# 📘 ****Django Admin Panel – Displaying Database Data****

## 🔹 1. What is Django Admin?

* Django comes with a **built-in admin panel**.
* It allows developers (and superusers) to **view, add, edit, and delete data** from models.
* Instead of writing SQL queries, you manage your data via a friendly UI.
* It automatically detects your **models** and creates forms.

## 🔹 2. Example Models (models.py)

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

*# Student profile model*

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

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

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

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

    roll **=** models**.**IntegerField()              *# student roll number*

**def** \_\_str\_\_(*self*)**:**   *# important for readable names in admin panel*

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

*# Student result model*

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

    stu\_class **=** models**.**CharField(**max\_length=**70) *# e.g. "10th Class"*

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

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

**return** f"{*self***.**stu\_class} - {*self***.**marks}"

👉 **Why add \_\_str\_\_?**  
When you see objects in the admin panel, instead of "profile object (1)", you will see "Talha" (name of student).  
This makes admin easier to read.

## 🔹 3. Register Models in admin.py

By default, models are **not visible** in admin.  
We must **register** them in admin.py.

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

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

*# ✅ Customizing admin for profile model*

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

*# fields you want to show in the list view (table format)*

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

*# extra options you can add:*

    search\_fields **=** ('name'**,**'city')          *# adds a search box*

    list\_filter **=** ('city'**,**)                  *# adds sidebar filter*

    ordering **=** ('name'**,**)                     *# orders by name*

    list\_per\_page **=** 10                       *# pagination (10 records per page)*

*# Register profile with its custom admin class*

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

*# ✅ Using decorator for Result model*

@admin**.**register(Result)

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

*# Display ID and Class*

    list\_display **=** ('id'**,**'stu\_class'**,**'marks')

*# more options here also possible*

    search\_fields **=** ('stu\_class'**,**)

    ordering **=** ('marks'**,**)

## 🔹 4. Explanation of Code

* list\_display → decides which fields you see in the **table list view**.
* search\_fields → enables search box (search inside selected fields).
* list\_filter → creates filters on right side (e.g., filter by city).
* ordering → defines default ordering (ascending/descending).
* list\_per\_page → breaks large data into pages.

## 🔹 5. Admin Panel Usage

1. Run your server:

python manage.py runserver

1. Go to:

http://127.0.0.1:8000/admin/

1. Login with **superuser** (created earlier).
2. You will see your app (student) and registered models (profile, Result).
3. You can now **Add / Edit / Delete / View** data.

## 🔹 6. Extra Features of Django Admin

Besides list\_display, there are more powerful methods:

* **list\_editable** → make fields editable directly in the list view.

list\_editable **=** ('city'**,** 'roll')

* **readonly\_fields** → makes fields uneditable.

readonly\_fields **=** ('email'**,**)

* **fieldsets** → group fields into sections in detail view.

fieldsets **=** (

    ('Basic Info'**,** {'fields'**:** ('name'**,**'email')})**,**

    ('Other Info'**,** {'fields'**:** ('city'**,**'roll')})**,**

)

* **list\_display\_links** → makes some fields clickable.

list\_display\_links **=** ('name'**,**)

## 🔹 7. Theoretical Information (for Notes)

* Django Admin is an **auto-generated backend interface**.
* Admin panel works with **Model + ModelAdmin class**.
* Steps to use:
  1. Create a **model** in models.py.
  2. Run migrations (makemigrations + migrate).
  3. Register model in admin.py using admin.site.register() or @admin.register.
  4. Customize with ModelAdmin class.
* Only **superusers/staff** can access the panel.
* Useful for **managing database** without writing queries.
* Not recommended for **public users** (only developers & staff).

✅ So in short, your code:

* Created two models (profile, Result).
* Registered them in admin.py.
* Customized list display (columns).
* Used **class-based registration** (for profile) and **decorator style** (for Result).