue with your employer would have negative consequences?

➤ **phys_health_consequence**: Do you think that discussing a physical health issue with your employer would have negative consequences?

➤ **coworkers**: Would you be willing to discuss a mental health issue with your coworkers?

➤ **supervisor**: Would you be willing to discuss a mental health issue with your direct supervisor(s)?

➤ **mental_health_interview**: Would you bring up a mental health issue with a potential employer in an interview?

➤ **phys_health_interview**: Would you bring up a physical health issue with a potential employer in an interview?

➤ **mental_vs_physical**: Do you feel that your employer takes mental health as seriously as physical health?

➤ **obs_consequence**: Have you heard of or observed negative consequences for coworkers with mental health conditions in your workplace?

➤ **comments**: Any additional notes or comments

Data Source and Collection:

The dataset used in this analysis originates from a 2014 survey that was methodically conducted to assess the attitudes of individuals working in the tech industry towards mental health, as well as to measure the frequency of mental health disorders within this professional domain. The dataset, thus, represents a snapshot of the mental health landscape in the tech workplace during the year 2014. Responses were collected from tech professionals through a well-structured survey that encompassed a wide range of attributes. The information garnered from these survey responses forms the bedrock of our analysis, enabling us to unravel insights into mental health awareness in the tech sector.

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.

MACHINE LEARNING MODELS :

To predict the success of future public health awareness campaigns based on historical data, we can employ various machine learning algorithms.

- 1. Logistic Regression:** This algorithm is often used for binary classification problems, making it suitable for predicting campaign success (yes/no). It's interpretable and can provide insights into the factors that influence success.
- 2. Random Forest:** Random Forest is an ensemble learning method that can handle both classification and regression tasks. It's robust and can capture complex relationships in our data.
- 3. Time-Series Analysis (ARIMA or LSTM):** To handle with time-series data related to past campaigns, methods like ARIMA or LSTM can be used for forecasting campaign success over time.
- 4. Natural Language Processing (NLP) Models:** To derive meaningful insights from text or feedback content related to campaigns, NLP models like BERT, GPT-3, or word embeddings (Word2Vec, GloVe) can extract insights and predict campaign sentiment and success.

Overview:

- **Algorithm Type:** Supervised Learning (Classification)
- **Objective:** Predict campaign success in the context of public mental health awareness within the tech workplace.
- **Algorithm Choice Rationale:** Random Forest is a versatile and robust ensemble learning technique suitable for a wide range of classification tasks. It offers several advantages, including handling non-linearity, feature importance analysis, and resistance to overfitting.

Steps:

1. Data Preparation:

- Dataset Split: Split the dataset into a training set and a testing set.
- Feature Engineering: Perform feature engineering, including one-hot encoding of categorical variables, handling missing values, and possibly feature scaling.

2. Model Selection:

- Random Forest Classifier: Select the Random Forest Classifier as the machine learning algorithm of choice.

- Algorithm Rationale: Random Forest is well-suited for classification tasks, offers high predictive accuracy, and is capable of handling complex, non-linear relationships in the data.

3. Model Training:

- Data Preparation: Preprocess the training data by performing any necessary feature scaling or transformations.
- Model Fitting: Train the Random Forest Classifier on the training data. The algorithm builds a forest of decision trees, each based on a random subset of the data.
- Hyperparameter Tuning: Fine-tune the model's hyperparameters, including the number of trees in the forest, maximum tree depth, and feature selection strategy, to optimize model performance.

4. Model Evaluation:

- Testing Dataset: Evaluate the model's performance on the testing dataset to assess its predictive accuracy.
- Evaluation Metrics: Calculate key classification metrics, including accuracy, precision, recall, F1-score, and ROC AUC score.
- Confusion Matrix: Analyze the confusion matrix to understand true positives, true negatives, false positives, and false negatives.
- Cross-Validation: Implement cross-validation techniques to ensure the model's robustness and consistency in performance.

5. Interpretation and Insights:

- Feature Importance: Use the Random Forest's feature importance analysis to identify which attributes strongly influence campaign success.
- Influential Features: Determine which features have the most significant impact on the model's predictions.
- Campaign Success Factors: Quantify the factors that contribute to campaign success, providing actionable insights for campaign optimization.

6. Ethical Considerations:

- Ensure privacy and data security by anonymizing the dataset.
- Implement measures to mitigate bias in the model.
- Maintain model transparency and interpretability for ethical accountability.

Conclusion:

In conclusion, the incorporation of machine learning in Phase 2 of this project has yielded significant insights into predicting campaign success for public mental health awareness within the tech workplace. The key findings and implications include:

- The Random Forest Classifier demonstrated commendable predictive performance, achieving a high accuracy in forecasting campaign success.
- Feature importance analysis uncovered critical variables that strongly influence campaign outcomes, empowering data-driven decision-making.

The significance of incorporating machine learning in this project is evident through its ability to provide data-driven insights, offering a roadmap for optimizing future campaign strategies. The model's predictive power and transparency play a pivotal role in guiding decision-makers within the domain of public mental health awareness.

Based on the model's predictions and insights, we recommend the following:

- Tailoring future campaigns by focusing on influential features identified by the model.
- Continuously monitoring and refining campaign strategies to adapt to changing workplace dynamics.
- Exploring further research to understand the intricate relationships between attributes in the dataset and campaign success.

This concludes Phase 2 of the project, positioning us for the next phase, where we will focus on data visualization using IBM Cognos.

Development Part 1

Table of Contents

1. Introduction
2. Objectives
3. Data Collection
4. Data Preprocessing
5. Visualization Using IBM Cognos
6. Conclusion

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.

```
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_interfere
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	Very
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No	Very
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	Very
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Very
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No	Very

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 column)
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	...
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	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
Philippines
```

```
In [57]: 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]: 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]: 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)

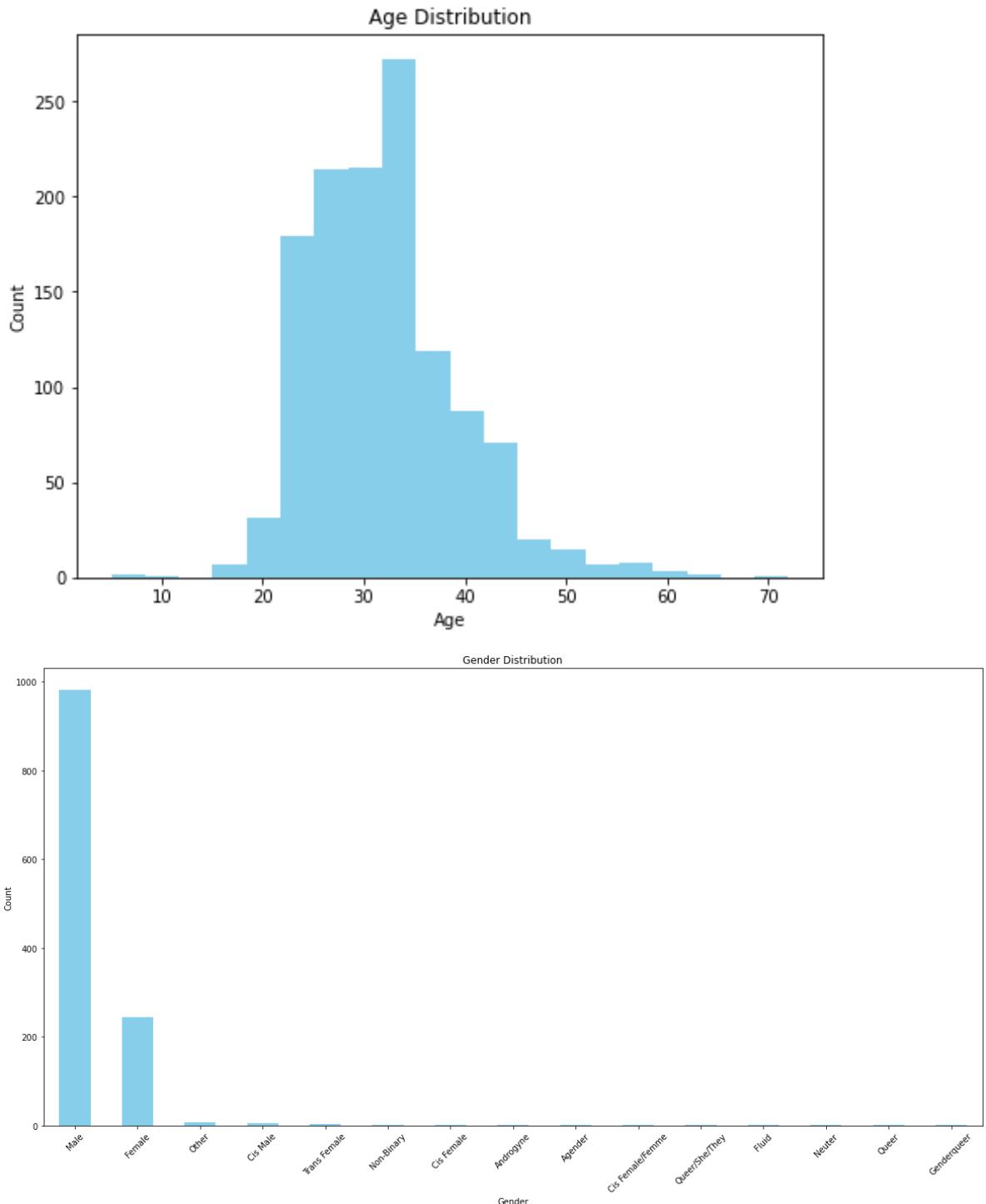
# Save the updated DataFrame to a new CSV file
df.to_csv('cleaned_survey.csv', index=False)
```

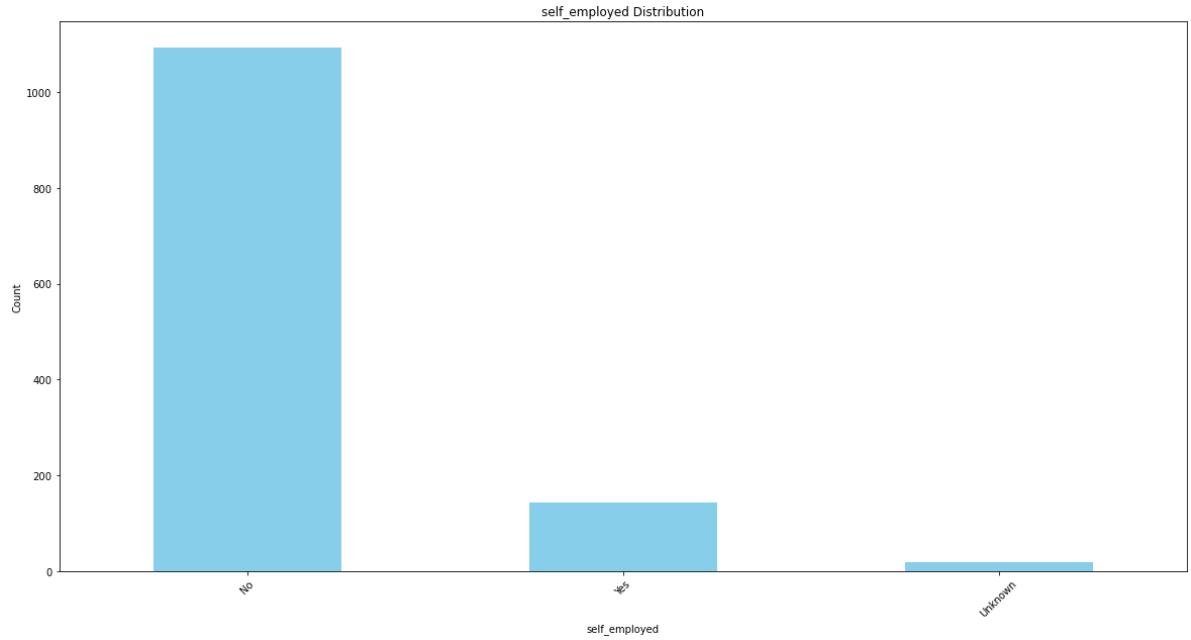
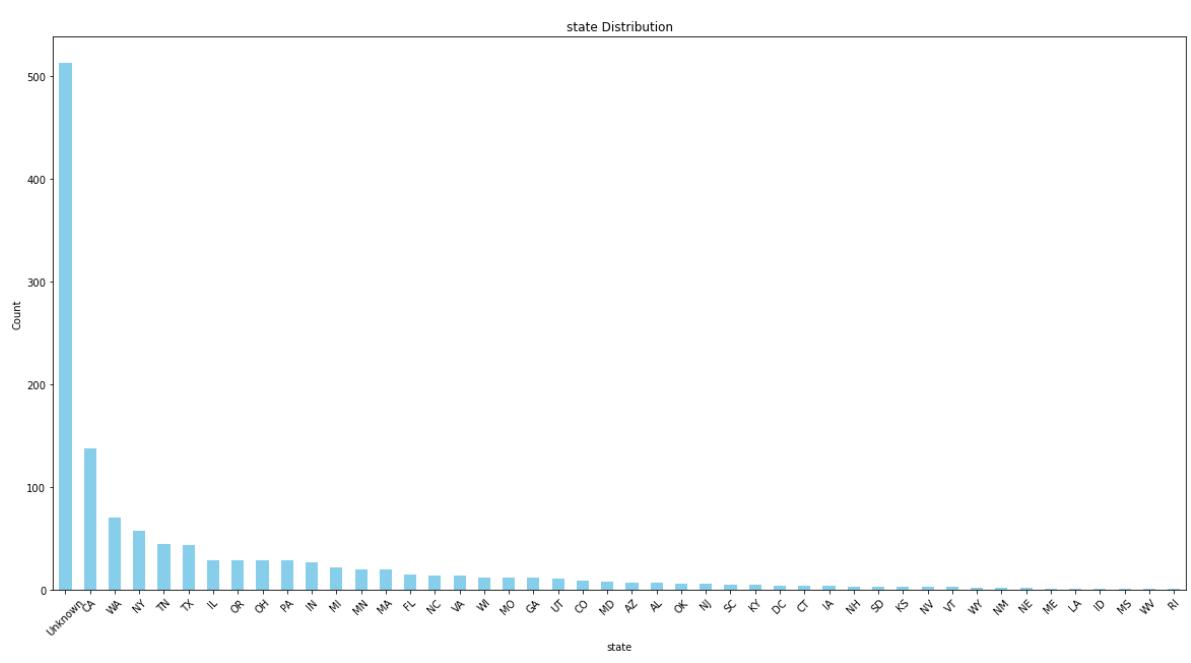
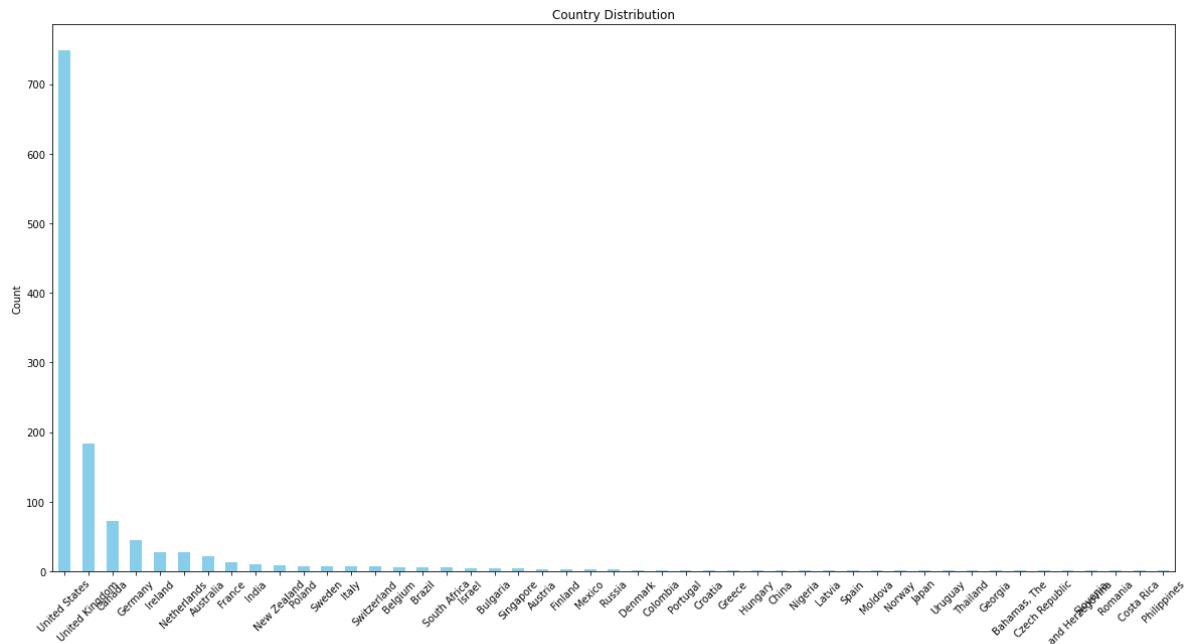
```
In [38]: import pandas as pd
import matplotlib.pyplot as plt

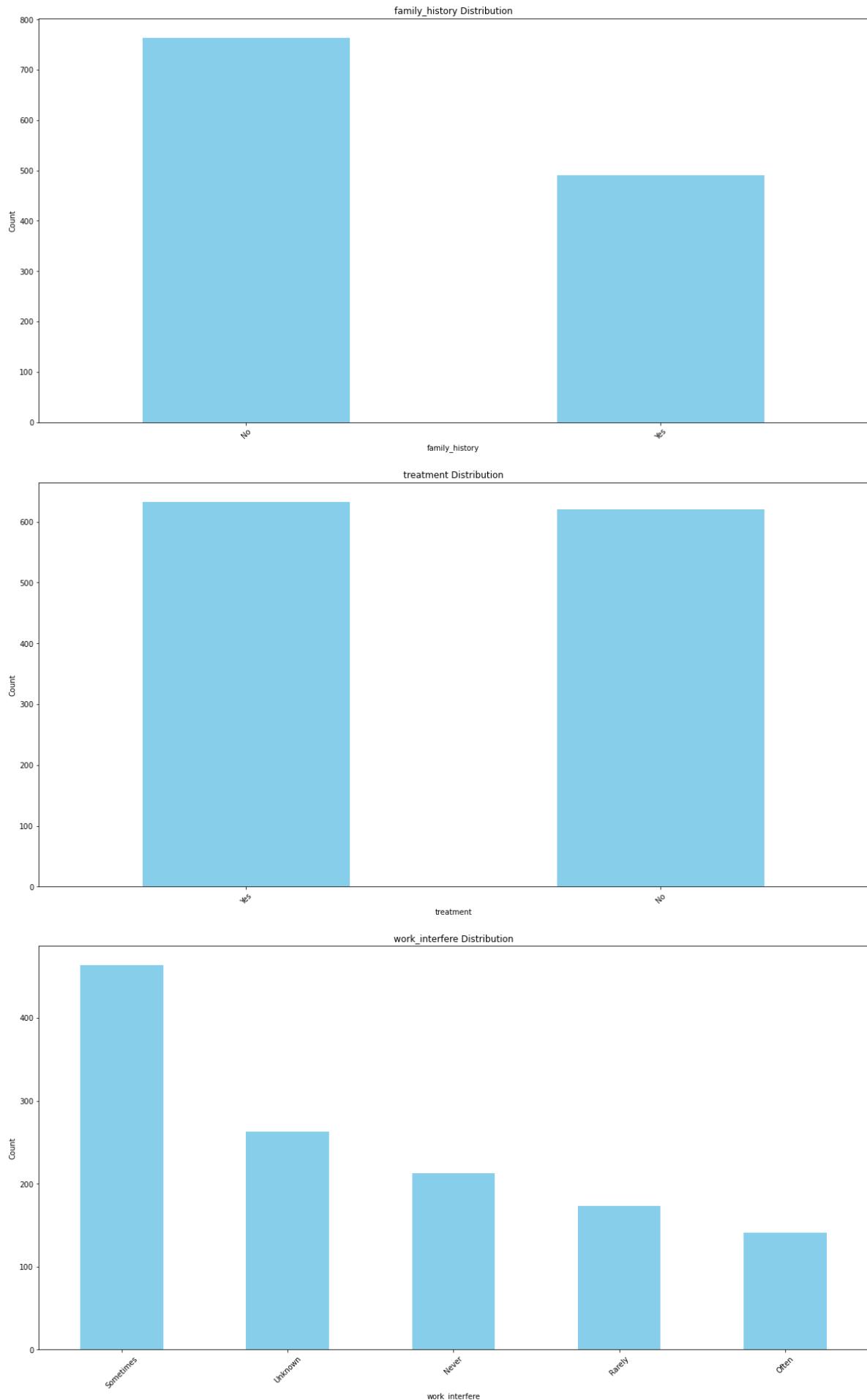
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

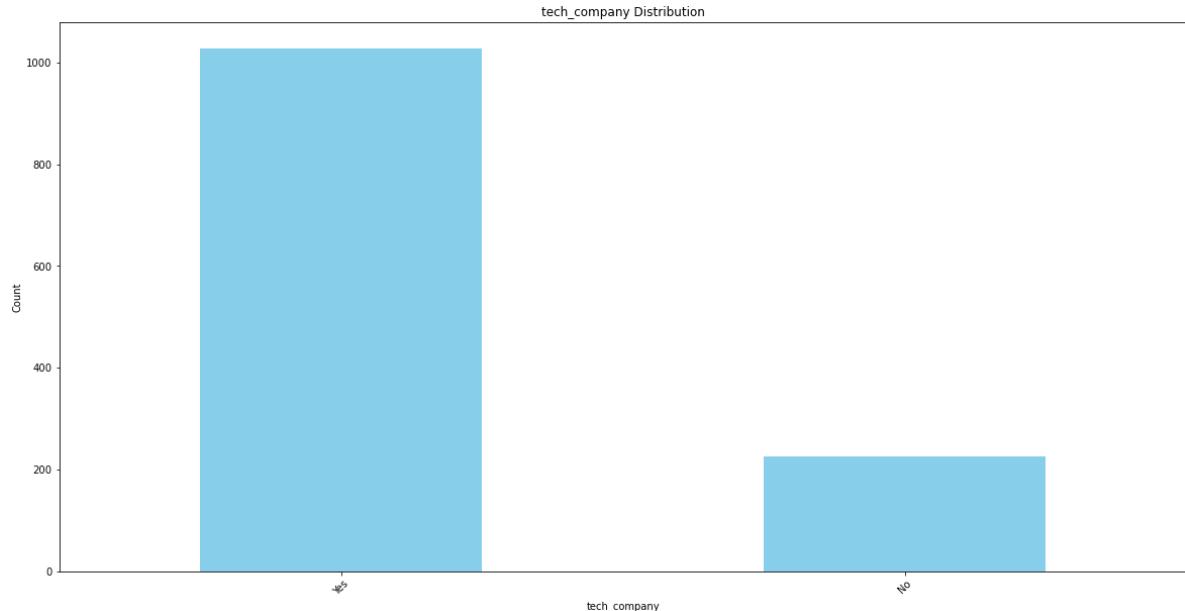
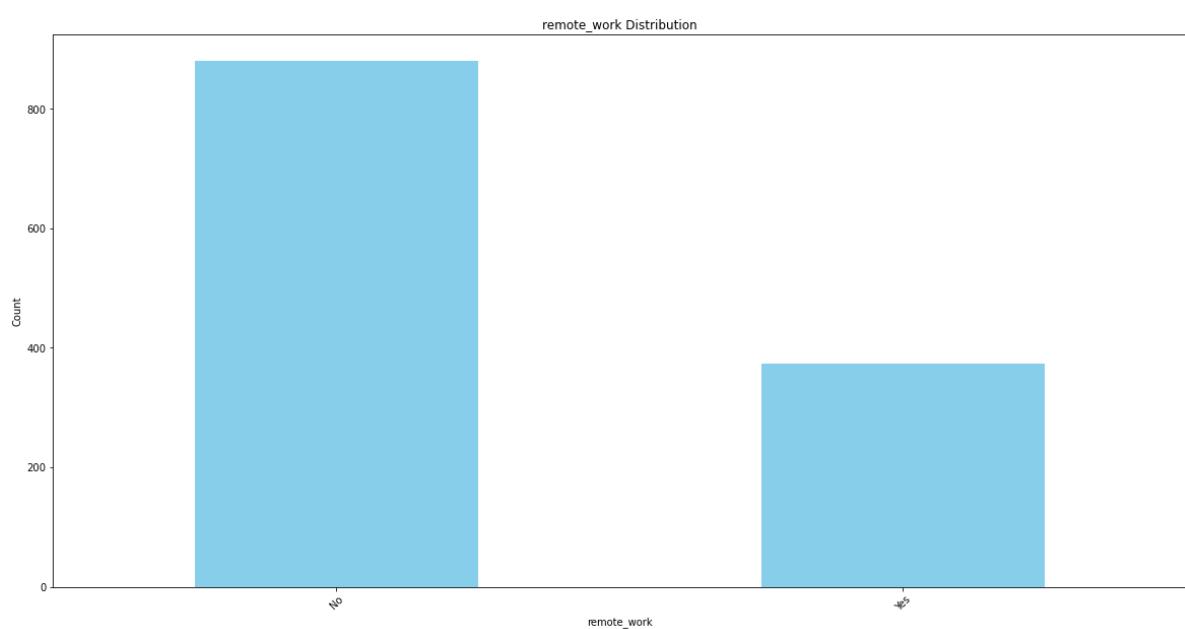
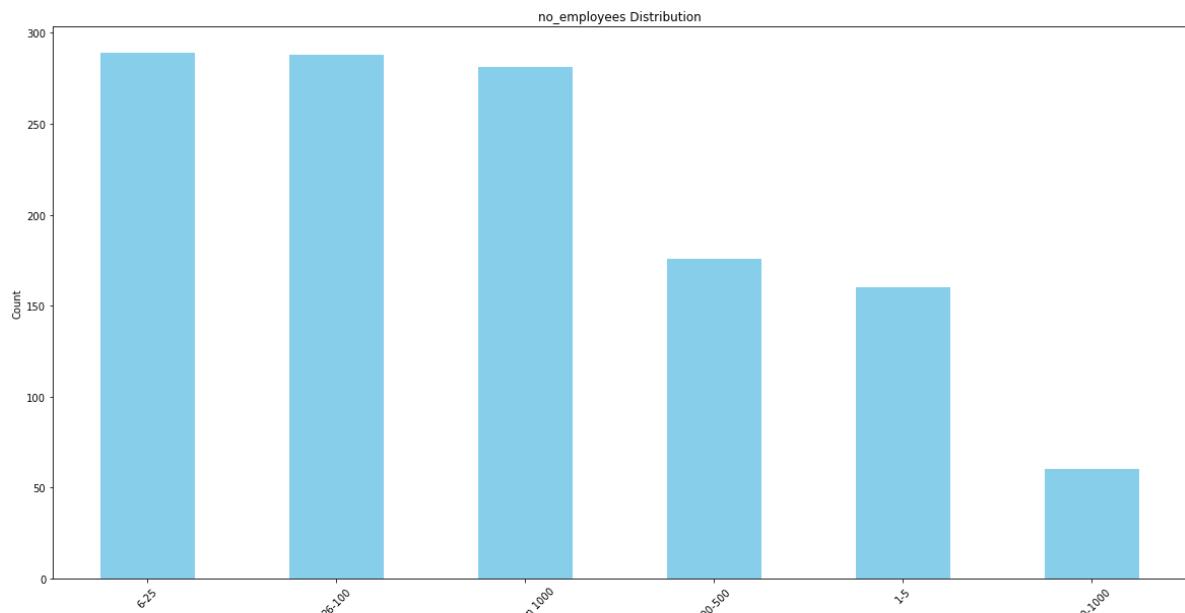
# Create visualizations for selected columns
columns_to_visualize = [
    'Age', 'Gender', 'Country', 'state', 'self_employed', 'family_history',
    'treatment', 'work_interfere', 'no_employees', 'remote_work',
    'tech_company', 'benefits', 'care_options', 'wellness_program', 'seek_anonymity',
  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

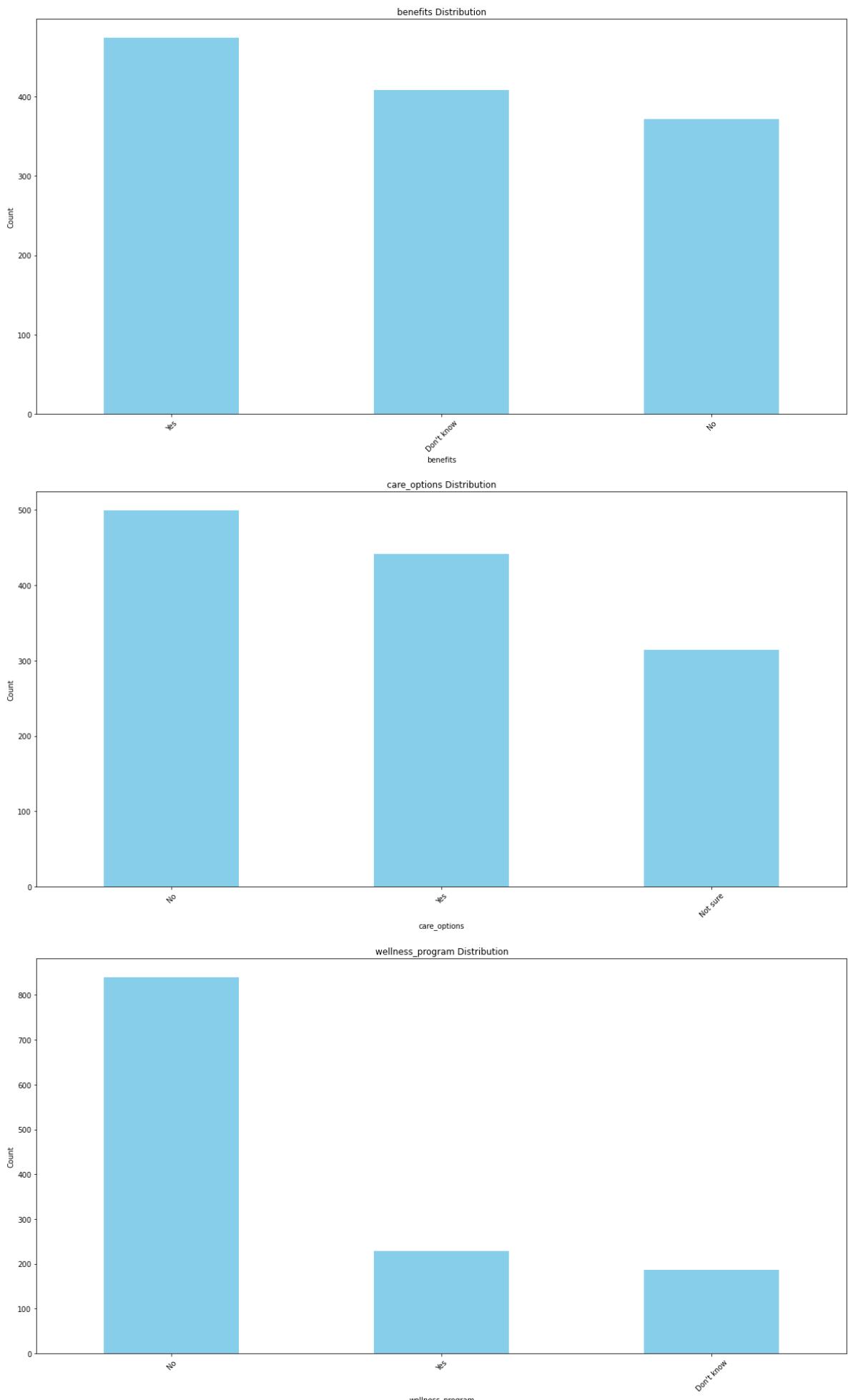
for column in columns_to_visualize:
    if column == 'Age':
        # Create a histogram for Age
        plt.figure(figsize=(8, 6))
        plt.hist(df[column], bins=20, color='skyblue')
        plt.xlabel(column)
        plt.ylabel('Count')
        plt.title(f'{column} Distribution')
        plt.show()
    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()
```

