System Documentation

Database Schema Structure

The database consists of the following tables:

1. movies:

- movie_id (INT, PRIMARY KEY)
- title (VARCHAR(255), NOT NULL)
- release_date (DATE, NOT NULL)
- vote_average (FLOAT)
- overview (TEXT)
- popularity (FLOAT)

2. persons:

- person_id (INT, PRIMARY KEY)
- name (VARCHAR(255), NOT NULL)

3. genres:

- genre_id (INT, PRIMARY KEY)
- name (VARCHAR(255), NOT NULL)

4. movie_genres:

- movie_id (INT, PRIMARY KEY, FOREIGN KEY REFERENCES movies(movie_id))
- genre_id (INT, PRIMARY KEY, FOREIGN KEY REFERENCES genres(genre_id))

5. movie_cast:

- movie_id (INT, PRIMARY KEY, FOREIGN KEY REFERENCES movies(movie_id))
- person_id (INT, PRIMARY KEY, FOREIGN KEY REFERENCES persons(person_id))
- role (VARCHAR(255))
- character_name (VARCHAR(255))

Reasoning for Database Design

The chosen database design aims to efficiently store and retrieve movie-related data while maintaining normalization to avoid redundancy. The design includes separate tables for movies, persons, genres, and their relationships, which allows for flexible querying and easy maintenance.

Efficiency Considerations:

- **Normalization:** The schema is normalized to the third normal form (3NF), ensuring minimal redundancy and efficient updates.
- **Indexes:** Indexes are added to optimize query performance, especially for read-heavy operations.

Drawbacks of Alternative Designs:

- **Denormalization:** While denormalization could improve read performance, it would lead to data redundancy and complicate updates.
- **Single Table Design:** Storing all data in a single table would result in a large, unwieldy table with many NULL values and inefficient queries.

Database Optimizations

Indexes:

- **Full-text Index:** idx_overview on movies(overview) for efficient text searches.
- **Single-column Indexes:** idx_popularity on movies(popularity) and idx_vote_average on movies(vote_average) for sorting and filtering.
- **Composite Indexes:** idx_movie_genres_genre_movie on movie_genres(genre_id, movie_id) and idx_movies_vote_movie on movies(vote_average, movie_id) for optimizing complex queries.
- **Additional Indexes:** Indexes on movie_cast(movie_id, person_id), movie_cast(person_id, role), genres(genre_id), and persons(person_id) for efficient joins and lookups.

Main Queries

Query 1: Find the top 5 movies mentioning 'gangster' in their overview with the highest average rating, including genres.

- **Purpose:** Identify highly-rated movies with specific keywords.
- **Support:** Full-text index on overview and join with movie_genres and genres.

Query 2: Find the top 5 most popular movies mentioning 'Action' in their overview, along with their genres and the number of actors in each movie.

- **Purpose:** Identify popular movies with specific keywords and their cast details.
- **Support:** Full-text index on overview, join with movie_genres, genres, and movie_cast.

Query 3: Find movies with the most diverse cast (actors from different genres).

- **Purpose:** Identify movies with a varied cast.
- **Support:** Join with movie_cast, movie_genres, and genres.

Query 4: Find the highest-rated movie in each genre, along with its rating, genre name, and popularity.

- **Purpose:** Identify top-rated movies by genre.
- **Support:** Composite indexes on movie_genres and movies.

Query 5: Find top 5 directors by average vote_average of the movies they directed.

- **Purpose:** Identify directors with highly-rated movies.
- **Support:** Join with movie_cast and persons.

Code Structure and API Usage

Code Structure:

- **src/create_db_script.py:** Contains functions to create the database and tables, and add indexes.
- **src/api_data_retrieve.py:** Fetches data from the TMDB API and populates the database.
- **src/queries_db_script.py:** Contains SQL query functions.
- **src/queries_execution.py:** Executes the queries and prints results.
- **src/utils.py:** Utility functions for database connection.

API Usage:

- **TMDB API:** Used to fetch movie data, genres, and credits.
- **Python requests library:** Used to make API calls.
- **mysql.connector library:** Used to interact with the MySQL database.