

Exploratory Data Analysis (EDA) on df Marks Dataset

1. Introduction

This project performs an Exploratory Data Analysis (EDA) on a *college Marks Dataset*.

Objectives:

- Understand the structure of the dataset (rows, columns, data types)
- Analyze marks at different levels:
 - SSC (10th standard)
 - HSC (12th standard)
 - college marks
- Study attendance patterns
- Explore how **Class** (Arts, Commerce, Science, Math) and **Grade** relate to marks
- Visualize the data using:
 - Matplotlib
 - Seaborn
- Use:
 - **NumPy** for numerical operations
 - **Pandas** for data handling
 - **Matplotlib + Seaborn** for plots

2. Importing Libraries (Code cell)

```
# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# For better looking plots
sns.set(style="whitegrid")

# Make plots larger
plt.rcParams["figure.figsize"] = (8, 5)

# Import file in google colab
from google.colab import files
uploaded = files.upload()
```

Choose Files | No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
# Mount Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True)
```

3. Loading the Dataset.

```
# 3. Loading the Dataset
df = pd.read_csv("College_Marks_Dataset.csv")
```

```
# Display the first few rows
df.head()
```

	Student_ID	Name	Class	SSC_Marks	HSC_Marks	College_Marks	Attendance_Percentage	Grade
0	S1000	Student_0	Commerce	535	452	692	84.71	C
1	S1001	Student_1	Commerce	494	535	551	81.99	D
2	S1002	Student_2	Science	542	460	634	92.06	B
3	S1003	Student_3	Science	441	483	686	79.27	D
4	S1004	Student_4	Arts	427	544	569	91.99	A+

4. Basic Dataset Overview.

```
# Basic Info
print("Number of rows and columns:", df.shape)
print("\nColumn names:")
print(df.columns.tolist())
print("\nDataset Info:")
df.info()

Number of rows and columns: (1000, 8)

Column names:
['Student_ID', 'Name', 'Class', 'SSC_Marks', 'HSC_Marks', 'College_Marks', 'Attendance_Percentage', 'Grade']

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Student_ID      1000 non-null   object 
 1   Name             1000 non-null   object 
 2   Class            1000 non-null   object 
 3   SSC_Marks       1000 non-null   int64  
 4   HSC_Marks       1000 non-null   int64  
 5   College_Marks   1000 non-null   int64  
 6   Attendance_Percentage  1000 non-null float64
 7   Grade            1000 non-null   object 
dtypes: float64(1), int64(3), object(4)
memory usage: 62.6+ KB
```

4.2 Descriptive Statistics

	SSC_Marks	HSC_Marks	College_Marks	Attendance_Percentage
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	476.197000	524.016000	603.05800	79.953460
std	44.340617	43.946775	58.34812	11.753637
min	400.000000	450.000000	500.00000	60.030000
25%	437.000000	484.750000	552.00000	69.572500
50%	476.000000	523.500000	602.00000	80.570000
75%	516.000000	564.000000	655.25000	89.955000
max	550.000000	600.000000	700.00000	99.950000

