

Project Title: Mental Health Data Analysis

Name: Pregya Ganjoo

Course: Bachelors Of Engineering (BE)

College: Vishwakarma Government Engineering College

Date: 04-06-2025

Tools Used: Power BI, MS Excel, MYSQL, Jupyter Notebook

Data Set And Description

- **Dataset:** Mental Health in Tech Survey
- **Format:** CSV file
- **Total Records:** ~1,250 rows
- **Attributes:** Includes age, gender, location, employment status, mental health treatment, workplace support, etc.
- **Time Period:** Mainly 2014, with a few entries from 2015–16

Mental Health Analysis (.ipynb file)

1. Loading the Dataset

The dataset is loaded into the environment using the pandas library, a versatile Python tool designed for data manipulation and analysis. The dataset is typically in CSV format, which is easy to handle and widely supported. Loading the data correctly allows us to inspect and understand the structure and content of the data.

```
] import pandas as pd
```

```
] df = pd.read_csv("C:/Users/khushi/Desktop/mental health data/survey.csv")  
df.head()
```

2. Checking for Missing Values

Real-world datasets often contain missing or null values due to errors during data collection or recording. These missing values can bias the results or cause errors during model training. Hence, identifying the presence and location of null values is essential. This step helps us plan how to address these gaps effectively.

```
# Check null values  
df.isnull().sum()
```

3. Cleaning Column Names

Column names may contain inconsistencies such as leading/trailing spaces, uppercase letters, or special characters. Standardizing column names by converting them to lowercase, removing spaces, or replacing special characters makes the dataset easier to work with. Clean column names improve code readability and reduce the chance of bugs during analysis.

```
: # Rename columns to remove spaces and make SQL-friendly
df.columns = df.columns.str.lower().str.replace(" ", "_").str.replace("-", "_")

: print(df.columns)
```

4. Selecting Relevant Features

Not all columns in a dataset contribute equally to the analysis. Irrelevant or redundant columns can introduce noise and degrade model performance. Selecting relevant features based on the problem context helps to streamline the dataset, reduce complexity, and focus the analysis on meaningful data.

```
|: df_cleaned = df[[  
    'age',  
    'gender',  
    'country',  
    'family_history',  
    'care_options',  
    'benefits',  
    'anonymity',  
    'leave',  
    'mental_health_consequence',  
    'phys_health_consequence',  
    'coworkers'  
]]
```

5. Handling Missing Data (Basic Strategies)

To handle missing values, simple techniques like imputing with the mode (most frequent value) for categorical columns or dropping rows with too many nulls are applied. Although advanced imputation methods exist, basic handling suffices for initial exploration and prevents data loss or distortion.

```
# Fill missing values with "Unknown"  
df_cleaned = df_cleaned.fillna("Unknown")
```

6. Saving the Cleaned Dataset

Once the data is cleaned and preprocessed, saving it as a new file creates a stable, error-free version of the dataset. This clean dataset can then be used in subsequent steps like visualization, feature engineering, and model building without repeatedly processing the raw data.

```
# Save the cleaned file for SQL import  
df_cleaned.to_csv("cleaned_survey.csv", index=False)
```


MYSQL (Performed initial data cleaning and filtering)

Query 1: Total Responses

Explanation:

This query counts the total number of survey respondents. It shows the size of our dataset, which is important to understand the scope and reliability of our analysis.

MySQL Workbench

Local instance MySQL80

FileEditViewQueryDatabaseServerToolsScriptingHelp

Navigator

SQL File 12* x orders

SCHEMAS

Filter objects

churn_1

demo

disney_movie_recommendation

ecommerce

mental_health

Tables

survey

Columns

Indexes

Foreign Keys

Triggers

Views

Stored Procedures

Functions

placement_project

practise

pregyadb

project

queries

sales

sales1

school

sys

Administration

Schemas

Information

Table: survey

Columns:

Timestamp

Age

Gender

Country

state

self_employed

Object Info

Session

1 • SELECT COUNT(*) AS total_responses FROM survey;

2

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

total_responses
1258

Result 1 x

Read Only

Query Completed

USD/INR
+0.42%

23:43
03-06-2025

Query 2: Gender Distribution

Explanation:

This query groups respondents by gender and counts each group. It helps us understand the gender diversity in our survey sample.

