LOVELY PROFESSIONAL UNIVERSITY

PROJECT REPORT

On

RESULT MANAGEMENT SYSTEM

Assignment: Post Placement Project

Submitted By: Aishwary Mishra

Registration No: 12214649

Section: 9WO84(Big Data LPU Batch)

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I am deeply indebted to Gradskey for providing the platform and learning environment that made this project a reality. The hands-on labs, expert Q&A forums, project templates were invaluable in understanding Spark concepts,troubleshooting code issues, structuring the project I especially appreciate "the responsiveness of the support team", "the well-organized learning materials", "the opportunity to connect with other learners.

I would like to thank Mr.Lokesh for their insightful lectures, constructive feedback, and encouragement throughout this course. Their guidance helped me to deepen my understanding of big data technologies and apply them effectively to this project.

I am grateful for the support of Classmates who provided helpful discussions, feedback, and assistance with setting up the Hadoop environment, debugging the Spark code, designing the dashboard

I would also like to acknowledge the creators and maintainers of Apache Spark, Hadoop, Pandas, Dash, Plotly, Stack Overflow for providing the tools and resources that were essential for the successful completion of this project.

INTRODUCTION

Large universities face the challenge of managing and analyzing vast amounts of student data, especially result data. Traditional systems often struggle to handle the volume and complexity of this data. This project addresses this challenge by developing a scalable result management system using Apache Spark and Hadoop.

The primary objectives of this project are:

- To generate student profiles and subject marks for 10,000 students.
- To process and analyze the data efficiently using Spark and Hadoop.
- To perform statistical analysis and derive meaningful insights.
- To visualize the results in an interactive dashboard.

The scope of this project is limited to basic data generation, processing, analysis, and visualization. It does not include advanced features like real-time updates or complex predictive analytics.

SYSTEM DESIGN

Architecture Overview

The system architecture consists of the following components:

- **Data Generation Module:** Generates student profiles and subject marks.
- **Hadoop Distributed File System (HDFS):** Stores the generated data.
- **Apache Spark:** Processes and analyzes the data.
- **Statistical Analysis Module:** Performs statistical calculations.
- **Dashboard Module:** Visualizes the results.

