ETL challenge.

The next challenge requires the development of a Django web application that manages a collection of movies extracted from IMDB that will be loaded into a Postgres database and that also exposes a series of endpoints that comply with the API-Rest scheme. Use **Python 3** and **venv** for the implementation of this challenge.

Download the necessary movies file

1. Generate a relational model using the entities you are able to identify when decomposing the movie file. write the basic unit tests using Pytest

2. Develop a new Django custom command that allows transforming the content of the movies file to a JSON format so that the initial loading of the database is done through fixtures. **At this point the performance is important, show by console the time it takes the command to do its work.**

3. Using the Django' Admin, register the models you consider in order to generate a dashboard that is close enough to the structure of the movies file.

4. In the admin, generate the necessary custom filters for these in ability to show the result of the following questions:

- Which are the 10 movies that raised the most money?
- What are the 10 films that raised the least money?
- Which are the 7 films that spent the most money to produce?
- What are the 7 films that spent the least money to produce?
- Which movie genre raised the most money for each year?
- Which genre do people like best?
- Which 5 directors have the best reputation?

At least 5 of the above filters must be implemented, an additional score is given if all are implemented.

5. Expose through an API-Rest the necessary endpoints that allow:

- Return all movies in which an actor has participated
- Return all films directed by a director, sorted by year of publication
- Return to the movies grouped by gender and order by major collection

The above endpoints are read-only.

6. Using **Angular** in its last version, design a very simple web page where it shows the results exposed by the previous APIs. be ingenious!

Extra bonus points: Docker-ize the solution, so that I can run the code and tests without any assumption of my local setup (including running a postgres instance in docker-compose)

You don't need to go through all of the steps, but there should be instructions on how I can run the code. I mainly want to see how you approach a problem and your coding style. There are multiple steps so you have the option to show me different skills. It's up to you.