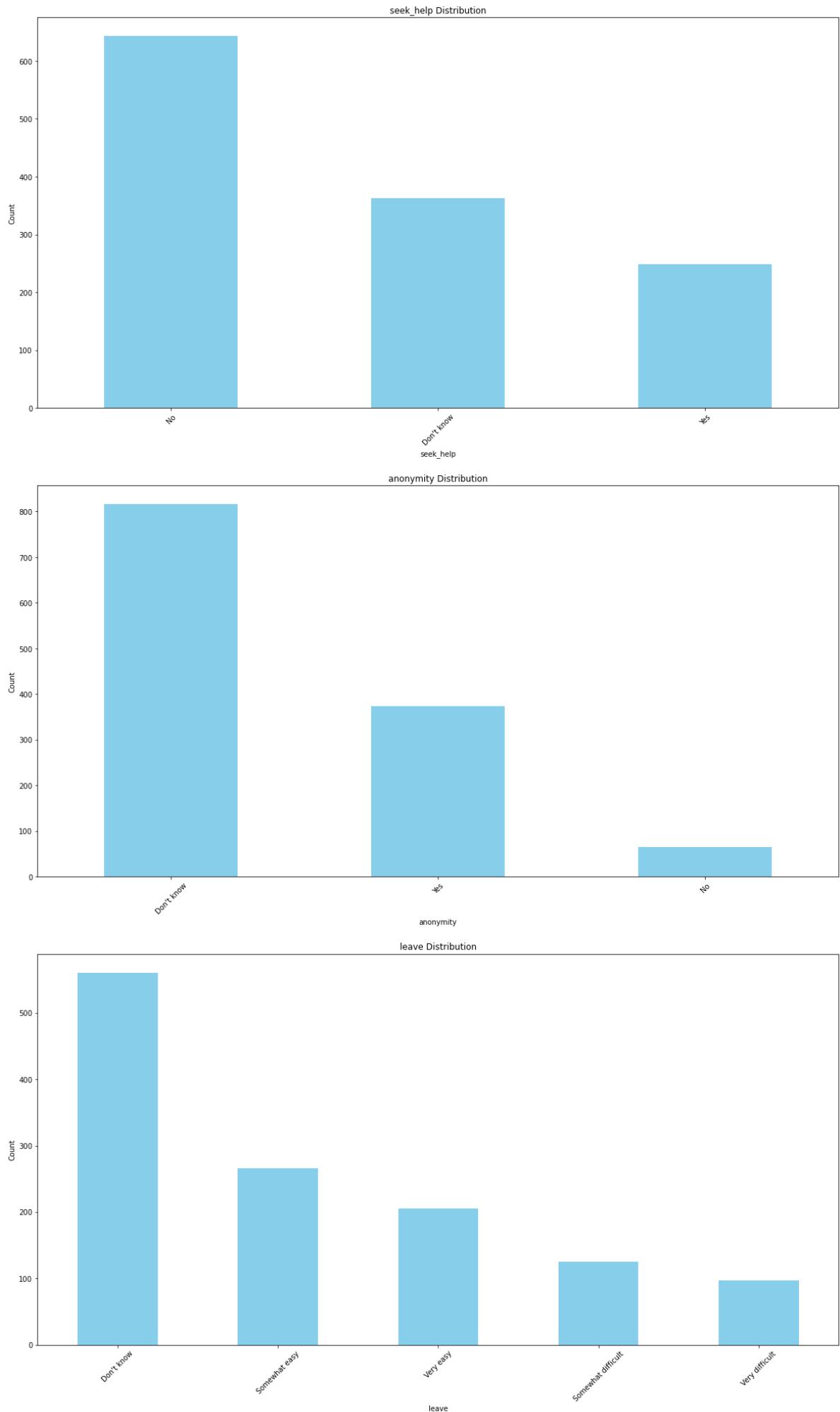


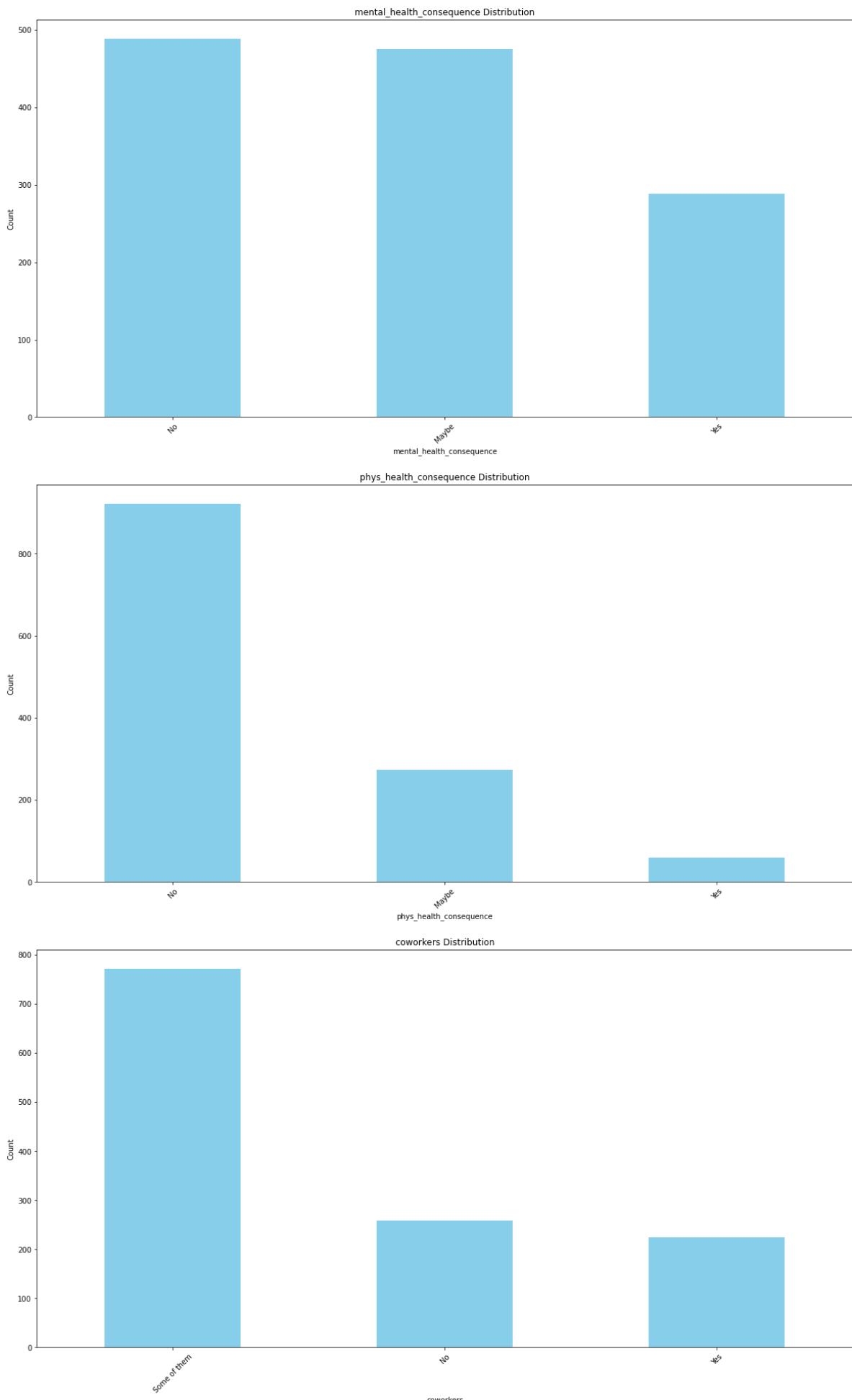


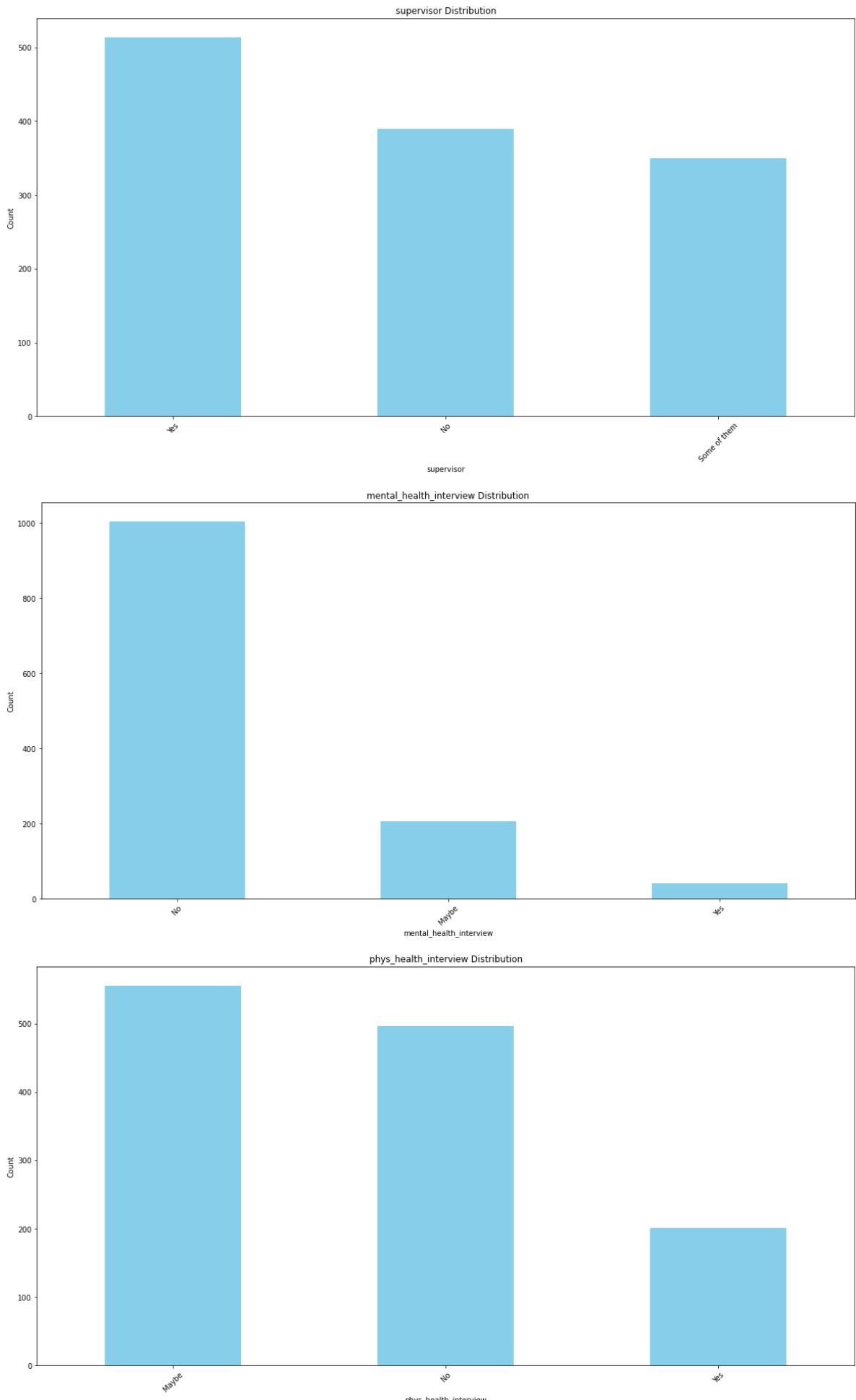


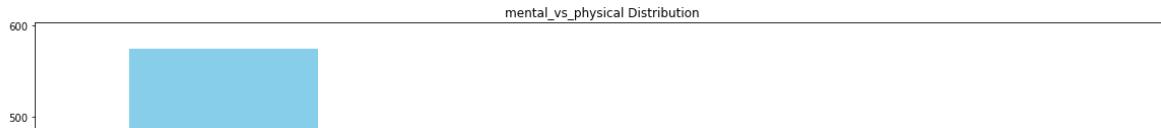












In [41]:

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

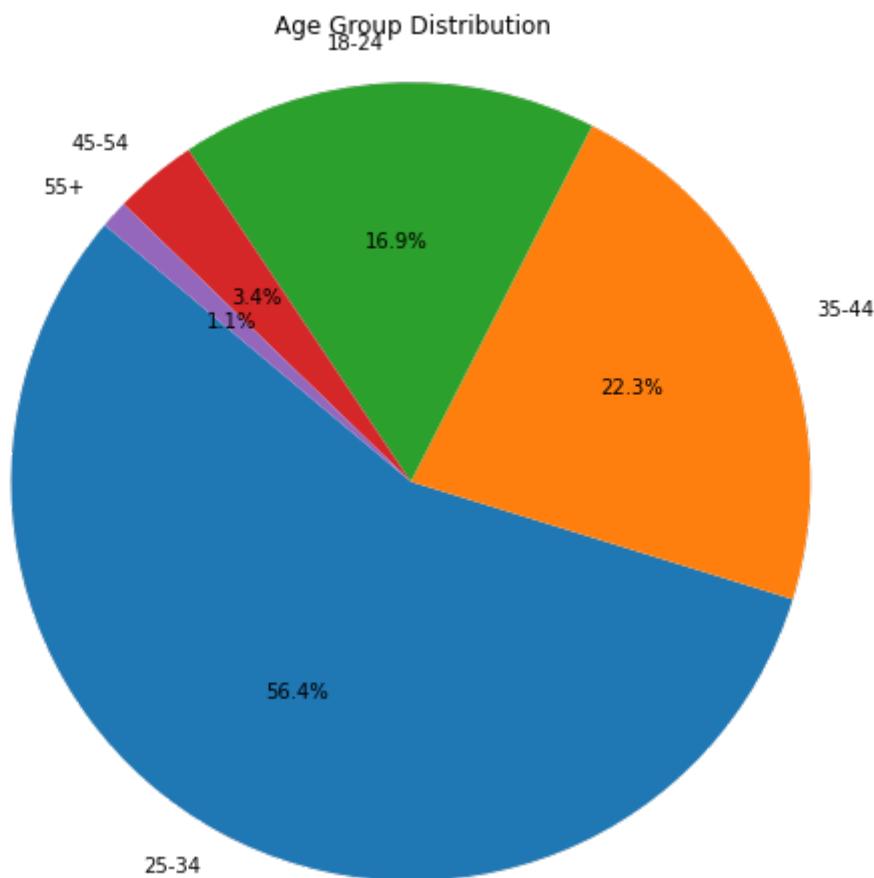
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# 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()

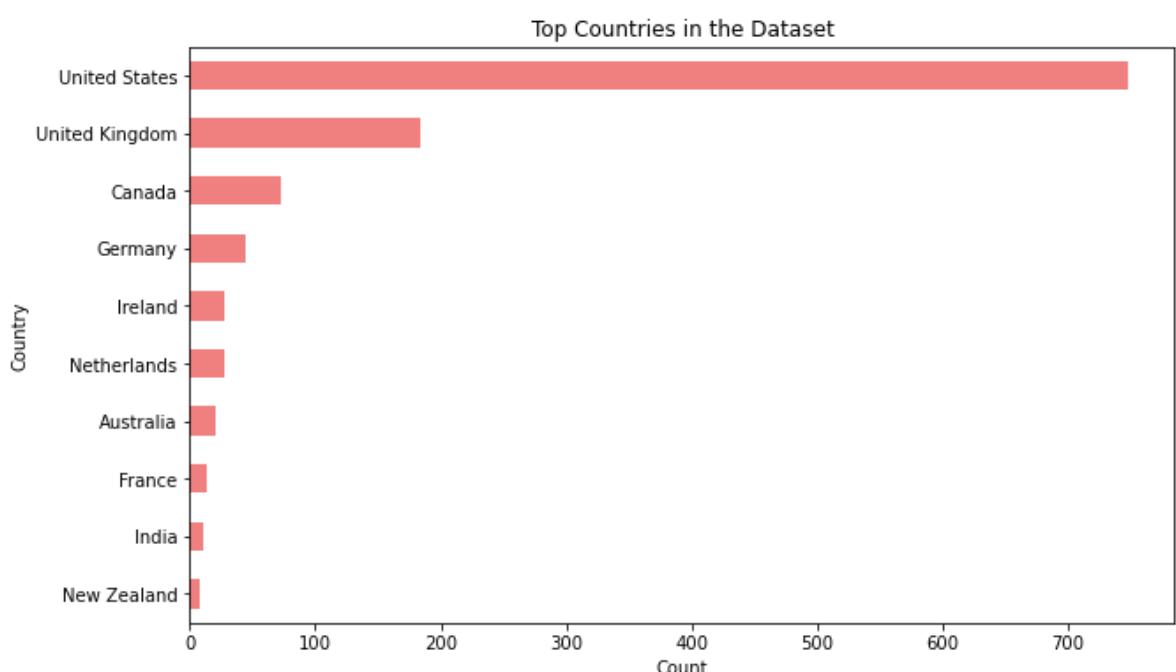
# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(age_group_counts, labels=age_group_counts.index, autopct='%1.1f%%')
plt.title('Age Group Distribution')
plt.axis('equal') # Equal aspect ratio ensures that the pie chart is circular
plt.show()
```



