# Rama_Rao_Vydadi_50604256

October 8, 2024

#### 0.0.1 Problem Statement :

#### 0.0.2 This analysis aims to uncover potential causes of mental health disorders by examining the behavioral patterns and histories of affected patients, ### leading to insights for better interventions.

[1]:
```
import pandas as pd
import os
from typing import List, Optional, Dict
import gc
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# !pip3 install pandas
#!pip3 install pyarrow
# !pip3 install pandas
#!pip3 install fastparquet
```

#### 0.0.3 Data Fetch

[3]:
```
def fetch_nsduh_data(year: int) -> Optional[pd.DataFrame]:
    url_placeholder = "https://www.datafiles.samhsa.gov/sites/default/files/
 ↪field-uploads-protected/studies/NSDUH-{year}/NSDUH-{year}-datasets/
 ↪NSDUH-{year}-DS0001/NSDUH-{year}-DS0001-bundles-with-study-info/
 ↪NSDUH-{year}-DS0001-bndl-data-tsv.zip"

    try:
        url = url_placeholder.format(year=year)
        df = pd.read_csv(url, compression='zip', sep='\t', low_memory=False)
        return df
    except Exception as e:
        print(f"An error occurred for year {year}: {e}")
        return None
```

[4]:
```
def check_parquet_exists(years: List[int], output_dir: str) -> Dict[int, bool]:
    existence_check = {}
    for year in years:
        year_path = os.path.join(output_dir, f'year={year}')
        existence_check[year] = os.path.exists(year_path)

    return existence_check
```

[5]:
```
def write_parquet(df: pd.DataFrame, year: int, output_dir: str, overwrite: bool␣
↪= False) -> None:
    try:
        year_dir = os.path.join(output_dir, f'year={year}')
        if overwrite and os.path.exists(year_dir):
            shutil.rmtree(year_dir)  # Remove existing directory to start fresh

        df['year'] = year  # Add the year column for partitioning
        # Write data to Parquet format with partitioning
        df.to_parquet(output_dir, partition_cols=['year'], index=False)

        print(f"Data for year {year} successfully saved to Parquet format in␣
↪{output_dir}")
    except Exception as e:
        print(f"Error saving data to Parquet for year {year}: {e}")
```

[6]:
```
def data_fetch(years_to_fetch: List[int], output_dir: str, overwrite: bool =␣
↪False) -> None:
    try:
        for year in years_to_fetch:
            if not overwrite and os.path.exists(os.path.join(output_dir,␣
↪f'year={year}')):
                print(f"Data for year {year} already exists. Skipping.")
                continue

            df = fetch_nsduh_data(year)
            if df is not None:
                print(f"Successfully fetched data for year: {year}")
                write_parquet(df, year, output_dir, overwrite)
                del df  # Remove the DataFrame from memory
                gc.collect()  # Force garbage collection

        print("All requested years processed.")
    except Exception as e:
        print(f"An unexpected error occurred in the data_fetch function: {e}")
```

[7]:
```
def read_parquet(input_dir: str, years: Optional[List[int]] = None) -> pd.
↪DataFrame:
    data_frames = []
    available_years = [int(d.split('=')[1]) for d in os.listdir(input_dir) if d.
↪startswith('year=')]
```

```
    years_to_read = years if years is not None else available_years

    for year in years_to_read:
        year_path = os.path.join(input_dir, f'year={year}')
        if os.path.exists(year_path):
            df = pd.read_parquet(year_path)

            # Convert data types to reduce memory usage
            for col in df.select_dtypes(include=['float64']).columns:
                df[col] = df[col].astype('float32')

            data_frames.append(df)
        else:
            print(f"Warning: No data found for year {year}")

    if data_frames:
        combined_df = pd.concat(data_frames, ignore_index=True)
        return combined_df
    else:
        print("Warning: No data was loaded.")
        return pd.DataFrame()
```

[8]:
```
if __name__ == "__main__":
    years = [2015, 2016, 2017, 2018, 2019]
    output_directory = "../data/DS/NSDUH"
    data_fetch(years, output_directory, overwrite=False)
    for year in years:
        df = read_parquet(output_directory, [year])
        if year in df:
            print(f"Data for year {year}:")
            print(df[year].head())
        #del df
        gc.collect()
```

```
Data for year 2015 already exists. Skipping.
Data for year 2016 already exists. Skipping.
Data for year 2017 already exists. Skipping.
Data for year 2018 already exists. Skipping.
Data for year 2019 already exists. Skipping.
All requested years processed.
```

#### 0.0.4 Name : Rama Rao Vydadi

#### 0.0.5 Person Number : 50604256

#### 0.0.6 Question 1: How does socioeconomic status (income, education, employment status) influence the likelihood of experiencing mental health disorders?

#### 0.0.7 Hypothesis 1: Individuals with lower income are more likely to experience mental health issues or Unemployment is a significant predictor of mental health disorders

#### 0.0.8 Significance of the question: Understanding the correlation between socioeconomic factors and mental health can help identify vulnerable populations and provide support for economically weak population.

#### 0.0.9 Question2: What role does marijuana use play in the aggravation of mental health disorders?

#### 0.0.10 Hypothesis 2: Frequent marijuana use is more common in states where marijuana is legalized, and this is associated with a higher prevalence of depression.

