# Exercises: Migrations and Django Admin

This document defines the **exercise assignments** for the [Python ORM course @ Software University](https://softuni.bg/trainings/4253/python-orm-october-2023).

Submit your solutions in the SoftUni [Judge system](https://judge.softuni.org/Contests/4302/Migrations-and-Django-Admin-Exercise).

## Shoe

Write a Django model called **"Shoe"** with the provided information:

* **"brand"** - character field, **consisting of a** **maximum of 25 characters**.
* **"size"** - positive integer field.

**Note: Apply all the migrations to the database before executing the following steps. Also, keep in mind that the exercises with custom data migration will NOT be tested in the Judge system.**

**Show all the migrations**

You can **list** the applied **migrations** with the command:

* "**python manage.py showmigrations -l**"

Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично

As you can see there is a result in the terminal with all the **applied** **migrations**.

**Represent SQL code through migrations**

You can print the generated **SQL** code with the command:

* "**python manage.py sqlmigrate main\_app 0001\_initial**".

**Картина, която съдържа текст, Шрифт, екранна снимка, линия

Описанието е генерирано автоматично**

**Constraints:**

* **"main\_app"** refers to the **application** name.
* **"0001\_initial"** refers to the **migration** name.

Create a **function** that migrates only the **unique** **brands** in a new **model** called "**UniqueBrands**" with the following field:

* "**brand\_name**" - character field, **consisting of a maximum of 25 characters, unique**.

**Note: Apply all the migrations to the database before executing the following steps.**

**Hint:**

First, create a new **empty** migration and name it "**migrate\_unique\_brands**". You can do it with the command:

* **"python manage.py makemigrations main\_app --name migrate\_unique\_brands --empty"**



In the "**0003\_migrate\_unique\_brands.py**" file create a new **function** and give it a name of your preference.

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Описанието е генерирано автоматично

Implement the following code:

Картина, която съдържа текст, Шрифт, екранна снимка

Описанието е генерирано автоматично

Let’s break the code step by step:

* First, we use the "**apps.get\_model()**" method to retrieve the **model** classes for the "**Shoe**" and "**UniqueBrands**" models. "**main\_app**"is the **name** of the app that contains these **models**.
* After that we take the connected database.
* Next, we **fetch** **unique** brand names from the "**Shoe**" model.
* Then, we use the "**values\_list**()" method with the "**flat**=**True**" argument to get a list of **unique** brand names from the "**Shoe**" model. The "**distinct**()" method ensures that we only get distinct brand names.
* Finally, we iterate through the "**unique\_brand\_names**" list and use the "**create()**" method of the "**UniqueBrands**" model to create a new "**UniqueBrands**" object for each **unique** brand **name**. We pass the "**brand\_name**" argument to set the value for the "**brand\_name**" field of the "**UniqueBrands**" object.

We need to add the function inside the operations list.Картина, която съдържа текст, Шрифт, линия, номер

Описанието е генерирано автоматично

We also need to fulfill information inside the "**Shoe**" model table:

Картина, която съдържа текст, екранна снимка, номер, Шрифт

Описанието е генерирано автоматично

After we apply all the **migrations**, the new **table** in the database will be filled with the **unique** data.Картина, която съдържа текст, екранна снимка, Шрифт, номер

Описанието е генерирано автоматично

**Reduce the number of migration files**

You can use the following command to **reduce** the number of **migration** **files** by **combining** them into fewer **files**:

* "**python manage.py squashmigrations main\_app 0003**".

Картина, която съдържа текст, Шрифт, екранна снимка, линия

Описанието е генерирано автоматично

## Event Registration

Write a Django model called **"EventRegistration"** with the provided information:

* **"event\_name"** - character field, **consisting of a** **maximum of 60 characters**.
* **"participant\_name"** - character field, **consisting of a** **maximum of 50 characters**.
* **"registration\_date"** - date field.

Apply all the migrations to the database.

### Methods

##### \_\_str\_\_()

**Return** the **name** of the participant and the **name** of the **event** as a string in the following format:

* **"{participant\_name} - {event\_name}**".

### Register the model in the admin site

Create a new Django **model** called "**EventRegistrationAdmin**" in the "**admin.py"** file. You must **register** the model. It is **optional** to create a **superuser** to navigate easily through the **admin** site.

### Customize the admin site

* The **fields** of the model - **"event\_name"**, **"participant\_name"** and **"registration\_date"** should be **displayed** as **columns**.
* **Filter** the fields of the model by **"event\_name"** and **"registration\_date"**.
* **Search** the records of the model by **"event\_name"** and **"participant\_name"**.

