## **Retrieve Data from Database Table in Django 5**

In Django, we use the **ORM (Object Relational Mapping)** to fetch data from database tables.  
The ORM allows us to query the database using Python methods instead of writing SQL.

## **1. Example Setup**

We have a **model.py**:

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

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

    name **=** models**.**CharField(**max\_length=**70)

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

    city **=** models**.**CharField(**max\_length=**70)

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

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

### 🔹 Role of \_\_str\_\_ Method in Django Models

When you define a model in Django, for example:

{**%** include 'components/welcome.html' **with** name="Special Name for About" only **%**}**from** django**.**db **import** models

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

    name **=** models**.**CharField(**max\_length=**100)

    email **=** models**.**EmailField()

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

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

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

#### 🔸 What it does?

* The \_\_str\_\_ method defines the **human-readable string representation** of the object.
* By default, if you don’t define \_\_str\_\_, Django will display something like:
* Student object (1)
* Student object (2)
* With \_\_str\_\_ defined as return self.name, Django will instead display:
* Ali
* Sara
* Talha
* This is especially useful in:
  + **Django Admin Panel** → so you see actual names, not "object (1)".
  + **Shell / Debugging** → printing the object shows meaningful info.

### 🔹 Example Without \_\_str\_\_

s **=** Student**.**objects**.**get(**id=**1)

print(s)

*# Output: Student object (1)*

🔹 Example With \_\_str\_\_

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

    name **=** models**.**CharField(**max\_length=**100)

    email **=** models**.**EmailField()

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

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

**return** f"{*self***.**name} - {*self***.**roll\_no}"

Now**:**

s **=** Student**.**objects**.**get(**id=**1)

print(s)

*# Output: Ali - 101*

✅ **In short:**  
The role of \_\_str\_\_ is to **make the object’s representation more readable** by returning a string (usually a field value like name, title, etc.).

## **2. Fetching Multiple Records —** .all()

In views.py:

**from** django**.**shortcuts **import** render

**from** student**.**models **import** Studentsdata

**def** alldata(**req**)**:**

    stu **=** Studentsdata**.**objects**.**all()  *# Fetch ALL rows from the table*

    print(stu)  *# QuerySet object*

**return** render(req**,** 'student/all.html'**,** {'studentdatastore'**:** stu})

**Explanation**:

* Studentsdata.objects.all() returns a **QuerySet** containing all rows from the table.
* The QuerySet can be **looped over** in the template.
* In the template, we use:
* {**%** **for** stuu **in** studentdatastore **%**}
* {{ stuu**.**name }} **<**br**>**
* {{ stuu**.**email }} **<**br**>**
* {{ stuu**.**city }} **<**br**>**
* {**%** endfor **%**}
* If there is no data, the {% if studentdatastore %} block will show "NO DATA".

## **3. Fetching a Single Record —** .get()

In views.py:

**def** singlestudentdata(**req**)**:**

    stusingl **=** Studentsdata**.**objects**.**get(**pk=**1)  *# Fetch row where primary key = 1*

    print(stusingl)

**return** render(req**,** 'student/single.html'**,** {'studentsingle'**:** stusingl})

**Explanation**:

* .get() is used to fetch **exactly one record**.
* If no record is found → DoesNotExist error.
* If more than one record matches → MultipleObjectsReturned error.
* Common parameters in .get():
  + pk=value → primary key
  + id=value → same as pk
  + fieldname=value → filter by a specific field (e.g., name="Ali").

Example:

student **=** Studentsdata**.**objects**.**get(**email=**"test@gmail.com")

## **4. Common ORM Methods for Retrieval**

| **Method** | **Description** | **Example** |
| --- | --- | --- |
| .all() | All rows | Studentsdata.objects.all() |
| .get() | Single row (exact match) | Studentsdata.objects.get(pk=1) |
| .filter() | Multiple rows matching condition | Studentsdata.objects.filter(city="Lahore") |
| .exclude() | All rows except those matching condition | Studentsdata.objects.exclude(city="Lahore") |
| .order\_by() | Sort results | Studentsdata.objects.order\_by('name') |
| .first() | First row from QuerySet | Studentsdata.objects.filter(city="Lahore").first() |
| .last() | Last row from QuerySet | Studentsdata.objects.last() |
| .count() | Count matching rows | Studentsdata.objects.filter(city="Lahore").count() |
| .exists() | Check if any match exists (True/False) | Studentsdata.objects.filter(city="Lahore").exists() |

## **5. Passing Data to Template**

* Data is passed as a **dictionary** in render():

**return** render(req**,** 'student/all.html'**,** {'studentdatastore'**:** stu})

* In the template:

{**%** **for** stuu **in** studentdatastore **%**}

    {{ stuu**.**name }}

{**%** endfor **%**}

* For single object:

{{ studentsingle**.**name }}

{{ studentsingle**.**email }}

## **6. Error Handling for** .get()

Since .get() throws an error if data is not found, we should handle exceptions:

**from** django**.**http **import** Http404

**def** singlestudentdata(**req**)**:**

**try:**

        stusingl **=** Studentsdata**.**objects**.**get(**pk=**1)

**except** Studentsdata**.**DoesNotExist**:**

**raise** Http404("Student not found")

**return** render(req**,** 'student/single.html'**,** {'studentsingle'**:** stusingl})

## **7. URL Mapping**

student/urls.py

**from** django**.**urls **import** path

**from** student**.**views **import** alldata**,** singlestudentdata

urlpatterns **=** [

    path('all/'**,** alldata)**,**

    path('single/'**,** singlestudentdata)**,**

]

## **8. Summary Table**

| **Method** | **Returns** | **When to Use** |
| --- | --- | --- |
| .all() | QuerySet (multiple rows) | Fetch all data |
| .get() | Single object | Fetch exactly one record |
| .filter() | QuerySet | Fetch multiple rows with condition |
| .exclude() | QuerySet | Fetch data excluding certain condition |
| .order\_by() | QuerySet | Sort data |
| .first() / .last() | Single object | Get first/last row |
| .count() | Integer | Get number of rows |
| .exists() | Boolean | Check if data exists |