

Dataset

<https://www.kaggle.com/datasets/meharshanali/student-dropout-prediction-dataset>

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv('student_dropout_dataset_v3.csv')
```

```
df.head()
```

	Student_ID	Age	Gender	Family_Income	Internet_Access	Study_Hours_per_Day
0	1	22.1	Male	25000.0	Yes	3.36
1	2	20.7	Male	25000.0	Yes	4.30
2	3	22.4	Male	40183.0	Yes	4.40
3	4	24.4	Male	NaN	Yes	NaN
4	5	20.5	Female	25319.0	Yes	4.19

Data Sanitization

Removing the Missing Values

```
df.dtypes
```

```
Student_ID      int64
Age             float64
Gender          object
Family_Income   float64
Internet_Access object
Study_Hours_per_Day float64
Attendance_Rate float64
```

```

Assignment_Delay_Days    int64
Travel_Time_Minutes      float64
Part_Time_Job            object
Scholarship              object
Stress_Index             float64
GPA                      float64
Semester_GPA             float64
CGPA                     float64
Semester                 object
Department               object
Parental_Education       object
Dropout                  int64
dtype: object

```

```
df.columns
```

```

Index(['Student_ID', 'Age', 'Gender', 'Family_Income', 'Internet_Access',
      'Study_Hours_per_Day', 'Attendance_Rate', 'Assignment_Delay_Days',
      'Travel_Time_Minutes', 'Part_Time_Job', 'Scholarship', 'Stress_Index',
      'GPA', 'Semester_GPA', 'CGPA', 'Semester', 'Department',
      'Parental_Education', 'Dropout'],
      dtype='object')

```

```
df.isnull().sum()
```

```

Student_ID      0
Age             0
Gender          0
Family_Income   500
Internet_Access 0
Study_Hours_per_Day 500
Attendance_Rate 0
Assignment_Delay_Days 0
Travel_Time_Minutes 0
Part_Time_Job   0
Scholarship     0
Stress_Index    500
GPA             0
Semester_GPA    0
CGPA            0
Semester        0
Department      0
Parental_Education 511
Dropout         0
dtype: int64

```

```
missing_percentage = df.isnull().mean() * 100
## the threshold should usually is 30% but for the seak of the practice it is 5
threshold = 5
columns_to_drop = missing_percentage[missing_percentage > threshold].index
df = df.drop(columns=columns_to_drop)
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns}")
```

```
Columns dropped: ['Parental_Education']
Remaining columns: Index(['Student_ID', 'Age', 'Gender', 'Family_Income', 'Inter
    'Study_Hours_per_Day', 'Attendance_Rate', 'Assignment_Delay_Days',
    'Travel_Time_Minutes', 'Part_Time_Job', 'Scholarship', 'Stress_Index',
    'GPA', 'Semester_GPA', 'CGPA', 'Semester', 'Department', 'Dropout'],
    dtype='object')
```

✓ Pre-processing on dataset

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
    df[column] = label_encoder.fit_transform(df[column])
df.head()
```

	Student_ID	Age	Gender	Family_Income	Internet_Access	Study_Hours_per_Day
0	1	22.1	1	25000.0	1	3.36
1	2	20.7	1	25000.0	1	4.30
2	3	22.4	1	40183.0	1	4.40
3	4	24.4	1	NaN	1	NaN
4	5	20.5	0	25319.0	1	4.19

✓ Filling missing values based on feature type

```
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean()
    df[column] = df[column].fillna(mean_value)
```

```
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0]
```

