Python Report

1. Initial Setup and Data Loading

First, necessary Python libraries for data manipulation and visualization were imported, including `pandas` for handling data, `numpy` for numerical operations, and `matplotlib` and `seaborn` for creating plots. The dataset, named `anxiety_depression_data.csv`, was then loaded into a pandas DataFrame for analysis.

Imports

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import os
import missingno as msno
from sklearn.preprocessing import LabelEncoder, StandardScaler
warnings.filterwarnings('ignore')
```

Reading Data

```
path = '/content/anxiety_depression_data.csv'
```

```
df = pd.read_csv(path)
```

2. Data Understanding and Missing Value Analysis

The initial step was to understand the dataset's basic characteristics. This involved:

- Checking the dimensions of the data (`df.shape`).
- Viewing the first few rows (`df.head()`).
- Generating descriptive statistics for both numerical and categorical columns (`df.describe()`).
- Getting a summary of the data types and non-null counts (`df.info()`).

```
df.shape
```

```
(1200, 21)
```

```
df.head()
```

	Age	Gender	Education_Level	Employment_Status	Sleep_Hours	Physical_Activity_Hrs	Social_Support_Score	Anxiety_Score	Depression_Score	Stress_Level	...	Chronic_Illnesses
0	56	Male	Bachelor's	Unemployed	6.0	0.4	3	4	2	9	...	0
1	69	Female	Bachelor's	Retired	8.8	2.8	6	18	7	6	...	0
2	46	Female	Master's	Employed	5.3	1.6	5	5	13	8	...	0
3	32	Female	High School	Unemployed	8.8	0.5	4	6	3	4	...	1
4	60	Female	Bachelor's	Retired	7.2	0.7	2	7	15	3	...	0

5 rows × 21 columns

```
df.describe()
```

	Age	Sleep_Hours	Physical_Activity_Hrs	Social_Support_Score	Anxiety_Score	Depression_Score	Stress_Level	Family_History_Mental_Illness	Chronic_Illnesses	Th
count	1200.000000	1200.00000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.00000	1200.0
mean	46.317500	6.46900	2.005750	5.055000	10.470000	10.674167	5.000833	0.318333	0.26750	0.2
std	16.451157	1.52955	2.037818	2.652893	5.911138	5.632889	2.538281	0.466024	0.44284	0.4
min	18.000000	2.00000	0.000000	1.000000	1.000000	1.000000	1.000000	0.000000	0.00000	0.0
25%	33.000000	5.40000	0.600000	3.000000	5.000000	6.000000	3.000000	0.000000	0.00000	0.0
50%	46.000000	6.40000	1.400000	5.000000	10.500000	11.000000	5.000000	0.000000	0.00000	0.0
75%	61.000000	7.50000	2.700000	7.000000	16.000000	15.000000	7.000000	1.000000	1.00000	0.0
max	74.000000	12.40000	15.100000	9.000000	20.000000	20.000000	9.000000	1.000000	1.00000	1.0