```
# Check max Marks and minimum marks
print(df.max())
df.min()
```

```
Student_ID      S1999
Name           Student_999
Class          Science
SSC_Marks       550
HSC_Marks       600
College_Marks    700
Attendance_Percentage 99.95
Grade            D
dtype: object
```

0

Student_ID	S1000
Name	Student_0
Class	Arts
SSC_Marks	400
HSC_Marks	450
College_Marks	500
Attendance_Percentage	60.03
Grade	A

dtype: object

5. Checking Missing Values

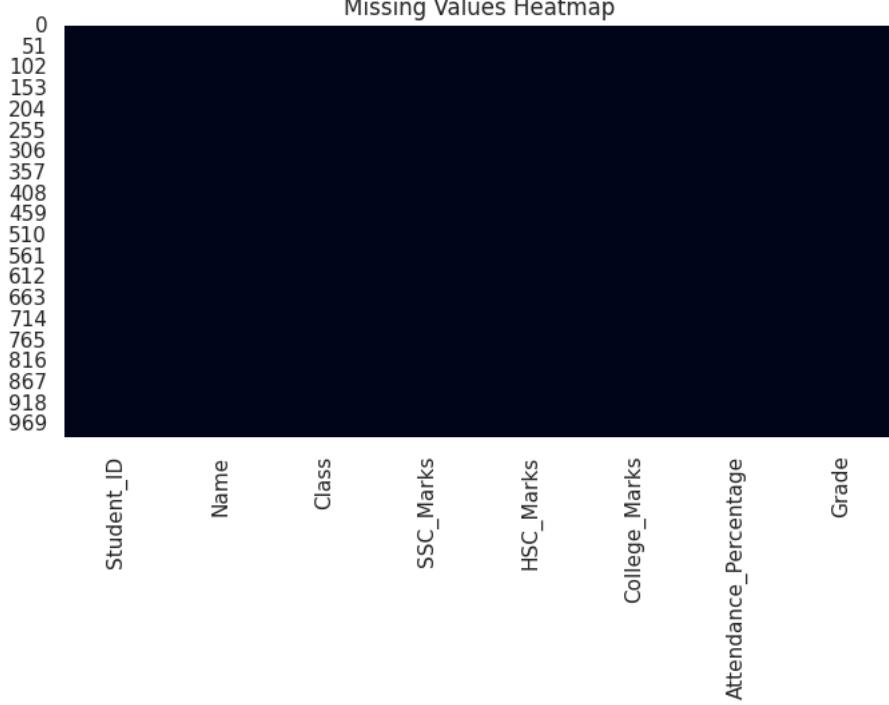
```
# 5. Missing Values Check
df.isnull().sum()
```

	0
Student_ID	0
Name	0
Class	0
SSC_Marks	0
HSC_Marks	0
College_Marks	0
Attendance_Percentage	0
Grade	0

dtype: int64

show a heatmap just to demonstrate the technique:

```
# 5.1 Missing Value Heatmap
plt.figure(figsize=(8, 4))
sns.heatmap(df.isnull(), cbar=False)
plt.title("Missing Values Heatmap")
plt.show()
```



6. Using NumPy for Basic Statistical Analysis

```
# 6. Using NumPy for numerical operations
arr = df[['SSC_Marks', 'HSC_Marks', 'College_Marks']].values
num1 = np.mean(arr, axis=0)
num2 = np.median(arr, axis=0)
num3 = np.std(arr, axis=0)

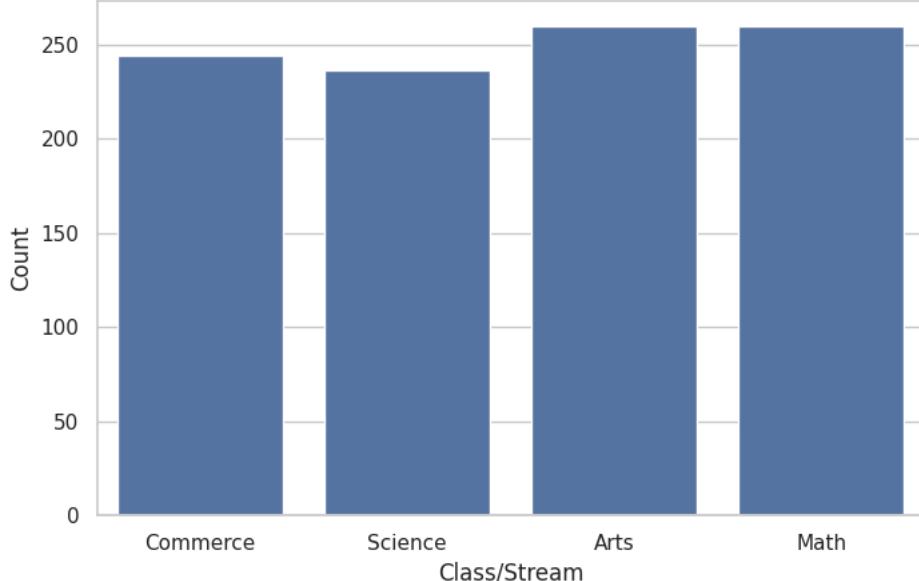
print("Mean of [SSC, HSC, College]:", num1)
print("Median of [SSC, HSC, College]:", num2)
print("Standard Deviation of [SSC, HSC, College]:", num3)
```

