In [1]:
```
## Import all libreries

import numpy as np # linear algebra
import pandas as pd # data preprocessinag
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Title : Student Depression Dataset

Description: This dataset contains information on students' mental health, focusing on depression levels, lifestyle factors, academic performance, and social influences. It includes features such as age, gender, sleep patterns, study hours, social interactions, stress levels, and self-reported depression scores. The dataset can be used for analyzing mental health trends, identifying risk factors, and developing predictive models for student well-being

Import Dataset

In [4]:
```
df = pd.read_csv(r"C:\Users\chitt\Downloads\student_depression_dataset.csv")
df
```

Out[4]:

	id	Gender	Age	City	Profession	Academic Pressure	Work Pressure	CGPA	S
0	2	Male	33.0	Visakhapatnam	Student	5.0	0.0	8.97	
1	8	Female	24.0	Bangalore	Student	2.0	0.0	5.90	
2	26	Male	31.0	Srinagar	Student	3.0	0.0	7.03	
3	30	Female	28.0	Varanasi	Student	3.0	0.0	5.59	
4	32	Female	25.0	Jaipur	Student	4.0	0.0	8.13	
...	...	...	...	...	...	...	...	...	
27896	140685	Female	27.0	Surat	Student	5.0	0.0	5.75	
27897	140686	Male	27.0	Ludhiana	Student	2.0	0.0	9.40	
27898	140689	Male	31.0	Faridabad	Student	3.0	0.0	6.61	
27899	140690	Female	18.0	Ludhiana	Student	5.0	0.0	6.88	
27900	140699	Male	27.0	Patna	Student	4.0	0.0	9.24	

27901 rows × 18 columns

In [5]:
```
df.head()
```

Out[5]:

	id	Gender	Age	City	Profession	Academic Pressure	Work Pressure	CGPA	Study Satisfactior
0	2	Male	33.0	Visakhapatnam	Student	5.0	0.0	8.97	2.(
1	8	Female	24.0	Bangalore	Student	2.0	0.0	5.90	5.(
2	26	Male	31.0	Srinagar	Student	3.0	0.0	7.03	5.(
3	30	Female	28.0	Varanasi	Student	3.0	0.0	5.59	2.(
4	32	Female	25.0	Jaipur	Student	4.0	0.0	8.13	3.(

In [6]:
```
df.tail()
```

Out[6]:

	id	Gender	Age	City	Profession	Academic Pressure	Work Pressure	CGPA	S Satisfa
27896	140685	Female	27.0	Surat	Student	5.0	0.0	5.75	
27897	140686	Male	27.0	Ludhiana	Student	2.0	0.0	9.40	
27898	140689	Male	31.0	Faridabad	Student	3.0	0.0	6.61	
27899	140690	Female	18.0	Ludhiana	Student	5.0	0.0	6.88	
27900	140699	Male	27.0	Patna	Student	4.0	0.0	9.24	

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27901 entries, 0 to 27900
Data columns (total 18 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   id                                     27901 non-null  int64
 1   Gender                                 27901 non-null  object
 2   Age                                    27901 non-null  float64
 3   City                                   27901 non-null  object
 4   Profession                             27901 non-null  object
 5   Academic Pressure                      27901 non-null  float64
 6   Work Pressure                          27901 non-null  float64
 7   CGPA                                   27901 non-null  float64
 8   Study Satisfaction                     27901 non-null  float64
 9   Job Satisfaction                       27901 non-null  float64
 10  Sleep Duration                         27901 non-null  object
 11  Dietary Habits                         27901 non-null  object
 12  Degree                                 27901 non-null  object
 13  Have you ever had suicidal thoughts ?  27901 non-null  object
 14  Work/Study Hours                       27901 non-null  float64
 15  Financial Stress                       27901 non-null  object
 16  Family History of Mental Illness       27901 non-null  object
 17  Depression                             27901 non-null  int64
dtypes: float64(7), int64(2), object(9)
memory usage: 3.8+ MB
```

In [8]:
```
df.describe()
```

Out[8]:

	id	Age	Academic Pressure	Work Pressure	CGPA	S Satisfa
count	27901.000000	27901.000000	27901.000000	27901.000000	27901.000000	27901.00
mean	70442.149421	25.822300	3.141214	0.000430	7.656104	2.94
std	40641.175216	4.905687	1.381465	0.043992	1.470707	1.36
min	2.000000	18.000000	0.000000	0.000000	0.000000	0.00
25%	35039.000000	21.000000	2.000000	0.000000	6.290000	2.00
50%	70684.000000	25.000000	3.000000	0.000000	7.770000	3.00
75%	105818.000000	30.000000	4.000000	0.000000	8.920000	4.00
max	140699.000000	59.000000	5.000000	5.000000	10.000000	5.00

