

California State University - Los Angeles

Mental Health Analysis in Tech

Daniel Ozorio Delfin

Atharva Borole

May 5, 2024

CIS 5270 - 01 Business Intelligence

Instructor Balan

Introduction

In modern society, the discourse surrounding mental health has undergone a remarkable evolution, going beyond its historical mark to become a crucial point of public attention and concern. This transformation is particularly remarkable within the workplace, where organizations are increasingly recognizing the importance of prioritizing employee well-being as a base of sustainable success. Within the dynamic landscape of the tech industry, renowned for its innovative spirit and relentless pursuit of progress, the topic of mental health has emerged as a critical concern. The "Mental Health in Tech Survey" dataset serves as a growth of insights into the intersection of technology and mental well-being. In recent years, this dataset has gathered attention as a valuable resource for researchers and the strongest matches seeking to understand the frequency, determinants, and consequences of mental health issues among tech professionals. Collected through a survey administered to individuals working in various tech-related roles, the dataset offers a rich tapestry of responses encompassing a wide range of demographic, occupational, and psychosocial variables. Within the tech sector, individuals often find themselves navigating a myriad of challenges, ranging from tight deadlines and high-stakes projects to intense competition and demanding work cultures. These pressures can take a toll on one's mental health, manifesting in various forms such as stress, anxiety, and burnout. Moreover, the common stigma surrounding mental illness in many workplace environments often worsens these issues, creating barriers to seeking help and support. By analyzing the huge lineup of responses collected through this survey, we can uncover insights into the factors influencing mental health outcomes among tech professionals. In an article (2022), the authors share their analysis with the help of a pie chart. In that chart, question 1 (seeking therapy) is represented by the inside pie, whereas question 2 is represented by the outer rim (degree of work interfere).

Although 49% of participants (blue section, n=436/883) sought therapy, signaling that they have or had a mental illness, the majority of them still suffer at work on a regular basis, ranging from rarely to frequently. There is still a group of people (14 percent, n=127/883) who suffer at work to varying degrees despite never seeking treatment (the red part). From statistics and job roles to organizational policies and access to resources, the dataset provides a holistic view of the multifaceted nature of mental health in the tech sector. Our project seeks to leverage the power of data analysis to extract meaningful insights from the "Mental Health in Tech Survey" dataset. Through a combination of statistical methods and machine learning techniques, we aim to identify patterns, trends, and associations that shed light on the frequency and determinants of mental health issues within the tech community. By uncovering actionable insights, we hope to inform strategies and involvement aimed at promoting mental well-being and creating a supportive work environment for tech professionals.

Dataset URL and Dataset Description

Dataset URL: [Mental Health in Tech Survey \(kaggle.com\)](https://www.kaggle.com/datasets/akshaykumar1995/mental-health-in-tech-survey)

Dataset Description:

Field Name	Data Description	Example Value
Timestamp	The time in which the survey was taken	2014-08-27 11:29:31
Age	The age of the participant	37
Gender	The gender of the participant	Female

Country	The country of where the survey was done	United States
state	The state of the participant if they live in the United States	IL
self_employed	If the participant is self-employed	N/A
family_history	If the participant has a history of mental illness in the family	No
treatment	If the participant has sought out treatment for a mental health condition	Yes
work_interfere	If the participant had any interferences at work because of their mental health	Often
no_employees	How many employees the company has	6-25
remote_work	Does the company allows at least 50% of remote work	No
tech_company	Is the company primarily a tech company	Yes
benefits	If the employer provide mental health benefits	Yes
care_options	If the participant is aware of the mental health options provided by the employer	Not sure
wellness_program	If the participant has ever discussed mental health as part of the employee wellness program they offer	No
seek_help	If there are resources for mental health being provided by the employer	Yes
anonymity	If the participant's anonymity is protected if they choose to take advantage of any resources	Yes

leave	How easy it is to take medical leave for a mental health condition	Somewhat easy
mental_health_consequences	If the participant thinks that discussing a mental health issue would result in a negative consequence	No
phys_health_consequence	If the participant thinks that discussing a physical health issue would result in a negative consequence	No
coworkers	If the participant is willing to discuss a mental health issue with their coworkers	Some of them
supervisor	If the participant is willing to discuss a mental health issue with their direct supervisor	Yes
mental_health_interview	If the participant is willing to discuss a mental health issue with a potential employer at a future interview	No
phys_health_interview	If the participant is willing to discuss a physical health issue with their coworkers	Maybe
mental_vs_physical	If the participant feels that mental health is taken as seriously as physical health	Yes
obs_consequences	If the participant has heard of observed negative consequences for coworkers with mental health conditions in your workplace	No
comments	Any additional notes that were made by the participant	N/A

Data Cleaning

1) Convert Data Type

We began by changing the "Timestamp" column's name to "Date." Given that this is a survey, analyzing the dataset by year provides a more comprehensive perspective than by specific timestamps. We chose to retain only the month and year since the precise time and day of survey completion were not necessary. To achieve this, we first renamed the "Timestamp" column to "Date" in our dataframe, making it simpler to understand when the survey responses were collected. Next, we converted the column's data type to "datetime" and extracted just the month and year from the entries. This step is crucial because it simplifies the data, allowing for easier analysis and interpretation of trends over time. Benefits to making it this way ensures consistency across the dataset, facilitating easier comparison and summarization.

Pre-Data Cleaning

The screenshot shows the Spyder Python IDE interface. The code editor window displays a script named 'Python Final Project.py' with the following content:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
final_project_df = pd.read_csv('survey.csv')
# Counting the number of rows in the DataFrame
num_rows = len(final_project_df)
# Displaying the number of rows
print("Number of rows:", num_rows)
print(final_project_df.head(15))
```

The IPython console window shows the output of the last two print statements. It displays the first 15 rows of the 'final_project_df' DataFrame, which has 1295 rows. The output includes columns for timestamp and comments, with many entries showing 'NaN' for comments.

Index	Timestamp	Comments
0	2014-08-27 11:29:31	NaN
1	2014-08-27 11:29:37	NaN
2	2014-08-27 11:29:44	NaN
3	2014-08-27 11:29:46	NaN
4	2014-08-27 11:30:22	NaN
5	2014-08-27 11:31:22	NaN
6	2014-08-27 11:31:50	NaN
7	2014-08-27 11:32:05	NaN
8	2014-08-27 11:32:39	NaN
9	2014-08-27 11:32:43	NaN
10	2014-08-27 11:32:44	NaN
11	2014-08-27 11:33:04	NaN
12	2014-08-27 11:33:23	NaN
13	2014-08-27 11:33:26	I'm not on my company's health insurance which...
14	2014-08-27 11:33:57	NaN

The status bar at the bottom right indicates the current time is 12:35 PM and the date is 5/15/2024.

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

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

# Counting the number of rows in the DataFrame
num_rows = len(final_project_df)

# Displaying the number of rows
print("Number of rows:", num_rows)

print(final_project_df.head(15))
```

```

Number of rows: 1259
      Timestamp    ...
0  2014-08-27 11:29:31  ...
1  2014-08-27 11:29:37  ...
2  2014-08-27 11:29:44  ...
3  2014-08-27 11:29:46  ...
4  2014-08-27 11:30:22  ...
5  2014-08-27 11:31:22  ...
6  2014-08-27 11:31:50  ...
7  2014-08-27 11:32:05  ...
8  2014-08-27 11:32:39  ...
9  2014-08-27 11:32:43  ...
10 2014-08-27 11:32:44  ...
11 2014-08-27 11:32:49  ...
12 2014-08-27 11:33:23  ...
13 2014-08-27 11:33:26  ... I'm not on my company's health insurance which...
14 2014-08-27 11:33:57  ...

