In [1]:
```
import sys, platform, psutil, subprocess

print("Python version:", sys.version.replace("\n", " "))
print("Python executable:", sys.executable)

print("OS:", platform.platform())
print("Machine:", platform.machine())
print("CPU:", platform.processor())
print("Physical cores:", psutil.cpu_count(logical=False))
print("Logical cores:", psutil.cpu_count(logical=True))
print("RAM (GB):", round(psutil.virtual_memory().total / (1024**3), 2))

try:
    out = subprocess.check_output(
        ["nvidia-smi", "--query-gpu=name,driver_version,memory.total", "--format=cs
        stderr=subprocess.STDOUT
    )
    print("GPU (NVIDIA):\n" + out.decode().strip())
except Exception:
    print("GPU (NVIDIA): Not found (or nvidia-smi not available).")
```

```
Python version: 3.10.19 | packaged by Anaconda, Inc. | (main, Oct 21 2025, 16:41:31)
[MSC v.1929 64 bit (AMD64)]
Python executable: C:\Users\123wi\anaconda3\envs\unsupervised\python.exe
OS: Windows-10-10.0.26200-SP0
Machine: AMD64
CPU: Intel64 Family 6 Model 191 Stepping 2, GenuineIntel
Physical cores: 16
Logical cores: 24
RAM (GB): 31.71
GPU (NVIDIA):
NVIDIA GeForce RTX 4070 Laptop GPU, 552.22, 8188 MiB
```

In [2]:
```
!pip install -q pandas numpy
```

In [3]:
```
#load the cvs data into a data frame
import pandas as pd
import numpy as np

file_path = r"C:\Users\123wi\OneDrive\Desktop\duits uni\unsupervised\project\mental


df = pd.read_csv(
    file_path,
    na_values=["", " ", "NA", "N/A", "n/a", "na", "null", "None"]
)


df = df.replace(r"^\s*$", np.nan, regex=True).fillna("N/A")

print("Shape:", df.shape)
df.head()
```

```
Shape: (1433, 63)
```

Out[3]:

	Are you self-employed?	How many employees does your company or organization have?	Is your employer primarily a tech company/organization?	Is your primary role within your company related to tech/IT?	Does your employer provide mental health benefits as part of healthcare coverage?	Do you know the options for mental health care available under your employer-provided coverage?	e disc exa com
0	0	26-100	1.0	N/A	Not eligible for coverage / N/A	N/A	
1	0	6-25	1.0	N/A	No	Yes	
2	0	6-25	1.0	N/A	No	N/A	
3	1	N/A	N/A	N/A	N/A	N/A	
4	0	6-25	0.0	1.0	Yes	Yes	

5 rows × 63 columns

In [4]:
```
# Collapse the "extra block" columns into existing employer-benefits/negative-conse

pct_col = "If yes, what percentage of your work time (time performing primary or se
prod_col = "Do you believe your productivity is ever affected by a mental health is

impact_cow_col = "If you have revealed a mental health issue to a coworker or emplo
neg_conseq_col = "Do you think that discussing a mental health disorder with your e

medcov_col = "Do you have medical coverage (private insurance or state-provided) wh
benefits_col = "Does your employer provide mental health benefits as part of health

extra_drop = [
    medcov_col,
    "Do you know local or online resources to seek help for a mental health disorde
    "If you have been diagnosed or treated for a mental health disorder, do you eve
    "If you have revealed a mental health issue to a client or business contact, do
```

```
    "If you have been diagnosed or treated for a mental health disorder, do you eve
    impact_cow_col,
    prod_col
]

def to_na(x):
    x = str(x).strip()
    return np.nan if x in ["", "N/A", "n/a", "NA", "na", "None", "none", "nan"] els

def is_unknown(x):
    x = str(x).strip().lower()
    return x in ["", "n/a", "na", "none", "nan", "i don't know", "i dont know", "i

def yn_from_text(x):
    x = str(x).strip().lower()
    if x.startswith("yes"):
        return "Yes"
    if x.startswith("no"):
        return "No"
    if x.startswith("not applicable"):
        return "Unknown"
    if x in ["n/a", "na", "none", "nan", ""]:
        return "Unknown"
    if "not sure" in x or "don't know" in x:
        return "Unknown"
    return "Unknown"

# 1) % time affected -> ordinal with missing = 0
df[pct_col] = df[pct_col].apply(to_na)

pct_map = {"1-25%": 1, "26-50%": 2, "51-75%": 3, "76-100%": 4}
df["pct_time_affected_ord"] = df[pct_col].map(pct_map).fillna(0).astype(int)

# 2) Merge coworker-impact into employer-negative-consequences (only if employer va
if impact_cow_col in df.columns and neg_conseq_col in df.columns:
    imp = df[impact_cow_col].apply(yn_from_text)
    base = df[neg_conseq_col].apply(yn_from_text)
    df[neg_conseq_col] = np.where(base == "Unknown", imp, base)

# 3) Merge medical-coverage (0/1) into employer-benefits (only if benefits is unkno
if medcov_col in df.columns and benefits_col in df.columns:
    mc_raw = df[medcov_col].astype(str).str.strip()
    mc = np.where(mc_raw == "1", "Yes", np.where(mc_raw == "0", "No", "Unknown"))
    b = df[benefits_col].apply(yn_from_text)
    df[benefits_col] = np.where(b == "Unknown", mc, b)

# 4) Drop redundant columns (+ original pct text column)
df = df.drop(columns=[pct_col] + [c for c in extra_drop if c in df.columns])

df[[benefits_col, neg_conseq_col, "pct_time_affected_ord"]].head(30)
```

Out[4]:

	Does your employer provide mental health benefits as part of healthcare coverage?	Do you think that discussing a mental health disorder with your employer would have negative consequences?	pct_time_affected_ord
0	No	No	0
1	No	No	0
2	No	Unknown	0
3	Unknown	Unknown	1
4	Yes	Yes	0
5	Yes	Yes	0
6	Unknown	No	0
7	Yes	No	0
8	Unknown	Yes	0
9	Unknown	No	1
10	Yes	No	0
11	Yes	Yes	0
12	Yes	No	0
13	Yes	No	0
14	Yes	No	0
15	Unknown	Unknown	0
16	No	Unknown	0
17	No	Unknown	0
18	Unknown	Yes	4
19	Unknown	Yes	0
20	No	No	0
21	Yes	No	0
22	Yes	Unknown	0
23	Yes	Unknown	0
24	Unknown	No	0
25	Yes	No	0
26	No	Yes	0
27	Unknown	No	0
28	Unknown	Unknown	0

	Does your employer provide mental health benefits as part of healthcare coverage?	Do you think that discussing a mental health disorder with your employer would have negative consequences?	pct_time_affected_ord
29	No	No	0

