iBHubs

IMDB cont'd

In this practice assignment let us continue with the IMDB Database tables defined in the previous assignment & add new functions to get the required information described in each task.

Coding Guidelines

- Create a django project `django_assignment_005` in `/home/ec2-user/environment/django/django_submissions` folder
- Create an app called `imdb` in the above created project
- You can reuse the models written in the previous assignment i.e copy the models from previous assignment.
- 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.
- You can use the **same models from Assignment 004** for this assignment as well.

> **Note**
>
> You can make use of the existing functions you have written in the previous assignments

> **Note**
>
> You need to write your code in the efficient way possible

```py
# models.py
# -*- coding: utf-8 -*-
from __future__ import unicode_literals

from django.db import models
```

```
class Actor(models.Model):
    actor_id = models.CharField(max_length=100, primary_key=True)
    name = models.CharField(max_length=100)
    gender = models.CharField(max_length=50, null=True)

    def __str__(self):
        return self.name


class Director(models.Model):
    name = models.CharField(max_length=50, unique=True)

    def __str__(self):
        return self.name


class Movie(models.Model):
    movie_id = models.CharField(max_length=100, primary_key=True)
    name = models.CharField(max_length=100)
    actors = models.ManyToManyField(Actor, through='Cast')
    director = models.ForeignKey(Director, on_delete=models.CASCADE)
    release_date = models.DateField()
    box_office_collection_in_crores = models.FloatField()

    def __str__(self):
        return self.name


class Cast(models.Model):
    movie = models.ForeignKey(Movie, on_delete=models.CASCADE)
    actor = models.ForeignKey(Actor, on_delete=models.CASCADE)
    role = models.CharField(max_length=50)
    is_debut_movie = models.BooleanField(default=False)

    def __str__(self):
        return self.role


class Rating(models.Model):
    movie = models.OneToOneField(Movie, on_delete=models.CASCADE)
    rating_one_count = models.IntegerField(default=0)
    rating_two_count = models.IntegerField(default=0)
```

```
    def __str__(self):
        return "Rating {}".format(self.id)
```

Task 1

You need to write a populate_database function which takes dictionaries specified as below and creates entries in database efficiently.

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

```
            "rating_one_count": 4,
            "rating_two_count": 4,
            "rating_three_count": 4,
            "rating_four_count": 4,
            "rating_five_count": 4
        }
    ]
    :return:
    """
```

Task 2

Remove all actors from a given movie (Don't delete actor objects).

Input: Movie model instance

```py
def remove_all_actors_from_given_movie(movie_object):
    """
    :param movie_object: A Movie model instance
    """
```

Task 3

Get all ratings for given list of movies

Input: List of Movie model instances **Output**: List of Rating model instances

```py
def get_all_rating_objects_for_given_movies(movie_objs):
    """
    :movie_objs: a list of Movie model instances
    :return: a list of rating model instances
    """
```

iBHubs

Task 4

Get movie details given movie names

Input: List of strings **Output**: List of dictionaries as mentioned below

```py
def get_movies_by_given_movie_names(movie_names):
    """
    :movie_names: list of strings
    :return:
    [{
        "movie_id": 1,
        "name": "Titanic",
        "cast": [
            {
                "actor": {
                    "name": "Kate Winslet",
                    "actor_id": 1
                },
                "role": "Lead Actress",
                "is_debut_movie": False
            }
        ],
        "box_office_collection_in_crores": "218.7",
        "release_date": "1997-11-18",
        "director_name": "James Cameron",
        "average_rating": 4.9,
        "total_number_of_ratings": 1000
    }]
    """
```

Task 5

Get list of unique actors objects who acted in given movie objects

Input: A list of Movie model objects **Output**: A list of Actor model objects

```
    :movie_objs: A list of Movie model objects
    :return: A list of Actor model objects
  """
```

Task 6

Get movies which have atleast 5 female actors. Return the movies details of all such movies (ignore any male actor in the cast while returning the dictionary)

Output: A list of movie details as mentioned below.

```py
def get_female_cast_details_from_movies_having_more_than_five_female_cast():
  """
  :return:
    [{
        "movie_id": 1,
        "name": "Titanic",
        "cast": [
            {
                "actor": {
                    "name": "Kate Winslet",
                    "actor_id": 1
                },
                "role": "Lead Actress",
                "is_debut_movie": False
            }
        ],
        "box_office_collection_in_crores": "218.7",
        "release_date": "1997-11-18",
        "director_name": "James Cameron",
        "average_rating": 4.9,
        "total_number_of_ratings": 1000
    }]
  """
```

Task 7

Output: A list of dictionaries

```py
def get_actor_movies_released_in_year_greater_than_or_equal_to_2000():
    """
    :return: a list of Movie model instances
    [
        {
            "name": "Kate Winslet",
            "actor_id": 1
            "movies": [
                {
                    "movie_id": 1,
                    "name": "Titanic",
                    "cast": [
                        {
                            "role": "Lead Actress",
                            "is_debut_movie": False
                        }
                    ],
                    "box_office_collection_in_crores": "218.7",
                    "release_date": "1997-11-18",
                    "director_name": "James Cameron",
                    "average_rating": 4.9,
                    "total_number_of_ratings": 1000
                }
            ]
        }
    ]
    """
```

Task 8

Reset all ratings of all movies which released in the given year

INPUT: year

```py
def reset_ratings_for_movies_in_given_year(year):
    """
```

iBHubs