```
df.describe(include='object')
```

	Gender	Education_Level	Employment_Status	Medication_Use	Substance_Use
count	1200	1200	1200	453	366
unique	4	5	4	2	2
top	Female	PhD	Employed	Regular	Occasional
freq	569	262	320	238	242

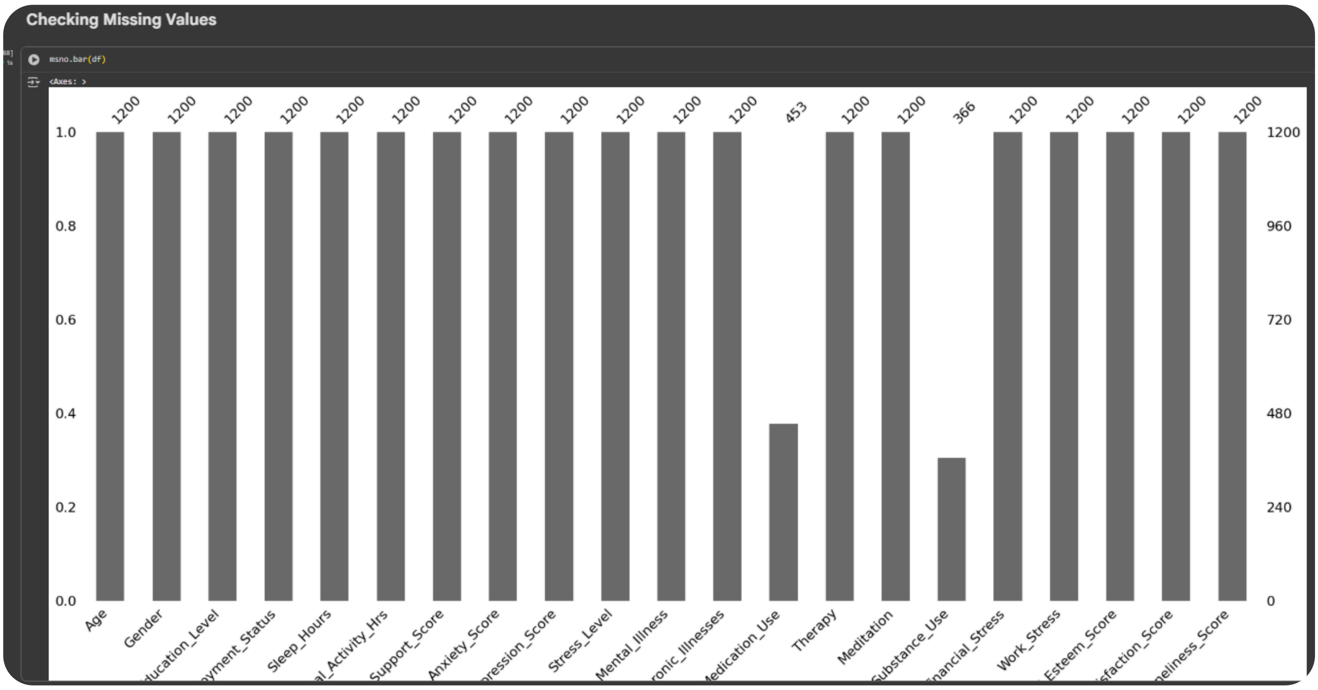
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 21 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Age                           1200 non-null   int64
 1   Gender                        1200 non-null   object
 2   Education_Level               1200 non-null   object
 3   Employment_Status             1200 non-null   object
 4   Sleep_Hours                   1200 non-null   float64
 5   Physical_Activity_Hrs         1200 non-null   float64
 6   Social_Support_Score          1200 non-null   int64
 7   Anxiety_Score                 1200 non-null   int64
 8   Depression_Score              1200 non-null   int64
 9   Stress_Level                  1200 non-null   int64
 10  Family_History_Mental_Illness 1200 non-null   int64
 11  Chronic_Illnesses             1200 non-null   int64
 12  Medication_Use                453 non-null    object
 13  Therapy                       1200 non-null   int64
 14  Meditation                    1200 non-null   int64
 15  Substance_Use                 366 non-null    object
 16  Financial_Stress              1200 non-null   int64
 17  Work_Stress                   1200 non-null   int64
 18  Self_Esteem_Score             1200 non-null   int64
 19  Life_Satisfaction_Score       1200 non-null   int64
 20  Loneliness_Score              1200 non-null   int64
dtypes: float64(2), int64(14), object(5)
memory usage: 197.0+ KB
```

A crucial part of this phase was to identify missing values. A bar chart was generated to visualize the completeness of each column. The analysis revealed that the **'Substance_Use' and 'Medication_Use' columns had a significant percentage of missing data**, with over 60% of their values being null. Due to this high volume of missing information, these two columns were dropped from the dataset to avoid introducing bias or errors in the analysis.

Additionally, the descriptive statistics generated by `df.describe()` showed that the 'Physical_Activity_Hrs' column had a maximum value of 15. The idea that a person could engage in 15 hours of physical activity in a single day seems highly improbable and suggests a likely data entry error or an outlier that needs to be addressed. This finding reinforces the need for the subsequent outlier detection and treatment step.

