----------- HESAPLAMA İŞLEMLERİ – AGGREGATION --------------

Aggregation, sorgularda toplam, ortalama, en çok, en az gibi hesaplama işlemleri yapmamızı sağlar.

aggregate() tablo için genel hesaplamalar, annotate() ise satır bazında hesaplamalar yapar. annotate() genelde ilişkili tablo hesaplamalarında kullanılır.

Örnekler üzerinden devam edelim.

from django.db.models import Avg, Count, Max, Min, Sum

# adet - 73

Product.objects.all().count()

############## AGGREGATE ##############

# ortalama – price\_\_avg : 23.34  
Product.objects.aggregate(Avg('price'))

# en çok – price\_\_max : 50

Product.objects.aggregate(Max('price'))

# en az – price\_\_min : 3

Product.objects.aggregate(Min('price'))

# toplam – price\_\_sum : 4

Product.objects.aggregate(Sum('price'))

# ortalama - istediğimiz isme atabiliriz – average\_price : 23.34

Product.objects.aggregate(average\_price=Avg('price'))

# birden fazla aggregate çalıştırabiliriz.

# {'price\_\_avg': 34.35, 'price\_\_max': Decimal('81.20'), 'price\_\_min': Decimal('12.99')}

Product.objects.aggregate(Avg('price')**,** Max('price')**,** Min('price'))

############## ANNOTATE ##############

# kullanıcılara ait ürün sayıları

users = Users.objects.annotate(Count('products'))

users[0].products\_\_count

# belirlediğimiz isme atayabiliriz.

users = Users.objects.annotate(num\_products=Count('products'))

users[0].num\_products

Birden fazla annotate birleştirilmesi hatalara sebep olur. Çoğu hesaplama için bunun çözümü yoktur fakat count için distinct ile çözülür.

# yanlış değerler döndürür.

users = Users.objects.annotate(Count('products'), Count('books'))

# doğru değerler döndürür.

users = Users.objects.annotate(Count('products' **,** distinct=True), Count('books' **,** distinct=True))

İlişkili işlemlerde annonate kullanılabilir. Kategorideki max ve min ürün fiyatlarını alalım. Birden fazla ilişki için de kullanılabilir. Çoka – çok ilişkiler için reverse relationship içinde kullanılabilir.

from django.db.models import Max**,** Min  
Category.objects.annotate(min\_price=Min('product\_\_price')**,** max\_price=Max('product\_\_price'))

# products\_\_authors\_\_age

# reverse relationship

Product.objects.annotate(Count('categories'))

filter() kullanımı.

Annotate filtrelenebilir. Toplam ürünü birden fazla olan kategoriler

Category.objects.annotate(num\_products=Count(products)).filter(num\_products\_\_gt=**1**)

Eğer birden fazla annotate varsa

Author.objects.annotate(num\_books=Count('book')**,** highly\_rated\_books= Count('book'**,** filter=Q(book\_\_rating\_\_gte=**7**)))

Annotate() ve filter() birlikte kullanımında sıralamaları önem arz etmektedir.

# Bu sorguda filtrenin annotate üzerinde bir etkisi yoktur. Her yayıncı için kitap sayılarını sayar.

a**,** b = Publisher.objects.annotate(num\_books=Count('book'**,** distinct=True)).filter(book\_\_rating\_\_gt=**3.0**)

# Bu sorgu her yayıncı için kitap ratingi 3 geçen kayıtları sayar.

a**,** b = Publisher.objects.filter(book\_\_rating\_\_gt=**3.0**).annotate(num\_books=Count('book'))

order\_by() kullanımı.

annotateler sıralanabilir

User.objects.annotate(num\_products=Count('products')).order\_by('num\_products')

values() kullanımı.

Annonate ile kullanıcılara ait ortalama kitap fiyatlarını seçelim. Bu sorgu her kullanıcı için ortalama kitap fiyatı hesaplayacaktır.

User.objects.annotate(average\_price=Avg('product\_\_price'))

Bu sorguyu first\_name alanına göre grupladığımızda durum biraz değişir. Aynı ismi kullanan birden fazla satır o varsa o satırların kayıtları birleştirilir.

User.objects.values('first\_name').annotate(average\_price=Avg('product\_price'))

values() metodu annonate() metodunundan sonra kullanılduğında ise annonate den gelen veriler üzerinden gelen alanlar alınır. Bu örneğimizde her bir kullanıcı için ayrı ayrı ürün fiyatı hesaplaması yapılır.

User.objects.annotate(average\_price=Avg('product\_price')).values('first\_name', 'average\_price')

ordering kullanımı :

User modelini name alanına göre sıraladığımızı düşünelim.

class Meta:  
 ordering = ["last\_name"]

Kullanıcı tablosunda kaçtane first\_name olduğunu saydıralım. Bu bize tam doğru sonucu vermez. Çünkü sorgu setimiz benzersiz (first\_name ve last\_name) göre gruplanır.

User.objects.values('first\_name').annotate(Count('id'))

İstediğimiz değeri alabilmek için verileri tekrardan first\_name göre sıralamamız gereklidir.

User.objects.values('first\_name').annotate(Count('id')).order\_by()

Aggregate ve filter() kullanımı

from django.db.models import Count  
Client.objects.aggregate(  
 regular=Count('pk'**,** filter=Q(account\_type=Client.REGULAR))**,** gold=Count('pk'**,** filter=Q(account\_type=Client.GOLD))**,** platinum=Count('pk'**,** filter=Q(account\_type=Client.PLATINUM))**,**)

**SELECT** **count**('id') FILTER (**WHERE** account\_type=1) **as** regular,

**count**('id') FILTER (**WHERE** account\_type=2) **as** gold,

**count**('id') FILTER (**WHERE** account\_type=3) **as** platinum

**FROM** clients;