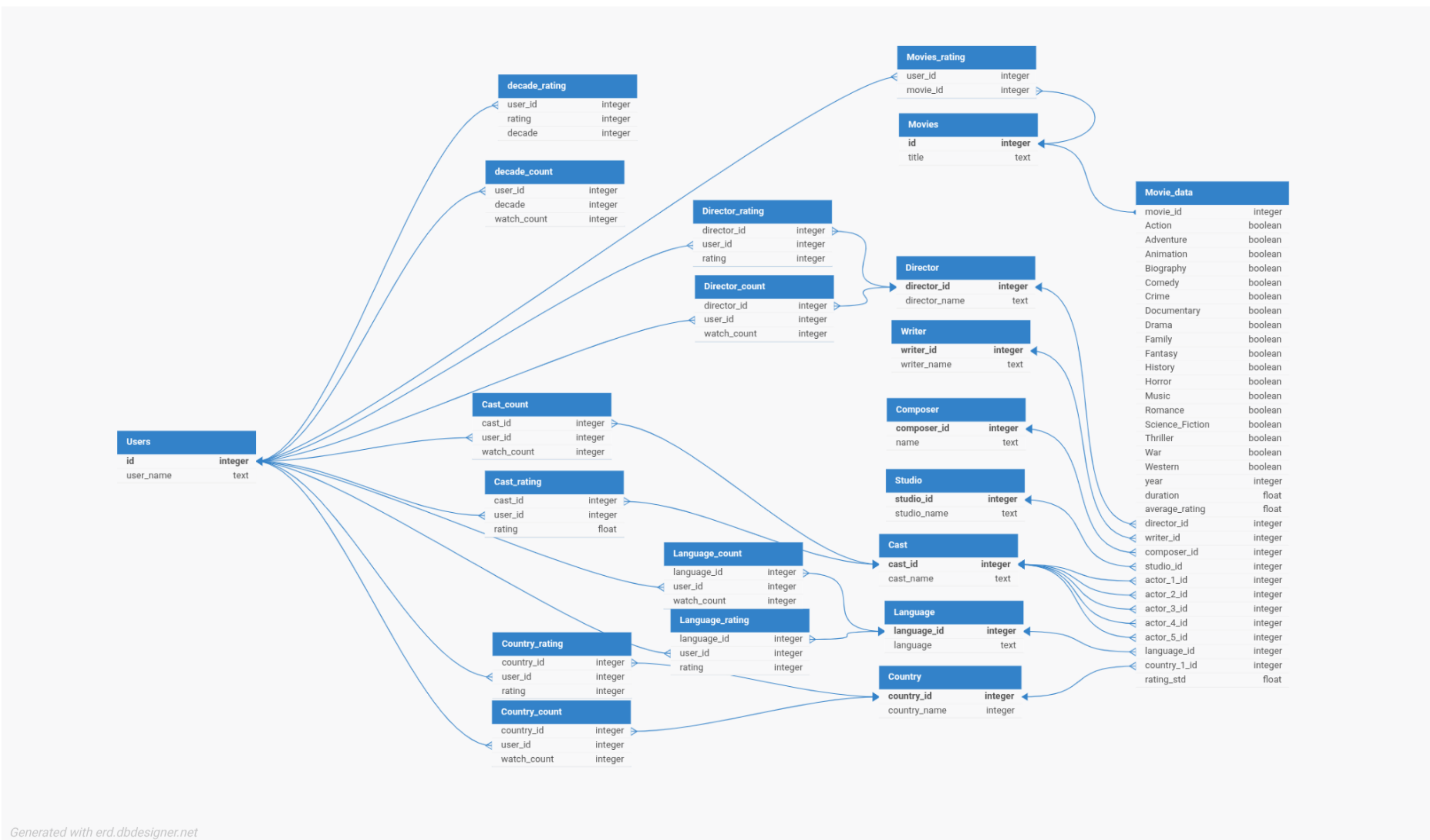
Phase 2

In The Name Of GOD

Datascience Prject: Phase 2
Members: Sepanta Ghonoodi, Shahab Sherafat, Mahdi Yari

Database schema



Link: https://dbdesigner.page.link/QQwD3K8r53MfD5P17

Queries On Database

1. A simple one:

```
SELECT movie_id, rating
FROM Movies_ratings;
```

This query gets the ratings of each movie from ratings table.
Result:

	movie_id	rating
1237	186	3.0
1238	186	4.0
1239	186	4.0
1240	451	4.5
1241	451	2.5
1242	451	5.0
1243	451	5.0
1244	451	3.0
1245	451	5.0
1246	451	4.0
1247	451	3.0
1248	451	5.0

