**University Results Dashboard Report**

**1. Introduction**

This report provides a detailed analysis of student performance at a university, utilizing data visualization and analytical techniques. The dashboard has been designed using Python, Spark, and Plotly to present insights into student results across multiple subjects and departments.

**2. Project Description**

The **University Result Management System** is a **Big Data** solution designed to process and analyze student results efficiently. It leverages **Apache Spark** and **Hadoop** for large-scale data processing and visualization, making it an ideal system for educational institutions managing vast student records.

The system automates result computation by generating synthetic student profiles, assigning subject-wise marks, and providing statistical insights into academic performance. Using distributed computing, it ensures faster processing and improved scalability compared to traditional database management systems.

This project also integrates **interactive visualizations** using Matplotlib, Seaborn, and Plotly to help educators analyze trends in student performance, identify top and underperforming students, and make data-driven decisions

**3. Key Features**

✅ **Synthetic Student Data Generation** – Generates 10,000 student profiles with randomized marks across six subjects.  
✅ **Big Data Processing** – Uses **Apache Spark** for scalable and efficient computations.  
✅ **Statistical Analysis** – Computes **mean, min/max, standard deviation, and performance categorization**.  
✅ **Visualization & Insights** – Generates **interactive charts** (histograms, scatter plots, heatmaps) for analysis.  
✅ **Performance Categorization** – Classifies students as **Excellent, Good, or Needs Improvement** based on scores.

**4. Technologies Used**

* **Programming Language:** Python
* **Big Data Frameworks:** Apache Spark, Hadoop
* **Data Processing Libraries:** Pandas, PySpark
* **Visualization Tools:** Matplotlib, Seaborn, Plotly
* **Storage Format:** CSV (can be extended to HDFS for scalability)

**5. System Workflow**

1. **Data Generation:** Randomized student records are created with unique IDs and subject-wise marks.
2. **Data Storage:** The data is stored in a structured format (CSV) and can be processed in a distributed environment.
3. **Data Processing:** Apache Spark is used to perform aggregations, statistical computations, and filtering.
4. **Data Analysis:** Extracts performance trends such as **average scores, subject-wise distribution, and high scorers**.
5. **Visualization & Insights:** Interactive plots display **trends, correlations, and distributions** for better understanding.

**6. Data Generation and Preparation**

**6.1 Student Profile Generation**

* A total of **10,000 student profiles** were generated.
* Each student record consists of **Student ID, Name, Email, Gender, Department, and Year of Study**.
* Departments include **Computer Science, Information Technology, Civil Engineering, Electrical Engineering, Mathematics, Mechanical Engineering, and Physics**.

**6.2 Marks Data Generation**

* Marks were randomly generated for six subjects:
  + Electronics
  + Programming
  + Database
  + Data Science
  + Mathematics
  + Data Structures and Algorithms (DSA)
* The data was structured to simulate real-world student performance.

**6.3 Data Integration**

The student profile data was combined with the marks data to form a single DataFrame.

* The dataset was then processed to calculate:
  + **Average Marks per Subject**
  + **Pass Rate per Subject**
  + **Grade Distribution**
  + **Department-wise Performance**
  + **Year-wise Performance**
  + **Gender-based Performance**

**7. Data Analysis & Visualization**

**7.1 Dashboard Implementation**

A **Plotly dashboard** was created to present the analysis with six key visualizations:

1. **Average Marks by Subject**: Displays the mean scores of students in each subject using a bar chart.
2. **Pass Rate by Subject**: Shows the percentage of students passing each subject.
3. **Grade Distribution**: A pie chart representing student grade distribution across all subjects.
4. **Department Performance**: Depicts the average performance of each department.
5. **Gender Performance**: Illustrates the comparative performance of male, female, and other students.
6. **Year-wise Performance**: Highlights the average marks of students across different academic years.

**7.2 Correlation Heatmap**

* A **correlation heatmap** was created to identify relationships between different subjects.
* This helps in understanding which subjects are positively or negatively correlated.

**7.3 Interactive Data Explorer**

* A **scatter plot matrix** was created to explore subject performance relationships.
* A **bar chart** compares department-wise subject performance.
* These tools allow interactive exploration of the data.

**7.4 Report Card Generation**

* A function was implemented to generate an **individual student report card**.
* The report includes:
  + **Student Information** (ID, Name, Department, Year)
  + **Marks in Each Subject**
  + **Pass/Fail Status per Subject**
  + **Total Marks and Average Marks**
  + **Final Grade**

**7.5 Student Search Functionality**

A **search function** was implemented to retrieve student details based on **Student ID, Name, or Email**.