In [5]:
```
# Create one column that is either "No" (if they report no condition) or the condit

col_past = "Have you had a mental health disorder in the past?"
col_now  = "Do you currently have a mental health disorder?"
col_yes  = "If yes, what condition(s) have you been diagnosed with?"
col_may  = "If maybe, what condition(s) do you believe you have?"
col_pro  = "Have you been diagnosed with a mental health condition by a medical pro
col_proc = "If so, what condition(s) were you diagnosed with?"

def clean_na(x):
    x = str(x).strip()
    return np.nan if x.lower() in ["", "n/a", "na", "none", "nan"] else x

for c in [col_past, col_now, col_yes, col_may, col_pro, col_proc]:
    df[c] = df[c].apply(clean_na)

past = df[col_past].astype(str).str.strip().str.lower()
now  = df[col_now].astype(str).str.strip().str.lower()

has_condition = (now.isin(["yes", "maybe"])) | (past.isin(["yes", "maybe"]))

conds = df[col_proc].combine_first(df[col_yes]).combine_first(df[col_may])
conds = conds.fillna("").astype(str).str.strip()

df["mh_condition"] = np.where(has_condition, conds, "No")
df.loc[df["mh_condition"].str.strip().eq(""), "mh_condition"] = "No"

df = df.drop(columns=[col_past, col_now, col_yes, col_may, col_pro, col_proc])

df["mh_condition"].head(25)
```

Out[5]:
```
0     Anxiety Disorder (Generalized, Social, Phobia,...
1     Anxiety Disorder (Generalized, Social, Phobia,...
2                                                    No
3     Anxiety Disorder (Generalized, Social, Phobia,...
4     Anxiety Disorder (Generalized, Social, Phobia,...
5     Anxiety Disorder (Generalized, Social, Phobia,...
6                                                    No
7     Anxiety Disorder (Generalized, Social, Phobia,...
8     Mood Disorder (Depression, Bipolar Disorder, etc)
9     Anxiety Disorder (Generalized, Social, Phobia,...
10    Mood Disorder (Depression, Bipolar Disorder, etc)
11    Anxiety Disorder (Generalized, Social, Phobia,...
12    Mood Disorder (Depression, Bipolar Disorder, etc)
13            Substance Use Disorder|Addictive Disorder
14    Anxiety Disorder (Generalized, Social, Phobia,...
15    Mood Disorder (Depression, Bipolar Disorder, etc)
16    Anxiety Disorder (Generalized, Social, Phobia,...
17                                                   No
18    Personality Disorder (Borderline, Antisocial, ...
19    Anxiety Disorder (Generalized, Social, Phobia,...
20                                                   No
21                                                   No
22                                                   No
23    Mood Disorder (Depression, Bipolar Disorder, e...
24                                                   No
Name: mh_condition, dtype: object
```

In [6]:
```
pd.set_option('future.no_silent_downcasting', True)

missing_mask = df.replace("N/A", np.nan)

missing_pct = missing_mask.isna().mean()
cols_to_drop = missing_pct[missing_pct >= 0.50].index
df = df.drop(columns=cols_to_drop)

print(f"Dropped {len(cols_to_drop)} columns (>=50% missing).")
print("New shape:", df.shape)
cols_to_drop.tolist()
```

```
Dropped 2 columns (>=50% missing).
New shape: (1433, 49)
```

Out[6]:
```
['Is your primary role within your company related to tech/IT?',
 'Have your observations of how another individual who discussed a mental health d
isorder made you less likely to reveal a mental health issue yourself in your curr
ent workplace?']
```

In [7]:
```
# Fix the company-size column (including Excel date mistakes), one-hot encode each

col = "How many employees does your company or organization have?"

df[col] = df[col].astype(str).str.strip()
df[col] = df[col].replace({"5-Jan": "1-5", "25-Jun": "6-25", "More than 1000": ">10
df[col] = df[col].replace(["N/A", "nan", "None", ""], np.nan)

df[col] = df[col].replace({"1-5": "1-5", "6-25": "6-25", "26-100": "26-100", "100-5
df[col] = df[col].fillna("Unknown")
```

```
size_dummies = pd.get_dummies(df[col], prefix="company_size", dtype=int)

needed = ["1-5", "6-25", "26-100", "100-500", "500-1000", ">1000", "Unknown"]
needed_cols = [f"company_size_{k}" for k in needed]
for n in needed_cols:
    if n not in size_dummies.columns:
        size_dummies[n] = 0
size_dummies = size_dummies[needed_cols]

rows_ok = (size_dummies.sum(axis=1) >= 1).all()
print("Each row has at least one '1' across company_size one-hot columns:", rows_ok

df = pd.concat([df.drop(columns=[col]), size_dummies], axis=1)

size_dummies.head(20)
```

```
Each row has at least one '1' across company_size one-hot columns: True
```

Out[7]:

	company_size_1-5	company_size_6-25	company_size_26-100	company_size_100-500	company_size
0	0	0	1	0	
1	0	1	0	0	
2	0	1	0	0	
3	0	0	0	0	
4	0	1	0	0	
5	0	0	0	0	
6	0	0	1	0	
7	0	0	0	0	
8	0	0	1	0	
9	0	0	0	0	
10	0	0	1	0	
11	0	0	0	1	
12	0	0	0	1	
13	0	0	0	1	
14	0	0	0	1	
15	0	0	0	1	
16	0	0	1	0	
17	0	0	0	0	
18	0	0	0	0	
19	0	0	0	1	

In [8]:
```
# Clean the tech-employer binary column (0/1) and fill missing values with the medi

col = "Is your employer primarily a tech company/organization?"

df[col] = df[col].replace(["N/A", "nan", "None", "", " "], np.nan)
df[col] = pd.to_numeric(df[col], errors="coerce")

median_val = df[col].median()
df[col] = df[col].fillna(median_val)

print("Median used for imputation:", median_val)
df[col].value_counts(dropna=False)
```

```
Median used for imputation: 1.0
```

Out[8]:
```
Is your employer primarily a tech company/organization?
1.0    1170
0.0     263
Name: count, dtype: int64
```

In [9]:
```
# One-hot encode mental-health benefits as Yes/No/NotEligible, merge "I don't know"

col = "Does your employer provide mental health benefits as part of healthcare cove

df[col] = df[col].astype(str).str.strip()
df[col] = df[col].replace(["", " ", "N/A", "nan", "None"], np.nan)

df[col] = df[col].replace({
    "Yes": "Yes",
    "No": "No",
    "Not eligible for coverage / N/A": "NotEligible",
    "I don't know": "Unknown"
})

df[col] = df[col].fillna("Unknown")

benefits_dummies = pd.get_dummies(df[col], prefix="benefits", dtype=int)
df = pd.concat([df.drop(columns=[col]), benefits_dummies], axis=1)

rows_ok = (benefits_dummies.sum(axis=1) >= 1).all()
print("Each row has at least one '1' across the one-hot columns:", rows_ok)
benefits_dummies.head()
```

```
Each row has at least one '1' across the one-hot columns: True
```

Out[9]:

	benefits_No	benefits_Unknown	benefits_Yes
0	1	0	0
1	1	0	0
2	1	0	0
3	0	1	0
4	0	0	1

