COLONEL'S ACADEMY

Subject code: (065)

"Student Academic Performance Analysis Report"

Submitted by:

Gargi Naroliya

Submitted to:

Mrs Mala Patidar

Class:

12'A'

Subject:

Information Practices

Introduction

This report provides a data-driven analysis of academic performance for five students across five subjects. The central theme revolves around Education, Data Science, and Visualization. The objective is to leverage Python's data processing capabilities to compute total marks, percentages, and grades, and to present the findings through meaningful visualizations.

Python libraries such as Panda for data manipulation and Matplotlib for data visualisation, were used to structure and analyse the data. Visual tools like bar charts and pie charts make the report more intuitive and help uncover patterns such as subject-wise strengths and overall grade distribution.

This project demonstrates how programming can simplify and clarify academic performance evaluation.

Importing libraries

Libraries in Python are essential toolkits that support specific tasks. To begin, we must import the required libraries. The import statement is used to access these libraries within our program, and the as keyword is used to assign an alias for convenience.

```
# Importing python libraries
import pandas as pd
import matplotlib.pyplot as plt
```

.

- Pandas enable us to create and manage structured data in tabular form.
- Matplotlib..pyplot allows us to for the creation of bar charts, pie charts, and other graphical representations..

Creating the Dataset

To construct the dataset, we first organize the raw data using a nested dictionary. This structure is then converted into a tabular format using the DataFrame() function from Pandas.

```
#A dictionary with student data.
data = {
    'S_No': [101, 102, 103, 104, 105],
    'Stu_Name': ['Avni', 'Bhavya', 'Lubhanshi', 'Anshika', 'Soumya'],
    'Humanities': [75, 82, 68, 10, 78],
    'English': [88, 76, 12, 80, 85],
    'Physics': [65, 20, 85, 75, 90],
    'Chemistry': [38, 85, 70, 68, 82],
    'Maths': [92, 88, 78, 95, 18]
}
```

```
# Converting the data dictionary into a pandas DataFrame and display it
df = pd.DataFrame(data)
df
```

This code will show the output :

	S_No	Stu_Name	Humanities	English	Physics	Chemistry	Maths
0	101	Avni	75	88	65	38	92
1	102	Bhavya	82	76	20	85	88
2	103	Lubhanshi	68	12	85	70	78
3	104	Anshika	10	80	75	68	95
4	105	Soumya	78	85	90	82	18

Calculating Total Marks and Percentage

To compute the total marks for each student, we use the sum(axis=1) function, which aggregates data across each row (i.e., student-wise totals)

```
# Adding a new column 'Total_Marks' to calculate the total subject marks
df['Total_Marks'] = df[['Humanities', 'English', 'Maths', 'Physics', 'Chemistry']].sum(axis=1)
df
```

This results in the creation of a new column, Total_Marks.

	S_No	Stu_Name	Humanities	English	Physics	Chemistry	Maths	Total_Marks
0	101	Avni	75	88	65	38	92	358
1	102	Bhavya	82	76	20	85	88	351
2	103	Lubhanshi	68	12	85	70	78	313
3	104	Anshika	10	80	75	68	95	328
4	105	Soumya	78	85	90	82	18	353

Next, we calculate each student's percentage out of 500 marks. A new column, Percentage, is created to reflect this data.

```
# Adding a new column 'Percentage' based on total marks to calculate percentage of each student.
df['Percentage'] = (df['Total_Marks'] / 500) * 100
df
```

The generated output will be :

	S_No	Stu_Name	Humanities	English	Physics	Chemistry	Maths	Total_Marks
0	101	Avni	75	88	65	38	92	358
1	102	Bhavya	82	76	20	85	88	351
2	103	Lubhanshi	68	12	85	70	78	313
3	104	Anshika	10	80	75	68	95	328
4	105	Soumya	78	85	90	82	18	353

Assigning Grades

Grades are assigned based on percentage values using a custom function defined with def. This function standardizes the grading logic and applies it across all student records:

```
# Defines the grading and adds a Grade column based on percentage.
def get_grade(percentage):
    if percentage >= 90:
        return 'A+'
    elif percentage >= 80:
        return 'A'
    elif percentage >= 70:
        return 'B'
    elif percentage >= 60:
        return 'C'
    elif percentage >= 40:
        return 'D'
    else:
        return 'F'

df['Grade'] = df['Percentage'].apply(get_grade)
df
```