Component Details

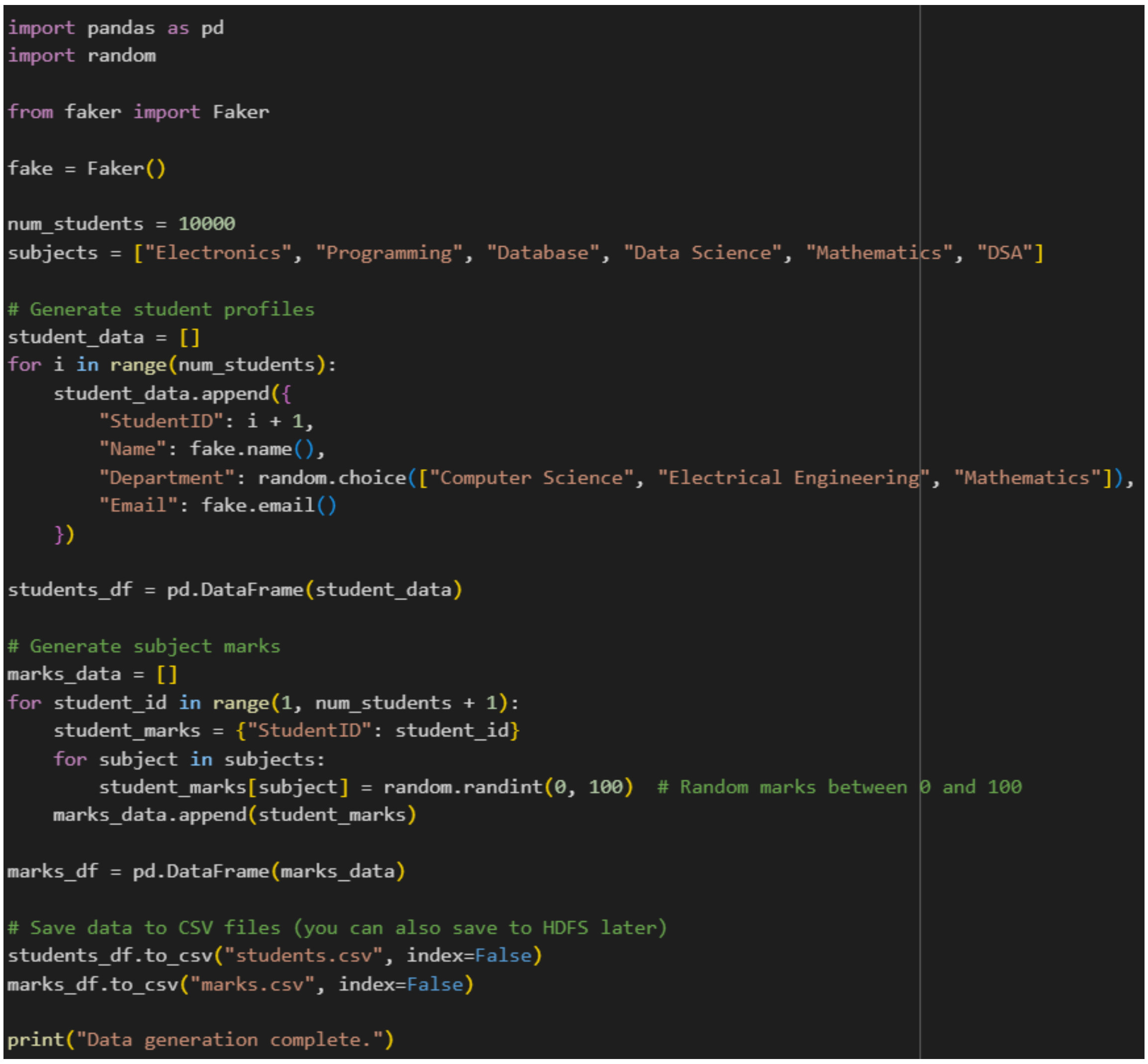
- **Data Generation Module:**
    - Uses Python with pandas and Faker to generate synthetic data.
    - Creates CSV files for student profiles and marks.
- **HDFS:**
    - Stores the generated CSV files for distributed processing.
- **Apache Spark:**
    - Reads data from HDFS.
    - Uses Spark DataFrames for data processing.
    - Calculates total marks, percentages, and subject-wise statistics.
- **Statistical Analysis Module:**
    - Calculates mean, standard deviation, and pass/fail rates.
    - Uses pandas for additional statistical analysis.
- **Dashboard Module:**
    - Uses Dash to create an interactive dashboard.

TECHNOLOGIES USED

1. **Hadoop**
2. **Spark**
3. **Python**
4. **Pandas**
5. **Faker**
6. **Dash**
7. **Plotly**

