iBHubs

django_assignment_002.md

IMDB

In this practice assignment you will be building a database for a web app similar to IMDB and will be doing on basic operations on the database.

Coding Guidelines

- You need to write all your models in `models.py` file
- Write all the functions given in different tasks in `utils.py` file
- Both **models.py** and **utils.py** should be in the `imdb` app given to you.
- Use the exact file, class, function, and field names given at each task.
- Try solving below questions with reverse relations or relation related fields instead of writing multiple queries.

Task 1: Writing Models

IMDB majorly contains movies, actors, ratings, directors, and the relations between these entities. Details related to the entities are mentioned below. Your task is to write the models to store data of these entities and their relationships.

- A `Movie` has a
  - `name` , a string value of maximum 100 characters long
  - `movie_id` a string of maximum 100 characters long can be used to uniquely identifiy a movie in the database.
  - `release_date` indicating the date on which the movie is released.
  - `box_office_collection_in_crores` a float value representing the box office collections of the movie in crores i.e. 100.23
  - Only one `director` directs a movie. And a movie must have a director

  - `actor_id` , a string of maximum 100 characters long, which can be used to uniquely identify an actor in the database.
  - a `name` , a string of maximum 100 characters long

- Each `Director` has a unique name
- `Cast` represents the actors in a movie.
  - An `actor` can act in more than one movie.
  - A `movie` can have more than one actor.
  - An actor can have more than one `role` (a string of maximum 50 characters long) in the movie.
  - `is_debut_movie` a boolean in `Cast` represents if it is a debut movie for that specific actor and is false by default.
- Rating
  - In rating, we store the number of 1,2,3,4 & 5 ratings a movie has got in the following fields `rating_one_count` , `rating_two_count` , `rating_three_count` , `rating_four_coun` & `rating_five_count` respectively.
  - Ratings start with 0 by default.

Task 2: Populate your DB

Write a function that takes the following dictionaries as input and populates the database.

```py
def populate_database(
        actors_list, movies_list, directors_list, movie_rating_list):
    """
    :param actors_list:[
        {
            "actor_id": "actor_1",
            "name": "Actor 1"
        }
    ]
    :param movies_list: [
        {
            "movie_id": "movie_1",
```

iBHubs

```
                {
                    "actor_id": "actor_1",
                    "role": "hero",
                    "is_debut_movie": False
                }
            ],
            "box_office_collection_in_crores": "12.3",
            "release_date": "2020-3-3",
            "director_name": "Director 1"
        }
    ]
    :param directors_list: [
        "Director 1"
    ]
    :param movie_rating_list: [
        {
            "movie_id": "movie_1",
            "rating_one_count": 4,
            "rating_two_count": 4,
            "rating_three_count": 4,
            "rating_four_count": 4,
            "rating_five_count": 4
        }
    ]
```

Task 3

Given an `actor_id` , Count the distinct number of movies he/she acted.

py

```
def get_no_of_distinct_movies_actor_acted(actor_id):
    """
    :param actor_id: 'actor_1'
    :return:
    Number of movies he/she acted
    Sample Output: 4
    """
```

iBHubs

Task 4

Given a director, Write a function which returns all the movies he/she directed

```py
def get_movies_directed_by_director(director_obj):
    """
    :param director_obj: <Director: Director 1>
    :return:
    List of movie objects
    Sample Output: [<Movie: movie_1_obj>, <Movie: movie_2_obj>]
    """
```

Task 5

Given a movie object, Write a function that gives the average rating. (If the movie doesn't have a rating associated then return 0)

```py
def get_average_rating_of_movie(movie_obj):
    """
    :param movie_obj: <Movie: movie_1>
    :return:
    Average Rating
    Sample Output: 4.5
    """
```

Task 6

Given a movie, Write a function that deletes the rating object associated with it.

```py
def delete_movie_rating(movie_obj):
    """
```

iBHubs

Task 7

Get all distinct actor objects who acted in given list of movies

```py
def get_all_actor_objects_acted_in_given_movies(movie_objs):
    """
    :param movie_objs: [<Movie: movie_1>, <Movie: movie_2>, ..]
    :return:
    List of actor objects
    Sample Output: [<Actor: actor_1>, <Actor: actor_2>, ..]
    """
```

Task 8

Write a function to update the director for a given movie.

```py
def update_director_for_given_movie(movie_obj, director_obj):
    """
    :param movie_obj: <Movie movie_1>
    :param director_obj: <Director: Director 1>
    :return:
    """
```

Task 9

Get all distinct movies that have an actor whose `name` contains “john”.

```
    :return:
    movie_objs: [<Movie movie_1>, <Movie movie_2>, ..]
    """
```

Task 10

Remove all actors from a given movie.

Note: Do not delete the actors from the database.

```py
def remove_all_actors_from_given_movie(movie_obj):
    """
    :param movie_obj: <Movie: movie_1>
    :return:
    """
```

Task 11

Get all rating objects associated with the given list of movies(Ignore the movies which don't have a rating)

```py
def get_all_rating_objects_for_given_movies(movie_objs):
    """
    :param movie_objs: [<Movie: movie_1>, <Movie: movie_2>, <Movie: movie_3>,..]
    :return:
    rating_objs: [<Rating: rating_1>, <Rating: rating_2>, ..]
    """
```