(أنا شاكه انه Physical_Activity_Hrs مليان outliers بس معنديش دليل)




```
percent_missing = df.isnull().sum() * 100 / len(df)
missing_value_df = pd.DataFrame({'column_name': df.columns,'percent_missing': percent_missing})
missing_value_df.sort_values('percent_missing', inplace=True)
missing_value_df
```

Self_Esteem_Score	Self_Esteem_Score	0.00
Medication_Use	Medication_Use	62.25
Substance_Use	Substance_Use	69.50

```
df['Medication_Use'].unique()
```

```
array([nan, 'Occasional', 'Regular'], dtype=object)
```

```
df['Substance_Use'].unique()
```

```
array([nan, 'Frequent', 'Occasional'], dtype=object)
```

```
df.drop(['Substance_Use', 'Medication_Use'], axis=1, inplace=True)
```

3. Data Cleaning and Standardization

Several cleaning and standardization steps were performed to improve data quality:

- **Standardizing Binary Columns:** The columns 'Meditation', 'Therapy', 'Chronic_Illnesses', and 'Family_History_Mental_Illness' contained numerical values (0 and 1) representing binary states. To improve readability and consistency, these were replaced with 'No' and 'Yes' respectively.

```
df['Meditation'].unique()
```

```
array([1, 0])
```

```
df['Chronic_Illnesses'].unique()
```

```
array([0, 1])
```

```
df['Therapy'].unique()
```

```
array([0, 1])
```

```
df['Family_History_Mental_Illness'].unique()
```

```
array([0, 1])
```

```
cols = ['Meditation', 'Therapy', 'Chronic_Illnesses','Family_History_Mental_Illness']
df[cols] = df[cols].replace({0: 'No', 1: 'Yes'})
```

- **Cleaning the 'Gender' Column:** The 'Gender' column contained 'Other' and 'Non-Binary' categories. To simplify the feature, these values were replaced with the mode (the most frequently occurring value) of the 'Gender' column.

```
mode_gender = df['Gender'].mode()[0]
df['Gender'] = df['Gender'].replace(['Other', 'Non-Binary'], mode_gender)
```

- **Checking for Duplicates:** The dataset was checked for any duplicated rows, and none were found.

```
df.duplicated().sum()
np.int64(0)
```

4. Outlier Detection and Treatment

Outliers, which are extreme values that can skew analysis, were identified in the numerical columns.

- **Detection:** The **Interquartile Range (IQR) method** was used for outlier detection. This method calculates the range between the first quartile (Q1) and the third quartile (Q3) and defines outliers as any data points that fall below `Q1 - 1.5 * IQR` or above `Q3 + 1.5 * IQR`. The analysis showed outliers were present in the 'Physical_Activity_Hrs' and 'Sleep_Hours' columns. Box plots were also used to visually confirm the presence of these outliers.

(الحمدلله بقى عندي دليل)

```python
def check_outliers(df, columns):
    """Simple outlier detection using IQR method"""

    for col in columns:
        # Calculate IQR
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1

        # Define outlier bounds
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Find outliers
        outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]

        print(f"\n{col}:")
        print(f"  Normal range: {lower_bound:.2f} to {upper_bound:.2f}")
        print(f"  Number of outliers: {len(outliers)}")
        print(f"  Outlier percentage: {len(outliers)/len(df)*100:.1f}%")

        if len(outliers) > 0:
            print(f"  Outlier values: {sorted(outliers[col].values)}")
```

```
Sleep_Hours:
  Normal range: 2.25 to 10.65
  Number of outliers: 6
  Outlier percentage: 0.5%
  Outlier values: [np.float64(2.0), np.float64(2.1), np.float64(2.1), np.float64(10.8), np.float64(11.4), np.float64(12.4)]

Physical_Activity_Hrs:
  Normal range: -2.55 to 5.85
  Number of outliers: 75
  Outlier percentage: 6.2%
  Outlier values: [np.float64(5.9), np.float64(5.9), np.float64(5.9), np.float64(6.0), np.float64(6.0), np.float64(6.1), np.float64(6.1), np.float64(6.1),
```