	S_No	Stu_Name	Humanities	English	Physics	Chemistry	Maths	Total_Marks	Percentage	Grade
0	101	Avni	75	88	65	38	92	358	71.6	B
1	102	Bhavya	82	76	20	85	88	351	70.2	B
2	103	Lubhanshi	68	12	85	70	78	313	62.6	C
3	104	Anshika	10	80	75	68	95	328	65.6	C
4	105	Soumya	78	85	90	82	18	353	70.6	B

Visualizations

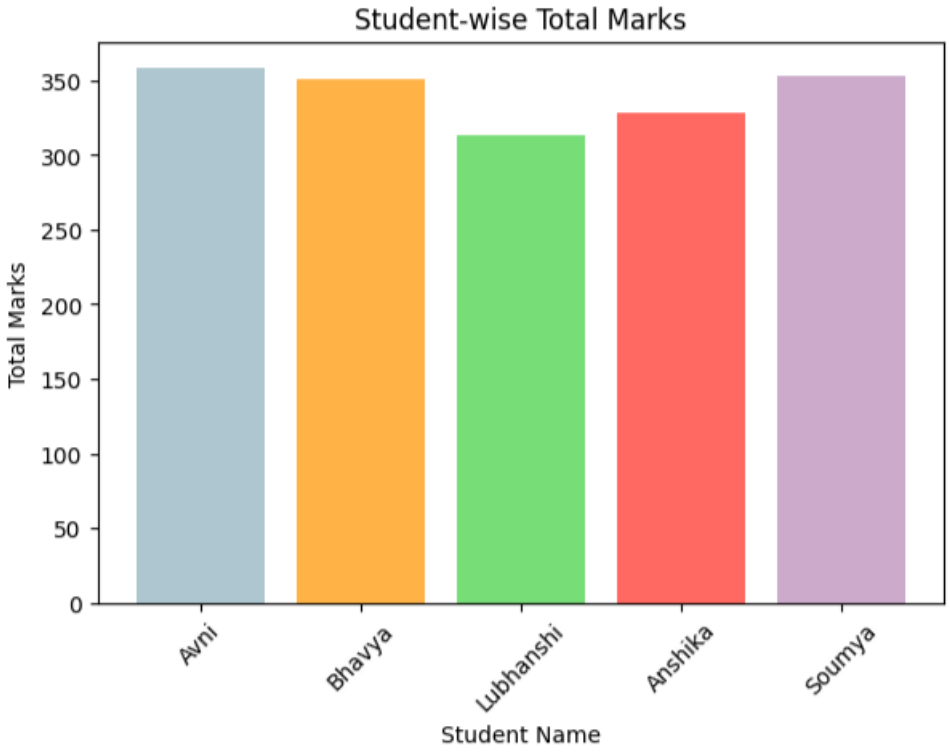
"Visualizations help us see patterns and comparisons more clearly than raw numbers alone.."

With the data prepared and calculations complete, we now move on to the visualization phase to better understand and communicate the results through graphical representations."

- Student-wise Total Marks – A bar chart showing each student's overall performance.

```
#1.graph for total mark of each student
plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
pastel_colors = ['#AEC6CF','#FFB347','#77DD77','#FF6961','#CBAACB']

plt.bar(df['Stu_Name'], df['Total_Marks'], color=pastel_colors)
plt.title('Student-wise Total Marks')
plt.xlabel('Student Name')
plt.ylabel('Total Marks')
plt.xticks(rotation=45)
```

