



UNITED
UNIVERSITY

**ASSIGNMENT REPORT
ON
STUDENT PERFORMANCE DATASET**

SUBMITTED BY :

NAME: SAMEER SINGH (24201020070)
ANKIT Kr. KAITHVASH (24201010029)
P. VYSHNAVI (24201010107)
FARISH ZAHEER (24201010066)

COURSE : BCA-IBM (2024-27) 2nd YEAR

**SUBMITTED TO:
PROF. SAMARTH AMRUTE**

**DEPARTMENT
COMPUTER APPLICATION
DATE:28 OCTOBER 2025**

1. Libraries Used: The code imports `numpy`, `pandas`, `matplotlib.pyplot`, and

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

```
df = pd.read_csv("StudentsPerformance.csv", lineterminator='\n')
df
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

`seaborn` for data analysis and visualization.

2. Dataset Loading: The dataset `StudentsPerformance.csv` is read into a DataFrame using `pd.read_csv()`.
3. Data Structure: The DataFrame (`df`) has 1000 rows and 8 columns.
4. Columns Included:
- o `gender`, `race/ethnicity`, `parental level of education`, `lunch`, `test preparation course`, `math score`, `reading score`, and `writing score`.
5. Purpose: The dataset records student performance across subjects along with demographic and background information — useful for analyzing factors affecting academic scores.

```
df.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

1. The code `df.head()` displays the first five rows of the `StudentsPerformance` dataset.

2. The dataset includes columns: gender, race/ethnicity, parental level of education, lunch, test preparation course, math score, reading score, and writing score.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   gender          1000 non-null    object 
 1   race/ethnicity  1000 non-null    object 
 2   parental level of education  1000 non-null    object 
 3   lunch           1000 non-null    object 
 4   test preparation course  1000 non-null    object 
 5   math score      1000 non-null    int64  
 6   reading score   1000 non-null    int64  
 7   writing score   1000 non-null    int64  
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

```
df["race/ethnicity"].head()
```

1. The dataset has 1000 entries and 8 columns, with no missing (null) values.
2. There are 5 categorical (object) columns — like gender, race/ethnicity, and education — and 3 numerical (int64) columns — math, reading, and writing scores.

```
df["race/ethnicity"].head()
```

race/ethnicity

0	group B
1	group C
2	group B
3	group A
4	group C

dtype: object

1. The code `df["race/ethnicity"].head()` displays the first few values from the “race/ethnicity” column.

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

```
df.describe()
```

	math score	reading score	writing score	grid icon
count	1000.00000	1000.00000	1000.00000	info icon
mean	66.08900	69.16900	68.054000	
std	15.16308	14.600192	15.195657	
min	0.00000	17.000000	10.000000	
25%	57.00000	59.000000	57.750000	
50%	66.00000	70.000000	69.000000	
75%	77.00000	79.000000	79.000000	
max	100.00000	100.000000	100.000000	

```
columns = ['lunch', 'test preparation course']  
df.drop(columns=columns, axis=1, inplace=True)  
df.columns  
  
Index(['gender', 'race/ethnicity', 'parental level of education', 'math score',  
       'reading score', 'writing score'],  
      dtype='object')
```

- No duplicate records — `df.duplicated().sum()` shows 0.
- Descriptive statistics — Dataset has 1000 entries for math, reading, and writing scores.
- Average scores — Mean values: Math (66.09), Reading (69.17), Writing (68.05).
- Score range — Minimum scores: 0 (math), 17 (reading), 10 (writing); Maximum: 100 for all.
- Columns retained — After dropping ‘lunch’ and ‘test preparation course’, columns are: `['gender', 'race/ethnicity', 'parental level of education', 'math score', 'reading score', 'writing score']`.

```
print(df['math score'].dtypes)
```

`int64`

```
df.head()
```

	gender	race/ethnicity	parental level of education	math score	reading score	writing score
0	female	group B	bachelor's degree	72	72	74
1	female	group C	some college	69	90	88
2	female	group B	master's degree	90	95	93
3	male	group A	associate's degree	47	57	44
4	male	group C	some college	76	78	75

Data type check: The `math score` column has data type `int64`.

First 5 rows displayed using `df.head()`.

Columns include: `gender`, `race/ethnicity`, `parental level of education`, `math score`, `reading score`, and `writing score`.

Example scores: Math scores range from 47 to 90 in the first few rows.

```
math_avg = np.mean(df['math score'])
reading_avg = np.mean(df['reading score'])
writing_avg = np.mean(df['writing score'])
print("Average Math Score:", math_avg)
print("Average Reading Score:", reading_avg)
print("Average Writing Score:", writing_avg, "\n")
```

Average Math Score: 66.089

Average Reading Score: 69.169

Average Writing Score: 68.054

Average Math Score: 66.089 — shows moderate math performance overall.

