Django Relationships & Querying Related Objects

1. Overview of Django Relationships

Django supports several types of relationships between models:

- ForeignKey (One-to-Many)
- OneToOneField (One-to-One)
- ManyToManyField (Many-to-Many)

You can access related objects from either side using dot notation, reverse relationships, or filter queries.

2. ForeignKey Example: Photo and Like

```
class Photo(models.Model):
    title = models.CharField(max_length=100)

class Like(models.Model):
    to_photo = models.ForeignKey(Photo, on_delete=models.CASCADE)
```

- One Photo can have many Likes.
- One Like is linked to one Photo.

3. Querying Related Objects (ForeignKey)

From Photo to Likes:

```
photo = Photo.objects.get(id=1)
likes = photo.like_set.all()
```

From Like to Photo:

```
like = Like.objects.get(id=10)
photo = like.to_photo
```

Filtering:

```
all_likes = Like.objects.filter(to_photo_id=1)
photos_with_likes = Photo.objects.filter(like__isnull=False).distinct()
```

4. ManyToMany Example: Pet and Photo

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

class Photo(models.Model):
    image = models.ImageField()
    tagged_pets = models.ManyToManyField(Pet)
```

- A Photo can tag many Pets.
- A Pet can be tagged in many Photos.

5. Querying Related Objects (ManyToMany)

From Photo to Pets:

```
photo = Photo.objects.get(id=1)
pets = photo.tagged_pets.all()
```

From Pet to Photos:

```
pet = Pet.objects.get(id=2)
photos = pet.photo_set.all()
```

Filtering:

```
Photo.objects.filter(tagged_pets__name='Max')
Pet.objects.filter(photo__id=1)
```

6. Like Toggle Example

```
def like(request, photo_id):
```

Django Relationships & Querying Related Objects

```
like_object = Like.objects.filter(to_photo_id=photo_id).first()
if like_object:
    like_object.delete()
else:
    Like.objects.create(to_photo_id=photo_id)
```

- This checks if a Like exists for a photo.
- If it exists, it removes it (unlike).
- If it doesn't, it adds a Like.