In [10]:
```
# Clean a list of yes/no/unclear columns into Yes/No/Unknown and one-hot encode the

import json

yn_mapping = {
    "yes": "Yes", "y": "Yes", "yeah": "Yes", "yep": "Yes", "true": "Yes", "1": "Yes
    "no": "No", "n": "No", "nope": "No", "false": "No", "0": "No",
    "i don't know": "Unknown", "dont know": "Unknown", "don't know": "Unknown", "id
    "maybe": "Unknown", "not sure": "Unknown", "unsure": "Unknown", "unknown": "Unk
    "n/a": np.nan, "na": np.nan, "none": np.nan, "nan": np.nan, "": np.nan
}

def one_hot_yes_no_unknown(df, cols, prefix="yn"):
    dummies_list = []
    for c in cols:
```

```
        s = df[c].astype(str).str.strip().str.lower()
        s = s.replace(yn_mapping)
        s = s.fillna("Unknown")
        d = pd.get_dummies(s, prefix=f"{prefix}__{c}", dtype=int)
        for needed in ["Yes", "No", "Unknown"]:
            col_name = f"{prefix}__{c}_{needed}"
            if col_name not in d.columns:
                d[col_name] = 0
        d = d[[f"{prefix}__{c}_Yes", f"{prefix}__{c}_No", f"{prefix}__{c}_Unknown"]]
        ok = (d.sum(axis=1) == 1).all()
        print(f"{c} -> rows with exactly one category:", ok)
        dummies_list.append(d)
    df_out = pd.concat([df.drop(columns=cols), *dummies_list], axis=1)
    return df_out


yn_cols = [
    "Has your employer ever formally discussed mental health (for example, as part
    "Does your employer offer resources to learn more about mental health concerns
    "Is your anonymity protected if you choose to take advantage of mental health o
    "Do you think that discussing a mental health disorder with your employer would
    "Do you think that discussing a physical health issue with your employer would
    "Would you feel comfortable discussing a mental health disorder with your cowor
    "Would you feel comfortable discussing a mental health disorder with your direc
    "Do you feel that your employer takes mental health as seriously as physical he
    "Have you heard of or observed negative consequences for co-workers who have be
    "Would you be willing to bring up a physical health issue with a potential empl
    "Would you bring up a mental health issue with a potential employer in an inter
    "Do you have a family history of mental illness?",
    "Have you had a mental health disorder in the past?",
    "Do you currently have a mental health disorder?",
    "Have you been diagnosed with a mental health condition by a medical profession
]

yn_cols = [c for c in yn_cols if c in df.columns]

df = one_hot_yes_no_unknown(df, yn_cols, prefix="yn")

mapping_json = json.dumps({"yes_no_unknown_mapping": yn_mapping, "columns_encoded":
print(mapping_json)
```

```
Has your employer ever formally discussed mental health (for example, as part of a w
ellness campaign or other official communication)? -> rows with exactly one categor
y: True
Does your employer offer resources to learn more about mental health concerns and op
tions for seeking help? -> rows with exactly one category: True
Is your anonymity protected if you choose to take advantage of mental health or subs
tance abuse treatment resources provided by your employer? -> rows with exactly one
category: True
Do you think that discussing a mental health disorder with your employer would have
negative consequences? -> rows with exactly one category: True
Do you think that discussing a physical health issue with your employer would have n
egative consequences? -> rows with exactly one category: True
Would you feel comfortable discussing a mental health disorder with your coworkers?
-> rows with exactly one category: True
Would you feel comfortable discussing a mental health disorder with your direct supe
rvisor(s)? -> rows with exactly one category: True
Do you feel that your employer takes mental health as seriously as physical health?
-> rows with exactly one category: True
Have you heard of or observed negative consequences for co-workers who have been ope
n about mental health issues in your workplace? -> rows with exactly one category: T
rue
Would you be willing to bring up a physical health issue with a potential employer i
n an interview? -> rows with exactly one category: True
Would you bring up a mental health issue with a potential employer in an interview?
-> rows with exactly one category: True
Do you have a family history of mental illness? -> rows with exactly one category: T
rue
{
  "yes_no_unknown_mapping": {
    "yes": "Yes",
    "y": "Yes",
    "yeah": "Yes",
    "yep": "Yes",
    "true": "Yes",
    "1": "Yes",
    "no": "No",
    "n": "No",
    "nope": "No",
    "false": "No",
    "0": "No",
    "i don't know": "Unknown",
    "dont know": "Unknown",
    "don't know": "Unknown",
    "idk": "Unknown",
    "maybe": "Unknown",
    "not sure": "Unknown",
    "unsure": "Unknown",
    "unknown": "Unknown",
    "n/a": NaN,
    "na": NaN,
    "none": NaN,
    "nan": NaN,
    "": NaN
  },
  "columns_encoded": [
    "Has your employer ever formally discussed mental health (for example, as part o
```

```
f a wellness campaign or other official communication)?",
    "Does your employer offer resources to learn more about mental health concerns a
nd options for seeking help?",
    "Is your anonymity protected if you choose to take advantage of mental health or
substance abuse treatment resources provided by your employer?",
    "Do you think that discussing a mental health disorder with your employer would
have negative consequences?",
    "Do you think that discussing a physical health issue with your employer would h
ave negative consequences?",
    "Would you feel comfortable discussing a mental health disorder with your cowork
ers?",
    "Would you feel comfortable discussing a mental health disorder with your direct
supervisor(s)?",
    "Do you feel that your employer takes mental health as seriously as physical hea
lth?",
    "Have you heard of or observed negative consequences for co-workers who have bee
n open about mental health issues in your workplace?",
    "Would you be willing to bring up a physical health issue with a potential emplo
yer in an interview?",
    "Would you bring up a mental health issue with a potential employer in an interv
iew?",
    "Do you have a family history of mental illness?"
  ]
}
```

In [11]:

```
# Standardize the gender column into Male/Female/Other, one-hot encode it, and veri

import re

col = "What is your gender?"

s = df[col].astype(str).str.strip().str.lower()
s = s.replace(["", " ", "n/a", "na", "none", "nan"], np.nan)

def gender_bucket(x):
    if pd.isna(x):
        return "Other"
    x = re.sub(r"[^a-z\s]", "", x).strip()
    if x in ["m", "male", "man", "cis male", "cisman", "cis man", "male cis", "male
        return "Male"
    if x in ["f", "female", "woman", "cis female", "ciswoman", "cis woman", "female
        return "Female"
    if "female" in x and "identify" in x:
        return "Female"
    if "male" in x and "identify" in x:
        return "Male"
    if "female" == x:
        return "Female"
    if "male" == x:
        return "Male"
    return "Other"

df[col] = s.apply(gender_bucket)

gender_dummies = pd.get_dummies(df[col], prefix="gender", dtype=int)
for needed in ["gender_Male", "gender_Female", "gender_Other"]:
```

```
    if needed not in gender_dummies.columns:
        gender_dummies[needed] = 0
gender_dummies = gender_dummies[["gender_Male", "gender_Female", "gender_Other"]]

rows_ok = (gender_dummies.sum(axis=1) >= 1).all()
print("Each row has at least one '1' across gender one-hot columns:", rows_ok)

df = pd.concat([df.drop(columns=[col]), gender_dummies], axis=1)
gender_dummies.head()
```

```
Each row has at least one '1' across gender one-hot columns: True
```