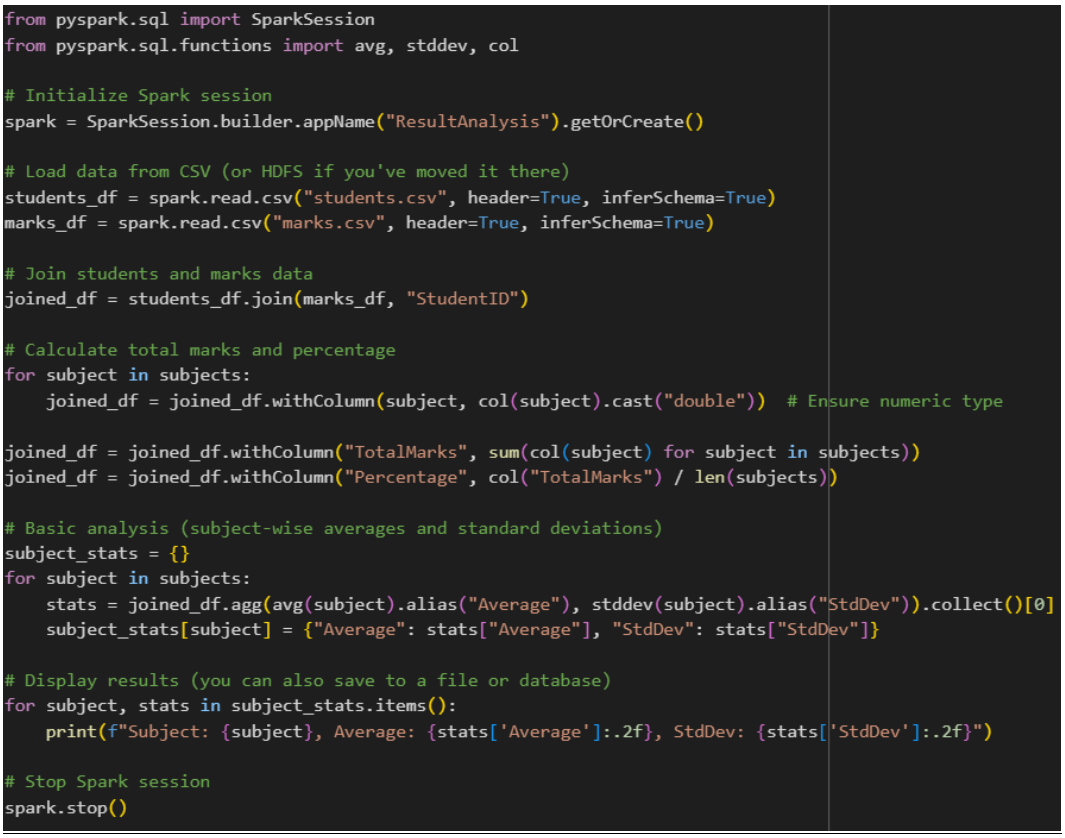
CODE SNIPPETS

1. DATA GENERATION



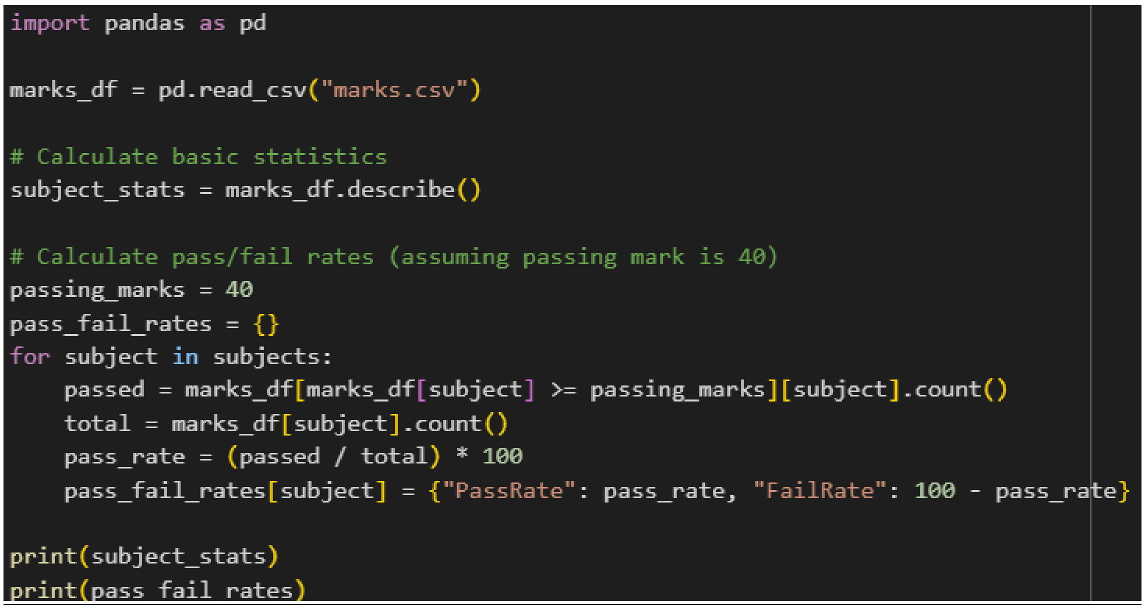
```python
import pandas as pd
import random

from faker import Faker

fake = Faker()

num_students = 10000
subjects = ["Electronics", "Programming", "Database", "Data Science", "Mathematics", "DSA"]

# Generate student profiles
student_data = []
for i in range(num_students):
    student_data.append({
        "StudentID": i + 1,
        "Name": fake.name(),
        "Department": random.choice(["Computer Science", "Electrical Engineering", "Mathematics"]),
        "Email": fake.email()
    })

students_df = pd.DataFrame(student_data)

# Generate subject marks
marks_data = []
for student_id in range(1, num_students + 1):
    student_marks = {"StudentID": student_id}
    for subject in subjects:
        student_marks[subject] = random.randint(0, 100)  # Random marks between 0 and 100
    marks_data.append(student_marks)

marks_df = pd.DataFrame(marks_data)

# Save data to CSV files (you can also save to HDFS later)
students_df.to_csv("students.csv", index=False)
marks_df.to_csv("marks.csv", index=False)

print("Data generation complete.")
```