```

Post-Data Cleaning:

The screenshot shows the Spyder IDE interface. On the left, the code editor displays a Python script named `Python Final Project.py`. The code reads a CSV file, counts the rows, renames the timestamp column to 'Date', converts it to datetime format, and extracts the year and month. A call to `print(final_project_df.head(15))` is shown. On the right, the IPython Console History shows the execution of this command, resulting in a DataFrame output. The output is highlighted with a yellow box, showing the first 15 rows with columns 'Date' and 'comments'. Row 13 contains the text "I'm not on my company's health insurance which...". The bottom status bar indicates the system is at 12:37 PM on 5/15/2024.

```

C:\Users\Danny\Downloads\Mental Health\Python Final Project.py
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 final_project_df = pd.read_csv('survey.csv')
6
7 # Counting the number of rows in the DataFrame
8 num_rows = len(final_project_df)
9
10 # Displaying the number of rows
11 print("Number of rows:", num_rows)
12
13 # Renaming the "Timestamp" column to "Date"
14 final_project_df.rename(columns={'Timestamp': 'Date'}, inplace=True)
15
16 # Converting the "Date" column to datetime format
17 final_project_df['Date'] = pd.to_datetime(final_project_df['Date'])
18
19 # Extracting the year and month from the datetime values and combine them
20 final_project_df['Date'] = final_project_df['Date'].dt.to_period('M')
21
22
23 print(final_project_df.head(15))
24

```

	Date	comments
0	2014-08	NaN
1	2014-08	NaN
2	2014-08	NaN
3	2014-08	NaN
4	2014-08	NaN
5	2014-08	NaN
6	2014-08	NaN
7	2014-08	NaN
8	2014-08	NaN
9	2014-08	NaN
10	2014-08	NaN
11	2014-08	NaN
12	2014-08	NaN
13	2014-08	I'm not on my company's health insurance which...
14	2014-08	NaN

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

# Counting the number of rows in the DataFrame
num_rows = len(final_project_df)

# Displaying the number of rows
#print("Number of rows:", num_rows)

#print(final_project_df.head(15))

# Renaming the "Timestamp" column to "Date"
final_project_df.rename(columns={'Timestamp': 'Date'}, inplace=True)

# Converting the "Date" column to datetime format
final_project_df['Date'] = pd.to_datetime(final_project_df['Date'])

# Extracting the year and month from the datetime values and combine them
final_project_df['Date'] = final_project_df['Date'].dt.to_period('M')

print(final_project_df.head(15))

```

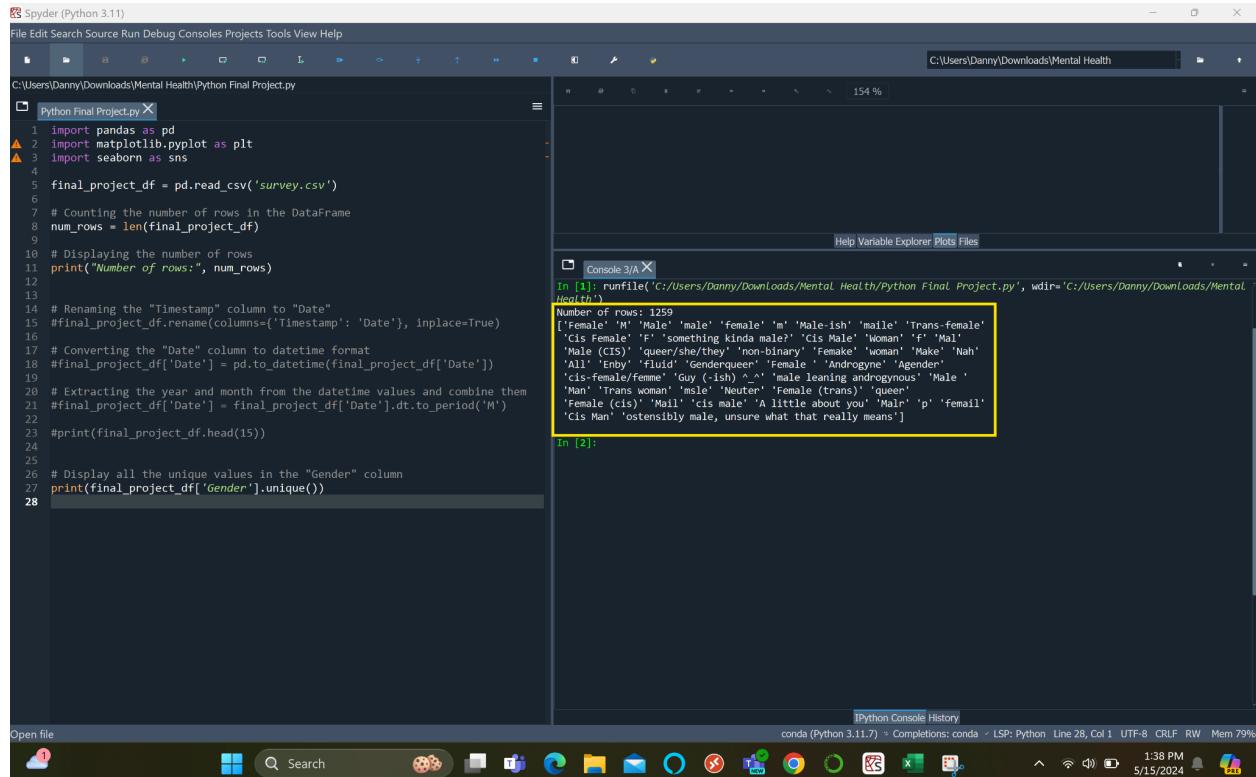
	Number of rows: 1259		comments
	Date	...	
0	2014-08	...	NaN
1	2014-08	...	NaN
2	2014-08	...	NaN
3	2014-08	...	NaN
4	2014-08	...	NaN
5	2014-08	...	NaN
6	2014-08	...	NaN
7	2014-08	...	NaN
8	2014-08	...	NaN
9	2014-08	...	NaN
10	2014-08	...	NaN
11	2014-08	...	NaN
12	2014-08	...	NaN
13	2014-08	... I'm not on my company's health insurance which...	
14	2014-08	...	NaN

2) Data Imputation

The survey received a wide range of responses to the question about gender. To simplify the data, we categorized gender into three groups: Male, Female, and Other. This approach ensures that all participants are represented. We included any response that was not explicitly "Male" or "Female" under the "Other" category. We accomplished this by identifying all unique values in the gender column and mapping them to our standardized categories.

This proved beneficial as it provided a standard category that was able to easily identify the variety of applicants when it came down to Gender. Furthermore, all the data was kept as accurate as possible as it was free from spelling errors and differences in punctuation.

Pre-Data Cleaning:



The screenshot shows the Spyder IDE interface with a Python script named 'Python Final Project.py' open. The script contains code for reading a CSV file, counting rows, renaming columns, converting dates, extracting year and month, and printing unique gender values. The output window displays the results of the print statements, including a large list of gender variations found in the 'Gender' column.

