🧠 Django ORM Methods – Complete Notes

💡 What is ORM?

ORM (Object Relational Mapper) allows you to interact with your **database using Python code** instead of writing raw SQL queries.
It automatically converts Python objects ↔ Database tables.

Example:

```
student = StudentNew.objects.create(name="Hari", age=23, email="hari@gmail.com")
```

➡️ This internally runs:

```
INSERT INTO studentnew (name, age, email) VALUES ('Hari', 23, 'hari@gmail.com');
```

⚙️ Basic CRUD Operations Using ORM

Operation	Method	Purpose
Create	`.create()`, `.save()`	Add new record
Read	`.all()`, `.get()`, `.filter()`, `.values()`	Fetch data
Update	`.save()`, `.update()`	Modify data
Delete	`.delete()`	Remove data

🧩 1. CREATE Methods

☑️ `create()`

Creates and saves a new record in the database.

Syntax:

```
StudentNew.objects.create(name="Hari", age=23, email="hari@gmail.com")
```

Behind the scenes SQL:

```
INSERT INTO studentnew (name, age, email) VALUES ('Hari', 23, 'hari@gmail.com');
```

Return: Returns the created object.

☑ `save()`

Used to save an object — works for both **create** and **update**.

Example 1 (Create):

```
student = StudentNew(name="Ram", age=24, email="ram@gmail.com")
student.save()
```

Example 2 (Update):

```
student = StudentNew.objects.get(id=1)
student.email = "updated@gmail.com"
student.save()
```

---

2. READ Methods

☑ `all()`

Fetches all records from the table.

```
StudentNew.objects.all()
```

☑ `get()`

Fetches **exactly one record** that matches the condition.

```
StudentNew.objects.get(id=1)
```

Raises Error: if no record or multiple records found.

☑ `filter()`

Fetches all records that match a condition.

```
StudentNew.objects.filter(age__gte=20)
```

Chaining Example:

```
StudentNew.objects.filter(age__gte=20, name__icontains="ram")
```

☑ `exclude()`

Fetches all records **except** those that match a condition.

```
StudentNew.objects.exclude(age__lt=18)
```

☑ `values()`

Returns records as a list of **dictionaries** (useful for JSON).

```
StudentNew.objects.values()
```

☑ `values_list()`

Returns records as a list of **tuples**.

```
StudentNew.objects.values_list("name", "email")
```

☑ `first()` / `last()`

Returns the first or last record.

```
StudentNew.objects.first()
StudentNew.objects.last()
```

☑ `count()`

Counts matching records.

```
StudentNew.objects.filter(age__gte=20).count()
```

☑ `exists()`

Checks if any record exists.

```
StudentNew.objects.filter(email="hari@gmail.com").exists()
```

☑ `order_by()`

Sorts results ascending or descending.

```
StudentNew.objects.order_by('age')
StudentNew.objects.order_by('-age')
```

☑ `distinct()`

Removes duplicate records.

```
StudentNew.objects.values('age').distinct()
```

🧩 3. UPDATE Methods

☑ `save()`

Updates an existing record when called on an existing object.

```
student = StudentNew.objects.get(id=1)
student.age = 26
student.save()
```

☑ `update()`

Performs a **bulk update** directly in DB.

```
StudentNew.objects.filter(id=1).update(email="newmail@gmail.com")
```

☑ `update_or_create()`

If record exists, update; else create.

```
StudentNew.objects.update_or_create(
    email="hari@gmail.com",
    defaults={"name": "Hari Krishna", "age": 25}
)
```

---

4. DELETE Methods

☑ delete()

Deletes record from database.

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

☑ Bulk Delete

Deletes multiple records.

```
StudentNew.objects.filter(age__lt=18).delete()
```

---

5. COMBINED Methods

☑ get_or_create()

If record exists, fetch; else create.

```
StudentNew.objects.get_or_create(
    email="ram@gmail.com",
    defaults={"name": "Ram", "age": 24}
)
```

☑ aggregate()

For aggregation operations (Sum, Avg, Count, etc.)

```
from django.db.models import Sum, Avg, Count
StudentNew.objects.aggregate(Avg('age'), Count('id'))
```

☑ annotate()

Adds calculated fields to each record.

```
from django.db.models import F, Value
StudentNew.objects.annotate(next_age=F('age') + Value(1))
```

📞 Quick Reference Table

Purpose	Method	Returns	Notes
Create	`create()`	Object	Direct insert
Save	`save()`	None	Create or update
Read All	`all()`	QuerySet	All records
Filter	`filter()`	QuerySet	Multiple matches
Get One	`get()`	Object	Single record
Exclude	`exclude()`	QuerySet	Opposite of filter
Values	`values()`	List of dicts	JSON-friendly
Values List	`values_list()`	List of tuples	Lightweight
Count	`count()`	Integer	Record count
Exists	`exists()`	Boolean	True/False
Order	`order_by()`	QuerySet	Sorted
Distinct	`distinct()`	QuerySet	Unique values
First/Last	`first()` / `last()`	Object	Quick pick
Update	`update()`	Int	Bulk update
Delete	`delete()`	None	Delete records
Get or Create	`get_or_create()`	(Object, Bool)	Fetch or insert
Update or Create	`update_or_create()`	(Object, Bool)	Update or insert
Aggregate	`aggregate()`	Dict	Sum, Avg, etc.
Annotate	`annotate()`	QuerySet	Adds virtual field