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
id                                      0
Gender                                  0
Age                                     0
City                                    0
Profession                              0
Academic Pressure                       0
Work Pressure                           0
CGPA                                    0
Study Satisfaction                      0
Job Satisfaction                        0
Sleep Duration                          0
Dietary Habits                          0
Degree                                  0
Have you ever had suicidal thoughts ?   0
Work/Study Hours                        0
Financial Stress                        0
Family History of Mental Illness        0
Depression                              0
dtype: int64
```

In [10]:
```
df.duplicated().sum()
```

Out[10]:
```
0
```

In [11]:
```
df.dtypes
```

Out[11]:
```
id                                       int64
Gender                                  object
Age                                    float64
City                                    object
Profession                              object
Academic Pressure                      float64
Work Pressure                          float64
CGPA                                   float64
Study Satisfaction                     float64
Job Satisfaction                       float64
Sleep Duration                          object
Dietary Habits                          object
Degree                                  object
Have you ever had suicidal thoughts ?   object
Work/Study Hours                       float64
Financial Stress                        object
Family History of Mental Illness        object
Depression                               int64
dtype: object
```

In [12]:
```
df.shape
```

Out[12]:
```
(27901, 18)
```

In [13]:
```
df.corr
```

Out[13]:
```
<bound method DataFrame.corr of            id  Gender   Age           City Prof
ession  Academic Pressure  \
0           2    Male  33.0  Visakhapatnam    Student               5.0
1           8  Female  24.0      Bangalore    Student               2.0
2          26    Male  31.0       Srinagar    Student               3.0
3          30  Female  28.0       Varanasi    Student               3.0
4          32  Female  25.0         Jaipur    Student               4.0
...       ...     ...   ...            ...        ...               ...
27896  140685  Female  27.0          Surat    Student               5.0
27897  140686    Male  27.0       Ludhiana    Student               2.0
27898  140689    Male  31.0      Faridabad    Student               3.0
27899  140690  Female  18.0       Ludhiana    Student               5.0
27900  140699    Male  27.0          Patna    Student               4.0

       Work Pressure  CGPA  Study Satisfaction  Job Satisfaction  \
0                0.0  8.97                 2.0               0.0
1                0.0  5.90                 5.0               0.0
2                0.0  7.03                 5.0               0.0
3                0.0  5.59                 2.0               0.0
4                0.0  8.13                 3.0               0.0
...              ...   ...                 ...               ...
27896            0.0  5.75                 5.0               0.0
27897            0.0  9.40                 3.0               0.0
27898            0.0  6.61                 4.0               0.0
27899            0.0  6.88                 2.0               0.0
27900            0.0  9.24                 1.0               0.0

            Sleep Duration Dietary Habits      Degree  \
0              '5-6 hours'        Healthy     B.Pharm
1              '5-6 hours'       Moderate         BSc
2      'Less than 5 hours'        Healthy          BA
3              '7-8 hours'       Moderate         BCA
4              '5-6 hours'       Moderate      M.Tech
...                    ...            ...         ...
27896          '5-6 hours'      Unhealthy  'Class 12'
27897  'Less than 5 hours'        Healthy         MSc
27898          '5-6 hours'      Unhealthy          MD
27899  'Less than 5 hours'        Healthy  'Class 12'
27900  'Less than 5 hours'        Healthy         BCA

      Have you ever had suicidal thoughts ?  Work/Study Hours  \
0                                       Yes               3.0
1                                        No               3.0
2                                        No               9.0
3                                       Yes               4.0
4                                       Yes               1.0
...                                     ...               ...
27896                                   Yes               7.0
27897                                    No               0.0
27898                                    No              12.0
27899                                   Yes              10.0
27900                                   Yes               2.0

      Financial Stress Family History of Mental Illness  Depression
0                  1.0                              No           1
1                  2.0                             Yes           0
2                  1.0                             Yes           0
3                  5.0                             Yes           1
4                  1.0                              No           0
...                ...                             ...         ...
```

```
27896             1.0                                Yes          0
27897             3.0                                Yes          0
27898             2.0                                 No          0
27899             5.0                                 No          1
27900             3.0                                Yes          1