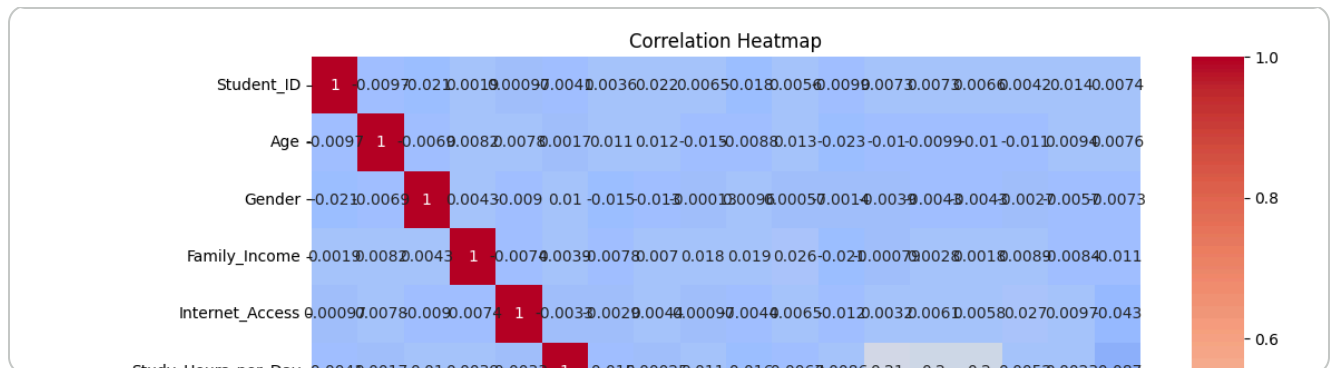
```
df[column] = df[column].fillna(mode_value)
```

```
df.isnull().sum()
```

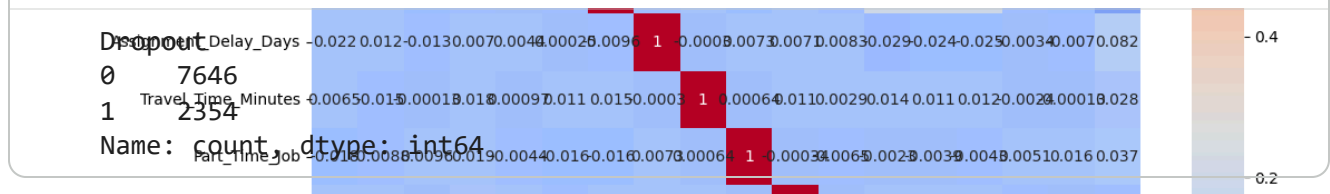
```
Student_ID      0
Age             0
Gender          0
Family_Income   0
Internet_Access 0
Study_Hours_per_Day 0
Attendance_Rate 0
Assignment_Delay_Days 0
Travel_Time_Minutes 0
Part_Time_Job   0
Scholarship     0
Stress_Index    0
GPA             0
Semester_GPA    0
CGPA            0
Semester        0
Department      0
Dropout         0
dtype: int64
```