Out[11]:

	gender_Male	gender_Female	gender_Other
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	0	1	0

In [12]:

```
pd.concat([df.filter(like="gender_")], axis=1).head(30)
```

Out[12]:

	gender_Male	gender_Female	gender_Other
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	0	1	0
5	1	0	0
6	1	0	0
7	0	1	0
8	0	1	0
9	1	0	0
10	1	0	0
11	1	0	0
12	0	1	0
13	1	0	0
14	1	0	0
15	1	0	0
16	0	1	0
17	1	0	0
18	0	1	0
19	1	0	0
20	1	0	0
21	1	0	0
22	1	0	0
23	1	0	0
24	1	0	0
25	1	0	0
26	1	0	0
27	1	0	0
28	0	1	0
29	0	0	1

In [13]:
```
# Clean the two "previous employers" columns into NoneDid/SomeDid/AllDid/Unknown (h

cols = [
    "Did your previous employers ever formally discuss mental health (as part of a 
    "Did your previous employers provide resources to learn more about mental healt
]

standard_map = {
    "none did": "NoneDid",
    "some did": "SomeDid",
    "yes, they all did": "AllDid",
    "yes they all did": "AllDid",
    "i don't know": "Unknown",
    "i dont know": "Unknown",
    "dont know": "Unknown",
    "unknown": "Unknown"
}

new_blocks = []

for col in cols:
    s = df[col].astype(str).str.strip().str.lower()
    s = s.replace(["", " ", "n/a", "na", "none", "nan"], np.nan)
    s = s.replace(standard_map).fillna("Unknown")

    d = pd.get_dummies(s, prefix=f"prev__{col}", dtype=int)

    needed = [f"prev__{col}_NoneDid", f"prev__{col}_SomeDid", f"prev__{col}_AllDid"
    for n in needed:
        if n not in d.columns:
            d[n] = 0
    d = d[needed]

    print(col, "-> each row has exactly one category:", (d.sum(axis=1) == 1).all())
    new_blocks.append(d)

df = pd.concat([df.drop(columns=cols), *new_blocks], axis=1)

df.filter(like="prev__").head(20)
```

```
Did your previous employers ever formally discuss mental health (as part of a wellne
ss campaign or other official communication)? -> each row has exactly one category:
True
Did your previous employers provide resources to learn more about mental health issu
es and how to seek help? -> each row has exactly one category: True
```

Out[13]:

	prev__Did your previous employers ever formally discuss mental health (as part of a wellness campaign or other official communication)? _NoneDid	prev__Did your previous employers ever formally discuss mental health (as part of a wellness campaign or other official communication)? _SomeDid	prev__Did your previous employers ever formally discuss mental health (as part of a wellness campaign or other official communication)? _AllDid	prev__Did your previous employers ever formally discuss mental health (as part of a wellness campaign or other official communication)? _Unknown	prev__Did your previous employers provide resources to learn more about mental health issues and how to seek help? _NoneDid	p er r is se _S
0	0	0	0	1	1	
1	1	0	0	0	0	
2	1	0	0	0	0	
3	1	0	0	0	1	
4	0	1	0	0	1	
5	1	0	0	0	1	
6	1	0	0	0	0	
7	0	1	0	0	0	
8	0	1	0	0	1	
9	1	0	0	0	1	
10	1	0	0	0	1	
11	1	0	0	0	1	
12	0	1	0	0	0	
13	1	0	0	0	1	
14	1	0	0	0	1	
15	0	1	0	0	0	
16	1	0	0	0	1	
17	1	0	0	0	1	
18	0	0	0	1	0	
19	1	0	0	0	0	

In [14]:
```
# Clean the 3 "previous workplace consequences" columns into None/Some/All/Unknown,
```

```
cols = [
    "Do you think that discussing a mental health disorder with previous employers
    "Do you think that discussing a physical health issue with previous employers w
    "Did you hear of or observe negative consequences for co-workers with mental he
]

standard_map = {
    "none of them": "None",
    "some of them": "Some",
    "yes, all of them": "All",
    "yes all of them": "All",
    "i don't know": "Unknown",
    "i dont know": "Unknown",
    "dont know": "Unknown",
    "unknown": "Unknown"
}

new_blocks = []

for col in cols:
    s = df[col].astype(str).str.strip().str.lower()
    s = s.replace(["", " ", "n/a", "na", "none", "nan"], np.nan)
    s = s.replace(standard_map).fillna("Unknown")

    d = pd.get_dummies(s, prefix=f"prevneg__{col}", dtype=int)

    needed = [f"prevneg__{col}_None", f"prevneg__{col}_Some", f"prevneg__{col}_All"
    for n in needed:
        if n not in d.columns:
            d[n] = 0
    d = d[needed]

    print(col, "-> each row has exactly one category:", (d.sum(axis=1) == 1).all())
    new_blocks.append(d)

df = pd.concat([df.drop(columns=cols), *new_blocks], axis=1)

df.filter(like="prevneg__").head(20)
```

```
Do you think that discussing a mental health disorder with previous employers would
have negative consequences? -> each row has exactly one category: True
Do you think that discussing a physical health issue with previous employers would h
ave negative consequences? -> each row has exactly one category: True
Did you hear of or observe negative consequences for co-workers with mental health i
ssues in your previous workplaces? -> each row has exactly one category: True
```

Out[14]:

	prevneg__Do you think that discussing a mental health disorder with previous employers would have negative consequences? _None	prevneg__Do you think that discussing a mental health disorder with previous employers would have negative consequences? _Some	prevneg__Do you think that discussing a mental health disorder with previous employers would have negative consequences? _All	prevneg__Do you think that discussing a mental health disorder with previous employers would have negative consequences? _Unknown	prevneg__Do you think that discussing a physical health issue with previous employers would have negative consequences? _None	prevn you thi discu p healt with p em woul n conseq
0	0	1	0	0	1	
1	1	0	0	0	1	
2	0	0	0	1	0	
3	0	1	0	0	0	
4	0	1	0	0	0	
5	0	0	1	0	0	
6	1	0	0	0	1	
7	0	1	0	0	0	
8	0	0	1	0	0	
9	0	1	0	0	0	
10	0	1	0	0	0	
11	0	0	1	0	0	
12	1	0	0	0	1	
13	0	0	1	0	0	
14	0	0	0	1	1	
15	0	1	0	0	0	
16	0	0	1	0	1	
17	0	1	0	0	0	
18	0	0	0	1	0	
19	0	0	1	0	1	