```
Mean of [SSC, HSC, College]: [476.197 524.016 603.058]
Median of [SSC, HSC, College]: [476. 523.5 602. ]
Standard Deviation of [SSC, HSC, College]: [44.31844076 43.92479646 58.31893891]
```

7. Univariate Analysis

```
#plot
sns.countplot(x='Class', data=df)
plt.title("Number of Students in Each Class/Stream")
plt.xlabel("Class/Stream")
plt.ylabel("Count")
plt.show()
```

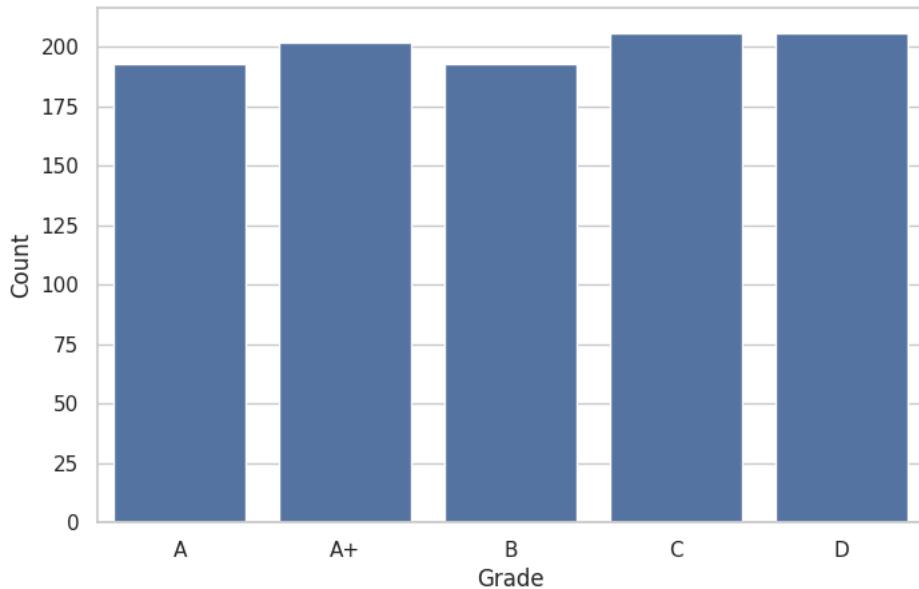
Number of Students in Each Class/Stream



7.2 Grade Distribution

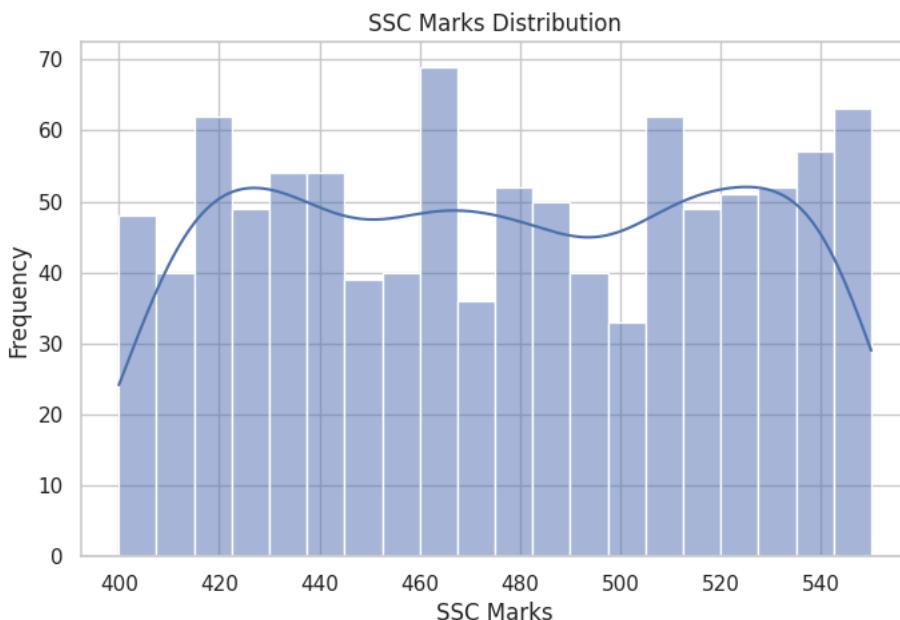
```
#plot
sns.countplot(x='Grade', data=df,
               order=sorted(df['Grade'].unique()))
plt.title("Grade Distribution")
plt.xlabel("Grade")
plt.ylabel("Count")
plt.show()
```

Grade Distribution



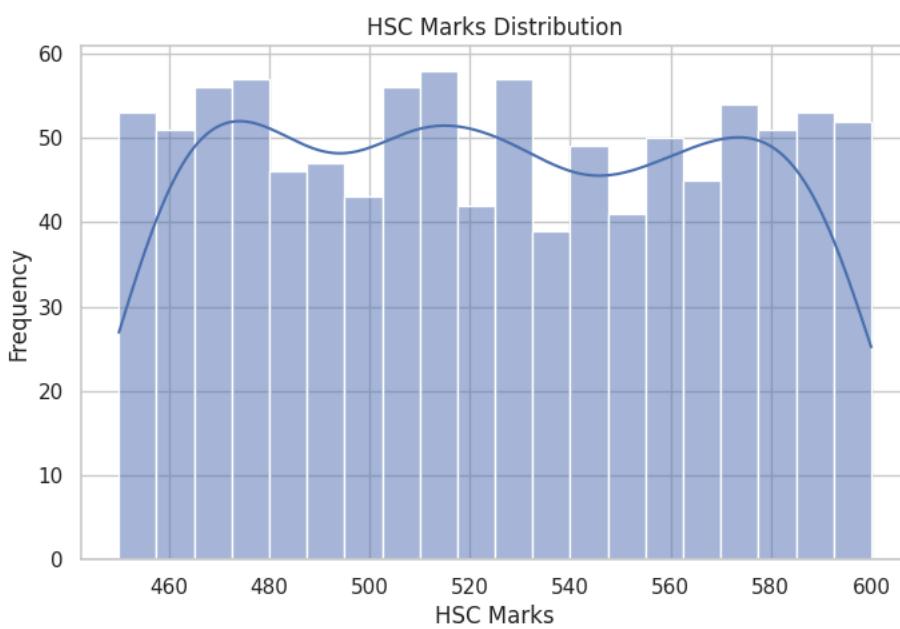
7.3 Distribution of SSC Marks

