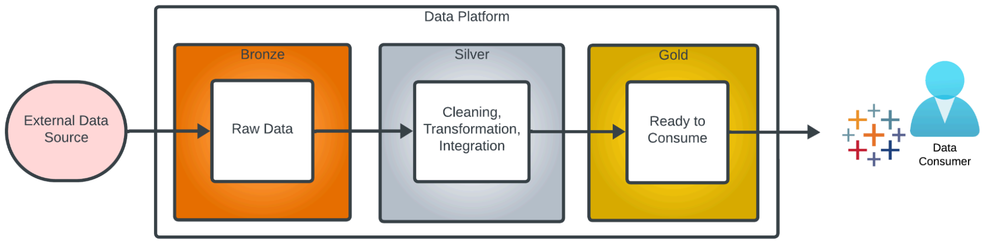
Problem Description

Build a movie rating aggregation platform that sources ratings from multiple partner platforms(IMDB, Rotten tomatoes, etc) and based on them calculate a market unique C-Rating. Data comes through APIs or Datafeeds. The C-Rating should be a normalized movie rating between 0 and 100 split into 3 categories named performance, screenplay, and soundtrack.

Solution Description

Our data platform will be using the **Medallion data architecture framework** which segments data layers into 3 layers namely **Bronze**, **Silver** and **Gold**. These layers show the level of data transformation and refinement.

Bronze layer(Raw Data Layer) : Stores raw, unprocessed data that is ingested from various source platforms. No transformations are done, maybe timestamped if necessary.
Silver Layer(Clean Processed Data Layer) : Process data from the bronze into refined and structured form. The layer involves processing like deduplication, data cleaning, normalization, missing data management, transformation etc.
Gold Layer(Curated Data Layer) : The gold layer contains data that is fully processed, aggregated and ready to be used by apps, BI tools, analysts and more.



Functional requirements

Aggregate ratings from multiple partner platforms to produce C-Rating for C-Movies.
A search on C-Movies interface should produce movie details and C-Ratings with splits as well.

Non-functional Requirements

The system should be scalable to handle a growing amount of data sources and different data formats.
The system must ensure data consistency during data updates and aggregation.

Assumptions

To manage sourcing from partner platform rating wrt **trends**, **data freshness** and **cost**.
Daily rating sourcing : Movie release date < 60 days.
Monthly rating sourcing : Movie release date > 60 days and Movie release date < 365 days.
Yearly rating sourcing : Movie release date >365 days and Movie release date < 5 years.
For building split ratings for performance, screenplay, and soundtrack, more data will be requested from the ratings partner or some other sources: movie casting data and movie reviews.
For Simplicity the split ratings will be based on the following details:
Performance : *Acting Quality and Casting (contains info on actors, directors, sound entities etc)*
Screenplay: *Story Structure, Directors Cumulative Ratings*
Soundtrack : *music composer mapping and historical adjusted rating.*
Search Movie feature design was ignored so as to focus on the aggregation.

Technology Stack:
Data Storage: Postgres, MongoDB (Cast Mapping)
Data Orchestration: Python & airflow
FTP data dump: S3
NLP : Sentiment Analysis on reviews