SQL File 12* × orders

- churn_1
- demo
- disney_movie_recommendation
- ecommerce
- mental_health**
 - Tables
 - survey
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
 - placement_project
 - practise
 - pregyadb
 - project
 - queries
 - sales
 - sales1
 - school
 - sys

Information :

Columns:
Timestamp
Age
Gender
Country
state
self_employed

Result 2 ✕

Read Only



99+



1



ENG
IN

03-0

23:44
5-2025

23:44
03-06-2025

Query 3: Mental Health Treatment Uptake

Explanation :

Here, we check how many respondents have received mental health treatment versus those who have not. This insight is crucial to gauge how many tech workers seek help.

MySQL Workbench

Local instance MySQL80

FileEditViewQueryDatabaseServerToolsScriptingHelp

Navigator

SCHEMAS

Filter objects

churn_1

demo

disney_movie_recommendation

ecommerce

mental_health

Tables

survey

Columns

Indexes

Foreign Keys

Triggers

Views

Stored Procedures

Functions

placement_project

practise

pregyadb

project

queries

sales

sales1

school

sys

Administration

Schemas

Information

SQL File 12*

orders

Don't Limit

1

2

SELECT treatment, COUNT(*) AS count FROM survey GROUP BY treatment;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	treatment	count
▶	Yes	636
	No	622

Result 3

Read Only

Table: survey

Columns:

Timestamp

Age

Gender

Country

state

self_employed

Object Info

Session

Query Completed

Hot days ahead 30°C

Windows Taskbar

23:45 03-06-2025

Query 4: Work Interference with Treatment

Explanation :

This shows how respondents feel their work interferes with their mental health treatment. It highlights potential workplace challenges impacting mental well-being.

MySQL Workbench

Local instance MySQL80

FileEditViewQueryDatabaseServerToolsScriptingHelp

Navigator

SQL File 12* x orders

```
1 • SELECT work_interfere, COUNT(*) AS count FROM survey GROUP BY work_interfere;
2
```

SCHEMAS

Filter objects

churn_1

demo

disney_movie_recommendation

ecommerce

mental_health

Tables

survey

Columns

Indexes

Foreign Keys

Triggers

Views

Stored Procedures

Functions

placement_project

practise

pregyadb

project

queries

sales

sales1

school

sys

Administration

Schemas

Information

Table: survey

Columns:
Timestamp
Age
Gender
Country
state
self_employed

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

work_interfere	count
Often	143
Rarely	173
Never	213
Sometimes	465
NA	264

Result 4 x

Read Only

Query Completed

Hot days ahead
30°C
























ENG IN 23:46
03-06-2025

Query 5: Average Age by Treatment Status

Explanation :

This calculates the average age of respondents who have and have not received treatment. It helps us understand if age influences seeking mental health support.

SQL File 12* × orders

- ▶  churn_1
- ▶  demo
- ▶  disney_movie_recommendation
- ▶  ecommerce
- ▼  **mental_health**
 - ▼  Tables
 - ▼  survey
 - ▶  Columns
 - ▶  Indexes
 - ▶  Foreign Keys
 - ▶  Triggers
 - ▶  Views
 - ▶  Stored Procedures
 - ▶  Functions
 - ▶  placement_project
 - ▶  practise
 - ▶  pregysadb
 - ▶  project
 - ▶  queries
 - ▶  sales
 - ▶  sales1
 - ▶  school
 - ▶  sys

Information :

Columns:
Timestamp
Age
Gender
Country
state
self_employed

Result 5

 Read Only



^



03-0

23:47
-2025

Query 6: Mental Health Consequences at Work

Explanation :

This query reveals if respondents faced negative consequences at work due to mental health issues, which indicates stigma or discrimination in the workplace.