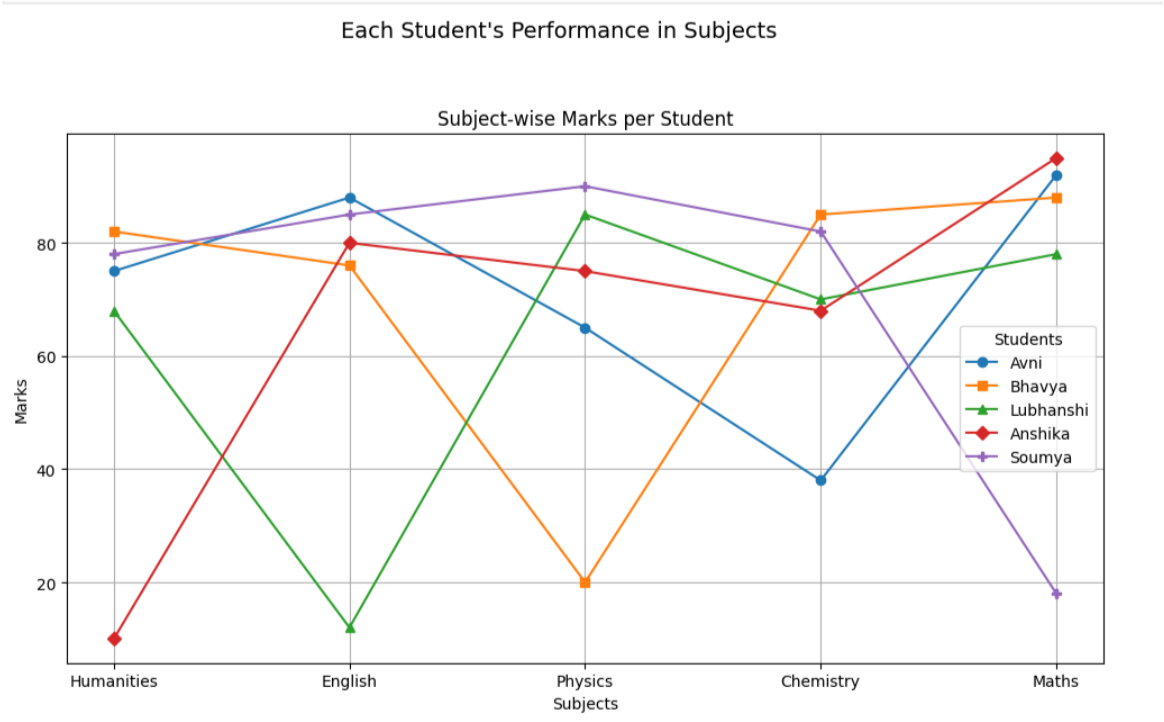
(Below is the graphical representation of each student's total marks.



- Subject-wise Average Marks – A bar chart representing the average marks in each subject.

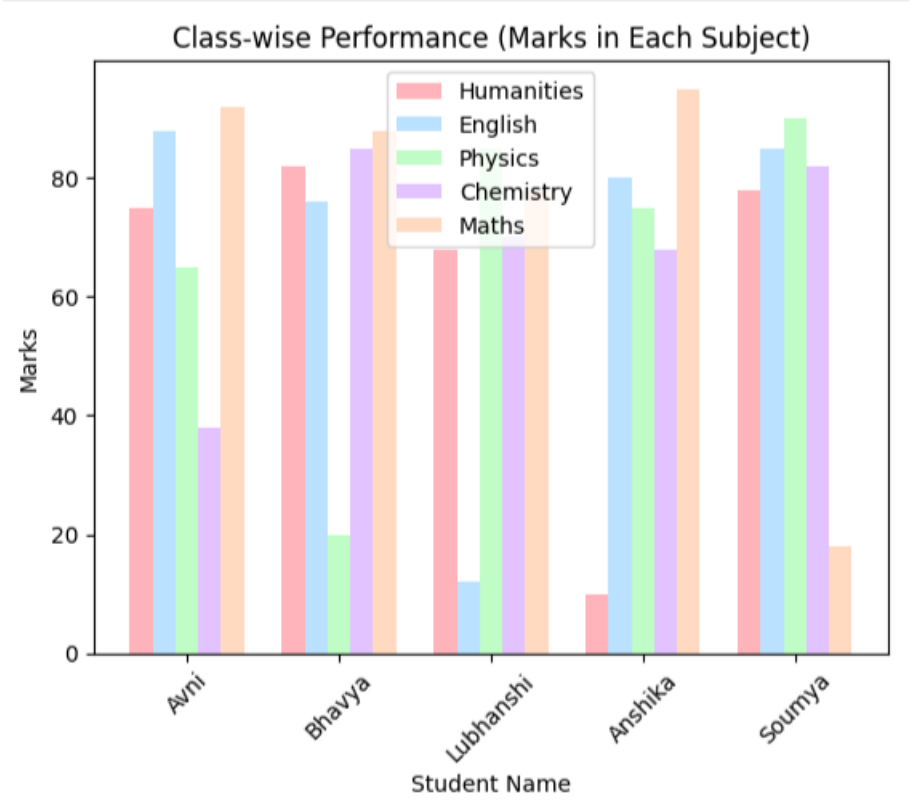
(line chart)

```
#2. line chart of performnce of each student in each subject
subjects = ['Humanities', 'English', 'Physics', 'Chemistry', 'Maths']

markers = ['o', 's', '^', 'D', 'P']
plt.figure(figsize=(10, 6))

for i, row in df.iterrows():
    student_name = row['Stu_Name']
    marks = [row[subject] for subject in subjects]
    plt.plot(subjects, marks, marker=markers[i % len(markers)], label=student_name)

plt.title('Subject-wise Marks per Student')
plt.xlabel('Subjects')
plt.ylabel('Marks')
plt.legend(title="Students")
plt.grid(True)
plt.suptitle("Each Student's Performance in Subjects", fontsize=14, y=1.05)
plt.tight_layout()
plt.show()
```



