----------- AND (&) OR(|) OPERATÖRLERİ ------------

----------- Q NESNESİ İLE GELİŞMİŞ FİLTRELEME ------------

Daha önce öğrendiğimiz gibi Django ORM sorguları AND ile birleşiyordu. OR kullanabilmek için Q Nesnesinden yararlanırız.

# WHERE question LIKE 'Who%'

Q(question\_\_startswith='Who')

# WHERE question LIKE 'Who%' OR question LIKE 'What%'

Q(question\_\_startswith='Who') | Q(question\_\_startswith='What')

# SELECT \* from polls WHERE question LIKE 'Who%' AND (pub\_date = '2005-05-02' OR pub\_date = '2005-05-06')

Poll.objects.get(  
 Q(question\_\_startswith='Who')**,** Q(pub\_date=date(**2005, 5, 2**)) | Q(pub\_date=date(**2005, 5, 6**))  
)

# Q tilde (yaklaşık) işareti ile olumsuz yapılır

# WHERE question NOT LIKE 'Who%'

~Q(question\_\_startswith='Who')

----------- F NESNESİ ------------

F() bir model alanının değerini almamızı sağlar.

Product tablomuzda sipariş adedi ürün adedinden çok olan kayıtları çekelim. Burada F nesnesi ile Product tablosunda num\_stocks alanına direkt ulaşım sağladık.

Product.objects.filter(num\_orders\_\_gt=F('num\_stocks))

İşlemler Python tarafında değil veritabanı tarafında gerçekleşir. Bizim bir Book nesnemiz olsun ve view\_count alanımız olsun. Alan her görüntülendiğinde bir artsın. Sayfa görüntülendiğinde view\_count 1 artıralım. Burada veritabanından book nesnesini belleğe aldık ve Python operatörlerini kullanarak bir artırdık ve veritabanına tekrar kaydettik.

book = Book.objects.get(id=**1**)

book.view\_count += 1

book.save()

Bu örneğimizde ise **book.view\_count = F('view\_count') + 1** işlemi görüntü olarak Python işlemi gibi görünse de aslında veritabanında gerçekleşmektedir. Django F() gördüğünde Python operatörlerin geçersiz kılar. Python hiçbir zaman view\_count alanı hakkında bilgi sahibi olmaz. Bu durumda yeni veriye ulaşabilmek için veritabanından gelen veri yenilenmelidir.

from django.db.models import F  
  
book = Book.objects.get(id=1)  
book.view\_count = F('view\_count') + **1** # veritabanı işlemibook.save()

book.refresh\_from\_db() # view\_count yeni değerine artık ulaşabiliriz

view\_count artırma işlemini update() metodu ile de yapabilirdik.

book = Book.objects.get(id=1)  
book.update(view\_count=F('view\_count') + **1**)

F nesneleri model kaydedildikten sonra yeni bir kayıtta tekrar uygulanır. Bunu önlemek için kayıttan sonra refresh\_from\_db kullanmak gerekir.

book = Book.objects.get(id=1)  
book.view\_count = F('view\_count') + **1** # veritabanı işlemibook.save()

# burada refresh\_from\_db yapılırsa aşağıdaki kod sadece ismi günceller

book.name = 'Yeni kitap adı'  
book.save() # view\_count burda tekrar artar

F nesneleri annotate ile birlikte kullanılabilir. Her ürün için gerekli ilave stok miktarını hesaplayalım.

products = Product.objects.annotate(needed\_stock=F('num\_orders') – F('view\_stocks'))

F nesnesi ile boş kayıtları sorgu setinin en başına veya sonuna sıralayabiliriz.

Product.objects.order\_by(F('name').desc(nulls\_last=True)