TRAINING DAY5 REPORT

28 JUNE 2025

What is a Django Model?

In Django, a Model is a Python class that defines the structure of a database table. Each model maps to a single table in your database, and each attribute of the model represents a field (column) in that table.

Models are defined inside a Django app's models.py file and are subclasses of django.db.models.Model.

Uses of Django Models:

1. Database Abstraction:

   Models abstract the database layer. You define models using Python classes, and Django translates them into SQL queries.

2. CRUD Operations:

   With models, you can easily perform Create, Read, Update, and Delete operations using Django's ORM.

3. Relationships:

   Models allow defining relationships between tables using fields like:

   - Foreign Key (One-to-Many)
   - Many To Many Field
   - One To One Field

4. Validations and Constraints:

   You can add validations such as max_length, unique, blank, null, and custom validators directly to model fields.

5. Auto-Generate Admin Interface:

   Models automatically integrate with Django's admin panel, allowing for quick data management.

6. Querying Made Easy:

   Django's ORM allows you to query your data using Python methods

   like .filter(), .get(), .exclude(), etc.

Common Data Types in Django Models:

Django provides a wide variety of field types that are mapped to database column types. Below is a categorized list of frequently used fields.

1. Text Fields:

Field	Description
CharField(max_length=…)	Short strings, like names or titles
TextField()	Large text bodies like descriptions or comments

2. Numeric Fields

Field	Description
IntegerField()	Integer numbers

FloatField()	Floating-point numbers
DecimalField(max_digits, decimal_places)	Fixed-precision decimals (e.g. for money)
PositiveIntegerField()	Only positive integers

3. Date and Time Fields

Field	Description
DateField()	Date only
TimeField()	Time only
DateTimeField(auto_now_add=True)	Date and time (record creation time)
DateTimeField(auto_now=True)	Date and time (record update time)

4. Boolean Fields

Field	Description
BooleanField()	True/False
NullBooleanField()	True/False/None (deprecated in newer versions)

5. File & Image Fields

Field	Description
FileField(upload_to='...')	For uploading files
ImageField(upload_to='...')	Specifically for image files (requires Pillow)

6. Relational Fields

Field	Description
ForeignKey(Model, on_delete=models.CASCADE)	One-to-many relationship
ManyToManyField(Model)	Many-to-many
OneToOneField(Model, on_delete=models.CASCADE)	One-to-one

7. Miscellaneous Fields

Field	Description
EmailField()	Validates email addresses
URLField()	For URLs
SlugField()	Used for SEO-friendly URLs
JSONField()	Stores structured JSON data

Features of Django Models:

1. Simple Syntax

   Models are defined using Python classes with clean, human-readable syntax.

2. Integrated ORM

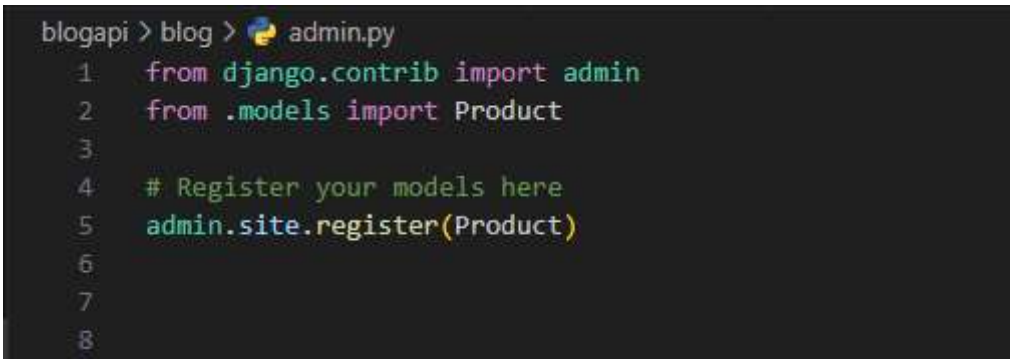
   Django models work with the ORM to perform all SQL operations in a Pythonic way.

3. Automatic Table Creation

   Django automatically creates database tables for models using makemigrations and migrate.

4. Admin Interface Support

   Models are registered in the Django Admin site using:



   This gives a full UI to create, update, and delete records.

5. Model Inheritance

   You can use abstract base classes or multi-table inheritance to reuse fields across models.

blogapi > blog > models.py > ...

```python
from django.db import models

# Create your models here.
class BaseModel(models.Model):
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        abstract = True

class Product(BaseModel):
    name = models.CharField(max_length=100)

```