High Level Design

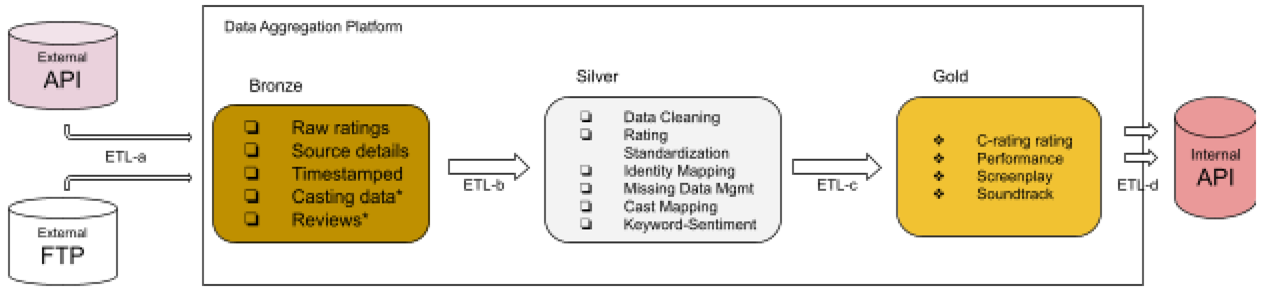


Diagram description

Data will be sourced from partner platforms and for simplicity we will assume ETL-a handles all scheduled collection from partner platforms, these data flow to the bronze layer where the raw data(ratings, source details, casting data, reviews) is time stamped without transformation. ETL-b handles transformation and dataflow to the silver layer, the transformations are rating standardization, data cleaning, sentiment analysis, cast mappings, Identity mapping. ETL-c handles final transformation and flow to the gold layer, the ETC-c pipelines computes the C-rating and its split ratings as well. ETL-d handles dataflow to the internal API services that will serve other stakeholders like BI tools, presentation interfaces and more.

Design Deep Dive

Bronze Layer	**Silver Layer**	**Gold Layer**
1. Raw ratings periodically flow into the table bronze_rating(partner_movie_id, rating, split_rating, source, datetime stamp). 2. One time collection of casting& review data : bronze_casting(partner_movie_id, casts, timestamped) 3. bronze_reviews(partner_movie_id, reviews, timestamped).	1. ETL-b will have transformations like ratings standardization, mapping partner_movie_id to C-ID, cast mapping, data cleaning and key word from reviews relating split ratings and sentiment analysis. 2. The silver layer should have processed resources/tables like: 3. silver_identity_map(partner_movie_id, C-ID)	ETL-c will handle the final gold table: gold_movie_rating(C-ID, C-movie, C-Performance rating, C-Screenplay rating, C-Soundtrack rating, Genre, Title).

	4. silver_standardized_rating(C-ID,rating,timestamped), 5. silver_aggregated_rating (C-ID, average_rating, time_stamped), 6. silver_split_rating(C-ID, rating_type, rating, timestamped) 7. silver_casts_map(C-ID,cast_id, cast_title,cast_full_name) 8. silver_reviews(C-ID, review, review_keywords, review_sentiment) 9. silver_get_performance_rating(top_5_casting_movie_rating) 10. silver_get_screenplay_rating(C-ID,directors_movie_average, review_sentiment_ratings) 11. silver_get_screenplay_rating(C-ID,screenplay_castid,review_sentiment_ratings)	

Pros & Cons of The Medallion Architecture

Pros

Unified Data Management : Integrates data storage, processing, and analytics on a single platform, reducing the need for separate systems for data lakes and data warehouses.
Simple & Easy to understand : The separation into bronze, silver, and gold layers helps in data curation
Flexibility : Capable of handling vast amounts of data, both structured and unstructured.

Cons

Implementation Complexity: Setting up and maintaining a medallion architecture requires significant technical expertise and can be complex, especially when ensuring data consistency across layers.
Data Storage Costs: Storing data in multiple layers (bronze, silver, gold) can increase storage costs, especially if not managed efficiently.
Data Freshness: Depending on the architecture's implementation and the data processing pipeline, there can be delays between the arrival of raw data and its availability in the business-ready state.

Storage Redundancy: The presence of data in multiple layers can lead to duplication, which, if not managed properly, can result in increased storage and operational costs.

Alternative Solution:
Cloud base solution(AWS) : Architecture remains the same but the DATA layers are kept as stored in S3 while ETL solutions are managed by AWS Glue and AWS Step functions. Cloudbase solutions are easier to scale because supporting services are available to create data centers , regions and more. The current solution is a good fit to an MVP.

Wrap-Up
To create a movie rating aggregation platform, we'll source ratings from IMDB, Rotten Tomatoes, and others to calculate a unique C-Rating, ranging from 0 to 100, in categories like performance, screenplay, and soundtrack. Using the Medallion architecture, data will be organized into Bronze (raw), Silver (cleaned), and Gold (curated) layers. The Bronze layer stores unprocessed data; the Silver layer refines it through cleaning and normalization; and the Gold layer aggregates and calculates the C-Ratings for use.

This architecture provides unified data management and flexibility but can be complex and costly to implement. An alternative cloud-based solution using AWS services like S3, Glue, and Step Functions could simplify scaling and management, making it an ideal choice for a minimum viable product (MVP).