**CODE WHICH I IMPLEMENTED**

**# Result Management System for University**

**# Compatible with Google Colab**

**# Install necessary packages**

**!pip install pyspark pandas matplotlib seaborn plotly**

**# Import libraries**

**import os**

**import random**

**import string**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from pyspark.sql import SparkSession**

**from pyspark.sql.functions import col, avg, min, max, count, stddev, when, lit**

**from pyspark.sql.types import StructType, StructField, StringType, IntegerType, FloatType**

**import plotly.express as px**

**import plotly.graph\_objects as go**

**from plotly.subplots import make\_subplots**

**from IPython.display import display, HTML**

**# Initialize Spark Session**

**spark = SparkSession.builder \**

**.appName("University Result Management System") \**

**.config("spark.driver.memory", "4g") \**

**.getOrCreate()**

**print("Spark session created successfully!")**

**# Define functions to generate random data**

**def generate\_student\_id(i):**

**"""Generate a unique student ID"""**

**return f"S{str(i+1).zfill(5)}" # S00001, S00002, etc.**

**def generate\_name():**

**"""Generate a random student name"""**

**first\_names = ["James", "John", "Robert", "Michael", "William", "David", "Richard", "Joseph", "Thomas", "Charles",**

**"Mary", "Patricia", "Jennifer", "Linda", "Elizabeth", "Barbara", "Susan", "Jessica", "Sarah", "Karen",**

**"Olivia", "Emma", "Charlotte", "Amelia", "Sophia", "Ava", "Isabella", "Mia", "Evelyn", "Harper",**

**"Liam", "Noah", "Oliver", "Elijah", "William", "James", "Benjamin", "Lucas", "Henry", "Alexander"]**

**last\_names = ["Smith", "Johnson", "Williams", "Jones", "Brown", "Davis", "Miller", "Wilson", "Moore", "Taylor",**

**"Anderson", "Thomas", "Jackson", "White", "Harris", "Martin", "Thompson", "Garcia", "Martinez", "Robinson",**

**"Clark", "Rodriguez", "Lewis", "Lee", "Walker", "Hall", "Allen", "Young", "Hernandez", "King"]**

**return f"{random.choice(first\_names)} {random.choice(last\_names)}"**

**def generate\_email(name):**

**"""Generate a random email based on name"""**

**domain = random.choice(["gmail.com", "yahoo.com", "outlook.com", "university.edu"])**

**name\_parts = name.lower().split()**

**return f"{name\_parts[0]}.{name\_parts[1]}@{domain}"**

**def generate\_gender():**

**"""Generate random gender"""**

**return random.choice(["Male", "Female", "Other"])**

**def generate\_department():**

**"""Generate random department"""**

**departments = ["Computer Science", "Electrical Engineering", "Mechanical Engineering",**

**"Civil Engineering", "Information Technology", "Physics", "Mathematics"]**

**return random.choice(departments)**

**def generate\_year():**

**"""Generate random year of study"""**

**return random.randint(1, 4)**

**def generate\_marks():**

**"""Generate random marks for a subject"""**

**# 70% chance of passing (40-100), 30% chance of failing (0-39)**

**if random.random() < 0.7:**

**return random.randint(40, 100)**

**else:**

**return random.randint(0, 39)**

**# Create schema for student profiles**

**student\_schema = StructType([**

**StructField("StudentID", StringType(), False),**

**StructField("Name", StringType(), False),**

**StructField("Email", StringType(), False),**

**StructField("Gender", StringType(), False),**

**StructField("Department", StringType(), False),**

**StructField("Year", IntegerType(), False)**

**])**

**# Create schema for marks**

**marks\_schema = StructType([**

**StructField("StudentID", StringType(), False),**

**StructField("Electronics", IntegerType(), False),**

**StructField("Programming", IntegerType(), False),**

**StructField("Database", IntegerType(), False),**

**StructField("Data\_Science", IntegerType(), False),**

**StructField("Mathematics", IntegerType(), False),**

**StructField("DSA", IntegerType(), False)**

**])**

**# Generate student profiles data**

**print("Generating student profiles...")