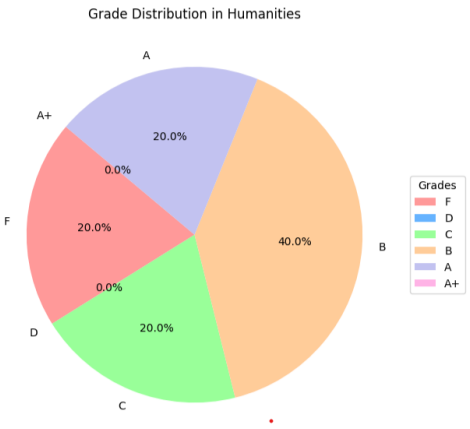
(bar graph)

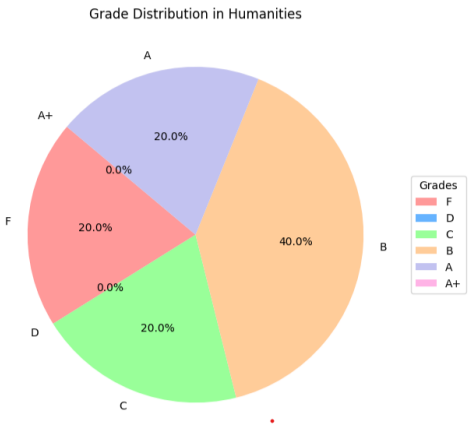
```
# 3.bar graph for classwise performance of each student in each subject
bar_width = 0.15
index = np.arange(len(df['Stu_Name']))
colors = {'Humanities': '#FFB3BA',   'English': '#BAE1FF','Physics': '#BFFCC6','Chemistry': '#E2C2FF',    'Maths': '#FFDAC1'}

plt.bar(index, df['Humanities'], bar_width, label='Humanities', color=colors['Humanities'])
plt.bar(index + bar_width, df['English'], bar_width, label='English', color=colors['English'])
plt.bar(index + 2 * bar_width, df['Physics'], bar_width, label='Physics', color=colors['Physics'])
plt.bar(index + 3 * bar_width, df['Chemistry'], bar_width, label='Chemistry', color=colors['Chemistry'])
plt.bar(index + 4 * bar_width, df['Maths'], bar_width, label='Maths', color=colors['Maths'])

plt.xlabel('Student Name')
plt.ylabel('Marks')
plt.title('Class-wise Performance (Marks in Each Subject)')
plt.xticks(index + 2 * bar_width, df['Stu_Name'], rotation=45)
plt.legend()
plt.show()
```

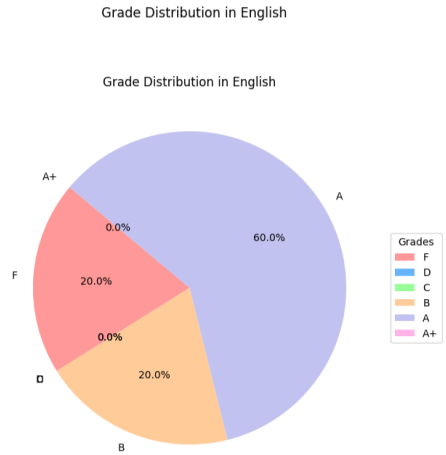


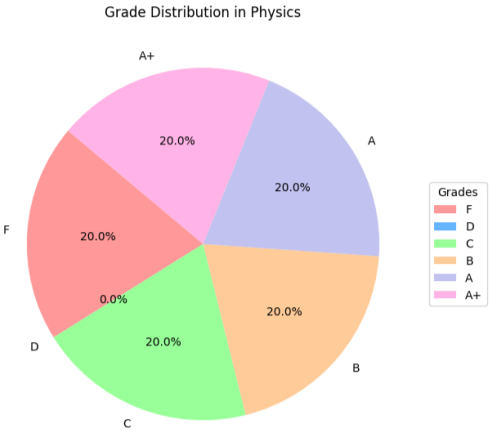
- Grade Distribution – A pie chart illustrating how grades are distributed among the students.