In [15]:
```
# Clean the 6 "previous employers" columns into consistent buckets, one-hot encode

cols = [
    "Have your previous employers provided mental health benefits?",
    "Were you aware of the options for mental health care provided by your previous
```

```
    "Was your anonymity protected if you chose to take advantage of mental health o
    "Would you have been willing to discuss a mental health issue with your previou
    "Would you have been willing to discuss a mental health issue with your direct
    "Did you feel that your previous employers took mental health as seriously as p
]

maps = {
    cols[0]: {
        "no, none did": "NoneDid",
        "some did": "SomeDid",
        "yes, they all did": "AllDid",
        "i don't know": "Unknown"
    },
    cols[1]: {
        "n/a (not currently aware)": "NotAware",
        "i was aware of some": "AwareSome",
        "yes, i was aware of all of them": "AwareAll",
        "no, i only became aware later": "AwareLater",
        "i don't know": "Unknown"
    },
    cols[2]: {
        "yes, always": "YesAlways",
        "sometimes": "Sometimes",
        "no": "No",
        "i don't know": "Unknown"
    },
    cols[3]: {
        "no, at none of my previous employers": "None",
        "some of my previous employers": "Some",
        "yes, at all of my previous employers": "All",
        "i don't know": "Unknown"
    },
    cols[4]: {
        "no, at none of my previous employers": "None",
        "some of my previous employers": "Some",
        "yes, at all of my previous employers": "All",
        "i don't know": "Unknown"
    },
    cols[5]: {
        "none did": "NoneDid",
        "some did": "SomeDid",
        "yes, they all did": "AllDid",
        "i don't know": "Unknown"
    }
}

new_blocks = []

for col in cols:
    s = df[col].astype(str).str.strip().str.lower()
    s = s.replace(["", " ", "n/a", "na", "none", "nan"], np.nan)
    s = s.replace(maps[col]).fillna("Unknown")

    d = pd.get_dummies(s, prefix=f"prev6__{col}", dtype=int)

    needed_vals = sorted(set(maps[col].values()) | {"Unknown"})
```

```
    needed_cols = [f"prev6__{col}_{v}" for v in needed_vals]
    for n in needed_cols:
        if n not in d.columns:
            d[n] = 0
    d = d[needed_cols]

    print(col, "-> each row has exactly one category:", (d.sum(axis=1) == 1).all())
    new_blocks.append(d)

df = pd.concat([df.drop(columns=cols), *new_blocks], axis=1)

df.filter(like="prev6__").head(20)
```

```
Have your previous employers provided mental health benefits? -> each row has exactl
y one category: True
Were you aware of the options for mental health care provided by your previous emplo
yers? -> each row has exactly one category: True
Was your anonymity protected if you chose to take advantage of mental health or subs
tance abuse treatment resources with previous employers? -> each row has exactly one
category: True
Would you have been willing to discuss a mental health issue with your previous co-w
orkers? -> each row has exactly one category: True
Would you have been willing to discuss a mental health issue with your direct superv
isor(s)? -> each row has exactly one category: True
Did you feel that your previous employers took mental health as seriously as physica
l health? -> each row has exactly one category: True
```

Out[15]:

	prev6__Have your previous employers provided mental health benefits? _AllDid	prev6__Have your previous employers provided mental health benefits? _NoneDid	prev6__Have your previous employers provided mental health benefits? _SomeDid	prev6__Have your previous employers provided mental health benefits? _Unknown	prev6__Were you aware of the options for mental health care provided by your previous employers? _AwareAll	prev6__Were you aware of the options for mental health care provided by your previous employers? _AwareLater	prev y op he pro em _Aw
0	0	1	0	0	0	0	
1	1	0	0	0	0	0	
2	0	1	0	0	0	0	
3	0	0	1	0	0	0	
4	0	0	0	1	0	0	
5	0	1	0	0	1	0	
6	0	0	1	0	0	0	
7	0	0	1	0	0	0	
8	0	0	0	1	0	0	
9	0	0	1	0	0	0	
10	0	1	0	0	0	0	
11	0	1	0	0	0	0	
12	0	0	1	0	1	0	
13	0	0	0	1	0	0	
14	0	0	0	1	0	1	
15	0	0	1	0	0	0	
16	0	1	0	0	0	1	
17	0	1	0	0	1	0	
18	0	0	0	1	0	0	
19	0	0	0	1	0	0	

20 rows × 25 columns

In [16]:
```
# One-hot encode the two "interferes with work" columns (Never/Rarely/Sometimes/Oft

cols = [
    "If you have a mental health issue, do you feel that it interferes with your wo
    "If you have a mental health issue, do you feel that it interferes with your wo
]

standard_map = {
    "never": "Never",
    "rarely": "Rarely",
    "sometimes": "Sometimes",
    "often": "Often",
    "not applicable to me": "Unknown"
}

new_blocks = []

for col in cols:
    s = df[col].astype(str).str.strip().str.lower()
    s = s.replace(["", " ", "n/a", "na", "none", "nan"], np.nan)
    s = s.replace(standard_map).fillna("Unknown")

    d = pd.get_dummies(s, prefix=f"interf__{col}", dtype=int)

    needed = [f"interf__{col}_Never", f"interf__{col}_Rarely", f"interf__{col}_Some
    for n in needed:
        if n not in d.columns:
            d[n] = 0
    d = d[needed]

    print(col, "-> each row has exactly one category:", (d.sum(axis=1) == 1).all())
    new_blocks.append(d)

df = pd.concat([df.drop(columns=cols), *new_blocks], axis=1)

df.filter(like="interf__").head(20)
```

```
If you have a mental health issue, do you feel that it interferes with your work whe
n being treated effectively? -> each row has exactly one category: True
If you have a mental health issue, do you feel that it interferes with your work whe
n NOT being treated effectively? -> each row has exactly one category: True
```

Out[16]:

	interf__If you have a mental health issue, do you feel that it interferes with your work when being treated effectively? _Never	interf__If you have a mental health issue, do you feel that it interferes with your work when being treated effectively? _Rarely	interf__If you have a mental health issue, do you feel that it interferes with your work when being treated effectively? _Sometimes	interf__If you have a mental health issue, do you feel that it interferes with your work when being treated effectively? _Often	interf__If you have a mental health issue, do you feel that it interferes with your work when being treated effectively? _Unknown	interf__If you have a mental health issue, do you feel that it interferes with your work when NOT being treated effectively? _Never	interf__I you have a menta healt issue, do you fee that i interferes with you work wher NOT being treated effectively _Rarely
0	0	0	0	0	1	0	0
1	0	1	0	0	0	0	0
2	0	0	0	0	1	0	0
3	0	0	1	0	0	0	0
4	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0
6	0	0	0	0	1	0	0
7	0	0	1	0	0	0	0
8	0	1	0	0	0	0	0
9	0	1	0	0	0	0	0
10	0	0	1	0	0	0	0
11	1	0	0	0	0	0	1
12	0	1	0	0	0	0	0
13	0	0	0	0	1	0	0
14	0	0	1	0	0	0	0
15	0	1	0	0	0	0	0
16	0	0	1	0	0	0	0
17	0	0	0	0	1	0	0
18	0	0	0	1	0	0	0
19	0	1	0	0	0	0	0

