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_004` 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 002** for this assignment as well.
- Add gender to Actor model, which is a string of max length 10. Values of gender can be "MALE" (OR) "FEMALE".

Task 1

Find the average `box_office_collection_in_crores` for all movies in the database.

> Note: round the result to 3 decimal places and if there are no movies in database return 0

py

```
def get_average_box_office_collections():
    """
    :return: float value
    100.123
    """
```

iBHubs

Task 2

Calculate the number of distinct actors for each movie. Your code should return a list of `Movie` model instances with `actors_count` attribute representing the number of unique actors in that movie.

```py
def get_movies_with_distinct_actors_count():
  """
  :return: a list of Movie model instances
  """
```

Task 3

Calculate the male and female actors count for each movie in the database. Your code should return a list of `Movie` model instances with `male_actors_count` & `female_actors_count` attributes representing the number of male actors and female actors count in that movie, respectively.

```py
def get_male_and_female_actors_count_for_each_movie():
    """
    :return: a list of Movie model instances
```

Task 4

Calculate the number of unique `roles` in each movie. Your code should return a list of `Movie` model instances with `roles_count` attributes representing the number of distinct `roles` in that movie, respectively.

```py
def get_roles_count_for_each_movie():
"""
:return:
```

iBHubs

Task 5

For each role in the database, Return the role, and the number of distinct actors played that role. Your code should return a dictionary with the role as key and the number of actors as value.

```py
def get_role_frequency():
  """
  :return: {
    "role_1": 3,
    "role_2": 5
  }
  """
```

Task 6

For each role in the database, find the number of times some actor has cast it. Your results should return the roles which have been cast recently first(movie released date). Output format is list of tuples, where each tuple contains the role and frequency of it.

```py
def get_role_frequency_in_order():
  """
  :return: [('role_2', 5), ('role_1', 3)]

  """
```

Task 7

Calculate the total number of movies and the number of distinct roles each actor has acted in. Your code should return a list of `Actor` model instances with `movies_count` & `roles_count`

```py
def get_no_of_movies_and_distinct_roles_for_each_actor():
    """
    :return: a list of Movie model instances
    """
```

Task 8

Find the movies which have at least 40 actors. Your code should return a list of `Movie` model instances with `actors_count` attribute representing the number of distinct `actors` in that movie, respectively.

```py
def get_movies_with_atleast_forty_actors():
    """
    :return: a list of Movie model instances
    """
```

Task 9

Find the average number of actors for all movies in database.

> Note: round the result to 3 decimal places and if there are no movies in database return 0

```py
def get_average_no_of_actors_for_all_movies():
    """
    :return: 4.123
    """
```