Django ORM

ITcoder

SELECT

- полный вывод
- вывод определенного столбца

SQL: **SELECT name, age FROM Person;**
Django ORM: **Person.objects.only('name','age')**

distinct

SQL: **SELECT DISTINCT name, age FROM Person;**
Django ORM: **Person.objects.values('name', 'age').distinct()**

offset / limit (pagination)

SQL: **SELECT * FROM Person OFFSET 5 LIMIT 5;**
Django ORM: **Person.objects.all()[5:10]**

WHERE

- Выходные данные**,** соответствующие условиям **filter/get**
- Вывод с помощью оператора

SQL	Django ORM
WHERE age > 18	Person.objects.filter(age__gt=18)
WHERE age >= 18	Person.objects.filter(age__gte=18)
WHERE age < 18	Person.objects.filter(age__lt=18)
WHERE age <= 18	Person.objects.filter(age__lte=18)
WHERE age != 18	Person.objects.exclude(age=18)

Between

SQL: **SELECT * FROM Person WHERE age BETWEEN 10 AND 20;**
Django ORM: **Person.objects.filter(age__range=(10, 20))**

ITcoder

Like

SQL	Django ORM
WHERE name like '%A%'	Person.objects.filter(name__icontains='A')
WHERE name like binary '%A%'	Person.objects.filter(name__contains='A')
WHERE name like 'A%'	Person.objects.filter(name__istartswith='A')
WHERE name like binary 'A%'	Person.objects.filter(name__startswith='A')
WHERE name like '%A'	Person.objects.filter(name__iendswith='A')
WHERE name like binary '%A'	Person.objects.filter(name__endswith='A')

IN

SQL:	WHERE id in (1, 2);
Django ORM:	Person.objects.filter(id__in=[1, 2])

AND ,OR , NOT

SQL	Django ORM
WHERE gender='male' AND age > 25;	Person.objects.filter(gender='male', age__gt=25)

WHERE gender='male' OR age > 25;	from django.db.models import Q
	
	Person.objects.filter(Q(gender='male') | Q(age__gt=25))

WHERE NOT gender='male';	Person.objects.exclude(gender='male')

NULL

WHERE age is NULL;
WHERE age is NOT NULL;
Person.objects.filter(age__isnull=True)
Person.objects.filter(age__isnull=False)

Person.objects.filter(age=None)
Person.objects.exclude(age=None)

ITcoder

ORDER BY

По возрастанию **(ASC)** По убыванию **(DESC)**

SQL	Django ORM
SELECT * FROM Person order by age;	Person.objects.order_by('age')
SELECT * FROM Person ORDER BY age DESC;	Person.objects.order_by('-age')

FIRST/LAST

вывести первое и последнее значение

SQL	Django ORM
SELECT FIRST(age) FROM Person;	Person.objects.order_by('age').first()
SELECT LAST(age) FROM Person;	Person.objects.order_by('age').last()

INSERT INTO(CREATE)

UPDATE

редактировать один столбец

UPDATE Person SET age = 20 WHERE id = 1;
person = Person.objects.get(id=1)
person.age = 20
person.save()

Person.objects.filter(id=1).update(age=20)

Редактировать несколько столбцов

UPDATE Person SET age = age * 1.5;
from django.db.models import F

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

ITcoder

DELETE

удалить все столбцы
Удалить конкретный столбец

DELETE FROM Person WHERE age < 10;
Person.objects.filter(age__lt=10).delete()

AGGREGATION

MIN, MAX, AVG, SUM, COUNT

SQL	Django ORM

SELECT MIN(age) FROM Person	from django.db.models import Min, Max, Avg, Sum
	Person.objects.all().aggregate(Min('age'))
	{'age__min': 0}

SELECT MAX(age) FROM Person	Person.objects.all().aggregate(Max('age'))
	{'age__max': 100}

SELECT AVG(age) FROM Person	Person.objects.all().aggregate(Avg('age'))
	{'age__avg': 50}

SELECT SUM(age) FROM Person	Person.objects.all().aggregate(Sum('age'))
	{'age__sum': 5050}

SELECT COUNT(*) FROM Person	Person.objects.count()

Данные **Min, Max, Average, Sum** могут быть агрегированы
При использовании **count()** подсчитывается общее количество данных в таблице**.**

GROUP BY(Annotate)

COUNT

SELECT gender, COUNT(*) as count FROM Person GROUP BY gender;
Person.objects.values('gender').annotate(count=Count('gender'))
Подсчитайте количество людей по полу

HAVING (COUNT IF)

SQL:

```
SELECT gender, COUNT('gender') as count FROM Person GROUP BY gender HAVING count > 1;
```

DJango ORM:

```
Person.objects.annotate(count=Count('gender')).values('gender', 'count').filter(count__gt=1)
```