2. Selecting the movies with highest rates using filtering and sorting:

```
SELECT movie_id, AVG(rating) as avg_rating
FROM Movies_ratings
GROUP BY movie_id
HAVING AVG(rating) > 4.0
ORDER BY avg_rating DESC;
```

Result:

	movie_id	avg_rating
1	335	5.0
2	190	5.0
3	153	4.89285714285714
4	218	4.75
5	523	4.71875
6	724	4.70909090909091
7	163	4.69495091164095
8	649	4.67362924281984
9	303	4.62686567164179
10	598	4.62570888468809
11	517	4.6216814159292
12	429	4.61979166666667

3. Joining movies and ratings tables:

```
SELECT m.movie_id, m.movie_title, m.genre, AVG(r.rating) AS avg_rating
FROM Movies m
JOIN Movies_ratings r ON m.movie_id = r.movie_id
GROUP BY m.movie_id
ORDER BY avg_rating DESC;
```

Result:

	movie_id	title	avg_rating
1	335	Children of Heaven	5.0
2	190	12th Fail	5.0
3	153	The Grapes of Wrath	4.89285714285714
4	218	Das Boot	4.75
5	523	The Best Years of Our Lives	4.71875
6	724	The General	4.70909090909091
7	163	12 Angry Men	4.69495091164095
8	649	Parasite	4.67362924281984

4. Counting the number of movies each user's seen:

```
SELECT user_id, COUNT(DISTINCT movie_id) AS num_movies_rated
FROM Movies_ratings
GROUP BY user_id
ORDER BY num_movies_rated DESC;
```

Result:

	user_id	num_movies_rated
2	296	711
3	147	710
4	603	705
5	653	698
6	277	696
7	847	694
8	826	694
9	727	694
10	537	694
11	561	693
12	343	693
13	272	693

5. Selecting the movies based which were released after 2010:

```
SELECT movie_id, title, year
FROM Movies
WHERE year > 2010;
```

Result:

	movie_id	title	year
1	2	Ant-Man	2015
2	3	Jai Bhim	2021
3	4	The Nice Guys	2016
4	5	Aftersun	2022
5	6	The Cabin in the Woods	2011

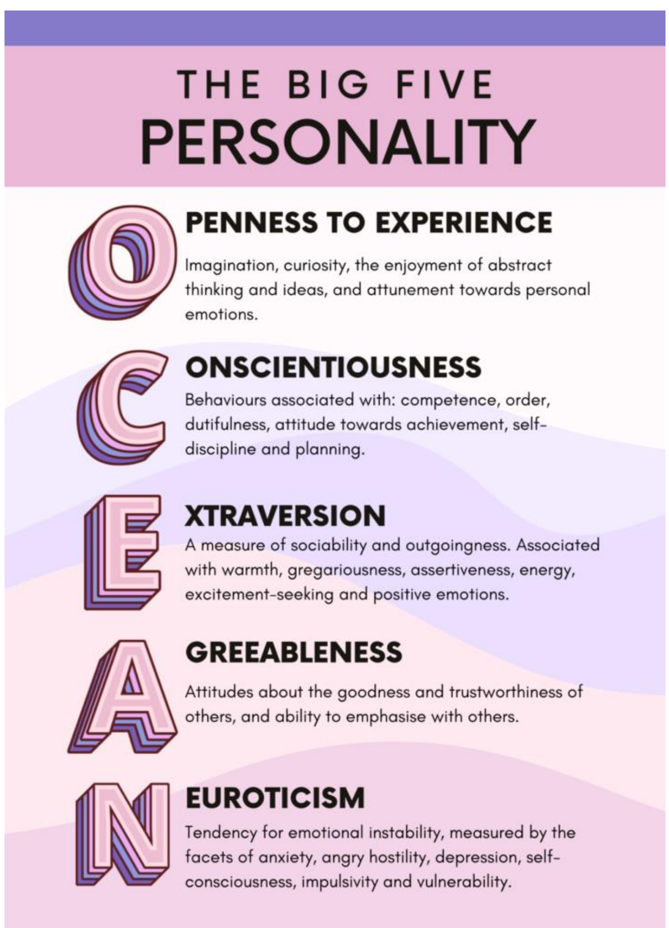
Feature Engineering

For this part we added some new features which rationally will help the model recommend better. We tried to consider the logic behind what the model will do and give it some useful extra information about how each movie or each user are.

1. **Favorite Genre For Each User:**
According to the ratings we have we found the average rating of each genre given by each user and then we assigned the genre with the maximum rating for each user.

