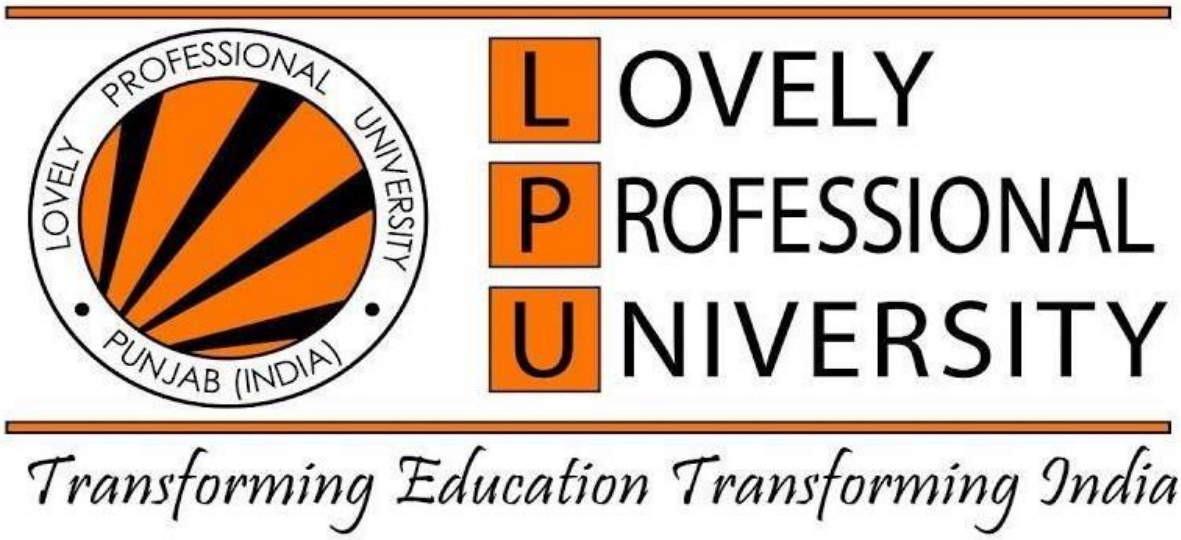


Name:	**K.Hitesh**
RegNo:	**12200832**
Section	**9w084**
Name of The Project	Result Management system

Project Title: Student Result Management and Analysis using PySpark and Hadoop

2. Introduction

- **Project Overview:** Briefly describe the project's goal: to manage and analyze student results using PySpark for efficient data processing.
- **Tools and Technologies:** List the tools used (PySpark, Pandas, Matplotlib, Seaborn) and mention the benefits of using PySpark (distributed processing, scalability).

3. Data Generation

- **Student Profiles:** Explain how you generated 10,000 student profiles with names and IDs using Pandas and converted it to a Spark DataFrame.
- **Subjects and Marks:** Describe the 6 subjects and how random marks were generated for each student using Spark.

  Code:

```
import pandas as pd

import numpy as np

import random

from pyspark.sql import SparkSession

from pyspark.sql import Row

from pyspark.sql import functions as F

import matplotlib.pyplot as plt

import seaborn as sns


# 1. Project Setup (Libraries imported)


# 2. Generate 10,000 Students' Profile

names = ["Ramesh", "Suresh", "Hitesh", "Mukesh", "Rajesh", "Mahesh", "Pankaj", "Sanjay", "Vikas", "Amit", "Karan", "Arjun"]

num_students = 10000

students = pd.DataFrame({

    'Student_ID': range(1, num_students + 1),

    'Name': [random.choice(names) + " " + random.choice(["Sharma", "Verma", "Patel", "Yadav", "Gupta"]) for _ in range(num_students)]

})

print(students.head())

spark = SparkSession.builder.appName("ResultManagement").getOrCreate()
```