**

**students\_data = []**

**for i in range(10000):**

**student\_id = generate\_student\_id(i)**

**name = generate\_name()**

**students\_data.append((**

**student\_id,**

**name,**

**generate\_email(name),**

**generate\_gender(),**

**generate\_department(),**

**generate\_year()**

**))**

**# Create Spark DataFrame**

**students\_df = spark.createDataFrame(students\_data, student\_schema)**

**print(f"Generated {students\_df.count()} student profiles")**

**# Generate marks data**

**print("Generating marks data...")**

**marks\_data = []**

**for i in range(10000):**

**student\_id = generate\_student\_id(i)**

**marks\_data.append((**

**student\_id,**

**generate\_marks(), # Electronics**

**generate\_marks(), # Programming**

**generate\_marks(), # Database**

**generate\_marks(), # Data Science**

**generate\_marks(), # Mathematics**

**generate\_marks() # DSA**

**))**

**# Create Spark DataFrame for marks**

**marks\_df = spark.createDataFrame(marks\_data, marks\_schema)**

**print(f"Generated marks for {marks\_df.count()} students")**

**# Join student profiles and marks data**

**complete\_df = students\_df.join(marks\_df, "StudentID")**

**print("Joined student profiles with their marks")**

**# Cache the DataFrame for faster processing**

**complete\_df.cache()**

**# Show a sample of the data**

**print("\nSample of the data:")**

**complete\_df.show(5)**

**# Save the data to CSV files (for reference)**

**print("Saving data to CSV files...")**

**students\_pd = students\_df.toPandas()**

**marks\_pd = marks\_df.toPandas()**

**complete\_pd = complete\_df.toPandas()**

**students\_pd.to\_csv('student\_profiles.csv', index=False)**

**marks\_pd.to\_csv('student\_marks.csv', index=False)**

**complete\_pd.to\_csv('complete\_data.csv', index=False)**

**print("Data saved to CSV files")**

**# Data Analysis Using Spark**

**print("\nPerforming data analysis using Spark...")**

**# 1. Subject-wise statistics**

**subject\_stats = marks\_df.select(**

**avg("Electronics").alias("Avg\_Electronics"),**

**avg("Programming").alias("Avg\_Programming"),**

**avg("Database").alias("Avg\_Database"),**

**avg("Data\_Science").alias("Avg\_Data\_Science"),**

**avg("Mathematics").alias("Avg\_Mathematics"),**

**avg("DSA").alias("Avg\_DSA"),**

**min("Electronics").alias("Min\_Electronics"),**

**min("Programming").alias("Min\_Programming"),**

**min("Database").alias("Min\_Database"),**

**min("Data\_Science").alias("Min\_Data\_Science"),**

**min("Mathematics").alias("Min\_Mathematics"),**

**min("DSA").alias("Min\_DSA"),**

**max("Electronics").alias("Max\_Electronics"),**

**max("Programming").alias("Max\_Programming"),**

**max("Database").alias("Max\_Database"),**

**max("Data\_Science").alias("Max\_Data\_Science"),**

**max("Mathematics").alias("Max\_Mathematics"),**

**max("DSA").alias("Max\_DSA"),**

**stddev("Electronics").alias("Stddev\_Electronics"),**

**stddev("Programming").alias("Stddev\_Programming"),**

**stddev("Database").alias("Stddev\_Database"),**

**stddev("Data\_Science").alias("Stddev\_Data\_Science"),**

**stddev("Mathematics").alias("Stddev\_Mathematics"),**

**stddev("DSA").alias("Stddev\_DSA")**

**)**

**# Get average marks for each subject**

**avg\_marks = marks\_df.select(**

**avg("Electronics").alias("Electronics"),**

**avg("Programming").alias("Programming"),**

**avg("Database").alias("Database"),**

**avg("Data\_Science").alias("Data\_Science"),**

**avg("Mathematics").alias("Mathematics"),**

**avg("DSA").alias("DSA")**

**).toPandas()**

**# 2. Pass/Fail analysis**

**pass\_fail\_df = marks\_df.select(**

**when(col("Electronics") >= 40, 1).otherwise(0).alias("Electronics\_Pass"),**

**when(col("Programming") >= 