## Movie

Write a Django model called **"Movie"** with the provided information:

* **"title"** - character field, **consisting of a** **maximum of 100 characters**.
* **"director"** - character field, **consisting of a** **maximum of 100 characters**.
* **"release\_year"**- positive integer field.
* **"genre"**- character field, **consisting of a** **maximum of 50 characters**.

Apply all the migrations to the database.

### Methods

##### \_\_str\_\_()

**Return** the **title** and the **director** of the movie as a string in the following format:

* **"Movie "{title}" by {director}"**.

### Register the model in the admin site

Create a new Django **model** called "**MovieAdmin**" in the "**admin.py"** file. You must **register** the model.

### Customize the admin site

* The **fields** of the model - **"title"**, **"director"**, **"release\_year"** and **"genre"** should be **displayed** as **columns**.
* **Filter** the fields of the model by **"release\_year"** and **"genre"**.
* **Search** the records of the model by **"title"** and **"director"**.

## Student

Write a Django model called **"Student"** with the provided information:

* **"first\_name"** - character field, **consisting of a** **maximum of 50 characters**.
* **"last\_name"** - character field, **consisting of a** **maximum of 50 characters**.
* **"age"**- positive integer field.
* **"grade"**- character field, **consisting of a** **maximum of 10 characters**.
* **"date\_of\_birth"** - date field.

Apply all the migrations to the database.

### Methods

##### \_\_str\_\_()

**Return** the **first name** andthe **last name** of the **student** as a string in the following format:

* **"{first\_name} {last\_name}"**.

### Register the model in the admin site

Create a new Django **model** called "**StudentAdmin**" in the "**admin.py"** file. You must **register** the model.

### Customize the admin site

* **Display** the model fields - **"first\_name"**, **"last\_name"**, **"age"** and **"grade"** as columns.
* **Filter** the model fields by **"age"**, **"grade"** and **"date\_of\_birth"**.
* **Search** the model’s records by **"first\_name"**.
* **Organize** the fields of the model into two sections. The first one **"Personal Information"** consists of four fields - **"first\_name"**, **"last\_name"**, **"age"** and **"date\_of\_birth"**. The second one **"Academic Information"** consists of one field - **"grade"**.

## Supplier

Write a Django model called **"Supplier"** with the provided information:

* **"name"** - character field, **consisting of a** **maximum of 100 characters**.
* **"contact\_person"** - character field, **consisting of a** **maximum of 50 characters**.
* **"email"** - email field, **unique**.
* **"phone"**- character field, **consisting of a** **maximum of 20 characters, unique**.
* **"address"** - text field.

Apply all the migrations to the database.

### Methods

##### \_\_str\_\_()

**Return** the **name** and the **phone** of the supplier as a string in the following format:

* **"{name} - {phone\_number}"**.

### Register the model in the admin site

Create a new Django **model** called "**SupplierAdmin**" in the "**admin.py"** file. You must **register** the model.

### Customize the admin site

* The fields of the model - **"name"**, **"email"**, and **"phone"** should be **displayed** as columns.
* **Filter** the model fields by **"name"**, and **"phone"**.
* **Search** the model’s records by **"email"**, **"contact\_person"** and **"phone"**.
* Objects **list per page -** 20**.**
* **Organize** the fields of the model into one section. **"Information"** consists of four fields - **"name"**, **"contact\_person"**, **"email"** and **"address"**.

## Course

Write a Django model called **"course"** with the provided information:

* **"title"** - character field, **consisting of a** **maximum of 90 characters**.
* **"lecturer"** - character field, **consisting of a** **maximum of 90 characters**.
* **"description"** - text field, **consisting of a** **maximum of 200 characters**.
* **"price"** - decimal field, with **maximum** of 10 **digits** and 2 **decimal** **places**.
* **"start\_date"**- date field. Every time a **new** **record** is created the **current time** of the creation of the **record** should be saved.
* **"is\_published"** - boolean field with **default** value **"True"**.

Apply all the migrations to the database.

### Methods

##### \_\_str\_\_()

**Return** the **title** and the **lecturer** of the course as a string in the following format:

* **"{title} - {lecturer}"**.

### Register the model in the admin site

Create a new Django **model** called "**CourseAdmin**" in the "**admin.py"** file. You must **register** the model.