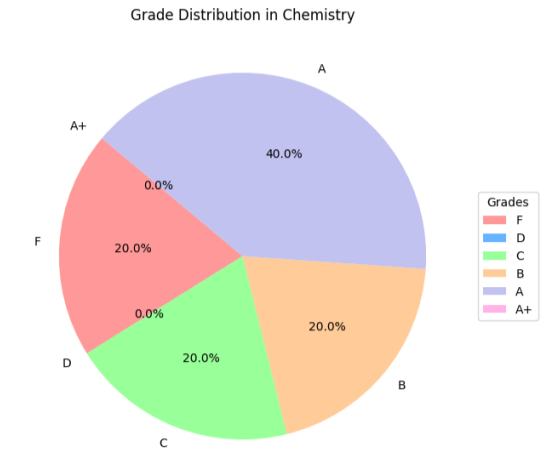
```
# 3.pie chart for classwise performance of each student in each subject
def assign_grade(marks):
    if marks >= 90:
        return 'A+'
    elif marks >= 80:
        return 'A'
    elif marks >= 70:
        return 'B'
    elif marks >= 60:
        return 'C'
    elif marks >= 40:
        return 'D'
    else:
        return 'F'

subjects = ['Humanities', 'English', 'Physics', 'Chemistry', 'Maths']
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#c2c2f0', '#ffb3e6']

for subject in subjects:
    plt.figure(figsize=(7, 7))

    df[subject] = pd.to_numeric(df[subject], errors='coerce')

    grades = df[subject].apply(assign_grade)

    grade_counts = grades.value_counts().reindex(['F', 'D', 'C', 'B', 'A', 'A+'], fill_value=0)

    wedges, texts, autotexts = plt.pie(
        grade_counts,
        labels=grade_counts.index,
        autopct='%1.1f%%',
        colors=colors,
        startangle=140
    )

    plt.title(f'Grade Distribution in {subject}')
    plt.legend(wedges, grade_counts.index, title="Grades", loc="center left", bbox_to_anchor=(1, 0.5))
    plt.show()
```

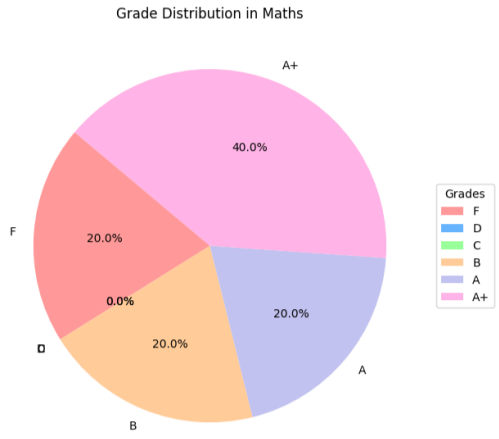

Grade Distribution in Humanities
A
A+
F
B
D
C
20.0%
0.0%
20.0%
40.0%
0.0%
20.0%
Grades
F
D
C
B
A
A+


Grade Distribution in Humanities
A
A+
F
B
D
C
20.0%
0.0%
20.0%
40.0%
0.0%
20.0%
Grades
F
D
C
B
A
A+


Grade Distribution in English
A+
A
F
B
0.0%
20.0%
0.0%
60.0%
20.0%
Grades
F
D
C
B
A
A+


Grade Distribution in Physics
A+
A
F
D
C
B
20.0%
20.0%
20.0%
0.0%
20.0%
20.0%
Grades
F
D
C
B
A
A+


Grade Distribution in Chemistry
A
A+
F
D
C
B
40.0%
0.0%
20.0%
0.0%
20.0%
20.0%
Grades
F
D
C
B
A
A+


Grade Distribution in Maths
A+
F
A
D
B
40.0%
20.0%
0.0%
20.0%
20.0%
Grades
F
D
C
B
A
A+

Conclusion

This project demonstrates the practical application of Python in educational data analysis. It simplifies the task of computing performance metrics and presents insights in a visually engaging manner. Through the use of libraries like Pandas and Matplotlib, we effectively transformed raw data into actionable information.