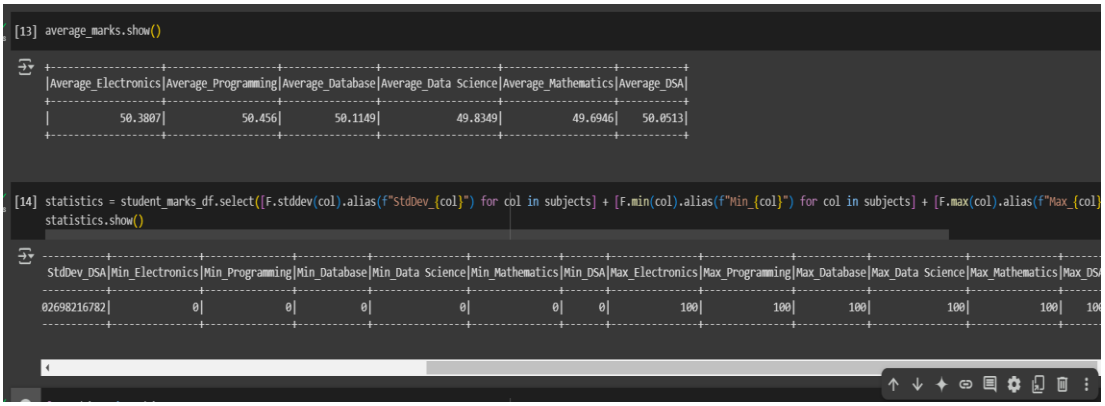
```
students_spark = spark.createDataFrame(students)


# 3. Generate 6 General Subjects
subjects = ["Electronics", "Programming", "Database", "Data Science",
"Mathematics", "DSA"]


# 4. Generate 6 Subject Marks for 10,000 Students
marks_data = []
for student in students_spark.collect():
    student_id = student.Student_ID
    marks_row = {"Student_ID": student_id}
    for subject in subjects:
        marks_row[subject] = random.randint(0, 100)
    marks_data.append(Row(**marks_row))
marks_df = spark.createDataFrame(marks_data)


# 5. Use Spark and Hadoop Framework
student_marks_df = students_spark.join(marks_df, "Student_ID")
for subject in subjects:
    student_marks_df = student_marks_df.withColumn(f"{subject}_Grade",
F.when(F.col(subject) >= 90, "A").when(F.col(subject) >= 80, "B").when(F.col(subject)
>= 70, "C").when(F.col(subject) >= 60, "D").otherwise("F"))
average_marks = student_marks_df.select([F.mean(col).alias(f"Average_{col}") for
col in subjects])


# 6. Do Basic Analysis and Statistics
average_marks.show()
statistics = student_marks_df.select([F.stddev(col).alias(f"StdDev_{col}")
```

Output

	Student_ID	Name
0	1	Karan Patel
1	2	Suresh Gupta
2	3	Pankaj Verma
3	4	Mukesh Verma
4	5	Mahesh Patel

4. Data Processing with PySpark

- **Joining DataFrames:** Explain how you joined the student profile and marks DataFrames using Student_ID.
- **Calculating Grades:** Describe the grade calculation logic (A, B, C, D, F) using PySpark's when function.
- **Calculating Average Marks:** Explain how you calculated the average marks for each subject using PySpark's aggregation functions.



5. Basic Analysis and Statistics

- **Average Marks:** Display the average marks for each subject in a table.
- **Other Statistics:** Include a table showing standard deviation, minimum, and maximum marks for each subject.
- **Top Performers:** List the top 5 performers in each subject with their Student_ID, Name, and marks.

```python
for subject in subjects:
    top_performers = student_marks_df.orderBy(F.col(subject).desc()).select("Student_ID", "Name", subject).limit(5)
    print(f"Top performers in {subject}:")
    top_performers.show()
```

```
Top performers in Electronics:
+----------+-------------+-----------+
|Student_ID|         Name|Electronics|
+----------+-------------+-----------+
|       414|Hitesh Sharma|        100|
|      4108| Mahesh Verma|        100|
|      1526| Mahesh Gupta|        100|
|        26| Mahesh Verma|        100|
|       823|Rajesh Sharma|        100|
+----------+-------------+-----------+

Top performers in Programming:
+----------+-------------+-----------+
|Student_ID|         Name|Programming|
+----------+-------------+-----------+
|      2235| Vikas Sharma|        100|
|      1733|  Vikas Yadav|        100|
|      1722| Ramesh Gupta|        100|
|      2812|  Karan Patel|        100|
|       768|Mahesh Sharma|        100|
```