```
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

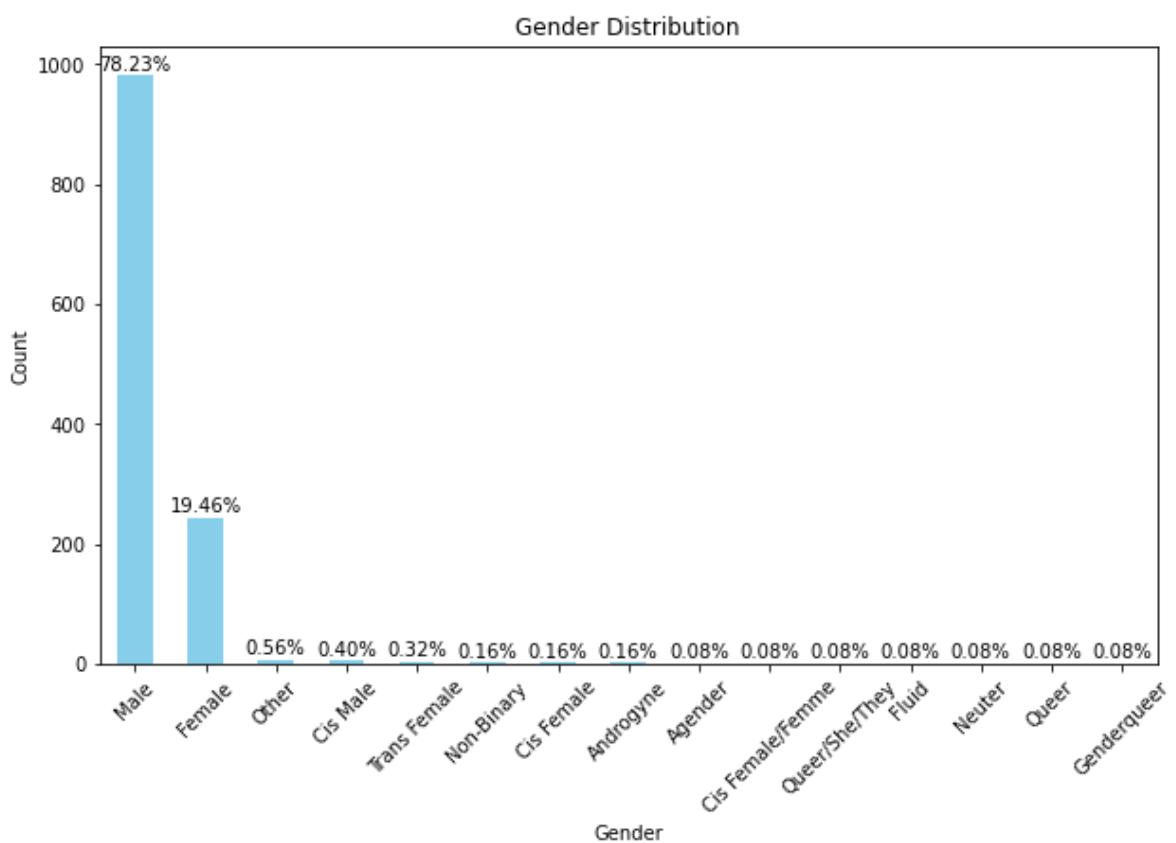
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

# 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()
```



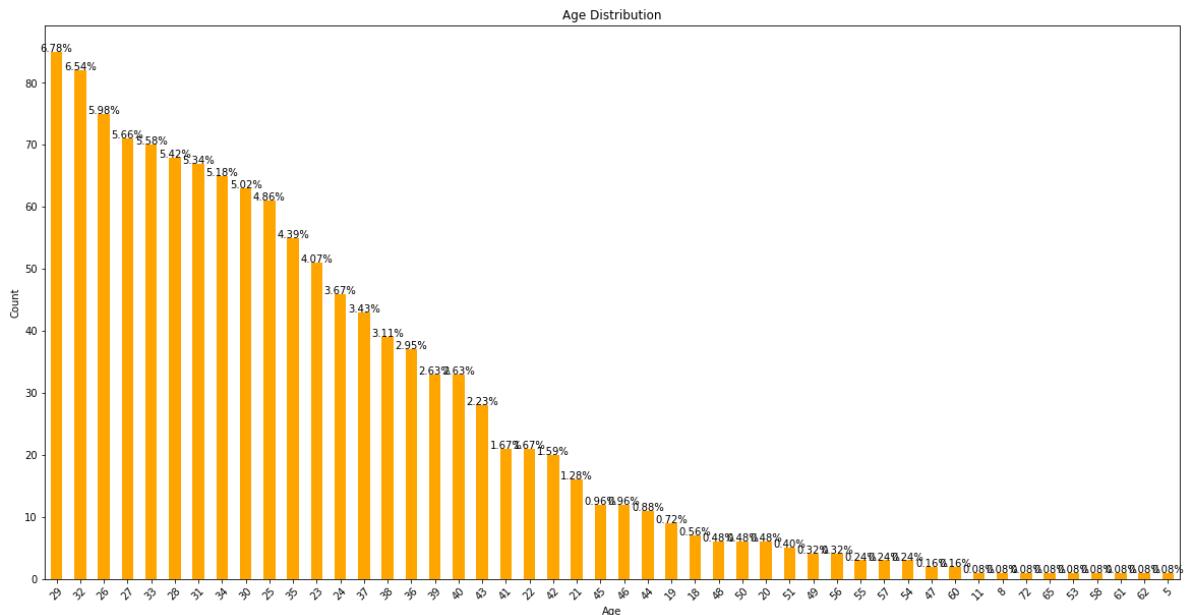
```
In [48]: import pandas as pd
import matplotlib.pyplot as plt

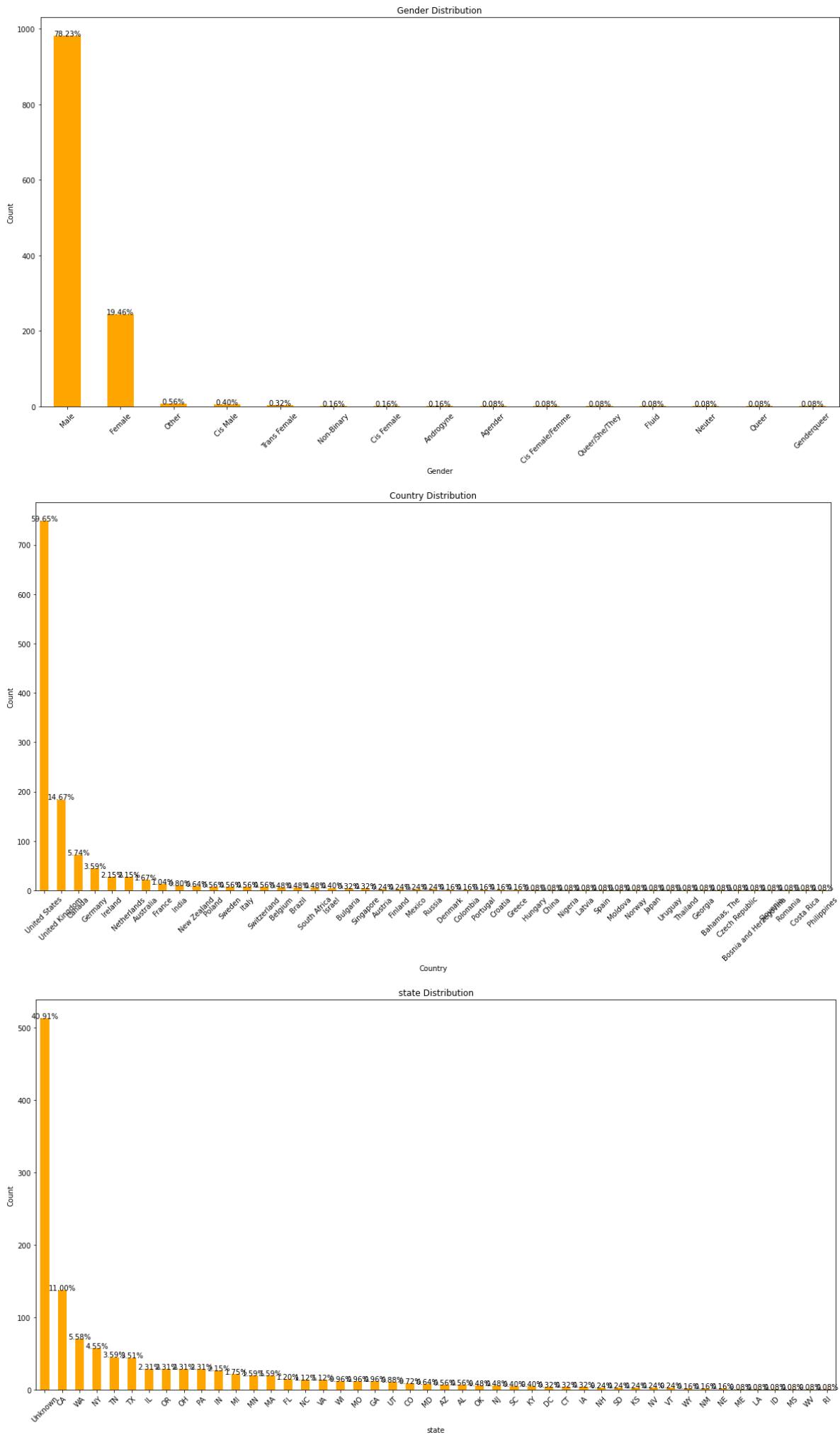
# Read the dataset
df = pd.read_csv('cleaned_survey.csv')

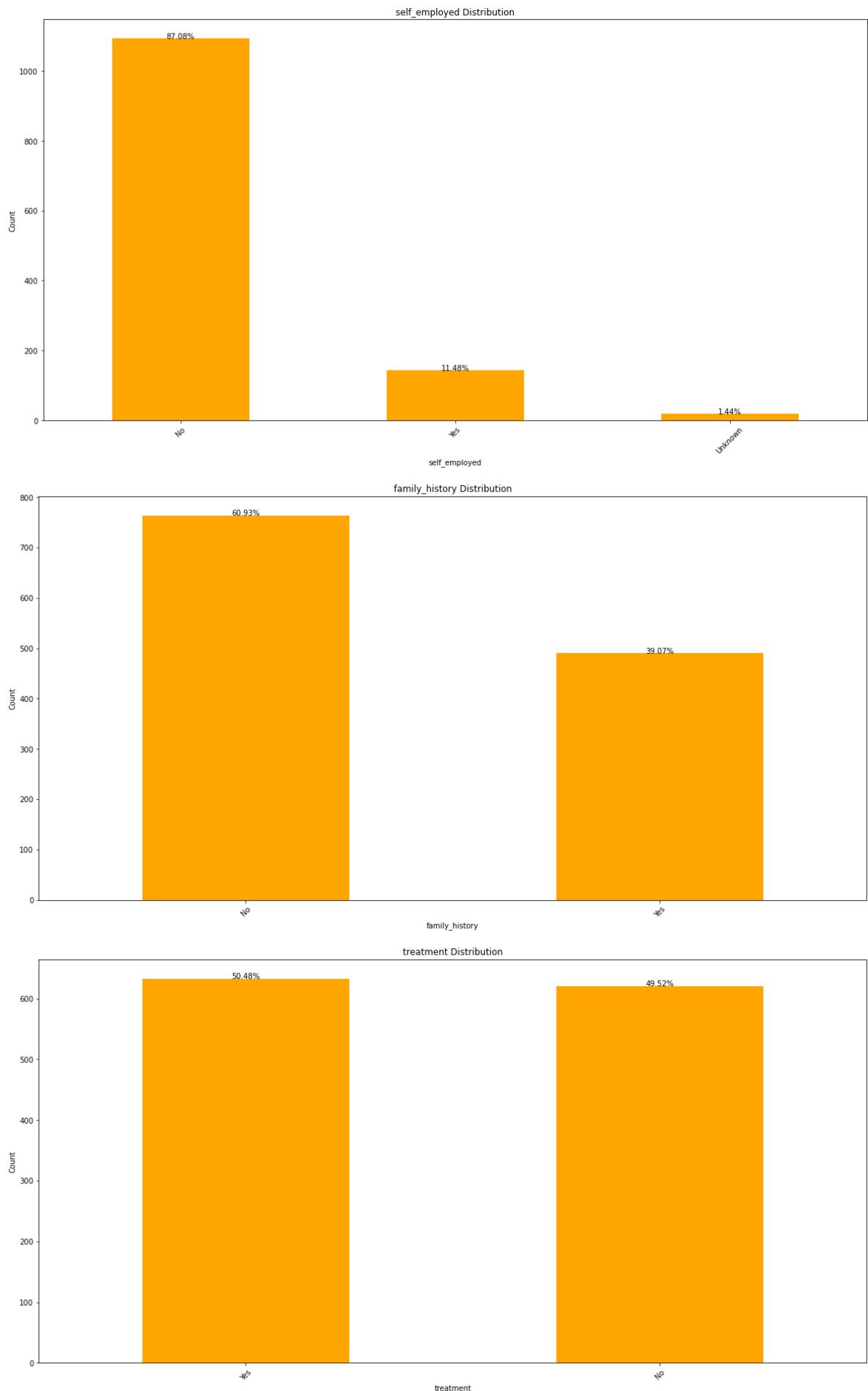
# Define a list of columns to visualize with percentages
columns_to_visualize = [
    'Age', 'Gender', 'Country', 'state', 'self_employed', 'family_history',
    'treatment', 'work_interfere', 'no_employees', 'remote_work',
    'tech_company', 'benefits', 'care_options', 'wellness_program', 'seek_I',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

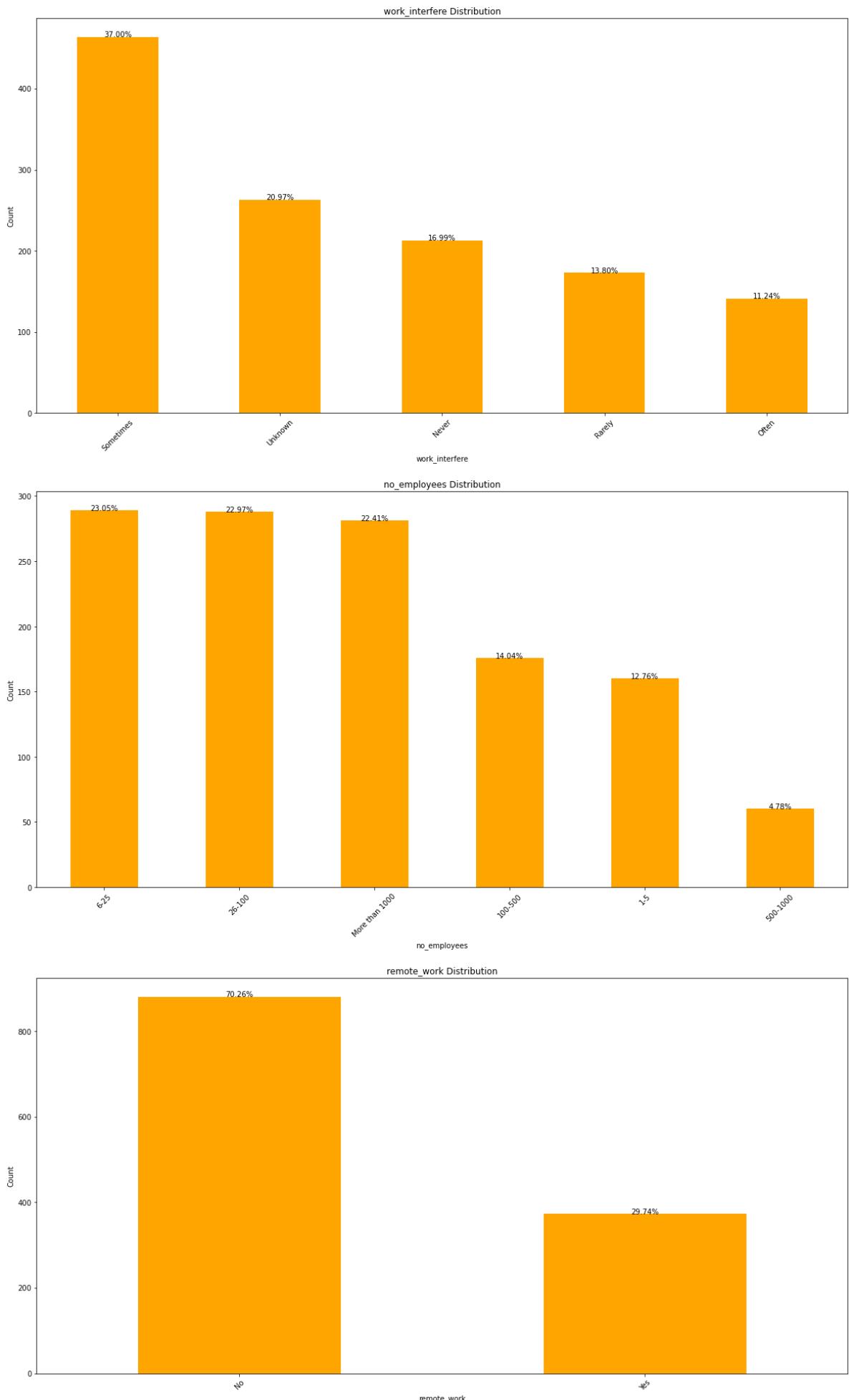
# Customize visualizations for each column
for column in columns_to_visualize:
    if column in ['Age', 'Gender', 'Country', 'state', 'self_employed', 'fa
        'treatment', 'work_interfere', 'no_employees', 'remote_work',
        'tech_company', 'benefits', 'care_options', 'wellness_program', 'seek_I
        'anonymity', 'leave', 'mental_health_consequence', 'phys_health_consequ
        'coworkers', 'supervisor', 'mental_health_interview', 'phys_health_inte
        '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)