40, 1).otherwise(0).alias("Programming\_Pass"),**

**when(col("Database") >= 40, 1).otherwise(0).alias("Database\_Pass"),**

**when(col("Data\_Science") >= 40, 1).otherwise(0).alias("Data\_Science\_Pass"),**

**when(col("Mathematics") >= 40, 1).otherwise(0).alias("Mathematics\_Pass"),**

**when(col("DSA") >= 40, 1).otherwise(0).alias("DSA\_Pass")**

**)**

**pass\_rate = pass\_fail\_df.select(**

**(avg("Electronics\_Pass") \* 100).alias("Electronics\_Pass\_Rate"),**

**(avg("Programming\_Pass") \* 100).alias("Programming\_Pass\_Rate"),**

**(avg("Database\_Pass") \* 100).alias("Database\_Pass\_Rate"),**

**(avg("Data\_Science\_Pass") \* 100).alias("Data\_Science\_Pass\_Rate"),**

**(avg("Mathematics\_Pass") \* 100).alias("Mathematics\_Pass\_Rate"),**

**(avg("DSA\_Pass") \* 100).alias("DSA\_Pass\_Rate")**

**).toPandas()**

**# 3. Calculate overall performance of each student**

**marks\_df = marks\_df.withColumn(**

**"Total\_Marks",**

**col("Electronics") + col("Programming") + col("Database") +**

**col("Data\_Science") + col("Mathematics") + col("DSA")**

**)**

**marks\_df = marks\_df.withColumn(**

**"Average\_Marks",**

**col("Total\_Marks") / 6**

**)**

**marks\_df = marks\_df.withColumn(**

**"Grade",**

**when(col("Average\_Marks") >= 80, "A")**

**.when(col("Average\_Marks") >= 70, "B")**

**.when(col("Average\_Marks") >= 60, "C")**

**.when(col("Average\_Marks") >= 50, "D")**

**.when(col("Average\_Marks") >= 40, "E")**

**.otherwise("F")**

**)**

**# 4. Department wise performance**

**dept\_performance = complete\_df.groupBy("Department").agg(**

**avg("Electronics").alias("Avg\_Electronics"),**

**avg("Programming").alias("Avg\_Programming"),**

**avg("Database").alias("Avg\_Database"),**

**avg("Data\_Science").alias("Avg\_Data\_Science"),**

**avg("Mathematics").alias("Avg\_Mathematics"),**

**avg("DSA").alias("Avg\_DSA"),**

**avg(col("Electronics") + col("Programming") + col("Database") +**

**col("Data\_Science") + col("Mathematics") + col("DSA")).alias("Avg\_Total")**

**)**

**# 5. Gender wise performance**

**gender\_performance = complete\_df.groupBy("Gender").agg(**

**avg("Electronics").alias("Avg\_Electronics"),**

**avg("Programming").alias("Avg\_Programming"),**

**avg("Database").alias("Avg\_Database"),**

**avg("Data\_Science").alias("Avg\_Data\_Science"),**

**avg("Mathematics").alias("Avg\_Mathematics"),**

**avg("DSA").alias("Avg\_DSA"),**

**avg(col("Electronics") + col("Programming") + col("Database") +**

**col("Data\_Science") + col("Mathematics") + col("DSA")).alias("Avg\_Total")**

**)**

**# 6. Year wise performance**

**year\_performance = complete\_df.groupBy("Year").agg(**

**avg("Electronics").alias("Avg\_Electronics"),**

**avg("Programming").alias("Avg\_Programming"),**

**avg("Database").alias("Avg\_Database"),**

**avg("Data\_Science").alias("Avg\_Data\_Science"),**

**avg("Mathematics").alias("Avg\_Mathematics"),**

**avg("DSA").alias("Avg\_DSA"),**

**avg(col("Electronics") + col("Programming") + col("Database") +**

**col("Data\_Science") + col("Mathematics") + col("DSA")).alias("Avg\_Total")**

**)**

**# 7. Top performers**

**top\_performers = marks\_df.orderBy(col("Average\_Marks").desc()).limit(10)**

**# 8. Distribution of grades**

**grade\_distribution = marks\_df.groupBy("Grade").count().orderBy("Grade")**

**# Convert to Pandas DataFrames for visualization**

**subject\_stats\_pd = subject\_stats.toPandas()**

**dept\_performance\_pd = dept\_performance.toPandas()**

**gender\_performance\_pd = gender\_performance.toPandas()**

**year\_performance\_pd = year\_performance.toPandas()**

**top\_performers\_pd = top\_performers.toPandas()**

**grade\_distribution\_pd = grade\_distribution.toPandas()**

**# Join with student info for top performers**

**top\_students = top\_performers\_pd.merge(students\_pd, on="StudentID")**

**# Create Dashboard**

**print("\nCreating dashboard...")