**Python ORM - Exam Prep I**

*Create three models that allow you to manage, manipulate, and extract data from the Database. These models together form the basis of a simple movie application.*

*Your project will manage directors, actors, and movies.*

# **Skeleton**

You are provided with a ready-to-use skeleton. Do not change the folder and file names! You are allowed to add additional files.

**Judge Submissions:**

* Once you have **completed a task**, you must **archive** the project files (**zip** format) and upload the zip file to the contest (for the **corresponding judge task**). You **do not need to include** your **venv**, **.idea**, **pycache**, and **\_\_MACOSX** (for Mac users), so you do **not exceed** the maximum allowed size of **31.25** **KB**
* Submit a solution (archived project files) for **each** **task**!

A screenshot of a computer

Description automatically generated

# **Database – 100 points**

You will need to create **three models** in the **models.py** file:

### **Director Model**

* + **full\_name**
    - A **character** field.
    - Represents the full name of the director.
    - Validation: **Minimum length** of **2** characters, **maximum length** of **120** characters.
  + **birth\_date**
    - A **date** field.
    - Represents the date of birth of the director**.**
    - Default value: **'1900-01-01'**.
* **nationality**
  + - A **character** field.
    - Represents the nationality (country of origin) of the director.
    - Validation: **Maximum length** of **50** characters.
    - Default value: **'Unknown'**.
* **years\_of\_experience**
  + - A **small integer** field.
    - Represents the years of experience of the director.
    - Validation: **Minimum value:** **0**.
    - Default Value: **0**.

### **Actor Model**

* + **full\_name**
    - A **character** field.
    - Represents the full name of the actor.
    - Validation: **Minimum length** of **2** characters, **maximum length** of **120** characters.
  + **birth\_date**
    - A **date** field.
    - Represents the date of birth of the actor**.**
    - Default value: **'1900-01-01'**.
* **nationality**
  + - A **character** field.
    - Represents the nationality (country of origin) of the actor.
    - Validation: **Maximum length** of **50** characters.
    - Default value: **'Unknown'**.
* **is\_** **awarded**
  + A **boolean** field.
* Indicates whether the actor is awarded.
* Default Value: **False** (Not Awarded).
* **last\_updated**
  + A **date/time** field.
* **Automatically** records the **date** and **time** when the **actor** is **updated**.

### **Movie Model**

* + **title**
    - A **character** field.
    - Represents the title of the movie.
    - Validation: **Minimum length** of **5** characters, **maximum length** of **150** characters.
* **release\_date**
  + A **date** field.
* Represents the release date of the movie.
  + **storyline**
    - A **text** field.
    - Represents the storyline (a short summary) of the movie.
    - This field is optional and can have **Null** values.
  + **genre**
    - A **character** field with predefined **choices**.
    - Represents the genre (category) to which the movie belongs.
    - Choices: '**Action**', '**Comedy**', '**Drama**', and '**Other**'.
    - Validation: **Maximum length** of **6** characters.
    - Default value: '**Other**'.
* **rating**
  + A **decimal** field.
* Represents the rating of the movie.
  + - Validation: **Maximum digits**: **3**, **decimal places**: **1**. **Minimum value** of **0.0**, **maximum value** of **10.0**.
    - Default Value: **0.0**
* **is\_classic**
* A **boolean** field.
* Indicates whether the movie is considered a classic.
* Default Value: **False** (Not Classic).
* **is\_awarded**
* A **boolean** field.
* Indicates whether the movie has received awards.
* Default Value: **False** (Not Awarded).
* **last\_updated**
  + A **date/time** field.
* **Automatically** records the **date** and **time** when the **movie** is **updated**.
* **director**
  + A **foreign key** to the **Director** model.
* Establishes a **one-to-many** relationship with the Director model, associating each movie with a director.
* **ON DELETE** constraint must be set to **CASCADE**
* **starring\_actor**
  + A **foreign key** to the **Actor** model.
* Establishes a **one-to-many** relationship with the Actor model, associating each movie with a starring actor who is the main character.
* This field can have **Null** values.
* **ON DELETE** constraint must be set to **Null**.
* **actors**
  + A **many-to-many** field to the **Actor** model.
* Establishes a **many-to-many** relationship with the Actor model, allowing multiple actors to participate in a movie and an actor to participate in multiple movies.

**Hint**: You can use Abstract Base Classes and Model Mixins to stick to good practices and avoid code repeating (DRY principle). Please note that code optimization is **not** **intended** for **testing** in the Judge System.