[27901 rows x 18 columns]>
```

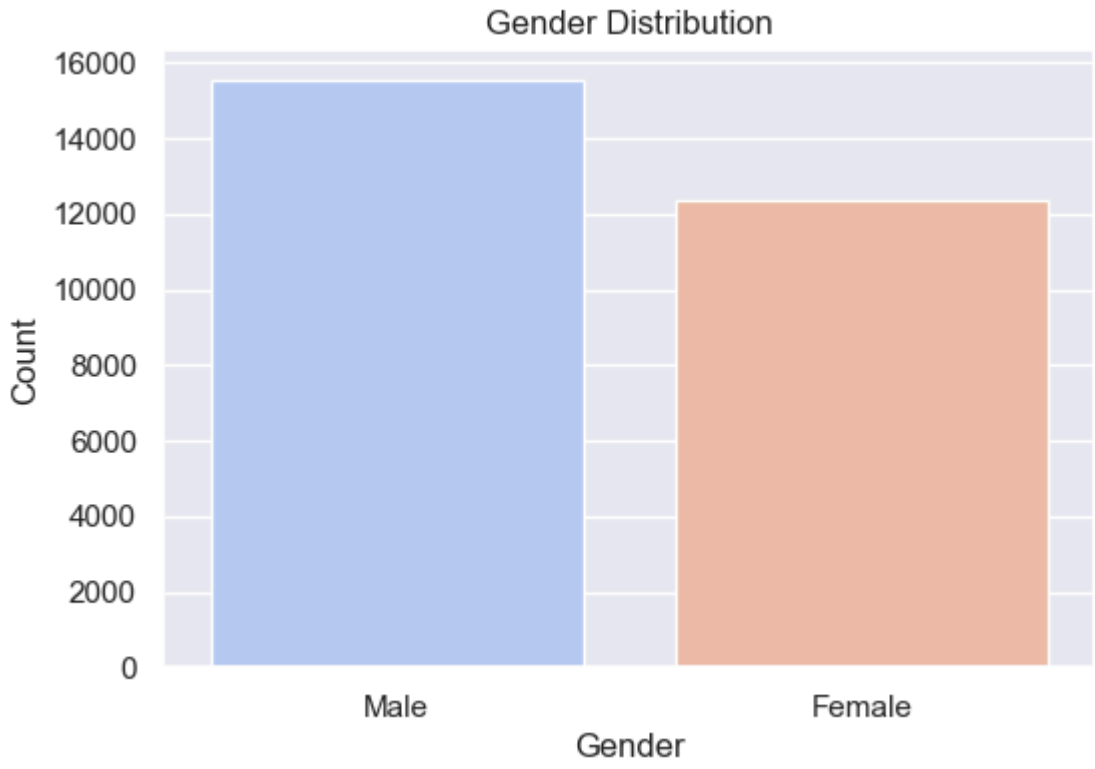
In [14]:
```
df.columns
```

Out[14]:
```
Index(['id', 'Gender', 'Age', 'City', 'Profession', 'Academic Pressure',
       'Work Pressure', 'CGPA', 'Study Satisfaction', 'Job Satisfaction',
       'Sleep Duration', 'Dietary Habits', 'Degree',
       'Have you ever had suicidal thoughts ?', 'Work/Study Hours',
       'Financial Stress', 'Family History of Mental Illness', 'Depression'],
      dtype='object')
```

Data Visualization

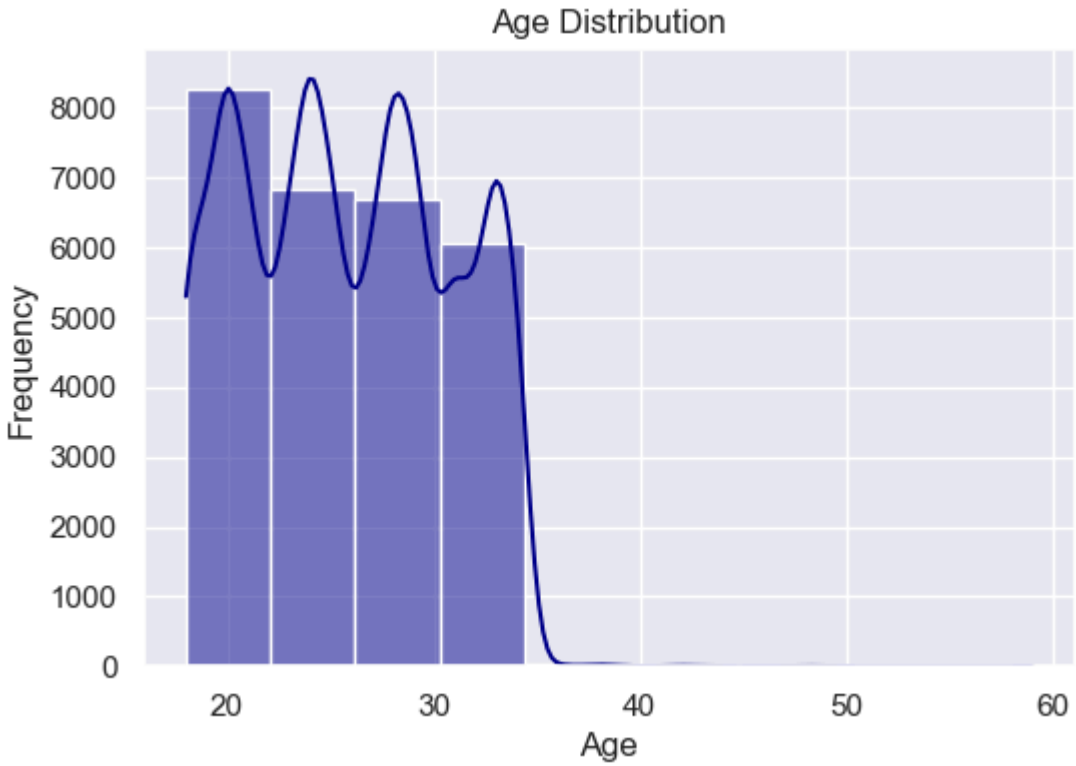
In [16]:
```
# Set style
sns.set_theme(style="darkgrid")