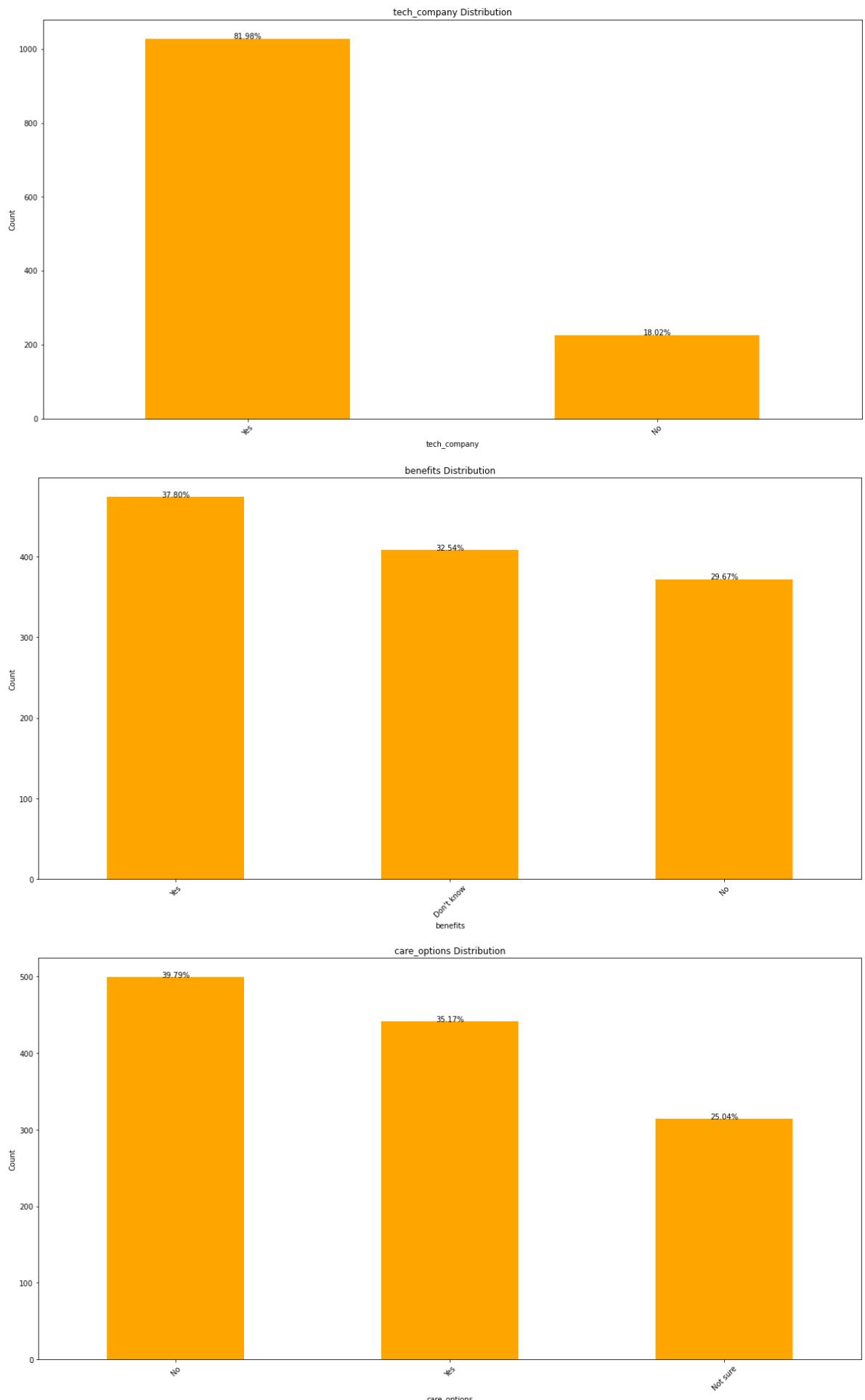
            # Add percentage labels to the bars
            total_count = len(df[column])
            for p in ax.patches:
                percentage = f"{100 * p.get_height() / total_count:.2f}%"
                ax.annotate(percentage, (p.get_x() + p.get_width() / 2, p.get_
                    plt.show()
```

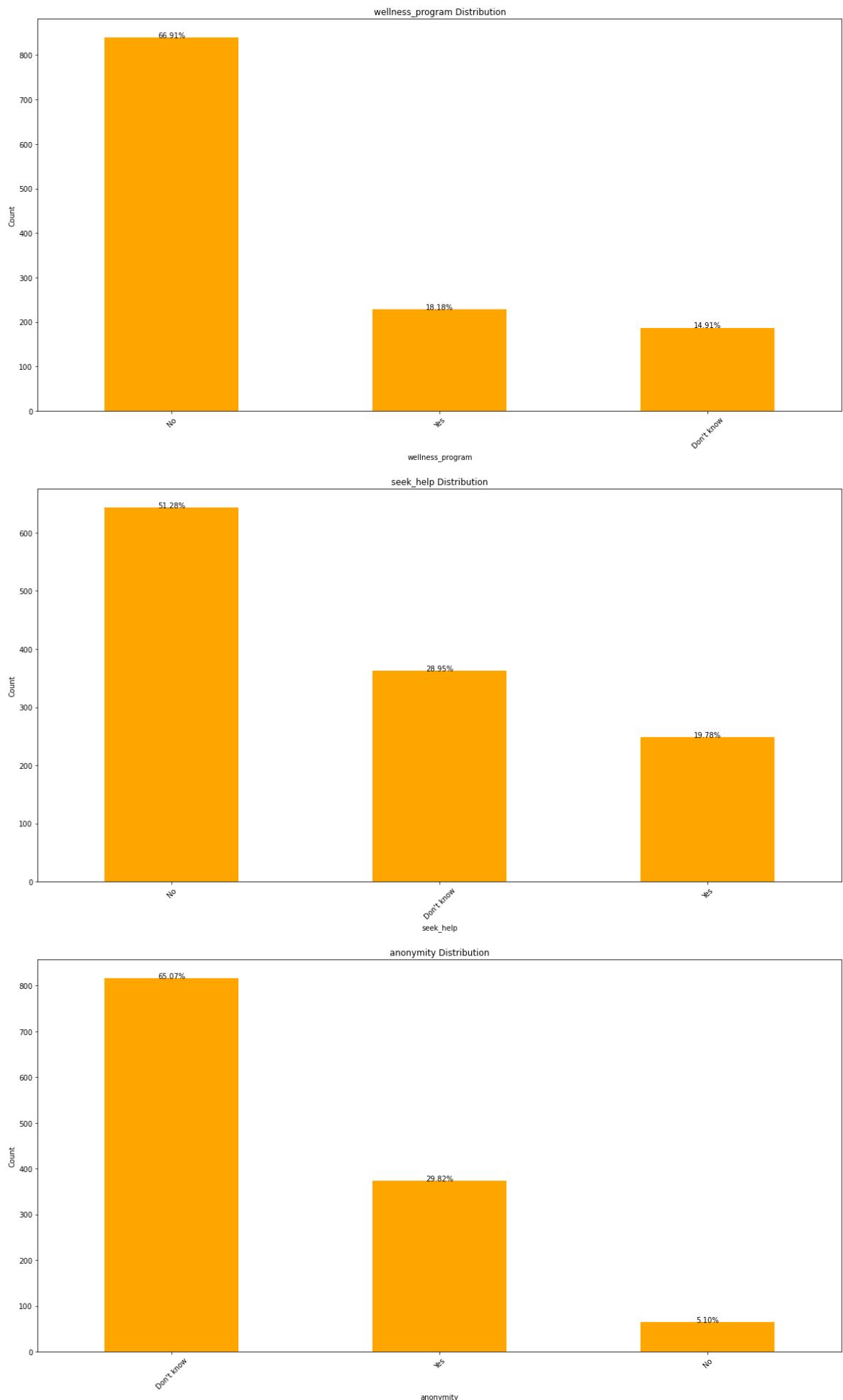


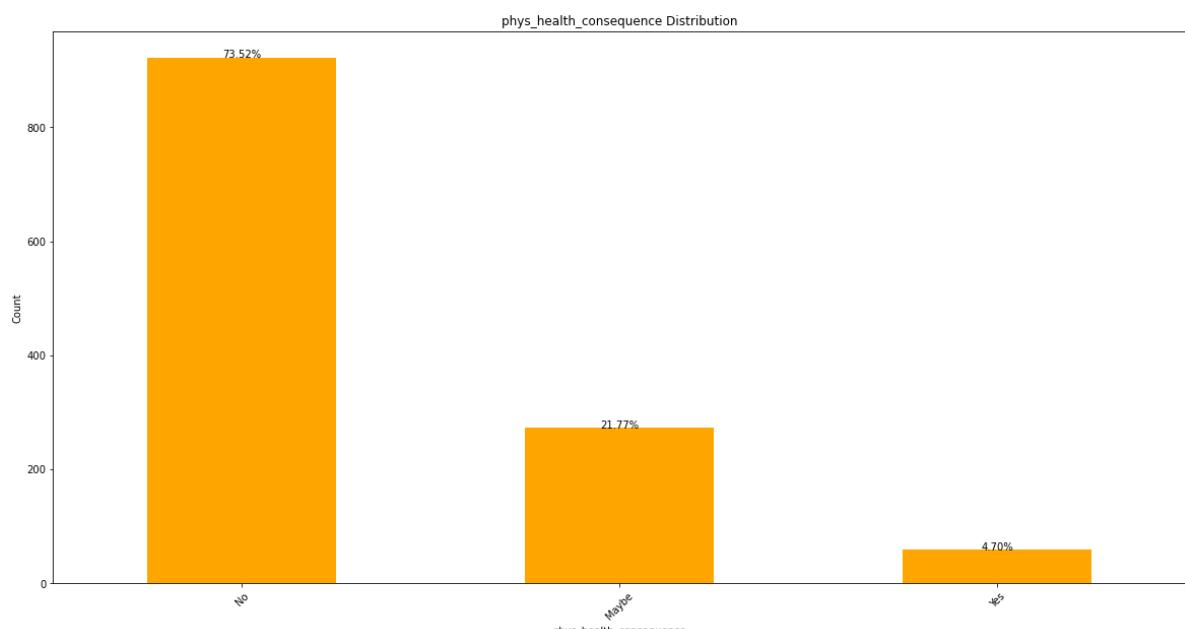
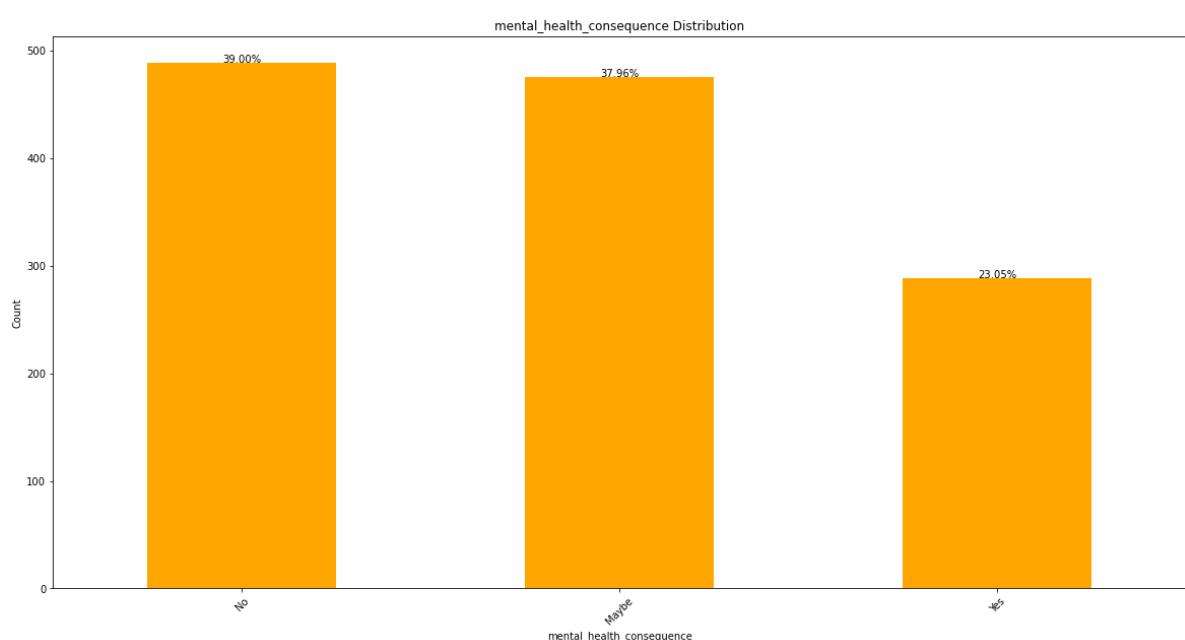
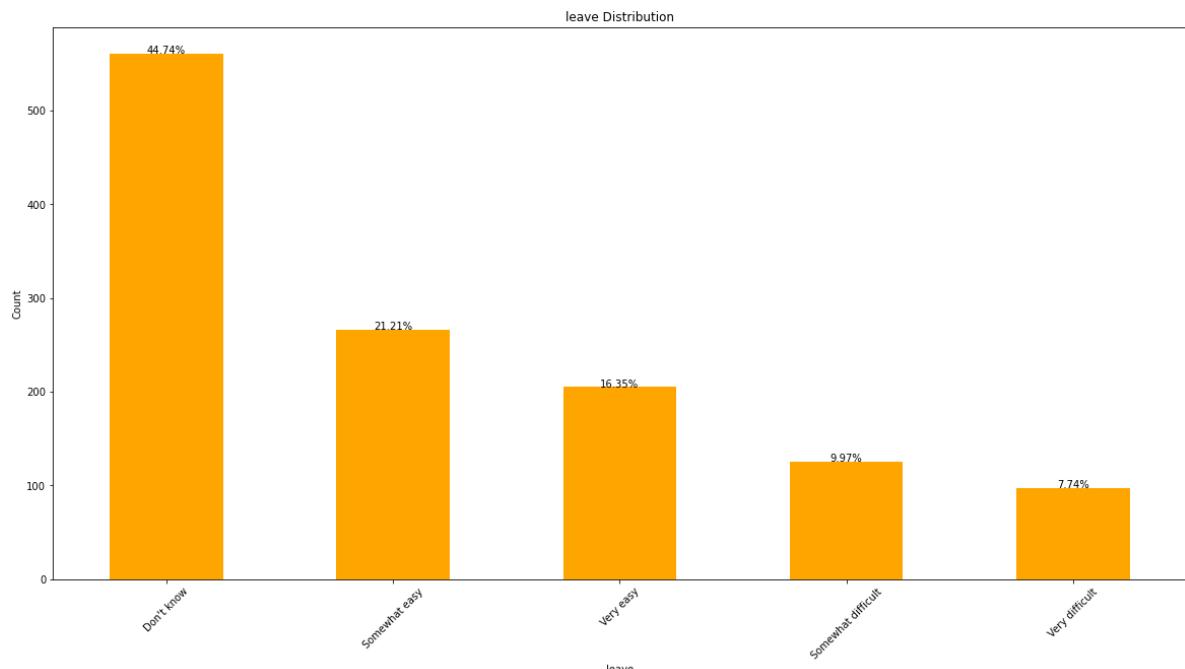


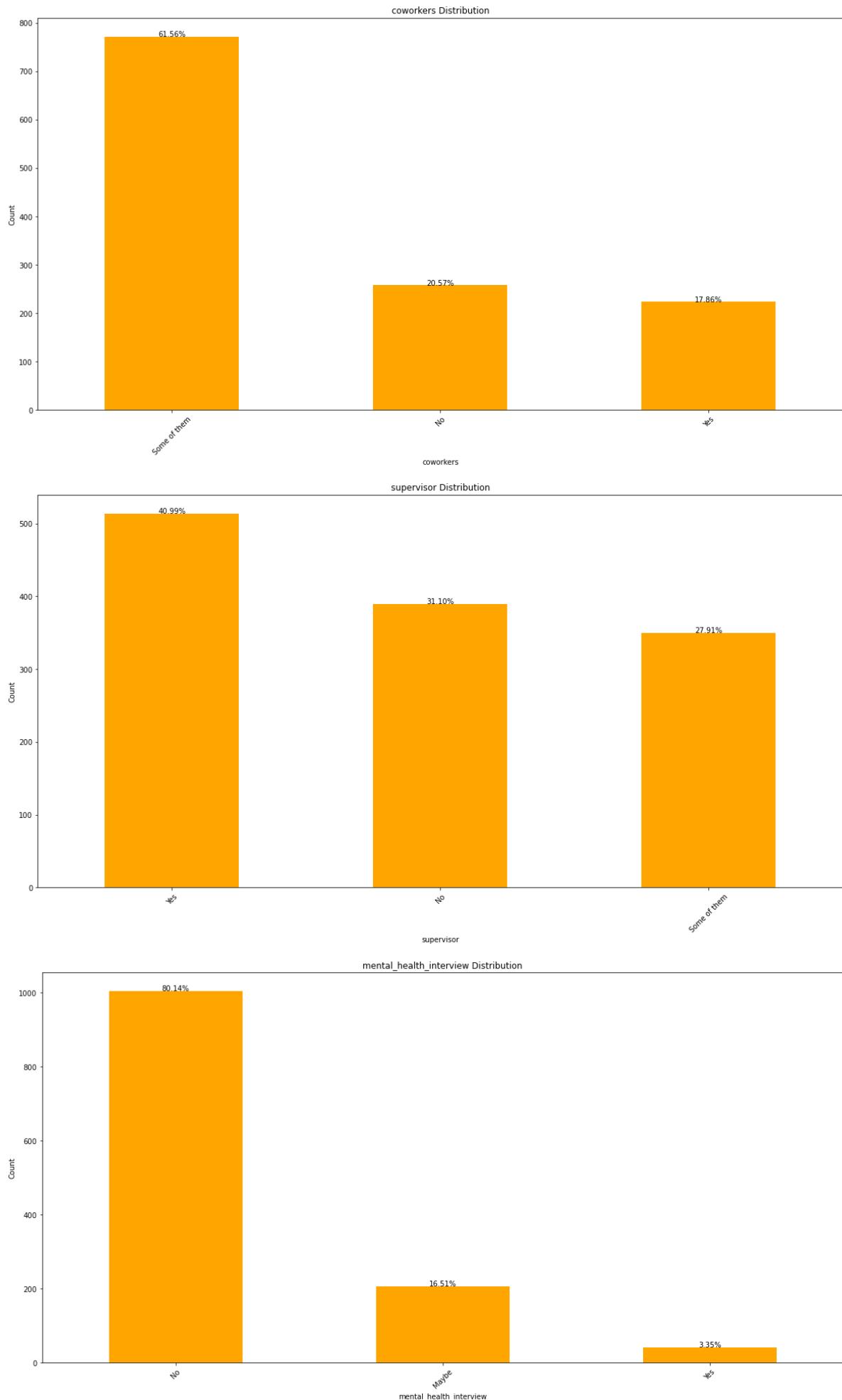


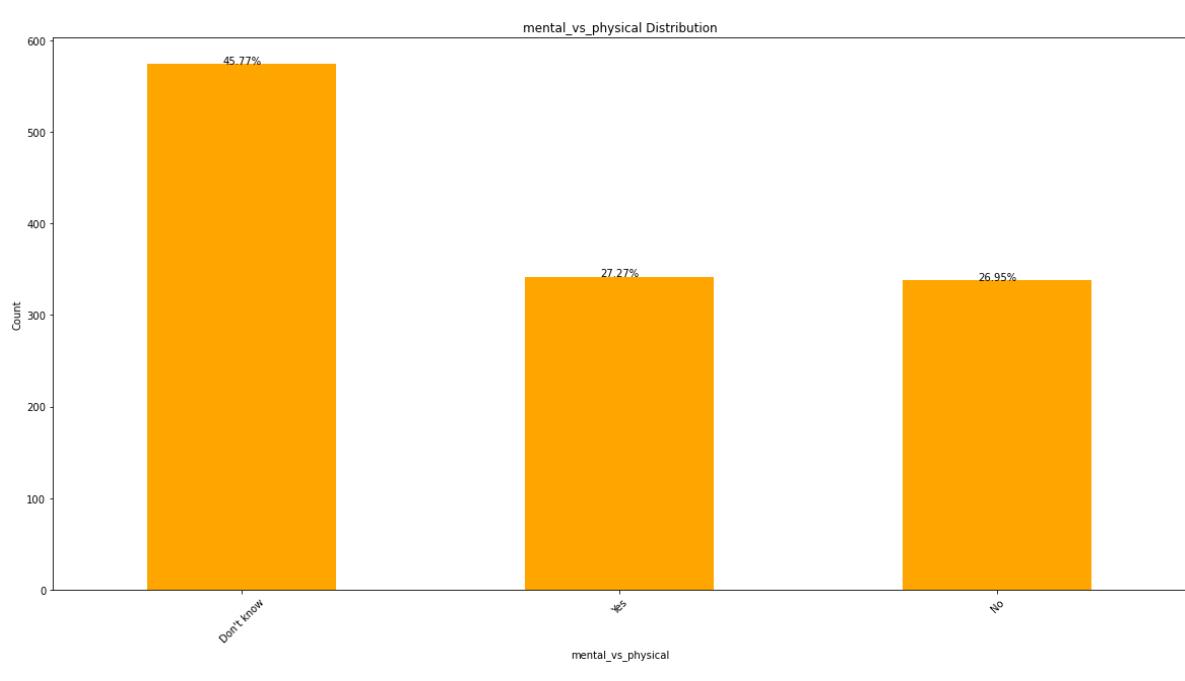
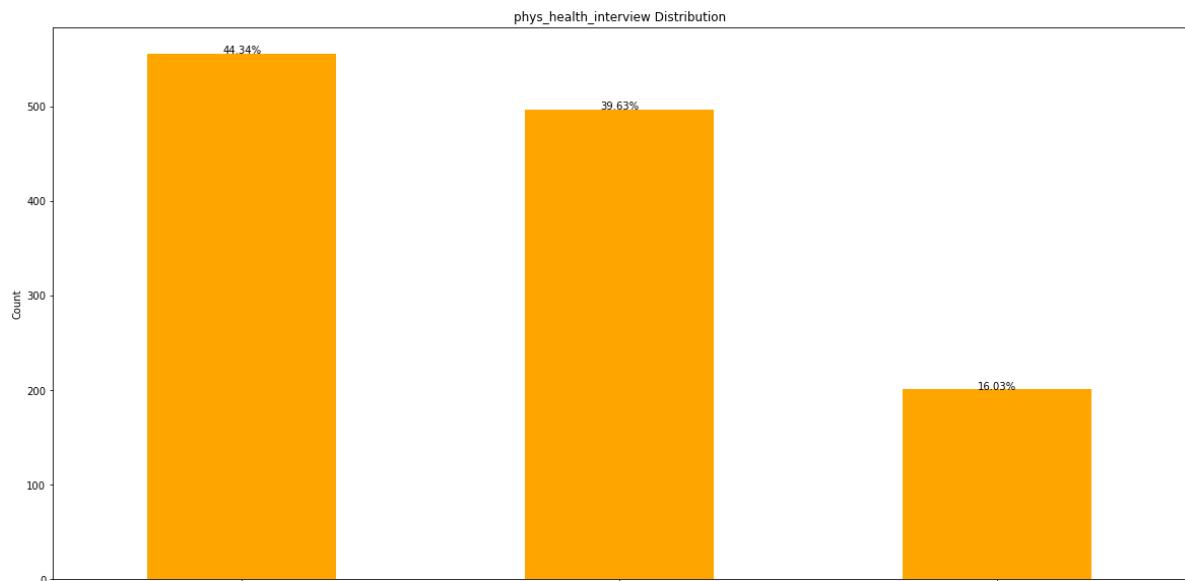










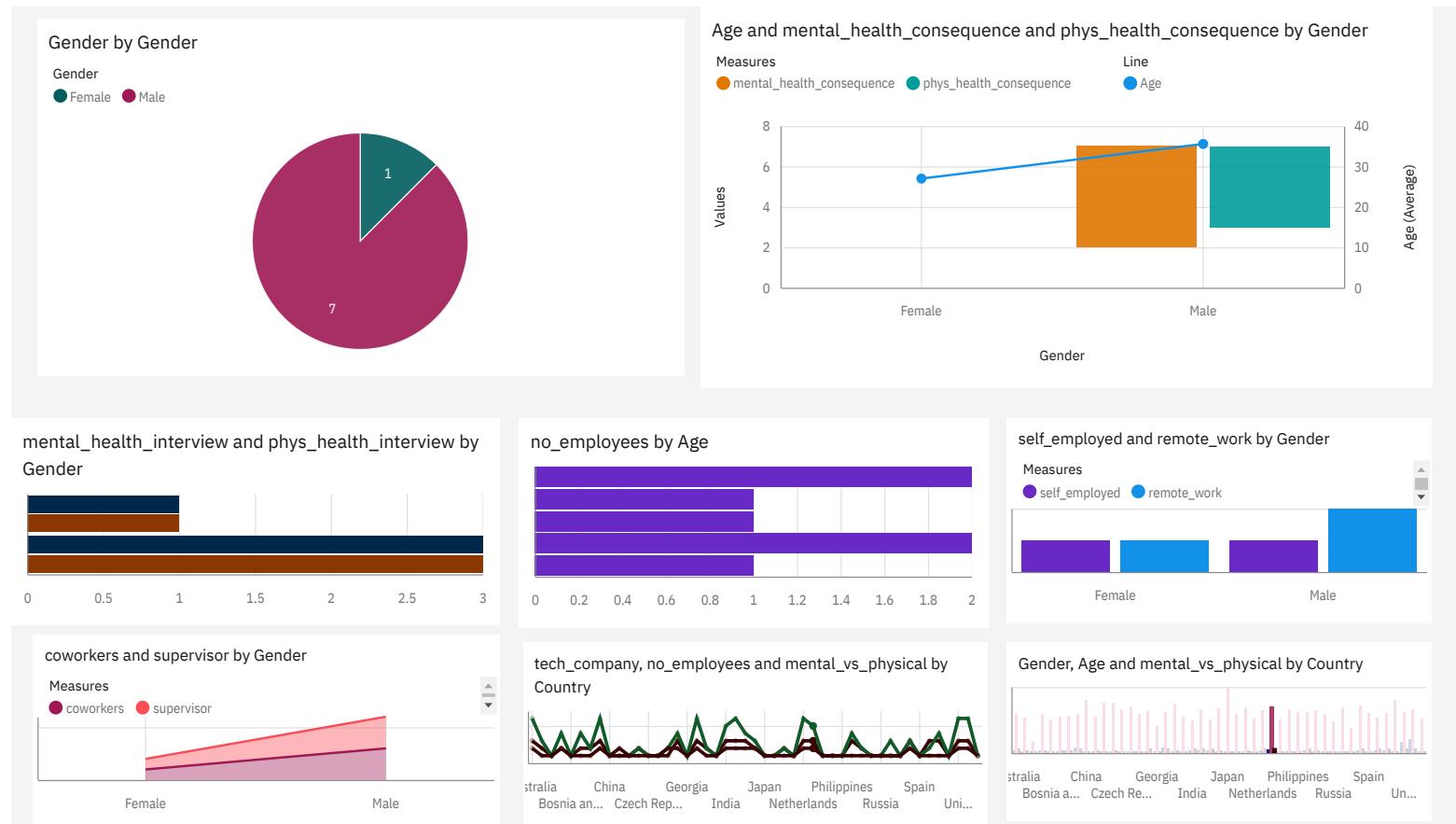


In []:

Tab 1



Tab 2



Model Development and Evaluation

Table of Contents

1. Introduction
2. Model Development
3. Model Evaluation
4. Conclusion
5. Recommendations

1. Introduction

Phase 4 of our project focuses on model development and evaluation. In this phase, we implemented various classification algorithms to predict or classify specific outcomes related to our public health awareness campaign analysis. The algorithms we employed are as follows:

- Logistic Regression
- Decision Tree
- k-Nearest Neighbors (KNN)
- Random Forest
- AdaBoost
- Gradient Boosting

Our goal is to assess the performance of these models and determine which one is best suited for our campaign's objectives.

2. Model Development

In this section, we describe the development of each model:

Logistic Regression:

Logistic Regression is a simple yet effective model for binary classification. We implemented it to predict whether an individual is likely to be affected by mental health issues based on the features available in the dataset.

Decision Tree:

The Decision Tree model is a tree-like structure that recursively splits the data based on features. We employed it to identify key factors contributing to mental health issues in the tech industry.

k-Nearest Neighbors (KNN):

KNN is a distance-based classification method. We used it to classify individuals into groups based on similar attributes, which can be helpful in targeting specific demographics in the campaign.

Random Forest:

Random Forest is an ensemble learning technique that combines multiple Decision Trees. It was employed to improve the accuracy of our classification by reducing overfitting and enhancing generalization.

AdaBoost:

AdaBoost is another ensemble learning method, which sequentially combines multiple weak learners to create a strong classifier. We utilized AdaBoost to enhance the performance of our classification model.

Gradient Boosting:

Gradient Boosting is yet another ensemble technique that builds models in a sequential manner. It was used to improve the predictive accuracy and precision of our campaign's outcome.

3. Model Evaluation

To evaluate the performance of these models, we employed several evaluation metrics, including but not limited to:

- Accuracy
- Precision
- Recall
- F1 Score
- ROC-AUC
- Confusion Matrix

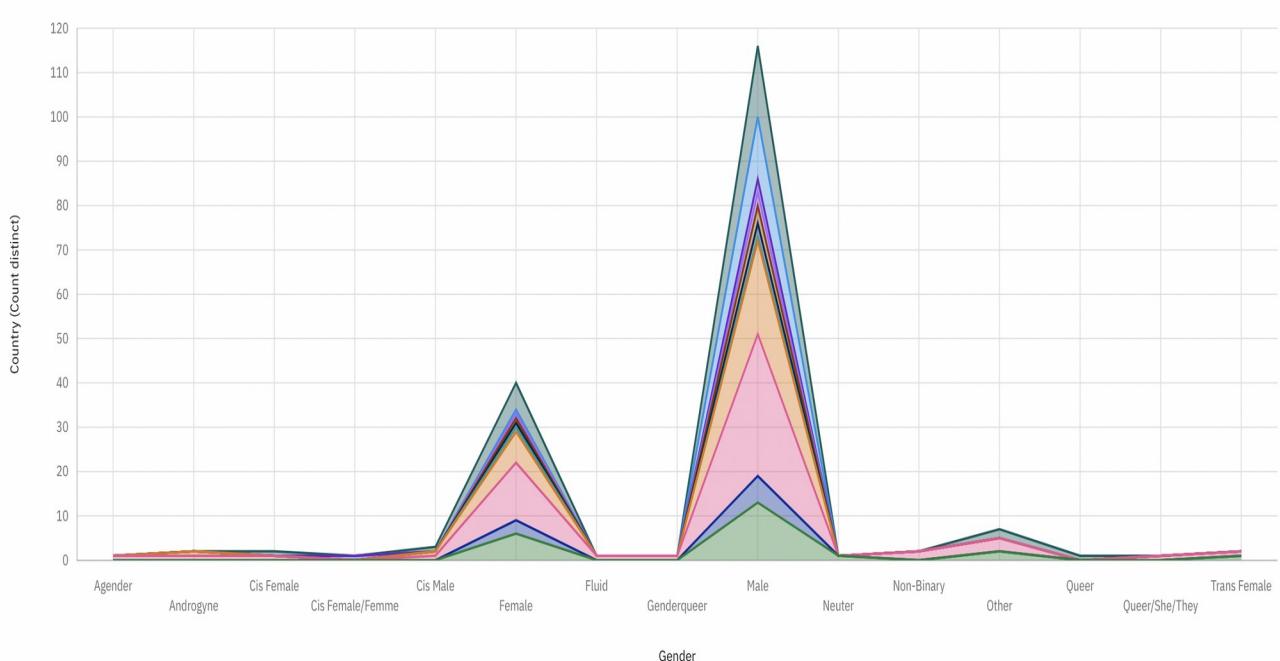