# **Customizing Django Admin Site – 30 points**

Register your models to the Django Admin Site (**admin.py** file) and make the following customizations which will enhance the admin interface by providing more meaningful and searchable information:

## **DirectorAdmin**

* **Display fields**: Specify the fields to be displayed in the list view of the admin site for the **Director model**.
  + Fields: **'full\_name'**, **'birth\_date'**, **'nationality'**
* **Filters**: Add a filter for **'years\_of\_experience'** in the admin site
* **Search fields**: Enable search by '**full\_name**' and '**nationality**' in the admin site.

## **ActorAdmin**

* **Display fields**: Specify the fields to be displayed in the list view of the admin site for the **Actor model**.
  + Fields: **'full\_name'**, **'birth\_date'**, **'nationality'**
* **Filters**: Add a filter for '**is\_awarded**' in the admin site.
* **Search fields**: Enable search by **'full\_name'** in the admin site.
* **Read-only** **fields**: '**last\_updated**'.

## **MovieAdmin**

* **Display fields**: Specify the fields to be displayed in the list view of the admin site for the **Movie model**.
  + Fields: **'title', 'storyline', 'rating', 'director'**
* **Filters**: Add filters for '**is\_awarded**', '**is\_classic**'and **'genre'** in the admin site.
* **Search fields**: Enable search by **'title'** and **director’s 'full\_name'** (searching movies by director's full name).
* **Read-only** **fields**: '**last\_updated**'.

\* Consider providing a search help text to enhance the user experience (UX). This can be a brief and informative message that guides users on how to effectively use the search feature, improving their overall interaction with the application. Please note that this feature is **not** **intended** for **testing** in the Judge System.

# **Custom Model Manager – 20 points**

Create a **custom model manager** for the **Director** **model** and add your **custom method**:

## **get\_directors\_by\_movies\_count()**

This method **retrieves** and **returns** all **director objects**, **ordered by** the **number** of **movies** eachdirectorhas **descending, then by** their **full names ascending**.

# **Django Queries I – 75 points**

In the **caller.py** file create the following functions:

## **get\_directors(search\_name=None, search\_nationality=None)**

This function accepts the following arguments with default **None** values:

* **search\_name** – string value or **None**
* **search\_nationality** – string value or **None**

It **retrieves** director objects by **partially** and **case-insensitively** matching the given searching criteria for **full name** and/or **nationality**.

**First**, check if **both values** are **not None**.Then **search** for **directors** whose **full names contain** the **search\_name** string **and** their **nationality contains** the **search\_nationality** string (searching by **both** criteria).

**Otherwise**, check if at least **one** of the values is **not None** and search for directors by the **corresponding** field.

**Finally**, if **both** arguments **are None**, **return** an **empty string** **("")**.

**If there are** director objects that **match** the criteria, **order** them by **full name**, **ascending.**

**Return** a **string** in the following format, each director's info on a new line:

**"Director: {full\_name1}, nationality: { nationality1}, experience: {years\_of\_experience1}**

**Director: {full\_name2}, nationality: { nationality2}, experience: { years\_of\_experience2}**

**…**

**Director: {full\_nameN}, nationality: { nationalityN}, experience: { years\_of\_experienceN}"**

* If **no directors** match the criteria, **return** an **empty string ("")**.
* **Hint:** You can use Q objects but first check if the search string is not None.

## **get\_top\_director()**

This function accepts no arguments.

It **retrieves** the director with the **greatest number** of **movies**.

If there is **more than one** **director** with the **same number** of movies, order them by **full name**, **ascending**, and return the **first one’s** info.

**Return** a **string** in the following format:

**"Top Director: {full\_name}, movies: {num\_of\_movies}."**

* If there are **no directors**, **return** an **empty string ("")**.
* **Hint:** You can use the custom model manager method.

## **get\_top\_actor()**

This function accepts no arguments.

It **retrieves** the **starring** **actor** with the **greatest number** of **movies** s/he **starred** in**.**

If there is **more than one** **actor** with the **same number** of movies, order them by **full name**, **ascending**, and return the **first one’s** info.

**Return** a **string** in the following format:

**"Top Actor: {actor\_full\_name}, starring in movies: {movie\_title1}, {movie\_title2}, … {movie\_titleN}, movies average rating: {movies\_avg\_rating}"**

* **Movie titles** must be **separated** by a **comma and space (", ")**
* **Average rating** represents the **average value** of the **ratings** from the **movies** in which the **actor** **stars**. **Format** it to the **first decimal place.**
* If there are **no movies** and/or **no starring actors**, **return** an **empty string ("")**.