- **Treatment:** To handle these outliers, a technique called **clipping** was applied. This method replaces the outlier values with the calculated upper and lower bounds of the normal range. This was done for both 'Physical_Activity_Hrs' and 'Sleep_Hours', effectively bringing the extreme values into a more reasonable range without removing the data points entirely. After treatment, a check confirmed that there were zero outliers remaining in these columns.

```python
def treat_outliers_iqr(df, column_name):

    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers_before = len(df[(df[column_name] < lower_bound) | (df[column_name] > upper_bound)])
    print(f"Outliers before treatment: {outliers_before}")

    df[column_name] = df[column_name].clip(lower=lower_bound, upper=upper_bound)

    outliers_after = len(df[(df[column_name] < lower_bound) | (df[column_name] > upper_bound)])
    print(f"Outliers after treatment: {outliers_after}")
    print(f"New range: {df[column_name].min():.2f} to {df[column_name].max():.2f}")

    return df

print("=" * 50)
df = treat_outliers_iqr(df, 'Physical_Activity_Hrs')

print("\n" + "=" * 50)
df = treat_outliers_iqr(df, 'Sleep_Hours')
```

```python
check_outliers(df, df_numerical.columns)

df_numerical.boxplot(figsize=(12, 6))
plt.title('Box Plots - Outliers shown as dots')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Age:
  Normal range: -9.00 to 103.00
  Number of outliers: 0
  Outlier percentage: 0.0%

Sleep_Hours:
  Normal range: 2.25 to 10.65
  Number of outliers: 0
  Outlier percentage: 0.0%

Physical_Activity_Hrs:
  Normal range: -2.55 to 5.85
  Number of outliers: 0
  Outlier percentage: 0.0%
```

5. Correcting Data Skewness

The distribution of numerical data was examined to check for skewness, which is a measure of asymmetry.

- **Identification:** The 'Physical_Activity_Hrs' column was identified as having high positive skewness, meaning the data was heavily concentrated on the left side with a long tail to the right. A histogram confirmed this skewed distribution.

(و كمان فيه skewness)

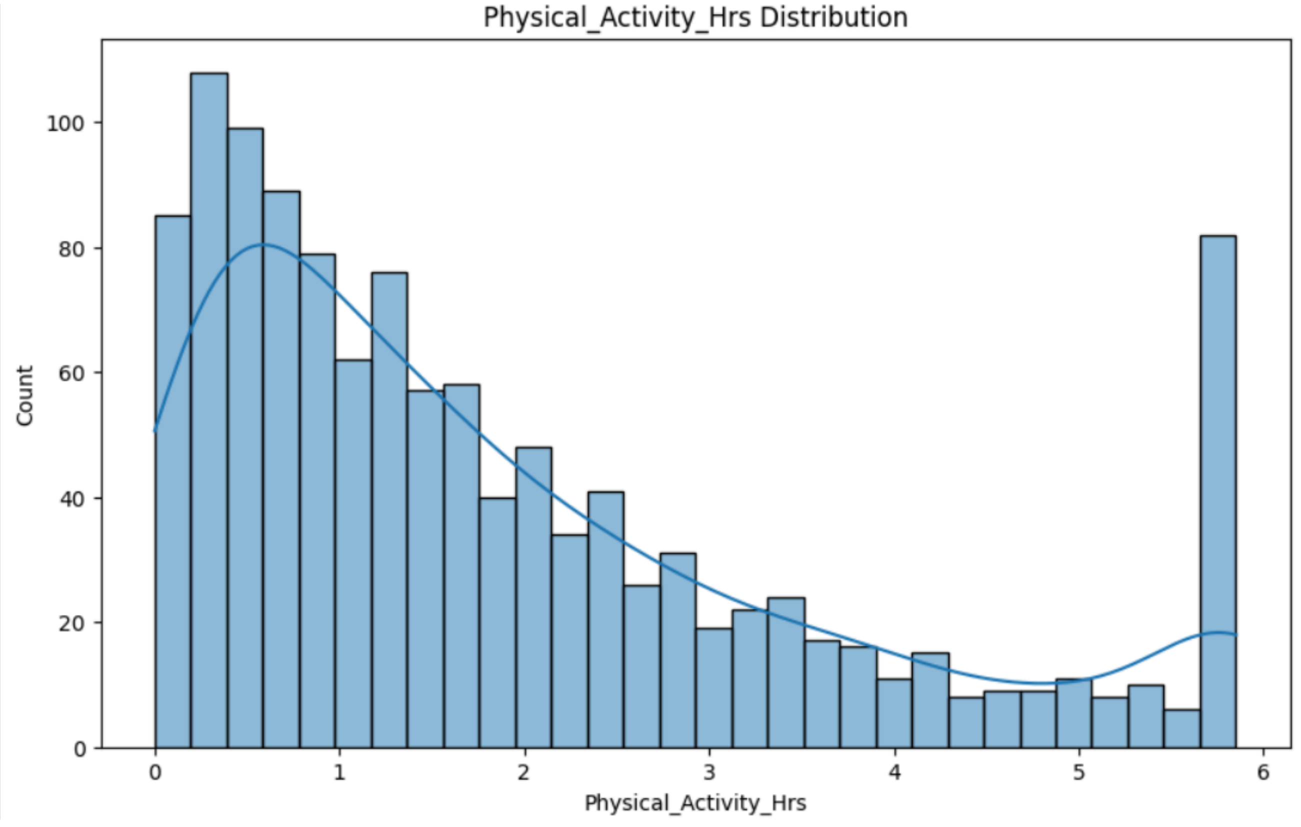
```python
print("SKEWNESS CHECK:")
print("="*40)

for col in df_numerical:
    skewness = df[col].skew()

    if abs(skewness) < 0.5:
        status = "✅ Normal"
    elif abs(skewness) < 1.0:
        status = "⚠️ Moderate"
    else:
        status = "❌ High - needs fix"

    print(f"{col}: {skewness:.2f} {status}")
```

```
SKEWNESS CHECK:
========================================
Age: -0.04 ✅ Normal
Sleep_Hours: -0.01 ✅ Normal
Physical_Activity_Hrs: 1.05 ❌ High - needs fix
Social_Support_Score: -0.04 ✅ Normal
Anxiety_Score: -0.03 ✅ Normal
Depression_Score: -0.05 ✅ Normal
Stress_Level: -0.00 ✅ Normal
Financial_Stress: 0.01 ✅ Normal
Work_Stress: 0.04 ✅ Normal
Self_Esteem_Score: -0.02 ✅ Normal
Life_Satisfaction_Score: -0.03 ✅ Normal
Loneliness_Score: -0.00 ✅ Normal
```