```
#plot
sns.histplot(df['SSC_Marks'], bins=20, kde=True)
plt.title("SSC Marks Distribution")
plt.xlabel("SSC Marks")
plt.ylabel("Frequency")
plt.show()
```



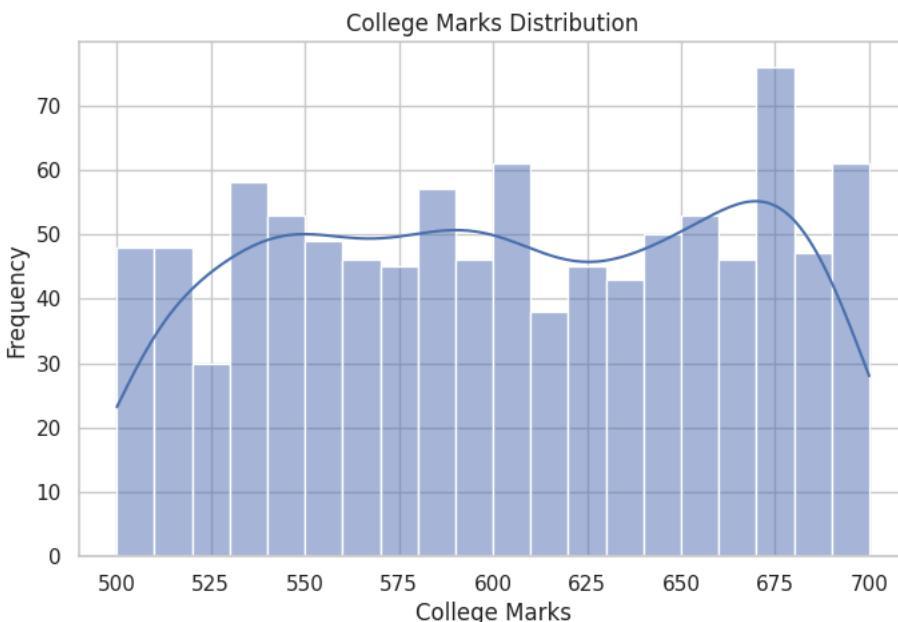
7.4 Distribution of HSC Marks