```

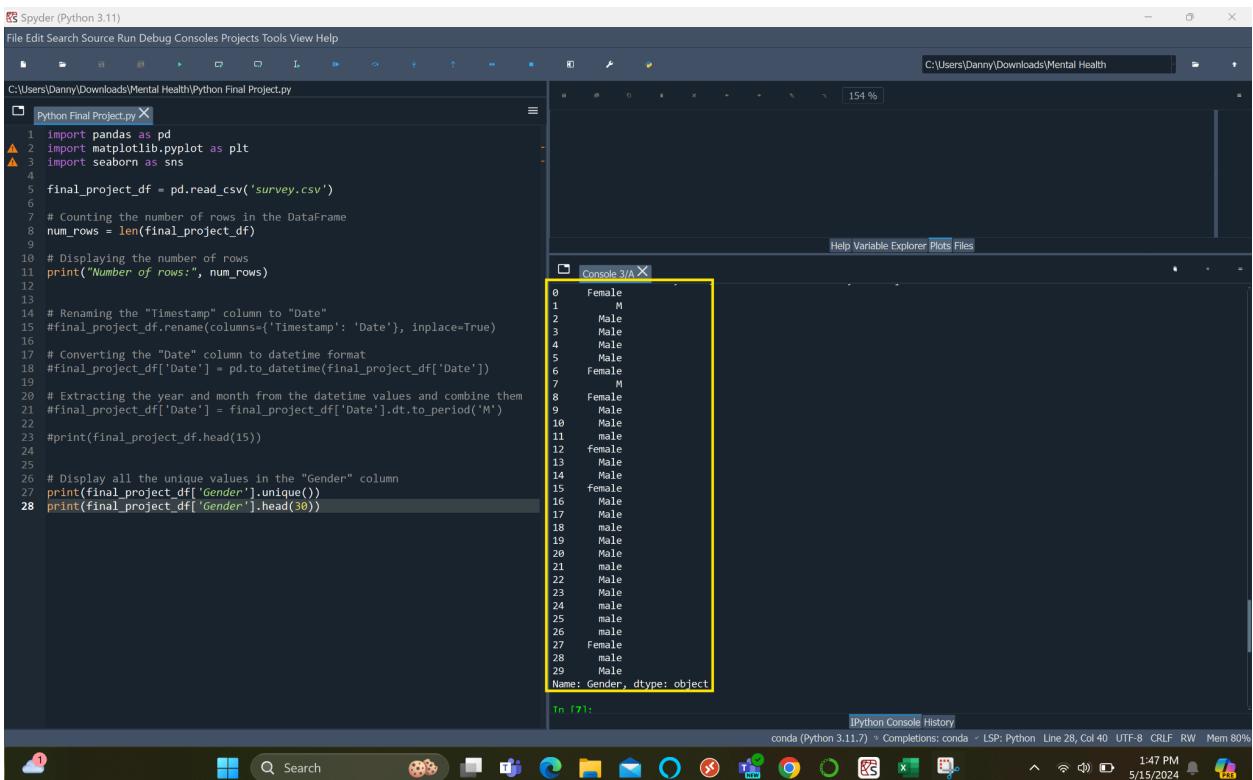
Spyder (Python 3.11)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:/Users/Danny/Downloads/Mental Health/Python Final Project.py
Python Final Project.py X
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 final_project_df = pd.read_csv('survey.csv')
6
7 # Counting the number of rows in the DataFrame
8 num_rows = len(final_project_df)
9
10 # Displaying the number of rows
11 print("Number of rows:", num_rows)
12
13
14 # Renaming the "Timestamp" column to "Date"
15 #final_project_df.rename(columns={'Timestamp': 'Date'}, inplace=True)
16
17 # Converting the "Date" column to datetime format
18 #final_project_df['Date'] = pd.to_datetime(final_project_df['Date'])
19
20 # Extracting the year and month from the datetime values and combine them
21 #final_project_df['Date'] = final_project_df['Date'].dt.to_period('M')
22
23 #print(final_project_df.head(15))
24
25
26 # Display all the unique values in the "Gender" column
27 print(final_project_df['Gender'].unique())
28

```

In [3]: runfile('C:/Users/Danny/Downloads/Mental Health/Python Final Project.py', wdir='C:/Users/Danny/Downloads/Mental Health')
Number of rows: 1289
{'male', 'M', 'Male', 'male', 'female', 'm', 'Male-ish', 'maile', 'Trans-female', 'cis female', 'F', 'something kinda male?', 'cis Male', 'Woman', 'f', 'Mal', 'Male (cis)', 'queen/she/they', 'non-binary', 'Female', 'woman', 'Male', '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', 'Male', 'p', 'femail', 'Cis Man', 'ostensibly male, unsure what that really means'}

```
# Display all the unique values in the "Gender" column
print(final_project_df['Gender'].unique())
```

```
Number of rows: 1259
['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']
```



```
# Display all the unique values in the "Gender" column
print(final_project_df['Gender'].unique())
print(final_project_df['Gender'].head(30))
```

```
0    Female
1      M
2    Male
3    Male
4    Male
5    Male
6    Female
7      M
8    Female
9    Male
10   Male
11   male
12  female
13  Male
14  Male
15  female
16  Male
17  Male
18  male
19  Male
20  Male
21  male
22  Male
23  Male
24  male
25  male
26  male
27  Female
28  male
29  Male
Name: Gender, dtype: object
```

Post-Data Cleaning:

The screenshot shows the Spyder IDE interface with two main panes. The left pane displays a Python script named `Python Final Project.py`. The right pane shows a Jupyter Notebook cell output.

Script Content (`Python Final Project.py`):

```
# Mapping non-standard gender values to standardized categories
gender_mapping = {
    'Female': 'Female',
    'female': 'Female',
    'f': 'Female',
    'F': 'Female',
    'woman': 'Female',
    'femake': 'Female',
    'cis female': 'Female',
    'cis-female/femme': 'Female',
    'female ': 'Female',
    'woman': 'Female',
    'queer/she/they': 'Other',
    'non-binary': 'Other',
    'male': 'Male',
    'm': 'Male',
    'M': 'Male',
    'Male': 'Male',
    'male ': 'Male',
    'man': 'Male',
    'msle': 'Male',
    'mail': 'Male',
    'mal': 'Male',
    'mehr': 'Male',
    'cis male': 'Male',
    'cis man': 'Male',
    'Make': 'Male',
    'Male (CIS)': 'Male',
    'Male-ish': 'Male',
    'malee': 'Male',
    'Male ': 'Male',
    'Guy (-ish) ^_^': 'Male',
    'ostensibly male, unsure what that really means': 'Male',
    'something kinda male?': 'Other',
    'queer': 'Other',
    'p': 'Other',
    'Naah': 'Other',
    'ALL': 'Other',
    'Emby': 'Other',
    'fluid': 'Other',
    'Genderqueer': 'Other',
    'Androgynie': 'Other',
    'Agender': 'Other'}
```

Console Output (Cell 16):

```
In [16]: runfile('C:/Users/Danny/Downloads/Mental Health/Python Final Project.py', wdir='C:/Users/Danny/Downloads/Mental Health')
Number of rows: 1259
['Female', 'Male', 'Other']
```

Console Output (Cell 17):

```
In [17]:
```

At the bottom, the status bar indicates: conda (Python 3.11.7) > Completions: conda < LSP: Python Line 93, Col 2 UTF-8 CRLF RW Mem 73% 2:31 PM 5/15/2024

```
# Mapping non-standard gender values to standardized categories
gender_mapping = {
    'Female': 'Female',
    'female': 'Female',
    'f': 'Female',
    'F': 'Female',
    'woman': 'Female',
    'femake': 'Female',
    'cis female': 'Female',
    'cis-female/femme': 'Female',
    'female ': 'Female',
    'woman': 'Female',
    'queer/she/they': 'Other',
    'non-binary': 'Other',
    'male': 'Male',
    'm': 'Male',
    'M': 'Male',
    'Male': 'Male',
    'male ': 'Male',
    'man': 'Male',
    'msle': 'Male',
    'mail': 'Male',
    'mal': 'Male',
    'malr': 'Male',
    'cis male': 'Male',
    'cis man': 'Male',
    'Make': 'Male',
    'Male (CIS)': 'Male',
    'Male-ish': 'Male',
    'maile': 'Male',
    'Male ': 'Male',
    'Guy (-ish) ^_^': 'Male',
    'ostensibly male, unsure what that really means': 'Male',
    'something kinda male?': 'Other',
    'queer': 'Other',
    'p': 'Other',
    'Nah': 'Other',
    'ALL': 'Other',
    'Enby': 'Other',
    'fluid': 'Other',
    'Genderqueer': 'Other',
    'Androgyn': 'Other',
    'Agender': 'Other',
```

```

'male Leaning androgynous': 'Other',
'Neuter': 'Other',
'Female (trans)': 'Other',
'Trans-female': 'Other',
'A Little about you': 'Other',
'Trans woman': 'Other',
'Female (cis)': 'Other',
'Mail': 'Other'
}

