# Python ORM Retake Exam – 11 December 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 tennis tournament platform.*

*Your project will manage tennis players, tournaments, and matches.*

## 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:

### TennisPlayer Model

* + **full\_name**
    - A **character** field.
    - Represents the full name of the tennis player.
    - Validation: **Minimum length** of **5** characters, **maximum length** of **120** characters.
* **birth\_date**
  + - A **date** field.
    - Represents the date of birth of the tennis player.
* **country**
  + - A **character** field.
    - Represents the country of origin of the tennis player.
    - Validation: **Minimum length** of **2** characters, **maximum length** of **100** characters.
* **ranking**
  + - A **positive integer** field.
    - Represents the ranking of the tennis player.
    - Validation: **Minimum** value of **1**, **maximum** value of **300**.
* **is\_active**
  + - A **boolean** field.
    - Indicates whether the tennis player is currently active.
    - Default Value: **True** (Active).

### Tournament Model

* **name**
* A **character** field.
* Represents the name of the tournament.
* Validation: **Minimum** **length** of **2** characters, **maximum length** of **150** characters.
* This field must contain **unique** values.
* **location**
  + A **character** field.
* Provides the location of the tournament.
* Validations: **Minimum** **length** of **2** characters, **maximum length** of **100** characters.
* **prize\_money**
  + A **decimal** field.
* Represents the prize money of the tournament.
* Validations: **Maximum digits**: **10**, **decimal places**: **2**.
* **start\_date**
  + A **date** field.
* Represents the starting date of the tournament.
* **surface\_type**
  + A **character** field with pre-defined **choices.**
* Represents the surface type of the tournament.
* Valid choices: "**Not Selected**", "**Clay**", "**Grass**",and"**Hard Court**".
* Validation: **Maximum** **length** of **12** characters.
* Default Value: "**Not Selected**".

### Match Model

* **score**
* A **character** field.
* Represents match scores.
* Validations: **Maximum** **length** of **100** characters.
* **summary**
* A **text** field.
* Provides a short summary of the match.
* Validations: **Minimum** **length** of **5** characters.
* **date\_played**
  + A **date/time** field.
* Represents the **date** and **time** when amatchis **played**.
* **tournament**
  + A **foreign key** to the **Tournament** model.
* Establishes a **many-to-one** relationship with the Tournament model, associating each match with a tournament.
* **ON DELETE** constraint must be set to **CASCADE**.
* **players**
  + A **many-to-many** field to **TennisPlayer** model.
* Establishes a **many-to-many** relationship with the TennisPlayer model, allowing multiple tennis players to participate in a match and a tennis player to participate in multiple matches.
* **winner**
  + A **foreign key** to the **TennisPlayer** model.
* Establishes a **many-to-one** relationship with the TennisPlayer model, associating each match with a tennis player, who is the winner.
* **ON DELETE** constraint must be set to **NULL**.
* This field must **be** **able** to contain **null** values.
* Make sure the **plural name** for the **Match** model is set correctly to **Matches** (**hint**: set it in the Meta class).

## 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:

### TennisPlayerAdmin

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

### TournamentAdmin

* **Display fields**: Specify the fields to be displayed in the list view of the admin site for the **Tournament model**.
  + Fields: **'name'**, **'location'**, **'prize\_money'**, **'surface\_type'**, **'start\_date'**
* **Filters**: Add a filter for **'surface\_type'** in the admin site.
* **Search fields**: Enable search by **'name'** and **'location'** in the admin site.

### **MatchAdmin**

* **Display fields**: Specify the fields to be displayed in the list view of the admin site for the **Match model**.
  + Fields: **'date\_played'**, **'score'**, **'summary'**
* **Filters**: Add a filter for the **'date\_played'** in the admin site.
* **Search fields**: Enable search bytournament’s **'name'** (searching matches by tournament's name).

## Custom Model Manager – 20 points

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

get\_tennis\_players\_by\_wins\_count()

This method **retrieves** and **returns** all **tennis player objects**, **ordered by** the **number** of **wins** each tennis player has **descending**, **then by** their **full names ascending**.

## Django Queries I – 75 points

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

get\_tennis\_players(search\_name=None, search\_country=None)

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

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

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

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

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

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

**If there are** tennis player objects that match the criteria, **order** them by **ranking**, **ascending.**

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

**"Tennis Player: {full\_name1}, country: {country1}, ranking: {ranking1}**

**Tennis Player: {full\_name2}, country: {country2}, ranking: {ranking2}**

**…**

**Tennis Player: {full\_nameN}, country: {countryN}, ranking: {rankingN}"**

* If **no tennis players** match the criteria, **return** an **empty string ("")**.

get\_top\_tennis\_player()

This function accepts no arguments.

It **retrieves** the tennis player with the **greatest number** of **wins.**

Ifthere is **more than one tennis player** with the **same number** ofwins, **order** them **by full name, ascending**, and **return** the **first one’s** info.

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

**"Top Tennis Player: {full\_name} with {num\_of\_wins} wins."**

* If there are **no players**, **return** an **empty string ("")**.

get\_tennis\_player\_by\_matches\_count()

This function accepts no arguments.