```
sns.histplot(df['HSC_Marks'], bins=20, kde=True)
plt.title("HSC Marks Distribution")
plt.xlabel("HSC Marks")
plt.ylabel("Frequency")
plt.show()
```



7.5 Distribution of df Marks

```
sns.histplot(df['College_Marks'], bins=20, kde=True)
plt.title("College Marks Distribution")
plt.xlabel("College Marks")
plt.ylabel("Frequency")
plt.show()
```

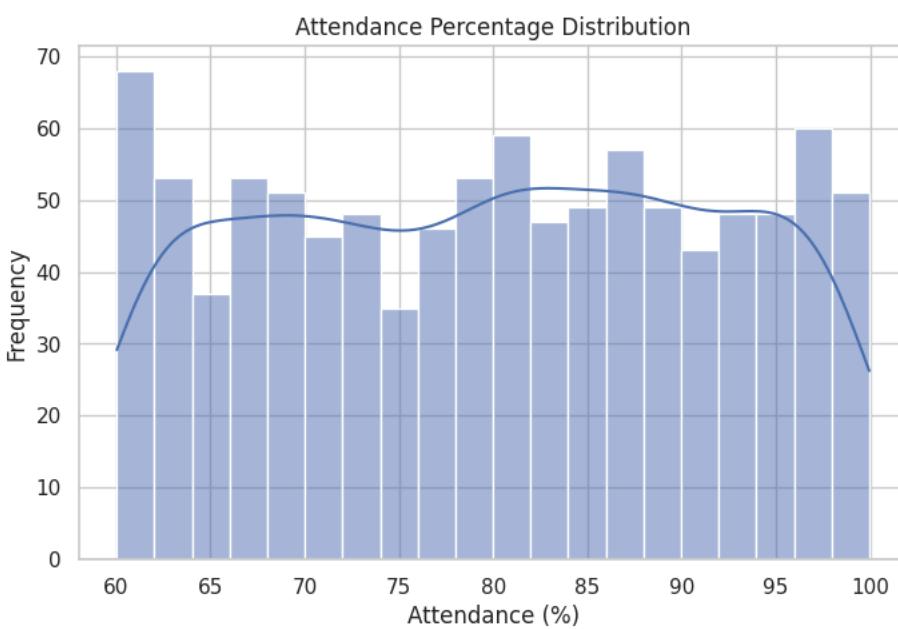


7.6 Attendance Distribution

```

sns.histplot(df['Attendance_Percentage'], bins=20, kde=True)
plt.title("Attendance Percentage Distribution")
plt.xlabel("Attendance (%)")
plt.ylabel("Frequency")
plt.show()

```



✓ Univariate Analysis – Summary:

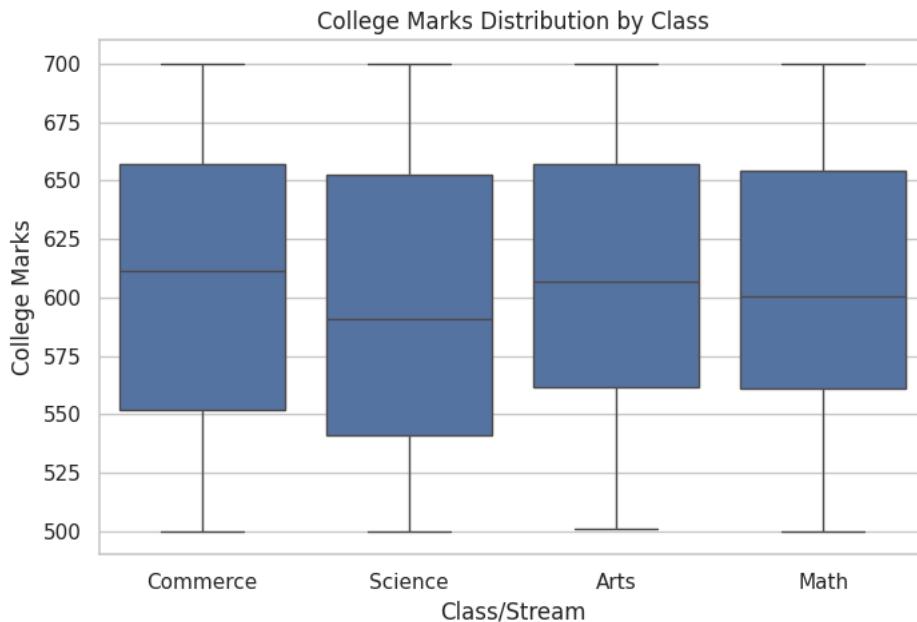
- Each **Class** (Arts, Science, Commerce, Math) has a similar number of students.
- Grades are fairly balanced across A+, A, B, C, and D.
- Marks distributions (SSC, HSC, df) appear concentrated in mid to high ranges.
- Attendance is mostly between 70% and 90%, with some students having very high attendance.

8. Bivariate Analysis