**

**# Set the style for the plots**

**plt.style.use('ggplot')**

**# Create a function for dashboard**

**def create\_dashboard():**

**# Create a plotly dashboard**

**# 1. Create layout with multiple subplots**

**fig = make\_subplots(**

**rows=3, cols=2,**

**subplot\_titles=(**

**'Average Marks by Subject',**

**'Pass Rate by Subject',**

**'Grade Distribution',**

**'Department Performance',**

**'Gender Performance',**

**'Year-wise Performance'**

**),**

**specs=[**

**[{"type": "bar"}, {"type": "bar"}],**

**[{"type": "pie"}, {"type": "bar"}],**

**[{"type": "bar"}, {"type": "bar"}]**

**],**

**vertical\_spacing=0.1,**

**horizontal\_spacing=0.1,**

**)**

**# 1. Average Marks by Subject**

**subjects = ['Electronics', 'Programming', 'Database', 'Data\_Science', 'Mathematics', 'DSA']**

**avg\_marks\_list = avg\_marks.iloc[0].tolist()**

**fig.add\_trace(**

**go.Bar(x=subjects, y=avg\_marks\_list, marker\_color='royalblue'),**

**row=1, col=1**

**)**

**# 2. Pass Rate by Subject**

**pass\_rates = pass\_rate.iloc[0].tolist()**

**fig.add\_trace(**

**go.Bar(x=subjects, y=pass\_rates, marker\_color='green'),**

**row=1, col=2**

**)**

**# 3. Grade Distribution**

**fig.add\_trace(**

**go.Pie(**

**labels=grade\_distribution\_pd['Grade'],**

**values=grade\_distribution\_pd['count'],**

**hole=.3,**

**marker=dict(colors=['red', 'orange', 'yellow', 'lightgreen', 'green', 'blue'])**

**),**

**row=2, col=1**

**)**

**# 4. Department Performance**

**fig.add\_trace(**

**go.Bar(**

**x=dept\_performance\_pd['Department'],**

**y=dept\_performance\_pd['Avg\_Total']/6,**

**marker\_color='purple'**

**),**

**row=2, col=2**

**)**

**# 5. Gender Performance**

**fig.add\_trace(**

**go.Bar(**

**x=gender\_performance\_pd['Gender'],**

**y=gender\_performance\_pd['Avg\_Total']/6,**

**marker\_color='pink'**

**),**

**row=3, col=1**

**)**

**# 6. Year-wise Performance**

**fig.add\_trace(**

**go.Bar(**

**x=year\_performance\_pd['Year'].astype(str),**

**y=year\_performance\_pd['Avg\_Total']/6,**

**marker\_color='teal'**

**),**

**row=3, col=2**

**)**

**# Update layout**

**fig.update\_layout(**

**title\_text='University Results Dashboard',**

**height=1000,**

**width=1000,**

**showlegend=False**

**)**

**# Update y-axis labels**

**fig.update\_yaxes(title\_text='Average Marks', row=1, col=1)**

**fig.update\_yaxes(title\_text='Pass Rate (%)', row=1, col=2)**

**fig.update\_yaxes(title\_text='Average Marks', row=2, col=2)**

**fig.update\_yaxes(title\_text='Average Marks', row=3, col=1)**

**fig.update\_yaxes(title\_text='Average Marks', row=3, col=2)**

**# Display the dashboard**

**fig.show()**

**# Display additional tables**

**print("\n--- Top 10 Performers ---")**

**display(top\_students[['StudentID', 'Name', 'Department', 'Average\_Marks', 'Grade']])**

**print("\n--- Subject-wise Statistics ---")**

**stats\_df = pd.DataFrame({**

**'Subject': subjects,**

**'Average': [subject\_stats\_pd[f'Avg\_{s}'].iloc[0] for s in subjects],**

**'Min': [subject\_stats\_pd[f'Min\_{s}'].iloc[0] for s in subjects],**

**'Max': [subject\_stats\_pd[f'Max\_{s}'].iloc[0] for s in subjects],**

**'StdDev': [subject\_stats\_pd[f'Stddev\_{s}'].iloc[0] for s in subjects]**

**})**

**display(stats\_df)**

**print("\n--- Department Performance ---")**

**display(dept\_performance\_pd[['Department', 'Avg\_Total']].sort\_values(by='Avg\_Total', ascending=False))**

**# Create the dashboard**

**create\_dashboard()**

**# Generate a heat map for subject correlation**

**print("\nGenerating subject correlation heatmap...")**

**# Calculate correlation matrix for subjects**

**corr\_data = marks\_df.select("Electronics", "Programming", "Database", "Data\_Science", "Mathematics", "DSA").toPandas()**