### Customize the admin site

* **Display** the model fields - **"title"**, "**lecturer"**, "**price"**,and **"start\_date"** as columns.
* **Filter** the model fields by **"is\_published"**, and **"lecturer".**
* **Search** the model’s records by **"title"** and **"lecturer"**.
* **Organize** the fields of the model into two sections. The first one **"Course Information"** consists of four fields - **"title"**, **"lecturer"**, **"price"**, **"start\_date"**, and **"is\_published"**. The second one **"Description"** consists of only one field - **"description"**.
* **Read-only** fields **- "start\_date"**.

## Person

Write a Django model called **"Person****"** with the provided information:

* **"name"** - character field, **consisting of a** **maximum of 40 characters.**
* **"age" -** positive integer field.
* **"age\_group"** - character field, **consisting of a** **maximum of 20 characters**, with a default value of "**No age group**".

Apply all the migrations to the database.

### Methods

##### \_\_str\_\_()

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

* **"Name: {name}"**.

### Functions inside the migration files

Create a new function that **migrates** information for the **age** **group** of every person based on their **age**.

* If the **age** is less than or equal to **12** - the **group** is **"Child"**.
* If the **age** is between **13** and **17** - the **group** is **"Teen"**.
* If the **age** is greater than or equal to **18** - the **group** is **"Adult"**.

Apply all the migrations to the database.

## Item

Write a Django model called **"Item"** with the provided information:

* **"name"** - character field, **consisting of a** **maximum of 100 characters.**
* **"price"** - decimal field, with **a maximum of 10 digits, 2 decimal places.**
* **"quantity"** - positive integer field, with a **default** value of **1**.
* **"rarity"** - character field, **consisting of a** **maximum of 20 characters**, with a **default** value of "**empty**".

Apply all the migrations to the database.

### Functions inside the migration files

Create a new function that applies a new value to the **rarity** of the **item** based on the **price** of the **item**.

* If the **price** is less than or equal to **10** - the **rarity** is **"Rare"**.
* If the **price** is between **11** and **20** (**both inclusive**) - the **rarity** is **"Very Rare"**.
* If the **price** is between **21** and **30** (**both inclusive**) - the **rarity** is **"Extremely Rare"**.
* If the **price** is greater than or equal to **31** - the **rarity** is **"Mega Rare"**.

Apply all the migrations to the database.

## Smartphone

Write a Django model called **"Smartphone"** with the provided information:

* **"brand"** - character field, **consisting of a** **maximum of 100 characters.**
* **"price"** - decimal field, with **a maximum of 10 digits, 2 decimal places,** and **a default** valueof **0** (**zero**)**.**
* **"category"** - character field, **consisting of a** **maximum of 20 characters,** with **default** value **"empty".**

Apply all the migrations to the database.

### Functions inside the migration files

Create two functions that generate new values for the **price** and the **category** fields:

* The first function generates a new **price** based on the **brand's** **length** multiplied by **120**. **Ensure that you've applied the required migrations before proceeding to create the next function.**
* The second one generates a new value - **"Expensive"** to the **category** field only if the **price** is greater than or equal to **750**. Otherwise, the new **value** should be **"Cheap"**.

Apply all the migrations to the database.

## Order

Write a Django model called **"Order"** with the provided information:

* **"product\_name"** - character field, **consisting of a** **maximum of 30 characters.**
* **"customer\_name"** - character field, **consisting of a** **maximum of 100 characters.**
* **"order\_date"-** date field.
* **"status"** - character field, **consisting of a** **maximum of 30 characters** with **choices** - **"Pending"**, **"Completed"**, or "**Cancelled"**.
* **"amount"** - positive integer field with **default** value **1**.
* **"product\_price"** - decimal field with a **maximum of 10 digits** and **2 decimal places.**
* **"total\_price"** - decimal field with a **maximum** of **10 digits,** 2 **decimal** **places**, with a **default** value of 0.
* **"warranty"** - character field, with a **default** value of "**No warranty**".
* **"delivery"** - date field, **optional**.

Apply all the migrations to the database.

### Methods

##### \_\_str\_\_()

**Return** the order **id** and the **name** of the customer as a string in the following format:

* **"Order #{order\_id} - {customer\_name}"**

### Functions inside the migration files

Create a function that modifies the **delivery** and **warranty** of all orders based on their **status**:

* If the order **status** is **"Pending"**, then the **delivery** is 3 **days** after the order **date**.
* If the order **status** is **"Completed"**, then the **warranty** should change to **"24 months"**.
* If the order **status** is **"Cancelled"**, then **remove** the **order** from the database.

Apply all the migrations to the database.