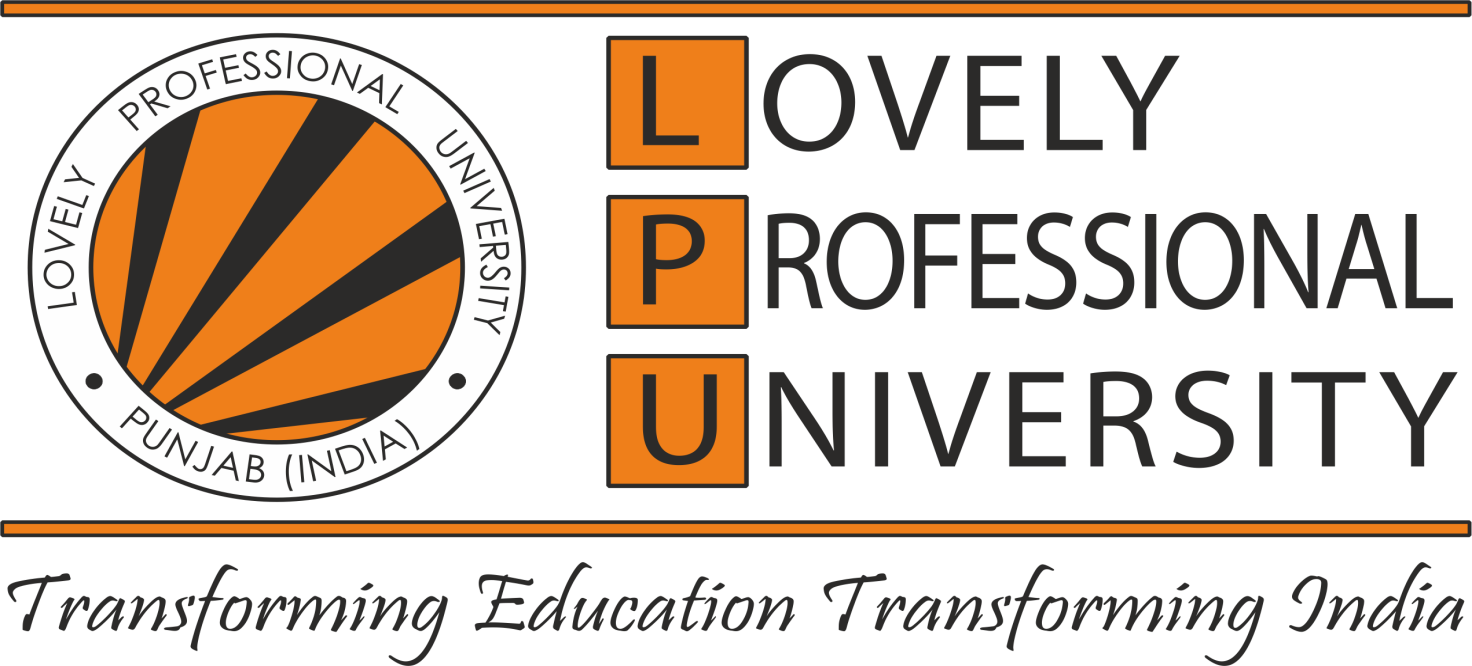
[**Big Data Project - Mentored by Industry SDE**](https://www.gradskey.com/discussions/opinion/291)

**Title:- RESULT MANAGEMENT SYSTEM.(Report)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. No. | Registration No | Section | Name of Students |  |
| 1 | 12206662 | 9W084 | Mohammed Umer Khan |  |



### **Submitted To Gradskey.**

Lovely Professional University

Jalandhar, Punjab, India

**1 .Introduction**

**Project Name:**

Result Management System

**Objective:**

The objective of this project is to develop a comprehensive **Result Management System** that efficiently manages student records, processes marks using **Big Data technologies**, and provides insightful **visualizations** to analyze student performance.