✓ Data Visualization

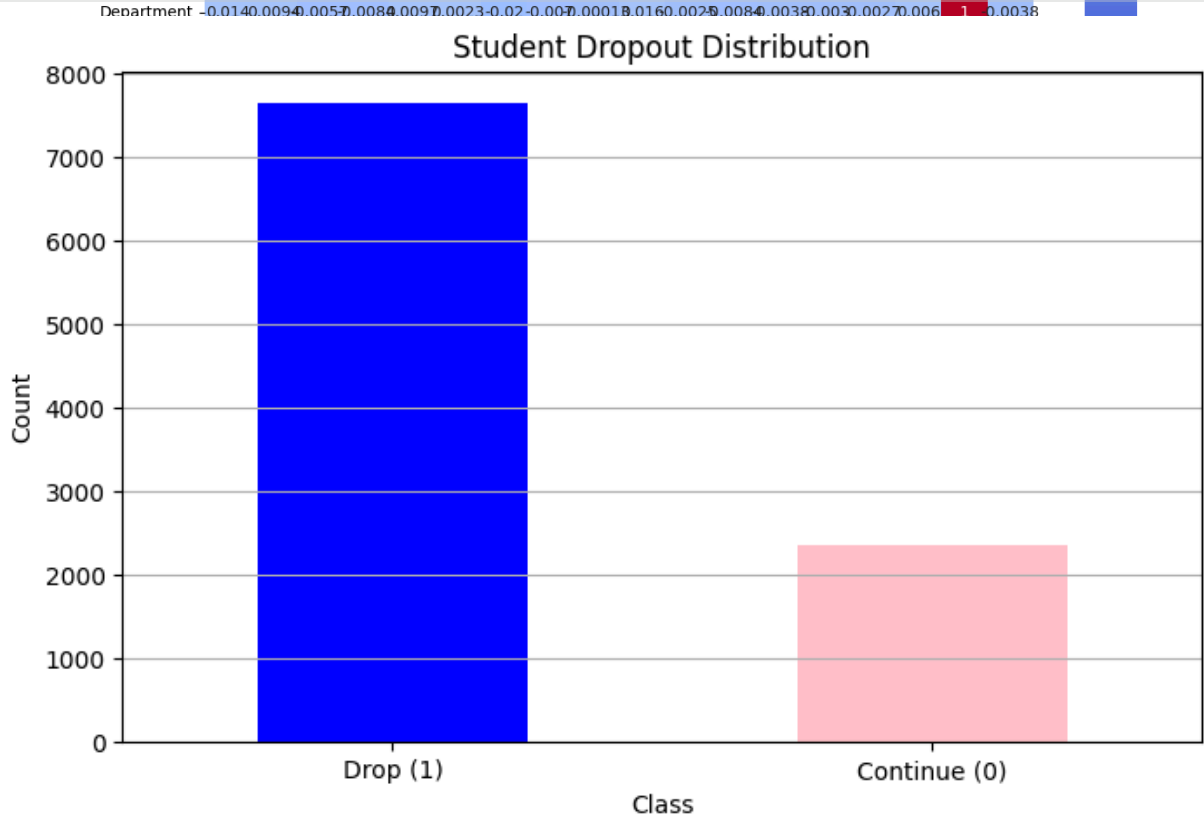
```
plt.figure(figsize=(12, 12))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

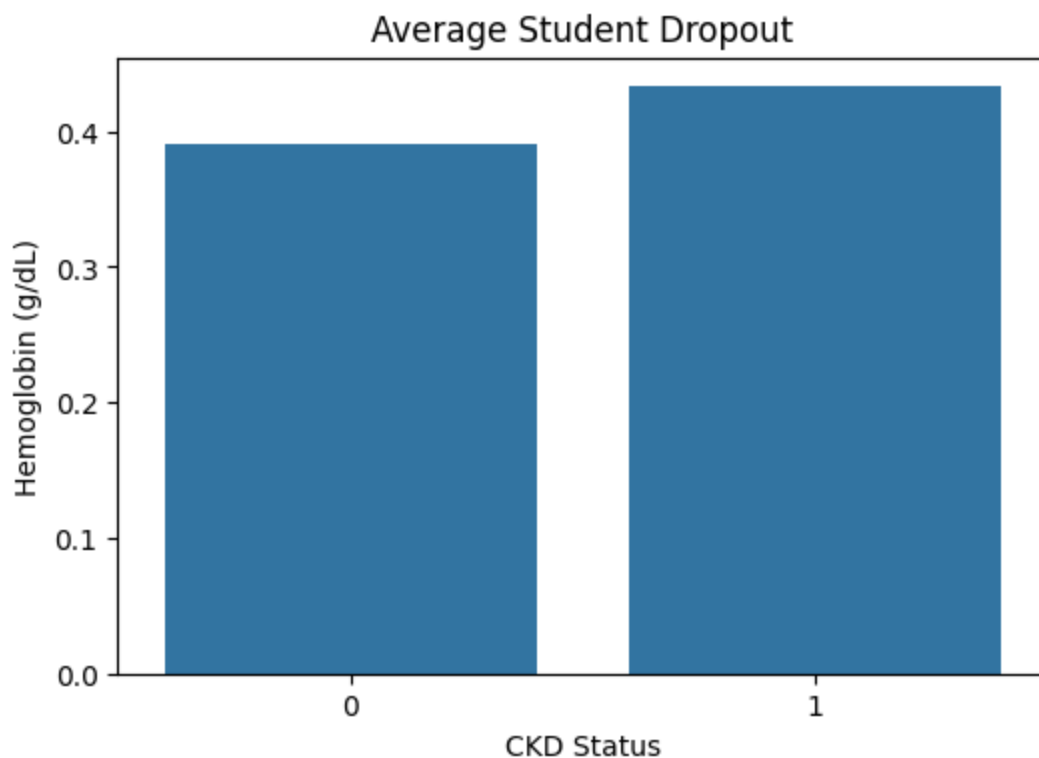
```
df['Dropout'].value_counts()
```



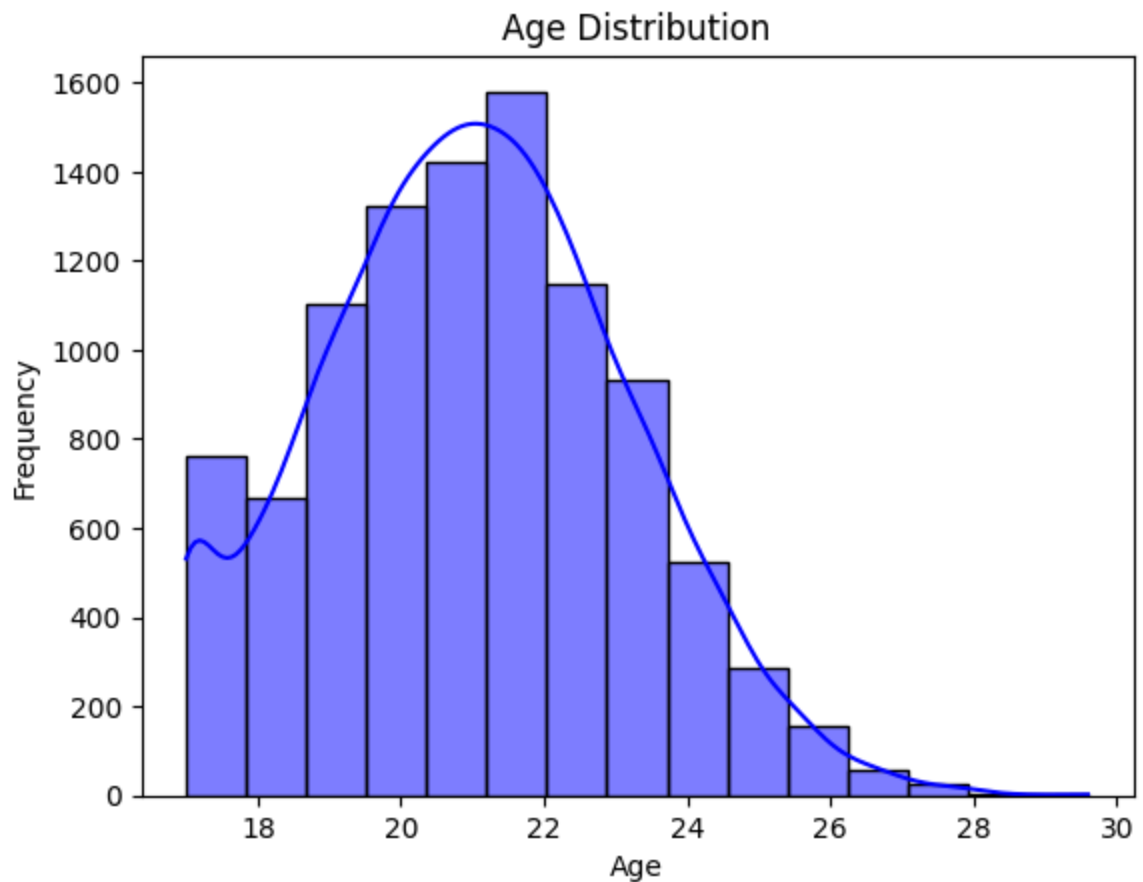
```
