E2E Cricket Analytics Project

Tags	
Done	☐
Due Date	@03/05/2025
Is Completed	0
Status	Not started

Cricket Analytics Project

Project Overview

This project aims to build an **end-to-end cricket analytics platform** that processes both **historical batch data** and **real-time match feeds**. The goal is to ingest, process, store, and analyze cricket match data, providing actionable insights through dashboards.

Project Scope

1. Data Sources

- **Historical Data (Batch Processing)**
  - Past match data (Players, Matches, Ball-by-Ball, Stadiums, Teams, etc.).
  - Ingested from CSV, APIs, or a simulated OLTP database.
- **Live Data (Streaming Processing)**
  - Ball-by-ball match events.
  - Player stats updates (Runs, Wickets, Strike Rate, Economy Rate, etc.).

- Ingested from a simulated API or Kafka producer.

2. Data Ingestion

- **Batch Layer**: Ingests historical match data to a **data lake (Azure Data Lake / S3)**.
- **Streaming Layer**: Uses **Kafka** to handle real-time match feeds.

3. Data Processing

Batch Processing (ETL in Spark)

- Extracts, transforms, and loads historical match data.
- Cleanses data and applies **Medallion Architecture (Bronze → Silver → Gold)**.
- Stores processed data in **Snowflake** for analytics.

Real-time Streaming Processing

- Consumes **live match events** from Kafka.
- Uses **Spark Structured Streaming** to calculate live KPIs.
- Stores live match state in **Redis / NoSQL** for real-time access.

4. Data Storage

- **Raw Layer (Bronze)** → Stores ingested data as-is in **Delta Lake**.
- **Cleansed Layer (Silver)** → Transformed and normalized match data.
- **Aggregated Layer (Gold)** → Final insights loaded into **Snowflake**.
- **Real-time Storage** → Redis or NoSQL (MongoDB/DynamoDB) for live match updates.

5. Orchestration & Automation (Airflow)

- **Batch DAG:** Runs daily to ingest and process historical data.
- **Streaming DAG:** Monitors Kafka topics, restarts streaming jobs if needed.
- **Reporting DAG:** Triggers Snowflake dashboard refresh.

6. Reporting & Visualization

- **Batch Insights:** Power BI / Tableau dashboard for historical analysis.
- **Real-time Dashboards:** Displays live match stats, player performance, and win probabilities.

Tech Stack

Data Processing

- **Batch:** PySpark, Delta Lake, Snowflake.
- **Streaming:** Kafka, Spark Structured Streaming, Redis.
- **Orchestration:** Apache Airflow.
- **Storage:** Snowflake (analytics), Redis (real-time queries).
- **Visualization:** Power BI, Web Dashboard (React/Django).

Key Features

✅ **Batch Processing for Historical Data**

✅ **Real-time Match Scoreboard & Player Stats**

✅ **Kafka + Spark Streaming for Live Processing**

✅ **Orchestrated Pipelines using Airflow**

✅ **Reporting & Dashboards in Snowflake**

Next Steps

- Finalize the **schema design** for OLTP and Snowflake.
- Set up **Kafka topics** and define **real-time events**.
- Develop **Spark Streaming job** for live match updates.
- Build an **Airflow DAG for orchestration**.
- Create **dashboards for visualization**.

This document serves as a blueprint for implementation. Let me know if any refinements are needed before development starts!