# **Python ORM Exam – 26 November 2023**

*Create three models that allow you to manage, manipulate, and extract data from the Database. These models together form the basis of a simple articles/reviews publishing platform.*

*Your project will manage authors, articles, and reviews.*

## **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:

### **Author Model**

* + **full\_name**
    - A **character** field.
    - Represents the full name of the author.
    - Validation: **Minimum length** of **3** characters, **maximum length** of **100** characters.
* **email**
  + - An **email** field.
    - Represents the email address of the author.
    - Validation: **Unique** values.
* **is\_banned**
  + - A **boolean** field.
    - Indicates whether the author is currently banned.
    - Default Value: **False** (Not Banned).
* **birth\_year**
  + - A **positive integer** field.
    - Represents the year of birth of the author.
    - Validation: **Minimum** value: **1900**, **maximum** value: **2005**
* **website**
  + - A **URL** field.
    - Represents the URL of the author's website.
    - Additional Note: This field is **not required** and can have **NULL** values.

### **Article Model**

* **title**
* A **character** field.
* Represents the title of the article.
* Validation: **Minimum** **length** of **5** characters, **maximum length** of **200** characters.
* **content**
  + A **text** field.
* Provides the content of the article.
* Validations: **Minimum** **length** of **10** characters.
* **category**
  + A **character** field with pre-defined **choices.**
* Represents the category of the article.
* Valid choices: "**Technology**", "**Science**",and"**Education**".
* Validation: **Maximum** **length** of **10** characters.
* Default Value: "**Technology**".
* **authors**
  + A **many-to-many** field to **Author** model.
* Establishes a **many-to-many** relationship with the Author model, allowing an article to have multiple authors and an author to have multiple articles.
* **published\_on**
  + A **date/time** field.
* **Auto-generated** with the **current date** and **time** when an **article** is **published** (created).
* This fieldis **not editable**.

### **Review Model**

* **content**
* A **text** field.
* Provides the content of the review.
* Validations: **Minimum** **length** of **10** characters.
* **rating**
  + A **float** field.
* Represents the rating given for the related article (e.g., 1.0 to 5.0).
* Validation: **Minimum value: 1.0, maximum value**: **5.0**.
* **author**
  + A **foreign key** to the **Author** model.
* Establishes a **many-to-one** relationship with the Author model, associating each review with an author.
* **ON DELETE** constraint must be set to **CASCADE**.
* **article**
  + A **foreign key** to the **Article** model.
* Establishes a **many-to-one** relationship with the Article model, associating each review with an article.
* **ON DELETE** constraint must be set to **CASCADE**.
* **published\_on**
  + A **date/time** field.
* **Auto-generated** with the **current date** and **time** when a **review** is **published** (created).
* This field is **not editable**.

## **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:

### **AuthorAdmin**

* **Display fields**: Specify the fields to be displayed in the list view of the admin site for the **Author model**.
  + Fields: **'full\_name'**, **'email'**, **'is\_banned'**
* **Filters**: Add a filter for **'is\_banned'** in the admin site.
* **Search fields**: Enable search by **'full\_name'** and **'email'** in the admin site.

### **ArticleAdmin**

* **Display fields**: Specify the fields to be displayed in the list view of the admin site for the **Article model**.
  + Fields: **'title'**, **'category'**, **'published\_on'**
* **Filters**: Add a filter for **'category'** in the admin site.
* **Search fields**: Enable search by **'title'** in the admin site.
* **Read**-**only** **fields**: **'published\_on'**.

### **ReviewAdmin**

* **Display fields**: Specify the fields to be displayed in the list view of the admin site for the **Review model**.
  + Fields: **'author', 'article', 'rating', 'published\_on'**
* **Filters**: Add filters for **'rating'** and **'published\_on'** in the admin site.
* **Search fields**: Enable search byarticle’s **'title'** (searching reviews by article's title).
* **Read-only fields**: **'published\_on'**.

## **Custom Model Manager – 20 points**

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

### **get\_authors\_by\_article\_count()**

This method **retrieves** and **returns** all **author objects**, **ordered by** the **number** of **articles** each author has, **descending**, **then by** their **emails ascending**.