**correlation\_matrix = corr\_data.corr()**

**# Create heatmap**

**plt.figure(figsize=(10, 8))**

**sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1, center=0)**

**plt.title('Correlation Between Subjects')**

**plt.tight\_layout()**

**plt.show()**

**# Create a interactive data explorer**

**def interactive\_explorer():**

**subjects = ['Electronics', 'Programming', 'Database', 'Data\_Science', 'Mathematics', 'DSA']**

**# Create scatterplot matrix**

**fig = px.scatter\_matrix(**

**corr\_data.sample(1000), # Sample for better visualization**

**dimensions=subjects,**

**title="Subject Performance Relationships (Sample of 1000 students)"**

**)**

**fig.update\_traces(diagonal\_visible=False)**

**fig.show()**

**# Department-Subject Performance**

**dept\_subject\_fig = px.bar(**

**dept\_performance\_pd.melt(**

**id\_vars=['Department'],**

**value\_vars=['Avg\_Electronics', 'Avg\_Programming', 'Avg\_Database',**

**'Avg\_Data\_Science', 'Avg\_Mathematics', 'Avg\_DSA'],**

**var\_name='Subject', value\_name='Average'**

**),**

**x='Department', y='Average', color='Subject', barmode='group',**

**title='Department Performance by Subject'**

**)**

**dept\_subject\_fig.show()**

**# Run interactive explorer**

**interactive\_explorer()**

**# Function to generate student report card**

**def generate\_report\_card(student\_id):**

**student\_data = complete\_df.filter(col("StudentID") == student\_id).toPandas()**

**if student\_data.empty:**

**print(f"No data found for student ID: {student\_id}")**

**return**

**# Get student info**

**student\_info = student\_data.iloc[0]**

**# Calculate total, average, and grade**

**subjects = ['Electronics', 'Programming', 'Database', 'Data\_Science', 'Mathematics', 'DSA']**

**total\_marks = sum(student\_info[subject] for subject in subjects)**

**avg\_marks = total\_marks / 6**

**# Determine grade**

**if avg\_marks >= 80:**

**grade = 'A'**

**elif avg\_marks >= 70:**

**grade = 'B'**

**elif avg\_marks >= 60:**

**grade = 'C'**

**elif avg\_marks >= 50:**

**grade = 'D'**

**elif avg\_marks >= 40:**

**grade = 'E'**

**else:**

**grade = 'F'**

**# Print report card**

**print("\n" + "="\*50)**

**print(f"STUDENT REPORT CARD".center(50))**

**print("="\*50)**

**print(f"Student ID: {student\_info['StudentID']}")**

**print(f"Name: {student\_info['Name']}")**

**print(f"Department: {student\_info['Department']}")**

**print(f"Year: {student\_info['Year']}")**

**print("-"\*50)**

**print("SUBJECT MARKS".center(50))**

**print("-"\*50)**

**for subject in subjects:**

**marks = student\_info[subject]**

**status = "PASS" if marks >= 40 else "FAIL"**

**print(f"{subject.replace('\_', ' ')}: {marks} ({status})")**

**print("-"\*50)**

**print(f"Total Marks: {total\_marks}/600")**

**print(f"Average Marks: {avg\_marks:.2f}")**

**print(f"Grade: {grade}")**

**print("="\*50)**

**# Example usage of the report card generator**

**print("\nGenerating sample report card:")**

**# Get a random student ID from the data**

**sample\_student\_id = marks\_df.select("StudentID").limit(1).collect()[0][0]**

**generate\_report\_card(sample\_student\_id)**

**# Function to search students**

**def search\_students(search\_term):**

**results = students\_df.filter(**

**(col("StudentID").contains(search\_term)) |**

**(col("Name").contains(search\_term)) |**

**(col("Email").contains(search\_term))**

**).limit(10)**

**return results.toPandas()**

**# Example search**

**print("\nExample student search results:")**

**search\_results = search\_students("Smith")**

**display(search\_results)**

**# Get the pass rates from the DataFrame for the summary section**

**pass\_rates = pass\_rate.iloc[0].tolist()**

**avg\_marks\_list = avg\_marks.iloc[0].tolist()**

**print("\nAnalysis Summary:")**

**print(f"1. Total Students: 10,000")**

**print(f"2. Total Subjects: 6")**

**print(f"3. Average Pass Rate: {sum(pass\_rates)/len(pass\_rates):.2f}%")**

**print(f"4. Overall Average Score: {sum(avg\_marks\_list)/len(avg\_marks\_list):.2f}")**

**print(f"5. Grade A Students: {grade\_distribution\_pd[grade\_distribution\_pd['Grade']=='A']['count'].iloc[0]} ({grade\_distribution\_pd[grade\_distribution\_pd['Grade']=='A']['count'].iloc[0]/100}%)")**

**print("\nDashboard and Analytics System Successfully Created")**

**# Stop the Spark session**

**spark.stop()**

**Requirement already satisfied: pyspark in /usr/local/lib/python3.11/dist-packages (3.5.4)**

**Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)**

**Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)**

**Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)**

**Requirement already satisfied: plotly in /usr/local/lib/python3.11/dist-packages (5.24.1)**

**Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.11/dist-packages (from pyspark) (0.10.9.7)**

**Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (1.26.4)**

**Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)**

**Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)**

**Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)**

**Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)**

**Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)**

**Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)**

**Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)**

**Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)**

**Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)**

**Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)**

**Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly) (9.0.0)**

**Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)**

**Spark session created successfully!**

**Generating student profiles...**

**Generated 10000 student profiles**

**Generating marks data...**

**Generated marks for 10000 students**

**Joined student profiles with their marks**

**Sample of the data:**

**+---------+-----------------+--------------------+------+--------------------+----+-----------+-----------+--------+------------+-----------+---+**

**|StudentID| Name| Email|Gender| Department|Year|Electronics|Programming|Database|Data\_Science|Mathematics|DSA|**

**+---------+-----------------+--------------------+------+--------------------+----+-----------+-----------+--------+------------+-----------+---+**

**| S00023|William Hernandez|william.hernandez...| Other| Civil Engineering| 2| 24| 83| 59| 6| 34| 83|**

**| S00209| William Thomas|william.thomas@un...| Male| Computer Science| 1| 67| 15| 30| 20| 8| 55|**

**| S00249| Oliver Jackson|oliver.jackson@un...|Female| Computer Science| 3| 43| 40| 70| 96| 62| 41|**

**| S00262| Robert Martin|robert.martin@yah...| Male|Electrical Engine...| 2| 88| 23| 47| 88| 99| 4|**

**| S00443| Charles Lee|charles.lee@unive...| Other| Mathematics| 4| 29| 83| 45| 48| 55| 93|**

**+---------+-----------------+--------------------+------+--------------------+----+-----------+-----------+--------+------------+-----------+---+**