# Gender Distribution
plt.figure(figsize=(6, 4))
sns.countplot(x='Gender', data=df, palette='coolwarm')
plt.title("Gender Distribution")
plt.xlabel("Gender")
plt.ylabel("Count")
plt.show()
```

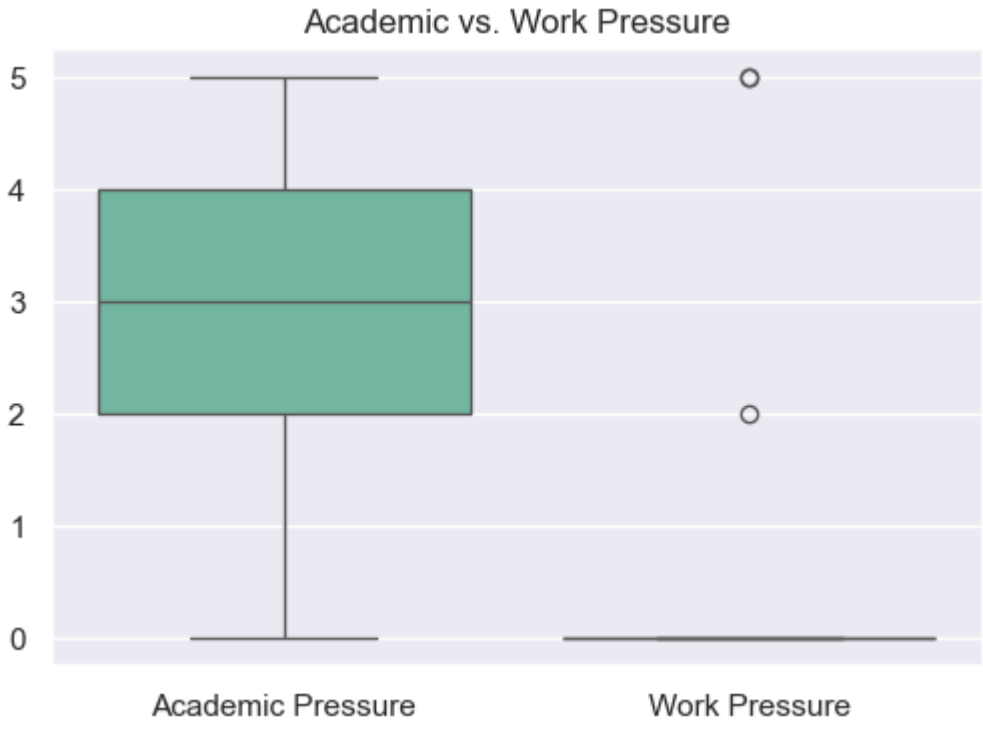


In [17]:
```
# Age Distribution
plt.figure(figsize=(6, 4))
sns.histplot(df['Age'], bins=10, kde=True, color='darkblue')
plt.title("Age Distribution")
```