**Technology Stack:**

* **Data Processing:** Apache Spark, Hadoop (MapReduce)
* **Database:** MySQL, PostgreSQL, or NoSQL (MongoDB)
* **Visualization:** Tableau, Power BI, Matplotlib
* **Backend:** Python (Flask)
* **Frontend:** React.js / HTML & CSS

**2. Project Planning & Setup**

**Define Scope:**

* Generate **10,000 student records** with relevant details
* Assign **random marks** to students across six subjects
* Perform **data analysis** and **visualization**

**Setup Environment:**

* Install all **required libraries and frameworks**
* Set up a **virtual environment** for dependency management

**3. Generating Student Profiles**

* Use **Faker Library** to create **synthetic student data** such as:
  + Student ID
  + Name
  + Department
  + Email
* Store generated student data in **CSV format** for further processing

from faker import Faker

import pandas as pd

fake = Faker()

students = []

for i in range(10000):

students.append({

"Student\_ID": i + 1,

"Name": fake.name(),

"Age": fake.random\_int(min=18, max=25),

"Department": fake.random\_element(["CSE", "ECE", "IT", "ME", "EEE"])

})

df\_students = pd.DataFrame(students)

df\_students.to\_csv("students.csv", index=False)

**4. Assign Subjects & Generate Marks**

* Define **six core subjects**:
  1. Electronics
  2. Programming
  3. Database
  4. Data Science
  5. Mathematics
  6. Data Structures & Algorithms (DSA)
* Assign **random marks** between **30-100** to each student for each subject

import random

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

marks = []

for i in range(10000):

student\_marks = {

"Student\_ID": i + 1,

"Electronics": random.randint(30, 100),

"Programming": random.randint(30, 100),

"Database": random.randint(30, 100),

"Data Science": random.randint(30, 100),

"Mathematics": random.randint(30, 100),

"DSA": random.randint(30, 100),

}

marks.append(student\_marks)

df\_marks = pd.DataFrame(marks)

df\_marks.to\_csv("marks.csv", index=False)

* Store data in structured format (CSV or database)

**5. Processing Data with Spark & Hadoop**

* Load student records into **Apache Spark DataFrame**
* Utilize **MapReduce** or **Spark SQL** for data aggregation, including:
  + **Total Marks Calculation**
  + **Highest & Lowest Scores Identification**
  + **Pass/Fail Status Computation**
* Store results for further analysis

from pyspark.sql import SparkSession

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

# Load data

students\_df = spark.read.csv("students.csv", header=True, inferSchema=True)

marks\_df = spark.read.csv("marks.csv", header=True, inferSchema=True)

# Compute statistics

marks\_df.createOrReplaceTempView("marks")

avg\_marks = spark.sql("SELECT AVG(Electronics) AS Avg\_Electronics, AVG(Programming) AS Avg\_Programming, "

"AVG(Database) AS Avg\_Database, AVG(Data Science) AS Avg\_DataScience, " # Use backticks to enclose column name with space

"AVG(Mathematics) AS Avg\_Mathematics, AVG(DSA) AS Avg\_DSA FROM marks")

avg\_marks.show()

**6. Data Analysis**

* Compute key performance metrics:
  + **Top Performers** (highest total marks)
  + **Average Scores** per subject
  + **Students failing in subjects**
* Use **Pandas and NumPy** for quick data insights

df\_marks["Total"] = df\_marks.iloc[:, 1:].sum(axis=1)

df\_marks["Percentage"] = df\_marks["Total"] / 6

df\_marks["Result"] = df\_marks["Percentage"].apply(lambda x: "Pass" if x >= 40 else "Fail")

# Get insights

print(df\_marks.describe()) # Basic statistics

print(df\_marks[df\_marks["Result"] == "Fail"].count()) # No. of failing students

print(df\_marks.nlargest(100, "Total")) # Top 10 students

**7. Visualization**

* Generate meaningful visualizations using **Matplotlib**:
  + **Bar Charts** for **subject-wise performance**
  + **Line Charts** for **trends in student performance**
  + **Pie Charts** for **pass/fail ratio**
* Optionally, integrate **Power BI** or **Tableau** for advanced analytics

import matplotlib.pyplot as plt

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

df\_marks[["Electronics", "Programming", "Database", "Data Science", "Mathematics", "DSA"]].mean().plot(kind='bar', color='skyblue')

plt.title("Average Marks per Subject")

plt.xlabel("Subjects")

plt.ylabel("Average Marks")

plt.show()