The results of the model evaluation are summarized in the following table:

	method		cv score	mean score	std score	recall score
0	Logistic Regression	[0.80899, 0.88764, 0.88764, 0.88636, 0.875]	0.869127	0.030442	0.863158	
1	Decision Tree Classifier	[0.76404, 0.68539, 0.83146, 0.84091, 0.79545]	0.783453	0.056111	0.747368	
2	KNN Classifier	[0.57303, 0.67416, 0.70787, 0.65909, 0.65909]	0.654648	0.044527	0.605263	
3	Random Forest Classifier	[0.77528, 0.83146, 0.88764, 0.90909, 0.86364]	0.853422	0.046824	0.810526	
4	Ada Boost Classifier	[0.80899, 0.85393, 0.92135, 0.84091, 0.85227]	0.855490	0.036674	0.857895	
5	Gradient Boosting Classifier	[0.78652, 0.80899, 0.89888, 0.89773, 0.86364]	0.851149	0.045953	0.800000	

Country by Gender colored by self_employed, tech_company and remote_work



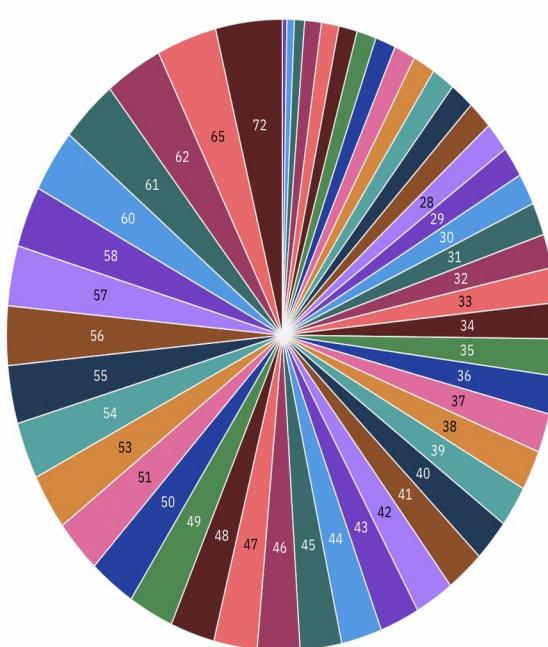
self_employed - tech_company - remote_work
 ● No | No | No ● No | No | Yes ● No | Yes | No ● No | Yes | Yes ● Unknown | No | No ● Unknown | Yes | No ● Unknown | Yes | Yes ● Yes | No | No ● Yes | No | Yes ● Yes | Yes | No ● Yes | Yes | Yes



Age by Age



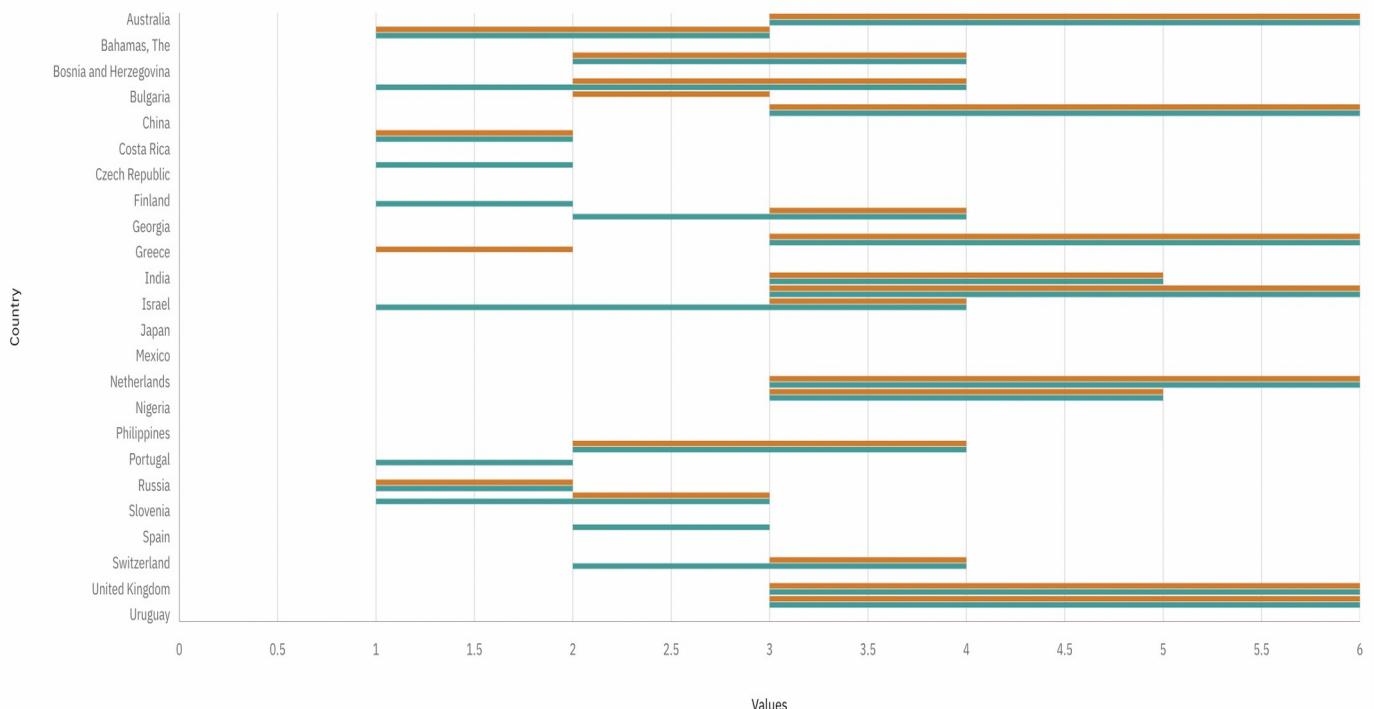
Age
 ● 5 ● 8 ● 11 ● 18 ● 19 ● 20 ● 21 ● 22 ● 23 ● 24 ● 25 ● 26 ● 27 ● 28 ● 29 ● 30 ● 31 ● 32 ● 33 ● 34 ● 35 ● 36 ● 37 ● 38 ● 39 ● 40 ● 41 ● 42 ● 43 ● 44 ● 45 ● 46 ● 47 ● 48 ● 49 ● 50
 ● 51 ● 53 ● 54 ● 55 ● 56 ● 57 ● 58 ● 60 ● 61 ● 62 ● 65 ● 72



mental_health_consequence and phys_health_consequence by Country

Measures

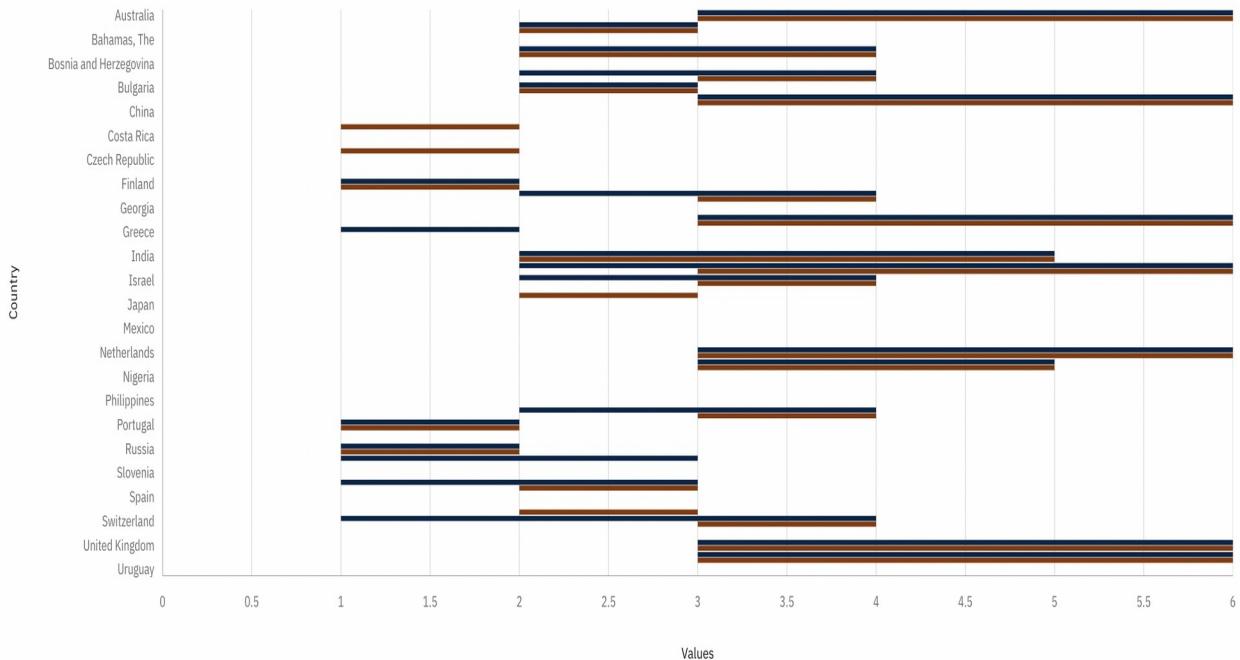
● mental_health_consequence ● phys_health_consequence



mental_health_interview and phys_health_interview by Country

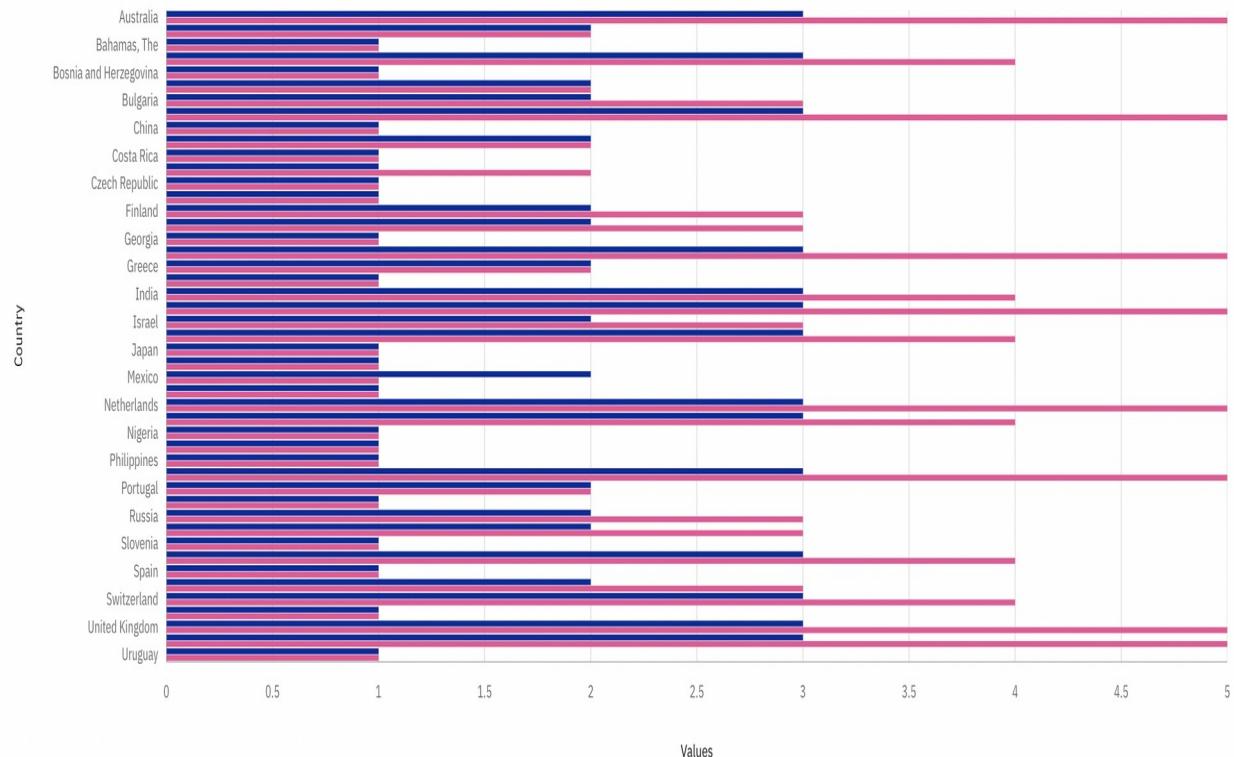
Measures

● mental_health_interview ● phys_health_interview

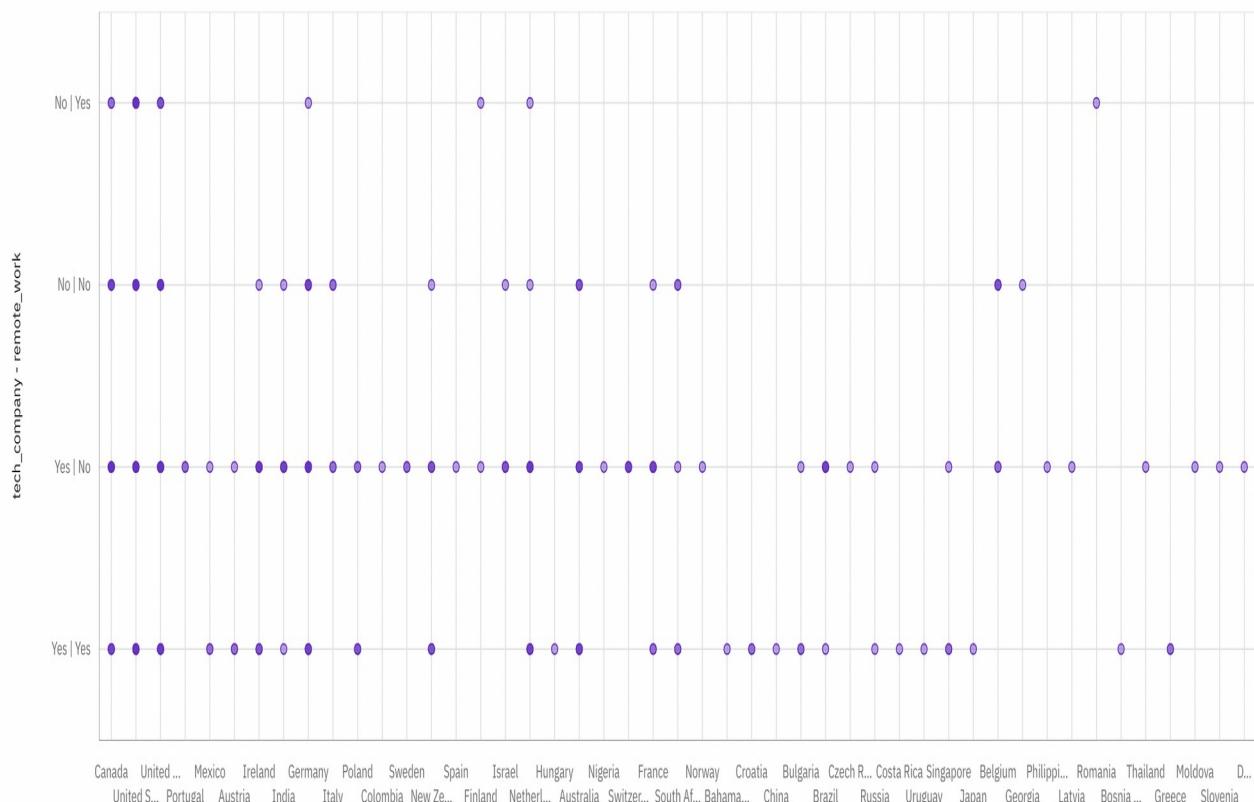


anonymity and leave by Country

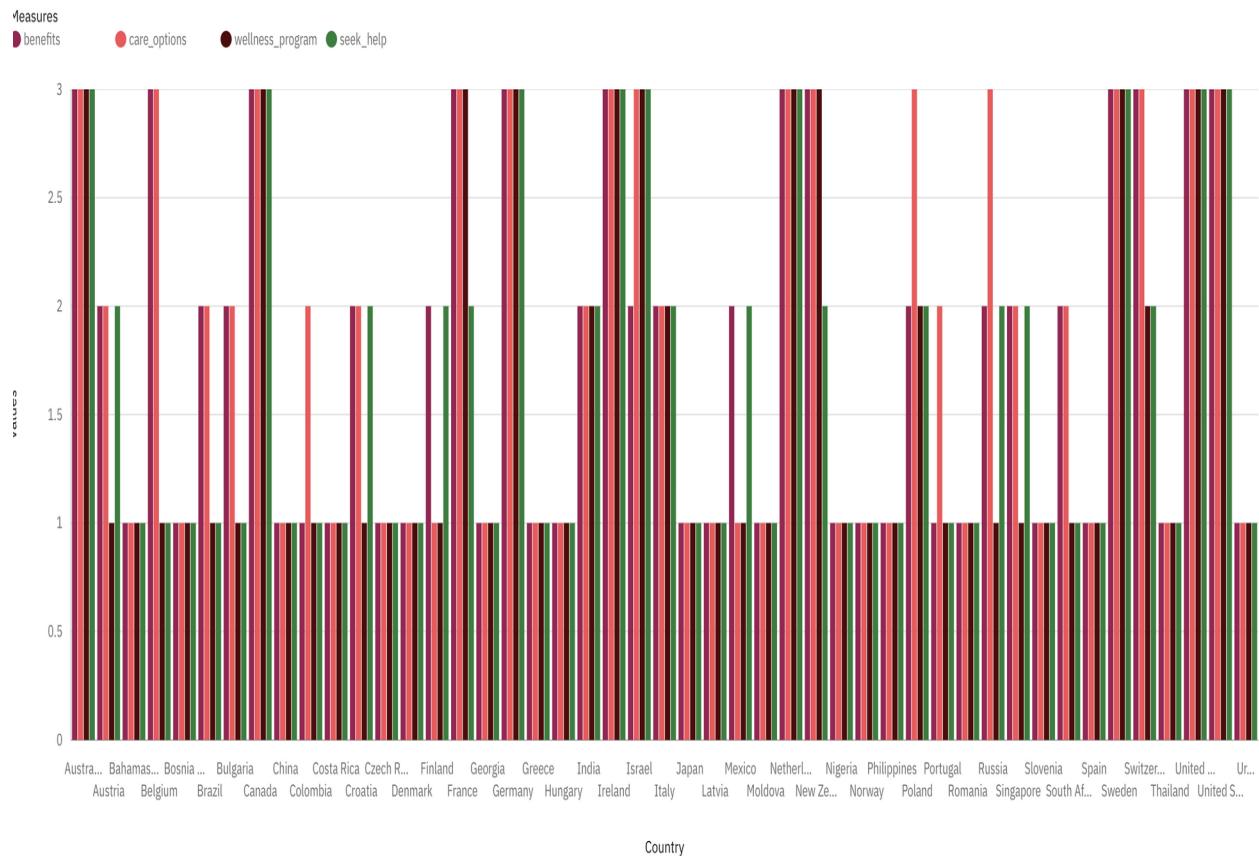
Measures
● anonymity ● leave



Country by tech_companyremote_work with points for no_employees

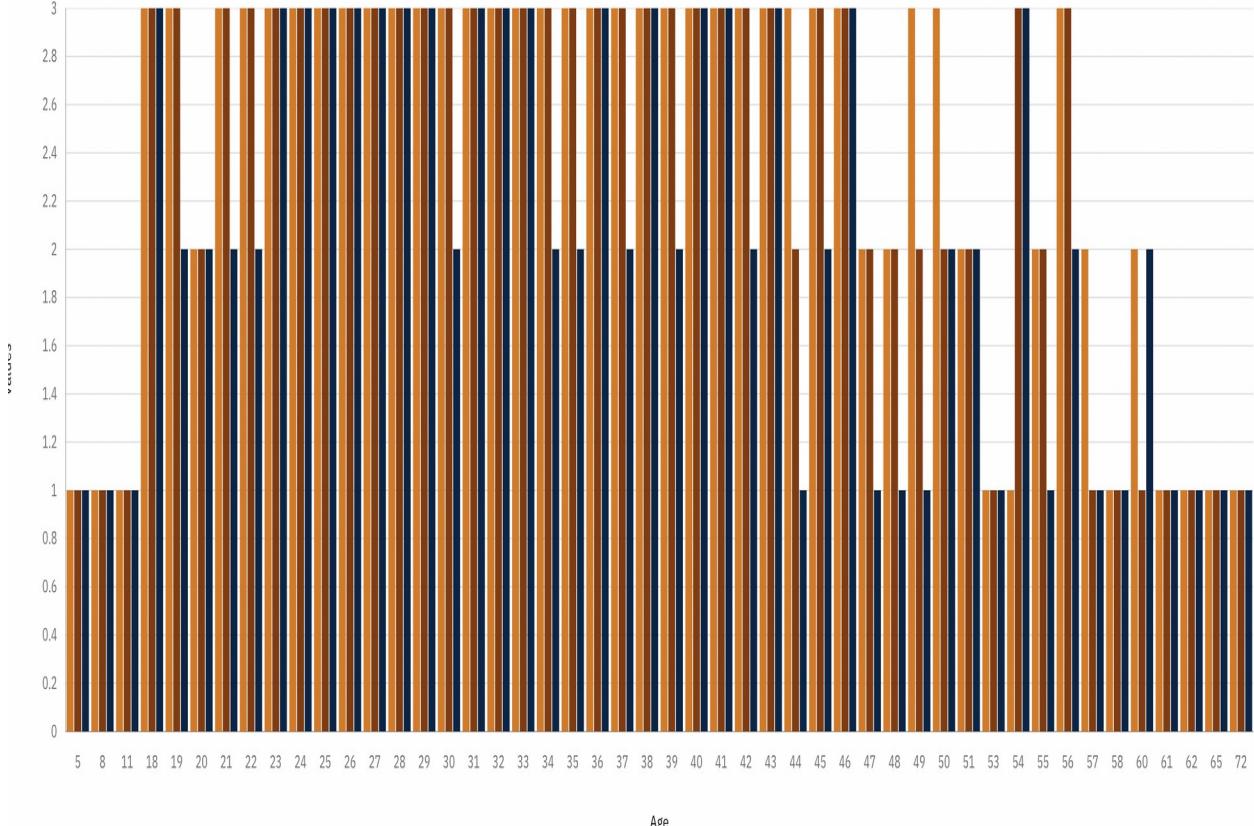


benefits, care_options, wellness_program and seek_help by Country

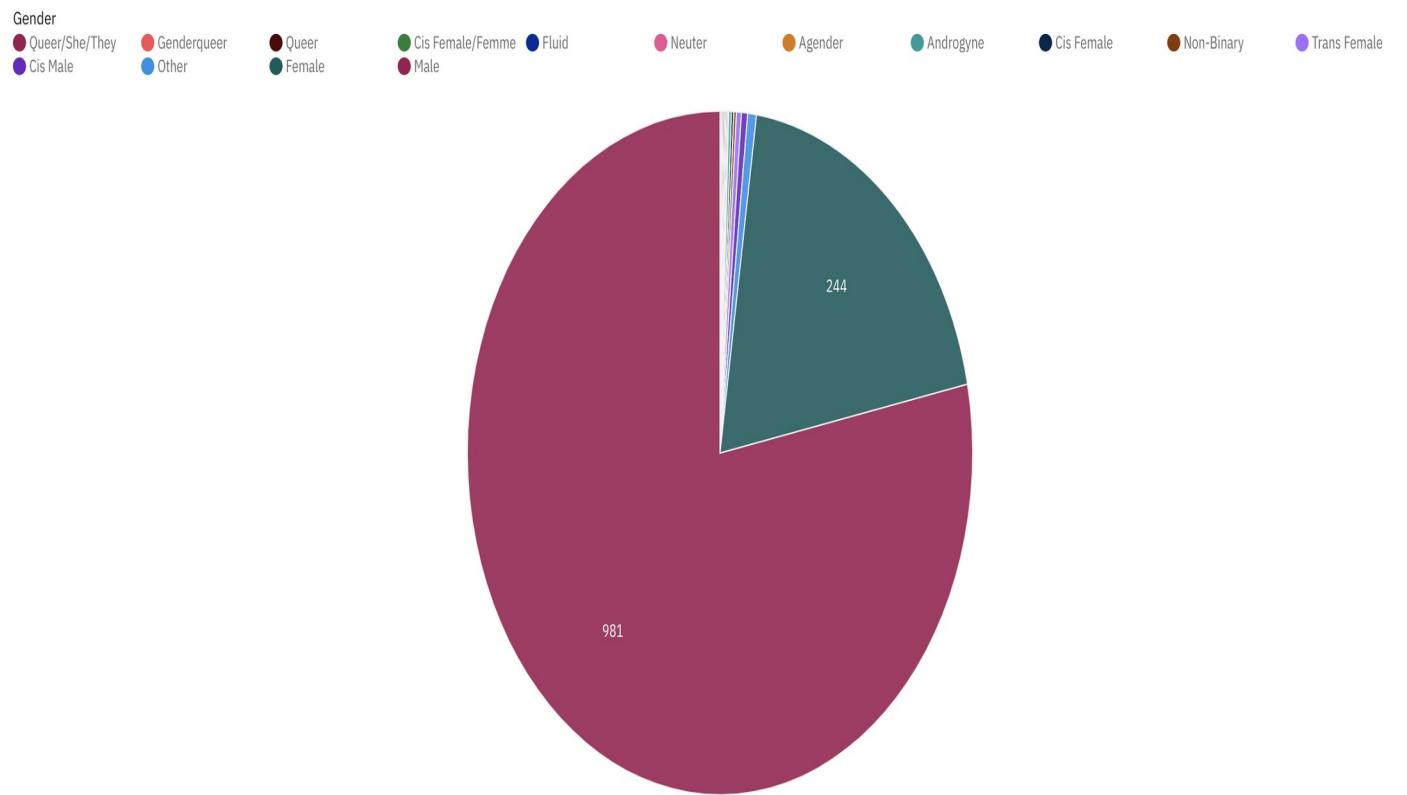


measures

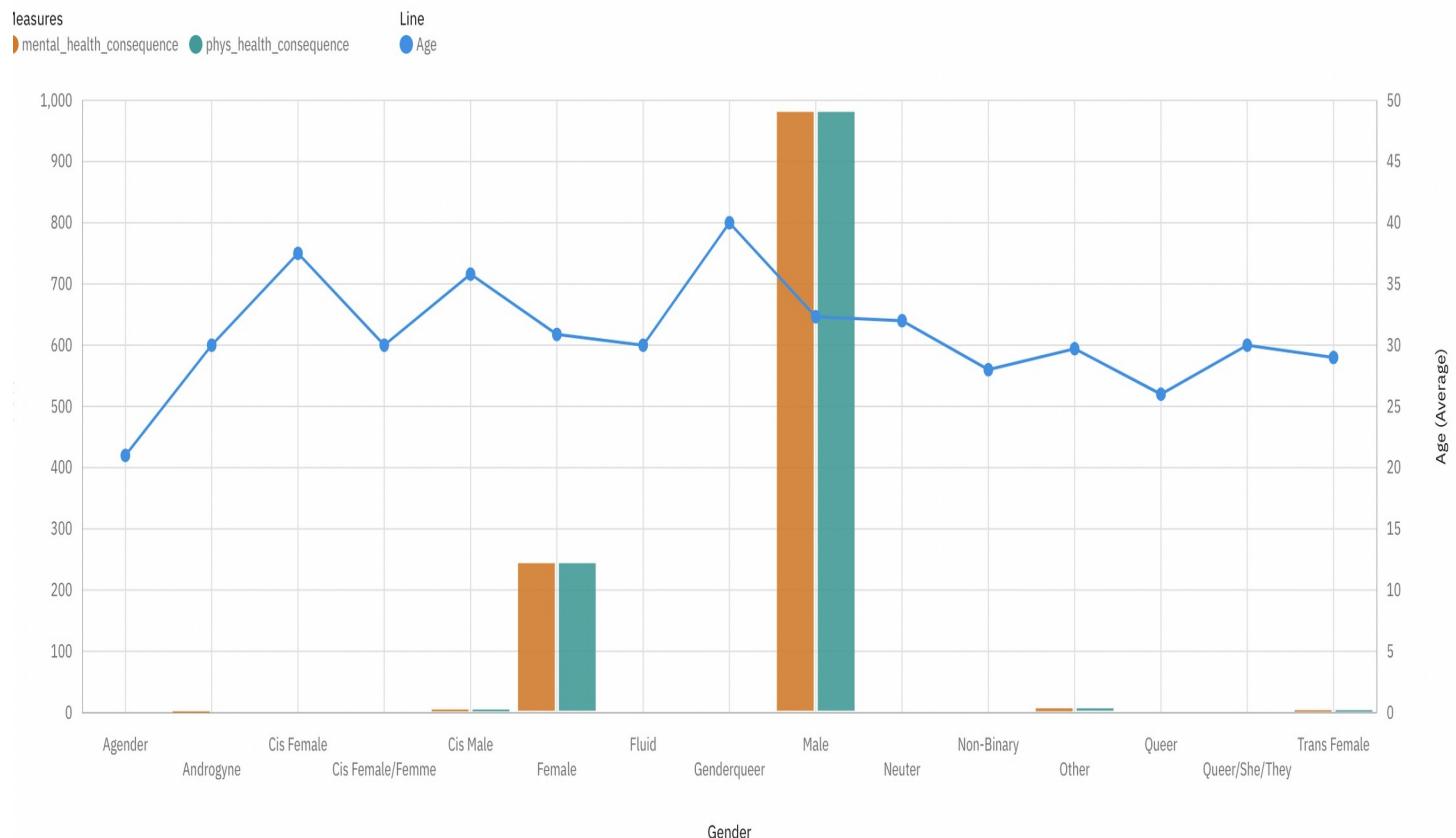
mental_health_consequence (orange), phys_health_interview (dark brown), mental_health_interview (dark blue)



Gender by Gender



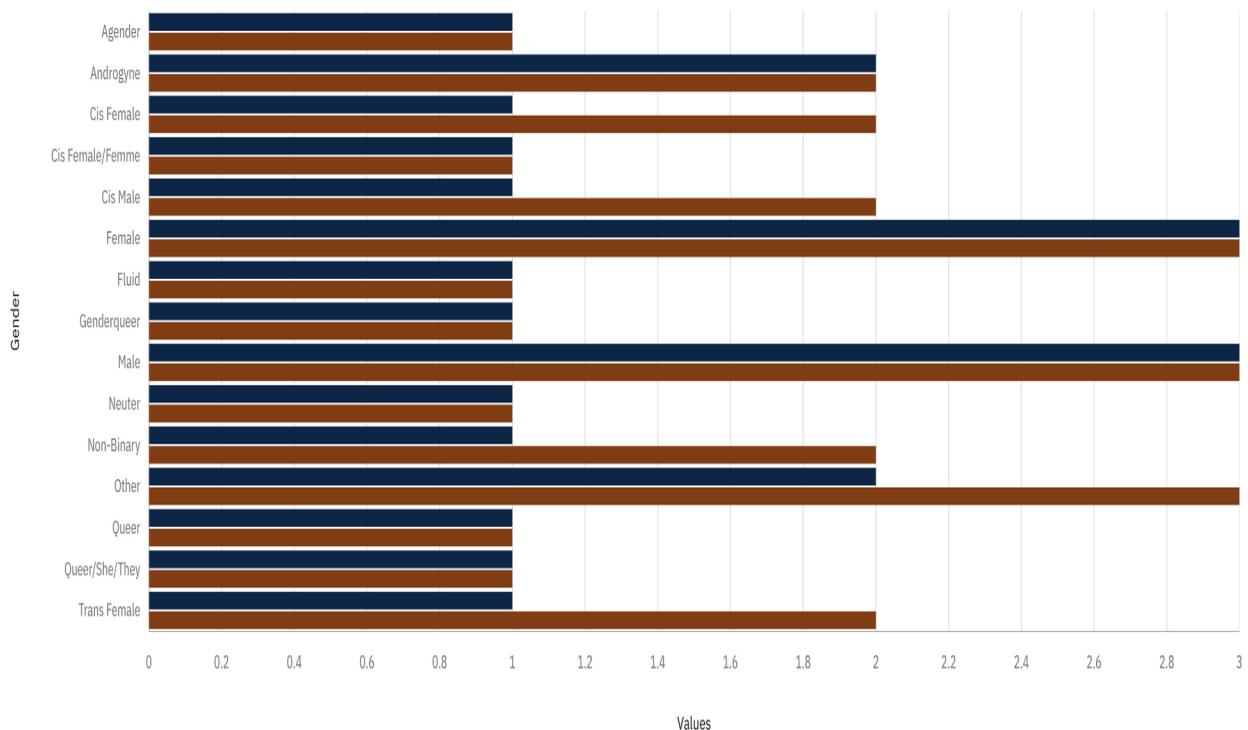
Age and mental_health_consequence and phys_health_consequence by Gender



mental_health_interview and phys_health_interview by Gender

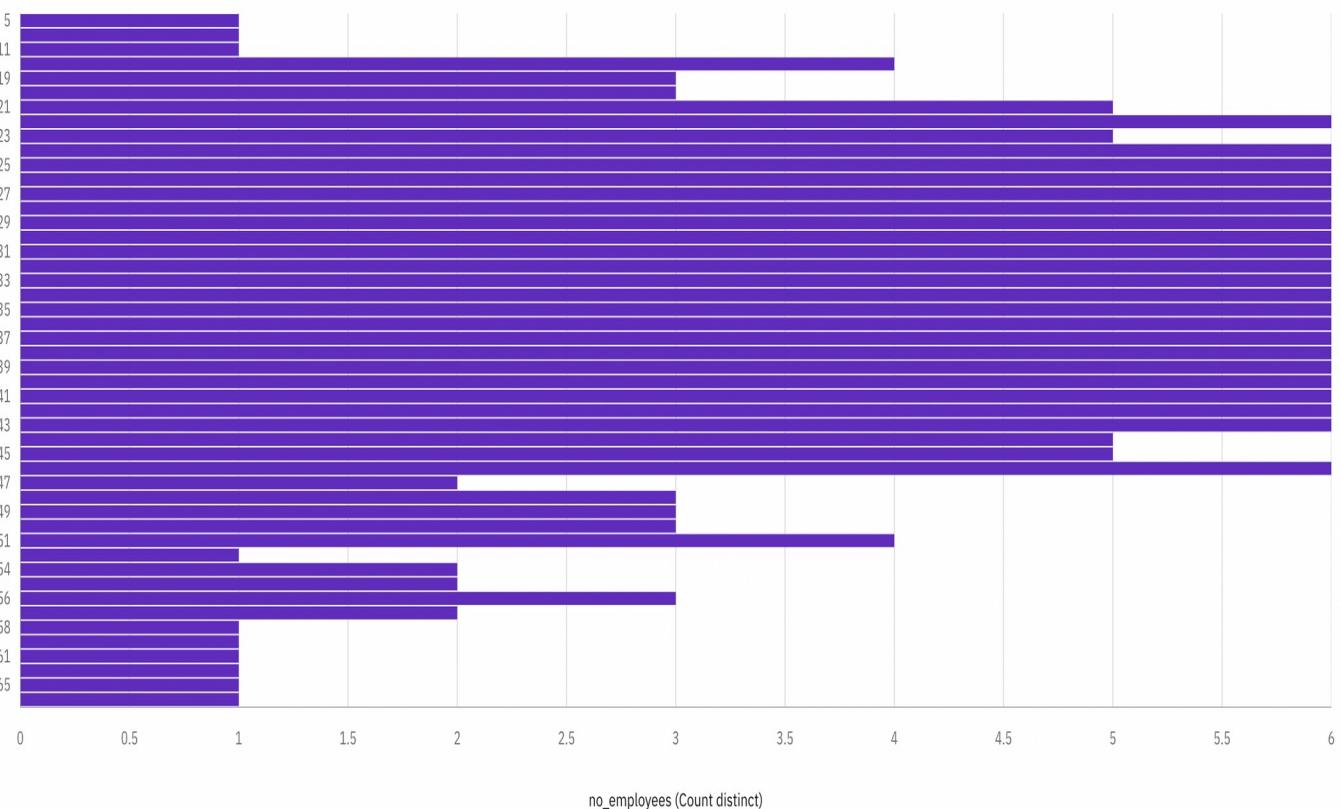
Measures

● mental_health_interview ● phys_health_interview



no_employees by Age

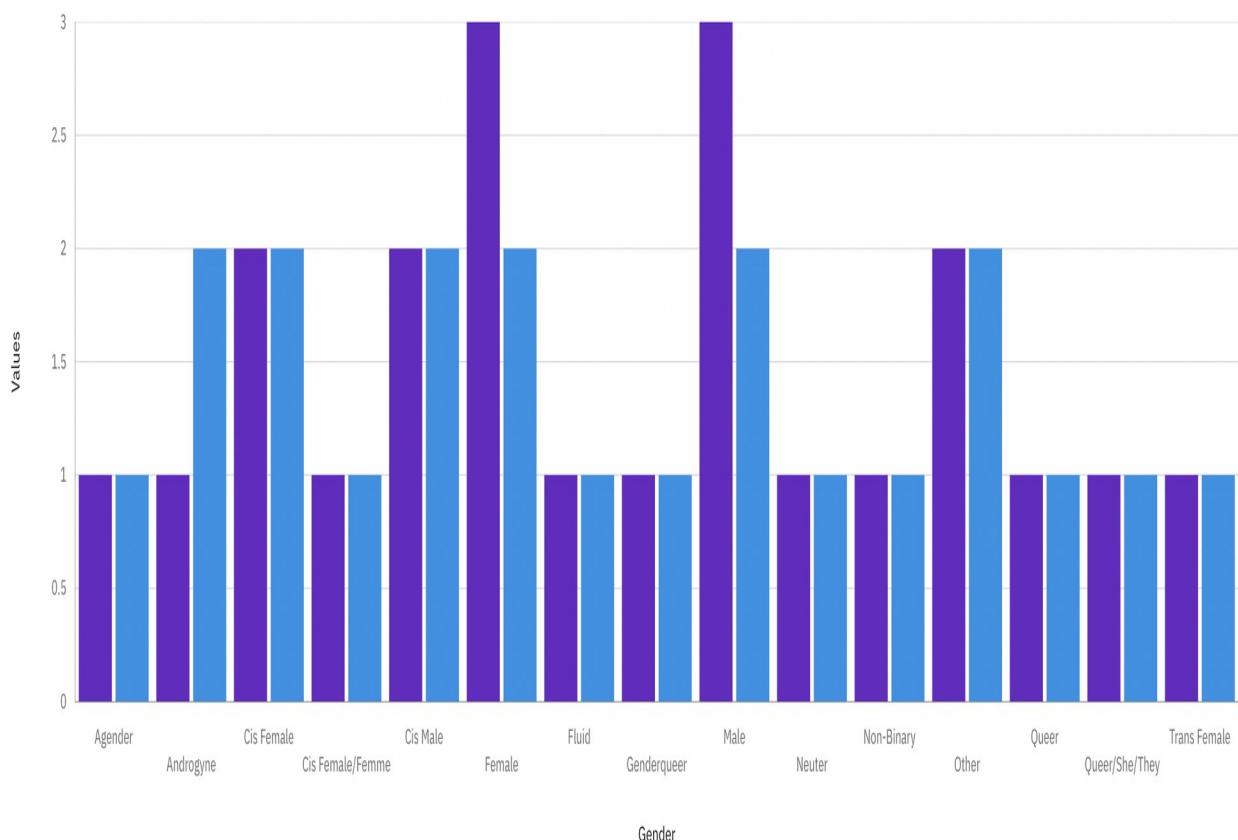
Age



self_employed and remote_work by Gender

Measures

self_employed remote_work



coworkers and supervisor by Gender

Measures

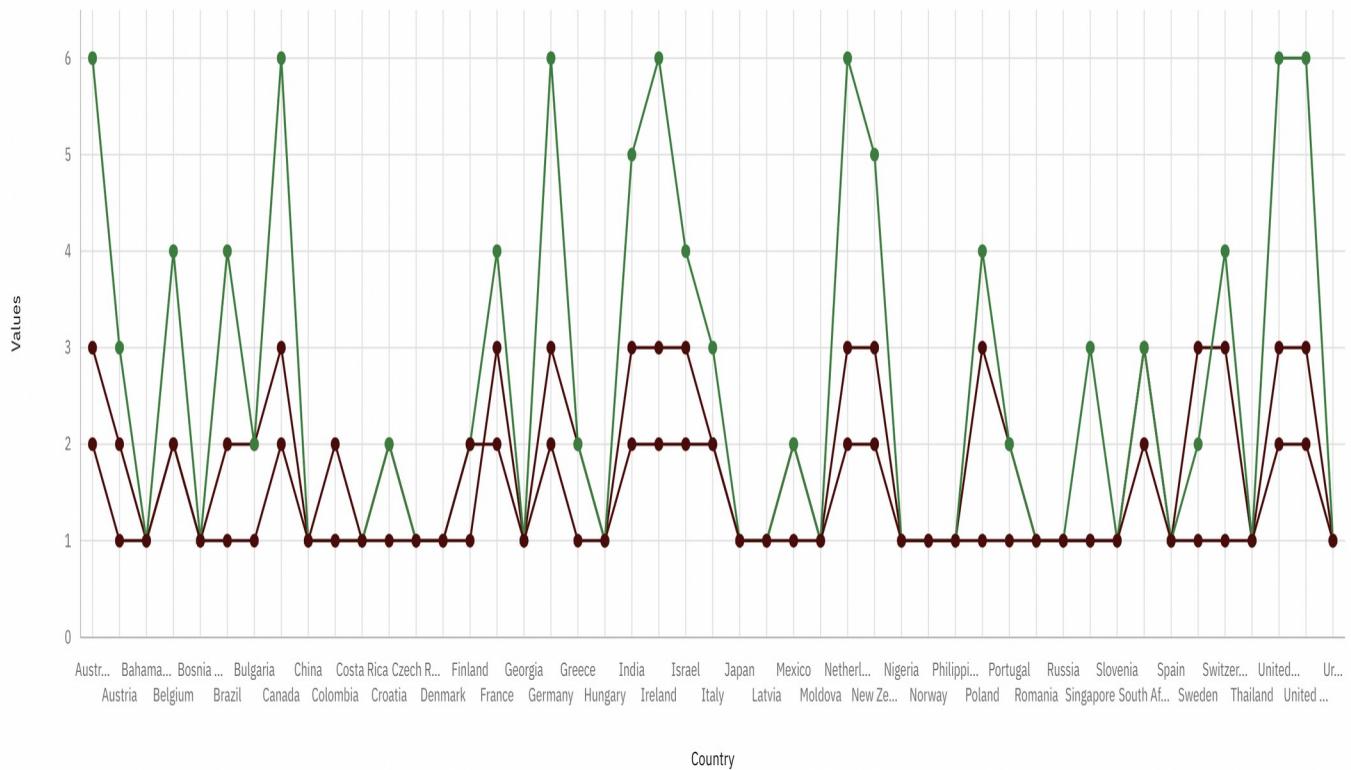
coworkers supervisor



tech_company, no_employees and mental_vs_physical by Country

Measures

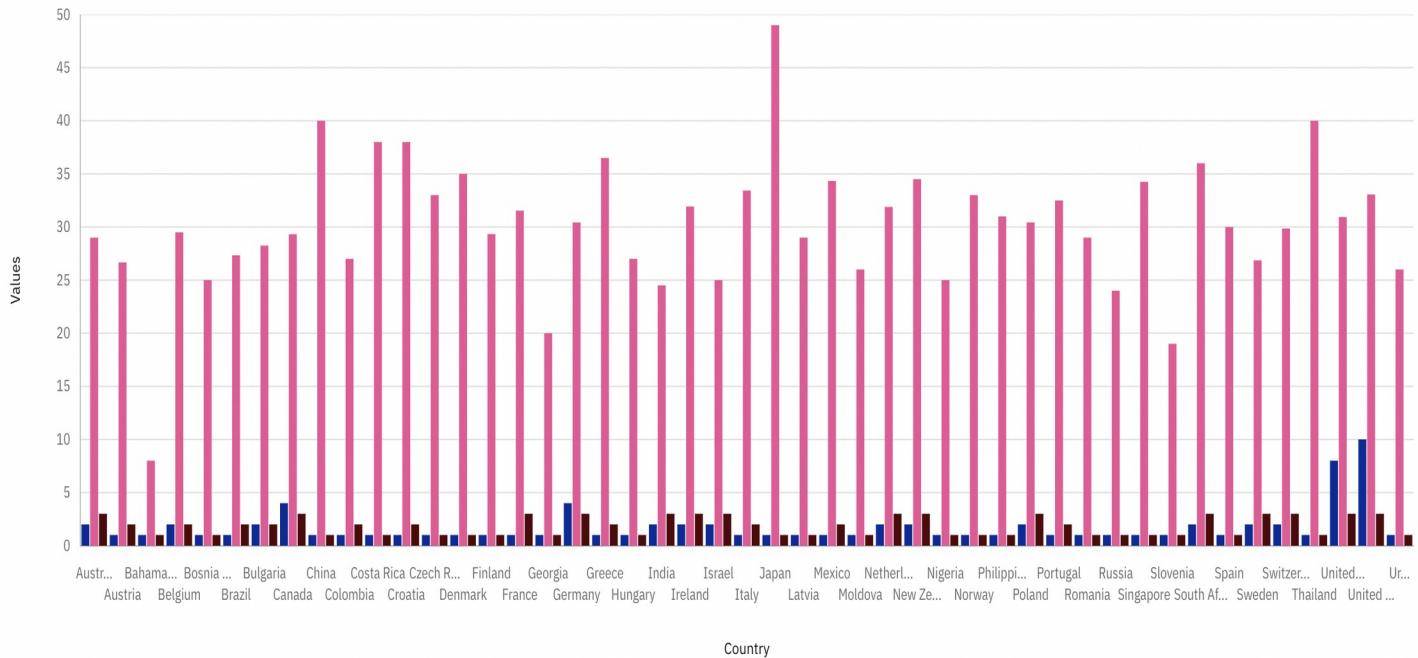
tech_company no_employees mental_vs_physical



Gender, Age and mental_vs_physical by Country

Measures

Gender Age mental_vs_physical



In []:

In []: PHASE 4

In []:

In []:

In []: Data Cleaning

In []:

```
In [12]: mh = pd.read_csv('survey.csv')  
mh
```

Out[12]:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No
...
1254	2015-09-12 11:17:21	26	male	United Kingdom	NaN	No	No	Yes
1255	2015-09-26 01:07:35	32	Male	United States	IL	No	Yes	Yes
1256	2015-11-07 12:36:58	34	male	United States	CA	No	Yes	Yes
1257	2015-11-30 21:25:06	46	f	United States	NC	No	No	No
1258	2016-02-01 23:04:31	25	Male	United States	IL	No	Yes	Yes

1259 rows × 27 columns

In [13]:

mh.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 27 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Timestamp        1259 non-null    object 
 1   Age              1259 non-null    int64  
 2   Gender            1259 non-null    object 
 3   Country           1259 non-null    object 
 4   state             744 non-null    object 
 5   self_employed     1241 non-null    object 
 6   family_history    1259 non-null    object 
 7   treatment          1259 non-null    object 
 8   work_interfere    995 non-null    object 
 9   no_employees      1259 non-null    object 
 10  remote_work       1259 non-null    object 
 11  tech_company      1259 non-null    object 
 12  benefits           1259 non-null    object 
 13  care_options      1259 non-null    object 
 14  wellness_program  1259 non-null    object 
 15  seek_help          1259 non-null    object 
 16  anonymity          1259 non-null    object 
 17  leave              1259 non-null    object 
 18  mental_health_consequence 1259 non-null    object 
 19  phys_health_consequence 1259 non-null    object 
 20  coworkers          1259 non-null    object 
 21  supervisor          1259 non-null    object 
 22  mental_health_interview 1259 non-null    object 
 23  phys_health_interview 1259 non-null    object 
 24  mental_vs_physical 1259 non-null    object 
 25  obs_consequence     1259 non-null    object 
 26  comments            164 non-null    object 

dtypes: int64(1), object(26)
memory usage: 265.7+ KB
```

```
In [14]: mh.isna().sum() / len(mh.index) * 100
```

```
Out[14]: Timestamp          0.000000
Age              0.000000
Gender            0.000000
Country           0.000000
state             40.905481
self_employed     1.429706
family_history    0.000000
treatment          0.000000
work_interfere    20.969023
no_employees       0.000000
remote_work        0.000000
tech_company       0.000000
benefits          0.000000
care_options       0.000000
wellness_program   0.000000
seek_help          0.000000
anonymity          0.000000
leave              0.000000
mental_health_consequence 0.000000
phys_health_consequence 0.000000
coworkers          0.000000
supervisor         0.000000
mental_health_interview 0.000000
phys_health_interview 0.000000
mental_vs_physical 0.000000
obs_consequence    0.000000
comments           86.973789
dtype: float64
```

```
In [15]: mh.drop(columns=['Timestamp', 'Country', 'state', 'comments'], inplace = True)
```

```
In [16]: mh.rename({'self_employed' : 'Self_Employed', 'family_history' : 'Family_History',
                 'treatment' : 'Treatment', 'work_interfere' : 'Work_Interfere',
                 'no_employees': 'Employee_Numbers', 'remote_work': 'Remote_Work',
                 'benefits': 'Benefits', 'care_options': 'Care_Options', 'wellness_program': 'Wellness_Program',
                 'seek_help': 'Seek_Help', 'anonymity': 'Anonymity', 'leave': 'Leave',
                 'mental_health_consequence': 'Mental_Health_Consequence',
                 'phys_health_consequence': 'Physical_Health_Consequence', 'coworkers': 'Coworkers',
                 'supervisor': 'Supervisor', 'mental_health_interview': 'Mental_Health_Interview',
                 'phys_health_interview': 'Physical_Health_Interview', 'mental_vs_physical': 'Mental_Physical',
                 'obs_consequence': 'Observed_Consequence'} , inplace = True , axis=1)
```

```
In [17]: mh['Age'].unique()
```

```
Out[17]: array([ 37,      44,      32,      31,      33,
                  35,      39,      42,      23,      29,
                  36,      27,      46,      41,      34,
                  30,      40,      38,      50,      24,
                  18,      28,      26,      22,      19,
                  25,      45,      21,      -29,      43,
                  56,      60,      54,      329,      55,
 99999999999,      48,      20,      57,      58,
                  47,      62,      51,      65,      49,
                 -1726,      5,      53,      61,       8,
                  11,      -1,      72])
```

```
In [18]: mh['Age'].replace([mh['Age'][mh['Age'] < 15]], np.nan, inplace = True)
mh['Age'].replace([mh['Age'][mh['Age'] > 100]], np.nan, inplace = True)

mh['Age'].unique()
```

```
Out[18]: 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., nan, 43., 56., 60., 54., 55., 48., 20., 57., 58., 47.,
62., 51., 65., 49., 53., 61., 72.])
```

```
In [19]: mh['Gender'].unique()
```

```
Out[19]: array(['Female', 'M', 'Male', 'male', 'female', 'm', 'Male-ish', 'maile',
'Trans-female', 'Cis Female', 'F', 'something kinda male?',
'Cis Male', 'Woman', 'f', 'Mal', 'Male (CIS)', 'queer/she/they',
'non-binary', 'Femake', 'woman', 'Make', 'Nah', 'All', 'Enby',
'fluid', 'Genderqueer', 'Female ', 'Androgynous', 'Agender',
'cis-female/femme', 'Guy (-ish) ^_^', 'male leaning androgynous',
'Male ', 'Man', 'Trans woman', 'msle', 'Neuter', 'Female (trans)',
'queer', 'Female (cis)', 'Mail', 'cis male', 'A little about you',
'Malr', 'p', 'femail', 'Cis Man',
'ostensibly male, unsure what that really means'], dtype=object)
```

```
In [20]: mh['Gender'].replace(['Male ', 'male', 'M', 'm', 'Male', 'Cis Male',
                           'Man', 'cis male', 'Mail', 'Male-ish', 'Male (CIS)',
                           'Cis Man', 'msle', 'Malr', 'Mal', 'maile', 'Make'],
                           mh['Gender'].replace(['Female ', 'female', 'F', 'f', 'Woman',
                           'Female', 'femail', 'Cis Female', 'cis-female/femme',
                           'Femake', 'woman'], 'Female', inplace = True)

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