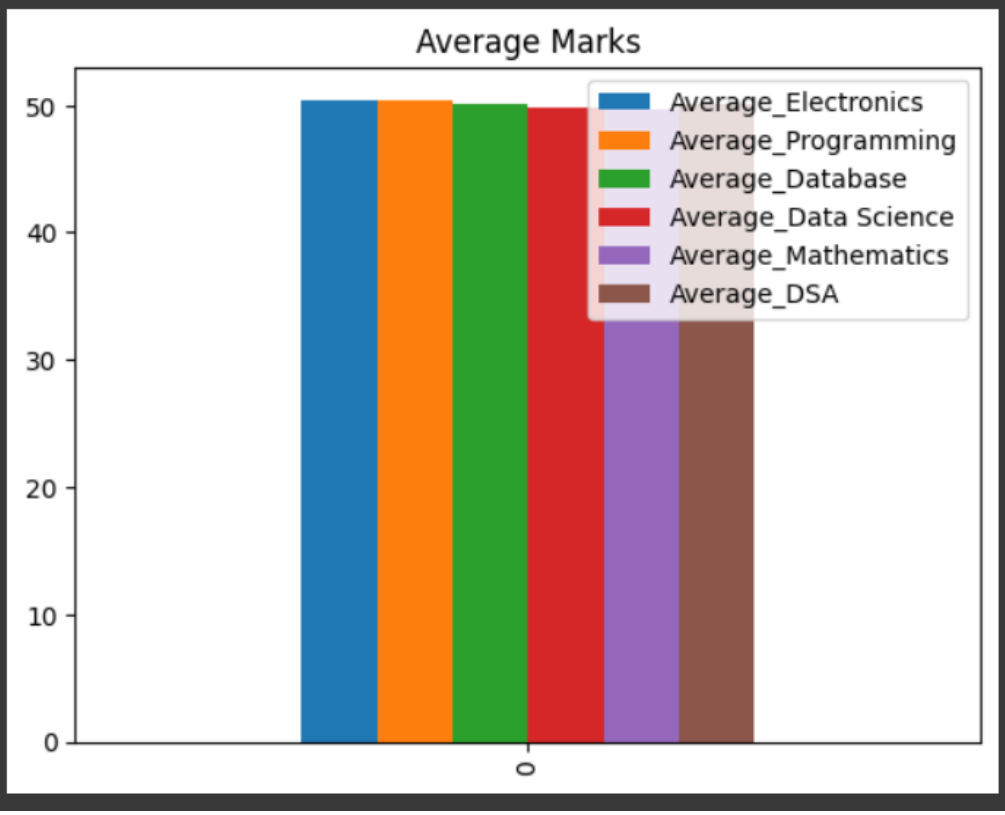
```
Top performers in Database:
+----------+-------------+--------+
|Student_ID|         Name|Database|
+----------+-------------+--------+
|      1362|Hitesh Sharma|     100|
|      2522| Arjun Sharma|     100|
|      1440| Rajesh Verma|     100|
|      1988| Mahesh Patel|     100|
|      5075|   Amit Yadav|     100|
+----------+-------------+--------+

Top performers in Data Science:
+----------+-------------+------------+
|Student_ID|         Name|Data Science|
+----------+-------------+------------+
|      3603|Mukesh Sharma|         100|
|       561|   Amit Gupta|         100|
|      3258| Sanjay Gupta|         100|
|      4171| Ramesh Gupta|         100|
|      4487| Ramesh Patel|         100|
+----------+-------------+------------+

Top performers in Mathematics:
+----------+-------------+-----------+
|Student_ID|         Name|Mathematics|
+----------+-------------+-----------+
|      1846|Pankaj Sharma|        100|
|      1650| Ramesh Gupta|        100|
|      4511|   Amit Patel|        100|
|      2612| Arjun Sharma|        100|
|      1219| Rajesh Yadav|        100|
+----------+-------------+-----------+
```

```
Top performers in Mathematics:
+----------+-------------+-----------+
|Student_ID|         Name|Mathematics|
+----------+-------------+-----------+
|      1846|Pankaj Sharma|        100|
|      1650| Ramesh Gupta|        100|
|      4511|   Amit Patel|        100|
|      2612| Arjun Sharma|        100|
|      1219| Rajesh Yadav|        100|
+----------+-------------+-----------+

Top performers in DSA:
+----------+-------------+---+
|Student_ID|         Name|DSA|
+----------+-------------+---+
|      2040| Mahesh Yadav|100|
|      3034| Arjun Sharma|100|
|      4220|   Amit Verma|100|
|      4354| Rajesh Verma|100|
|      1768|Rajesh Sharma|100|
+----------+-------------+---+
```