#### 0.0.11 • Understanding the relationship between marijuana use and mental health will help us to answer sensitive questions like should marijuana be banned all over the world

#### 0.0.12 Question 3: What role does hallucinogens play in mental health disorders?

#### 0.0.13 Hypothesis 3: Unregulated use of hallucinogens can cause mental health problems like anxiety and depression.

#### 0.0.14 • Understanding the relationship between use of various hallucinogens and mental health will suggest us to regulate the supply of medical hallucinogens

#We will first perform the basic data cleaning steps and then perform EDA

```
[9]: df.head()
```

```
[9]:    QUESTID2    FILEDATE  CIGEVER  CIGOFRSM  CIGWILYR  CIGTRY  CIGYFU  CIGMFU  \
     0  43295143  10/09/2020        1        99        99      13    9999      99
     1  65095143  10/09/2020        2        99        99     991    9991      91
     2  49405143  10/09/2020        1        99        99      22    9999      99
     3  51015143  10/09/2020        2        99        99     991    9991      91
     4  31825143  10/09/2020        2        99        99     991    9991      91

        CIGREC  CIG30USE  …  POVERTY3  TOOLONG  TROUBUND  PDEN10  COUTYP4  \
     0       4        93  …       3.0        2         2       2        2
     1      91        91  …       3.0        2         2       2        2
     2       4        93  …       3.0        2         2       2        2
     3      91        91  …       1.0        2         2       2        2
     4      91        91  …       3.0        2         2       2        2

        MAIIN102  AIIND102     ANALWT_C  VESTR  VEREP
     0         2         2  6613.865723  40004      2
```

```
1          2          2  6321.580566  40003        1
2          2          2  5045.607422  40008        1
3          2          2  2419.558838  40031        1
4          2          2   575.225464  40010        2

[5 rows x 2741 columns]
```

[10]:
```
df.shape
```

[10]:
```
(56136, 2741)
```

[11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56136 entries, 0 to 56135
Columns: 2741 entries, QUESTID2 to VEREP
dtypes: float32(407), int64(2332), object(2)
memory usage: 1.1+ GB
```

[12]:
```
df.describe()
```

[12]:
```
           QUESTID2       CIGEVER      CIGOFRSM      CIGWILYR        CIGTRY  \
count  5.613600e+04  56136.000000  56136.000000  56136.000000  56136.000000
mean   5.434607e+07      1.542700     78.595554     78.605458    550.575816
std    2.563167e+07      0.498178     39.056828     39.037478    485.236660
min    1.000945e+07      1.000000      1.000000      1.000000      1.000000
25%    3.198245e+07      1.000000     99.000000     99.000000     16.000000
50%    5.403939e+07      2.000000     99.000000     99.000000    991.000000
75%    7.625105e+07      2.000000     99.000000     99.000000    991.000000
max    9.999669e+07      2.000000     99.000000     99.000000    997.000000

             CIGYFU        CIGMFU        CIGREC      CIG30USE       CG30EST  \
count  56136.000000  56136.000000  56136.000000  56136.000000  56136.000000
mean    9840.048846     92.951600     50.651080     80.111248     92.784915
std     1099.156156     12.225758     43.980321     26.734751      4.208800
min     2017.000000      1.000000      1.000000      1.000000      1.000000
25%     9991.000000     91.000000      3.000000     91.000000     91.000000
50%     9991.000000     91.000000     91.000000     91.000000     91.000000
75%     9999.000000     99.000000     91.000000     93.000000     93.000000
max     9999.000000     99.000000     91.000000     98.000000     99.000000

       …      POVERTY3       TOOLONG      TROUBUND        PDEN10  \
count  …  55609.000000  56136.000000  56136.000000  56136.000000
mean   …      2.427179      2.173650      2.199854      1.662997
std    …      0.776759      4.797938      4.794185      0.627146
min    …      1.000000      1.000000      1.000000      1.000000
25%    …      2.000000      2.000000      2.000000      1.000000
50%    …      3.000000      2.000000      2.000000      2.000000
```

```
75%    …      3.000000      2.000000      2.000000      2.000000
max    …      3.000000     98.000000     98.000000      3.000000

            COUTYP4       MAIIN102       AIIND102       ANALWT_C          VESTR  \
count  56136.000000  56136.000000  56136.000000  56136.000000  56136.000000
mean       1.747827      1.982827      1.982560   4902.758301  40025.570899
std        0.762371      0.129915      0.130904   5952.114746     14.388113
min        1.000000      1.000000      1.000000      3.581148  40001.000000
25%        1.000000      2.000000      2.000000   1262.476593  40013.000000
50%        2.000000      2.000000      2.000000   2855.374878  40025.000000
75%        2.000000      2.000000      2.000000   6076.500732  40038.000000
max        3.000000      2.000000      2.000000  77284.484375  40050.000000

              VEREP
count  56136.000000
mean       1.504400
std        0.499985
min        1.000000
25%        1.000000
50%        2.000000
75%        2.000000
max        2.000000

[8 rows x 2739 columns]
```

[13]:
```
df.columns
```

[13]:
```
Index(['QUESTID2', 'FILEDATE', 'CIGEVER', 'CIGOFRSM', 'CIGWILYR', 'CIGTRY',
       'CIGYFU', 'CIGMFU', 'CIGREC', 'CIG30USE',
       …
       'POVERTY3', 'TOOLONG', 'TROUBUND', 'PDEN10', 'COUTYP4', 'MAIIN102',
       'AIIND102', 'ANALWT_C', 'VESTR', 'VEREP'],
      dtype='object', length=2741)
```

[14]:
```
filtered_df=df.copy()
```

Selecting only required columns from the entire dataset (This is an iterative step after performing EDA)

[15]:
```
desired_columns = [
    'QUESTID2', 'IRWRKSTAT', 'IREDUHIGHST2', 'INCOME', 'IRSEX', 'MJEVER',
    'PNRANYLIF', 'COUTYP4', 'MEDMJPA2', 'DSTCHR30', 'ADDPREV', 'LSD', 'PCP',␣
  ↪'PEYOTE', 'MESC', 'PSILCY', 'ECSTMOLLY', 'KETMINESK', 'DMTAMTFXY',␣
  ↪'SALVIADIV', 'HALLUCOTH'
]

filtered_df = df[desired_columns]
```

[16]:
```
#We have various hallucinogens in our dataset we can standardize the mapping␣
↪and remove the unwanted data
#like "Dont know", "Refused" etc with nan values
```

[17]:
```
lsd_mapping = {
    1: 'Yes',  # "Yes"
    2: 'No',   # "No"
    3: 'Yes',  # "Yes logically assigned"
    91: 'No',  # "Never used hallucinogens"
    94: np.nan,  # "Don't know" replaced with NaN
    97: np.nan   # "Refused" replaced with NaN
}