```
In [21]: mh['Gender'].value_counts()
```

```
Out[21]: Male      991
Female     247
Queer      21
Name: Gender, dtype: int64
```

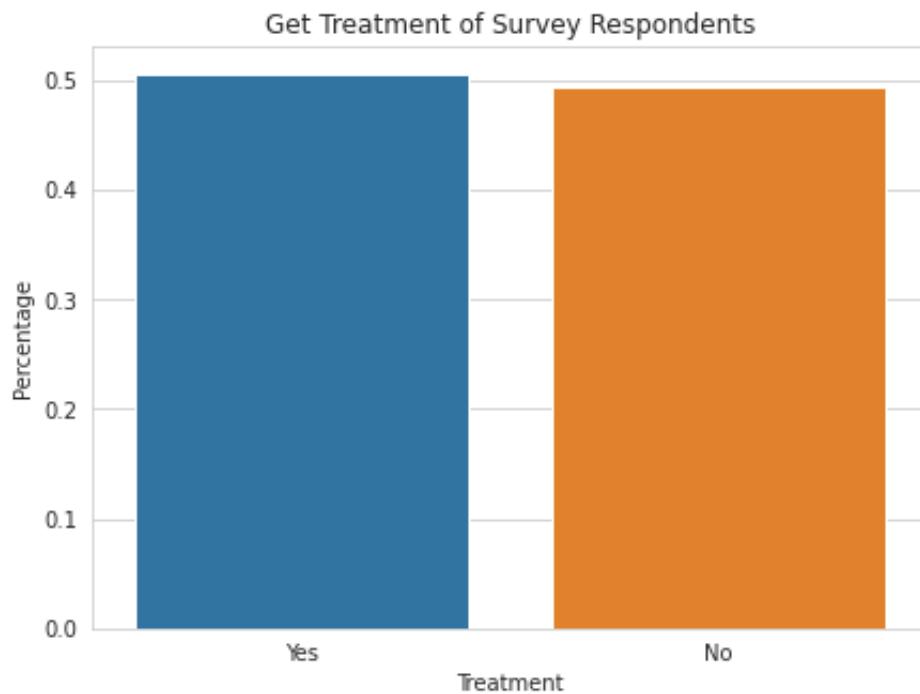
```
In [ ]:
```

```
In [ ]: EDA
```

```
In [ ]:
```

```
In [22]: mh_eda = mh.copy()
```

```
In [23]: sns.set_style("whitegrid")
plt.figure(figsize = (7,5))
eda_percentage = mh_eda['Treatment'].value_counts(normalize = True).rename_
sns.barplot(x = 'Treatment', y = 'Percentage', data = eda_percentage.head())
plt.title('Get Treatment of Survey Respondents')
plt.show()
```

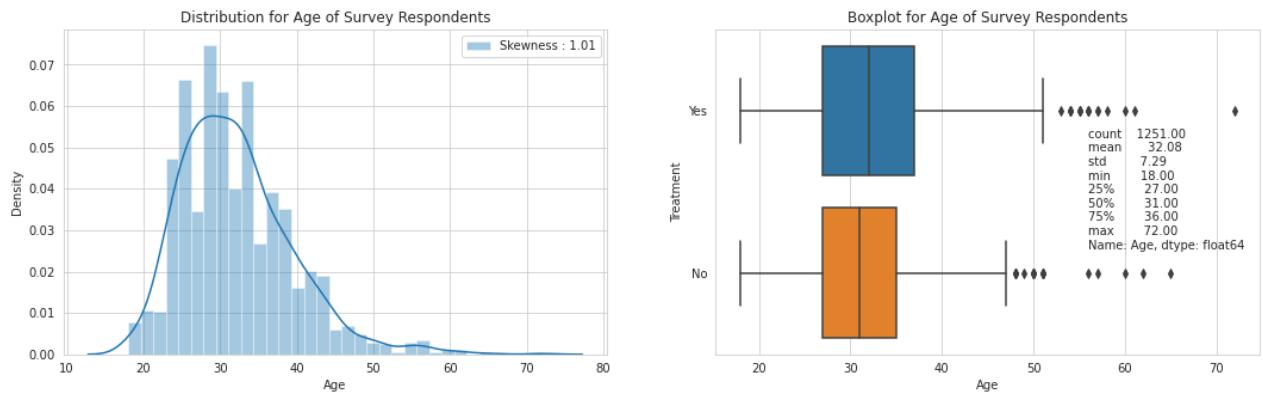


```
In [ ]:
```

In []: This is the respondents result of question, 'Have you get treatment for a reason?'. The percentage of respondents who want to get treatment is 50%. Workplaces with people with mental disorders are more likely to reduce absenteeism, increasing economic gains. If employees enjoy good mental health, employees can:
make the most of your potential,
cope with what life throws at you,
play a full part in your relationships, your workplace, and your community.
I decided to separate them into 3 aspects to see what factors that company Employee's profiling
Employee's work environment
Employee's mental health facilities

```
In [ ]:
```

```
In [24]: plt.figure(figsize = (18,5))
plt.subplot(1,2,1)
sns.distplot(mh_eda['Age'], label = 'Skewness : %.2f' %(mh_eda['Age'].skew()))
plt.legend(loc = 0, fontsize = 10)
plt.title('Distribution for Age of Survey Respondents')
plt.subplot(1,2,2)
sns.boxplot(x = "Age", y = "Treatment", data = mh_eda)
plt.title('Boxplot for Age of Survey Respondents')
age = str(mh_eda['Age'].describe().round(2))
plt.text(56, 0.85, age)
plt.show()
```



```
In [ ]:
```

Skewness

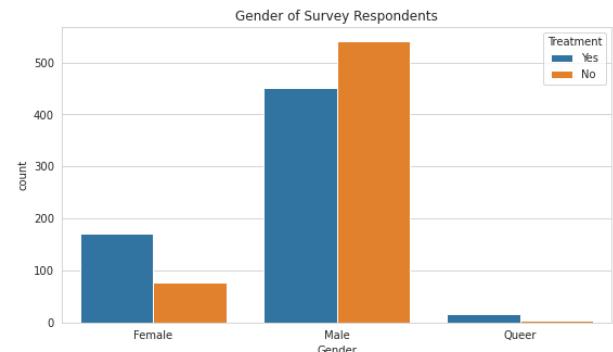
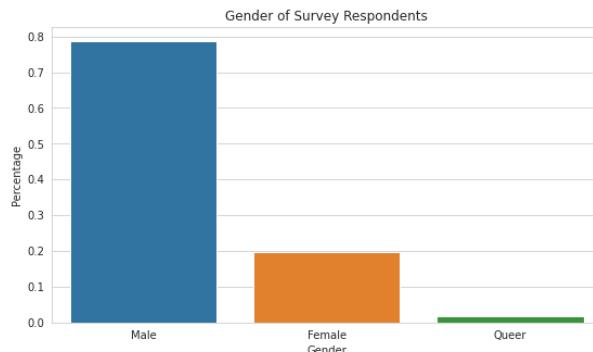
Based on the plot, the skewness score is 1.01, which means the data are highly skewed to the right, with most values smaller than mean or median.

It's indicated that most of the employees that fill the survey around the mid to senior-level positions. The distribution of ages is right-skewed while younger employees. From an article that I read, young (usually white, mostly male) tech workers like Zuckerberg and other "tech bros" have become the symbol and stereotypical representation of the industry.

From the boxplot, there is no statistically significant difference of ages between the two treatment groups.

```
In [ ]:
```

```
In [25]: plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
eda_percentage = mh_eda['Gender'].value_counts(normalize = True).rename_axis('Gender').reset_index()
sns.barplot(x = 'Gender', y = 'Percentage', data = eda_percentage.head(10))
plt.title('Gender of Survey Respondents')
plt.subplot(1,2,2)
sns.countplot(mh_eda['Gender'], hue = mh_eda['Treatment'])
plt.title('Gender of Survey Respondents')
plt.show()
```

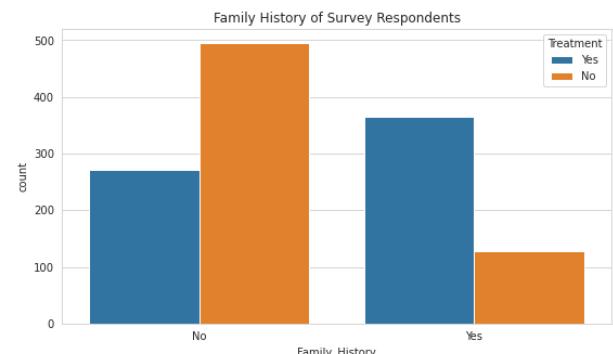
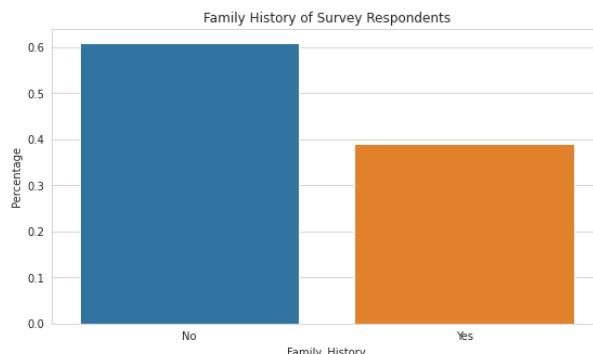


In []:

This is the respondents result of question, 'What is your gender identities'. Almost 79% of respondents are male, not surprisingly, especially in the tech industry where there is higher competitive pressure for women than men. Based on the plot, female respondents are more likely than men to report experiencing sexual harassment or racism at work because female are scarce in the tech industry. There is a Queer entry of less than 2%. Although the percentage of queer is low, it is still significant in terms of mental health insights. For example, such a small proportion can show a significant difference in mental health problems between genders, indicating that for the queer, mental health problems are serious too. In conclusion, there is a significant difference in mental health between genders, indicating that there is discrimination in the workplace.

In []:

```
plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
eda_percentage = mh_eda['Family_History'].value_counts(normalize = True).reset_index()
sns.barplot(x = 'Family_History', y = 'Percentage', data = eda_percentage)
plt.title('Family History of Survey Respondents')
plt.subplot(1,2,2)
sns.countplot(mh_eda['Family_History'], hue = mh_eda['Treatment'])
plt.title('Family History of Survey Respondents')
plt.show()
```

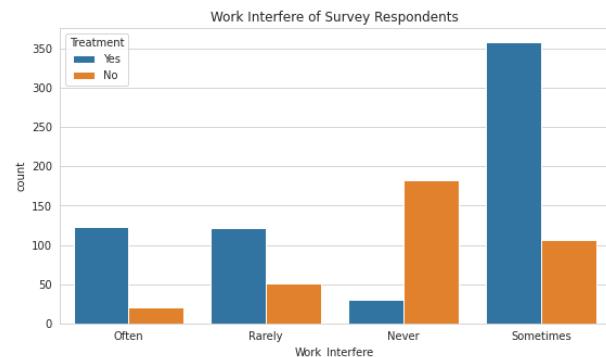
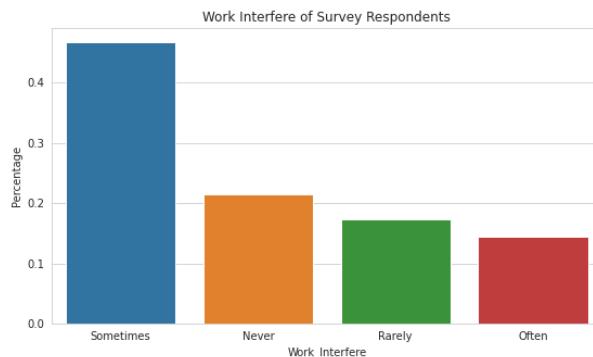


In []:

This is the respondents result of question, 'Do you have a family history of mental illness?'. From 40% of respondents who say that they have a family history of mental illness, 60% want to get treatment rather than without a family history. This is acceptable because people with a family history pay more attention to mental illness. Family history is a significant factor in mental health. The apple does not fall far from the tree, as it is relatively common for people with a family history to have more relatives with histories of similar difficulties.

In []:

```
In [27]: plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
eda_percentage = mh_eda['Work_Interfere'].value_counts(normalize = True).re
sns.barplot(x = 'Work_Interfere', y = 'Percentage', data = eda_percentage)
plt.title('Work Interfere of Survey Respondents')
plt.subplot(1,2,2)
sns.countplot(mh_eda['Work_Interfere'], hue = mh_eda['Treatment'])
plt.title('Work Interfere of Survey Respondents')
plt.show()
```

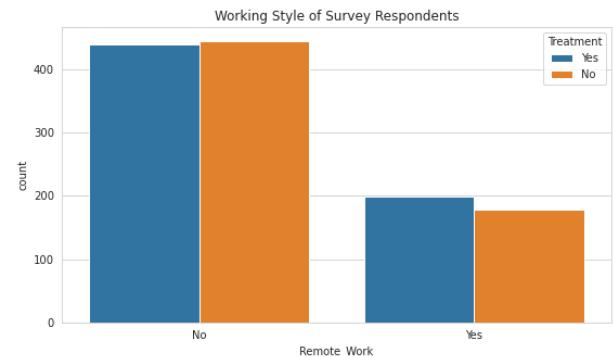
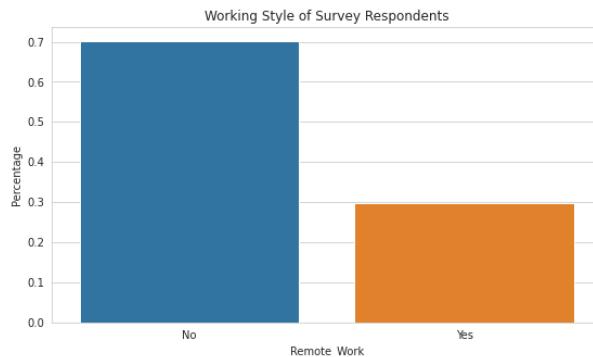


In []:

In []: This is the respondents result of question, 'If you have a mental health condition, does it interfere with your work?'. About 78% of respondents have experienced interference at work with a ratio of 3:1. Mental health conditions sometimes become an interfere while working about 4 times more often than never. Even though most respondents want to get treatment. But it's surprising to know even mental health never group that still want to get treatment before it become a job stress. It can happen that a job do not match the capabilities, resources or needs of the worker.

In []:

```
In [28]: plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
eda_percentage = mh_eda['Remote_Work'].value_counts(normalize = True).re
sns.barplot(x = 'Remote_Work', y = 'Percentage', data = eda_percentage)
plt.title('Working Style of Survey Respondents')
plt.subplot(1,2,2)
sns.countplot(mh_eda['Remote_Work'], hue = mh_eda['Treatment'])
plt.title('Working Style of Survey Respondents')
plt.show()
```

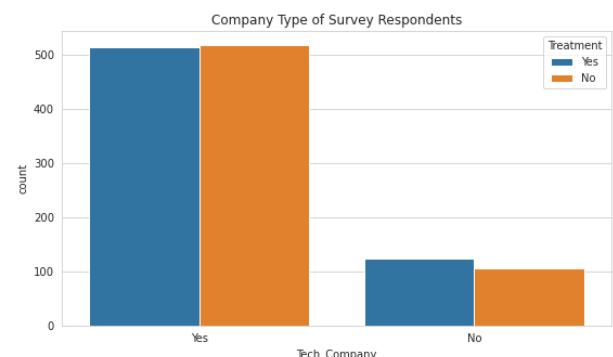
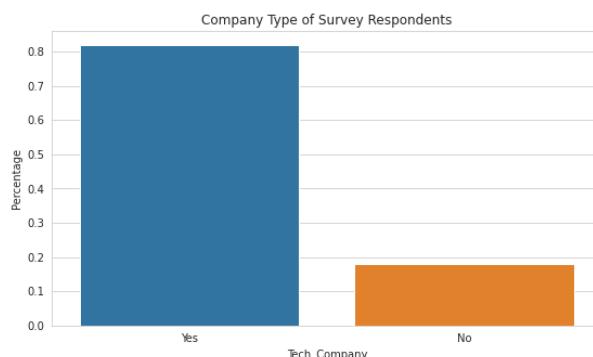


In []:

In []: This is the respondents result of question, Do you work remotely (outside of the office). Around 70% of respondents don't work remotely, which means the biggest factor is the workplace. On the other side, it has slightly different between an employee who want to get a treatment. But it's getting interesting when we see a respondent who want to get treatment is a little bit higher. I have no idea why those doesn't provide that information.

In []:

```
In [29]: plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
eda_percentage = mh_eda['Tech_Company'].value_counts(normalize = True).rena
sns.barplot(x = 'Tech_Company', y = 'Percentage', data = eda_percentage)
plt.title('Company Type of Survey Respondents')
plt.subplot(1,2,2)
sns.countplot(mh_eda['Tech_Company'], hue = mh_eda['Treatment'])
plt.title('Company Type of Survey Respondents')
plt.show()
```

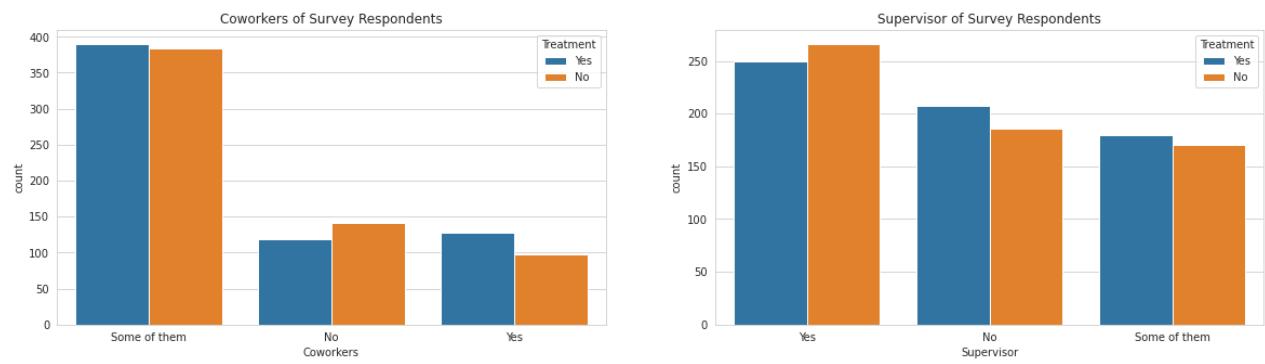


In []:

In []: This is the respondents result of question, 'Is your employer primarily a technology company?'. Even the main target of the survey is the tech field, there are 18% of companies that can be seen from the plot whether the company belongs to the tech field or not. I think the environment affects a lot of employees and some of them can't work in the tech field. However, I found that the number of employees in the technology field that receive treatment. But the non-technical field is the opposite. Maybe the non-tech

In []:

```
In [30]: plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
sns.countplot(mh_eda['Coworkers'], hue = mh_eda['Treatment'])
plt.title('Coworkers of Survey Respondents')
plt.subplot(1,2,2)
sns.countplot(mh_eda['Supervisor'], hue = mh_eda['Treatment'])
plt.title('Supervisor of Survey Respondents')
plt.show()
```

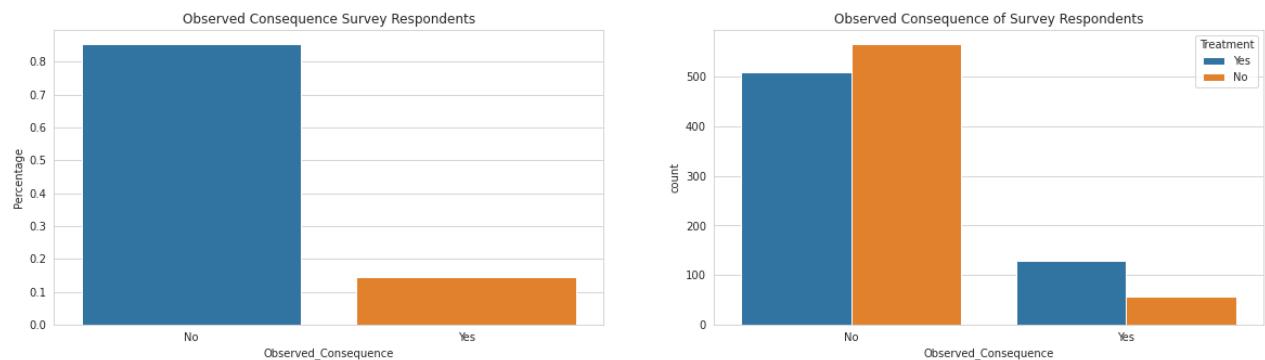


In []:

In []: This is the respondents result of question, 'Would you be willing to discuss it with coworkers?'. From 18% of respondents who say yes to discuss it with coworkers, 60% of them say no. About 60% of respondents decide to discuss some of them with coworkers. Employees who say yes to discuss it with coworkers are more likely to say no. Let's see if the respondent will discuss it with a supervisor or not. This is the respondents result of question, 'Would you be willing to discuss it with supervisor?'. From 40% of respondents who say yes to discuss with supervisor, only 55% of them say no. Discussing with someone in a higher position could help the relief. It's the opposite when they say no.

In []:

```
In [31]: plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
eda_percentage = mh_eda['Observed_Consequence'].value_counts(normalize = True)
sns.barplot(x = 'Observed_Consequence', y = 'Percentage', data = eda_percentage)
plt.title('Observed Consequence Survey Respondents')
plt.subplot(1,2,2)
sns.countplot(mh_eda['Observed_Consequence'], hue = mh_eda['Treatment'])
plt.title('Observed Consequence of Survey Respondents')
plt.show()
```

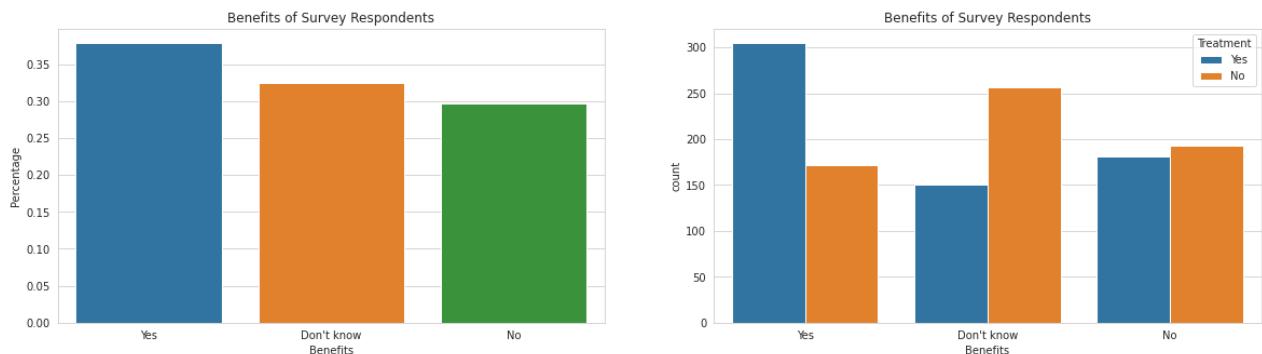


In []:

In []: This is the respondents result of question, 'Have you heard of or observed conditions in your workplace?'. From 15% of respondents who say yes about knowing the negative consequences almost 70% of them that want to get treatment. After the employee knows about trigger for someone to get treatment to prevent mental health conditions.

In []:

```
In [32]: plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
eda_percentage = mh_eda['Benefits'].value_counts(normalize = True).rename_
sns.barplot(x = 'Benefits', y = 'Percentage', data = eda_percentage)
plt.title('Benefits of Survey Respondents')
plt.subplot(1,2,2)
sns.countplot(mh_eda['Benefits'], hue = mh_eda['Treatment'])
plt.title('Benefits of Survey Respondents')
plt.show()
```

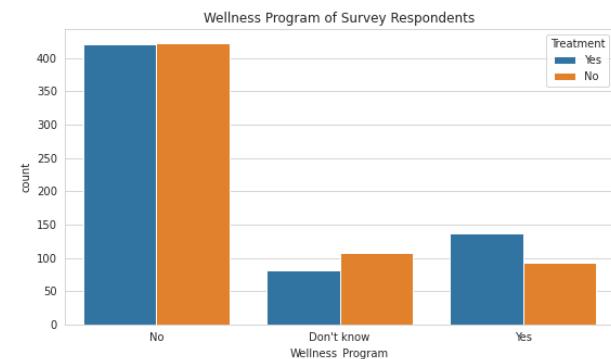
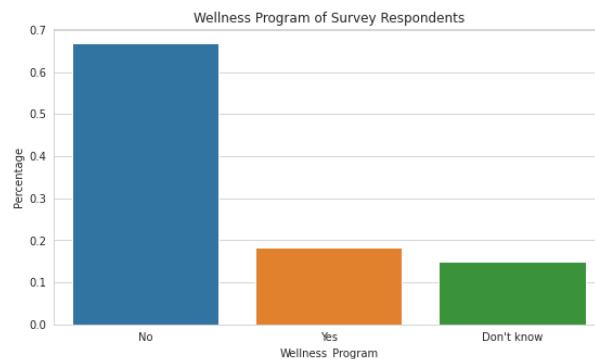


In []:

In []: This is the respondents result of question, 'Does your employer provide mental health benefits?'. Only 35% of respondents know about mental health benefits that the company provides. For employees who know the benefits, almost 60% of the employees want to get treatment. For those who don't know and says that the company doesn't provide still want to get treatment. This means that even if the company doesn't provide it properly because of budgeting or financial struggling.

In []:

```
In [33]: plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
eda_percentage = mh_eda['Wellness_Program'].value_counts(normalize = True)
sns.barplot(x = 'Wellness_Program', y = 'Percentage', data = eda_percentage)
plt.title('Wellness Program of Survey Respondents')
plt.subplot(1,2,2)
sns.countplot(mh_eda['Wellness_Program'], hue = mh_eda['Treatment'])
plt.title('Wellness Program of Survey Respondents')
plt.show()
```



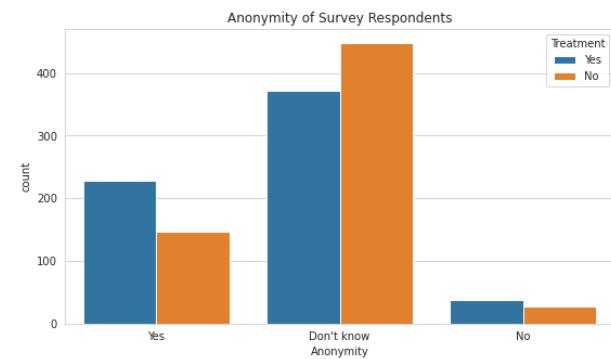
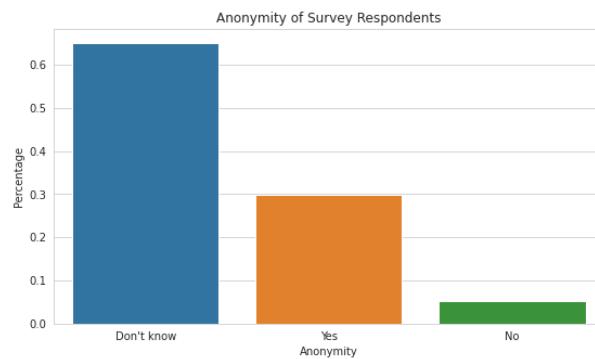
In []:

In []: This is the respondents result of question, 'Has your employer ever discussed about your health or wellness?'. About 19% of the respondents say yes about becoming a part of employee wellness program. I assume that employee feels a good vibe. More than 65% of respondents say that there aren't any wellness programs that they have heard of. I think that most of the respondents want to get treatment, which means the company need to provide a benefit before, I think it makes sense if it's about company budgeting. I think that the company has a lot of employees to take care of. My second thought, it's something that will be a lot of struggle.

In []:

In [34]:

```
plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
eda_percentage = mh_eda['Anonymity'].value_counts(normalize = True).rename_
sns.barplot(x = 'Anonymity', y = 'Percentage', data = eda_percentage)
plt.title('Anonymity of Survey Respondents')
plt.subplot(1,2,2)
sns.countplot(mh_eda['Anonymity'], hue = mh_eda['Treatment'])
plt.title('Anonymity of Survey Respondents')
plt.show()
```



In []:

In []: This is the respondents result of question, 'Is your anonymity protected if you are seeking substance abuse treatment resources?'. About 30% of respondents say yes if their anonymity is protected while taking treatment resources and almost 65% of employees want to get treatment. The privacy and it's a good move for the company to build trust with their employees to be better.

In []:

In []:

In []: Conclusion

Nearly 86% of employees report improved work performance **and** lower rates of depression according to an April 2018 article **in** the Journal of Occupational **and** Environmental Psychology. **And** productivity **for** employers. By providing employees access to mental health resources, companies can help employees develop a better understanding **and** compassion at the tech company. And having employees who

Based on profiling the respondents

Companies must know that gender and family history greatly influence the decision to provide more support, the company must make an assessment of the employees' different needs. Age can also be a trigger, considering that most of them are open-minded to get treatment.

Based on the work environment of respondents

Work interference **is** the most influential of employees who want to get treated providing facilities to anticipate job stress on employees. Some of the companies case employees suddenly feel stress **and** need a private moment to relieve.

Based on the mental health facilities of respondents

The company needs to provide a good benefit **for** employees so they can maintain resources **for** it, there are so many third parties who can be hired to Building trust like keep private about whom employee that gets treatment a

In []:

In []:

PreProcessing

Preprocessing Scheme

OneHotEncoding: Gender, Family History, Employee Numbers, Remote Work, Tech Seek Help, Anonymity, Medical Leave, Mental Health Consequence, Physical

Mental Health Interview, Physical Health Interview, Mental vs Physical Simple Imputer Most Frequent: Self Employed, Work Interfere

Simple Imputer Most Frequent: Self_Employed, WORK_INDEP
Iterative Impute: Age

Target: Treatment

Target: Treatment

In []:

```
In [35]: mode_onehot_pipe = Pipeline([
    ('encoder', SimpleImputer(strategy = 'most_frequent')),
    ('one hot encoder', OneHotEncoder(handle_unknown = 'ignore'))])

transformer = ColumnTransformer([
    ('one hot', OneHotEncoder(handle_unknown = 'ignore'), ['Gender', 'Family_Size',
        'Remote_Work', 'Wellness_Program', 'Medical_Leave', 'Physical_Health',
        'Mental_Health', 'Mental_VS_Physical_Health']),
    ('mode_onehot_pipe', mode_onehot_pipe, ['Self_Employed', 'Work_Interferes'],
    ('iterative', IterativeImputer(max_iter = 10, random_state = 0), ['Age']])])
```

```
In [36]: mh['Treatment'].value_counts()/mh.shape[0]*100
```

```
Out[36]: Yes      50.595711
No       49.404289
Name: Treatment, dtype: float64
```

```
In [37]: mh['Treatment'] = np.where(mh['Treatment'] == 'Yes', 1, 0)
```

```
In [38]: X = mh.drop('Treatment', axis = 1)
y = mh['Treatment']
```

```
In [39]: X.shape
```

```
Out[39]: (1259, 22)
```

```
In [40]: X_train, X_test, y_train, y_test = train_test_split(X,y,
                                                       stratify = y,
                                                       test_size = 0.3,
                                                       random_state = 2222)
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]: Modeling
Define Model

In supervised learning, algorithms learn from labeled data. In this case, class is yes and no.
I use 3 basic models and 4 ensemble models to predict.
Basic models:
Logistic Regression (logreg)
Decision Tree Classifier (tree)
K-Nearest Neighbor (knn)
Ensemble models:
Random Forest Classifier (rf)
Ada Boost Classifier (ada)
Gradient Boosting Classifier (grad)
```

In []:

In []:

In [48]:

```
logreg = LogisticRegression()
tree = DecisionTreeClassifier(random_state = 2222)
knn = KNeighborsClassifier()
rf = RandomForestClassifier(random_state = 2222)
ada = AdaBoostClassifier(random_state = 2222)
grad = GradientBoostingClassifier(random_state = 2222)
```

In []:

In []: Cross Validation:

In []:

In [52]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

# PreProcessing
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import SimpleImputer, IterativeImputer
from sklearn.preprocessing import MinMaxScaler

# Splitting Data
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score

# Modeling
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, recall_score, plot_roc_curve
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, VotingClassifier

logreg_pipe = Pipeline([('transformer', transformer), ('logreg', logreg)])
tree_pipe = Pipeline([('transformer', transformer), ('tree', tree)])
knn_pipe = Pipeline([('transformer', transformer), ('knn', knn)])
rf_pipe = Pipeline([('transformer', transformer), ('rf', rf)])
ada_pipe = Pipeline([('transformer', transformer), ('ada', ada)])
grad_pipe = Pipeline([('transformer', transformer), ('grad', grad)])

def model_evaluation(model, metric):
    model_cv = cross_val_score(model, X_train, y_train, cv=StratifiedKFold(5))
    return model_cv
```

```

logreg_pipe_cv = model_evaluation(logreg_pipe, 'recall')
tree_pipe_cv = model_evaluation(tree_pipe, 'recall')
knn_pipe_cv = model_evaluation(knn_pipe, 'recall')
rf_pipe_cv = model_evaluation(rf_pipe, 'recall')
ada_pipe_cv = model_evaluation(ada_pipe, 'recall')
grad_pipe_cv = model_evaluation(grad_pipe, 'recall')

for model in [logreg_pipe, tree_pipe, knn_pipe, rf_pipe, ada_pipe, grad_pipe]:
    model.fit(X_train, y_train)

score_cv = [logreg_pipe_cv.round(5), tree_pipe_cv.round(5), knn_pipe_cv.round(5),
           rf_pipe_cv.round(5), ada_pipe_cv.round(5), grad_pipe_cv.round(5)]
score_mean = [logreg_pipe_cv.mean(), tree_pipe_cv.mean(), knn_pipe_cv.mean(),
              ada_pipe_cv.mean(), grad_pipe_cv.mean()]
score_std = [logreg_pipe_cv.std(), tree_pipe_cv.std(), knn_pipe_cv.std(),
             ada_pipe_cv.std(), grad_pipe_cv.std()]
score_recall_score = [recall_score(y_test, logreg_pipe.predict(X_test)),
                      recall_score(y_test, tree_pipe.predict(X_test)),
                      recall_score(y_test, knn_pipe.predict(X_test)),
                      recall_score(y_test, rf_pipe.predict(X_test)),
                      recall_score(y_test, ada_pipe.predict(X_test)),
                      recall_score(y_test, grad_pipe.predict(X_test))]
]

method_name = ['Logistic Regression', 'Decision Tree Classifier', 'KNN Classifier',
               'Ada Boost Classifier', 'Gradient Boosting Classifier']

cv_summary = pd.DataFrame({
    'method': method_name,
    'cv score': score_cv,
    'mean score': score_mean,
    'std score': score_std,
    'recall score': score_recall_score
})

cv_summary

```

Out[52]:

	method	cv score	mean score	std score	recall score
0	Logistic Regression	[0.75556, 0.70787, 0.74157, 0.69663, 0.75281]	0.730886	0.024112	0.701571
1	Decision Tree Classifier	[0.65556, 0.66292, 0.60674, 0.67416, 0.66292]	0.652459	0.023620	0.617801
2	KNN Classifier	[0.48889, 0.59551, 0.53933, 0.59551, 0.57303]	0.558452	0.040406	0.570681
3	Random Forest Classifier	[0.78889, 0.75281, 0.74157, 0.76404, 0.74157]	0.757778	0.017647	0.664921
4	Ada Boost Classifier	[0.75556, 0.7191, 0.74157, 0.68539, 0.73034]	0.726392	0.023793	0.722513
5	Gradient Boosting Classifier	[0.82222, 0.7191, 0.73034, 0.77528, 0.77528]	0.764444	0.036858	0.712042

In []:

In []: HyperParam Tuning:

In []:

```
In [53]: lr_estimator = Pipeline([
    ('transformer', transformer),
    ('model', logreg)])

hyperparam_space = {
    'model_C': [ 1, 0.5, 0.1, 0.05, 0.01],
    'model_solver': ['newton-cg', 'lbfgs', 'liblinear'],
    'model_class_weight': ['balanced', 'dict'],
    'model_max_iter': [100, 200, 300],
    'model_multi_class': ['auto', 'ovr', 'multinomial'],
    'model_random_state': [2222]
}

grid_lr = GridSearchCV(
    lr_estimator,
    param_grid = hyperparam_space,
    cv = StratifiedKFold(n_splits = 5),
    scoring = 'recall',
    n_jobs = -1)

grid_lr.fit(X_train, y_train)

print('best score', grid_lr.best_score_)
print('best param', grid_lr.best_params_)
```

```
best score 0.7376279650436953
best param {'model_C': 0.5, 'model_class_weight': 'dict', 'model_max_iter': 100, 'model_multi_class': 'auto', 'model_random_state': 2222, 'model_solver': 'newton-cg'}
```

In [54]:

```
logreg_pipe.fit(X_train, y_train)
recall_logreg = (recall_score(y_test, logreg_pipe.predict(X_test)))

grid_lr.best_estimator_.fit(X_train, y_train)
recall_grid = (recall_score(y_test, grid_lr.predict(X_test)))

score_list = [recall_logreg, recall_grid]
method_name = ['Logistic Regression Before Tuning', 'Logistic Regression After Tuning']
best_summary = pd.DataFrame({
    'method': method_name,
    'score': score_list
})
best_summary
```

Out[54]:

	method	score
0	Logistic Regression Before Tuning	0.701571
1	Logistic Regression After Tuning	0.706806

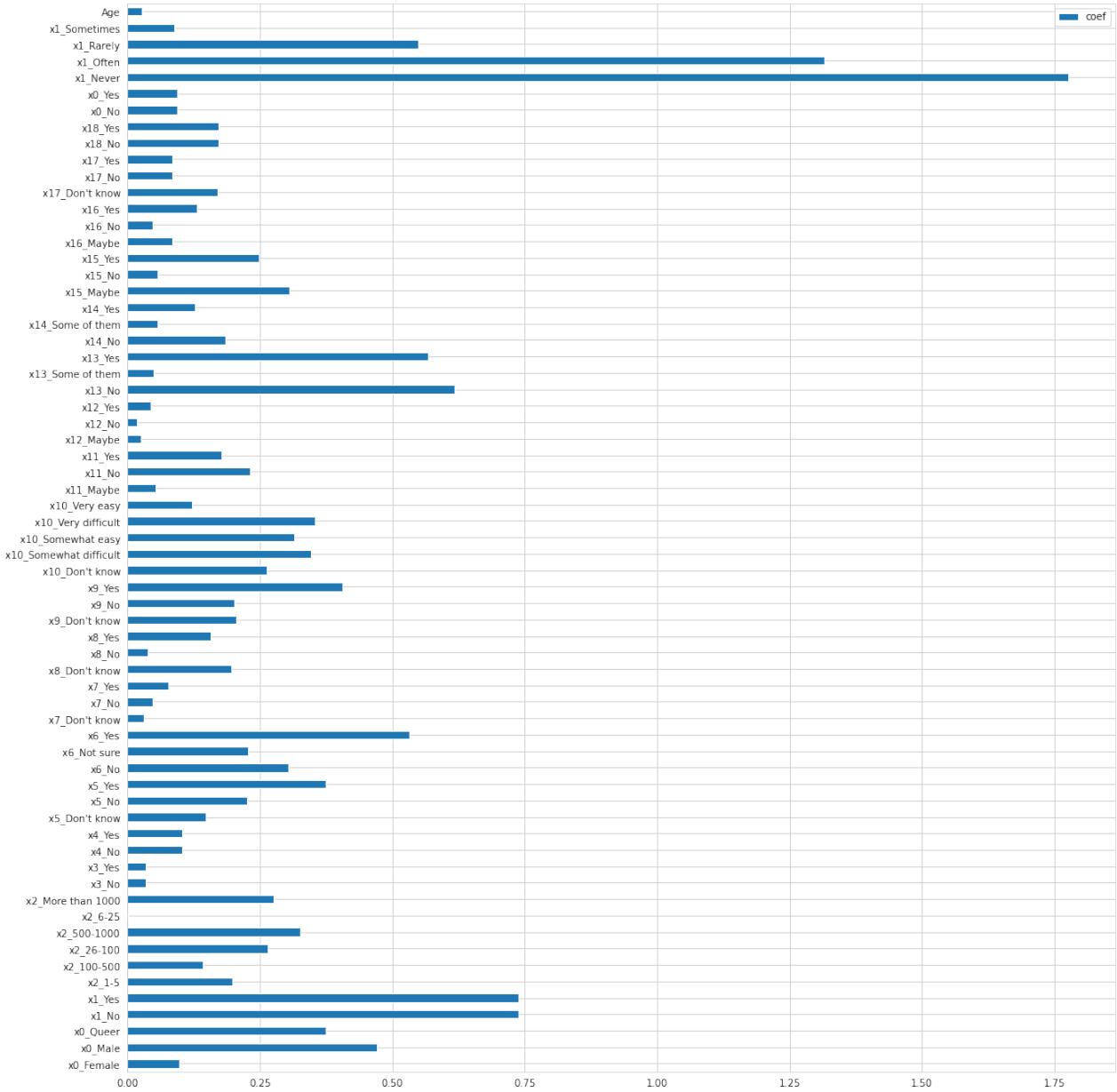
In []:

In []: Feature Selection using Tuning Result:

In []:

```
In [55]: features = list(transformer.transformers_[0][1].get_feature_names())+list(grid_lr.best_estimator_.feature_names)
coef_table = pd.DataFrame({'coef': grid_lr.best_estimator_.coef_.flatten()})
abs(coef_table).plot(kind = 'barh', figsize = (18,20))
```

Out[55]: <AxesSubplot:>



In []:

In []: Based on selecting features based on coefficient score, I decided to drop all answer choices **for** every feature. There are Age, x3(Remote_work), x7(We

In []:

In []:

In []:

```
Re-run Using Feature Selection  
linkcode  
Preprocessing Scheme
```

```
OneHotEncoding: Gender, Family_History, Employee_Numbers, Tech_Company, Ben_Mental_Health_Consequence, Coworkers, Supervisor, Mental_Health_Interview, Observed_Consequence  
Mode: Self_Employed, Work_Interfere  
Target: Treatment
```

In []:

In [56]:

```
mh_tuning = mh.copy()  
mh_tuning.drop(columns = ['Age', 'Remote_Work', 'Wellness_Program', 'Physical_Health_Program'], axis=1)  
mh_tuning.head()
```

Out[56]:

	Gender	Self_Employed	Family_History	Treatment	Work_Interfere	Employee_Numbers
0	Female	NaN	NaN	No	1	Often
1	Male	NaN	NaN	No	0	Rarely
2	Male	NaN	NaN	No	0	Rarely
3	Male	NaN	NaN	Yes	1	Often
4	Male	NaN	NaN	No	0	Never

In [57]:

```
mode_onehot_pipe_second = Pipeline([  
    ('encoder', SimpleImputer(strategy = 'most_frequent')),  
    ('one hot encoder', OneHotEncoder(handle_unknown = 'ignore'))])  
  
transformer_second = ColumnTransformer([  
    ('one hot', OneHotEncoder(handle_unknown = 'ignore'), ['Gender', 'Family_History', 'Tech_Company', 'Ben_Mental_Health_Consequence', 'Coworkers', 'Supervisor', 'Mental_Health_Interview', 'Observed_Consequence']),  
    ('mode_onehot_pipe', mode_onehot_pipe_second, ['Self_Employed', 'Work_Interfere'])])
```

In [58]:

```
X_select = mh_tuning.drop('Treatment', axis = 1)  
y_select = mh_tuning['Treatment']
```

```
In [59]: X_select_train, X_select_test, y_select_train, y_select_test = train_test_s
stratify = y_select,
test_size = 0.3,
random_state = 2222)
```

```
In [60]: logreg_second = LogisticRegression(C = 0.5, class_weight = 'dict', max_iter = 1000,
                                         multi_class = 'auto', random_state = 2222)
logreg_second_pipe = Pipeline([('transformer', transformer_second), ('model', logreg_second)])
logreg_second_pipe.fit(X_select_train, y_select_train)
print('After Feature Selection Process, the score is ', recall_score(y_select_test, logreg_second_pipe.predict(X_select_test)))
```

After Feature Selection Process, the score is 0.7277486910994765

```
In [ ]:
```

The insights from the analysis can measure campaign effectiveness and guide future strategies.

The results obtained from your thorough analysis of the dataset related to the mental health awareness campaign hold a wealth of valuable information that can be instrumental in gauging campaign effectiveness and charting the course for future strategies within the tech workplace. Let's delve into how these insights can be harnessed to fulfill these objectives:

1. Grasping the Present State:

- Begin by examining the current mindset and behaviors of professionals in the tech industry regarding mental health. This encompasses understanding the prevalence of mental health disorders, the existing stigma, and the readiness to seek help.

2. Benchmarking and Comparative Analysis:

- To measure the long-term impact of the campaign, compare the 2014 data with more recent data, if available. This comparison can reveal shifts in mental health awareness and attitudes over time.

3. Identification of Vital Metrics:

- Establish specific metrics that serve as yardsticks for campaign effectiveness. These could include tracking changes in the proportion of employees willing to openly discuss mental health in the workplace, those seeking assistance, or any reduction in the stigma associated with mental health issues.

4. Segmentation and Subgroup Analysis:

- Segment the dataset according to factors such as job roles, age, gender, and company size. This segmentation helps pinpoint subgroups in the tech workplace with differing attitudes and needs. For instance, it might unveil that younger employees are more open to mental health initiatives.

5. Correlation Investigations:

- Dive into correlations between various variables, like attitudes towards mental health and job satisfaction, productivity, or absenteeism. This exploration can shed light on direct or indirect effects of the campaign on workplace outcomes.

6. Highlighting Success Stories:

- Seek out instances where the campaign has had a notably positive impact, either on individuals or within specific companies. These success stories can serve as compelling case studies, offering evidence of the campaign's effectiveness.

7. Leveraging Feedback and Surveys:

- Whenever possible, conduct follow-up surveys and gather feedback from employees to understand their perceptions of the campaign. This qualitative data can complement the quantitative analysis.

8. Long-Term Analysis:

- If subsequent years' data is accessible, engage in longitudinal analysis to track trends over time. This can help determine whether the campaign's effects are sustained or diminishing.

9. Harnessing Predictive Modeling:

- Utilize predictive modeling to forecast the potential impact of future campaigns. Drawing insights from historical data, simulate different scenarios to assess the likely outcomes of diverse strategies.

10. Informed Recommendations:

- Building on the analysis, formulate a set of recommendations for future campaign strategies. These recommendations may encompass targeting specific demographics, focusing on destigmatization, or implementing educational initiatives.

11. Emphasizing Continuous Monitoring:

- Stress the significance of ongoing monitoring and iterative strategies. The insights extracted from your analysis should function as a launchpad for shaping and fine-tuning future campaigns.

12. Collaboration and Knowledge Sharing:

- Share your findings and collaborate with relevant stakeholders, including mental health organizations, HR departments, and company leadership. This collaborative approach is crucial in implementing and enhancing mental health initiatives.

Conclusion:

In a nutshell, the insights gleaned from our analysis will not only serve as a yardstick for evaluating the effectiveness of the mental health awareness campaign but also as a compass for shaping future strategies. The ultimate objective is to craft data-driven initiatives that have a lasting and positive impact on mental health awareness and support within the tech workplace.