```
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```

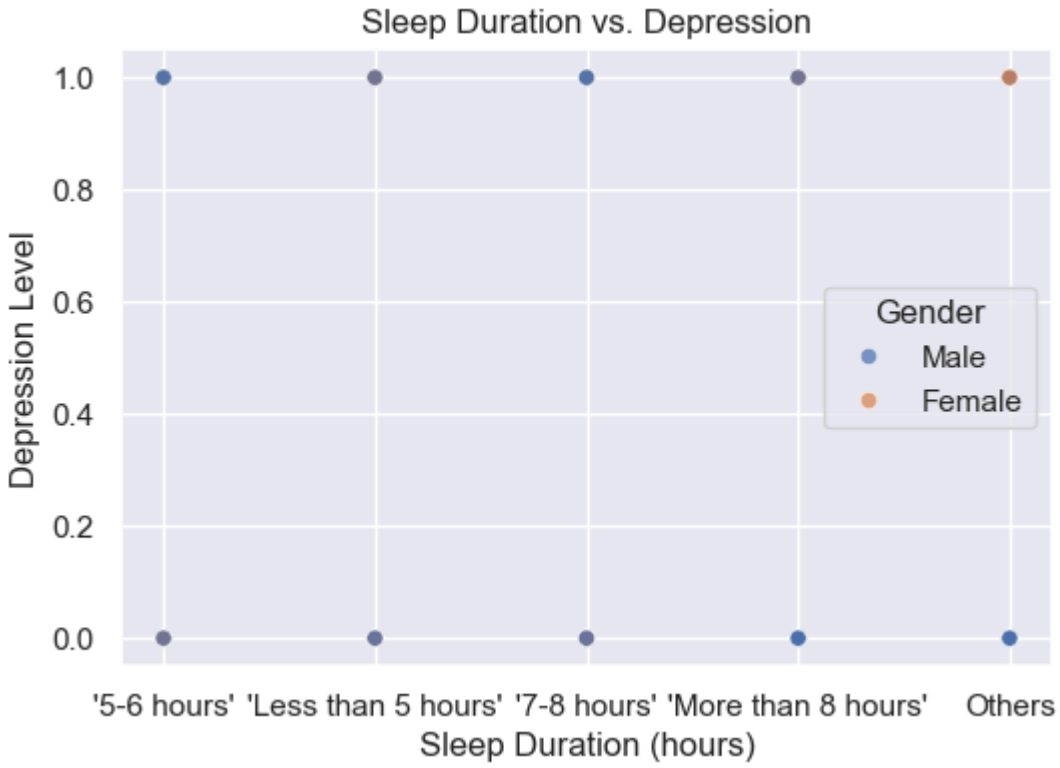


In [18]:
```
# Academic vs. Work Pressure (Box Plot)
plt.figure(figsize=(6, 4))
sns.boxplot(data=df[['Academic Pressure', 'Work Pressure']], palette='Set2')
plt.title("Academic vs. Work Pressure")
plt.show()
```