**8. Flask API Development**

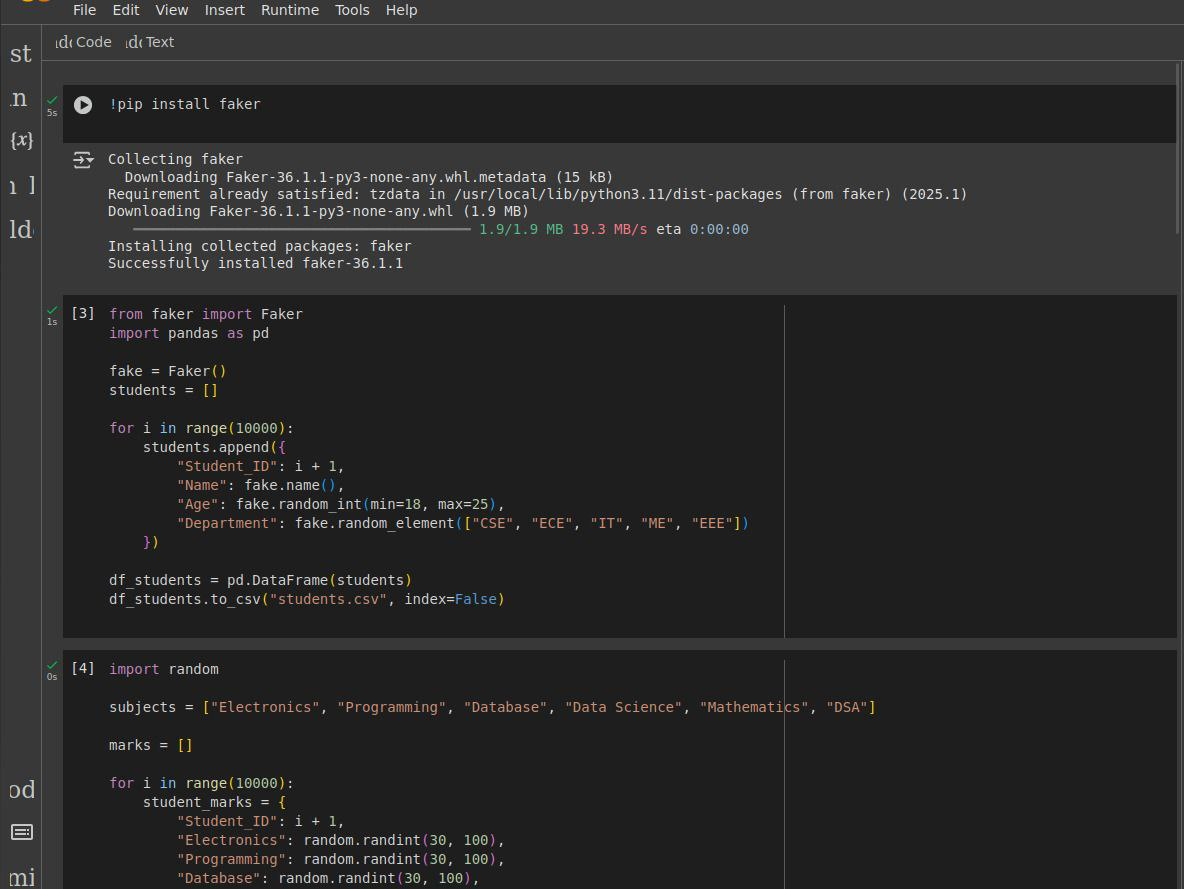
* Develop a **REST API** using Flask to serve student data
* API Endpoints include:
  + Fetch **student details by ID**
  + Fetch **marks of all students**
  + Compute **aggregate performance metrics**
* Deploy the API to be accessible for frontend integration

**9. Deployment**

* Deploy Flask application on **Google-Colab, AWS, or Google Cloud**
* **Containerize** the application using **Docker**
* Ensure scalability and performance optimization

**10. Conclusion & Next Steps**

**Key Achievements:**

* Successfully generated **10,000 student records**
* Processed student marks using **Apache Spark & Hadoop**
* Developed a **Flask API** to manage and serve student data
* Created **data visualizations** using Matplotlib