plt.figure(figsize=(8, 5))
df['Dropout'].value_counts().plot(kind='bar', color=['blue', 'pink'])
plt.xticks([0, 1], ['Drop (1)', 'Continue (0)'], rotation=0)
plt.title('Student Dropout Distribution')
plt.xlabel('Class')
plt.ylabel('Count')
plt.grid(axis='y')
plt.show()
```



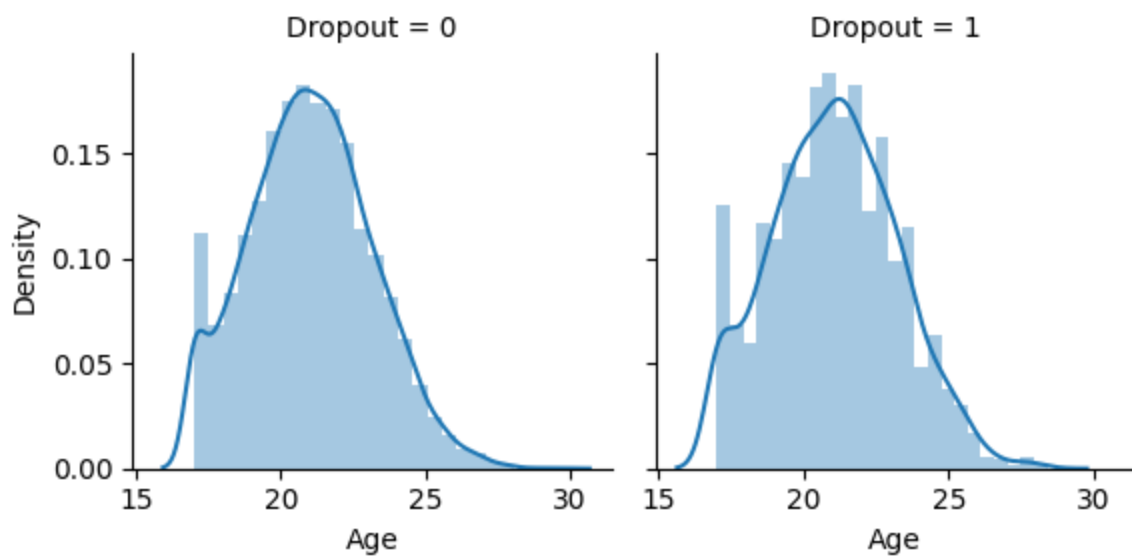
```
import seaborn as sns
plt.figure(figsize=(6,4))
sns.barplot(x='Dropout', y='Part_Time_Job', data=df, estimator='mean', ci=None)
plt.title("Average Student Dropout")
plt.xlabel("CKD Status")
plt.ylabel("Hemoglobin (g/dL)")
plt.show()
```