- **Transformation:** To correct this, a **logarithmic transformation** (**np.log1p**) was applied to the 'Physical_Activity_Hrs' column. This transformation is effective at reducing positive skewness and making the data distribution more normal, which is often a requirement for machine learning models. A subsequent check and histograms confirmed that the skewness was successfully reduced to a normal level.

```
df['Physical_Activity_Hrs'] = np.log1p(df['Physical_Activity_Hrs'])
```

```
SKEWNESS CHECK:
========================================
Age: -0.04 ✅ Normal
Sleep_Hours: -0.01 ✅ Normal
Physical_Activity_Hrs: 0.27 ✅ Normal
Social_Support_Score: -0.04 ✅ Normal
Anxiety_Score: -0.03 ✅ Normal
Depression_Score: -0.05 ✅ Normal
Stress_Level: -0.00 ✅ Normal
Financial_Stress: 0.01 ✅ Normal
Work_Stress: 0.04 ✅ Normal
Self_Esteem_Score: -0.02 ✅ Normal
Life_Satisfaction_Score: -0.03 ✅ Normal
Loneliness_Score: -0.00 ✅ Normal
```

6. Feature Encoding and Scaling

The final step was to prepare the data for machine learning by converting it into a suitable numerical format.

- **Label Encoding:** All categorical columns were transformed into numerical representations using **Label Encoding**. This process assigns a unique integer to each unique category within a column.
- **Standard Scaling (Normalization):** After all columns were in a numerical format, all numerical features were scaled using **Standard Scaling**. This process standardizes features by subtracting the mean and dividing by the standard deviation. The result is that each numerical column has a mean of 0 and a standard deviation of 1, ensuring that all features are on a comparable scale and preventing features with larger ranges from dominating the model's learning process.

```
print("1. Label Encoding...")

for col in df_categorical:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    print(f"{col} encoded")

print("\n2. Standard Scaling...")
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns

scaler = StandardScaler()
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
print("All numerical columns scaled")

print(f"\nDone! Dataset ready for analysis.")
print(f"Shape: {df.shape}")
```

```
1. Label Encoding...
Gender encoded
Education_Level encoded
Employment_Status encoded
Family_History_Mental_Illness encoded
Chronic_Illnesses encoded
Therapy encoded
Meditation encoded

2. Standard Scaling...
All numerical columns scaled

Done! Dataset ready for analysis.
Shape: (1200, 19)
```

After these steps, the cleaned, processed, and normalized dataset was ready for further analysis and model building.