2. **Ratings STD For Each Movie**:
We know that the movies which have more variated ratings might help the model categorize and recommend users to each other better. So using the ratings we found the STD of the ratings for each movie and assigned it as a feature of movies.

3. **Analyzing Each Person's Personality**:
It is possible to find a person's characteristics by the movies they've seen! Using the big five model we tried to define a new one hot encoded feature for users which determines the five personality traits for each user according to the genres they like.



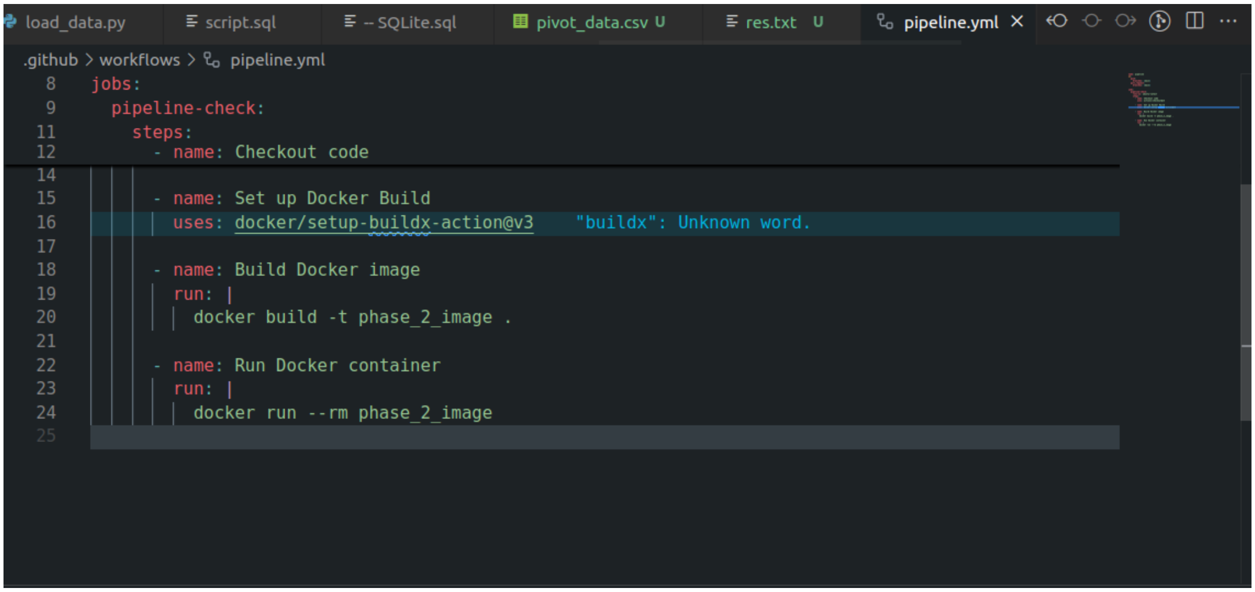
Phase 2

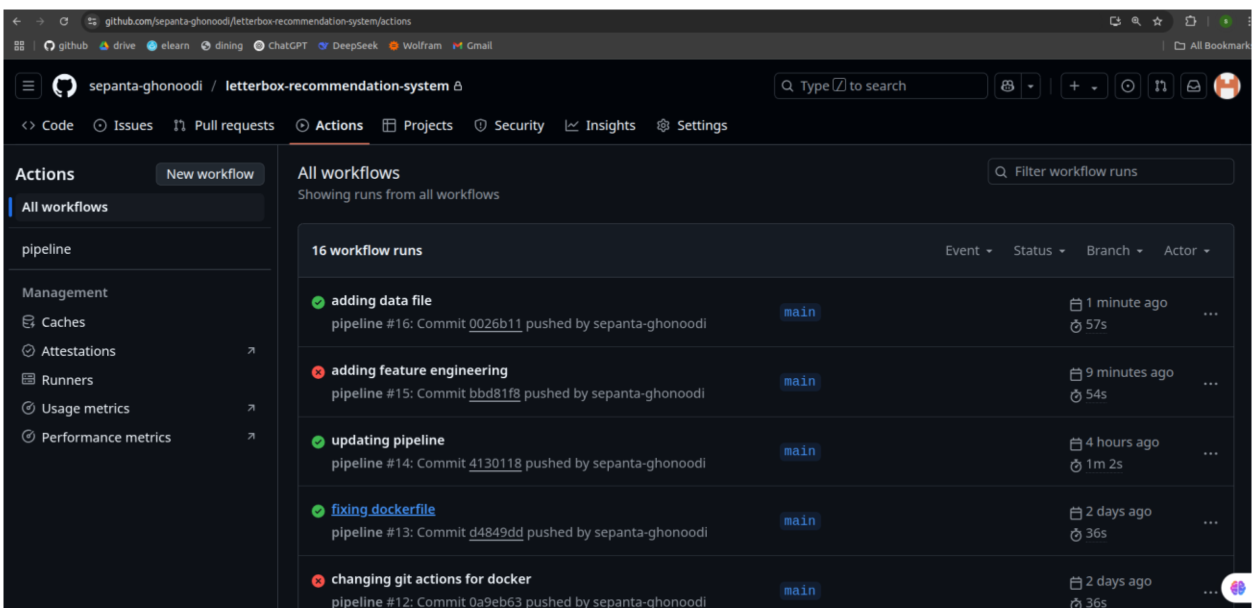
For each personality trait inside each user we calculated the average rating of the genres that trait might be related to. Then we assigned a threshold(3.5) for the ratings to know whether a person likes that genre or not. If a person likes all the genres of a trait, the one hot feature of that trait will be equal to 1.

