LOVELY PROFESSIONAL UNIVERSITY

NAME	**PAGADALA RISHEETH REDDY**
REG.NO	**12206174**
SECTION	**9W084**
PROJECT	**RESULT MANAGEMENT SYSTEM**

Introduction:

In large-scale educational institutions, managing and analysing student results efficiently is a challenging task. With 10,000 students and 6 core subjects, a traditional database system may not be scalable enough to handle massive datasets and perform real-time analytics. To address this, our Result Management System (RMS) leverages Big Data technologies, including Apache Spark, Hadoop, and MapReduce, to process and analyse student performance efficiently.

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:

Step 1:

pip install faker

Faker is a Python library that generates fake but realistic data such as names, addresses, emails, phone numbers, job titles, and much more.

Step 2:

```
import pandas as pd
import numpy as np
from faker import Faker
# Initialize Faker for Indian names
fake = Faker("en_IN")

# Define Subjects
subjects = ['Electronics', 'Programming', 'Database',
'Data Science', 'Mathematics', 'DSA']

# Generate 10,000 student profiles
num_students = 10000
unique_names = set()

# Generate unique names
while len(unique_names) < num_students:
    unique_names.add(fake.name())

unique_names = list(unique_names)
```

```
# Create student data
student_data = {
    "Student_ID": np.arange(1, 1 + num_students),
    "Name": unique_names,
}
# Generate random marks for each subject
for subject in subjects:
    student_data[subject] = np.random.randint(0, 95, num_students)  # Marks range 30-100

# Convert to DataFrame
df = pd.DataFrame(student_data)

# Save as CSV
df.to_csv("students_data.csv", index=False)

print("Student Data Generated Successfully!")
```

Output: Student Data Generated Successfully!

Data Processing with PySpark

- **Calculating Pass Marks:** Explain how you calculated the pass marks for each subject using PySpark's.
- **Joining DataFrames**: Explain how you joined the student profile and marks DataFrames using Student_ID.
- **Calculating Average Marks:** Explain how you calculated the average marks for each subject using PySpark's.

Code:

Step 1:

```
!pip install pyspark
```

Step 2:

```
from pyspark.sql import SparkSession
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
spark = SparkSession.builder.appName("Result Management System").getOrCreate()
df=pd.read_csv('students_data.csv')
```

step 3:

df_spark = spark.createDataFrame(df)

step 4:

from pyspark.sql.functions import avg, col

subjects = ["Electronics", "Programming", "Database", "Data Science", "Mathematics", "DSA"]

df_spark.select([avg(col(subject)).alias(f"Avg_{subject}") for subject in subjects]).show()

output:

```
+---------------+--------------+------------+----------------+---------------+-------+
|Avg_Electronics|Avg_Programming|Avg_Database|Avg_Data Science|Avg_Mathematics|Avg_DSA|
+---------------+--------------+------------+----------------+---------------+-------+
|        46.5634|       46.6442|     47.4887|         47.1029|        47.0849|46.5519|
+---------------+--------------+------------+----------------+---------------+-------+
```

Step 5:

from pyspark.sql.functions import count, when

pass_criteria = 40

```
pass_counts =
df_spark.select([(count(when(col(subject) >=
pass_criteria, subject)) * 100 /
df_spark.count()).alias(f"Pass_Percentage_{subject}")
for subject in subjects])

pass_counts.show()
```

optput:

```
+--------------------------+-------------------------+------------------------+-----------------------------+--------------------------+----------------------+
|Pass_Percentage_Electronics|Pass_Percentage_Programming|Pass_Percentage_Database|Pass_Percentage_Data Science|Pass_Percentage_Mathematics|Pass_Percentage_DSA|
+--------------------------+-------------------------+------------------------+-----------------------------+--------------------------+----------------------+
|                     57.23|                    57.71|                   58.46|                        58.29|                     57.81|                 57.24|
+--------------------------+-------------------------+------------------------+-----------------------------+--------------------------+----------------------+
```

Step 6:

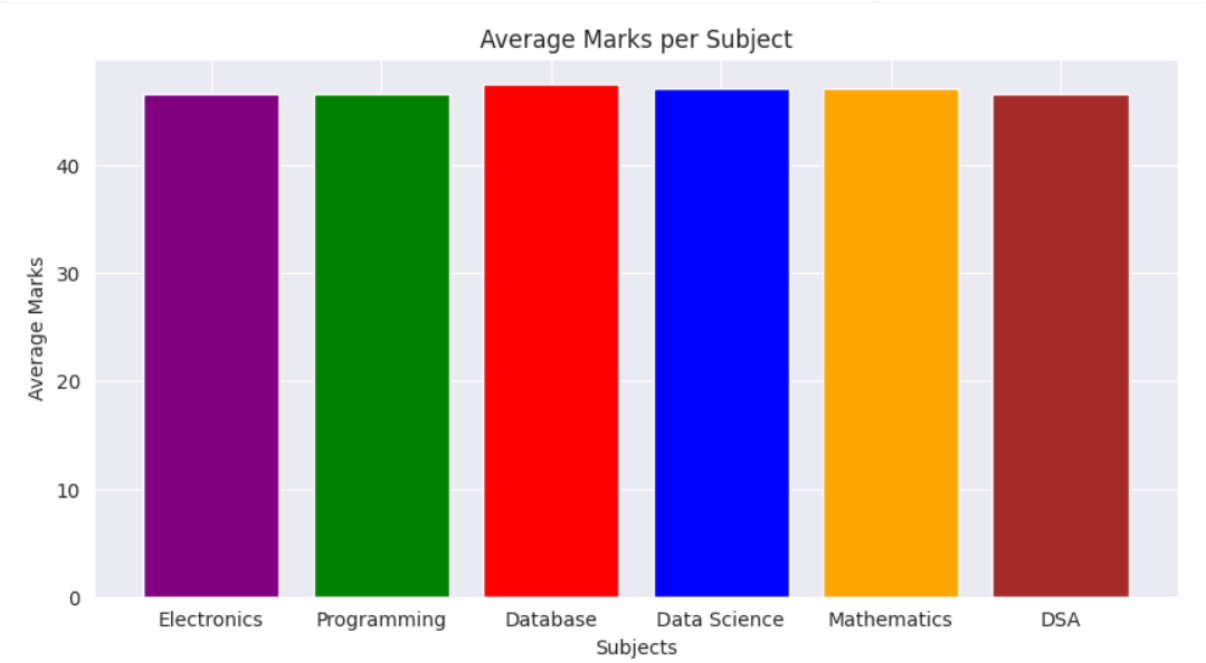
```
from pyspark.sql.functions import col

import pyspark.sql.functions as F


# Ensure you use the correct column names

for subject in subjects:

    top_performers =
df_spark.orderBy(F.col(subject).desc()).select("Stude
nt_ID", "Name", subject).limit(5)

    print(f"Top performers in {subject}:")

    top_performers.show()
```

```
Top performers in Electronics:
+----------+------------------+-----------+
|Student_ID|              Name|Electronics|
+----------+------------------+-----------+
|      5576|       Odika Ravel|         94|
|        76|         Avi Bains|         94|
|      5575|    Vedhika Sunder|         94|
|       250| Tamanna Chaudhari|         94|
|      5565|Anusha Chakrabarti|         94|
+----------+------------------+-----------+

Top performers in Programming:
+----------+------------------+-----------+
|Student_ID|              Name|Programming|
+----------+------------------+-----------+
|       182|     Oviya Trivedi|         94|
|      5418|     Yachana Chada|         94|
|       227|    Mahika Bhavsar|         94|
|      5861|      Yochana Rege|         94|
|       243|Nirja Ramachandran|         94|
+----------+------------------+-----------+
```

Data Visualizations:

Average marks per subject using Bar Graph:

```
avg_marks = df_pandas[subjects].mean()


plt.figure(figsize=(10, 5))
plt.bar(subjects, avg_marks, color=['purple', 'green', 'red', 'blue', 'orange', 'brown'])
plt.xlabel("Subjects")
plt.ylabel("Average Marks")
plt.title("Average Marks per Subject")
plt.show()
```



Correlation between subjects:

Code:

```
df_spark.select([col(subject).cast("double") for subject in subjects]).summary("mean").show()


for sub1 in subjects:
    for sub2 in subjects:
        if sub1 != sub2:
            print(f"Correlation between {sub1} and {sub2}: {df_spark.stat.corr(sub1, sub2)}")
```