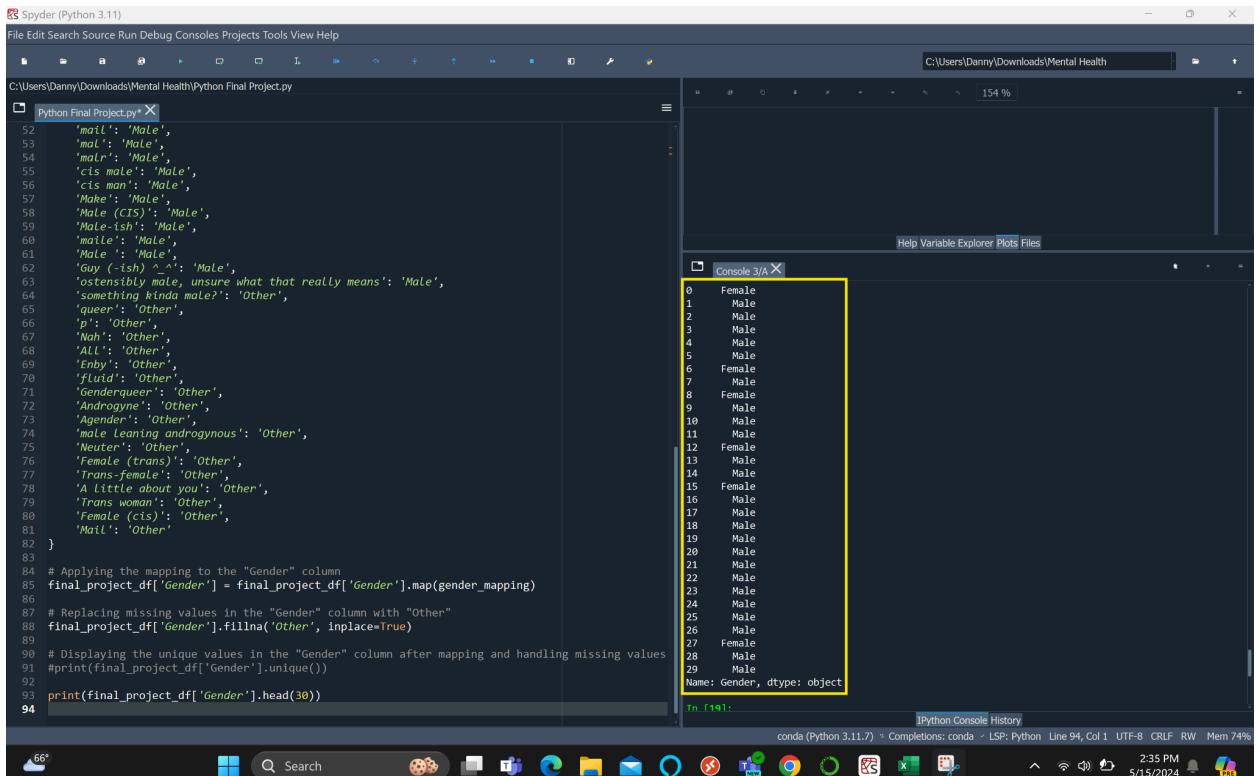
# Applying the mapping to the "Gender" column
final_project_df[ 'Gender' ] = final_project_df[ 'Gender' ].map(gender_mapping)

# Replacing missing values in the "Gender" column with "Other"
final_project_df[ 'Gender' ].fillna('Other', inplace=True)

# Displaying the unique values in the "Gender" column after mapping and handling missing values
print(final_project_df[ 'Gender' ].unique())

```

Number of rows: 1259
['Female' 'Other' 'Male']



The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays the script `Python Final Project.py` with several lines of gender mapping logic. On the right, the IPython Console window shows the output of the command `print(final_project_df['Gender'].head(30))`. The console output is highlighted with a yellow box and lists the first 30 entries from the 'Gender' column, which are all 'Male'. The status bar at the bottom indicates the environment is conda Python 3.11.7.

```

0   Male
1   Male
2   Male
3   Male
4   Male
5   Male
6   Female
7   Male
8   Female
9   Male
10  Male
11  Male
12  Female
13  Male
14  Male
15  Female
16  Male
17  Male
18  Male
19  Male
20  Male
21  Male
22  Male
23  Male
24  Male
25  Male
26  Male
27  Female
28  Male
29  Male
Name: Gender, dtype: object

```

```
print(final_project_df[ 'Gender' ].head(30))
```

```
0    Female
1    Male
2    Male
3    Male
4    Male
5    Male
6    Female
7    Male
8    Female
9    Male
10   Male
11   Male
12   Female
13   Male
14   Male
15   Female
16   Male
17   Male
18   Male
19   Male
20   Male
21   Male
22   Male
23   Male
24   Male
25   Male
26   Male
27   Female
28   Male
29   Male
Name: Gender, dtype: object
```

3) Removing Missing Values

We are updating the "Age" column to replace any blank entries or implausible numbers.

We set the minimum age to 18, as this is the age at which young adults can work without a permit, and the maximum age to 80, since it is unlikely for individuals in their 80s to be employed at a tech firm. Guessing the survey respondent's age introduces a lot of uncertainty. Adding arbitrary numbers would distort the column's descriptive statistics. Therefore, we opted to replace any such values with the average age to maintain the accuracy of the column's statistical representation.

This proved to be beneficial as that maintains as much data integrity as possible. Any other number outside the range we decided upon would not make sense.

Pre-Data Cleaning:

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a file named 'Python Final Project.py' containing Python code for gender mapping and data cleaning. On the right, the IPython Console shows the execution of a command to print unique values from the 'Age' column of a DataFrame. The output is a list of integers representing age values.

```

55     'cis male': 'Male',
56     'cis man': 'Male',
57     'Make': 'Male',
58     'Male (CIS)': 'Male',
59     'Male-ish': 'Male',
60     'male': 'Male',
61     'Male ': 'Male',
62     'Guy (-ish) ^_^': 'Male',
63     'ostensibly male, unsure what that really means': 'Male',
64     'something kinda male?': 'Other',
65     'queer': 'Other',
66     'p': 'Other',
67     'Nah': 'Other',
68     'ALL': 'Other',
69     'Enby': 'Other',
70     'fluid': 'Other',
71     'Genderqueer': 'Other',
72     'Androgynie': 'Other',
73     'Agender': 'Other',
74     'male leaning androgynous': 'Other',
75     'Neuter': 'Other',
76     'Female (trans)': 'Other',
77     'Trans-female': 'Other',
78     'A Little about you': 'Other',
79     'Trans woman': 'Other',
80     'Female (cis)': 'Other',
81     'Mall': 'Other'
82 }
83
84 # Applying the mapping to the "Gender" column
85 final_project_df['Gender'] = final_project_df['Gender'].map(gender_mapping)
86
87 # Replacing missing values in the "Gender" column with "Other"
88 final_project_df['Gender'].fillna('Other', inplace=True)
89
90 # Displaying the unique values in the "Gender" column after mapping and handling missing values
91 #print(final_project_df['Gender'].unique())
92
93 #print(final_project_df['Gender'].head(30))
94
95 print(final_project_df['Age'].unique())
96
97

```

In [19]: runfile('C:/Users/Danny/Downloads/Mental Health/Python Final Project.py', wdir='C:/Users/Danny/Downloads/Mental Health')

Number of rows: 1259					
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	999999999999
48	20	57	58	47	62
51	65	49	-1726	5	53
61	8	11	-1	72]	

In [20]:

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

Number of rows: 1259					
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	999999999999
48	20	57	58	47	62
51	65	49	-1726	5	53
61	8	11	-1	72]	

Post-Data Cleaning:

The screenshot shows the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The title bar indicates the file is 'Python Final Project.py' located at 'C:\Users\Danny\Downloads\Mental Health'. The code editor contains a script for transforming gender and age data. The IPython console window displays the execution of the script, showing the number of rows (1259) and a list of unique age values: [37. 44. 32. 31. 33. 35. 39. 42. 23. 29. 36. 27. 46. 41. 34. 30. 40. 38. 50. 24. 18. 28. 26. 22. 19. 25. 45. 21. 43. 56. 60. 54. 55. 48. 20. 57. 58. 47. 62. 51. 65. 49. 53. 61. 72.]