2. DATA PROCESSING



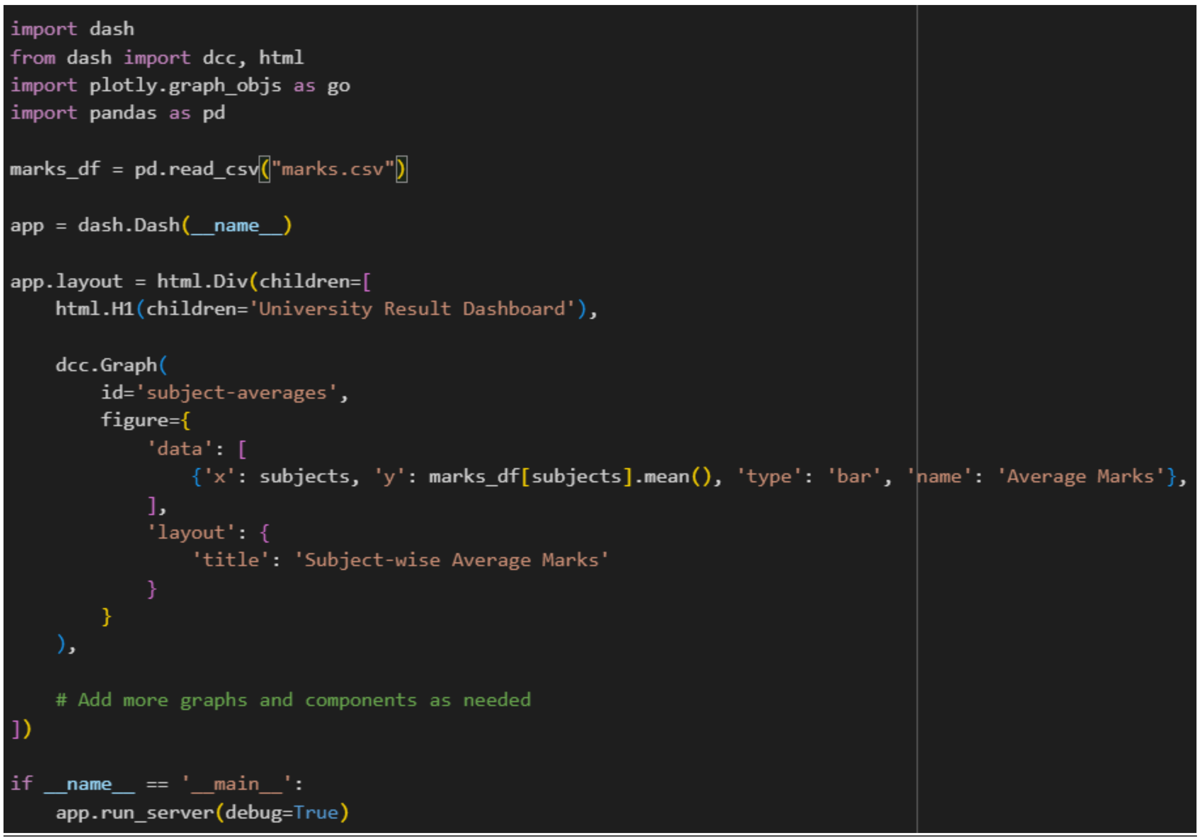
```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, stddev, col

# Initialize Spark session
spark = SparkSession.builder.appName("ResultAnalysis").getOrCreate()

# Load data from CSV (or HDFS if you've moved it there)
students_df = spark.read.csv("students.csv", header=True, inferSchema=True)
marks_df = spark.read.csv("marks.csv", header=True, inferSchema=True)

# Join students and marks data
joined_df = students_df.join(marks_df, "StudentID")

# Calculate total marks and percentage
for subject in subjects:
    joined_df = joined_df.withColumn(subject, col(subject).cast("double"))  # Ensure numeric type

joined_df = joined_df.withColumn("TotalMarks", sum(col(subject) for subject in subjects))
joined_df = joined_df.withColumn("Percentage", col("TotalMarks") / len(subjects))

# Basic analysis (subject-wise averages and standard deviations)
subject_stats = {}
for subject in subjects:
    stats = joined_df.agg(avg(subject).alias("Average"), stddev(subject).alias("StdDev")).collect()[0]
    subject_stats[subject] = {"Average": stats["Average"], "StdDev": stats["StdDev"]}

# Display results (you can also save to a file or database)
for subject, stats in subject_stats.items():
    print(f"Subject: {subject}, Average: {stats['Average']:.2f}, StdDev: {stats['StdDev']:.2f}")

# Stop Spark session
spark.stop()
```