# Apply the mapping to the 'LSD' column
filtered_df['LSD'] = filtered_df['LSD'].map(lsd_mapping)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\1422645912.py:11:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df['LSD'] = filtered_df['LSD'].map(lsd_mapping)
```

[18]:
```
#we can create similar mapping for all hallucinogens
```

[19]:
```
substance_mapping = {
    1: 'Yes',  # "Yes"
    2: 'No',   # "No"
    91: 'No',  # "Never used hallucinogens"
    94: np.nan,  # "Don't know" replaced with NaN
    97: np.nan   # "Refused" replaced with NaN
}

substance_columns = ['PCP', 'PEYOTE', 'MESC', 'PSILCY', 'ECSTMOLLY',␣
↪'KETMINESK', 'DMTAMTFXY', 'SALVIADIV', 'HALLUCOTH']

for col in substance_columns:
    filtered_df[col] = filtered_df[col].map(substance_mapping)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\497884309.py:12:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  filtered_df[col] = filtered_df[col].map(substance_mapping)
```

[20]:
```
# Since we have a lot of hallucinogens we can generalize the use of them by␣
  ↪creating a new column which will have yes if any one of the
#hallucinogens in dataset is used
```

[21]:
```
hallucinogen_columns = ['LSD', 'PCP', 'PEYOTE', 'MESC', 'PSILCY', 'ECSTMOLLY',␣
  ↪'KETMINESK', 'DMTAMTFXY', 'SALVIADIV', 'HALLUCOTH']

filtered_df['hallucinogens'] = filtered_df[hallucinogen_columns].apply(
    lambda row: 'Yes' if 'Yes' in row.values else ('No' if all(val == 'No' for␣
  ↪val in row.values) else np.nan), axis=1
)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\3382138147.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df['hallucinogens'] = filtered_df[hallucinogen_columns].apply(
```

[22]:
```
# Now only considering hallucination column created in filtered_df
desired_columns = [
    'QUESTID2', 'IRWRKSTAT', 'IREDUHIGHST2', 'INCOME', 'IRSEX', 'MJEVER',
    'PNRANYLIF', 'COUTYP4', 'MEDMJPA2', 'DSTCHR30', 'ADDPREV', 'hallucinogens' ]

filtered_df = filtered_df[desired_columns]
```

### 0.0.15 Data Cleaning

[23]:
```
filtered_df.head()
```

[23]:
```
   QUESTID2  IRWRKSTAT  IREDUHIGHST2  INCOME  IRSEX  MJEVER  PNRANYLIF  \
0  43295143          1            11       4      1       1          1
1  65095143          1            11       4      2       2          1
2  49405143          1            11       4      1       1          1
3  51015143          4             6       1      2       2          1
4  31825143          4             7       4      1       2          2

   COUTYP4  MEDMJPA2  DSTCHR30  ADDPREV hallucinogens
0        2         2         3        1            No
1        2         1         5        1            No
2        2         1         5        2            No
3        2         1         4        2            No
4        2         1         4        1            No
```

### 0.0.16 Handling Special codes like Bad Data, Legitimate Skip etc in depression column which does not add value to the analysis

Replace special codes which does not add value with NaN values 85: 'Bad Data', 97: 'Refused', 98: 'Blank', 99: 'Legitimate Skip'

[24]:
```
filtered_df['ADDPREV'].replace([85, 97, 98, 99], np.nan, inplace=True)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\2321724339.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  filtered_df['ADDPREV'].replace([85, 97, 98, 99], np.nan, inplace=True)
```

[25]:
```
filtered_df['ADDPREV'].unique()
```

[25]:
```
array([ 1.,  2., nan, 94.])
```

In the IRWRKSTAT (EMPLOYMENT STATUS) column - 99 indicates 12-14 year olds, this data might not be useful for analysis hence we can remove this data as these people might also be having some income, since we cannot say the exact income level it is better to drop these values

[26]:
```
filtered_df['IRWRKSTAT'].replace([99], np.nan, inplace=True)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\753615126.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  filtered_df['IRWRKSTAT'].replace([99], np.nan, inplace=True)
```

Since we have more than 11 levels of education in the dataset, we can categorize the education levels into 3 different levels which make more sense i.e Primary, Secondary and Higher education based on the grade studying

[27]:
```
# Define a function to categorize the education levels
def categorize_education(value):
    if value < 7:
        return 'primary education'
    elif value in [8, 9]:
        return 'intermediate education'
    else:
        return 'higher education'

filtered_df['IREDUHIGHST2'] = filtered_df['IREDUHIGHST2'].
 ↪apply(categorize_education)
```

[28]:
```
filtered_df.head()
```

[28]:
```
   QUESTID2  IRWRKSTAT        IREDUHIGHST2  INCOME  IRSEX  MJEVER  PNRANYLIF  \
0  43295143        1.0   higher education       4      1       1          1
1  65095143        1.0   higher education       4      2       2          1
2  49405143        1.0   higher education       4      1       1          1
3  51015143        4.0  primary education       1      2       2          1
4  31825143        4.0   higher education       4      1       2          2

   COUTYP4  MEDMJPA2  DSTCHR30  ADDPREV hallucinogens
0        2         2         3      1.0            No
1        2         1         5      1.0            No
2        2         1         5      2.0            No
3        2         1         4      2.0            No
4        2         1         4      1.0            No
```

MJEVER contains 94 and 97 which are dont know and refused to answer, these rows are not useful in our analysis hence we can exclude this data

[29]:
```
filtered_df['MJEVER'].replace([94, 97], np.nan, inplace=True)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\3512628858.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  filtered_df['MJEVER'].replace([94, 97], np.nan, inplace=True)
```

```
[30]: #Pain releiver use in lifetime 5 indicates logically assigned yes which means␣
       ↪this value is logically made as yes, so we can replace this
      #with 1 in the final data
```

```
[31]: filtered_df['PNRANYLIF'].replace(5, 1, inplace=True)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\422161004.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  filtered_df['PNRANYLIF'].replace(5, 1, inplace=True)
```

