Django ORM Queries book by Naveen

models.py

```
class Person(models.Model):
    name = models.CharField(max_length=50, blank=True)
    age = models.IntegerField()
    gender = models.CharField(max_length=10, blank=True)
```

The most used data types are:

SQL	Django
INT	IntegerField()
VARCHAR(n)	CharField(max_length=n)
TEXT	TextField()
FLOAT(n)	FloatField()
DATE	DateField()
TIME	TimeField()
DATETIME	DateTimeField()

SELECT Statement

Fetch all rows

IN SQL : SELECT * FROM Person;

IN DJANGO : persons = Person.objects.all()

To read and display the all records

```
Person.objects.all()

for person in persons:

    print(person.name)

    print(person.gender)

    print(person.age)
```

Fetch specific columns

SQL: SELECT name, age FROM Person;

Django: Person.objects.only('name', 'age')

Fetch distinct rows

SQL: SELECT DISTINCT name, age FROM Person;

Django: Person.objects.values('name', 'age').distinct()

Fetch specific number of rows

SQL: SELECT * FROM Person LIMIT 10;

Django: Person.objects.all()[:10]

LIMIT AND OFFSET keywords

SQL: SELECT * FROM Person OFFSET 5 LIMIT 5;

Django: Person.objects.all()[5:10]

WHERE Clause

Filter by single column

SQL: SELECT * FROM Person WHERE id = 1;

Django: Person.objects.filter(id=1)

Filter by comparison operators

SQL: WHERE age > 18;

WHERE age >= 18;

WHERE age < 18;

WHERE age <= 18;

WHERE age != 18;

Django: Person.objects.filter(age__gt=18)

Person.objects.filter(age__gte=18)

Person.objects.filter(age__lt=18)

Person.objects.filter(age__lte=18)

Person.objects.exclude(age=18)

BETWEEN Clause

SQL: SELECT * FROM Person WHERE age BETWEEN 10 AND 20;

Django: Person.objects.filter(age__range=(10, 20))

LIKE operator

SQL:

WHERE name like '%A%';

WHERE name like binary '%A%';

WHERE name like 'A%';

WHERE name like binary 'A%';

WHERE name like '%A';

WHERE name like binary '%A';

Django:

Person.objects.filter(name__icontains='A')

Person.objects.filter(name__contains='A')

Person.objects.filter(name__istartswith='A')

Person.objects.filter(name__startswith='A')

Person.objects.filter(name__iendswith='A')

Person.objects.filter(name__endswith='A')

IN operator

SQL: WHERE id in (1, 2);

Django: Person.objects.filter(id__in=[1, 2])

AND, OR and NOT Operators

SQL: WHERE gender='male' AND age > 25;

Django: Person.objects.filter(gender='male', age__gt=25)

SQL: WHERE gender='male' OR age > 25;

Django: from django.db.models import QPerson.objects.

WHERE NOT gender='male' OR age > 25;

SQL : WHERE gender='male' OR age > 25;

Django: Person.objects.exclude(gender='male')

filter(Q(gender='male') | Q(age__gt=25))

SQL: WHERE NOT gender='male';

Django: Person.objects.exclude(gender='male')

NULL Values

SQL: WHERE age is NULL;

WHERE age is NOT NULL;

Django:

Person.objects.filter(age__isnull=True)

Person.objects.filter(age__isnull=False)

Alternate approach

Person.objects.filter(age=None)

Person.objects.exclude(age=None)

ORDER BY Keyword

Ascending Order

SQL: SELECT * FROM Person order by age;

Django: Person.objects.order_by('age')

Descending Order

SQL: SELECT * FROM Person ORDER BY age DESC;

Django: Person.objects.order_by('-age')

INSERT INTO Statement

SQL: INSERT INTO Person VALUES ('Jack', '23', 'male');

Django: Person.objects.create(name='jack', age=23, gender='male)

UPDATE Statement

Update single row

SQL: UPDATE Person SET age = 20 WHERE id = 1;

Django: person = Person.objects.get(id=1)

person.age = 20

person.save()

Update multiple rows

SQL: UPDATE Person SET age = age * 1.5;

Django: from django.db.models import F

Person.objects.update(age=F('age')*1.5)

DELETE Statement

Delete all rows

SQL: DELETE FROM Person;