```

# College Marks by Class
sns.boxplot(x='Class', y='College_Marks', data=df)
plt.title("College Marks Distribution by Class")
plt.xlabel("Class/Stream")
plt.ylabel("College Marks")
plt.show()

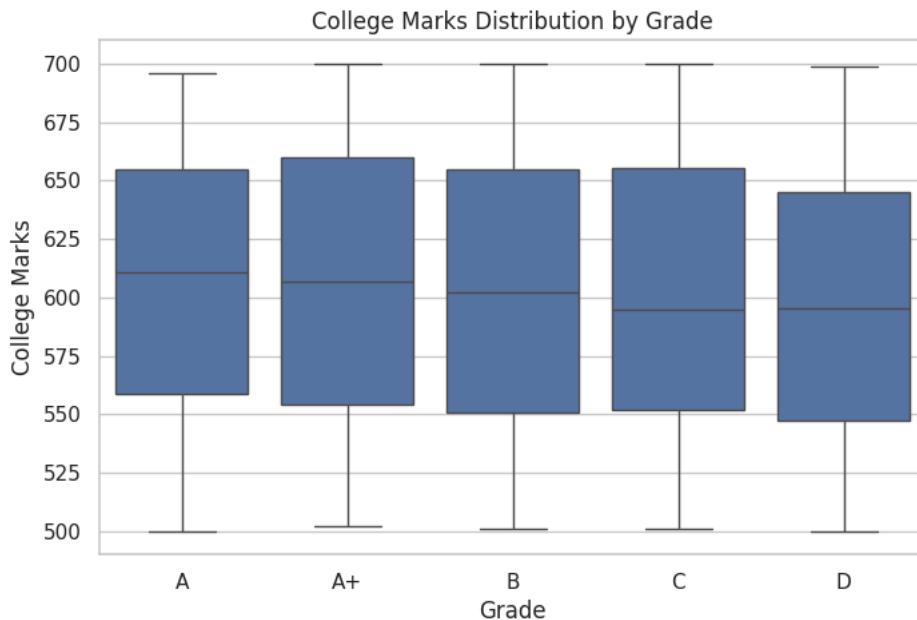
```



8.2 Grade vs df Marks

```

sns.boxplot(x='Grade', y='College_Marks', data=df,
            order=sorted(df['Grade'].unique()))
plt.title("College Marks Distribution by Grade")
plt.xlabel("Grade")
plt.ylabel("College Marks")
plt.show()
    