```
cat_cols = [col for col in df.columns if df[col].dtype=='object']
num_cols = [col for col in df.columns if df[col].dtype!='object']
for col in cat_cols:
    print(f'{col} has {df[col].unique()} values')
sns.histplot(df['Age'], bins=15, kde=True, color='blue')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



```
g = sns.FacetGrid(df,col="Dropout")
g.map(sns.distplot,"Age", bins=25)
plt.show()
```



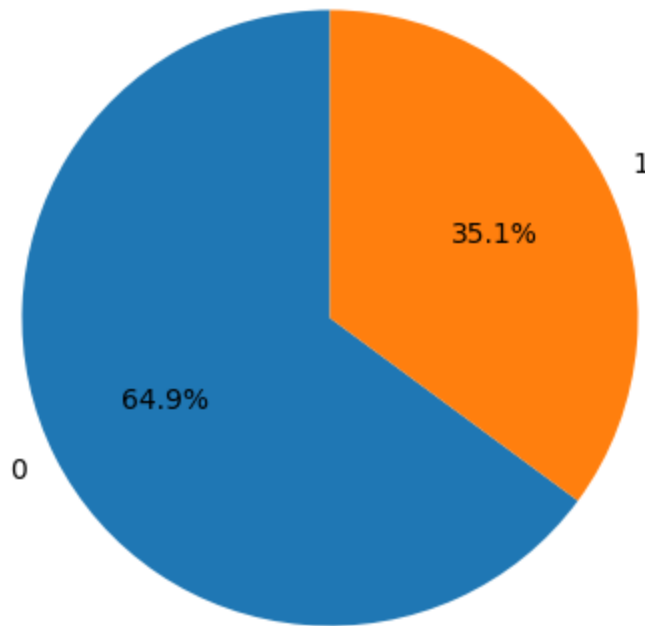
```
df.columns
```

```
Index(['Student_ID', 'Age', 'Gender', 'Family_Income', 'Internet_Access',
      'Study_Hours_per_Day', 'Attendance_Rate', 'Assignment_Delay_Days',
```

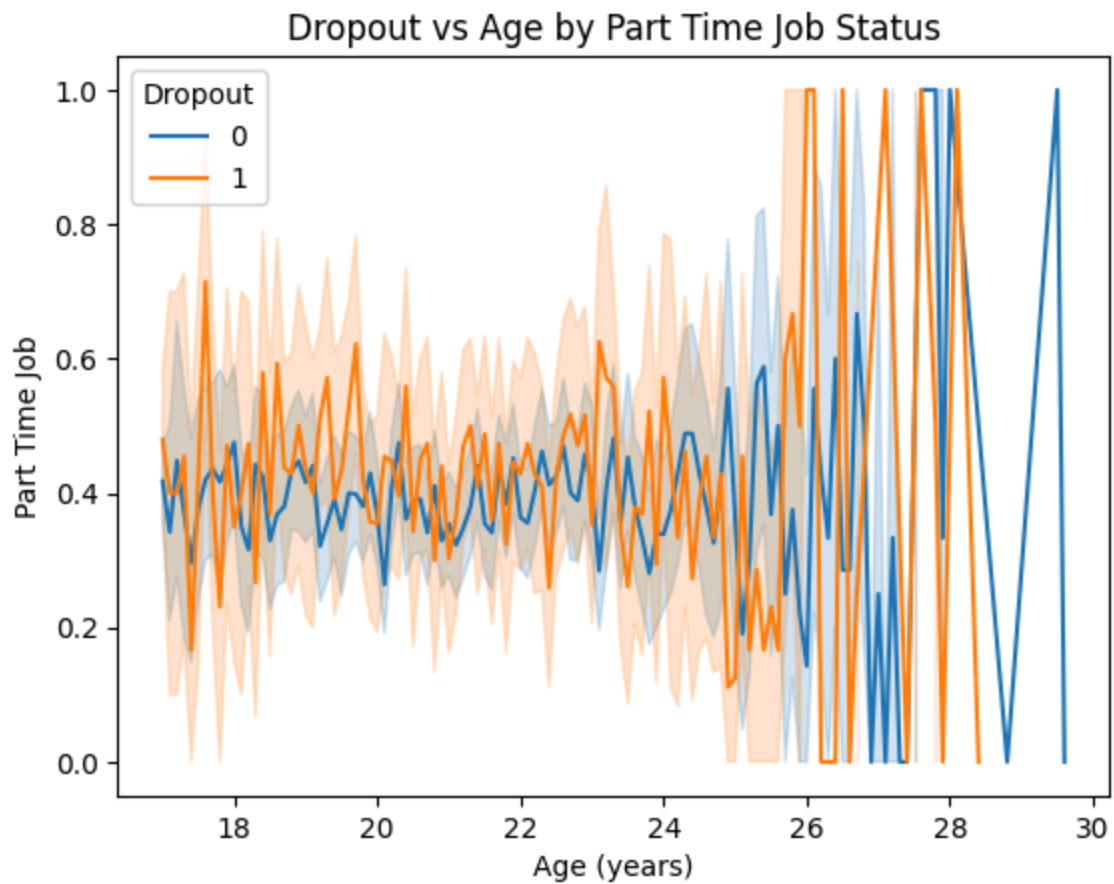
```
'Travel_Time_Minutes', 'Part_Time_Job', 'Scholarship', 'Stress_Index',  
'GPA', 'Semester_GPA', 'CGPA', 'Semester', 'Department', 'Dropout'],  
dtype='object')
```

```
appet_counts = df['Scholarship'].value_counts()  
plt.figure(figsize=(5,5))  
plt.pie(appet_counts, labels=appet_counts.index, autopct='%1.1f%%', startangle=  
plt.title("Student Scholarship Distribution")  
plt.show()
```

