**Django Model**

A **model** in Django is a Python class that represents a database table. It defines the structure of the database, including fields and behaviors.

**Purpose of models.Model**

When defining a model in Django, inheriting from models.Model provides built-in functionality like database table creation, data validation, and ORM capabilities.

**Common Field Types in Django Models**

- CharField – For short text (e.g., names).
- TextField – For long text.
- IntegerField – For integer values.
- FloatField – For decimal values.
- BooleanField – For True/False values.
- DateTimeField – For date and time values.
- ForeignKey – For many-to-one relationships.
- ManyToManyField – For many-to-many relationships.

**Purpose of ForeignKey Field**

A ForeignKey in Django models establishes a **many-to-one** relationship between two tables. It helps link related data efficiently.
Example:

python

```
class Author(models.Model):
  name = models.CharField(max_length=100)


class Book(models.Model):
  title = models.CharField(max_length=200)
  author = models.ForeignKey(Author, on_delete=models.CASCADE)
```

This means each Book is linked to a single Author, but an Author can have multiple Books.

**Django ORM (Object-Relational Mapping)**

Django ORM allows developers to interact with the database using Python code instead of SQL queries. It automatically translates model operations into SQL statements. Example:

python

```
# Create and save an object

author = Author(name="John Doe")

author.save()


# Querying the database

books = Book.objects.filter(author=author)
```