University of Tartu

Faculty of Science and Technology

Institute of Computer Science

Department of Information Technology

Movie Data Analysis

Project Part 2

Max Sebastian Segerkrantz, Kevin Christian Eriksson, Oto Pruul, Siim Turban

Tartu

Changes to Project 1

3. Tooling

The entire data pipeline is orchestrated using Airflow, running inside a Docker Compose environment for full reproducibility. Airflow manages both ingestion and transformation workflows, ensuring the pipeline can be executed daily or triggered manually.

Ingestion / Loading:

- Docker to containerize the environment and make dataset ingestion repeatable.
- Download raw IMDb .tsv files and Kaggle CSV manually (or with simple scripts) into a local Docker volume.
- Version control and collaboration managed through GitHub (project repository, scripts, schema files, and documentation). **https://github.com/kevineriksson/Project-1**

Storage / Database:

All data is stored in ClickHouse, serving as both the staging and analytical warehouse. The pipeline follows a Medallion architecture consisting of Bronze and Gold layers.

Bronze layer – Raw ingested data from each source:

- bronze.tmdb_raw
- bronze.imbd_name.basics_raw (to get the nconst that is in title crew to connect to a real name)
  - nconst (string) - alphanumeric unique identifier of the name/person primaryName (string)– name by which the person is most often credited birthYear – in YYYY format deathYear – in YYYY format if applicable, else '\N' primaryProfession (array of strings)– the top-3 professions of the person knownForTitles (array of tconsts) – titles the person is known for

- bronze.imdb_title_basics_raw
- bronze.imdb_title_crew_raw

Idempotency is handled by staging new data into temporary tables, followed by an INSERT INTO ... SELECT DISTINCT step that avoids duplicates for the current ingestion date.

Gold layer – Dimensional model built via dbt:

- gold.fact_movie
- gold.dim_movie
- gold.dim_genre • gold.dim_release_date
- gold.dim_production This structure provides one fact table and at least three dimensions, fulfilling analytical model requirements.

Data Modeling:

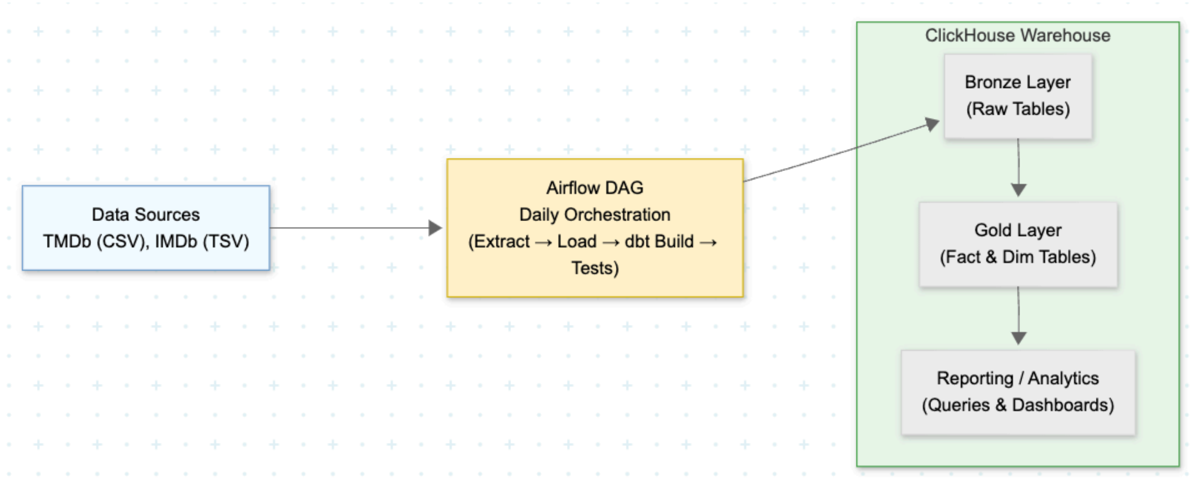
- ER diagrams to design the logical structure of how datasets relate.
- Translate the ER design into a dimensional star schema inside Postgres.

Analytics / Querying:

- Run SQL queries directly on Postgres (pgAdmin).
- Answer the business questions using the star schema tables.

