


Agenda :

- Inheritance
- IDS
- Custom Queries
- ORM Query
- Raw Queries.

Inheritance

 created_at
updated_at
created_by
updated_by

```
class AuditData(models.Model)
```

```
    updated_at = models.DateTimeField...
```

```
    created_at
```

```
    updated_by = models...
```

```
class Meta:
```

```
    abstract = True
```

class Products (Audi+ data):

'''

class chapter(models.model)

name = _____

class Book(chapter)

Author = _____

behind the scenes There would be a
1:1 field.

IDS:

Every model automatically gets
a PK called id.

class Category(models.model):

c_name = models.CharField(Primary

Key = True)

Class enrollment (models. model):

{ Student = _____
 { Course = _____

Class meta:

unique_together = ("student",
 "course")

Table without PK

class meta:

managed = False

Queries using ORM

Products.objects.all()

Products.objects.get(id=1)

Products.objects.filter(price__gt=500)

Products.objects.filter(price__gte=500)

Products.objects.exclude(is_available=False)

.filter(name__icontains="shirt")

.order_by(-price)

Select * from _____

..... where id = 1

where price > 500

where price >= 500

where is_available != True

where name ILIKE '%shirt%'

order by price desc

Products.objects.aggregate(Avg('Price'))

objects.count()

objects.values('id', 'Price')

Select Avg(Price) _____

Select count(*) from _____

Select id, Price _____

limit

Products.objects.all()[: n]

↑
offset

Q Object:

from django.db.models import Q

Products.objects.filter(Q('Price' >= 500) | Q('is_available' = false))

If we write

Products.objects.filter(Price >= 500),

filter(, is available = false)

↓
Select * from Products where Price >= 500

AND is available = false

$Q(\text{Price_gt} = 500) \& Q(\text{is_over} = \text{false})$

Same as

$\text{Products.objects.filter}(\text{Price_gt}=500),$
 $\text{filter}(\text{is_over}=\text{false})$

USE Case	filter	Q
AND	✓	optional
OR	x	✓
!	x	✓ ($\neg Q(\dots)$)
Complex queries	x	✓

$\text{Products.objects.filter}(Q(\dots))$
 \uparrow
 $\text{aggregate}(\text{Avg}(\dots))$

Raw Query:

```
Products.objects.raw("select * from  
Products where Price > %.5",  
[500])
```

Print Query:

```
qs = Products.objects.all()
```

```
print (qs.query)
```

Tip: pip install django-debug-toolbar

doubt :

from django.db.models.functions
import Range, ...