Final Project Report

Movie Rating Analysis Using PySpark with Medallion Architecture

**1. Introduction**

**Project Overview**

The **Movie Rating Analysis** project aims to analyze movie rating data using **PySpark** while leveraging the **Medallion Architecture** for an efficient data processing workflow. The project focuses on loading, cleaning, transforming, and aggregating large-scale movie rating datasets to extract valuable insights.

The **Medallion Architecture** is implemented in three layers:

* **Bronze Layer**: Stores raw movie rating data.
* **Silver Layer**: Cleans and processes the data.
* **Gold Layer**: Aggregates and derives insights.

By using **distributed computing in PySpark**, this project handles large datasets efficiently, making it suitable for big data applications. The final analysis includes **top-rated movies, rating trends over years, and genre-based insights**.

**2. Objectives**

1. **Bronze Layer**:
   * Load and store raw movie rating data.
2. **Silver Layer**:
   * Clean and preprocess data (handle missing values, remove duplicates, etc.).
   * Compute the **average rating** per movie.
3. **Gold Layer**:
   * Identify **top 10 highest-rated movies**.
   * Analyze the **relationship between movie release years and ratings**.
   * Build a recommendation system for the clean data.

**3. Technologies & Tools**

* **PySpark** for data processing and analysis.
* **Fabric Notebook / Python IDE** for writing and executing PySpark code.
* **Power BI** for dashboard visualization.
* **GitHub** for code versioning and documentation.

**4. Data Pipeline & Implementation**

**4.1 Bronze Layer: Data Ingestion**

* Loaded movie ratings dataset (CSV format) into PySpark.
* Stored raw data in a structured table format.

**4.2 Silver Layer: Data Cleaning & Transformation**

* Handled **missing values** and **duplicate records**.
* Converted column data types and standardized naming conventions.
* Computed **average ratings** per movie using groupBy() and agg() functions.
* Performed **release year-wise rating aggregation**.

**4.3 Gold Layer: Data Aggregation & Insights**

* Extracted **Top 10 movies** with the highest ratings.
* Analyzed rating **trends over the years**.
* Build a recommendation system for continuous and categorical data using ALS.
* Generated a **dashboard** using Power BI for interactive visualizations.

**5. Key Insights & Findings**

* **Most popular movies**: Identified top-rated movies based on average ratings.
* **Genre-wise distribution**: Certain genres received higher ratings.
* **Trends in movie ratings**: Older movies had stable ratings, while modern movies had varied ratings.
* **User rating behavior**: A large proportion of ratings were concentrated in mid-range values.

**6. Challenges & Solutions**

| **Challenge** | **Solution** |
| --- | --- |
| Handling large datasets | Used **Spark’s distributed computing** to optimize processing |
| Missing and duplicate data | Applied **data cleansing techniques** in Silver Layer |
| Aggregation complexity | Leveraged **SQL queries in Spark** for optimized computations |
| Visualization of insight | Created **Power BI Dashboard** for interactive reports |

**7. Project Validation & Best Practices**

✅ Created **structured tables** for seamless integration with Power BI.

✅ Maintained **proper code documentation** and uploaded it to GitHub.

✅ Followed **standard naming conventions** in scripts.

✅ Added **comments** to each code snippet for clarity.

**8. Conclusion**

This project demonstrated how **PySpark and Medallion Architecture** can be used to analyze large-scale movie rating datasets efficiently. By leveraging **big data processing techniques**, it provided meaningful insights into user rating behaviors, movie popularity trends, and genre preferences.

The structured approach ensured **clean, organized, and high-performance data processing**, making it suitable for **real-world data analytics applications**.

**Future Scope:**

1. Implement a **movie recommendation system** based on user preferences.
2. Extend analysis with **real-time streaming** data.
3. Optimize the dashboard for dynamic user inputs and advanced filtering.

**Author:** Madhuri Rikame  
**Date:** 05 April 2025