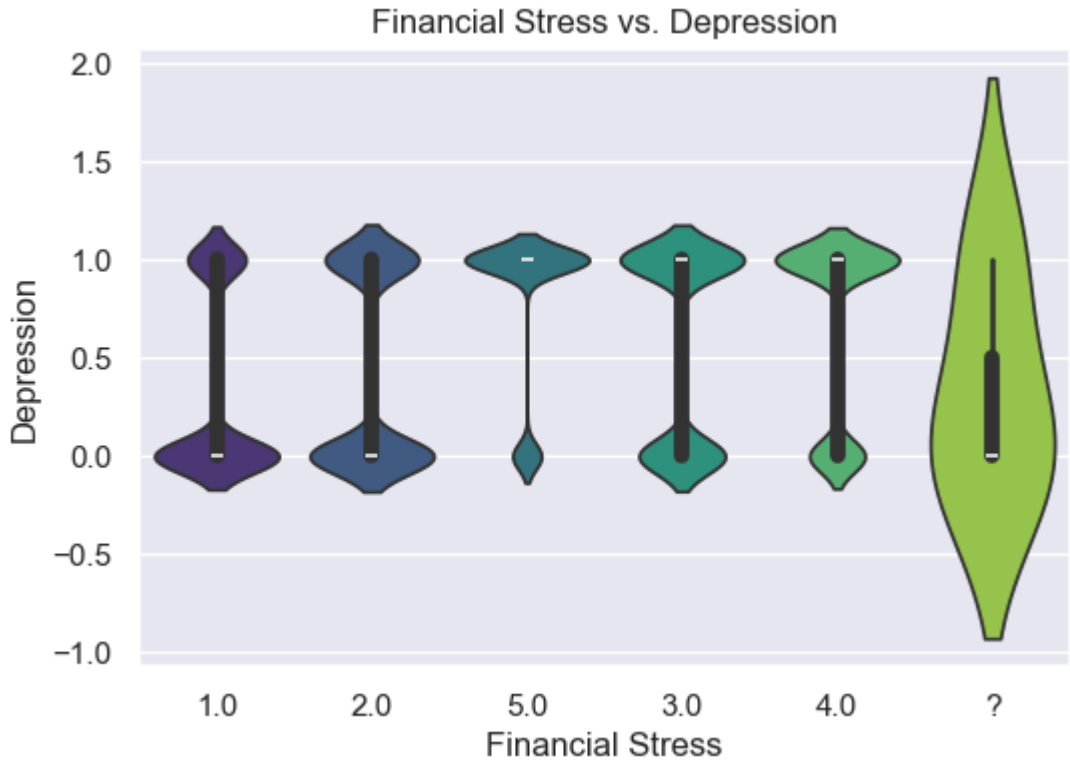


In [19]:
```
# Sleep Duration vs. Depression (Scatter Plot)
plt.figure(figsize=(6, 4))
sns.scatterplot(x='Sleep Duration', y='Depression', hue='Gender', data=df, alpha
```

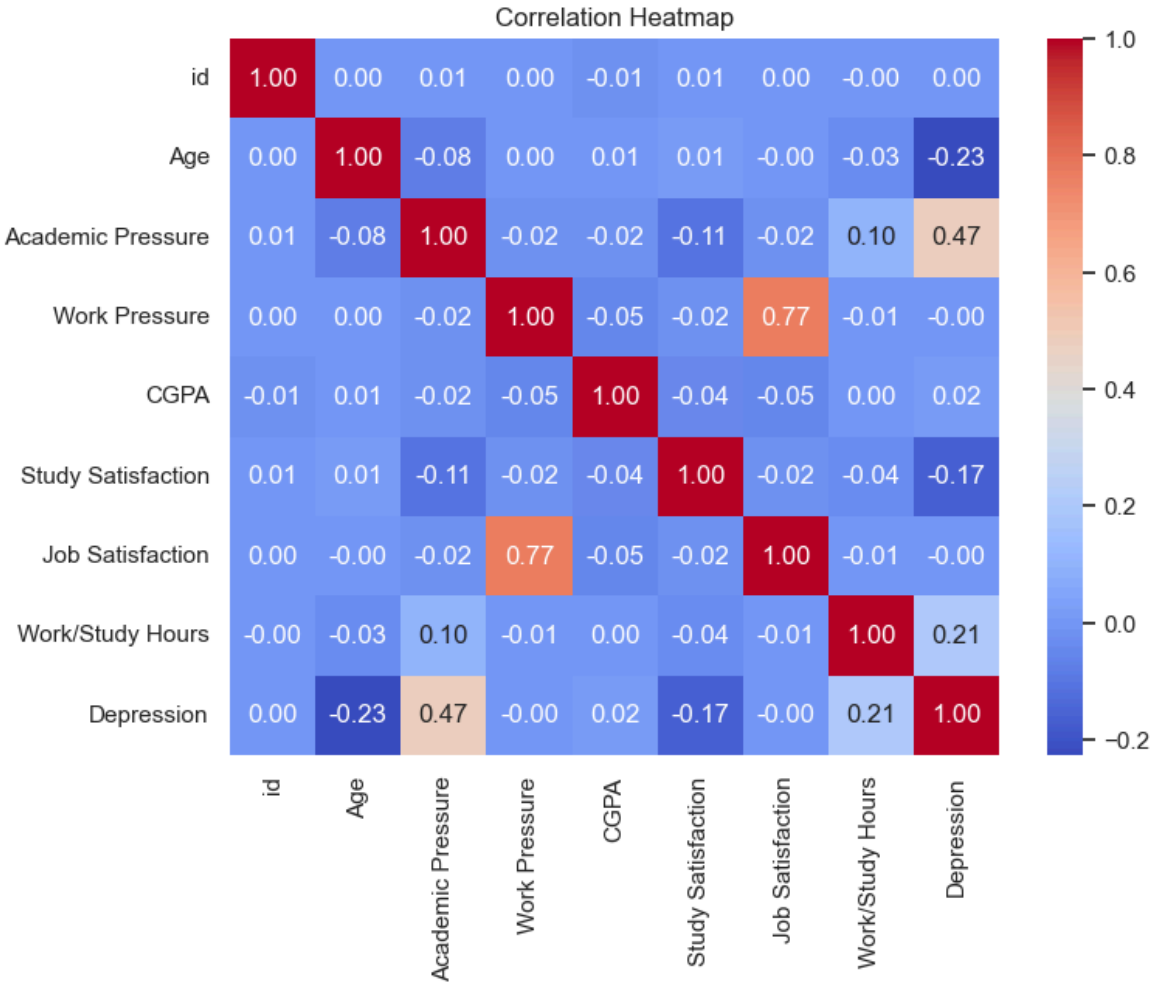
```
plt.title("Sleep Duration vs. Depression")
plt.xlabel("Sleep Duration (hours)")
plt.ylabel("Depression Level")
plt.show()
```