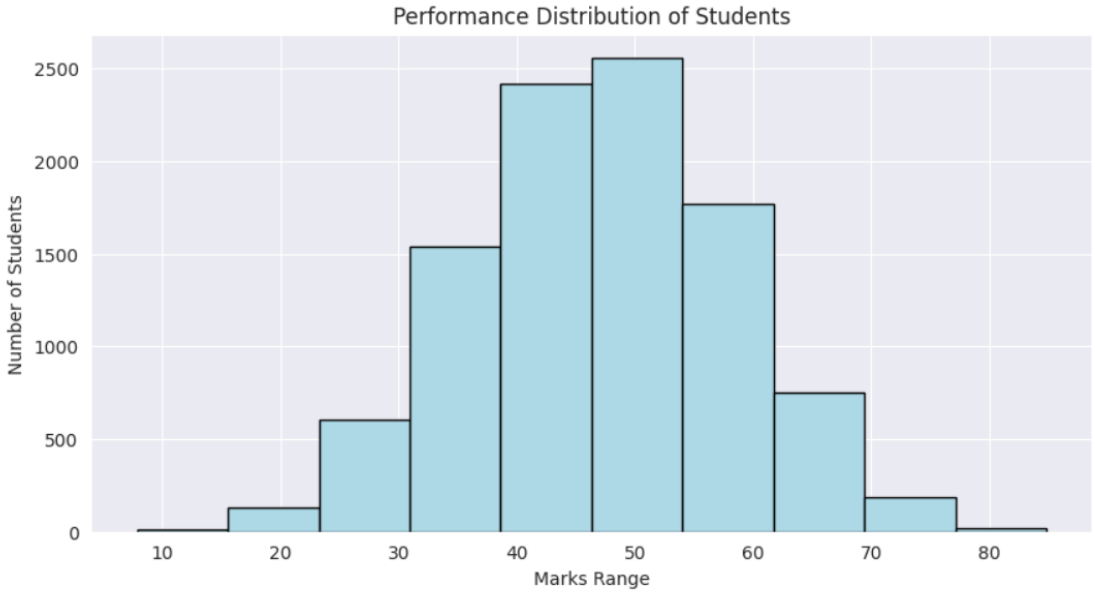
```
+-------+-----------+-----------+--------+------------+-----------+-------+
|summary|Electronics|Programming|Database|Data Science|Mathematics|    DSA|
+-------+-----------+-----------+--------+------------+-----------+-------+
|   mean|    47.2596|    47.1457| 47.4025|     47.0586|    47.1794|46.6242|
+-------+-----------+-----------+--------+------------+-----------+-------+

Correlation between Electronics and Programming: -0.013620469551361078
Correlation between Electronics and Database: 0.008318612145807407
Correlation between Electronics and Data Science: -0.011311323794809897
Correlation between Electronics and Mathematics: -0.014252268464066022
Correlation between Electronics and DSA: 0.01009253667261573
Correlation between Programming and Electronics: -0.013620469551361057
Correlation between Programming and Database: -0.002992532266844138
Correlation between Programming and Data Science: -0.0024855640823466334
Correlation between Programming and Mathematics: 0.0011215729990886715
Correlation between Programming and DSA: 0.0024896986691890018
Correlation between Database and Electronics: 0.008318612145807392
Correlation between Database and Programming: -0.002992532266844132
Correlation between Database and Data Science: 0.015709929497801985
Correlation between Database and Mathematics: -0.003220099347210993
Correlation between Database and DSA: -0.004005235266672666
```

Performance Distribution (Histogram of Marks)

Code:

```
import matplotlib.pyplot as plt


df_pandas = df_spark.toPandas()


plt.figure(figsize=(10, 5))

plt.hist(df_pandas["Average_Marks"], bins=10,
color="lightblue", edgecolor="black")
```