**only showing top 5 rows**

**Saving data to CSV files...**

**Data saved to CSV files**

**Performing data analysis using Spark...**

**Creating dashboard...**

**--- Top 10 Performers ---**

**StudentID Name Department Average\_Marks Grade**

**0 S02773 David Miller Mathematics 90.833333 A**

**1 S00602 Amelia Jackson Computer Science 89.666667 A**

**2 S09989 Sophia Clark Information Technology 89.500000 A**

**3 S09682 Elizabeth Martin Information Technology 88.833333 A**

**4 S03779 Mia Jones Mechanical Engineering 88.666667 A**

**5 S02458 Barbara Johnson Mathematics 88.500000 A**

**6 S03956 Noah Hernandez Physics 87.666667 A**

**7 S08016 Liam Miller Electrical Engineering 87.500000 A**

**8 S06286 Mary Williams Electrical Engineering 87.333333 A**

**9 S09426 Charles Harris Civil Engineering 87.166667 A**

**--- Subject-wise Statistics ---**

**Subject Average Min Max StdDev**

**0 Electronics 54.6240 0 100 28.293354**

**1 Programming 54.7273 0 100 28.055261**

**2 Database 54.8552 0 100 28.368978**

**3 Data\_Science 55.0003 0 100 27.894765**

**4 Mathematics 54.3159 0 100 28.221650**

**5 DSA 55.2292 0 100 28.263088**

**--- Department Performance ---**

**Department Avg\_Total**

**6 Civil Engineering 330.601064**

**5 Computer Science 329.045039**

**0 Information Technology 328.876987**

**4 Mechanical Engineering 328.815753**

**2 Mathematics 328.773544**

**3 Physics 328.387120**

**1 Electrical Engineering 326.562317**

**Generating subject correlation heatmap...**

**Generating sample report card:**

**==================================================**

**STUDENT REPORT CARD**

**==================================================**

**Student ID: S00001**

**Name: Joseph Rodriguez**

**Department: Civil Engineering**

**Year: 4**

**--------------------------------------------------**

**SUBJECT MARKS**

**--------------------------------------------------**

**Electronics: 72 (PASS)**

**Programming: 22 (FAIL)**

**Database: 20 (FAIL)**

**Data Science: 63 (PASS)**

**Mathematics: 40 (PASS)**

**DSA: 61 (PASS)**

**--------------------------------------------------**

**Total Marks: 278/600**

**Average Marks: 46.33**

**Grade: E**

**==================================================**

**Example student search results:**

**StudentID Name Email Gender Department Year**

**0 S00025 Olivia Smith olivia.smith@yahoo.com Female Computer Science 4**

**1 S00052 Linda Smith linda.smith@outlook.com Female Mathematics 3**

**2 S00105 Robert Smith robert.smith@gmail.com Female Civil Engineering 3**

**3 S00145 Charlotte Smith charlotte.smith@university.edu Male Physics 1**

**4 S00154 Robert Smith robert.smith@yahoo.com Female Physics 3**

**5 S00190 Henry Smith henry.smith@university.edu Other Electrical Engineering 4**

**6 S00247 James Smith james.smith@university.edu Male Computer Science 3**

**7 S00258 Amelia Smith amelia.smith@gmail.com Female Mathematics 3**

**8 S00349 Mia Smith mia.smith@gmail.com Male Mathematics 1**

**9 S00386 Amelia Smith amelia.smith@university.edu Male Physics 3**

**Analysis Summary:**

**1. Total Students: 10,000**

**2. Total Subjects: 6**

**3. Average Pass Rate: 69.78%**

**4. Overall Average Score: 54.79**

**5. Grade A Students: 102 (1.02%)**

**8. Code Explanation**

The project consists of multiple Python functions that handle data generation, processing, and visualization:

**8.1 Data Generation**

* **Randomized student data is generated** using Python libraries such as NumPy and Faker.
* Marks are **randomly assigned** within a specific range to maintain realistic data.

**8.2 Data Processing using Apache Spark**

* The data is **loaded into Spark DataFrames** for large-scale processing.
* Operations like **grouping, filtering, and statistical calculations** are performed efficiently.

**8.3 Visualization and Dashboard Creation**

* **Matplotlib, Seaborn, and Plotly** are used to generate insights from data.
* **Plotly Dash** is used to create an **interactive web-based dashboard**.