# **Django Queries II – 75 points**

## **get\_actors\_by\_movies\_count()**

This function accepts no arguments.

It **retrieves** the **top three** **actors** from all movies, **ordered** **by** the **number of times** the **actor** has participated in movies, **descending**, **then** **ascending** by the **actor’s** **full** **name**.

Take the **top three ordered actors**.

**Return** a **string** in the following format:

**"{actor\_full\_name1}, participated in {num\_movies1} movies**

**{actor\_full\_name2}, participated in {num\_movies2} movies**

**{actor\_full\_name3}, participated in {num\_movies3} movies"**

* **Each** **actor's** info is on a **new line**.
* In case the **actors** are **less than three** in total, **return all of them**, **ordered** as described.
* If there are **no movies** and respectively **no actors participating**, **return** an **empty string ("")**.

## **get\_top\_rated\_awarded\_movie()**

This function accepts no arguments.

It **retrieves** a **movie** **object** with the **highest** **rating** that has been **awarded** and its **status** is "**Awarded**" (**is\_awarded=True**).

If there are **more awarded** movies with the **same highest rating**, **order** them by **title,** **ascending,** and get the **first one**.

**Return** a **string** in the following format:

**"Top rated awarded movie: {movie\_title}, rating: {movie\_rating}. Starring actor: {starring\_actor\_full\_name/'N/A'}. Cast: {participating\_actor1}, {participating\_actor2}, …{participating\_actorN}."**

* **Movie rating** must be **formatted** to the **first decimal place.**
* If the **starring actor is None**,return **'N/A'** instead of the actor’s full name**.**
* **Cast** representsthe **list** of **all actors' full names** who **participated** in the **movie**. **Order** them **by full name**, **ascending**, and **separate** each full name with a **comma and space** (**", "**).
* If **no movies** are **awarded**, return an **empty string** (**""**).

## **increase\_rating()**

This function accepts no arguments.

It **increases** the **rating** for **all movies** that are **considered classic** – their **status** is "**Classic**" (**is\_classic=True**) and their **rating** is **not** already set to the **maximum** level.

**Increase** the **rating** by **0.1** (zero point one).

Make sure you **do not exceed** the **maximum rating value** of **10.0**.

**Return** a **string** in the following format:

**"Rating increased for {num\_of\_updated\_movies} movies."**

* If a movie **already has** the **maximum rating** and is **considered classic**, you should **not count** it as updated as no change will occur.
* If there are **no updated movies**, **return** the string:

**"No ratings increased."**

* Hint: You can use the F object to efficiently update the ratings.

# **Testing Data Constraints**

* There will always be directors and actors when creating movies.
* The following outputs show the **expected behavior** of the functions. Populate the database with your own testing data and then check if the functions produce the expected results.

# **Examples**

|  |
| --- |
| **Test Code** |
| print(Director.objects.get\_directors\_by\_movies\_count()) |
| **Output** |
| <QuerySet [<Director: Francis Ford Coppola>, <Director: Akira Kurosawa>, <Director: Martin Scorsese>]> |
| **Test Code** |
| print(get\_directors(search\_name='S', search\_nationality=None)) |
| **Output** |
| Director: Akira Kurosawa, nationality: Unknown, experience: 0  Director: Francis Ford Coppola, nationality: Unknown, experience: 50  Director: Martin Scorsese, nationality: American and Italian, experience: 60 |
| **Test Code** |
| print(get\_directors(search\_name='Martin', search\_nationality='Canadian')) |
| **Output** |
|  |
| **Test Code** |
| print(get\_top\_director()) |
| **Output** |
| Top Director: Francis Ford Coppola, movies: 2. |
| **Test Code** |
| print(get\_top\_actor()) |
| **Output** |
| Top Actor: Al Pacino, starring in movies: The Godfather, Apocalypse Now, movies average rating: 9.5 |
| **Test Code** |
| print(get\_actors\_by\_movies\_count()) |
| **Output** |
| Al Pacino, participated in 2 movies  Robert Duvall, participated in 2 movies  Joaquin Phoenix, participated in 1 movies |
| **Test Code** |
| print(get\_top\_rated\_awarded\_movie()) |
| **Output** |
| Top rated awarded movie: The Godfather, rating: 9.9. Starring actor: Al Pacino. Cast: Al Pacino, Robert Duvall. |
| **Test Code** |
| print(increase\_rating()) |
| **Output** |
| Rating increased for 1 movies. |