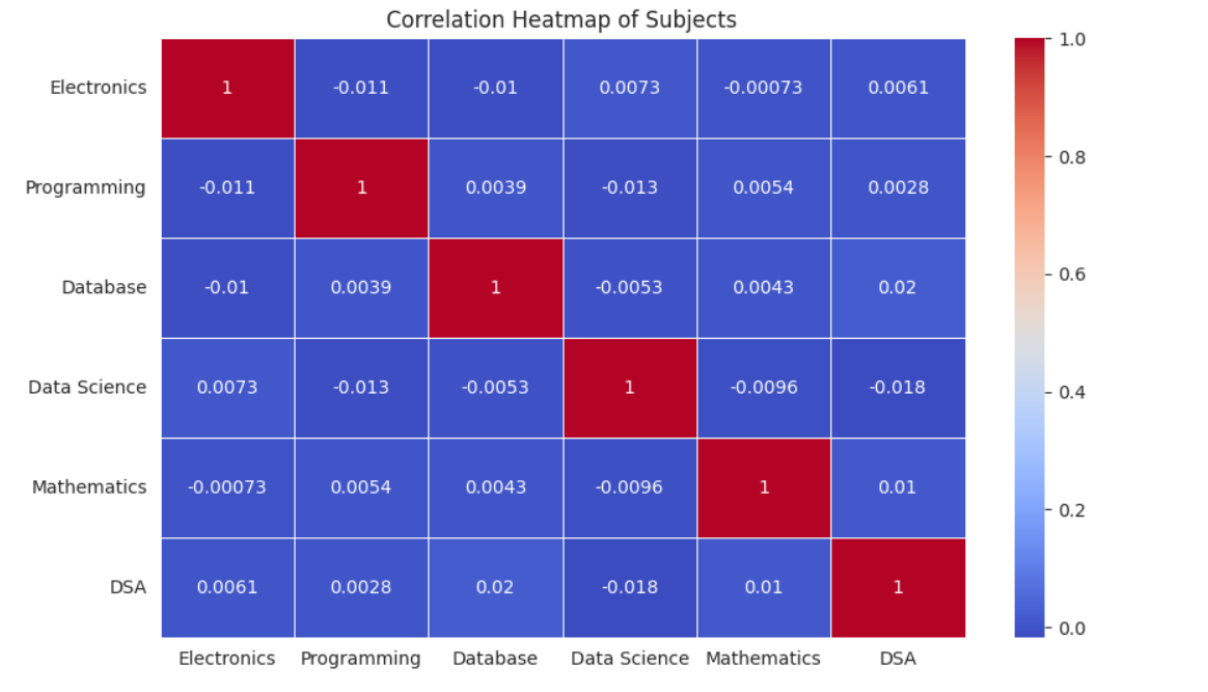
```
plt.xlabel("Marks Range")
plt.ylabel("Number of Students")
plt.title("Performance Distribution of Students")
plt.show()
```



Correlation Heatmap of Subjects

Code:

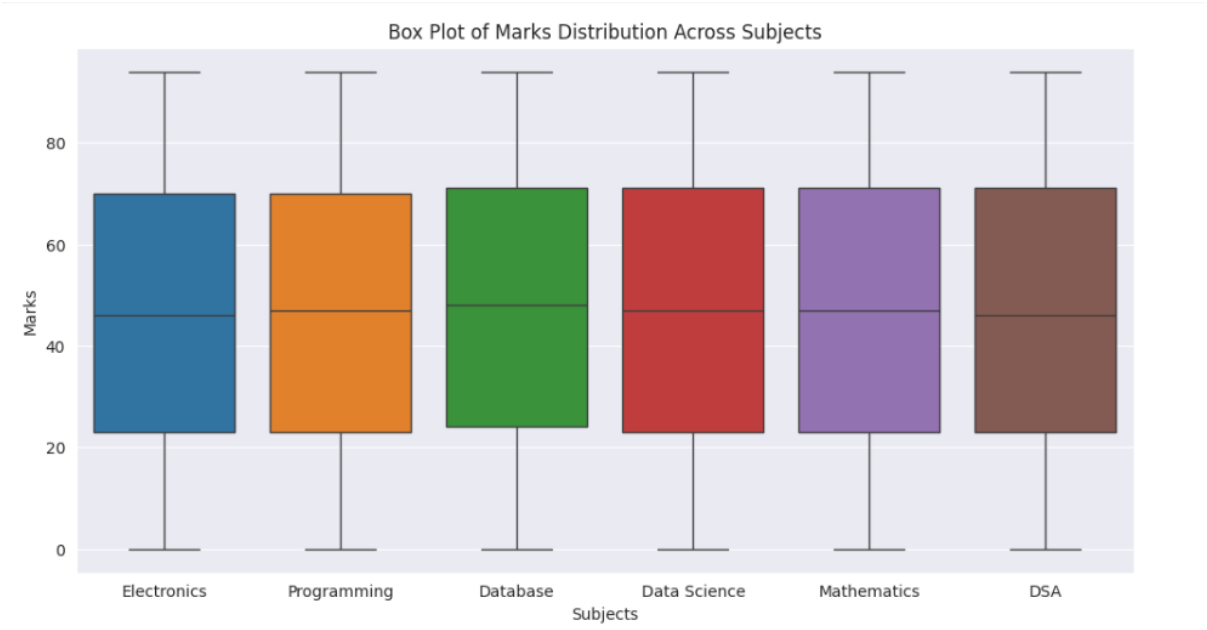
```
plt.figure(figsize=(10, 6))
corr_matrix = df[['Electronics', 'Programming', 'Database', 'Data Science', 'Mathematics', 'DSA']].corr()
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Correlation Heatmap of Subjects")
plt.show()
```