In [17]:
```
# One-hot encode the 5 columns shown (medical leave ease + 4 attitude/experience co
```

```
col_leave = "If a mental health issue prompted you to request a medical leave from
col_career = "Do you feel that being identified as a person with a mental health is
col_view = "Do you think that team members/co-workers would view you more negativel
col_share = "How willing would you be to share with friends and family that you hav
col_unsup = "Have you observed or experienced an unsupportive or badly handled resp

cols = [col_leave, col_career, col_view, col_share, col_unsup]

leave_map = {
    "very easy": "VeryEasy",
    "somewhat easy": "SomewhatEasy",
    "neither easy nor difficult": "Neutral",
    "somewhat difficult": "SomewhatDifficult",
    "very difficult": "VeryDifficult",
    "i don't know": "Unknown",
    "i dont know": "Unknown",
    "not applicable to me": "Unknown",
    "n/a": np.nan
}

career_map = {
    "yes, i think it would": "Yes",
    "no, i don't think it would": "No",
    "no, it has not": "No",
    "yes, it has": "Yes",
    "maybe": "Maybe",
    "i don't know": "Unknown",
    "i dont know": "Unknown"
}

view_map = {
    "yes, i think they would": "Yes",
    "yes, they do": "Yes",
    "no, i don't think they would": "No",
    "no, they do not": "No",
    "maybe": "Maybe",
    "i don't know": "Unknown",
    "i dont know": "Unknown"
}

share_map = {
    "very open": "VeryOpen",
    "somewhat open": "SomewhatOpen",
    "neutral": "Neutral",
    "somewhat not open": "SomewhatNotOpen",
    "not open at all": "NotOpen",
    "not applicable to me (i do not have a mental illness)": "NotApplicable",
    "i don't know": "Unknown",
    "i dont know": "Unknown"
}

unsup_map = {
    "no": "No",
    "yes, i experienced": "Experienced",
    "yes, i observed": "Observed",
    "maybe/not sure": "Maybe",
```

```
    "maybe": "Maybe",
    "i don't know": "Unknown",
    "i dont know": "Unknown"
}

maps = {
    col_leave: leave_map,
    col_career: career_map,
    col_view: view_map,
    col_share: share_map,
    col_unsup: unsup_map
}

expected = {
    col_leave: ["VeryEasy","SomewhatEasy","Neutral","SomewhatDifficult","VeryDiffic
    col_career: ["Yes","No","Maybe","Unknown"],
    col_view: ["Yes","No","Maybe","Unknown"],
    col_share: ["VeryOpen","SomewhatOpen","Neutral","SomewhatNotOpen","NotOpen","No
    col_unsup: ["No","Experienced","Observed","Maybe","Unknown"]
}

new_blocks = []

for col in cols:
    s = df[col].astype(str).str.strip().str.lower()
    s = s.replace(["", " ", "n/a", "na", "none", "nan"], np.nan)
    s = s.replace(maps[col]).fillna("Unknown")

    d = pd.get_dummies(s, prefix=f"mhatt__{col}", dtype=int)

    needed_cols = [f"mhatt__{col}_{v}" for v in expected[col]]
    for n in needed_cols:
        if n not in d.columns:
            d[n] = 0
    d = d[needed_cols]

    print(col, "-> each row has exactly one category:", (d.sum(axis=1) == 1).all())
    new_blocks.append(d)

df = pd.concat([df.drop(columns=cols), *new_blocks], axis=1)

df.filter(like="mhatt__").head(20)
```

```
If a mental health issue prompted you to request a medical leave from work, asking f
or that leave would be: -> each row has exactly one category: True
Do you feel that being identified as a person with a mental health issue would hurt
your career? -> each row has exactly one category: True
Do you think that team members/co-workers would view you more negatively if they kne
w you suffered from a mental health issue? -> each row has exactly one category: Tru
e
How willing would you be to share with friends and family that you have a mental ill
ness? -> each row has exactly one category: True
Have you observed or experienced an unsupportive or badly handled response to a ment
al health issue in your current or previous workplace? -> each row has exactly one c
ategory: True
```

Out[17]:

	mhatt__If a mental health issue prompted you to request a medical leave from work, asking for that leave would be:_VeryEasy	mhatt__If a mental health issue prompted you to request a medical leave from work, asking for that leave would be:_SomewhatEasy	mhatt__If a mental health issue prompted you to request a medical leave from work, asking for that leave would be:_Neutral	mhatt__If a mental health issue prompted you to request a medical leave from work, asking for that leave would be:_SomewhatDifficult	mhatt__If a mental health issue prompted you to request a medical leave from work, asking for that leave would be:_VeryDifficult
0	1	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	0	0
4	0	0	1	0	0
5	0	1	0	0	0
6	0	1	0	0	0
7	1	0	0	0	0
8	0	0	0	0	1
9	0	0	0	0	0
10	1	0	0	0	0
11	0	0	0	1	0
12	1	0	0	0	0
13	0	1	0	0	0
14	1	0	0	0	0
15	0	0	1	0	0
16	0	0	0	0	1
17	0	0	0	0	0
18	0	0	0	0	0
19	0	0	0	0	0

20 rows × 26 columns

In [18]:
```
# One-hot encode the remote-work column into Never/Sometimes/Always/Unknown and san

col = "Do you work remotely?"

s = df[col].astype(str).str.strip().str.lower()
s = s.replace(["", " ", "n/a", "na", "none", "nan"], np.nan)

s = s.replace({
    "never": "Never",
    "sometimes": "Sometimes",
    "always": "Always"
}).fillna("Unknown")

d = pd.get_dummies(s, prefix="remote", dtype=int)

needed = ["remote_Never", "remote_Sometimes", "remote_Always", "remote_Unknown"]
for n in needed:
    if n not in d.columns:
        d[n] = 0
d = d[needed]

print("Do you work remotely? -> each row has exactly one category:", (d.sum(axis=1)

df = pd.concat([df.drop(columns=[col]), d], axis=1)

d.head(20)
```

```
Do you work remotely? -> each row has exactly one category: True
```

Out[18]:

	remote_Never	remote_Sometimes	remote_Always	remote_Unknown
0	0	1	0	0
1	1	0	0	0
2	0	0	1	0
3	0	1	0	0
4	0	1	0	0
5	0	1	0	0
6	0	1	0	0
7	0	0	1	0
8	0	1	0	0
9	0	0	1	0
10	1	0	0	0
11	0	1	0	0
12	0	0	1	0
13	0	1	0	0
14	0	0	1	0
15	0	1	0	0
16	0	0	1	0
17	1	0	0	0
18	0	0	1	0
19	0	1	0	0