```
[32]: #replcaing values which are not useful with NA
      filtered_df['PNRANYLIF'].replace([94, 97, 98], np.nan, inplace=True)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\3256104701.py:2:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  filtered_df['PNRANYLIF'].replace([94, 97, 98], np.nan, inplace=True)
```

In the column How often do you feel sad, there are many columns which cannot be used for analysis we can replace them with NaN

```
[33]: #replcaing values which are not useful with NA
      filtered_df['DSTCHR30'].replace([85, 94, 97, 98, 99], np.nan, inplace=True)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\1097070963.py:2:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  filtered_df['DSTCHR30'].replace([85, 94, 97, 98, 99], np.nan, inplace=True)
```

[34]:
```
filtered_df.head()
```

[34]:
```
   QUESTID2  IRWRKSTAT        IREDUHIGHST2  INCOME  IRSEX  MJEVER  PNRANYLIF  \
0  43295143        1.0    higher education       4      1     1.0        1.0
1  65095143        1.0    higher education       4      2     2.0        1.0
2  49405143        1.0    higher education       4      1     1.0        1.0
3  51015143        4.0   primary education       1      2     2.0        1.0
4  31825143        4.0    higher education       4      1     2.0        2.0

   COUTYP4  MEDMJPA2  DSTCHR30  ADDPREV hallucinogens
0        2         2       3.0      1.0            No
1        2         1       5.0      1.0            No
2        2         1       5.0      2.0            No
3        2         1       4.0      2.0            No
4        2         1       4.0      1.0            No
```

[35]:
```
#Removal of duplicates from the data frame
filtered_df.drop_duplicates(inplace=True)
```

[36]:
```
#Set display options to have 2 decimals to have proper scale for future␣
 ↪operations
pd.options.display.float_format = '{:.2f}'.format
```

[37]:
```
filtered_df.describe()
```

[37]:
```
         QUESTID2  IRWRKSTAT   INCOME    IRSEX   MJEVER  PNRANYLIF  COUTYP4  \
count    56136.00   49581.00 56136.00 56136.00 56097.00   55670.00 56136.00
mean  54346070.01       2.22     2.72     1.52     1.56       1.49     1.75
std   25631667.47       1.30     1.14     0.50     0.50       0.50     0.76
min   10009454.00       1.00     1.00     1.00     1.00       1.00     1.00
25%   31982452.50       1.00     2.00     1.00     1.00       1.00     1.00
50%   54039390.00       2.00     3.00     2.00     2.00       1.00     2.00
75%   76251052.50       4.00     4.00     2.00     2.00       2.00     2.00
max   99996688.00       4.00     4.00     2.00     2.00       2.00     3.00

       MEDMJPA2  DSTCHR30  ADDPREV
count  56136.00  42411.00 42508.00
mean       1.30      4.37     1.89
std        0.46      0.98     4.59
min        1.00      1.00     1.00
```

```
25%          1.00      4.00     1.00
50%          1.00      5.00     2.00
75%          2.00      5.00     2.00
max          2.00      5.00    94.00
```

[38]:
```
education_mapping = {
    'higher education': 3,
    'primary education': 1,
    'intermediate education': 2
}
filtered_df['IREDUHIGHST2'] = filtered_df['IREDUHIGHST2'].
  ↪replace(education_mapping)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\4283032743.py:6:
FutureWarning: Downcasting behavior in `replace` is deprecated and will be
removed in a future version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  filtered_df['IREDUHIGHST2'] =
filtered_df['IREDUHIGHST2'].replace(education_mapping)
```

[39]:
```
filtered_df.head()
```

[39]:
```
   QUESTID2  IRWRKSTAT  IREDUHIGHST2  INCOME  IRSEX  MJEVER  PNRANYLIF  \
0  43295143       1.00             3       4      1    1.00       1.00
1  65095143       1.00             3       4      2    2.00       1.00
2  49405143       1.00             3       4      1    1.00       1.00
3  51015143       4.00             1       1      2    2.00       1.00
4  31825143       4.00             3       4      1    2.00       2.00

   COUTYP4  MEDMJPA2  DSTCHR30  ADDPREV hallucinogens
0        2         2      3.00     1.00            No
1        2         1      5.00     1.00            No
2        2         1      5.00     2.00            No
3        2         1      4.00     2.00            No
4        2         1      4.00     1.00            No
```

[40]:
```
filtered_df.shape
```

[40]:
```
(56136, 12)
```

[41]:
```
#Since we have already replaced unnecessary values with NaN we can drop these␣
  ↪values as they are no longer useful
```

[42]:
```
null_values = filtered_df.isnull().sum()

print(null_values)
```

```
QUESTID2             0
IRWRKSTAT         6555
IREDUHIGHST2         0
INCOME               0
IRSEX                0
MJEVER              39
PNRANYLIF          466
COUTYP4              0
MEDMJPA2             0
DSTCHR30         13725
ADDPREV          13628
hallucinogens      245
dtype: int64
```

[43]:
```
filtered_df_cleaned = filtered_df.dropna(subset=['IRWRKSTAT', 'DSTCHR30',␣
↪'ADDPREV', 'PNRANYLIF','MJEVER' ])
```

[44]:
```
filtered_df_cleaned.shape
```

[44]:
```
(42166, 12)
```

[45]:
```
filtered_df_cleaned['IRWRKSTAT'].unique()
```

[45]:
```
array([1., 4., 3., 2.])
```

[46]:
```
irwrkstat_mapping = {
    1: 'Employed full time',
    2: 'Employed part time',
    3: 'Unemployed',
    4: 'Other (incl. not in labor force)',
    99: '12-14 year olds'
}

filtered_df_cleaned['IRWRKSTAT'] = filtered_df_cleaned['IRWRKSTAT'].
↪map(irwrkstat_mapping)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\2354628446.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df_cleaned['IRWRKSTAT'] =
filtered_df_cleaned['IRWRKSTAT'].map(irwrkstat_mapping)
```

[47]:
```
filtered_df_cleaned[['IRWRKSTAT']].head()
```

