# Exercises: Django Models Basics

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/4300/Django-Models-Basics-Exercise).

## Person

Write a Django model called **"Person"** which stores information about their **name** and **age**:

* **"name"**:
* **Character field**.
* Every "**name**" has a length with a **maximum of 30 characters**.
* **"age"**:
* **Positive integer field.**

### Hint:

First, let us open and configure the **Ready-to-use Skeleton**. After all the packages are installed, we need to start to write the model in our application - **"main\_app"**. To begin with, we need to ensure that the **virtual environment** is turned on. We can turn it on with the command: **"venv\Scripts\activate"**.

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

Описанието е генерирано автоматично

Once our virtual environment is ready, we proceed to **install the required dependencies** listed in the **requirements.txt** file. This is accomplished by **entering** the **command** **"py -m pip install -r requirements.txt"** in the terminal:

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

Описанието е генерирано автоматично

Now, let us open the **"models**.**py"** file positioned in the **"main\_app"** application, where we will define our **model**. First, we create a class called **"Person"** which subclasses the built-in Django **"models.Model" class**. In **Django**, a model represents a **database** table, where each attribute of the model maps to a column in that table. Consequently, when you create an instance of the **"Person"** **class**, it will be saved as a record/entity (a row) within the **database**:

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

Описанието е генерирано автоматично

Next, let us define the attribute called "**name"** and assign it an instance of the "**models.CharField**" class. This field corresponds to a **varchar** column in the database table and is responsible for storing the person's name.

**Note**: when defining the field, it is necessary to specify the **maximum** **length** of this field using the "**max\_length**" parameter.

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

Описанието е генерирано автоматично

Finally, let us define the attribute called **"age"** and assign it an instance of the "**models.PositiveIntegerField**" class. This field corresponds to an integer column with a constraint that it should only accept positive values and is responsible for storing the person's **age**.

**Note**: unlike the "**CharField**", there are no mandatory parameters that need to be specified for this field:

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

Описанието е генерирано автоматично

In order for our **model** to function properly, it is necessary to establish our database and implement certain **migrations**.

To set up the database:

* Select the appropriate database for our needs, in this case, "**PostgreSQL**".
* Locate the "**settings.py"** file and navigate to the **DATABASES** dictionary.
* By default, the database is set to "**sqlite3**", which needs to be modified. Please refer to the provided link for detailed instructions on making the necessary changes.

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

Описанието е генерирано автоматично

The configuration should look like this:Картина, която съдържа текст, екранна снимка, Шрифт, номер

Описанието е генерирано автоматично

**Note, it is necessary to edit the NAME, USER, and PASSWORD information with your personal!**

To establish a server for the database, we can utilize PyCharm. Follow the steps below:

* In PyCharm, locate the top right corner of the screen and click on the "**Database**" option.

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

Описанието е генерирано автоматично

Картина, която съдържа текст, екранна снимка, дисплей, софтуер

Описанието е генерирано автоматично

From the available options, select "**PostgreSQL**". Also, provide the necessary **personal information** and verify if the connection is successful.

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

Описанието е генерирано автоматично

Now, let's create the actual database. First, Initialize the name of the database in "**settings.py**".

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

Описанието е генерирано автоматично

Then right-click on the server, select "**Database**," then choose "**New**" and assign a name to the database. For instance, we can name it **"models\_db"**.

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

Описанието е генерирано автоматично

You will notice that PyCharm generates authentic **SQL** code, and you also have the option to write custom **SQL** **queries**.

To ensure that the generated tables are visible later on, make the "**public**" schema visible.

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

Описанието е генерирано автоматично

* Remember to establish a connection between the database and the **DATABASES** dictionary in the "**setings.py**" file. Additionally, ensure that you provide your **username** and **password** as required.

Now, we are ready to apply some **migrations**.

When you first create the "**Person**" **class** and run the initial migration, Django will generate a migration file that represents the initial state of the "**Person**" model. This migration file contains instructions on how to create the corresponding database table and columns based on the model's fields ("**name**" and "**age**").

To manage migrations in **Django**, you can use two essential commands:

* **"python manage.py makemigrations"**: This command is used to generate migration files based on the changes you've made to your models. It examines the differences between the current state of your models and the previous state recorded in existing migrations, creating new migration files to capture these changes.
* **"python manage.py migrate"**: After using "**makemigrations"** to create migration files, you use this command to apply those migrations and update the database schema. It executes any pending migrations that haven't been applied to the database, keeping track of which migrations have already been applied to ensure only the necessary ones are executed.

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

Описанието е генерирано автоматично

You will see an **error** like this:

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

Описанието е генерирано автоматично

"**Psycopg2**" is a "**PostgreSQL**" adapter for Python, which allows Python programs to interact with a "**PostgreSQL**" database. It is a widely used library that provides an efficient and easy-to-use interface for working with "**PostgreSQL**" databases in Python. To connect Django to the database we need to execute the following command:

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

Описанието е генерирано автоматично

Finally, we can apply the **migrations**. They are created in the "**migrations**" folder inside its application.

To transfer all the migrations to the database, we must run the command: "**python manage.py migrate**"

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

Описанието е генерирано автоматично

When you run the **"migrate"** command, Django reads the migration files and compares them with the current state of the database. It then determines which migrations need to be applied in order to bring the database schema up to date.

After applying the command and reloading the database, inside the **public** **schema**, you can see several generated tables. They are the applied migrations.

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

Описанието е генерирано автоматично

Upon selecting the **"main\_app\_person"** table, you will have visibility into all the stored data within it. At this point, we can proceed to create a new entity by adding a record with the "**name**" "**Kaloyan**" and the "**age**" "**18**". It's worth noting that the **"id"** field will be automatically generated by the system, ensuring a **unique** identifier for the new entry.