It **retrieves** the tennis player with the **greatest number** of **matches played.**

Ifthere is **more than one tennis player** with the **same number** ofmatches**, order** them **by ranking, ascending**, and **return** the **first one’s** info.

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

**"Tennis Player: {full\_name} with {num\_of\_matches} matches played."**

* If there are **no matches** or **no players**, **return** an **empty string ("")**.

## Django Queries II – 75 points

get\_tournaments\_by\_surface\_type(surface=None)

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

* **surface** – string value or **None**

It **retrieves** the **tournament objects** whose **surface type matches** the **given** string **partially** and **case-insensitively**. **Order** them by **start date**, **descending**.

**Return** a **string** in the following format. Each tournament’s info on a **new line**:

**"Tournament: {tournament\_name1}, start date: {start\_date1}, matches: {num\_matches1}**

**Tournament: {tournament\_name2}, start date: {start\_date2}, matches: {num\_matches2}**

**…**

**Tournament: {tournament\_nameN}, start date: {start\_dateN}, matches: {num\_matchesN}"**

* **"num\_matches"** represents the **number of matches** each tournament has.
* If there are **no matching** tournaments, **no tournaments**, or the surface **value** is **None**, **return** an **empty string ("")**.

get\_latest\_match\_info()

This function accepts no arguments.

It **retrieves** the **latest** **match** considering its **date played**. If you have **matches** with the **same date and time**, get the **last one** (last id).

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

**"Latest match played on: {date\_played}, tournament: {tournament\_name}, score: {score}, players: {player1\_full\_name} vs {player2\_full\_name}, winner: {"TBA"/winner\_full\_name}, summary: {summary}"**

* Players' **full names** must be **separated** by **" vs "** and **ordered** by **full name**, **ascending**.
* If the **winner** **is None**, return "**TBA**" instead of the winner’s full name.
* If there are **no matches**, return an **empty string** **("")**.

get\_matches\_by\_tournament(tournament\_name=None)

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

* **tournament\_name** – string, representing the **exact** value of the tournament’s name or **None**

It **retrieves** all **matches** by the given **tournament name** (**exact** match) and **orders them by** the **date played**, **descending**.

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

**"Match played on: {date\_played1}, score: {score1}, winner: {"TBA"/winner\_full\_name1}**

**Match played on: {date\_played2}, score: {score2}, winner: {"TBA"/winner\_full\_name2}**

**…**

**Match played on: {date\_playedN}, score: {scoreN}, winner: {"TBA"/winner\_full\_nameN}"**

* If there is **no tournament** with the given **name** (exact match), **no tournaments** at all, **no matches** in the tournament, or the given **name** **is** **None**, **return**:

**"No matches found."**

* If the **winner** **is None**, return "**TBA**" instead of the winner’s full name.

## Testing Data Constraints

* There will always be tennis players and tournaments when creating matches.
* Type of tennis matches: singles (**two players** participate in a match, **one of them** wins). You are **not** supposed to check the number of participating players.
* 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(TennisPlayer.objects.get\_tennis\_players\_by\_wins\_count()) |
| **Output** |
| <QuerySet [<TennisPlayer: Novak Djokovic>, <TennisPlayer: Grigor Dimitrov>, <TennisPlayer: Coco Vandeweghe>, <TennisPlayer: Ella Seidel>]> |
| **Test Code** |
| print(get\_tennis\_players(search\_name='N', search\_country=None)) |
| **Output** |
| Tennis Player: Novak Djokovic, country: SRB, ranking: 1  Tennis Player: Coco Vandeweghe, country: USA, ranking: 300 |
| **Test Code** |
| print(get\_tennis\_players(search\_name='Grigor', search\_country='CA')) |
| **Output** |
|  |
| **Test Code** |
| print(get\_top\_tennis\_player()) |
| **Output** |
| Top Tennis Player: Novak Djokovic with 2 wins. |
| **Test Code** |
| print(get\_tennis\_player\_by\_matches\_count()) |
| **Output** |
| Tennis Player: Novak Djokovic with 3 matches played. |
| **Test Code** |
| print(get\_tournaments\_by\_surface\_type('ha')) |
| **Output** |
| Tournament: Australian Open 2024, start date: 2024-01-15, matches: 0  Tournament: US Open 2023, start date: 2023-08-22, matches: 2 |
| **Test Code** |
| print(get\_tournaments\_by\_surface\_type('noneExistentType')) |
| **Output** |
|  |
| **Test Code** |
| print(get\_latest\_match\_info()) |
| **Output** |
| Latest match played on: 2023-08-31 22:00:00+00:00, tournament: US Open 2023, score: 7:6(7:4) 6:3 6:4, players: Grigor Dimitrov vs Novak Djokovic, winner: Grigor Dimitrov, summary: Stunning! |
| **Test Code** |
| print(get\_matches\_by\_tournament('French Open 2023')) |
| **Output** |
| Match played on: 2023-06-10 19:00:00+00:00, score: 7:6(7:4) 4:3 (suspended), winner: TBA  Match played on: 2023-06-08 19:00:00+00:00, score: 7:6(11:9) 4:6 6:3 6:4, winner: Novak Djokovic |
| **Test Code** |
| print(get\_matches\_by\_tournament('us O')) |
| **Output** |
| No matches found. |