In [20]:
```
# Financial Stress vs. Depression (Violin Plot)
plt.figure(figsize=(6, 4))
sns.violinplot(x='Financial Stress', y='Depression', data=df, palette='viridis')
plt.title("Financial Stress vs. Depression")
plt.show()
```



In [21]:
```
numeric_df = df.select_dtypes(include=['number'])
plt.figure(figsize=(8, 6))
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Heatmap")
plt.show()
```



Predictive Modeling

In [23]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

In [24]:
```
# Drop 'id' column if it exists
df.drop(columns=['id'], inplace=True, errors='ignore')
```

In [25]:
```
# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns.tolist()
```

In [26]:
```
from sklearn.preprocessing import LabelEncoder

# Identify categorical columns
categorical_cols = ['Gender', 'City', 'Profession', 'Degree', 'Dietary Habits',
```

```
                    'Sleep Duration', 'Have you ever had suicidal thoughts ?',
                    'Family History of Mental Illness']
```

In [27]:
```
# Apply Label Encoding
le = LabelEncoder()
for col in categorical_cols:
    df[col] = df[col].astype(str).str.strip("'")  # Remove extra quotes
    df[col] = le.fit_transform(df[col])
```

In [28]:
```
# Check if all values are numeric
print(df.dtypes)
```

```
Gender                                   int32
Age                                    float64
City                                     int32
Profession                               int32
Academic Pressure                      float64
Work Pressure                          float64
CGPA                                   float64
Study Satisfaction                     float64
Job Satisfaction                       float64
Sleep Duration                           int32
Dietary Habits                           int32
Degree                                   int32
Have you ever had suicidal thoughts ?    int32
Work/Study Hours                       float64
Financial Stress                        object
Family History of Mental Illness         int32
Depression                               int64
dtype: object
```

In [29]:
```
# Handling missing values (filling with median for numerical columns)
df.fillna(df.median(numeric_only=True), inplace=True)
```

In [30]:
```
# Define features and target
X = df.drop(columns=['Depression'])
y = df['Depression']
```

In [31]:
```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [32]:
```
# Initialize models
models = {
    "Logistic Regression": LogisticRegression(),
    "Random Forest": RandomForestClassifier(),
    "SVM": SVC(),
    "KNN": KNeighborsClassifier(),
    "Gradient Boosting": GradientBoostingClassifier()
}
```

In [33]:
```
X_train.replace('?', np.nan, inplace=True)
X_test.replace('?', np.nan, inplace=True)
```

In [34]:
```
num_cols = X_train.select_dtypes(include=['number']).columns
cat_cols = X_train.select_dtypes(include=['object']).columns

print("Numerical Columns:", num_cols)
print("Categorical Columns:", cat_cols)
```

```
Numerical Columns: Index(['Gender', 'Age', 'City', 'Profession', 'Academic Pressu
re',
       'Work Pressure', 'CGPA', 'Study Satisfaction', 'Job Satisfaction',
       'Sleep Duration', 'Dietary Habits', 'Degree',
       'Have you ever had suicidal thoughts ?', 'Work/Study Hours',
       'Family History of Mental Illness'],
      dtype='object')
Categorical Columns: Index(['Financial Stress'], dtype='object')
```

In [35]:
```python
from sklearn.impute import SimpleImputer

# Impute numerical columns with median
num_imputer = SimpleImputer(strategy='median')
X_train[num_cols] = num_imputer.fit_transform(X_train[num_cols])
X_test[num_cols] = num_imputer.transform(X_test[num_cols])

# Impute categorical columns with mode
cat_imputer = SimpleImputer(strategy='most_frequent')
X_train[cat_cols] = cat_imputer.fit_transform(X_train[cat_cols])
X_test[cat_cols] = cat_imputer.transform(X_test[cat_cols])
```

In [36]:
```python
# Train and evaluate models
accuracy_results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred) * 100
    accuracy_results[name] = accuracy
```

In [37]:
```python
# Display accuracy results
for model, acc in accuracy_results.items():
    print(f"{model}: {acc:.2f}%")
```

