Module_3_EDA(Exploratory Data Analysis): Understanding the Data

In this module we will perform EDA on student performance dataset ¶

you can download dataset here https://github.com/kundetivamsi2001/Datasets_AI-ML/blob/main/student_performance.csv (https://github.com/kundetivamsi2001/Datasets_AI-ML/blob/main/student_performance.csv)

In [76]:
```
#step 1: Load the libraries dataset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('student_performance.csv')
df.head()
```

Out[76]:

	Student_ID	Age	Gender	Study_Hours	Attendance(%)	Test_Score	Grade
0	S0524	21	Female	2.7	74.0	37.0	F
1	S0603	19	Male	1.5	93.1	26.0	F
2	S0527	20	Female	6.8	97.2	88.0	A
3	S0032	18	Female	6.2	93.5	55.0	C
4	S0617	21	Male	7.4	99.2	77.0	B

2. Have a look on the Data and clean it (if required)

In [29]:
```
#2.1 Basic info about dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1020 entries, 0 to 1019
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Student_ID      1020 non-null   object 
 1   Age             1020 non-null   int64  
 2   Gender          1020 non-null   object 
 3   Study_Hours     989 non-null    float64
 4   Attendance(%)   990 non-null    float64
 5   Test_Score      1020 non-null   float64
 6   Grade           1011 non-null   object 
dtypes: float64(3), int64(1), object(3)
memory usage: 55.9+ KB
```

In [30]:
```
#2.2check For duplicate values
df.duplicated().sum()
```

Out[30]:
```
20
```

In [31]:
```
# 2.3 There are 20 duplicate values ,remove them
df=df.drop_duplicates()
df.duplicated().sum()
```

Out[31]:
```
0
```

In [32]:
```
#2.4 Inspect for missing values and handle them
df.isnull().sum()
```

Out[32]:
```
Student_ID        0
Age               0
Gender            0
Study_Hours      30
Attendance(%)    30
Test_Score        0
Grade             8
dtype: int64
```

In [36]:
```
#Fill missing values with mean for Study_Hours with 0,Attendance(%) with m
df['Study_Hours']=df['Study_Hours'].fillna(0)
df['Attendance(%)']=df['Attendance(%)'].fillna(df['Attendance(%)'].mean())
#Fill grade with none
df['Grade']=df['Grade'].fillna('None')
df.isnull().sum()
```

Out[36]:
```
Student_ID       0
Age              0
Gender           0
Study_Hours      0
Attendance(%)    0
Test_Score       0
Grade            0
dtype: int64
```

3. Descriptive Analysis (Summarizing the Numbers)

In [37]:
```
1 df.describe()
```

Out[37]:

	Age	Study_Hours	Attendance(%)	Test_Score
count	1000.0000	1000.000000	1000.000000	1000.00000
mean	20.9600	5.006700	84.845773	49.80800
std	2.0036	2.671674	9.820483	21.18745
min	18.0000	-2.000000	50.300000	0.00000
25%	19.0000	3.600000	78.000000	35.00000
50%	21.0000	5.000000	84.845773	50.00000
75%	23.0000	6.300000	91.725000	64.00000
max	24.0000	25.000000	115.600000	100.00000

Interpretation

Example interpretation:

Mean Test_Score = 49.8

Median Test_Score = 50

Insight Some low scorers are pulling the average down.

4. Categorical Data Analysis

In [75]:
```
1 #check whether the data is balanced or not
2 df["Gender"].value_counts()
3
```

Out[75]:
```
Male      519
Female    481
Name: Gender, dtype: int64
```

We have to Check:

Is dataset balanced?

Any bias risk?

In [39]:
```
#Grade Distribution
df["Grade"].value_counts()

```

Out[39]:
```
F       507
C       257
B       148
A        48
A+       32
None      8
Name: Grade, dtype: int64
```