```

import numpy as np

# Finding values outside the range [18, 80]
out_of_range = (final_project_df['Age'] < 18) | (final_project_df['Age'] > 80)

# Replace out-of-range values with NaN
final_project_df.loc[out_of_range, 'Age'] = np.nan

# Calculating the mean of the "Age" column, excluding NaN values, and round it to the nearest whole number
mean_age_rounded = round(final_project_df['Age'].mean())

# Replacing NaN values with the rounded mean of the "Age" column
final_project_df['Age'].fillna(mean_age_rounded, inplace=True)

# Display the unique values in the "Age" column after replacing out-of-range values with rounded mean
print(final_project_df['Age'].unique())

```

```

Number of rows: 1259
[37. 44. 32. 31. 33. 35. 39. 42. 23. 29. 36. 27. 46. 41. 34. 30. 40. 38.
 50. 24. 18. 28. 26. 22. 19. 25. 45. 21. 43. 56. 60. 54. 55. 48. 20. 57.
 58. 47. 62. 51. 65. 49. 53. 61. 72.]

```

4) Data Transformation

We decided to transform the data for the “self_employed” column so that we could standardize any response that was given for this question into a Yes or No. Responses included N/A in which we reasoned that if the person was self-employed, then they would simply say

“yes” instead of putting N/A. By this logic, we made any N/A to equal no. This is beneficial because we are able to gain more insights and get more counts out of the values we currently have.

Pre-Data Cleaning:

The screenshot shows the Spyder IDE interface with a Python script file open and a Jupyter Notebook cell below it.

Python Script (Python Final Project.py):

```
74     'male leaning androgynous': 'Other',
75     'Neuter': 'Other',
76     'Female (trans)': 'Other',
77     'Trans-female': 'Other',
78     'A little about you': 'Other',
79     'Trans woman': 'Other',
80     'Female (cis)': 'Other',
81     'Male': 'Other'
82 }
83
84 # Applying the mapping to the "Gender" column
85 final_project_df['Gender'] = final_project_df['Gender'].map(gender_mapping)
86
87 # Replacing missing values in the "Gender" column with "Other"
88 final_project_df['Gender'].fillna('Other', inplace=True)
89
90 # Displaying the unique values in the "Gender" column after mapping and handling nulls
91 #print(final_project_df['Gender'].unique())
92
93 #print(final_project_df['Gender'].head(30))
94
95 #print(final_project_df['Age'].unique())
96
97 import numpy as np
98
99 # Finding values outside the range [18, 80]
100 out_of_range = (final_project_df['Age'] < 18) | (final_project_df['Age'] > 80)
101
102 # Replace out-of-range values with NaN
103 final_project_df.loc[out_of_range, 'Age'] = np.nan
104
105 # Calculating the mean of the "Age" column, excluding NaN values, and round it to two decimal places
106 mean_age_rounded = round(final_project_df['Age'].mean())
107
108 # Replacing NaN values with the rounded mean of the "Age" column
109 final_project_df['Age'].fillna(mean_age_rounded, inplace=True)
110
111 # Displaying the unique values in the "Age" column after replacing out-of-range values
112 #print(final_project_df['Age'].unique())
113
114 print(final_project_df['self_employed'].head(15))
115 print(final_project_df['self_employed'].unique())
116
```

Jupyter Notebook Cell (In [4]):

```
In [4]: runfile('C:/Users/Danny/Downloads/Mental Health/Python Final Project.py', wdir='C:/Users/Danny/Downloads/Mental Health')
Number of rows: 1259
0    NaN
1    NaN
2    NaN
3    NaN
4    NaN
5    NaN
6    NaN
7    NaN
8    NaN
9    NaN
10   NaN
11   NaN
12   NaN
13   NaN
14   NaN
Name: self_employed, dtype: object
[nan 'Yes' 'No']
```

Jupyter Notebook Cell (In [5]):

```
In [5]:
```

Output (IPython Console History):

```
conda (Python 3.11.7) ▶ Completions: conda ▶ LSP: Python Line 115, Col 38 ASCII CRLF RW Mem 71%
9:22 PM 5/17/2024
```

```
Number of rows: 1259
0      NaN
1      NaN
2      NaN
3      NaN
4      NaN
5      NaN
6      NaN
7      NaN
8      NaN
9      NaN
10     NaN
11     NaN
12     NaN
13     NaN
14     NaN
Name: self_employed, dtype: object
[nan 'Yes' 'No']
```

Post-Data Cleaning:

The screenshot shows the Spyder IDE interface with a Python script file open and a Jupyter Notebook cell running.

Python Script Content:

```
84 # Applying the mapping to the "Gender" column
85 final_project_df['Gender'] = final_project_df['Gender'].map(gender_mapping)
86
87 # Replacing missing values in the "Gender" column with "Other"
88 final_project_df['Gender'].fillna('Other', inplace=True)
89
90 # Displaying the unique values in the "Gender" column after mapping and handling missing values
91 #print(final_project_df['Gender'].unique())
92
93 #print(final_project_df['Gender'].head(30))
94
95 #print(final_project_df['Age'].unique())
96
97 import numpy as np
98
99 # Finding values outside the range [18, 80]
100 out_of_range = (final_project_df['Age'] < 18) | (final_project_df['Age'] > 80)
101
102 # Replace out-of-range values with NaN
103 final_project_df.loc[out_of_range, 'Age'] = np.nan
104
105 # Calculating the mean of the "Age" column, excluding NaN values, and round it to two decimal places
106 mean_age_rounded = round(final_project_df['Age'].mean())
107
108 # Replacing NaN values with the rounded mean of the "Age" column
109 final_project_df['Age'].fillna(mean_age_rounded, inplace=True)
110
111 # Displaying the unique values in the "Age" column after replacing out-of-range values
112 #print(final_project_df['Age'].unique())
113
114 #print(final_project_df['self_employed'].head(15))
115 #print(final_project_df['self_employed'].unique())
116
117 # Replacing NaN values with "No" in the "self_employed" column
118 final_project_df['self_employed'].replace(np.nan, 'No', inplace=True)
119
120 # Converting the "self_employed" column to numerical values
121 final_project_df['self_employed'] = final_project_df['self_employed'].map({'Yes': 1, 'No': 0})
122
123 # Displaying the first few rows to verify the changes
124 print(final_project_df[['self_employed']].head(15))
125 print(final_project_df['self_employed'].unique())
```

Jupyter Notebook Cell Output (In [6]):

```
In [6]: runfile('C:/Users/Danny/Downloads/Mental Health/Python Final Project.py', wdir='C:/Users/Danny/Downloads/Mental Health')
Number of rows: 1259
self_employed
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
10     0
11     0
12     0
13     0
14     0
[0 1]
```

IPython Console History:

```
conda (Python 3.11.7) > Completions: conda > LSP: Python Line 125, Col 50 ASCII CRLF RW Mem 64%
```

```
# Replacing NaN values with "No" in the "self_employed" column
final_project_df['self_employed'].replace(np.nan, 'No', inplace=True)

# Converting the "self_employed" column to numerical values
final_project_df['self_employed'] = final_project_df['self_employed'].map({'Yes': 1, 'No': 0})

# Displaying the first few rows to verify the changes
print(final_project_df[['self_employed']].head(15))
print(final_project_df['self_employed'].unique())
```

```
Number of rows: 1259
  self_employed
0              0
1              0
2              0
3              0
4              0
5              0
6              0
7              0
8              0
9              0
10             0
11             0
12             0
13             0
14             0
[0 1]
```

Summary Statistics

Statistical Analysis for Age:

The screenshot shows the Spyder Python IDE interface. The code editor on the left contains a script named 'Python Final Project.py' with several lines of Python code. Lines 114 through 118 are highlighted with a yellow box. The code performs various data manipulations, including mapping gender values and calculating summary statistics for the 'Age' column. The IPython console on the right displays the output of the 'describe()' method for the 'Age' column, showing summary statistics like count, mean, std, min, 25%, 50%, 75%, and max. The status bar at the bottom indicates the environment is 'conda (Python 3.11.7)'.