SQL File 12* × orders

- ▶ churn_1
- ▶ demo
- ▶ disney_movie_recommendation
- ▶ ecommerce
- ▼ **mental_health**

▼ survey

- placement_project
- practise
- pregyadb
- project
- queries
- sales
- sales1
- school
- sys

Information : : : : :

Columns:
Timestamp
Age
Gender
Country
state
self_employed

Result 6 

Read Only



^



23:48
03-06-2025

03-06-2025

Query 7: Anonymity Preference

Explanation :

This checks how many respondents prefer anonymity when seeking mental health help, shedding light on stigma and privacy concerns.

Navigator:.....

SQL File 12* x orders

SCHEMAS

Filter objects

- churn_1
- demo
- disney_movie_recommendation
- ecommerce
- mental_health**
 - Tables
 - survey
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
 - placement_project
 - practise
 - pregyadb
 - project
 - queries
 - sales
 - sales1
 - school
 - sys

Administration Schemas


Information :

Table: survey

Columns:

Timestamp
Age
Gender
Country
state
self_employ

Object Info

Result 7 

Read Only

Query Completed

Query 8: Remote Work Impact

Explanation :

This groups respondents by whether they work remotely, which can influence mental health due to isolation or flexibility.

Navigator:.....

SQL File 12* x orders

Filter objects

- churn_1
- demo
- disney_movie_recommendation
- ecommerce
- mental_health**
 - Tables
 - survey
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
 - placement_project
 - practise
 - pregyadb
 - project
 - queries
 - sales
 - sales1
 - school
 - sys

Administration Schemas

Information

Table: survey

Columns:

Timestamp
Age
Gender
Country
state
self_employ

Object Info

Result 8 

Read Only

Query Completed

4 30°C
Smoke



🔍 Search



ENG
IN



23:54
06-2025

Query 9: Benefit Availability
























Explanation :

This checks how many respondents have mental health benefits offered by their employer, an important factor for treatment accessibility.

Query 10: Leave Policy Impact


Explanation :

This explores if respondents feel the leave policies at their workplace support mental health needs, affecting their ability to take time off when necessary.

- ▶  churn_1
- ▶  demo
- ▶  disney_movie_recommendation
- ▶  ecommerce
- ▼  **mental_health**
 - ▼  Tables
 - ▼  survey
 - ▶  Columns
 - ▶  Indexes
 - ▶  Foreign Keys
 - ▶  Triggers
 - ▶  Views
 - ▶  Stored Procedures
 - ▶  Functions
 - ▶  placement_project
 - ▶  practise
 - ▶  pregysadb
 - ▶  project
 - ▶  queries
 - ▶  sales
 - ▶  sales1
 - ▶  school
 - ▶  sys

Information

Columns:
Timestamp
Age
Gender
Country
state
self_employed

Result 10 

Read Only



^



03-0

23:56
-2025

Mental Health Analysis SQL Connector

1. Connecting to MySQL Database

Using the **my sql-connector-python** or **SQL Alchemy** library, we established a connection to a MySQL database. This is a vital skill when working with large-scale enterprise data, as most organizational data resides in relational databases. The connection allows us to run SQL queries directly from Python and import data into a Pandas Data Frame for further analysis.

```
import mysql.connector
import pandas as pd

# Connect to MySQL
conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='pregya123',
    database='mental_health'
)

print("MySQL Connection Successful!")
```

MySQL Connection Successful!

```
from sqlalchemy import create_engine
import pandas as pd

# Create connection engine (replace with your creds)
engine = create_engine("mysql+pymysql://root:pregya123@localhost/mental_health")

# Query to fetch data
query = "SELECT * FROM survey;"

# Load data into pandas dataframe
df = pd.read_sql(query, engine)

print("Data Loaded:")
print(df.head())
```

2. Exploring the Data

After loading the data from MySQL into a Data Frame, we explored its structure using functions like `.head()`, `.info()`, and `.describe()`. This helps us understand the number of rows, data types, distribution, and potential inconsistencies in the dataset.

```
# Shape and info
print("Data shape:", df.shape)
print(df.info())

# Check missing values
print("Missing values:\n", df.isnull().sum())

# Quick stats on numeric columns
print(df.describe())
```