```
[47]:                            IRWRKSTAT
      0                 Employed full time
      1                 Employed full time
      2                 Employed full time
      3   Other (incl. not in labor force)
      4   Other (incl. not in labor force)
```

```
[48]: income_mapping = {
          1: 'Less than $20,000',
          2: '$20,000 - $49,999',
          3: '$50,000 - $74,999',
          4: '$75,000 or more'
      }

      filtered_df_cleaned['INCOME'] = filtered_df_cleaned['INCOME'].
       ↪map(income_mapping)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\4064115886.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df_cleaned['INCOME'] =
filtered_df_cleaned['INCOME'].map(income_mapping)
```

```
[49]: filtered_df_cleaned[['INCOME']].head()
```

```
[49]:               INCOME
      0    $75,000 or more
      1    $75,000 or more
      2    $75,000 or more
      3  Less than $20,000
      4    $75,000 or more
```

```
[50]: gender_mapping = {
          1: 'Male',
          2: 'Female'
      }

      filtered_df_cleaned['IRSEX'] = filtered_df_cleaned['IRSEX'].map(gender_mapping)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\4015784729.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df_cleaned['IRSEX'] =
filtered_df_cleaned['IRSEX'].map(gender_mapping)
```

[51]:
```
filtered_df_cleaned.head()
```

[51]:
```
   QUESTID2                        IRWRKSTAT  IREDUHIGHST2  \
0  43295143               Employed full time             3
1  65095143               Employed full time             3
2  49405143               Employed full time             3
3  51015143  Other (incl. not in labor force)             1
4  31825143  Other (incl. not in labor force)             3

              INCOME   IRSEX  MJEVER  PNRANYLIF  COUTYP4  MEDMJPA2  DSTCHR30  \
0    $75,000 or more    Male    1.00       1.00        2         2      3.00
1    $75,000 or more  Female    2.00       1.00        2         1      5.00
2    $75,000 or more    Male    1.00       1.00        2         1      5.00
3  Less than $20,000  Female    2.00       1.00        2         1      4.00
4    $75,000 or more    Male    2.00       2.00        2         1      4.00

   ADDPREV hallucinogens
0     1.00            No
1     1.00            No
2     2.00            No
3     2.00            No
4     1.00            No
```

[52]:
```
filtered_df_cleaned['IREDUHIGHST2'].unique()
```

[52]:
```
array([3, 1, 2])
```

[53]:
```
# Define a function to categorize the education levels
def categorize_education(value):
    if value==1:
        return 'primary education'
    elif value==2:
        return 'High School education'
    else:
        return 'College Degree'

filtered_df_cleaned['IREDUHIGHST2'] = filtered_df_cleaned['IREDUHIGHST2'].
  ↪apply(categorize_education)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\809615653.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df_cleaned['IREDUHIGHST2'] =
filtered_df_cleaned['IREDUHIGHST2'].apply(categorize_education)
```

[54]:
```
filtered_df_cleaned['IREDUHIGHST2'].unique()
```

[54]:
```
array(['College Degree', 'primary education', 'High School education'],
      dtype=object)
```

[55]:
```
mj_mapping = {
    1: 'Yes',
    2: 'No'
}

filtered_df_cleaned['MJEVER'] = filtered_df_cleaned['MJEVER'].map(mj_mapping)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\3622425528.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df_cleaned['MJEVER'] = filtered_df_cleaned['MJEVER'].map(mj_mapping)
```

[56]:
```
mj_mapping = {
    1.00: 'Yes',
    2.00: 'No'
}


filtered_df_cleaned['PNRANYLIF'] = filtered_df_cleaned['PNRANYLIF'].
 ↪map(mj_mapping)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\2351626243.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df_cleaned['PNRANYLIF'] =
filtered_df_cleaned['PNRANYLIF'].map(mj_mapping)
```

[57]:
```
metro_mapping = {
    1: 'Large Metro',
    2: 'Small Metro',
    3: 'Non Metro'
}


filtered_df_cleaned['COUTYP4'] = filtered_df_cleaned['COUTYP4'].
  ↪map(metro_mapping)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\1158216921.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df_cleaned['COUTYP4'] =
filtered_df_cleaned['COUTYP4'].map(metro_mapping)
```

[58]:
```
mapping = {
    1: 'All of the time',
    2: 'Most of the time',
    3: 'Some of the time',
    4: 'A little of the time',
    5: 'None of the time',
    99: 'Legitimate skip'
}
filtered_df_cleaned['DSTCHR30'] = filtered_df_cleaned['DSTCHR30'].map(mapping)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\2251930963.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df_cleaned['DSTCHR30'] = filtered_df_cleaned['DSTCHR30'].map(mapping)
```

[59]:
```
filtered_df_cleaned['ADDPREV'].unique()
```

[59]:
```
array([ 1.,  2., 94.])
```

[60]:
```
mapping = {
    1.00: 'Yes',
    2.00: 'No',
    94.00: 'No'
}
filtered_df_cleaned['ADDPREV'] = filtered_df_cleaned['ADDPREV'].map(mapping)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\3008980501.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df_cleaned['ADDPREV'] = filtered_df_cleaned['ADDPREV'].map(mapping)
```

[61]:
```
mapping = {
    1: 'Yes',
    2: 'No',
}
filtered_df_cleaned['MEDMJPA2'] = filtered_df_cleaned['MEDMJPA2'].map(mapping)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\88440773.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df_cleaned['MEDMJPA2'] = filtered_df_cleaned['MEDMJPA2'].map(mapping)
```

[ ]:
```

```

Renaming columns for better readability

[62]:
```
column_mapping = {
    'IRWRKSTAT': 'Employment',
    'IREDUHIGHST2': 'education',
    'INCOME': 'income',
    'IRSEX': 'sexual orientation',
    'MJEVER': 'Ever used marijuana',
    'PNRANYLIF': 'ANY PAIN RELIEVER USE IN LIFETIME',
    'COUTYP4': 'COUNTY METRO/NONMETRO STATUS',
    'MEDMJPA2': 'STATE MEDICAL MJ LAW PASSED AT TIME OF INTERVIEW',
    'DSTCHR30': 'HOW OFTEN FELT SAD NOTHING COULD CHEER YOU UP',
    'ADDPREV': 'SEVERAL DAYS OR LNGR WHEN FELT SAD/EMPTY/DPRSD'
}