4. Data Architecture

Data flow:



Data Sources:

- **IMDb datasets**
- **TMDb movies dataset** (CSV from Kaggle)

Ingestion & Orchestration:

- Data is ingested and orchestrated using an Airflow DAG, which runs daily.
- The DAG performs the following steps: Extract → Load → dbt Transform → Tests.
- Both IMDb and TMDb datasets are ingested through this daily workflow, ensuring up-to-date data.

ClickHouse Warehouse:

- **Bronze Layer:** Stores raw ingested data from IMDb and TMDb.
- **Gold Layer:** Contains cleaned and transformed fact and dimension tables built by dbt.
- **Reporting / Analytics:** Analysts can query the Gold layer for SQL-based reporting, dashboards, or other analytical use cases.

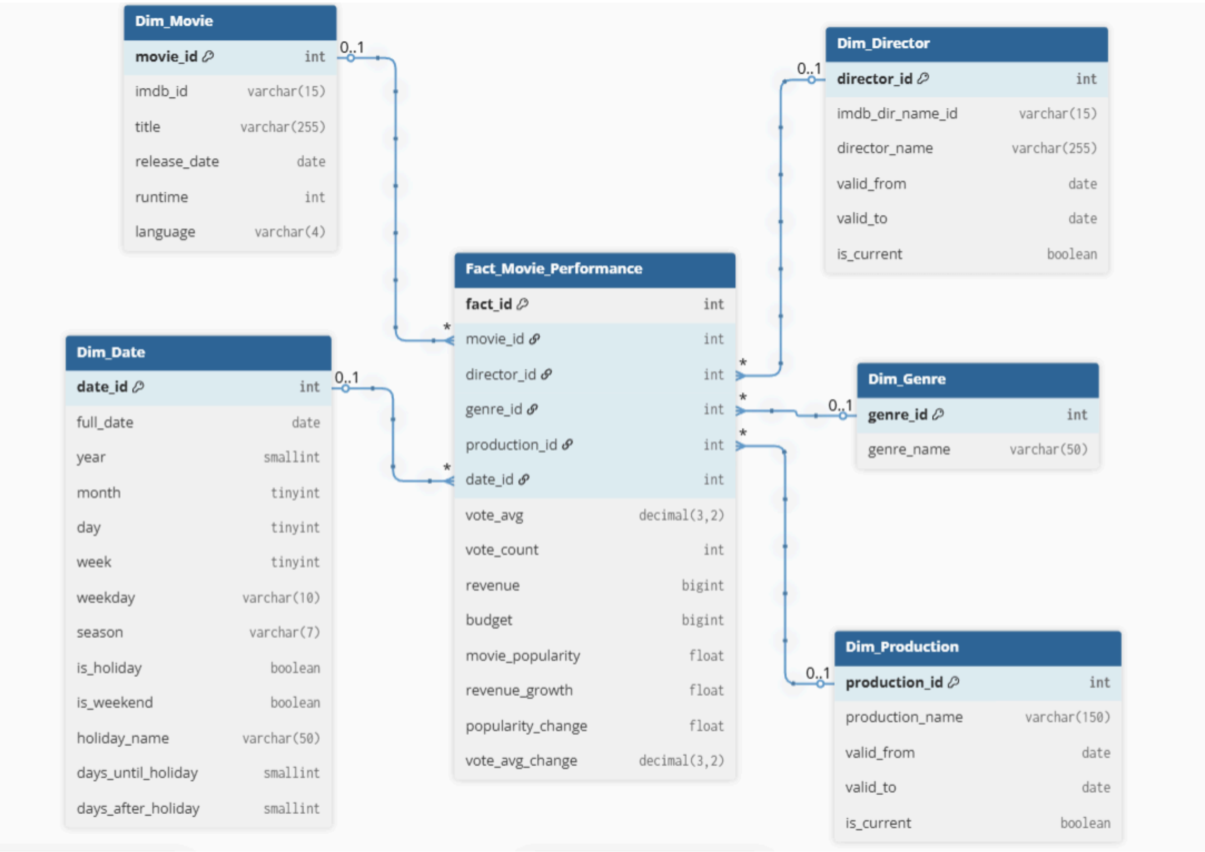
Summary Flow:

IMDb TSVs + Kaggle CSV → Airflow DAG (daily) → ClickHouse Warehouse → Bronze → Gold → Reporting

Data quality checks:

- Null check: fact_ID must not be NULL in fact table.
- Uniqueness check: imdb_ID must be unique in the movie dimension table.
- Vote average check: ratings between 0 and 10.