In [19]:

```
# Split the multi-role work-position column, create a binary feature for each uniqu
import pandas as pd
import json

col = "Which of the following best describes your work position?"

s = df[col].astype(str).str.strip().fillna("")
roles_series = s.str.get_dummies(sep="|").astype(int)

roles_series.columns = [c.strip() for c in roles_series.columns]
roles_series = roles_series.loc[:, roles_series.columns != ""]

role_to_id = {role: i for i, role in enumerate(roles_series.columns)}
role_table_json = json.dumps({"work_position_role_to_id": role_to_id}, indent=2)

roles_series = roles_series.add_prefix("role_")
df = pd.concat([df.drop(columns=[col]), roles_series], axis=1)
```

```
print("Number of unique roles found:", len(role_to_id))
print(role_table_json)
df.filter(like="role_").head(20)
```

```
Number of unique roles found: 12
{
  "work_position_role_to_id": {
    "Back-end Developer": 0,
    "Designer": 1,
    "Dev Evangelist/Advocate": 2,
    "DevOps/SysAdmin": 3,
    "Executive Leadership": 4,
    "Front-end Developer": 5,
    "HR": 6,
    "One-person shop": 7,
    "Other": 8,
    "Sales": 9,
    "Supervisor/Team Lead": 10,
    "Support": 11
  }
}
```

Out[19]:

	role_Back-end Developer	role_Designer	role_Dev Evangelist/Advocate	role_DevOps/SysAdmin	role_Executive Leadership
0	1	0	0	0	0
1	1	0	0	0	0
2	1	0	0	0	0
3	0	0	0	0	0
4	1	0	1	1	1
5	1	1	0	1	0
6	1	0	0	0	0
7	1	0	1	0	0
8	1	0	0	0	0
9	1	0	0	0	0
10	0	0	0	0	0
11	0	0	0	0	1
12	1	0	1	0	0
13	1	0	0	1	0
14	0	1	0	0	0
15	0	0	0	0	1
16	0	0	0	0	0
17	1	0	0	0	0
18	0	0	0	0	0
19	1	0	0	0	0

In [20]:

```
# Replace USA in the live/work country columns with the corresponding US region (ba

col_live_country = "What country do you live in?"
col_live_state   = "What US state or territory do you live in?"
col_work_country = "What country do you work in?"
col_work_state   = "What US state or territory do you work in?"

loc_cols = [col_live_country, col_live_state, col_work_country, col_work_state]

for c in loc_cols:
    df[c] = df[c].astype(str).str.strip()
    df[c] = df[c].replace(["", " ", "N/A", "n/a", "NA", "na", "None", "none", "nan"

country_fix = {
```

```
    "United States of America": "United States",
    "USA": "United States",
    "U.S.A.": "United States",
    "UK": "United Kingdom"
}
df[col_live_country] = df[col_live_country].replace(country_fix)
df[col_work_country] = df[col_work_country].replace(country_fix)

us_region_map = {
    "Northeast": ["Connecticut","Maine","Massachusetts","New Hampshire","New Jersey
    "Midwest": ["Illinois","Indiana","Iowa","Kansas","Michigan","Minnesota","Missou
    "South America": ["Alabama","Arkansas","Delaware","Florida","Georgia","Kentucky
    "West America": ["Alaska","Arizona","California","Colorado","Hawaii","Idaho","M
}
state_to_region = {st: region for region, states in us_region_map.items() for st in

live_region = df[col_live_state].map(state_to_region)
work_region = df[col_work_state].map(state_to_region)

df.loc[df[col_live_country] == "United States", col_live_country] = live_region.fil
df.loc[df[col_work_country] == "United States", col_work_country] = work_region.fil

df = df.drop(columns=[col_live_state, col_work_state])

df[[col_live_country, col_work_country]].head(30)
```

Out[20]:

	What country do you live in?	What country do you work in?
0	United Kingdom	United Kingdom
1	Midwest	Midwest
2	United Kingdom	United Kingdom
3	United Kingdom	United Kingdom
4	Midwest	Midwest
5	United Kingdom	United Kingdom
6	South America	South America
7	South America	South America
8	West America	West America
9	South America	South America
10	West America	West America
11	Northeast	Northeast
12	Northeast	Northeast
13	Canada	Canada
14	South America	South America
15	West America	West America
16	Canada	Canada
17	United Kingdom	United Kingdom
18	West America	West America
19	Northeast	Northeast
20	Northeast	Northeast
21	Midwest	Midwest
22	Germany	Germany
23	Canada	Canada
24	Northeast	Northeast
25	Germany	Germany
26	Netherlands	Netherlands
27	Germany	Germany
28	United Kingdom	United Kingdom
29	United Kingdom	United Kingdom

In [21]:
```
# Count how many times each unique value appears in each column (e.g., "United King

def print_value_counts(df, columns, top_n=None):
    for col in columns:
        print("\n" + "="*80)
        print(col)
        counts = df[col].value_counts(dropna=False)
        if top_n is not None:
            counts = counts.head(top_n)
        for k, v in counts.items():
            print(f"{k} = {v}")

cols_to_summarize = ["What country do you live in?", "What country do you work in?"
print_value_counts(df, cols_to_summarize, top_n=None)
```

```
================================================================================
What country do you live in?
West America = 262
Midwest = 260
South America = 187
United Kingdom = 180
Northeast = 131
Canada = 78
Germany = 58
Netherlands = 48
Australia = 35
Sweden = 19
France = 16
Ireland = 15
Switzerland = 10
Brazil = 10
Russia = 9
India = 9
New Zealand = 9
Bulgaria = 7
Finland = 7
Denmark = 7
Belgium = 5
Italy = 5
Poland = 4
Spain = 4
Austria = 4
South Africa = 4
Romania = 4
Chile = 3
Czech Republic = 3
Pakistan = 3
Norway = 3
Lithuania = 2
Japan = 2
Mexico = 2
Afghanistan = 2
Colombia = 2
Other = 2
Israel = 2
Estonia = 2
Bosnia and Herzegovina = 2
Venezuela = 1
Algeria = 1
Bangladesh = 1
Vietnam = 1
Costa Rica = 1
Argentina = 1
Slovakia = 1
Brunei = 1
Hungary = 1
Iran = 1
Greece = 1
Ecuador = 1
China = 1
Guatemala = 1
```

```
Taiwan = 1
Serbia = 1

================================================================================
What country do you work in?
West America = 270
Midwest = 262
South America = 187
United Kingdom = 183
Northeast = 132
Canada = 74
Germany = 58
Netherlands = 47
Australia = 34
Sweden = 20
Ireland = 15
France = 14
Switzerland = 10
Brazil = 10
Russia = 9
India = 9
New Zealand = 9
Bulgaria = 7
Finland = 7
Denmark = 7
Belgium = 5
Poland = 4
South Africa = 4
Austria = 4
Chile = 3
Romania = 3
Italy = 3
Norway = 3
Spain = 3
Czech Republic = 3
Pakistan = 2
Mexico = 2
Israel = 2
Other = 2
Estonia = 2
Colombia = 2
Afghanistan = 2
Bosnia and Herzegovina = 2
Lithuania = 1
Venezuela = 1
United Arab Emirates = 1
Slovakia = 1
Bangladesh = 1
Costa Rica = 1
Turkey = 1
Argentina = 1
Vietnam = 1
Greece = 1
Hungary = 1
Iran = 1
Brunei = 1
```

```
Japan = 1
China = 1
Ecuador = 1
Guatemala = 1
Serbia = 1
```