We normalized the numerical features using sklearn and removed the null data.
About removing unuseful features, because each feature is an independent feature from the others and we've concluded that there might be no irrelevant feature and no correlation between them. So we decided to keep all the features.

Phase 2

CI/CD Implementation





MLOps

Docker file:

Dockerfile × pipeline.py preprocess.ipynb db_export.ipynb feature_engineering.ipynb

Dockerfile > ...

```
You, 2 days ago | 1 author (You)
FROM python:3.8-slim
WORKDIR /app
COPY requirements.txt .

RUN pip install --upgrade pip && pip install -r requirements.txt
COPY phase2/ .        You, 2 days ago • fixing dockerfile …
CMD ["python3", "pipeline.py"]
```

Docker build:

```
~/D/1S4/DSC/letterbox-project (main*) » docker build --network=host -t phase_2_image .          wren@Wren
2025/05/05 23:40:02 in: []string{}
2025/05/05 23:40:02 Parsed entitlements: []
[+] Building 2.6s (10/10) FINISHED                                                          docker:default
 => [internal] load build definition from Dockerfile                                                  0.1s
 => => transferring dockerfile: 209B                                                                  0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim                                    1.7s
 => [internal] load .dockerignore                                                                     0.1s
 => => transferring context: 2B                                                                       0.0s
 => [1/5] FROM docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a831beb13a0e4d073280665ea7be7f69ce2382f29c5a613f  0.0s
 => [internal] load build context                                                                     0.1s
 => => transferring context: 1.03kB                                                                   0.0s
 => CACHED [2/5] WORKDIR /app                                                                         0.0s
 => CACHED [3/5] COPY requirements.txt .                                                              0.0s
 => CACHED [4/5] RUN pip install --upgrade pip && pip install -r requirements.txt                     0.0s
 => [5/5] COPY phase2/ .                                                                              0.2s
 => exporting to image                                                                                0.2s
 => => exporting layers                                                                               0.1s
 => => writing image sha256:6629d96cdd6842d50137b80358e65218a9820e477f104be4edf14ede304c8fd1          0.0s
 => => naming to docker.io/library/phase_2_image                                                      0.0s
```

Docker run:

```
~/D/1S4/DSC/letterbox-project (main*) » docker run --rm -v $(pwd)/data:/app/data phase_2_image    wren@Wren
wrote Cast_df to file
wrote Director_df to file
wrote Language_df to file
wrote Country_df to file
wrote decade_rating_df to file
wrote decade_count_df to file
wrote User_df to file
wrote Director_rating_df to file
wrote Director_count_df to file
wrote Language_rating_df to file
wrote Languages_count_df to file
wrote Countries_count_df to file
wrote Countries_rating_df to file
wrote Cast_rating_df to file
wrote Cast_count_df to file
wrote new_data_df to file
wrote Movie_Cast_df to file
wrote sqlite_sequence_df to file
wrote Movie_Genres_df to file
wrote Writer_df to file
wrote Composer_df to file
wrote Studio_df to file
wrote Movies_df to file
wrote Movie_Composer_df to file
wrote Movie_Director_df to file
wrote Movie_Writer_df to file
wrote Movie_Country_df to file
wrote Movie_Language_df to file
wrote Movie_Studio_df to file
wrote ratings_df to file
wrote Genre_df to file
wrote User_Favorite_Genre_df to file
wrote Final_Movie_Data_df to file
wrote Movies_ratings_df to file
Executing: 100%|██████████| 6/6 [00:02<00:00,  2.85cell/s]
Executing: 100%|██████████| 11/11 [00:01<00:00,  6.78cell/s]
```