```

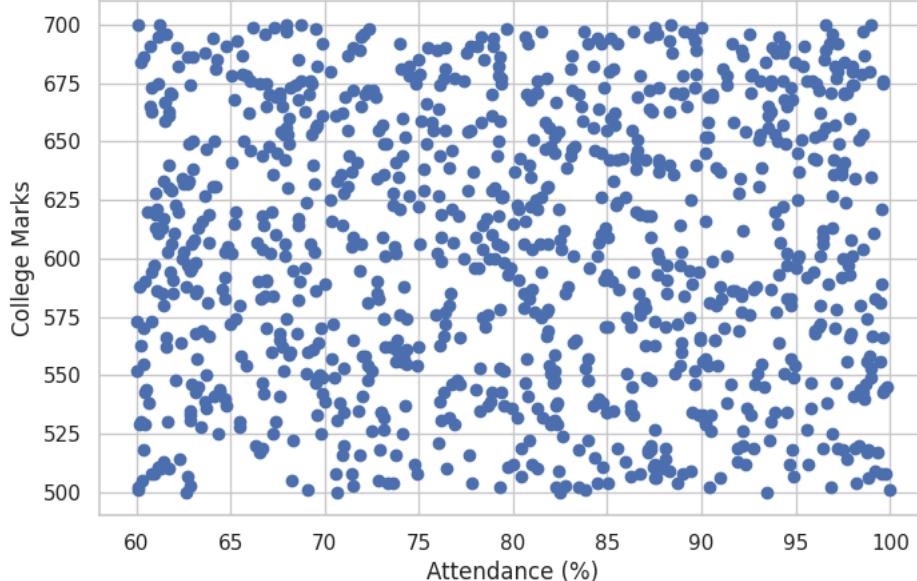


8.3 Attendance vs df Marks (Scatter Plot)

```

plt.scatter(df['Attendance_Percentage'], df['College_Marks'])
plt.title("Attendance vs College Marks")
plt.xlabel("Attendance (%)")
plt.ylabel("College Marks")
plt.show()
    
```

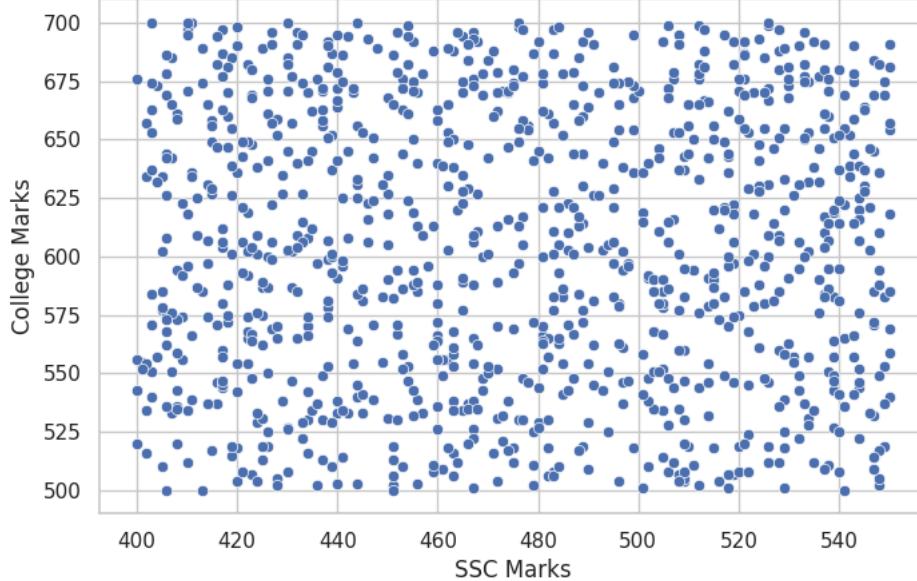
Attendance vs College Marks



8.4 SSC vs df Marks

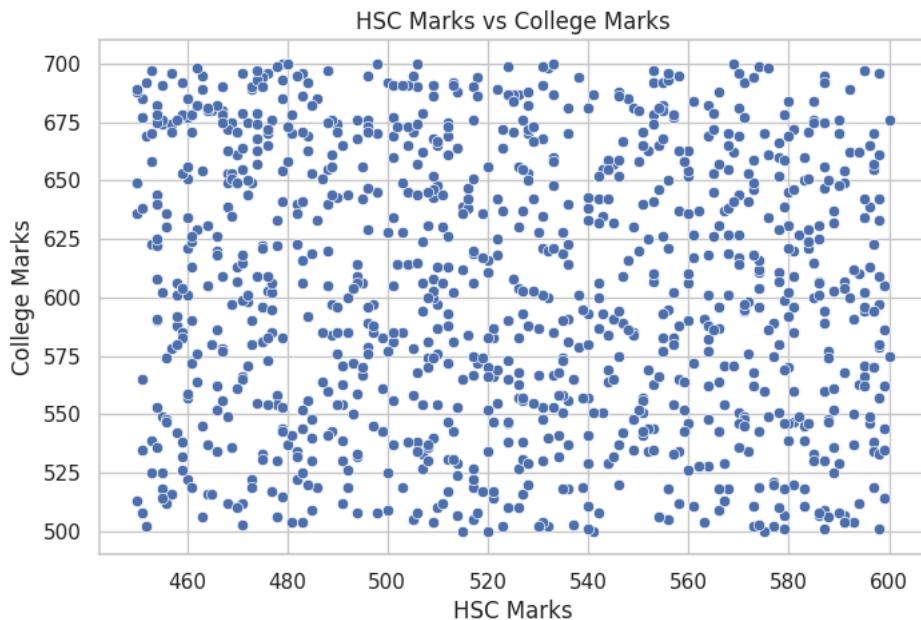
```
sns.scatterplot(x='SSC_Marks', y='College_Marks', data=df)
plt.title("SSC Marks vs College Marks")
plt.xlabel("SSC Marks")
plt.ylabel("College Marks")
plt.show()
```

SSC Marks vs College Marks



8.5 HSC vs df Marks