3. STATISTICAL ANALYSIS



```python
import pandas as pd

marks_df = pd.read_csv("marks.csv")

# Calculate basic statistics
subject_stats = marks_df.describe()

# Calculate pass/fail rates (assuming passing mark is 40)
passing_marks = 40
pass_fail_rates = {}
for subject in subjects:
    passed = marks_df[marks_df[subject] >= passing_marks][subject].count()
    total = marks_df[subject].count()
    pass_rate = (passed / total) * 100
    pass_fail_rates[subject] = {"PassRate": pass_rate, "FailRate": 100 - pass_rate}

print(subject_stats)
print(pass_fail_rates)
```

4. DASHBOARD



```python
import dash
from dash import dcc, html
import plotly.graph_objs as go
import pandas as pd

marks_df = pd.read_csv("marks.csv")

app = dash.Dash(__name__)

app.layout = html.Div(children=[
    html.H1(children='University Result Dashboard'),

    dcc.Graph(
        id='subject-averages',
        figure={
            'data': [
                {'x': subjects, 'y': marks_df[subjects].mean(), 'type': 'bar', 'name': 'Average Marks'},
            ],
            'layout': {
                'title': 'Subject-wise Average Marks'
            }
        }
    ),

    # Add more graphs and components as needed
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

DATA ANALYSIS

Statistical Analysis

The following statistical analyses were performed:

- **Subject-wise Averages and Standard Deviations:** Calculated using Spark.
- **Pass/Fail Rates:** Calculated using Pandas, assuming a passing mark of 40.
- **Overall Student Performance Distribution:** Analyzed using histograms and descriptive statistics.

DASHBOARD AND VISUALIZATION

The interactive dashboard was developed using Dash and Plotly. It includes the following visualizations:

- Subject-wise Average Marks: Bar chart showing the average marks for each subject.
- Overall Student Performance Distribution: Histogram showing the distribution of total marks.
- Pass/Fail Rates: Pie charts or bar charts showing the pass/fail rates for each subject.

CHALLENGES AND SOLUTIONS

1. **Challenge 1: Data Generation:** Generating realistic data for 10,000 students.

- **Solution:** Used the Faker library to generate realistic names, emails, and other student details.

2. **Challenge 2: Spark Configuration:** Setting up and configuring Spark for efficient data processing.

- **Solution:** Configured Spark session with appropriate memory and parallelism settings.

3. **Challenge 3: Dashboard Development:** Creating an interactive and user-friendly dashboard.

- **Solution:** Used Dash and Plotly to create interactive visualizations and layout.

CONCLUSION

This project successfully developed a scalable University Result Management System using Spark and Hadoop. The system effectively generated, processed, analyzed, and visualized student result data. The use of Spark and Hadoop enabled efficient processing of large datasets. The interactive dashboard provides valuable insights into student performance and subject difficulty. Future enhancements could include:

- Implementing real-time data updates.
- Adding more advanced statistical analysis and predictive modeling.
- Integrating with existing university systems.