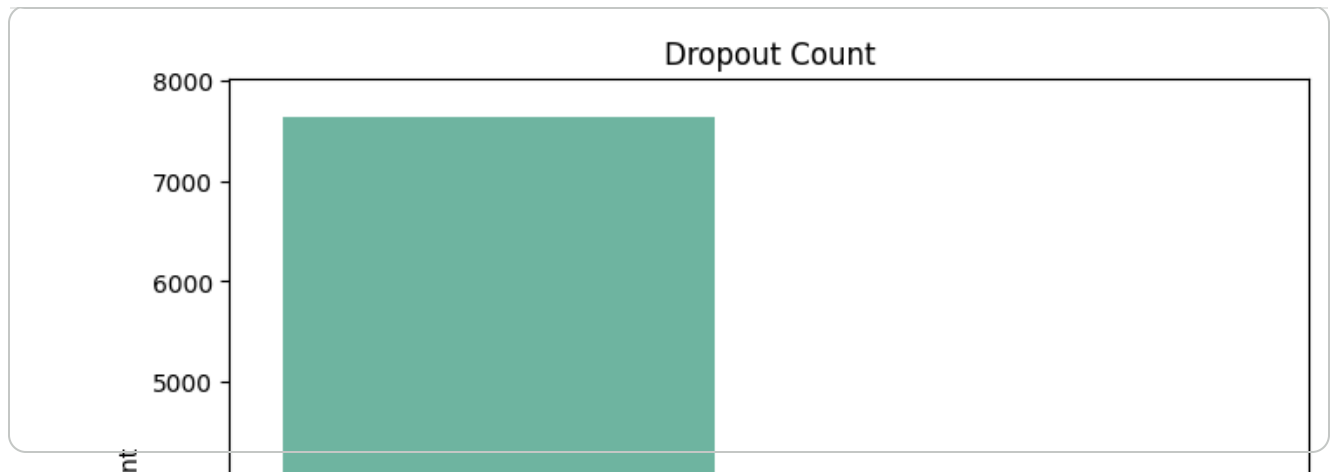
Student Scholarship Distribution



```
sns.lineplot(x='Age', y='Part_Time_Job', hue='Dropout', data=df)  
plt.title("Dropout vs Age by Part Time Job Status")  
plt.xlabel("Age (years)")  
plt.ylabel("Part Time Job")  
plt.show()
```



```
plt.figure(figsize=(8, 6))
sns.countplot(x='Dropout', data=df, palette='Set2')
plt.title('Dropout Count')
plt.xlabel('Dropout')
plt.ylabel('Count')
plt.show()
```



```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Dropout', y='Age', data=df, palette='Set1')
plt.title('Age Levels by Dropout')
plt.xlabel('Dropout')
plt.ylabel('Age')
plt.show()
```

