TASK 1

The research document is in the upload.

TASK 2

Attached document of the ER Diagram

TASK 3

STEP 1: Mapping of regular (strong) entity types

For entity USER_PROFILES create relation USERS.
Include all simple attributes of USER_PROFILES.
For entity MOVIE create relation MOVIES.
Include all simple attributes of MOVIE.
For entity SERIES create relation SERIES.
Include all simple attributes of SERIES.
For entity ACTOR create relation ACTORS.
Include all simple attributes of ACTOR.
For entity PRODUCTION_COMPANY create relation PRODUCTION_COMPANIES.
Include all simple attributes of PRODUCTION_COMPANY.
For entity EMPLOYEE create relation EMPLOYEES.
Include all simple attributes of EMPLOYEE.
For entity ACCOUNT_SUBSCRIPTION create relation SUBSCRIPTIONS.
Include all simple attributes of ACCOUNT_SUBSCRIPTION.

STEP 2: Mapping of weak entity types

For entity FAVOURITES create relation FAVOURITES.
Include all simple attributes of FAVOURITES.
Include primary keys of USER_PROFILE and SERIES/MOVIES
For entity CAST create relation CASTINGS.
Include all simple attributes of CAST.
Include primary keys of ACTORS and SERIES/MOVIES

STEP 3: Mapping of binary 1:1 relationships
For relationship HOLDS between USERS and SUBSCRIPTION include as foreign key in USERS the primary key of subscription.
STEP 4: Mapping of binary 1:N relationships
For the relationship EMLOY'S the relation on the N side is EMPLOYEE.
Include in EMPLOYEE the primary key of PRODUCTION_COMPANY.
STEP 5: Mapping of binary M:N relationships
Create a new relation HANDLE to represent relationship HANDLE.
Include primary keys of USERS and EMPLOYEES.
Combine these into foreign key.
STEP 6: Mapping of multivalued attributes
Create relations Reviews.
Include seriesID/movieID as foreign key.
STEP 7: Mapping of multivalued relationships
Create new relation ACT and include all the primary keys of the participating relationships.
STEP 8: Mapping specialisation and generalisation
Choose option 8C:
Create superclass SUBSCRIPTION with type paymentMethod.
This is because the sub-classes are disjoint simple attributes.
STEP 9: Mapping unions
Create new relations DEVELOPER, POST_PRODUCTION and ACCOUNT_MANAGEMENT.
Use surrogate key employeeID.
This is because the sub-classes have attributes of their own.

TASK 4
Used myPhpAdmin to create a database based on the ER diagram and mapping steps.

TASK 5

User Registration and Login:
Users should be able to log in securely using their credentials (username/email and password).
Web application should be able to show series and movies pulled from the database.
Be able to manage actors and production companies (create a page showing production companies and actors page).
Have the ability to add, edit and delete movies for admin users.
Be able to add a movie to favourites (create a favourites page).
Be able to sort and filter listings.
Recommend the movies based on algorithms.
Be able to share favourites.
Users are able to review a movie or series.

TASK 6

Used a python script to parse the data into the database using an IMBD website. Randomly generated data used in edge cases. The IMBD website used relevant data which is constantly being updated.

TASK 7

Q
EXPLAIN SELECT 'Movie' AS type, movies.movieID AS id, movies.title, movies.genre, movies.plot, movies.language, movies.country, movies.director AS creator_director, movies.duration, movies.releaseDate, movies.rating, movies.productionCompany, movies.userRatings, movies.lead, movies.supporting, movies.extras, movies.images FROM movies WHERE movies.genre LIKE '%Short%' -- Example filter for
genre UNION ALL SELECT 'Series' AS type, series.seriesID AS id, series.title, series.genre, series.plot, series.language, series.country, series.creator AS creator_director, series.seasons AS duration, NULL AS releaseDate, series.rating, series.productionCompany, series.userRatings, series.lead, series.supporting, series.recurring AS extras, series.images FROM series WHERE series.genre LIKE '%Short%'Ensure that both movies and series tables have indexes on columns that are frequently filtered or joined. In this case, genre and title could benefit from indexing.

Execution Plan Analysis:

- Use EXPLAIN

Analysis after query:

Query took 0.0522 seconds

Create indexing to improve performance.

CREATE INDEX idx_movies_genre ON movies(genre);

CREATE INDEX idx_series_genre ON series(genre);

CREATE INDEX idx_movies_title ON movies(title);

CREATE INDEX idx_series_title ON series(title);

Execute query again

Query took 0.0304 seconds.)

The performance increased as there was a gain due to the query execution time decreasing by 0.02 seconds.

Individual contributions:

R (Rhulani) Matiane:

Responsible for part of the research.

Created the ER diagram.

Responsible for mapping of the ER diagram.

Populated the database using a python script for movies, series, actors and production_companies.

AO (Andile) Mahlaba:

-Contributed to part of the research.

-Made contributions to the ER diagram.

- Responsible for front-end structure and user navigation.

-Kept track of readMe documentation.

TS (Tebatso) Mahlathini:

-Responsible for part of the research.

-Responsible for creation of the database schema using an RDMS.

-Implemented functionality in some of the pages.

-Implemented series manipulation functionality.

J (Jongisapho) Ndeya:

-Contributed to part of the research.

-Implemented foreign constraints in the database.

-Part of team creating the main api.php.

-Populated the rest of the database using artificial mock data.

-Implemented sort, filter and movie manipulation functionalities.

TK (Tshegofatso) Mahlase:

Made contributions to the ER diagram.

-Created the main backend api.php.

Responsible for front-end structure and user navigation.

Kept track of the readMe file.

R (Rethabile) Mokoena:

-Responsible for creation of the database schema using an RDMS.

-Part of the team creating the main api.php.

-Implemented user registration and login functionality.

-Part of the research team.