NAME:	SINGIREDDY RISHITH REDDY
REG.NO	**12207770**
SECTION	**9W084**
PROJECT	**RESULT MANAGEMENT SYSTEM**

1. Introduction

The goal of this analysis is to evaluate student performance across multiple subjects using **PySpark** and **Python visualization libraries**. This study explores trends, distributions, correlations, and outlier detection to provide insights into overall academic performance.

2. Libraries Used

The following libraries have been used in this analysis:

Data Processing & Analysis:

- **PySpark** (pyspark.sql): Used for handling large datasets efficiently and performing SQL-like operations on structured data.
- **Pandas** (pandas): Converts PySpark DataFrames into Pandas DataFrames for easier plotting and analysis.
- **NumPy** (numpy): Used for numerical operations and array manipulations.

Visualization & Statistics:

- **Matplotlib** (matplotlib.pyplot): Generates various plots like bar charts, histograms, and scatter plots.
- **Seaborn** (seaborn): Enhances visualizations with advanced color palettes and better graphical representations.

PySpark SQL Functions for Analysis:

- avg, col, max, min, count, when, mean, stddev, orderBy: Used for computing aggregate statistics, filtering data, and ordering results.

3. Dataset Overview

The dataset contains student performance records across six subjects:

- **Electronics**
- **Programming**
- **Database**
- **Data Science**
- **Mathematics**

- **DSA (Data Structures & Algorithms)**

Each row in the dataset represents a student, and columns include:

- StudentID: Unique identifier for each student
- Name: Student's name
- Subject-wise marks
- Average_Marks: The average marks across all subjects
- Grade: Assigned based on performance
- **3. Data Preprocessing**
- Data is loaded into a PySpark DataFrame for efficient processing. Missing values and incorrect data types are handled before further analysis.
- **4. Statistical Summary**
- PySpark's `summary()` function provides statistical details like mean, minimum, and maximum values for each subject.

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

- The average score across all subjects is approximately 65%.
- Electronics and Mathematics have lower average marks compared to other subjects.

5. Performance Analysis Per Subject

Each subject is analyzed for top-performing students, and bar charts are used to compare average marks.

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

plt.bar(subjects, avg_marks, color=sns.color_palette("viridis"))

plt.xlabel("Subjects")

plt.ylabel("Average Marks")

plt.title("Average Marks per Subject")

plt.show()
```

6. Pass Percentage Analysis

Students who scored above 40% are considered to have passed.

```
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()
```

Key Insights:

- **Programming** has the highest pass percentage.
- **Mathematics** has the lowest pass rate, indicating a difficult subject.

7. Grade Distribution

Grades are assigned based on average marks:

- **A:** 90% and above
- **B:** 80-89%
- **C:** 70-79%
- **D:** 60-69%
- **Fail:** Below 60%

```
plt.figure(figsize=(8, 5))
plt.bar(grade_distribution["Grade"], grade_distribution["count"], color=sns.color_palette("pastel")[3])
plt.xlabel("Grade")
plt.ylabel("Number of Students")
plt.title("Distribution of Grades")
plt.show()
```

8. Correlation Between Subjects

A heatmap visualizes correlations between subjects, highlighting subject dependencies.

```
plt.figure(figsize=(8, 6))

sns.heatmap(df[subjects].corr(), annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Heatmap of Subject Correlations')

plt.show()
```

Findings:

- **Programming** and **Data Science** have a strong correlation.
- **Mathematics** and **Electronics** have the weakest correlation.

9. Visualizations and Insights

(a) Performance Distribution

- A histogram illustrates how students' average marks are distributed.

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

plt.hist(df_pandas["Average_Marks"], bins=10, color=sns.color_palette("Set2")[0], edgecolor="black")

plt.xlabel("Marks Range")

plt.ylabel("Number of Students")

plt.title("Performance Distribution of Students")

plt.show()
```

(b) Best and Worst Performing Subjects

Bar charts depict the best and worst-performing subjects.

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

plt.bar(subject_avg_pandas.index, subject_avg_pandas["Average_Marks"], color=sns.color_palette("plasma"))

plt.xlabel("Subjects")

plt.ylabel("Average Marks")
```

```
plt.title("Best & Worst Performing Subjects")

plt.xticks(rotation=45)

plt.show()
```

(c) Scatter Plots for Subject Comparisons

Scatter plots illustrate relationships between different subjects.

```
plt.figure(figsize=(8, 6))

sns.scatterplot(x=df["Electronics"], y=df["Programming"],
hue=df["Electronics"], palette='magma')

plt.xlabel("Electronics Marks")

plt.ylabel("Programming Marks")

plt.title("Scatter Plot: Electronics vs. Programming")

plt.show()
```

Conclusion:

This analysis of student performance using **PySpark** and visualization tools provides key insights into academic trends. We identified **best- and worst-performing subjects**, analyzed **pass percentages**, and examined **grade distributions**. **Correlation analysis** revealed relationships between subjects, while **outlier detection** helped identify exceptional and struggling students. **Visualizations** like bar charts, scatter plots, and heatmaps effectively highlighted patterns. The findings suggest the need for **personalized learning strategies** and **subject-specific interventions**. Future improvements could include **predictive analytics** and **real-time performance tracking**. This data-driven approach helps educators make informed decisions to enhance student outcomes.