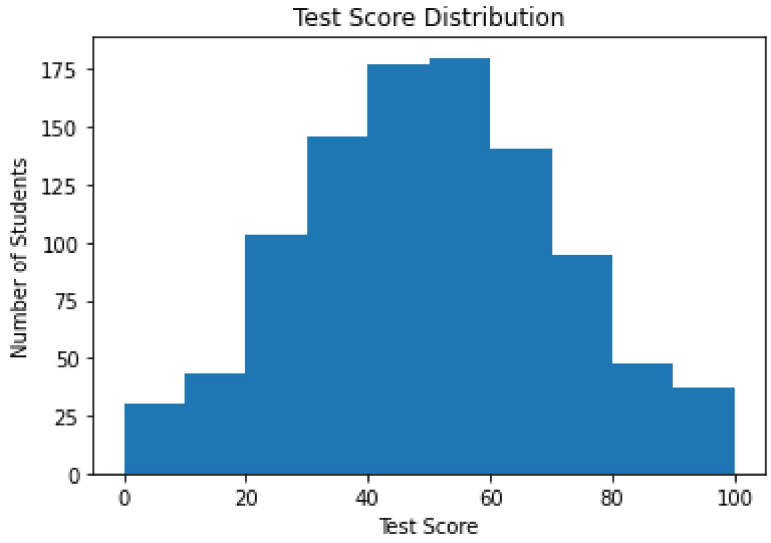
Insight:

Too many C or D grades may indicate teaching or attendance issues.

5. Visual Analytics: Seeing Patterns

Univariate Analysis

In [40]:
```
#5.1 Histogram for Test Score Distribution
plt.hist(df["Test_Score"], bins=10)
plt.xlabel("Test Score")
plt.ylabel("Number of Students")
plt.title("Test Score Distribution")
plt.show()

```



What to look for:

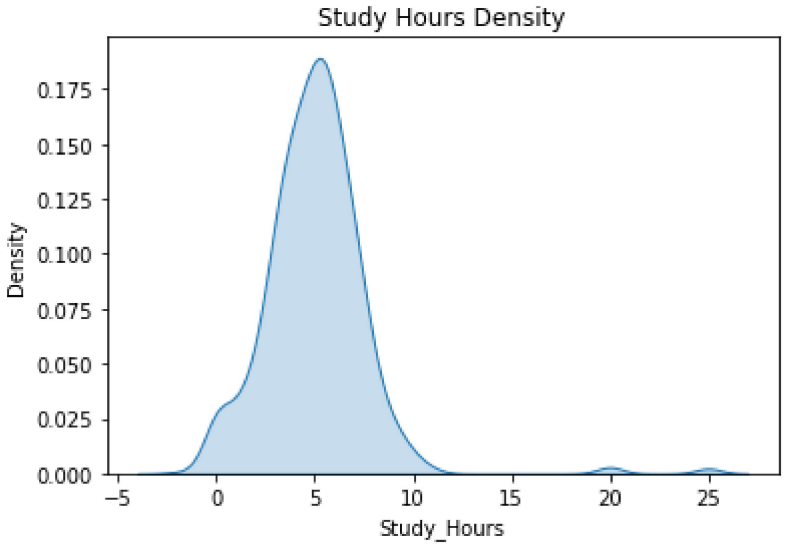
Skewed distribution

Too many low or high scores

Possible cut-off effects

Interpretation: The test scores are normally distributed with mean approx. 50

In [41]:
```
#5.2 KDE Plot for Study Hours
import seaborn as sns
sns.kdeplot(df["Study_Hours"], fill=True)
plt.title("Study Hours Density")
plt.show()
```



Insight:

Most students study in a narrow range

Very high study hours may not be common

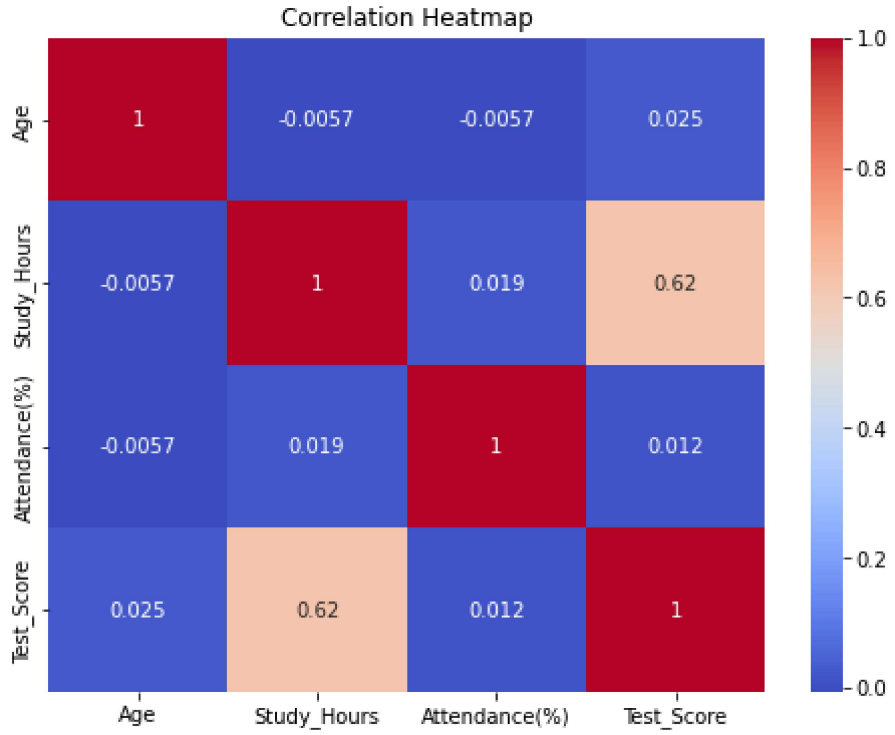
6. Correlation Study

In [42]:
```
df.corr()

```

Out[42]:

	Age	Study_Hours	Attendance(%)	Test_Score
Age	1.000000	-0.005747	-0.005721	0.024649
Study_Hours	-0.005747	1.000000	0.019061	0.623135
Attendance(%)	-0.005721	0.019061	1.000000	0.012120
Test_Score	0.024649	0.623135	0.012120	1.000000

In [43]:
```
#Heatmap
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()

```



Key questions

Does Attendance correlate with Test_Score?

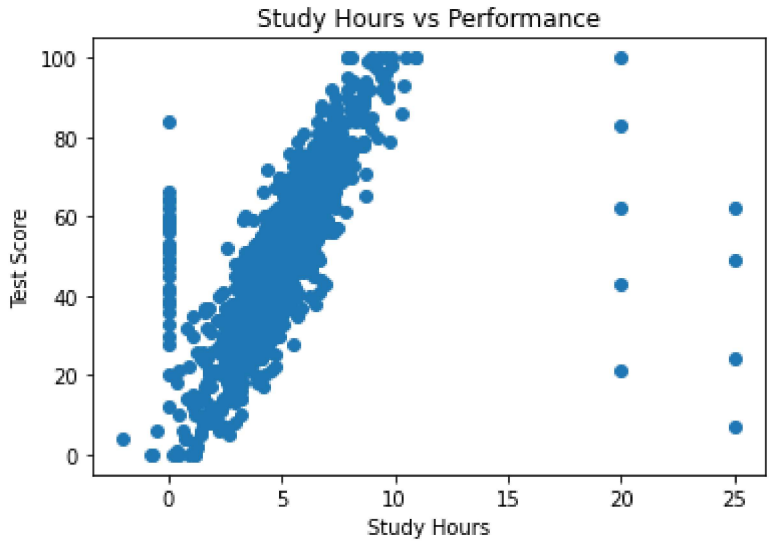
Does Study_Hours matter more than Attendance?

Is Age irrelevant?

7. Scatter Plots (Relationship in Action)

Bivariate Analysis

In [44]:
```
plt.scatter(df["Study_Hours"], df["Test_Score"])
plt.xlabel("Study Hours")
plt.ylabel("Test Score")
plt.title("Study Hours vs Performance")
plt.show()

```



Core pattern (the main story)

There is a strong positive relationship between study hours and test score. This means:

In general, more study hours → better performance.

Outliers and unusual behavior

Now look at the right side:

There are students studying 20–25 hours, but:

Some score high

Some score medium

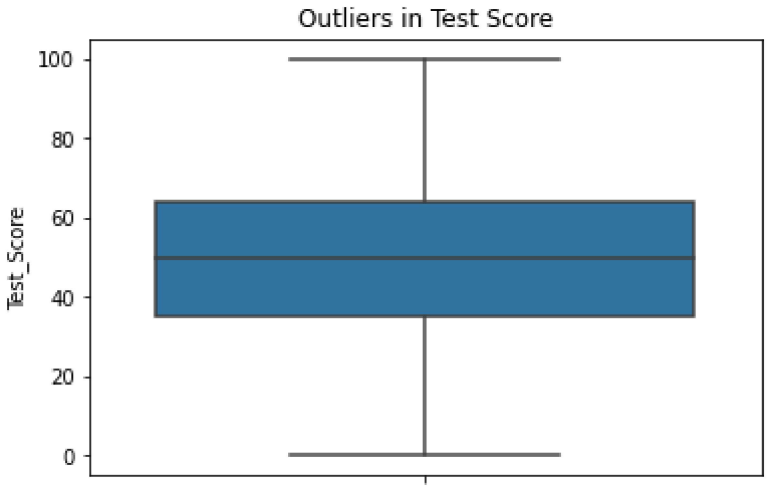
Some score very low

This tells us:

Conclusion : Studying more does not guarantee high scores.

8. Outlier Exploration (Very Important)

In [45]:
```
sns.boxplot(y=df["Test_Score"])
plt.title("Outliers in Test Score")
plt.show()
```



9. Feature Interactions (EDA Level-Up)

In [46]:
```
#Grade vs Attendance
df.groupby("Grade")["Attendance(%)"].mean()
```

Out[46]:
```
Grade
A       87.364231
A+      85.450000
B       84.882685
C       84.923360
F       84.551379
None    82.800000
Name: Attendance(%), dtype: float64
```

Insight:

Higher grades usually have higher attendance.

In [47]:
```
#Gender vs Performance
df.groupby("Gender")["Test_Score"].mean()
```

Out[47]:
```
Gender
Female    48.717256
Male      50.818882
Name: Test_Score, dtype: float64
```

Conclusion : Male Category have high scores

10. Create Buckets (creating new feature)

In [48]:
```
#Attendance Buckets
df["Attendance_Level"] = pd.cut(df["Attendance(%)"],bins=[0, 60, 80, 100],
    labels=["Low", "Medium", "High"]
)
df.head()

```

Out[48]:

	Student_ID	Age	Gender	Study_Hours	Attendance(%)	Test_Score	Grade	Attendance_Level
0	S0524	21	Female	2.7	74.0	37.0	F	Medium
1	S0603	19	Male	1.5	93.1	26.0	F	High
2	S0527	20	Female	6.8	97.2	88.0	A	High
3	S0032	18	Female	6.2	93.5	55.0	C	High
4	S0617	21	Male	7.4	99.2	77.0	B	High

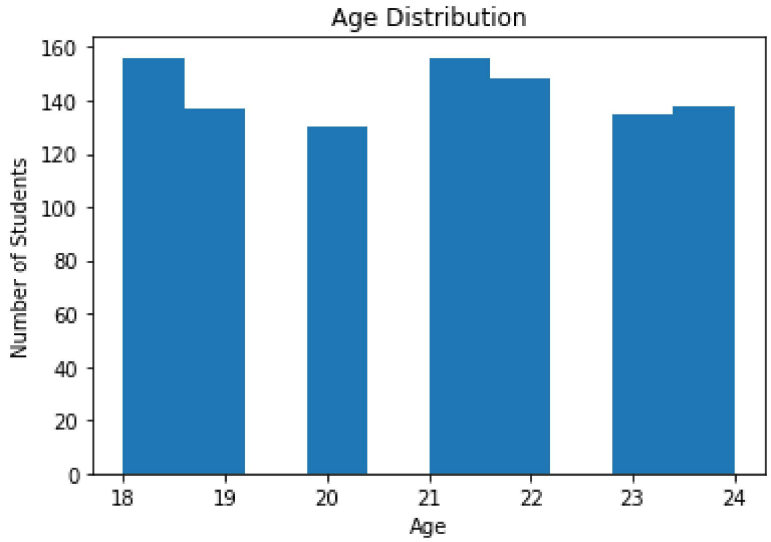
In [49]:
```
df.groupby("Attendance_Level")["Test_Score"].mean()
```

Out[49]:
```
Attendance_Level
Low       44.200000
Medium    49.859016
High      49.794913
Name: Test_Score, dtype: float64
```

We have created new feature called Attendence_level and analysed

11 Univariate Analysis on Numerical Variables

In [50]:
```
#visualize
plt.hist(df["Age"], bins=10)
plt.xlabel("Age")
plt.ylabel("Number of Students")
plt.title("Age Distribution")
plt.show()
```



In [ ]:
```
1
```