filtered_df_cleaned.rename(columns=column_mapping, inplace=True)
```

```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\3523334497.py:14:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df_cleaned.rename(columns=column_mapping, inplace=True)
```

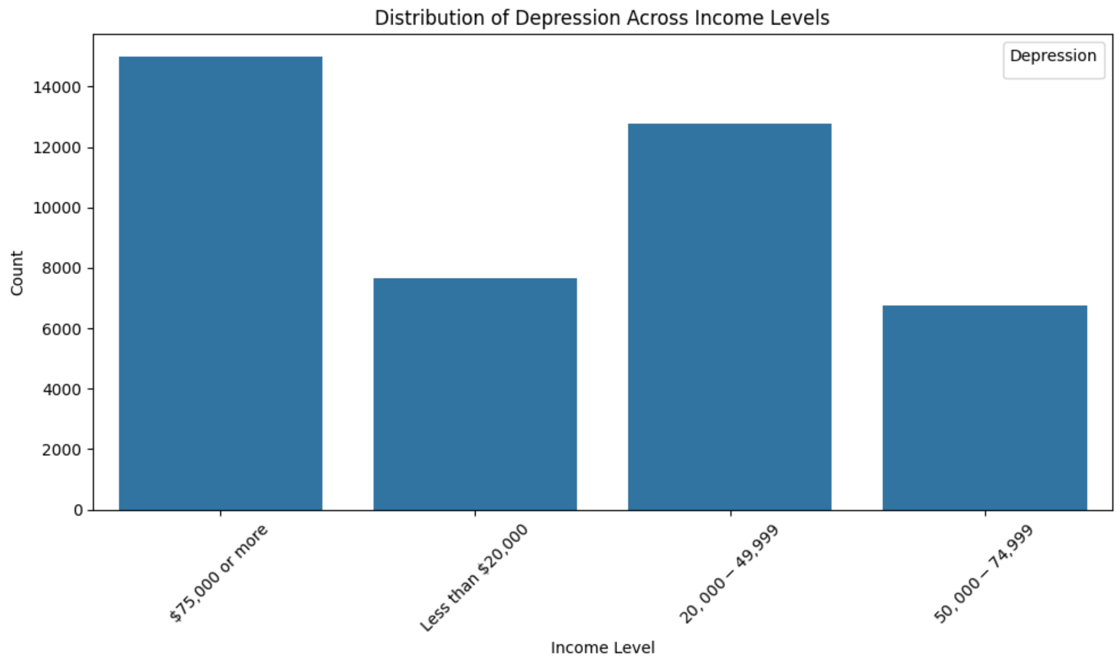
[63]:
```
filtered_df_cleaned.to_excel('filtered_df_cleaned.xlsx')
```

[64]:
```
filtered_df_cleaned.head()
```

[64]:
```
   QUESTID2                      Employment          education  \
0  43295143                Employed full time     College Degree
1  65095143                Employed full time     College Degree
2  49405143                Employed full time     College Degree
3  51015143  Other (incl. not in labor force)  primary education
4  31825143  Other (incl. not in labor force)     College Degree

              income sexual orientation Ever used marijuana  \
0    $75,000 or more               Male                 Yes
1    $75,000 or more             Female                  No
2    $75,000 or more               Male                 Yes
3  Less than $20,000             Female                  No
4    $75,000 or more               Male                  No

  ANY PAIN RELIEVER USE IN LIFETIME COUNTY METRO/NONMETRO STATUS  \
0                              Yes                   Small Metro
1                              Yes                   Small Metro
2                              Yes                   Small Metro
3                              Yes                   Small Metro
4                               No                   Small Metro

  STATE MEDICAL MJ LAW PASSED AT TIME OF INTERVIEW  \
0                                               No
1                                              Yes
2                                              Yes
3                                              Yes
4                                              Yes

  HOW OFTEN FELT SAD NOTHING COULD CHEER YOU UP  \
0                              Some of the time
1                              None of the time
2                              None of the time
3                          A little of the time
4                          A little of the time

  SEVERAL DAYS OR LNGR WHEN FELT SAD/EMPTY/DPRSD hallucinogens
0                                            Yes            No
1                                            Yes            No
2                                             No            No
3                                             No            No
4                                            Yes            No
```

[65]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=filtered_df_cleaned, x='income')
plt.title('Distribution of Depression Across Income Levels')
plt.xlabel('Income Level')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Depression')
plt.tight_layout()
plt.show()
```