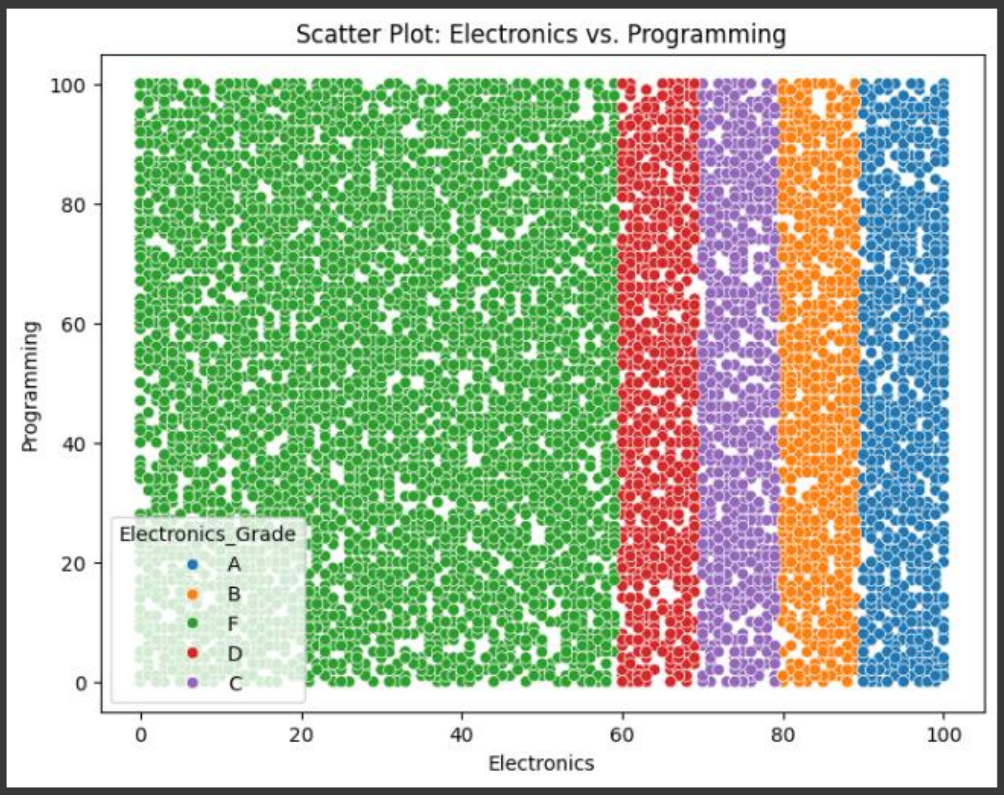
6. Data Visualization

- **Heatmap:** Include the heatmap generated using Seaborn, showing the correlation between subject marks. Briefly explain any observed patterns.
- **Average Marks Graph:** Include the bar graph showing average marks per subject.
- **Distribution Graphs:** Include histograms for each subject showing the distribution of marks.