```
sns.scatterplot(x='HSC_Marks', y='College_Marks', data=df)
plt.title("HSC Marks vs College Marks")
plt.xlabel("HSC Marks")
plt.ylabel("College Marks")
plt.show()
```



✓ Bivariate Analysis – Summary:

- **Class vs df Marks:**
 - Average df marks are slightly different across Arts, Science, Commerce, and Math, but not extremely far apart.
- **Grade vs df Marks:**
 - Higher grades generally correspond to somewhat higher df marks, but the difference between grades is not extremely large.
- **Attendance vs df Marks:**
 - There is no very strong visible linear relationship, but extremely low attendance is rare.
- **SSC/HSC vs df Marks:**
 - SSC and HSC marks do not show a very strong linear relationship with df marks in this dataset.

9. Group-wise Analysis Using Pandas

```
# Average Marks by Class
class_group = df.groupby('Class')[['SSC_Marks', 'HSC_Marks', 'College_Marks', 'Attendance_Percentage']].mean()
class_group
```

	SSC_Marks	HSC_Marks	College_Marks	Attendance_Percentage
Class				
Arts	472.769231	523.080769	607.880769	79.750500
Commerce	478.381148	522.815574	606.737705	80.728975
Math	477.415385	526.138462	601.619231	79.075154
Science	476.372881	523.949153	595.525424	80.342881

9.2 Average Marks by Grade

```
grade_group = df.groupby('Grade')[['SSC_Marks', 'HSC_Marks', 'College_Marks', 'Attendance_Percentage']].mean()
grade_group
```

	SSC_Marks	HSC_Marks	College_Marks	Attendance_Percentage
Grade				
A	479.932642	522.756477	608.668394	79.625181
A+	470.099010	525.653465	605.881188	79.811287
B	475.357513	525.720207	602.414508	79.892850
C	484.752427	524.519417	601.645631	79.939320

▼ Observations:

- Grouping by **Class** helps us see which stream performs slightly better on average.
- Grouping by **Grade** shows how marks and attendance differ across grade categories.
- This is a powerful feature of Pandas for summarizing data.

10. Correlation Analysis

```
# Correlation Matrix for Numerical Columns
corr = df[['SSC_Marks', 'HSC_Marks', 'College_Marks', 'Attendance_Percentage']].corr()
corr
```

	SSC_Marks	HSC_Marks	College_Marks	Attendance_Percentage
SSC_Marks	1.000000	-0.003043	0.005014	0.013463
HSC_Marks	-0.003043	1.000000	-0.082895	-0.062323
College_Marks	0.005014	-0.082895	1.000000	0.002088
Attendance_Percentage	0.013463	-0.062323	0.002088	1.000000

```
# Heatmap
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

