Validation in DRF

In Django, validation refers to the process of checking that the data submitted by a user (e.g. through a form or API) is correct, complete, and in the expected format before it is saved to the database or used in your application.

Validation ensures data integrity, security, and user-friendly feedback.

Types of validation in Django

1. Field-Level Validation

Validation applied to a single field (one form or model field at a time). In ModelForms or Forms using clean_(). Automatically enforced by field types and attributes (e.g., max_length, blank=False).

You are validating individual values (e.g. age, username format, email validity).

Logic doesn't depend on other fields.

2. Object-Level (Form or Model-Level) Validation

Validation that involves multiple fields together or the object as a whole.

- In Forms/ModelForms via clean()
- In Models via clean()

You need to compare or cross-check multiple fields (e.g., passwords match, discount < price).

The validation logic depends on more than one field.

3. Validator Functions and Classes

Reusable validation logic that can be attached to fields.Directly on model fields using validators=[...]

You want to apply the same validation in multiple places. The logic is self-contained and doesn't depend on other fields. You need parameters (use class-based validators for this).

Which One Should Prefer

Goal	Prefer This Validation Type
Validate one field's format/value	Field-Level (`clean_<field>()`)
Compare two or more fields	Object-Level (`clean()`)
Reuse logic across models/forms	Validator Function or Class
Enforce model integrity on save	Model `clean()` + `full_clean()`