```
!pip install pyspark

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

 df_spark = spark.createDataFrame(df)

df_spark.printSchema()
df_spark.describe().show()

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()

from pyspark.sql.functions import max, min

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

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

from pyspark.sql.functions import col
```

```
import pyspark.sql.functions as F

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

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()

df_pandas = df_spark.toPandas()

avg_marks = df_pandas[subjects].mean()

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

from pyspark.sql.functions import when, col
from functools import reduce
```

```
df_spark = df_spark.withColumn("Average_Marks",reduce(lambda x, y: x + y, [col(subject) for subject in subjects]) / len(subjects))


df_spark = df_spark.withColumn(
    "Grade",
    when(col("Average_Marks") >= 90, "A")
    .when(col("Average_Marks") >= 80, "B")
    .when(col("Average_Marks") >= 70, "C")
    .when(col("Average_Marks") >= 60, "D")
    .otherwise("Fail")
)


df_spark.select("StudentID", "Name", "Average_Marks", "Grade").show(10)


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)}")


from pyspark.sql.functions import when, col
from functools import reduce


df_spark = df_spark.withColumn("Failed_Subjects",reduce(lambda x, y: x + y, [when(col(subject) < 40, 1).otherwise(0) for subject in subjects]))
```

```
weak_students = df_spark.filter(col("Failed_Subjects") >= 2)

weak_students.select("StudentID", "Name", "Failed_Subjects").show(10)

import matplotlib.pyplot as plt

df_pandas = df_spark.toPandas()

plt.figure(figsize=(10, 5))
plt.hist(df_pandas["Average_Marks"], bins=10, color="skyblue", edgecolor="black")
plt.xlabel("Marks Range")
plt.ylabel("Number of Students")
plt.title("Performance Distribution of Students")
plt.show()

from pyspark.sql.functions import mean

subject_avg = df_spark.select([mean(col(subject)).alias(subject) for subject in subjects])

subject_avg_pandas = subject_avg.toPandas().T
subject_avg_pandas.columns = ["Average_Marks"]

subject_avg_pandas = subject_avg_pandas.sort_values(by="Average_Marks",
ascending=False)

plt.figure(figsize=(10, 5))
plt.bar(subject_avg_pandas.index, subject_avg_pandas["Average_Marks"], color=['blue',
'green', 'red', 'purple', 'orange', 'brown'])
plt.xlabel("Subjects")
```

```
plt.ylabel("Average Marks")
plt.title("Best & Worst Performing Subjects")
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(12, 10))
for i in range(len(subjects)):
    for j in range(i + 1, len(subjects)):
        plt.figure(figsize=(8, 6))
        sns.scatterplot(x=df[subjects[i]], y=df[subjects[j]], hue=df[subjects[i]],
palette='coolwarm')
        plt.title(f'Scatter Plot: {subjects[i]} vs. {subjects[j]}')
        plt.xlabel(f'{subjects[i]} Marks')
        plt.ylabel(f'{subjects[j]} Marks')
        plt.legend(title=f'{subjects[i]} Marks')
        plt.show()

plt.figure(figsize=(8, 6))
sns.heatmap(df[subjects].corr(), annot=True, cmap='viridis', fmt='.2f', linewidths=0.5)
plt.title('Heatmap of Subject Correlations')
plt.show()

from pyspark.sql.functions import count

pass_fail_summary = df_spark.select(
    *[
        (count(when(col(subject) >= 40, subject)) / df_spark.count() *
100).alias(f"{subject}_Pass_Percentage")
        for subject in subjects
```

```
    ]
)

pass_fail_summary.show()

import matplotlib.pyplot as plt

grade_distribution = df_spark.groupBy("Grade").count().toPandas()

plt.figure(figsize=(8, 5))
plt.bar(grade_distribution["Grade"], grade_distribution["count"], color="orange", edgecolor="black")
plt.xlabel("Grade")
plt.ylabel("Number of Students")
plt.title("Distribution of Grades")
plt.show()

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()
```

```
import seaborn as sns

df_pandas = df_spark.select(*subjects).toPandas()

plt.figure(figsize=(10, 6))
sns.boxplot(data=df_pandas)
plt.title("Distribution of Marks Across Subjects")
plt.ylabel("Marks")
plt.xticks(rotation=45)
plt.show()

df_spark.orderBy(col("Average_Marks").desc()).select("StudentID", "Name",
"Average_Marks").show(10)

df_spark.orderBy(col("Average_Marks").asc()).select("StudentID", "Name",
"Average_Marks").show(10)

from pyspark.sql.functions import stddev

stats = df_spark.select(mean(col("Average_Marks")).alias("Mean"),
stddev(col("Average_Marks")).alias("StdDev")).collect()

mean_marks = stats[0]["Mean"]
std_dev = stats[0]["StdDev"]

high_outliers = df_spark.filter(col("Average_Marks") > mean_marks + 2 * std_dev)
low_outliers = df_spark.filter(col("Average_Marks") < mean_marks - 2 * std_dev)

print("Top Outliers (Exceptional Performers):")
```

```
high_outliers.select("StudentID", "Name", "Average_Marks").show(5)


print("Low Outliers (Poor Performers):")
low_outliers.select("StudentID", "Name", "Average_Marks").show(5)


subject_variability = df_spark.select([stddev(col(subject)).alias(subject) for subject in
subjects])


subject_variability.show()
```