Average Reading Score: 69.169 — highest among all subjects, indicating stronger reading skills.

Average Writing Score: 68.054 — slightly lower than reading, but close in performance.

Observation: Students generally perform better in language-based subjects than in math.

```
df['total_marks'] = df['math score'] + df['reading score'] + df['writing score']
top_student = df.loc[df['total_marks'].idxmax()]
print("Top Student Details:")
print(top_student)
```

Top Student Details:

```
gender           female
race/ethnicity    group E
parental level of education   bachelor's degree
math score        100
reading score     100
writing score     100
total_marks       300
Name: 458, dtype: object
```

1.The code creates a new column **total_marks** by summing math, reading, and writing scores.

2.It identifies the student with the highest total marks using **idxmax()**.

3.The details of this top student are stored in the variable **top_student**.

4.The program then prints a label “Top Student Details:”.

```
top_students = df.sort_values(by='percentage', ascending=False).head()
print("Top 5 Students Based on Percentage:\n", top_students)
```

Top 5 Students Based on Percentage:

```
gender race/ethnicity parental level of education  math score \
916   male      group E          bachelor's degree      100
962 female      group E          associate's degree    100
458 female      group E          bachelor's degree      100
114 female      group E          bachelor's degree      99
712 female      group D          some college         98

      reading score  writing score  total_marks  percentage
916            100          100       300  100.000000
962            100          100       300  100.000000
458            100          100       300  100.000000
114            100          100       299  99.666667
712            100           99       297  99.000000
```

1.The code sorts the DataFrame **df** in descending order based on the **percentage** column.

2. It then selects the top 5 rows (students) with the highest percentages using `.head()`.

3. The result is stored in the variable `top_students`.

4. Finally, it prints the message “Top 5 Students Based on Percentage:” along with the data.

5. This helps to identify and display the five best-p

```
subjects = ['math score', 'reading score', 'writing score']
median_values = df[subjects].median()
print("Median Marks:\n", median_values, "\n")
```

```
Median Marks:
math score      66.0
reading score   70.0
writing score   69.0
dtype: float64
```

1. The code defines a list `subjects` containing three columns: *math score*, *reading score*, and *writing score*.

2. It calculates the median (middle value) for each subject using `.median()`.

3. The result is stored in the variable `median_values`.

4. It then prints the message “Median Marks:” followed by the median values.

```
subjects = ['math score', 'reading score', 'writing score']
mode_values = df[subjects].mode().iloc[0]
print("Mode Marks:\n", mode_values, "\n")
```

```
Mode Marks:
math score      65
reading score   72
writing score   74
Name: 0, dtype: int64
```

- The code lists three subjects: math score, reading score, and writing score.
- It calculates the mode (most frequently occurring value) for each subject using `.mode()`.
- The first row of mode values is selected with `.iloc[0]` and stored in `mode_values`.
- It prints “Mode Marks:” followed by the calculated mode values.
- This helps to identify the most common scores achieved by students in each subject.

```

print("Updated dataset with Average Marks:")
print(df.head())

```

Updated dataset with Average Marks:					
	gender	race/ethnicity	parental group	level of education	math score
0	female	group B		bachelor's degree	72
1	female	group C		some college	69
2	female	group B		master's degree	90
3	male	group A		associate's degree	47
4	male	group C		some college	76

	reading score	writing score	total_marks	percentage
0	72	74	218	72.666667
1	90	88	247	82.333333
2	95	93	278	92.666667
3	57	44	148	49.333333

- The code prints the message “Updated dataset with Average Marks:”.
- It then displays the first five rows of the DataFrame using `df.head()`.
- This allows a quick look at the updated dataset, likely after adding a new column (like average marks).
- It helps verify that the new data or calculations were added correctly.
- Essentially, it’s used for data validation and preview after modification.

4 78 75 229 76.333333

```

subjects = ['math score', 'reading score', 'writing score']
for subject in subjects:
    mean_val = np.mean(df[subject])
    median_val = np.median(df[subject])
    mode_val = df[subject].mode()[0]
    print(f" {subject.replace('_', ' ').title()}:")
    print(f"    Mean = {mean_val}")
    print(f"    Median = {median_val}")
    print(f"    Mode = {mode_val}\n")

```

 Math Score:
Mean = 66.089
Median = 66.0
Mode = 65

 Reading Score:
Mean = 69.169
Median = 70.0
Mode = 72

 Writing Score:
Mean = 68.054
Median = 69.0
Mode = 74

- The code calculates Mean, Median, and Mode for three subjects — Math, Reading, and Writing.
- It uses NumPy functions for mean (`np.mean`) and median (`np.median`).
- The mode is obtained using Pandas' `.mode()` method.
- For each subject, the results are printed neatly with formatted output.
- It provides a statistical summary of student scores for quick comparison.