Django: Person.objects.all().delete()

Delete specific rows

SQL: DELETE FROM Person WHERE age < 10;

Django: Person.objects.filter(age__lt=10).delete()

Aggregation

MIN Function

SQL: SELECT MIN(age) FROM Person;

Django: from django.db.models import Min

Person.objects.all().aggregate(Min('age'))

MAX Function

SQL: SELECT MAX(age) FROM Person;

Django: from django.db.models import Max

Person.objects.all().aggregate(Max('age'))

AVG Function

SQL: SELECT AVG(age) FROM Person;

Django: from django.db.models import Avg

Person.objects.all().aggregate(Avg('age'))

SUM Function

SQL: SELECT SUM(age) FROM Person;

Django: from django.db.models import Sum

Person.objects.all().aggregate(Sum('age'))

COUNT Function

SQL: SELECT COUNT(*) FROM Person;

Django: Person.objects.count()

GROUP BY Statement

Count of Person by gender

SQL: SELECT gender, COUNT(*) as count FROM Person GROUP BY gender;

Django: Person.objects.values('gender').annotate(count=Count('gender'))

HAVING Clause

Count of Person by gender if number of person is greater than 1

SQL: SELECT gender, COUNT('gender') as count FROM Person

GROUP BY gender HAVING count > 1;

Django: Person.objects.annotate(count=Count('gender'))

.values('gender', 'count') .filter(count__gt=1)

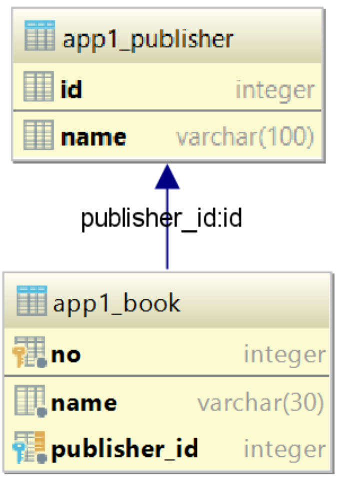
JOINS

Consider a foreign key relationship between books and publisher.

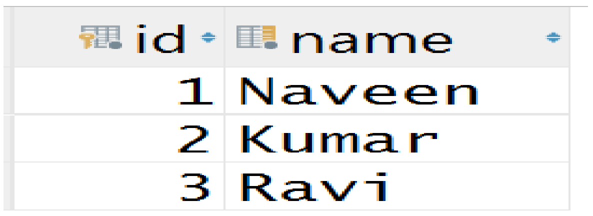
1) Models.py

```
class Publisher(models.Model):
    name = models.CharField(max_length=100,unique=True)
class Book(models.Model):
    no = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=30)
    publisher = models.ForeignKey(Publisher,
on_delete=models.CASCADE)
```

Tables in visualisation

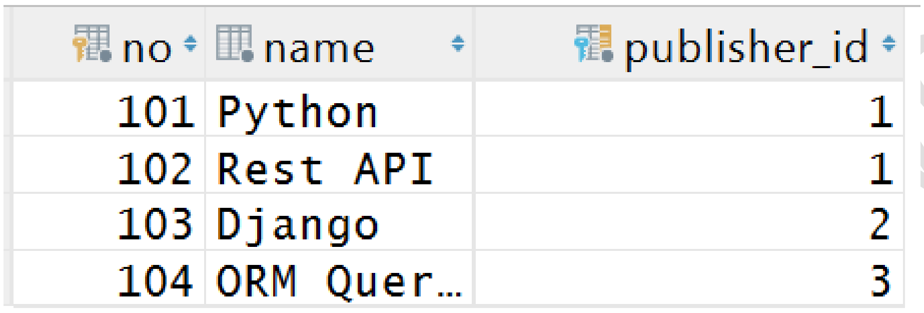


Publisher Data



id	name
1	Naveen
2	Kumar
3	Ravi

Book Data



no	name	publisher_id
101	Python	1
102	Rest API	1
103	Django	2
104	ORM Quer…	3

2) views.py

```
from django.shortcuts import render
from app1.models import Book

res = Book.objects.select_related('publisher').filter
(publisher_id=1)
for x in res:
    print(x.publisher.name)
```

3) Output

```
Naveen

Naveen
```