In [22]:
```python
# Collapse live/work country columns to the top 15 values (rest -> Other) and one-h

cols = ["What country do you live in?", "What country do you work in?"]
top_n = 15

for col in cols:
    top_vals = df[col].value_counts(dropna=False).head(top_n).index
    df[col] = df[col].where(df[col].isin(top_vals), "Other")

country_dummies = pd.get_dummies(df[cols], prefix=["live_country", "work_country"],
df = pd.concat([df.drop(columns=cols), country_dummies], axis=1)

print("Added country one-hot columns:", country_dummies.shape[1])
country_dummies.head(20)
```

```
Added country one-hot columns: 32
```

Out[22]:

	live_country_Australia	live_country_Brazil	live_country_Canada	live_country_France	live
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	0	0	0	0	
5	0	0	0	0	
6	0	0	0	0	
7	0	0	0	0	
8	0	0	0	0	
9	0	0	0	0	
10	0	0	0	0	
11	0	0	0	0	
12	0	0	0	0	
13	0	0	1	0	
14	0	0	0	0	
15	0	0	0	0	
16	0	0	1	0	
17	0	0	0	0	
18	0	0	0	0	
19	0	0	0	0	

20 rows × 32 columns

In [23]:
```
# One-hot encode the top 15 most common mental-health conditions (counted per-condi
import pandas as pd

col = "mh_condition"
top_n = 15

top_conditions = (
    df[col]
    .fillna("No")
    .astype(str)
    .str.split("|")
    .explode()
    .str.strip()
```

```
    .value_counts()
    .head(top_n)
    .index
    .tolist()
)

def conditions_to_row(c):
    parts = [p.strip() for p in str(c).split("|") if p.strip()]
    if len(parts) == 0 or parts == ["No"]:
        return {"No": 1}
    out = {"No": 0}
    has_other = False
    for p in parts:
        if p in top_conditions:
            out[p] = 1
        else:
            has_other = True
    if has_other:
        out["Other"] = 1
    return out

cond_df = pd.DataFrame([conditions_to_row(x) for x in df[col]]).fillna(0).astype(in

needed = ["No"] + top_conditions + ["Other"]
for n in needed:
    if n not in cond_df.columns:
        cond_df[n] = 0
cond_df = cond_df[needed]

cond_df = cond_df.add_prefix("mhcond_")
df = pd.concat([df.drop(columns=[col]), cond_df], axis=1)

print("Top 15 conditions used:", top_conditions)
print("Example rows check (No + conditions + Other):")
print(cond_df.sum(axis=1).head(20).tolist())

df.filter(like="mhcond_").head(20)
```

```
Top 15 conditions used: ['Mood Disorder (Depression, Bipolar Disorder, etc)', 'Anxie
ty Disorder (Generalized, Social, Phobia, etc)', 'No', 'Attention Deficit Hyperactiv
ity Disorder', 'Post-traumatic Stress Disorder', 'Personality Disorder (Borderline,
Antisocial, Paranoid, etc)', 'Stress Response Syndromes', 'Obsessive-Compulsive Diso
rder', 'Substance Use Disorder', 'Addictive Disorder', 'Eating Disorder (Anorexia, B
ulimia, etc)', 'Psychotic Disorder (Schizophrenia, Schizoaffective, etc)', 'Dissocia
tive Disorder', 'Autism', 'Depression']
Example rows check (No + conditions + Other):
[1, 2, 2, 2, 1, 3, 2, 2, 1, 2, 1, 1, 1, 2, 2, 1, 2, 2, 2, 2]
```

Out[23]:

	mhcond_No	mhcond_Mood Disorder (Depression, Bipolar Disorder, etc)	mhcond_Anxiety Disorder (Generalized, Social, Phobia, etc)	mhcond_No	mhcond_Attention Deficit Hyperactivity Disorder	mhcon tra D
0	0	0	1	0	0	
1	0	1	1	0	0	
2	1	0	0	1	0	
3	0	1	1	0	0	
4	0	0	1	0	0	
5	0	0	1	0	0	
6	1	0	0	1	0	
7	0	0	1	0	0	
8	0	1	0	0	0	
9	0	1	1	0	0	
10	0	1	0	0	0	
11	0	0	1	0	0	
12	0	1	0	0	0	
13	0	0	0	0	0	
14	0	0	1	0	0	
15	0	1	0	0	0	
16	0	1	1	0	0	
17	1	0	0	1	0	
18	0	0	0	0	1	
19	0	1	1	0	0	

In [24]:

```
# One-hot encode the "know coverage options" column into Yes/No/Unknown (Unknown =

col = "Do you know the options for mental health care available under your employer

s = df[col].astype(str).str.strip().str.lower()
s = s.replace(["", " ", "n/a", "na", "none", "nan"], np.nan)

s = s.replace({
    "yes": "Yes",
    "no": "No",
    "i am not sure": "Unknown",
    "i'm not sure": "Unknown",
```

```
    "not sure": "Unknown"
}).fillna("Unknown")


cols_to_drop  = ["Why or why not?", "Why or why not?.1"]
df = df.drop(columns=[c for c in cols_to_drop if c in df.columns])


d = pd.get_dummies(s, prefix="mhcare_options", dtype=int)

needed = ["mhcare_options_Yes", "mhcare_options_No", "mhcare_options_Unknown"]
for n in needed:
    if n not in d.columns:
        d[n] = 0
d = d[needed]

print("Know coverage options -> each row has exactly one category:", (d.sum(axis=1)

df = pd.concat([df.drop(columns=[col]), d], axis=1)

d.head(20)
```

```
Know coverage options -> each row has exactly one category: True
```

Out[24]:

	mhcare_options_Yes	mhcare_options_No	mhcare_options_Unknown
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	1	0	0
5	0	0	1
6	0	1	0
7	1	0	0
8	0	1	0
9	0	0	1
10	0	0	1
11	1	0	0
12	0	0	1
13	1	0	0
14	1	0	0
15	0	1	0
16	0	1	0
17	0	1	0
18	0	0	1
19	0	1	0

In [25]:
```
# Save the current dataframe to an Excel file exactly as it is right now.
output_path = r"C:\Users\123wi\OneDrive\Desktop\duits uni\unsupervised\project\ment
df.to_excel(output_path, index=False)
print("Saved to:", output_path)
```

```
Saved to: C:\Users\123wi\OneDrive\Desktop\duits uni\unsupervised\project\mental_heal
th_processed.xlsx
```