```
Logistic Regression: 83.64%
Random Forest: 83.09%
SVM: 83.43%
KNN: 74.65%
Gradient Boosting: 83.75%
```

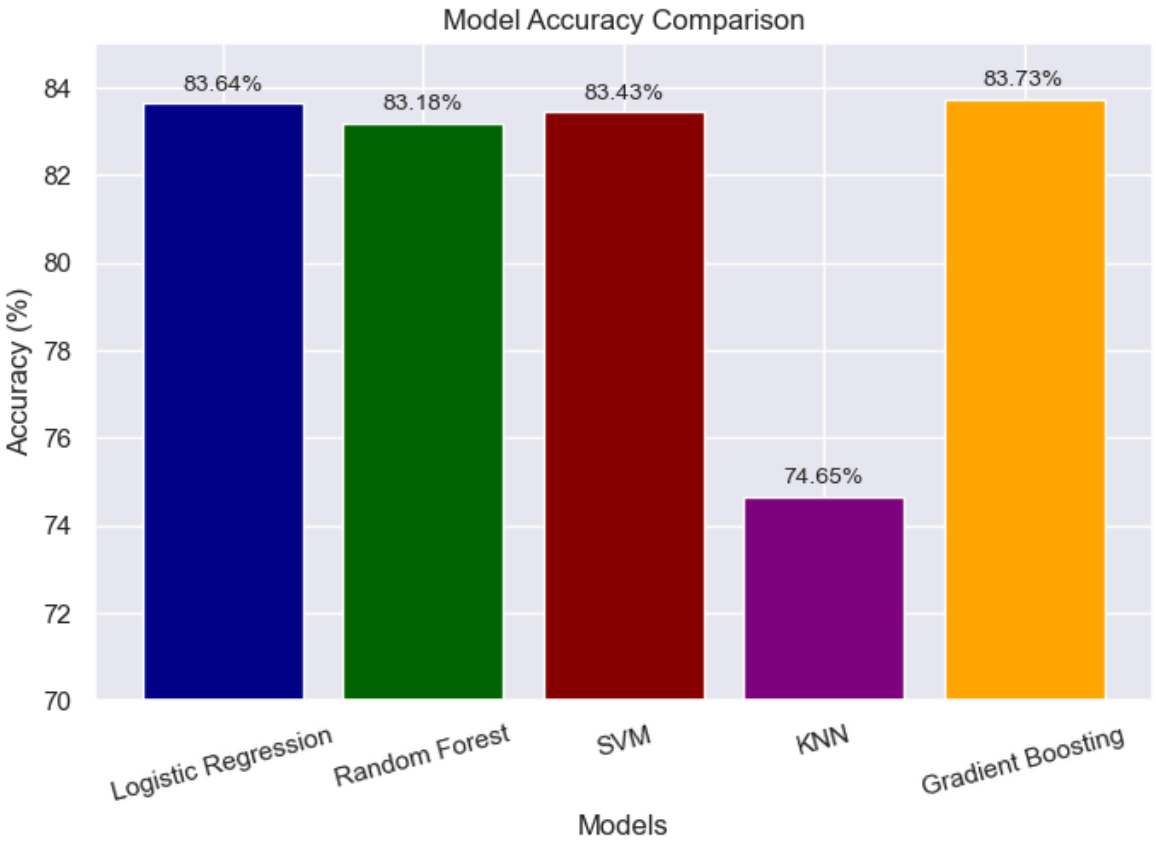
In [38]:
```
# Model names and their corresponding accuracy scores
models = ['Logistic Regression', 'Random Forest', 'SVM', 'KNN', 'Gradient Boosti
accuracy = [83.64, 83.18, 83.43, 74.65, 83.73]

# Creating the bar plot
plt.figure(figsize=(8, 5))
plt.bar(models, accuracy, color=['darkblue', 'darkgreen', 'darkred', 'purple', '

# Adding labels and title
plt.xlabel('Models')
plt.ylabel('Accuracy (%)')
plt.title('Model Accuracy Comparison')
plt.ylim(70, 85)  # Setting y-axis limits for better visibility

# Display values on top of bars
for i, v in enumerate(accuracy):
    plt.text(i, v + 0.3, f"{v:.2f}%", ha='center', fontsize=10)

# Show the plot
plt.xticks(rotation=15)  # Rotate x-axis labels for readability
plt.show()
```



Completed

In [ ]:
```

```