## **Django Queries I – 75 points**

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

### **get\_authors(search\_name=None, search\_email=None)**

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

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

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

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

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

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

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

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

**"Author: {full\_name1}, email: {email1}, status: {Banned/Not Banned}**

**Author: {full\_name2}, email: {email2}, status: {Banned/Not Banned}**

**…**

**Author: {full\_nameN}, email: {emailN}, status: {Banned/Not Banned}"**

* If there are **no authors** matching the criteria, **return** an **empty string ("")**.
* The **status** (**'Banned'** or **'Not Banned'**) depends on the **is\_banned** value (True or False).

### **get\_top\_publisher()**

This function accepts no arguments.

It **retrieves** the author with the **greatest number** of **published articles.**

Ifthere is **more than one author** with the **same number** ofarticles**, order** them **by email, ascending**, and **return** the **first one’s** info.

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

**"Top Author: {full\_name} with {num\_of\_articles} published articles."**

* If there are **no published articles** at all, **return** an **empty string ("")**.

### **get\_top\_reviewer()**

This function accepts no arguments.

It **retrieves** the author with the **greatest number** of **published reviews.**

Ifthere is **more than one author** with the **same number** ofreviews**, order** them **by email, ascending**, and **return** the **first one’s** info.

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

**"Top Reviewer: {full\_name} with {num\_of\_reviews} published reviews."**

* If there are **no published reviews** at all, **return** an **empty string ("")**.

## **Django Queries II – 75 points**

### **get\_latest\_article()**

This function accepts no arguments.

It **retrieves** the **last published article** and **returns** a **string** in the following format:

**"The latest article is: {article\_title}. Authors: {article\_author1\_full\_name}, …, {article\_authorN\_full\_name}. Reviewed: {num\_reviews} times. Average Rating: {avg\_reviews\_rating}."**

* Authors' **full names** must be **separated** by a **comma and space (", ")** and **ordered by full name**, **ascending**.
* **Format** the **average rating** to the **second decimal place**.
* If there are **no articles** at all, **return** an **empty string ("")**.

### **get\_top\_rated\_article()**

This function accepts no arguments.

It **retrieves** the **top-rated** **article** by consideringthe **ratings** of published **reviews**. If you happen to have **articles** with the **same top rating results**, **order** them **by title**, **ascending**, and get the **first one**.

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

**"The top-rated article is: {article\_title}, with an average rating of {avg\_rating}, reviewed {num\_reviews} times."**

* **Format** the **average rating** to the **second decimal place**.
* If there are **no reviews** at all, return an **empty string** **("")**.

### **ban\_author(email=None)**

This function accepts one argument with a default **None** value:

* **email** – string, representing the **exact** value of the author’s **email** or **None**.

It **retrieves** the **author** objectwith thegiven **email** (**exact** match) and **changes** his/her status to "**Banned**" (**is\_banned=True**).

You must **delete all** author’s **reviews** but first, you will need to knowtheir **number.**

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

**"Author: {full\_name} is banned! {num\_reviews} reviews deleted."**

* If there is **no author** with the given **email** (exact match), or **no authors** at all, or the given **email** **is** **None**, **return**:

**"No authors banned."**

## **Testing Data Constraints**

* There will always be authors and articles when publishing reviews.
* 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(Author.objects.get\_authors\_by\_article\_count()) |
| **Output** |
| <QuerySet [<Author: Anna Williams>, <Author: Adam Smith>]> |
| **Test Code** |
| print(get\_authors(search\_name='I', search\_email=None)) |
| **Output** |
| Author: Anna Williams, email: aw@test.com, status: Not Banned  Author: Adam Smith, email: as@dev.com, status: Not Banned |
| **Test Code** |
| print(get\_authors(search\_name='z', search\_email='com')) |
| **Output** |
|  |
| **Test Code** |
| print(get\_top\_publisher()) |
| **Output** |
| Top Author: Anna Williams with 2 published articles. |
| **Test Code** |
| print(get\_top\_reviewer()) |
| **Output** |
| Top Reviewer: Adam Smith with 2 published reviews. |
| **Test Code** |
| print(get\_latest\_article()) |
| **Output** |
| The latest article is: Test Article. Authors: Adam Smith, Anna Williams. Reviewed: 2 times. Average Rating: 4.90. |
| **Test Code** |
| print(get\_top\_rated\_article()) |
| **Output** |
| The top-rated article is: The first ever article, with an average rating of 4.95, reviewed 2 times. |
| **Test Code** |
| print(ban\_author('as@dev.com')) |
| **Output** |
| Author: Adam Smith is banned! 2 reviews deleted. |
| **Test Code** |
| print(ban\_author('aw@test.co')) |
| **Output** |
| No authors banned. |