3. Cleaning the Data (If Needed)

Depending on the output from the exploration step, basic cleaning was applied. This included:

- Renaming columns for better readability
- Handling null or inconsistent values
- Ensuring data types matched the intended use (e.g., converting text numbers to integers)
- Clean data is essential to prevent errors and inconsistencies during analysis or visualization.

```
# Drop rows with missing values (if any)
df_clean = df.dropna()

# Convert Age column to integer (if needed)
df_clean['Age'] = df_clean['Age'].astype(int)

print("After cleaning:")
print(df_clean.info())
```

4. SQL Analysis with Pandas

Instead of using pure SQL syntax in a MySQL interface, we used SQL-like queries in Pandas (`df.query()`, `groupby()`, etc.) or even raw SQL queries via the connection. This hybrid approach combines the familiarity of SQL with the flexibility of Python. We ran analysis like:

- Counting categorical entries
- Grouping data based on certain features
- Filtering data for specific conditions
- These queries helped derive actionable insights from the data.

```
# How many people received treatment? Group by treatment status
treatment_count = pd.read_sql("SELECT treatment, COUNT(*) as count FROM survey GROUP BY treatment;", engine)
print(treatment_count)

# Average age by gender
avg_age = pd.read_sql("SELECT Gender, AVG(Age) as avg_age FROM survey GROUP BY Gender;", engine)
print(avg_age)
```


5. Visualization with Matplotlib and Seaborn

To make the analysis more interpretable, we created visualizations using:

- Matplotlib for basic plots (bar charts, line graphs)
- Seaborn for advanced statistical visualizations (box plots, heatmaps)
- These charts helped communicate trends, distributions, and relationships clearly and visually.

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

```
# Count of people who received treatment (bar plot)
sns.countplot(x='treatment', data=df_clean)
plt.title("Count of Treatment Received")
plt.show()
```

6. Exploring Clean Data for Reporting

Once visualizations and analyses were done, the cleaned and processed data was reviewed for consistency and final reporting. This step ensured that all missing values were handled, insights were clearly interpretable, and any anomalies were flagged or explained.

```
: # Export to CSV to use in Power BI or Excel
df_clean.to_csv('cleaned_mental_health_survey.csv', index=False)
print("Cleaned data exported as CSV!")
```

7. (Optional) Saving the Visualizations

To document and reuse the visuals in presentations or reports, the charts were saved as .png or .jpg files using `plt.savefig()`. This step ensures that the visuals can be included in offline reports or embedded into dashboards and web pages.

```
: plt.figure()
  sns.countplot(x='treatment', data=df_clean)
  plt.title("Count of Treatment Received")
  plt.savefig('treatment_count.png')
```

Power BI Dashboards and Visualizations

The goal of using Power BI in this project was to perform a comprehensive visual analysis of the mental health survey dataset. The focus was on transforming raw data into a clean, structured form and then using a wide range of chart types to uncover patterns, trends, and insights that are difficult to extract through tabular data alone. Power BI was chosen for its robust data modeling, transformation, and visualization capabilities.

Data Loading and Preparation:

The process began with loading a CSV dataset into Power BI. This was done by selecting the “Folder” option in Power BI’s “Get Data” feature, enabling the consolidation of data files stored in a single directory. Upon loading, Power BI automatically navigated to Power Query Editor, which allowed a deeper transformation of the dataset.

In Power Query Editor, several data cleaning and preparation tasks were performed:

- Headers were promoted to represent field names.
- Columns were renamed to be more descriptive and standardized.
- Column data types were explicitly defined. For instance, the Timestamp column was converted to datetime, Age to whole number, and all categorical fields to text.
- Invalid values and missing data were addressed to ensure consistency. For example, fields such as no_employees that had mixed types (text and numeric) were cleaned or standardized.
- Records from years other than 2014 were sparse and scattered, so the focus remained on 2014 data, which formed the bulk of the responses.