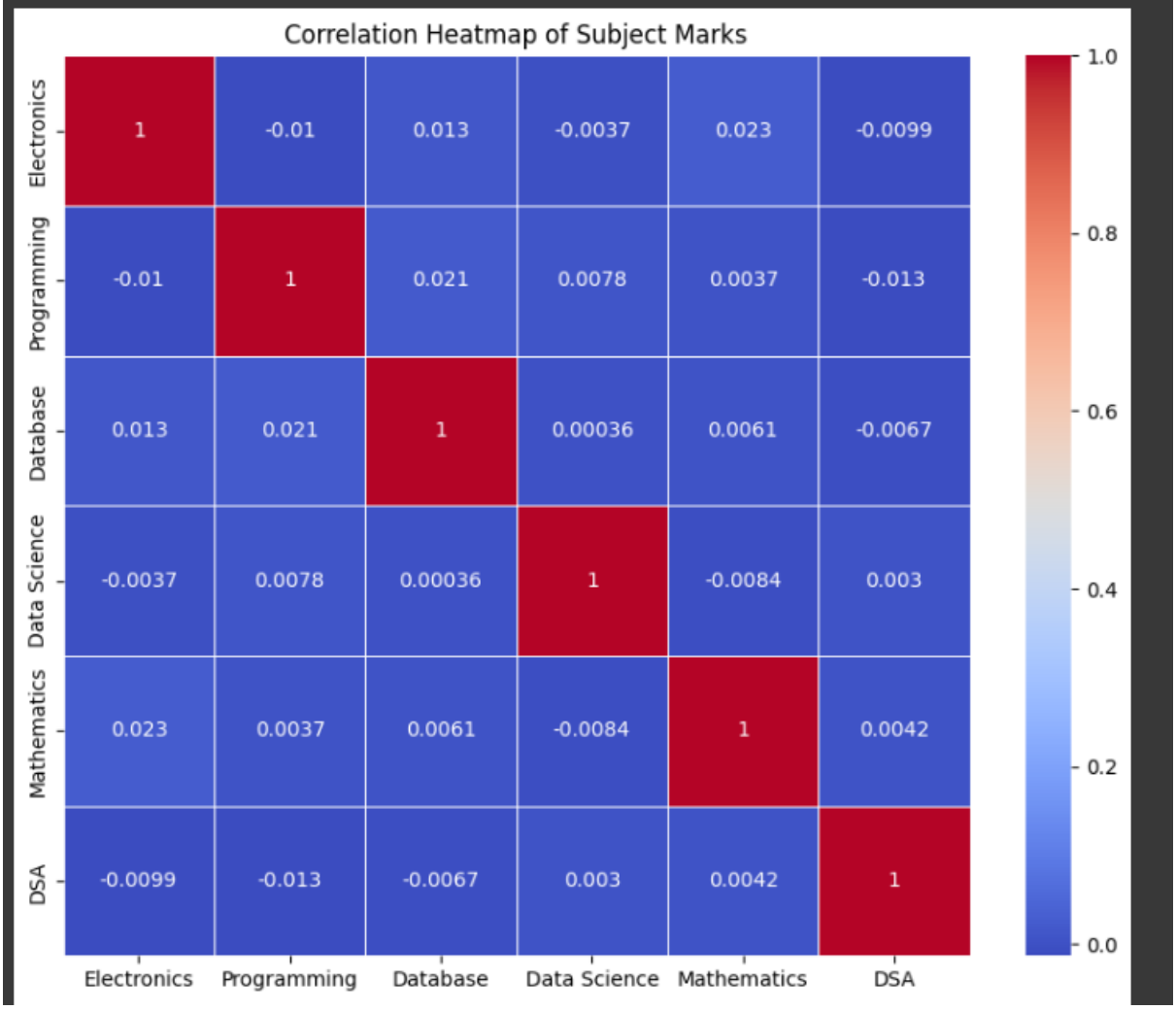
```
average_marks_pd.plot(kind='bar', title='Average Marks')
plt.show()
```



