Django ORM - Notes

ORM (Object-Relational Mapping) - Django

What is ORM?
ORM is a technique that lets you interact with a database using Python code instead of writing raw SQL.

Why Use ORM?

Feature	Benefit
Pythonic	Use Python code instead of SQL
Secure	Prevents SQL injection
Faster Development	Less boilerplate, more readable
Structure-Friendly	Follows your Django model definitions
Database Agnostic	Easily switch between databases (e.g., SQLite, PostgreSQL)

How It Works in Django:

1. Define a Model:

```
class Student(models.Model):
    name = models.CharField(max_length=100)
    age = models.IntegerField()
```

This creates a SQL table:

```
CREATE TABLE student (
    id INTEGER PRIMARY KEY,
    name VARCHAR(100),
    age INTEGER
);
```

2. Create Data:

```
Student.objects.create(name="Ali", age=20)
```

3. Read Data:

```
Student.objects.all()
Student.objects.filter(age=20)
Student.objects.get(id=1)
```

4. Update Data:

```
student = Student.objects.get(id=1)
student.age = 21
student.save()
```

5. Delete Data:

```
student = Student.objects.get(id=1)
student.delete()
```

Comparison Table:

ORM (Python)	SQL

Django ORM - Notes

Student.objects.all()	SELECT * FROM student;
Student.objects.get(id=1)	SELECT * FROM student WHERE id=1;
student.save()	UPDATE student SET ...
student.delete()	DELETE FROM student WHERE ...

Summary:
Django ORM maps your Python classes (models) to database tables.
It helps you perform all database operations in Python code, making your app secure, scalable, and easier to maintain.