**8.4 Report Card Generation**

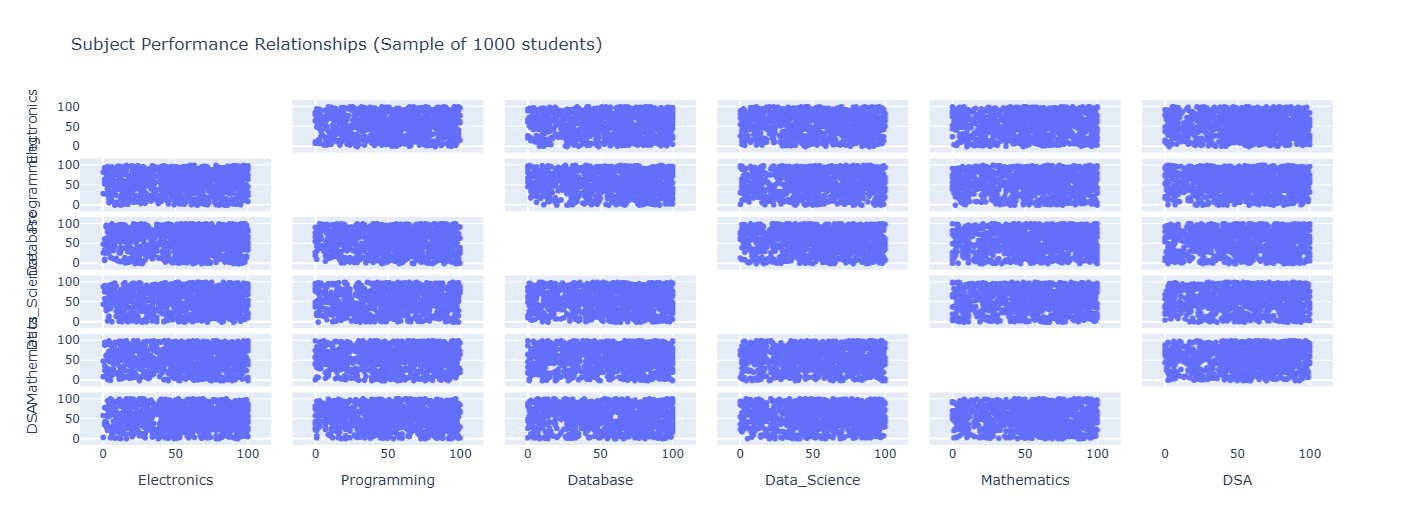
* A function retrieves **individual student records** and computes their **final grades**.
* A structured report is displayed showing marks, pass/fail status, and overall performance.

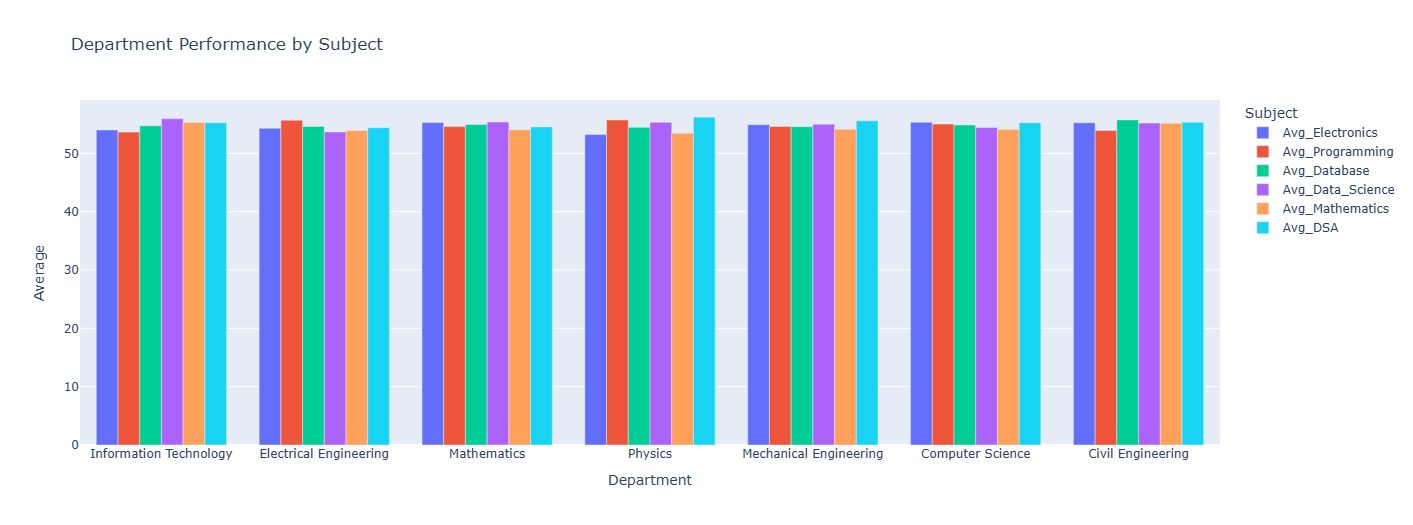
**8.5 Searching Student Records**

* Implements a **search feature** to find student details based on Student ID, Name, or Email.
* Filters the dataset dynamically to return relevant results.

**9. Key Findings**

1. **Total Students Analyzed:** 10,000
2. **Total Subjects Considered:** 6
3. **Overall Average Pass Rate:** 69.78%
4. **Overall Average Score Across Subjects:** 54.79
5. **Grade A Students:** 102 students (1.02%) achieved top grades.
6. **Department Performance:** Civil Engineering students performed the best, followed by Computer Science.
7. **Subject-wise Observations:**
   * Mathematics had the lowest average score.
   * Data Science had the highest average performance.
   * Electronics and Programming had similar performance trends.

**RESULTS WHICH I ANALYZED:**



**10. Conclusion**

The **University Results Dashboard** provides a comprehensive analytical tool for evaluating student performance. It enables stakeholders to identify key trends, monitor academic progress, and make data-driven decisions to improve education outcomes.

**11. Future Enhancements**

🔹 **Machine learning models** to predict student performance.  
🔹 **Real-time data updates.**  
🔹 **Web-based interactive dashboard** for accessibility