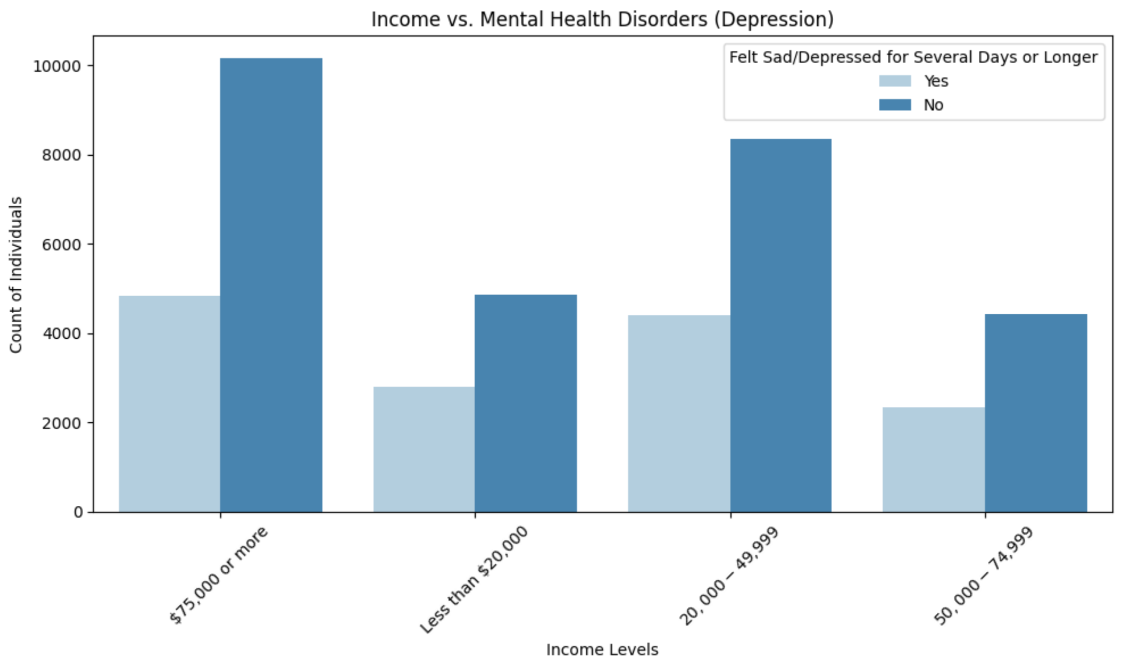
```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\1276217604.py:7:
UserWarning: No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is called with no
argument.
  plt.legend(title='Depression')
```



```
[ ]:
```

```
[66]: # Question 1: How does socioeconomic status (income, education, employment␣
        ↪status) influence mental health?
      # Bar plot to show the distribution of depression (Sadness) across different␣
        ↪income levels.

      # Create a plot for the influence of income on experiencing mental health␣
        ↪disorders (depression)
      plt.figure(figsize=(10, 6))
      sns.countplot(data=filtered_df_cleaned, x='income', hue='SEVERAL DAYS OR LNGR␣
        ↪WHEN FELT SAD/EMPTY/DPRSD', palette="Blues")
```

```
plt.title('Income vs. Mental Health Disorders (Depression)')
plt.xlabel('Income Levels')
plt.ylabel('Count of Individuals')
plt.xticks(rotation=45)
plt.legend(title='Felt Sad/Depressed for Several Days or Longer')
plt.tight_layout()
plt.show()
```

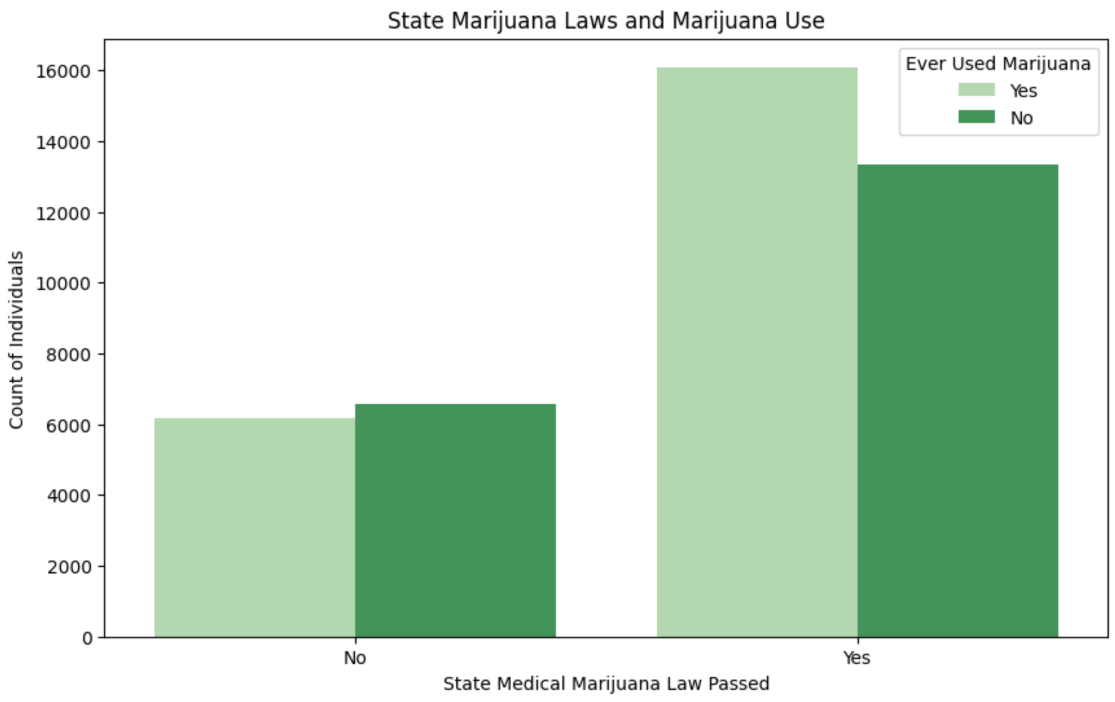


### 0.0.17 Observations:

# 1 From the above graphs we might conclude that people whose income is less than $20,000 (which is the least among the others) are more likely to

