**Django shell & CRUD operations**Django shell is tailor-made for Django project testing and debugging.

* Test Django models and relationships quickly
* Create, read, update, delete (CRUD) objects directly
* Debug complex queries or application logic
* Automate data patching and quick fixes

**CREATE — Add New Records**

**1. Using create()**

**->** from myapp.models import Post, User

-> user = User.objects.get(username="X")

-> Post.objects.create(title="New Post", content="Hello World", author=user)

**2. Using .save() manually**

**-**> post = Post(title="New Post", content="Hello World", author=user)

-> post.save()

**3. Bulk create ( bulk\_create)**

**-**> posts = [Post(title="Post 1", content="Bulk content 1", author=user),

Post(title="Post 2", content="Bulk content 2", author=user),]

-> Post.objects.bulk\_create(posts)

**4. get\_or\_create()**

-> user, created = User.objects.get\_or\_create(username="bob", defaults={"password": "default"})

**5. update\_or\_create()**

-> post, created = Post.obxjects.update\_or\_create( title="New or Existing",

defaults={"content": "Updated or created", "author": user})

**READ — Retrieve Data**

**1. All records**

-> Post.objects.all()

**2. One record (throws error if not found)**

-> Post.objects.get(id=1)

**3. Filtering**

-> Post.objects.filter(author=user)

-> Post.objects.filter(title\_\_icontains="Django")

**4. Exclude**

-> Post.objects.exclude(title\_\_icontains="Draft")

### **5. First/last record**

-> Post.objects.first()

-> Post.objects.last()

### **6. Values & values\_list**

-> Post.objects.values("title", "created\_at")

-> Post.objects.values\_list("title", flat=True)

### **7. Count & existence**

-> Post.objects.count()

-> Post.objects.filter(title="Hello").exists()

### **8. Aggregation** like count, sum, avg, min, max

-> from django.db.models import Count

-> Post.objects.aggregate(total=Count('id'))

## **CREATE — Add New Records**

### **1. Using create()**

from myapp.models import Post, User

user = User.objects.get(username="alice")

Post.objects.create(title="New Post", content="Hello World", author=user)

### **2. Using .save() manually**

tpost = Post(title="Manual Post", content="Saved later", author=user)

post.save()

### **3. Bulk create**

posts = [

Post(title="Post 1", content="Bulk content 1", author=user),

Post(title="Post 2", content="Bulk content 2", author=user),

]

Post.objects.bulk\_create(posts)

### **4. get\_or\_create()**

user, created = User.objects.get\_or\_create(username="bob", defaults={"password": "default"})

### **5. update\_or\_create()**

post, created = Post.objects.update\_or\_create(

title="New or Existing",

defaults={"content": "Updated or created", "author": user}

)

## **🟢 READ — Retrieve Data**

### **1. All records**

Post.objects.all()

### **2. One record (throws error if not found)**

Post.objects.get(id=1)

### **3. Filtering**

Post.objects.filter(author=user)

Post.objects.filter(title\_\_icontains="Django")

### **4. Exclude**

Post.objects.exclude(title\_\_icontains="Draft")

### **5. First/last record**

Post.objects.first()

Post.objects.last()

### **6. Values & values\_list**

Post.objects.values("title", "created\_at")

Post.objects.values\_list("title", flat=True)

### **7. Count & existence**

Post.objects.count()

Post.objects.filter(title="Hello").exists()

### **8. Aggregation**

like count, sum, avg, min, max

from django.db.models import Count

Post.objects.aggregate(total=Count('id'))

## **UPDATE — Modify Existing Data**

### **1. Fetch + change + save**

post = Post.objects.get(id=1)

post.title = "Updated Title"post.save()

### **2. Bulk update**

Post.objects.filter(author=user).update(content="Same content for all")

## **DELETE — Remove Data**

### **1. Delete one object**

post = Post.objects.get(id=1)

post.delete()

### **2. Bulk delete**

Post.objects.filter(author=user).delete()

### **3. Delete all**

Post.objects.all().delete()

## **— Q Object** Q allows bitwise operations (&, |, ~) for combining query conditions. Because Django’s filter() method uses AND logic by default and does not support OR directly.

## -> from django.db.models import Q -> Model.objects.filter(Q(condition1) | Q(condition2))

## **Mixing Q and regular filters** **->** post.objects.filter(Q(title\_\_icontains="django") | Q(content\_\_icontains="rest"), author=user) -> Post.objects.filter(Q(title\_\_icontains="hello") | Q(content\_\_icontains="world"))

## **— F Expression – Field-to-Field Comparison**

## Allows you to reference model fields directly in queries.

## Used when you want to compare a field to another field, not a value.

## Also used to perform database-level arithmetic. **->** from django.db.models import F -> Post.objects.filter(id\_\_gt=F('author\_id'))