Box Plot of Marks Distribution Across Subjects

Code:

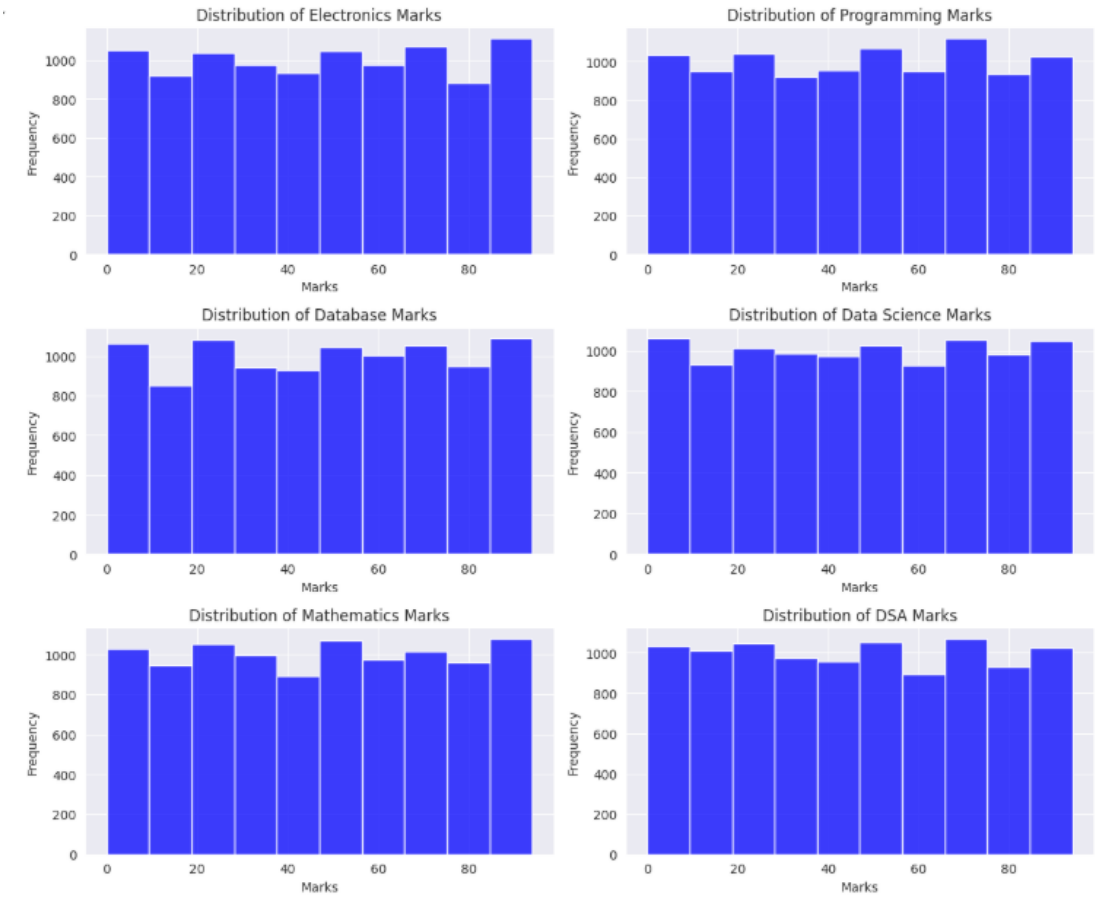
```
plt.figure(figsize=(12, 6))

sns.boxplot(data=df[['Electronics', 'Programming', 'Database', 'Data Science', 'Mathematics', 'DSA']])

plt.title("Box Plot of Marks Distribution Across Subjects")

plt.xlabel("Subjects")

plt.ylabel("Marks")

plt.show()
```



Distribution of Marks

Code:

```
plt.figure(figsize=(12, 10))
for i, subject in enumerate(subjects, 1):
    plt.subplot(3, 2, i)
    sns.histplot(df[subject], bins=10, kde=False, color='b')
    plt.title(f'Distribution of {subject} Marks')
    plt.xlabel('Marks')
    plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```



Conclusion & Learning Outcomes:

- The project successfully implemented PySpark for large-scale data handling and analysis.
- The use of various visualization techniques provided valuable insights into student performance.
- The project enhanced proficiency in big data processing and analytics.