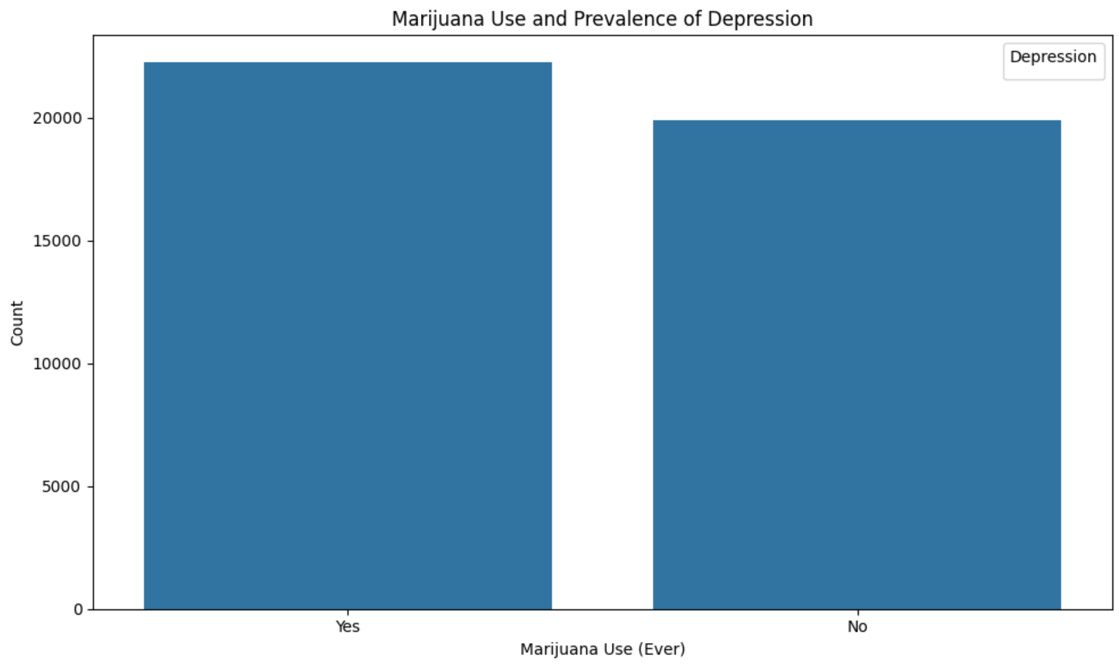
# 2 feel depressed.

[67]:
```
# Question 2: The role of marijuana use in mental health
# We will create a bar plot to see the relation between marijuana use, state␣
↪marijuana laws, and depression

plt.figure(figsize=(10, 6))
sns.countplot(data=filtered_df_cleaned, x='STATE MEDICAL MJ LAW PASSED AT TIME␣
↪OF INTERVIEW', hue='Ever used marijuana', palette="Greens")
plt.title('State Marijuana Laws and Marijuana Use')
plt.xlabel('State Medical Marijuana Law Passed')
plt.ylabel('Count of Individuals')
plt.legend(title='Ever Used Marijuana')
#plt.tight_layout()
plt.show()
```



[68]:
```
# Relationship between marijuana use and depression
plt.figure(figsize=(10, 6))
sns.countplot(data=filtered_df_cleaned, x='Ever used marijuana')
plt.title('Marijuana Use and Prevalence of Depression')
plt.xlabel('Marijuana Use (Ever)')
plt.ylabel('Count')
plt.legend(title='Depression')
plt.tight_layout()
plt.show()
```

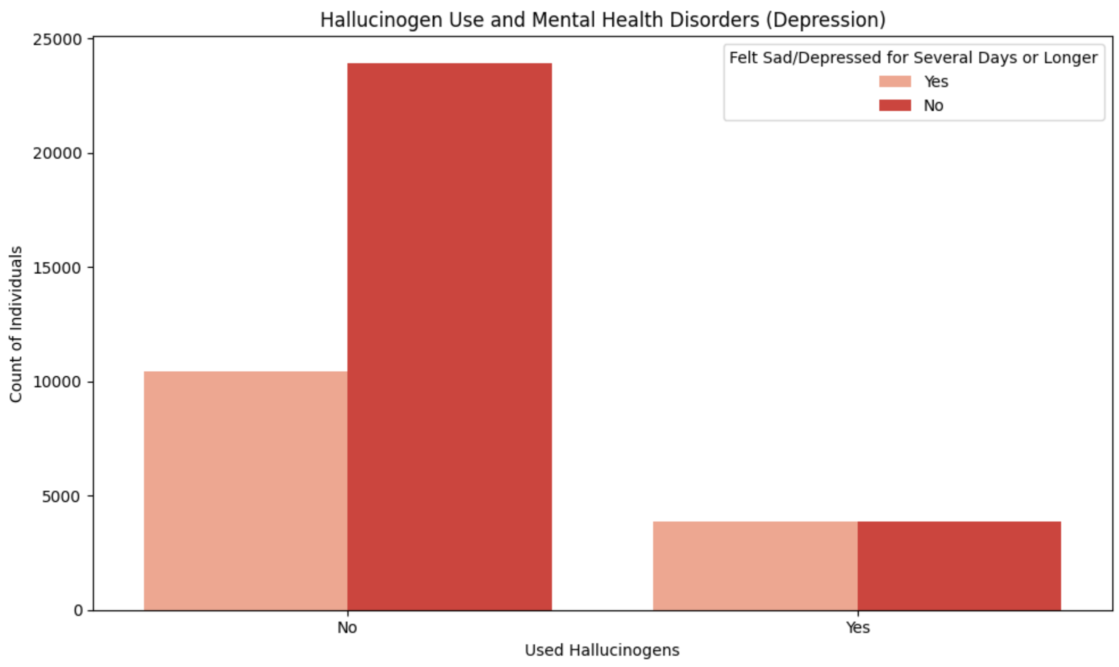
```
C:\Users\Rama Rao\AppData\Local\Temp\ipykernel_15216\1028580500.py:7:
UserWarning: No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is called with no
argument.
  plt.legend(title='Depression')
```



### 2.0.1 Observation:

# 3 We have observed that the locations where marijuana is legalized have higher usage of marijuana when compared to areas with stricter laws, and these regions with increased marijuana use also exhibit a higher prevalence of depression.

[69]:
```
# Question 3: The role of hallucinogen use in mental health
# We will create a plot to examine how hallucinogen use is related to␣
 ↪experiencing mental health disorders.

plt.figure(figsize=(10, 6))
sns.countplot(data=filtered_df_cleaned, x='hallucinogens', hue='SEVERAL DAYS OR␣
 ↪LNGR WHEN FELT SAD/EMPTY/DPRSD', palette="Reds")
plt.title('Hallucinogen Use and Mental Health Disorders (Depression)')
plt.xlabel('Used Hallucinogens')
plt.ylabel('Count of Individuals')
plt.legend(title='Felt Sad/Depressed for Several Days or Longer')
plt.tight_layout()
plt.show()
```



### 3.0.1 Observation: The above graph clearly shows that the percentage of people experiencing depression is higher among those who use hallucinogens compared to those who do not use them.

```
[ ]:
```