```
78     'A little about you': 'Other',
79     'Trans woman': 'Other',
80     'Female (cis)': 'Other',
81     'Mail': 'Other'
82 }
83
84 # Applying the mapping to the "Gender" column
85 final_project_df['Gender'] = final_project_df['Gender'].map(gender_mapping)
86
87 # Replacing missing values in the "Gender" column with "Other"
88 final_project_df['Gender'].fillna('Other', inplace=True)
89
90 # Displaying the unique values in the "Gender" column after mapping and handling n
91 #print(final_project_df['Gender'].unique())
92
93 #print(final_project_df['Gender'].head(30))
94
95 #print(final_project_df['Age'].unique())
96
97 import numpy as np
98
99 # Finding values outside the range [18, 80]
100 out_of_range = (final_project_df['Age'] < 18) | (final_project_df['Age'] > 80)
101
102 # Replace out-of-range values with NaN
103 final_project_df.loc[out_of_range, 'Age'] = np.nan
104
105 # Calculating the mean of the "Age" column, excluding NaN values, and round it to
106 mean_age_rounded = round(final_project_df['Age'].mean())
107
108 # Replacing NaN values with the rounded mean of the "Age" column
109 final_project_df['Age'].fillna(mean_age_rounded, inplace=True)
110
111 # Display the unique values in the "Age" column after replacing out-of-range value
112 #print(final_project_df['Age'].unique())
113
114 # The summary statistics for the "Age" column
115 age_summary = final_project_df['Age'].describe()
116 print("Summary statistics for Age:")
117 print(age_summary)
118
```

```
# The summary statistics for the "Age" column
age_summary = final_project_df['Age'].describe()
print("Summary statistics for Age:")
print(age_summary)
```

```
Number of rows: 1259
Summary statistics for Age:
count      1259.000000
mean       32.076251
std        7.265063
min        18.000000
25%        27.000000
50%        31.000000
75%        36.000000
max        72.000000
Name: Age, dtype: float64
```

The summary statistics for the "Age" column show data from 1,259 entries. The statistics that are revealed are the number of observations, minimum, maximum, mode, medium, mean, and standard deviation. The average age is approximately 32.08 years, with a standard deviation of 7.27 years. The ages range from 18 to 72 years, with the 25th percentile at 27 years, the median (50th percentile) at 31 years, and the 75th percentile at 36 years.

Some insights that we were able to discover were that most of the people that the average age of people that are working in tech are about 31 years old.

Statistical Analysis for Self_Employed:

The screenshot shows the Spyder Python IDE interface. The left pane displays the code file `Python Final Project.py`. The right pane shows the IPython console output. A yellow box highlights the line of code `self_employed_summary = final_project_df['self_employed'].describe()`. Another yellow box highlights the resulting summary statistics output.

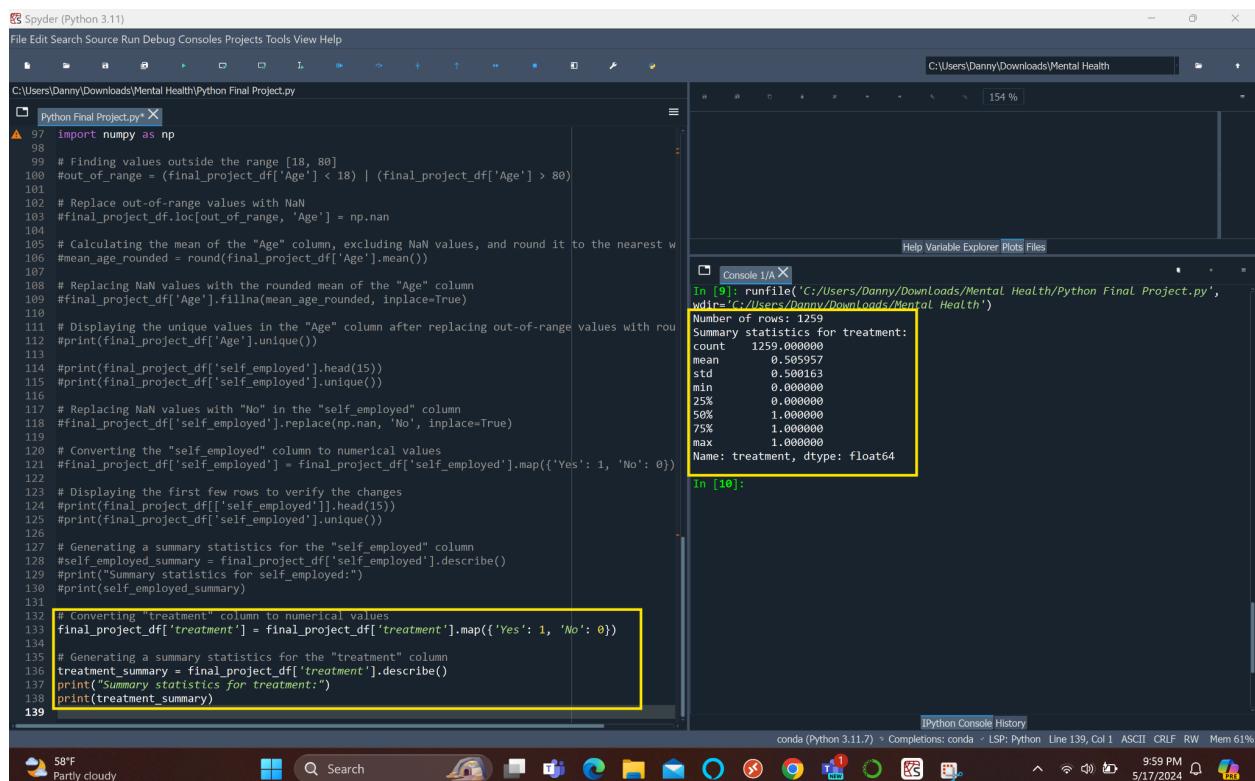
```
89 # Displaying the unique values in the "Gender" column after mapping and handling missing values
90 #print(final_project_df['Gender'].unique())
91
92 #print(final_project_df['Gender'].head(30))
93
94 #print(final_project_df['Age'].unique())
95
96
97 import numpy as np
98
99 # Finding values outside the range [18, 80]
100 out_of_range = (final_project_df['Age'] < 18) | (final_project_df['Age'] > 80)
101
102 # Replace out-of-range values with NaN
103 final_project_df.loc[out_of_range, 'Age'] = np.nan
104
105 # Calculating the mean of the "Age" column, excluding NaN values, and round it to the nearest w
106 mean_age_rounded = round(final_project_df['Age'].mean())
107
108 # Replacing NaN values with the rounded mean of the "Age" column
109 final_project_df['Age'].fillna(mean_age_rounded, inplace=True)
110
111 # Displaying the unique values in the "Age" column after replacing out-of-range values with rou
112 #print(final_project_df['Age'].unique())
113
114 #print(final_project_df['self_employed'].head(15))
115 #print(final_project_df['self_employed'].unique())
116
117 # Replacing NaN values with "No" in the "self_employed" column
118 final_project_df['self_employed'].replace(np.nan, 'No', inplace=True)
119
120 # Converting the "self_employed" column to numerical values
121 final_project_df['self_employed'] = final_project_df['self_employed'].map({'Yes': 1, 'No': 0})
122
123 # Displaying the first few rows to verify the changes
124 #print(final_project_df['self_employed'].head(15))
125 #print(final_project_df['self_employed'].unique())
126
127 # Generating a summary statistics for the "self_employed" column
128 self_employed_summary = final_project_df['self_employed'].describe()
129 print("Summary statistics for self_employed:")
130 print(self_employed_summary)
```

```
# Generating a summary statistics for the "self_employed" column
self_employed_summary = final_project_df['self_employed'].describe()
print("Summary statistics for self_employed:")
print(self_employed_summary)
```

```
Number of rows: 1259
Summary statistics for self_employed
count    1259.000000
mean      0.115965
std       0.320310
min      0.000000
25%      0.000000
50%      0.000000
75%      0.000000
max      1.000000
Name: self_employed, dtype: float64
```

The summary statistics for the `self_employed` column reveal that out of 1,259 respondents, the vast majority are not self-employed. This is indicated by the mean value of 0.116, where 1 represents "Yes" for self-employed and 0 represents "No." The data shows a significant skew towards respondents not being self-employed, as reflected by the median (50th percentile) being 0, and the same for the 25th and 75th percentiles. The standard deviation of 0.32 suggests some variation, but the predominant value is 0. The minimum and maximum values are 0 and 1 respectively, reinforcing the nature of this data. Overall, the majority of respondents are not self-employed, with a small proportion indicating self-employment.

Statistical Analysis for Treatment:



```

Spyder (Python 3.11)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Danny\Downloads\Mental Health\Python Final Project.py
Python Final Project.py* X
97 import numpy as np
98
99 # Finding values outside the range [18, 80]
100 out_of_range = (final_project_df['Age'] < 18) | (final_project_df['Age'] > 80)
101
102 # Replace out-of-range values with NaN
103 final_project_df.loc[out_of_range, 'Age'] = np.nan
104
105 # Calculating the mean of the "Age" column, excluding NaN values, and round it to the nearest w
106 mean_age_rounded = round(final_project_df['Age'].mean())
107
108 # Replacing NaN values with the rounded mean of the "Age" column
109 final_project_df['Age'].fillna(mean_age_rounded, inplace=True)
110
111 # Displaying the unique values in the "Age" column after replacing out-of-range values with rou
112 #print(final_project_df['Age'].unique())
113
114 #print(final_project_df['self_employed'].head(15))
115 #print(final_project_df['self_employed'].unique())
116
117 # Replacing NaN values with "No" in the "self_employed" column
118 final_project_df['self_employed'].replace(np.nan, 'No', inplace=True)
119
120 # Converting the "self_employed" column to numerical values
121 final_project_df['self_employed'] = final_project_df['self_employed'].map({'Yes': 1, 'No': 0})
122
123 # Displaying the first few rows to verify the changes
124 #print(final_project_df[['self_employed']].head(15))
125 #print(final_project_df['self_employed'].unique())
126
127 # Generating a summary statistics for the "self_employed" column
128 self_employed_summary = final_project_df['self_employed'].describe()
129 #print("Summary statistics for self_employed:")
130 #print(self_employed_summary)
131
132 # Converting "treatment" column to numerical values
133 final_project_df['treatment'] = final_project_df['treatment'].map({'Yes': 1, 'No': 0})
134
135 # Generating a summary statistics for the "treatment" column
136 treatment_summary = final_project_df['treatment'].describe()
137 print("Summary statistics for treatment:")
138 print(treatment_summary)

```

Console 1/A

```

In [9]: runfile('C:/Users/Danny/Downloads/Mental Health/Python Final Project.py',
               wdir='C:/Users/Danny/Downloads/Mental Health')
Number of rows: 1259
Summary statistics for treatment:
count    1259.000000
mean     0.505957
std      0.500163
min      0.000000
25%     0.000000
50%     1.000000
75%     1.000000
max     1.000000
Name: treatment, dtype: float64

```

In [10]:

iPython Console History

conda (Python 3.11.7) Completions: conda LSP: Python Line 139, Col 1 ASCII CRLF RW Mem 61%

58°F Party cloudy

Search

9:59 PM 5/17/2024

```
# Converting "treatment" column to numerical values
final_project_df['treatment'] = final_project_df['treatment'].map({'Yes': 1, 'No': 0})

# Generating a summary statistics for the "treatment" column
treatment_summary = final_project_df['treatment'].describe()
print("Summary statistics for treatment:")
print(treatment_summary)
```

```
Number of rows: 1259
Summary statistics for treatment:
count      1259.000000
mean       0.505957
std        0.500163
min       0.000000
25%       0.000000
50%       1.000000
75%       1.000000
max       1.000000
Name: treatment, dtype: float64
```

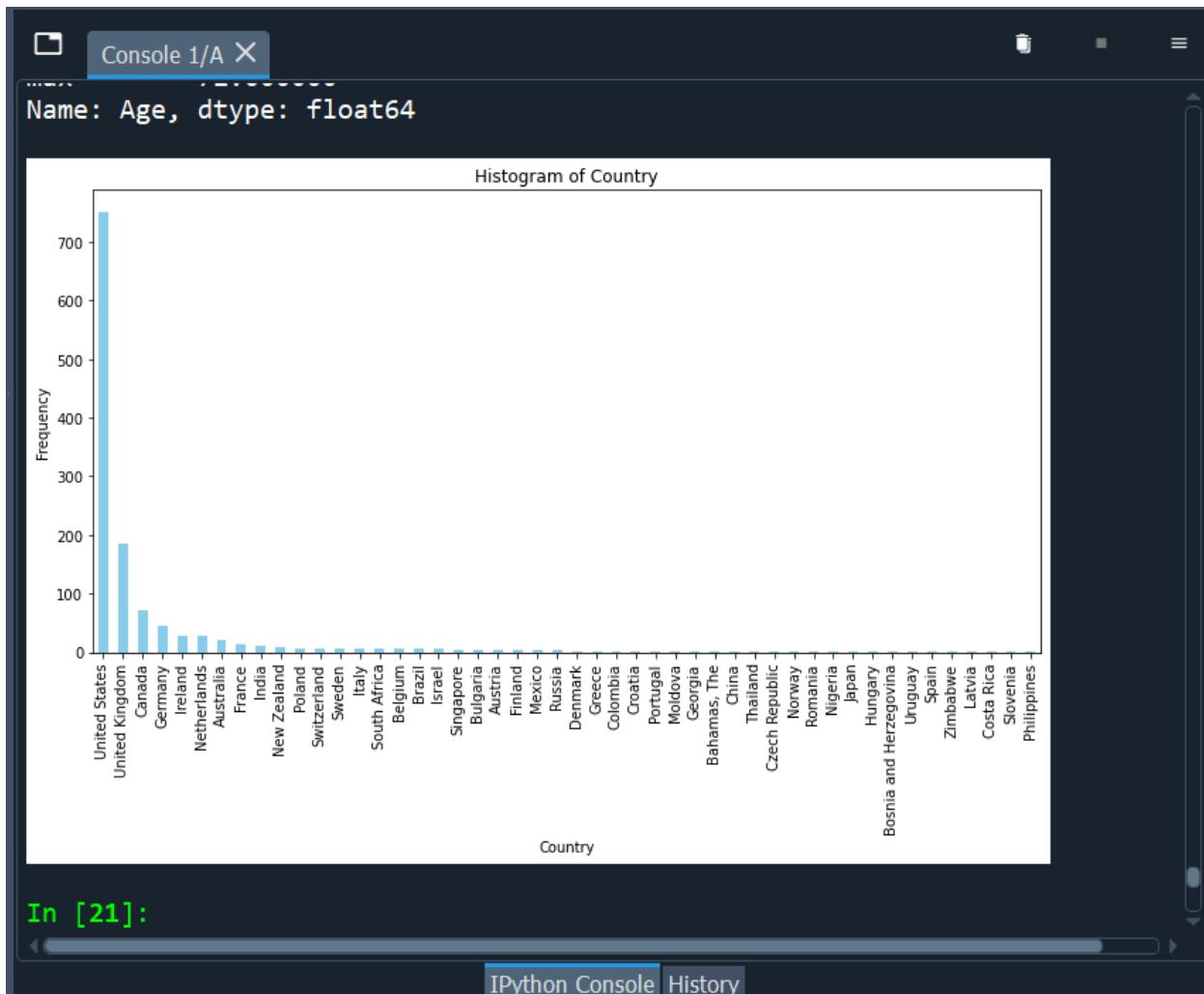
The summary statistics for the treatment column indicate that out of 1,259 respondents, approximately 50.6% have sought treatment for mental health issues, as reflected by the mean value of 0.506. The distribution of responses is fairly balanced, with the standard deviation of 0.500 showing typical variability for the data. The minimum value is 0 (representing "No"), while the maximum value is 1 (representing "Yes"). Notably, the median and the 75th percentile are both 1, suggesting that more than half of the respondents have sought treatment. This indicates a slight skew towards respondents being more likely to seek treatment, highlighting significant mental health treatment engagement within the dataset.

Analysis & Visualizations

Visualization 1:

Which country has the highest frequency in mental health identification all over the world?

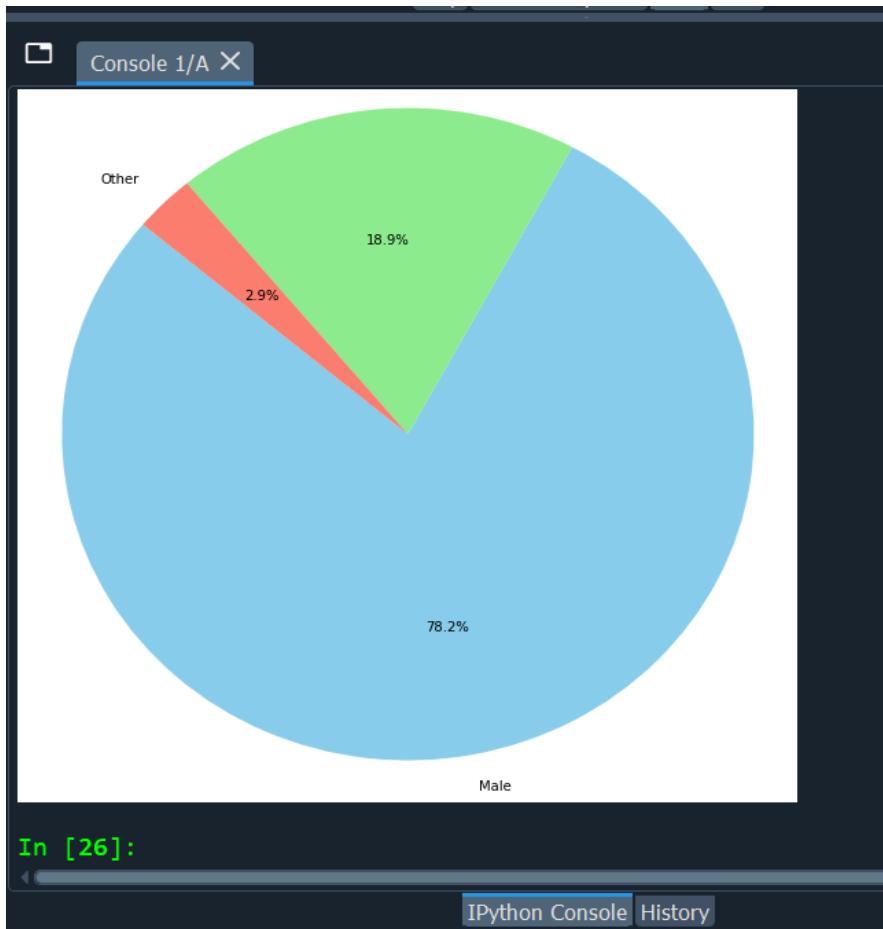
```
128
129     # Plot the histogram using Matplotlib
130     plt.figure(figsize=(12, 6))
131     country_counts.plot(kind='bar', color='skyblue')
132     plt.title('Histogram of Country')
133     plt.xlabel('Country')
134     plt.ylabel('Frequency')
135     plt.show()
136
137
```



The bar chart above represents the distribution of survey respondents across different countries. Each bar in the chart corresponds to a specific country, with the length of the bar indicating the number of respondents from that country. The x-axis displays the countries included in the dataset, while the y-axis represents the count of respondents. The color choice of sky blue provides a visually appealing and easily distinguishable representation of the data. By examining the heights of the bars, we can quickly identify which countries have the highest and lowest numbers of respondents in the survey. The USA is at the top of the list followed by the United Kingdom and Canada. The difference between the responders from the USA and UK is remarkable as the USA is more than 700 and the UK is close to 200.

Visualization 2: Comparison of mental health issues among genders in Tech.

