🗓️ Day 6: Update & Delete (CRUD Complete)

Goal: Learn how to Edit (Update) existing data and Remove (Delete) unwanted data using **IDs**.

🔑 The Key Concept: Dynamic URLs

To touch a specific record in the database, we need its **ID (Primary Key)**.

Our URLs will look like this:

- `/delete/1/` -> Delete Student with ID 1
- `/update/5/` -> Update Student with ID 5

Part 1: Delete Operation

Deleting is simple: Find the student by ID -> Delete -> Go back home.

1. Views Logic (views.py)

```
def delete_student(request, id):
    # 1. Get the specific student from DB
    student = Student.objects.get(id=id)

    # 2. Delete the record
    student.delete()

    # 3. Go back to Home
    return redirect('home')
```

Code Line	Explanation

`def delete_student(request, id):`	We pass `id` as a parameter because the URL will send a number (e.g., 5) to this function.
`Student.objects.get(id=id)`	**get()** fetches only ONE item. It says: "Find the student whose database ID matches the ID passed in the URL".
`student.delete()`	This command permanently removes that row from the database table.

2. URL Configuration (urls.py)

We use `<int:id>` to tell Django that a number will be passed here.

```
urlpatterns = [
    # ... other paths ...
    path('delete/<int:id>/', views.delete_student, name='delete_data'),
]
```

3. Template Link (home.html)

We must pass the ID of the student inside the link.

```
<a href="{% url 'delete_data' s.id %}">Delete</a>
```

Part 2: Update Operation

Updating is tricky because we need to open the form **Pre-filled** with old data.

1. Views Logic (views.py)

```
def update_student(request, id):
    # 1. Fetch old data
    student = Student.objects.get(id=id)

    # 2. Fill the form with old data (instance=student)
    form = StudentForm(request.POST or None, instance=student)
```

```
    # 3. Save if valid
    if form.is_valid():
        form.save()
        return redirect('home')

    return render(request, 'myapp/update.html', {'form': form})
```

Code Line	Explanation
`request.POST or None`	This is a shortcut. It means: "If there is new data (POST), take it. If not, keep it empty/None".
`instance=student`	**The Magic Line!** It tells the Form: "Don't create a new blank form. Fill the boxes with this student's existing data".
`form.save()`	Because we used `instance=`, Django knows it should **UPDATE** the existing row, not Create a new one.

2. URL Configuration (urls.py)

```
path('update/<int:id>/', views.update_student, name='update_data'),
```

3. Create Update Template (update.html)

Create a new file `update.html`. It looks exactly like `add.html`.

```
<!DOCTYPE html>
<html>
<body>
    <h1>Edit Student Details</h1>
    <form method="POST">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Update Changes</button>
    </form>
</body>
</html>
```

4. Template Link (home.html)

Add the Edit button next to the Delete button.

```
<a href="{% url 'update_data' s.id %}">Edit</a>
```