5. Data model



Dim_Movie	
movie_id	int
imdb_id	varchar(15)
title	varchar(255)
release_date	date
runtime	int
language	varchar(4)

Dim_Director	
director_id	int
imdb_dir_name_id	varchar(15)
director_name	varchar(255)
valid_from	date
valid_to	date
is_current	boolean

Fact_Movie_Performance	
fact_id	int
movie_id	int
director_id	int
genre_id	int
production_id	int
date_id	int
vote_avg	decimal(3,2)
vote_count	int
revenue	bigint
budget	bigint
movie_popularity	float
revenue_growth	float
popularity_change	float
vote_avg_change	decimal(3,2)

Dim_Date	
date_id	int
full_date	date
year	smallint
month	tinyint
day	tinyint
week	tinyint
weekday	varchar(10)
season	varchar(7)
is_holiday	boolean
is_weekend	boolean
holiday_name	varchar(50)
days_until_holiday	smallint
days_after_holiday	smallint

Dim_Genre	
genre_id	int
genre_name	varchar(50)

Dim_Production	
production_id	int
production_name	varchar(150)
valid_from	date
valid_to	date
is_current	boolean

Changes to the data model

1. **Removing bridge tables** - after considerations, it made more sense in our case to simplify queries by flattening genres, production companies, and directors into the fact table. For example, when there is a movie with 2 directors, 4 production companies and 5 genres, there are effectively 2 x 4 x 5 = 40 rows in the fact table. This does create duplicate metrics and increases storage requirements over time, but supports analytical queries and their performance, which is a priority. Dimensions are now directly joinable and the schema simplified.
2. **Expanding Dim_date** - we also added additional date information to have more options in time based analysis: week nr, the day of the week, season, if its a holiday or weekend release, and how close is the release to the nearest holiday. For holiday comparison, we chose from the holidays that can have a significant impact on most releases: Christmas, New Years, the 4th of July, Thanksgiving, Memorial Day, Labor Day, the Superbowl, and Halloween.

3. **Dim_movie language addition** - we also added the original language of the release, which can also provide useful information when considering making a film not in English and what makes it perform well globally.
4. **The fact table** - the performance metrics of the fact table now come from daily snapshots to track daily changes in the popularity, revenue growth, and average vote of a movie - all adding value to tracking changes. Also, movie_popularity was corrected and placed into the fact table.
5. **Adding Dim_production separately** - Since production companies can change names or merge, and have many-to-many relationships with movies, it was decided to have it as a separate dimension in the same way as directors.

Granularity and SCD

- **Grain** - one row per movie, per day, per genre, per director, per production company and the corresponding performance metrics. Overall, a daily snapshot of the movies' performance metrics. Having daily snapshots aids in efficiently tracking changes in popularity and ratings.
- **Dim_date**: STATIC because the date is fixed and unchanging.
- **Dim_genre**: TYPE1 because genres usually do not change after classification, but there is a chance that a genre is added to the list later.
- **Dim_production**: TYPE 2 because we might want to keep track of a production company's previous names or mergers between companies to compare performance before or after the changes.
- **Dim_movie**: TYPE 1 because title and runtime changes can happen, but are rare. Rather a new "edition" might be released that has a separate entry in the database.
- **Dim_director**: TYPE 2 because director names can change and it is a good idea to keep track of the change.

PROJECT PART 2:

Group member’s roles and contributions:

Due to the loss of a group member who did not contribute in Part 1 and was expected to take on more work in Part 2, the remaining members redistributed the workload evenly among themselves. Contributions are as follows:

- Max Sebastian Segerkrantz (33%)
  Responsible for the bronze schema and the original movie pipeline DAG.

- Kevin Christian Eriksson (33%)
  Handled the bronze → gold workflow and fine-tuned the movie pipeline DAG.

- Oto Pruul (33%)
  Led the redesign and modeling of the complete star schema, incorporating major structural changes from Part 1.

LLM disclosure:

- ChatGPT and Google Gemini were used during the development process primarily for debugging and troubleshooting issues encountered in data pipelines and schema design. These tools assisted in identifying errors and optimizing code to ensure functionality. No direct code or text was copied from LLMs; they were used strictly as debugging and problem-solving aids.

  No specific conversation links are provided, as the process involved numerous iterative debugging sessions and extensive adjustments throughout the project.