Visualization Strategy and Chart Types:

To provide a multidimensional analysis of the data, a diverse set of visualizations was created. These visuals were placed together on a single Power BI report page. Each chart served a unique purpose in interpreting different aspects of the dataset. The following chart types were utilized:

- **Stacked Column Chart:**

This chart type was used to represent categorical variables such as treatment received across different countries or genders. The stacking helped compare subgroups within each category.

- **Stacked Area Chart:**

This visualization helped track changes and cumulative values over time, although the dataset's time distribution was skewed mostly toward 2014. It allowed an intuitive view of how different groups contributed to totals.

- **Pie Chart:**

Used to visualize proportions of categories such as gender or remote work preference. Pie charts provided an immediate understanding of distribution across a single categorical variable.

- **Waterfall Chart:**

This chart illustrated incremental changes, which was helpful in demonstrating the flow of responses or the relative contribution of various consequences to mental health outcomes.

- **Scatter Chart:**

Scatter plots were used to identify relationships between numerical variables, such as age and mental health treatment, or age and perception of mental health support.

- **Line Chart:**

Useful for showing trends and progression over time. Although the timestamp data was limited mostly to one year, the line chart served to highlight patterns in sequential variables or monthly breakdowns.

- **100% Stacked Column Chart:**

This visual was chosen to display percentage contributions of categories to a whole. It was particularly effective in comparing normalized proportions across categories, such as different company sizes or employment types.

Donut Chart:

Functionally similar to the pie chart, the donut chart was used to reinforce proportional data while also providing a central space to label totals or key metrics.

Dashboard Design and Layout:

The report was designed to be informative and user-friendly. All eight charts were arranged on a single report page for better comparative analysis and storytelling. Care was taken to label each chart clearly, add legends where necessary, and apply a consistent visual theme across all visuals.

The interactivity of Power BI was leveraged through slicers and filters. This allowed viewers to dynamically explore the data by selecting different filters such as gender, year, or treatment status. Clicking on one visual would automatically update the others due to cross-filtering.

Exporting and Sharing the Report:

Once the dashboard was finalized, it was saved in .pbix format for ongoing edits and review. Additionally, the complete report page was exported as a high-quality PDF to be included as a part of the project documentation. Each chart was also screenshotted and inserted into the main report to ensure clarity for reviewers who may not access the Power BI dashboard interactively.

Conclusion:

The use of Power BI significantly enhanced the ability to interpret and present the dataset's key patterns and insights. Through its data transformation tools and a variety of rich visualizations, Power BI enabled the creation of a coherent, interactive, and informative dashboard. The visual representations allowed for faster and more intuitive understanding of complex data, making it a powerful tool in the overall data analysis process of this project.

MS Excel Pivot Table and Visualization

Objective:

The main goal of using MS Excel was to create a quick and structured summary of the mental health survey data using **pivot tables** and support it with basic visualizations. This helped in understanding the data distribution and extracting patterns without advanced tools.

Data Preparation:

The cleaned CSV file was loaded into Excel and converted into a proper table format to make it easier to use with pivot tables.

Any remaining formatting issues or blank values were handled manually where needed.

Pivot Table Creation:

- Multiple pivot tables were created to break down data by relevant categories such as Gender, Country, Treatment, Work Interfere, and more.
- Each pivot table was kept compact—3 to 4 columns max—to avoid clutter, and spread across multiple pages when required.
- Calculations were mostly based on counts, helping identify how many respondents fell into specific categories (e.g., how many males received treatment, how many said their work interferes with mental health).

Visualizations Used:

Each pivot table was paired with a chart to give it visual meaning. Here are the ones you used:

- **Pie Chart:**
Used to show proportion—like treatment distribution among different genders or work interference levels.
- **Clustered Column Bar Chart:**
This helped in comparing two or more groups side-by-side. For example, how responses varied between tech and non-tech companies.
- **100% Stacked Column Chart:**
Useful to show the percentage breakdowns of categories, especially when analyzing answers across company sizes or countries.

Data Set And Description

Data Set And Description

Data Set And Description

Data Set And Description

Data Set And Description