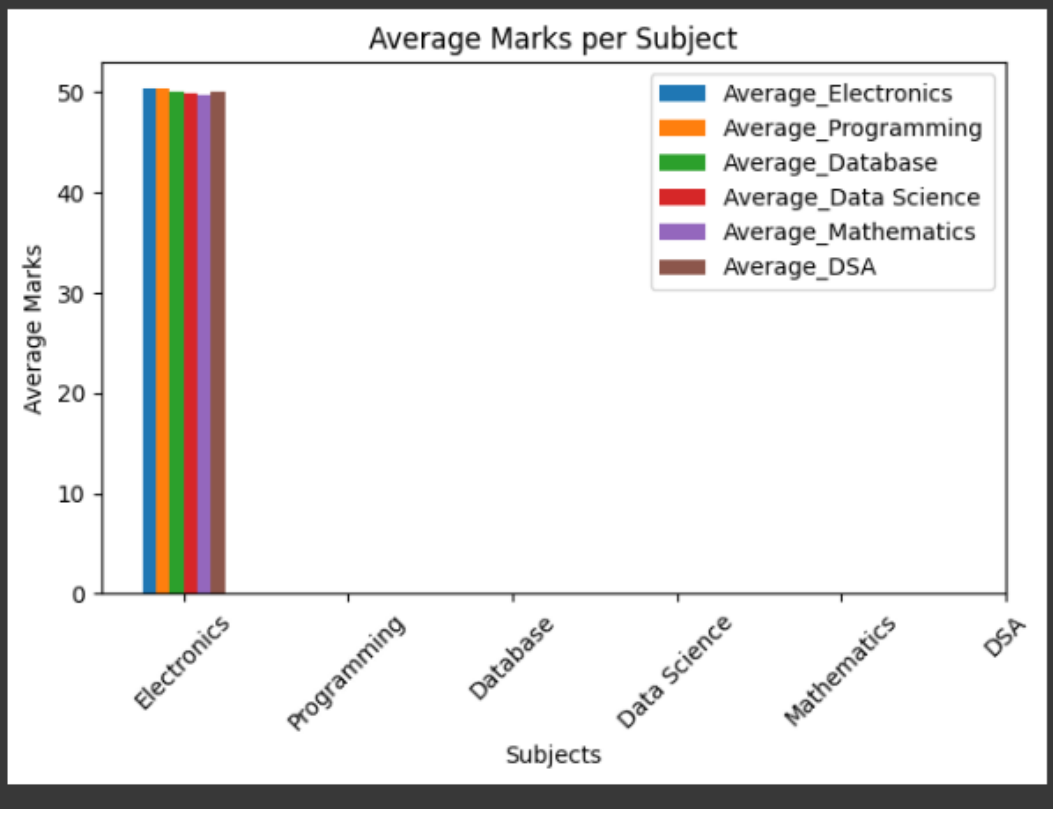
```python
# Scatter Plot (Example: Electronics vs. Programming)
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Electronics', y='Programming', data=student_marks_pd, hue='Electronics_Grade')
plt.title('Scatter Plot: Electronics vs. Programming')
plt.show()

spark.stop()
```



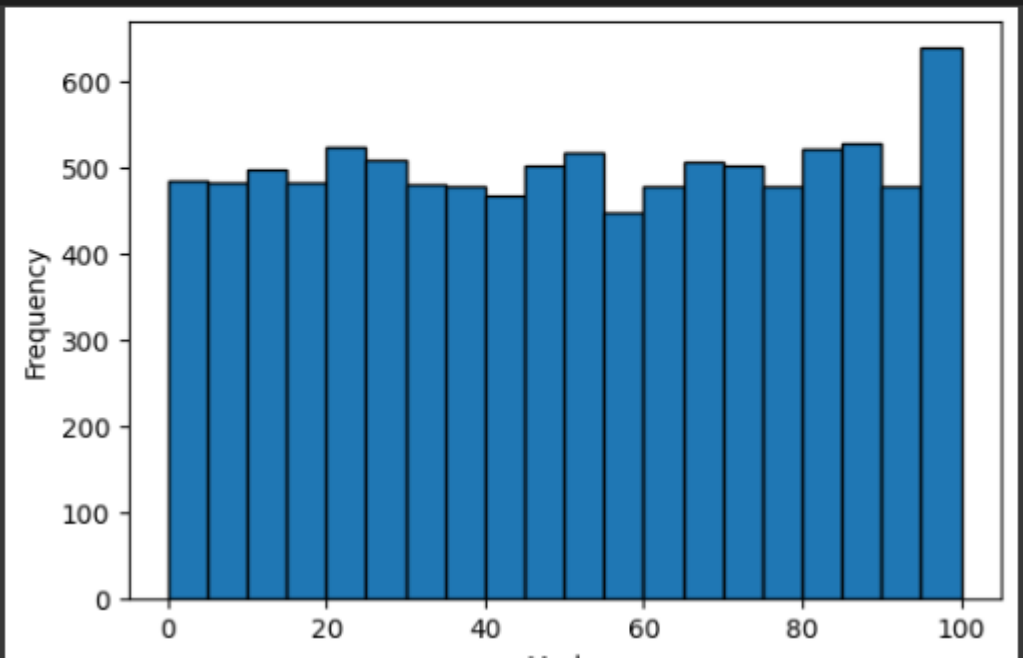
```
plt.figure(figsize=(10, 8))
sns.heatmap(subjects_marks.corr(), annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Correlation Heatmap of Subject Marks')
plt.show()
```

