**Register Model Class to Show in Admin Panel Django 5**

## **1. Why Register a Model in Django Admin?**

Django's admin panel only displays models that you **explicitly register** in admin.py.  
By default, even if you create models in models.py, they won’t appear in the admin site until you **register** them.

## **2. Example Code Recap**

You have this:

**models.py**

**from** django**.**db **import** models

class **profile**(*models***.***Model*)**:**

    name **=** models**.**CharField(**max\_length=**50)

    email **=** models**.**EmailField(**max\_length=**254)

    city **=** models**.**CharField(**max\_length=**50)

    roll **=** models**.**IntegerField()

*# If we want to display name in admin list*

*# def \_\_str\_\_(self):*

*#     return self.name*

*# If we want to display roll number (integer)*

**def** \_\_str\_\_(*self*)**:**

**return** str(*self***.**roll)

class **Result**(*models***.***Model*)**:**

    stu\_class **=** models**.**CharField(**max\_length=**70)

    marks **=** models**.**IntegerField()

**admin.py // main registeration are here**

**from** django**.**contrib **import** admin

**from** student**.**models **import** profile**,** Result

#registration are here

admin**.**site**.**register(profile)

admin**.**site**.**register(Result)

## **3. Steps to Register a Model in Admin Panel**

### **Step 1: Create Model in models.py**

* Each table in your database is represented by a **model class**.
* Example:
  + profile table → stores student name, email, city, roll number.
  + Result table → stores class name and marks.

### **Step 2: Add** \_\_str\_\_() **Method**

* The \_\_str\_\_() method **controls what text is shown** when a model object is displayed (for example, in Django admin lists, shell, etc.).
* **If you don’t define it** → Django will display something like profile object (1) which isn’t very descriptive.

Example 1: Returning a string field

**def** \_\_str\_\_(*self*)**:**

**return** *self***.**name

Output in admin list:

John

Mary

Alex

Example 2: Returning an integer field (must convert to string)

**def** \_\_str\_\_(*self*)**:**

**return** str(*self***.**roll)

Output in admin list:

1

2

3

💡 **Note**:

* \_\_str\_\_() **must return a string**.
* If you want to display multiple fields, you can format them:

**def** \_\_str\_\_(*self*)**:**

**return** f"{*self***.**name} - Roll {*self***.**roll}"

### **Step 3: Register Model in admin.py**

* The admin.py file tells Django which models should be visible in the **Admin interface**.
* admin.site.register(ModelName) registers the model.

Example:

**from** django**.**contrib **import** admin

**from** student**.**models **import** profile**,** Result

admin**.**site**.**register(profile)  *# Shows Profile table in admin*

admin**.**site**.**register(Result)   *# Shows Result table in admin*

## **4. After Registration**

* Run the server:
* python manage**.**py runserver
* Go to:
* http**://**127.0**.**0**.**1**:**8000**/**admin
* Login with **superuser** credentials.
* You’ll now see **Profile** and **Result** in the admin panel.
* You can **add, update, delete, and search** records directly from there.

## **5. Summary Table for Notes**

| **Part** | **Purpose** |
| --- | --- |
| models.py | Defines database tables (models). |
| \_\_str\_\_() method | Controls how the model object is displayed in admin or shell (must return a string). |
| admin.py | Registers models with admin.site.register() so they appear in admin panel. |
| profile model | Stores student personal info. |
| Result model | Stores class and marks. |

✅ **Pro Tip**:  
You can customize admin display (fields shown, search, filters) using **ModelAdmin** class instead of simple registration:

class **ProfileAdmin**(*admin***.***ModelAdmin*)**:**

    list\_display **=** ('name'**,** 'email'**,** 'roll')

admin**.**site**.**register(profile**,** ProfileAdmin)

This will show columns for name, email, and roll in the admin list view.