```
136
137 # Count the occurrences of each gender in the "Gender" column
138 gender_counts = final_project_df['Gender'].value_counts()
139
140 # Plot the pie chart using Matplotlib
141 #plt.figure(figsize=(8, 8))
142 gender_counts.plot(kind='pie', autopct='%.1f%%', startangle=140, colors=['skyblue', 'Lightgreen', 'salmon'])
143 plt.title('Pie Chart of Gender Distribution')
144 plt.ylabel('')
145 plt.axis('equal') # Ensure the pie is drawn as a circle
146 plt.tight_layout() # Adjust layout
147 #plt.show()
148
```

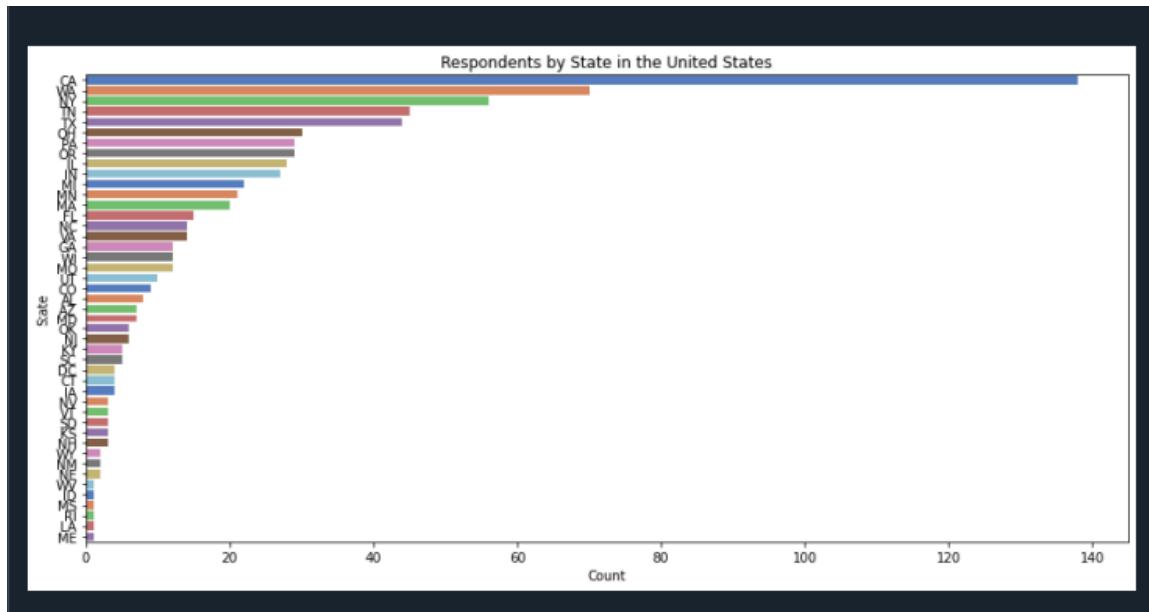


The pie chart illustrates the gender distribution among respondents in our dataset. The majority of respondents were male represented in color blue, comprising 78.2% of the total, followed by female in color green respondents at 18.9%. Additionally, a smaller proportion of respondents identify as non-binary, genderqueer, or other gender identities, making up 2.9%. The predominance of male respondents suggests potential gender biases in survey participation or broader population tally trends, highlighting the importance of inclusive data collection practices. Recognizing and understanding the diversity of gender identities represented in our dataset is essential for promoting inclusivity and ensuring that our analyses and decision-making processes are representative of all voices.

Visualization 3

What are the stats of respondents among states in the dominant country in question 1?

```
149  
150 # Filter the data for respondents from the United States  
151 us_data = final_project_df[final_project_df['Country'] == 'United States']  
152  
153 # Count the occurrences of each state in the filtered data  
154 state_counts = us_data['state'].value_counts()  
155  
156 # Plot the bar plot using Seaborn  
157 plt.figure(figsize=(12, 6))  
158 sns.barplot(x=state_counts.values, y=state_counts.index, palette='muted')  
159 plt.title('Respondents by State in the United States')  
160 plt.xlabel('Count')  
161 plt.ylabel('State')  
162 plt.tight_layout()  
163 plt.show()  
164
```



The bar plot illustrates the distribution of survey respondents across various states in the United States. Each bar represents a specific state, with the length of the bar corresponding to the count of respondents from that state. The most dominant state is California followed by Washington and New York. This visualization provides a clear comparison of respondent distribution across states, allowing for easy identification of states with the highest and lowest

participation rates in the survey. By examining the heights of the bars, we can discern which states contributed the most to the dataset and which states had comparatively lower representation. Additionally, the bar plot enables us to identify any outliers or anomalies in the data, such as states with exceptionally high or low respondent counts, which may warrant further investigation or analysis. Overall, the bar plot offers valuable insights into the geographic distribution of survey respondents, aiding in understanding the regional representation and statistical composition of the dataset.

References

- 1) *Mental Health Analysis in Tech Workplace - IEOM*,
ieomsociety.org/proceedings/2022orlando/495.pdf.
- 2) Rawal, Jahnvi. "Mental Health in Tech Survey." *Medium*, Medium, 1 Sept. 2022,
medium.com/@jahnvirawal/mental-health-in-tech-survey-c059e4db87a2.
- 3) Uddin, Md Milon, et al. "Mental Health Analysis in Tech Workplace." *IEOM Society*, IEOM Society, 11 June 2022, index.ieomsociety.org/index.cfm/article/view/ID/3684.