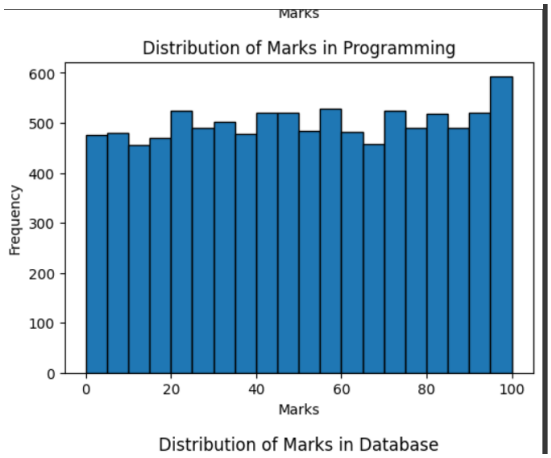


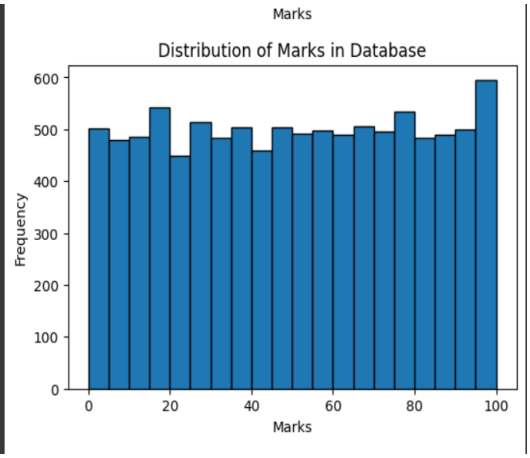
```
average_marks_pd.plot(kind='bar', title='Average Marks per Subject')
plt.xlabel('Subjects')
plt.ylabel('Average Marks')
plt.xticks(range(len(subjects)), subjects, rotation=45)
plt.tight_layout()
plt.show()
```



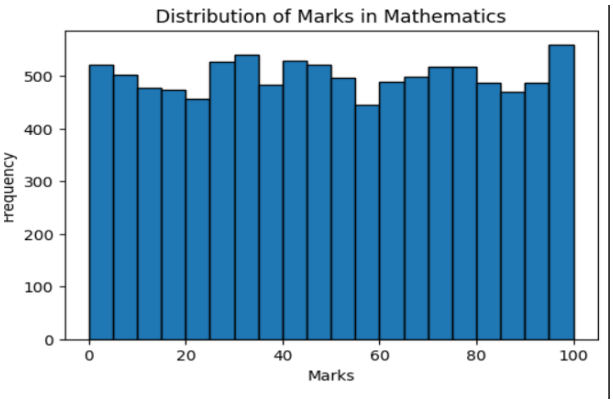
```python
for subject in subjects:
    plt.figure(figsize=(6, 4))
    plt.hist(student_marks_pd[subject], bins=20, edgecolor='black')
    plt.title(f'Distribution of Marks in {subject}')
    plt.xlabel('Marks')
    plt.ylabel('Frequency')
    plt.show()
```







Distribution of Marks in Database



7. Conclusion

- **Summary of Findings:** Briefly summarize the key insights from the analysis (e.g., average marks, top performers, correlations).
- **Benefits of PySpark:** Reiterate the advantages of using PySpark for this project.
- **Future Enhancements:** Suggest potential improvements (e.g., more detailed analysis, interactive dashboard, using real-world data).

The Mapreduce Function has Been Done In the Ubuntu