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

Описанието е генерирано автоматично

## Blog

Write a Django model called **"Blog"** which stores information about the blog’s **"post"** and **"author"**:

* **"post"**:
* **Text field**.
* **"author"**:
* **Character field**.
* Every **"author"** has a **length** with a maximumof **35 characters**.

Apply all the migrations to the database.

## Weather Forecast

Write a Django model called **"WeatherForecast"** with the following parameters:

* **"date"**:
* **Date field.**
* **"temperature"**:
* **Float field.**
* **"humidity"**:
* **Float field.**
* "**precipitation**":
* **Float field.**

Apply all the migrations to the database.

## Recipe

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

* **"name"**:
* **Character field.**
* Every **"name"** has a **length** with a maximumof **100 characters**.
* Also, each "**name**" must be **unique**.
* **"description"**:
* **Text field.**
* **"ingredients"**:
* **Text field.**
* **"cook\_time"**:
* **Positive integer field.**
* **"created\_at"**:
* **Date time field.**
* Every time a **new** **record** is created the **current time** of the creation of the **record** should be saved.

Apply all the migrations to the database.

## Product

Write a Django model called **"Product"** with the following parameters:

* **"name"**:
* **Character field**.
* Every **"name"** has a **length** with a maximumof **70 characters**.
* **"description"**:
* **Text field.**
* **"price"**:
* **Decimal field**.
* Every **"price"** has **a** maximumof **10 digits** and **2 decimal places**.
* **"created\_at"**:
* **Date time field.**
* Every time a **new** **record** is created the **current time** of the creation of the **record** should be saved.

Apply all the migrations to the database.

## User Profile

Write a Django model called **"UserProfile"** with the following parameters:

* **"username"**:
* **Character field**.
* Each **"username"** has a **length** with a maximumof **65 characters**.
* Also, make the "**username**" **unique**.
* **"first\_name"**:
* **Character field**.
* Each **"first\_name"** has a **length** with a maximumof **40 characters**.
* Make the "**first\_name**" **unique**.
* **"last\_name"**:
* **Character field**.
* Each **"last\_name"** has a **length** with a maximumof **40 characters**.
* Make the "**last\_name**" **unique**.
* **"email"**:
* **Email field**.
* Each **"email"** is **unique**,
* Set a **default** value – **"**[**students@softuni.bg**](mailto:students@softuni.bg)**"**.
* **"bio"**:
* **Text field.**
* Each **"bio"** has a **length** with a maximumof **120 characters**.
* **"profile**\_**image**\_**url"**:
* **URL field**.
* **"created\_at"**:
* **Date time field.**
* Every time a **new** **record** is created the **current time** of the creation of the **record** should be saved.

Apply all the migrations to the database.

## Exercise

Write a Django model called **"Exercise"** with the following parameters:

* **"name"**:
* **Character field.**
* Each **"name"** has a **length** with a maximumof **100 characters**.
* **"description":**
* **Text field.**
* **"difficulty\_level"**:
* **Character field.**
* Each **"difficulty\_level"** has a **length** with a maximumof **20 characters**.
* **"duration\_minutes"**:
* **Positive integer field.**
* **"equipment"**:
* **Character field.**
* Each **"equipment"** has a **length** with a maximumof **90 characters**.
* **"video\_url"**:
* **URL field.**
* This **field** is **not required.**
* **"calories\_burned**":
* **Positive integer field.**
* The field has a **default value of 0** (zero).
* **"is\_favorite"**:
* **Boolean field**.
* The field has a **default value of "False"**.

Apply all the migrations to the database.

## \*Book

Write a Django model called **"Book"** with the following parameters and **register the model in the Django admin site**.

* **"title"**:
* **Character field.**
* Each **"title"** has a **length** with a maximumof **30 characters**.
* **"author"** –
* **Character field.**
* Each **"author"** has a **length** with a maximumof **100 characters**.
* **"genre"**:
* **Character field.**
* Each **"genre"** has a **length** with a maximumof **20 characters**.
* The **choices** are – **"Fiction"**, **"Non-Fiction"**, **"Science Fiction"**, and **"Horror"**.
* **"publication\_date"**:
* **Date field.**
* The "**date**" should **not** be **editable**.
* Every time a **new** **record** is created the **current time** of the creation of the **record** should be saved.
* **"price"**:
* **Decimal field.**
* Every **"price"** has **a** maximumof **8 digits** and **2 decimal places**.
* **"is\_available"**:
* **Boolean field.**
* The **default** value of the field is **"True"**.
* **"rating"**:
* **Float field.**
* **"description"**:
* **Text field.**

Apply all the migrations to the database.

### Methods

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

* **The "\_\_str\_\_()"** methodshould **return** the **"title"** as a single string.

### Hint:

To get started, make sure to register your model in the **"admin.py"** file within the "**main\_app**" application.

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

Описанието е генерирано автоматично

Then you need to run this command:

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

Описанието е генерирано автоматично

Executing this command will create a **"super** **user"** for the application. A super user has complete control over the CRUD operations within the application. Enter your preferred **"username"**, **"email"**, and **"password"**, and you'll be able to access the following URL: [**http://127.0.0.1:8000/admin/login/?next=/admin/**](http://127.0.0.1:8000/admin/login/?next=/admin/)**.**

Once you've successfully **logged** in as the super user, you will assume the role of the application's **"administrator"**.

When an **object** is created the **"\_\_str\_\_()"** method will **return** the objects displayed with every book’s **"title"**.

As you can see, we have already created an object of the class **"Book"**.

Картина, която съдържа текст, екранна снимка, софтуер, дисплей

Описанието е генерирано автоматично

Every **created** **object** is displayed with the book’s **"title"**.

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

Описанието е генерирано автоматично