Aggregate()

with aggregate functions you condense all data of a model to a single value.

from django.db import models

class Author(models.Model):

name = models.CharField(max\_length=100)

age = models.IntegerField()

class Publisher(models.Model):

name = models.CharField(max\_length=300)

class Book(models.Model):

name = models.CharField(max\_length=300)

pages = models.IntegerField()

price = models.DecimalField(max\_digits=10, decimal\_places=2)

rating = models.FloatField()

authors = models.ManyToManyField(Author)

publisher = models.ForeignKey(Publisher, on\_delete=models.CASCADE)

pubdate = models.DateField()

class Store(models.Model):

name = models.CharField(max\_length=300)

books = models.ManyToManyField(Book)

Cheat sheet

In a hurry? Here’s how to do common aggregate queries, assuming the models above:

# Total number of books.  
>>> Book.objects.count()

2452

# Total number of books with publisher=BaloneyPress

>>> Book.objects.filter(publisher\_\_name='BaloneyPress').count()

73

# Average price across all books.

>>> **from** **django.db.models** **import** Avg

>>> Book.objects.all().aggregate(Avg('price'))  
>>> # Book.objects.aggregate(Avg('price')) # or by removing .all()

{'price\_\_avg': 34.35}

# Max price across all books.

>>> **from** **django.db.models** **import** Max

>>> Book.objects.all().aggregate(Max('price'))

{'price\_\_max': Decimal('81.20')}

# Difference between the highest priced book and the average price of all books.

>>> **from** **django.db.models** **import** FloatField

>>> Book.objects.aggregate(

... price\_diff=Max('price', output\_field=FloatField()) - Avg('price'))

{'price\_diff': 46.85}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Example:

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

>>> from country.models import Country, City

>>> num\_countries = Country.objects.aggregate(Count('name'))

>>> num\_countries

{'name\_\_count': 2}

>>> min\_population = City.objects.all().aggregate(Min('population'))

>>> min\_population

{'population\_\_min': 8485}  
>>> max\_population = City.objects.all().aggregate(Max('population'))

>>> max\_population

{'population\_\_max': 10000000}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Example: Print total numbers of cities of each Country: assuming we have two countries each Canada and Afghanistan in database.

>>> res = Country.objects.filter(name='Afghanistan').aggregate(Count('city\_\_country'))

>>> res

{'city\_\_country\_\_count': 2}

>>> res = Country.objects.filter(name='Canada').aggregate(Count('city\_\_country'))

>>> res

{'city\_\_country\_\_count': 3}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Annotation

<https://www.youtube.com/watch?v=KbwmdKl-QbI>

Example: find which city of countries among list of countries has biggest population .

>>> res = Country.objects.annotate(Min('city\_\_population'))

>>> res

<QuerySet [Canada - Country Object, Afghanistan - Country Object]>

>>> res[0]

Canada - Country Object

>>> vars(res[0])

{'\_state': <django.db.models.base.ModelState object at 0x000001A54528DAC8>, 'id': 1, 'name': 'Canada', 'city\_\_population\_\_min': 8485}

>>> res = Country.objects.annotate(min\_city\_pop = Min('city\_\_population'))

>>> vars(res[0])

{'\_state': <django.db.models.base.ModelState object at 0x000001A54528DB70>, 'id': 1, 'name': 'Canada', 'min\_city\_pop': 8485}

>>> res[0].min\_city\_pop  
8485

Example: find the minimum population of city of Canada

>>> res = Country.objects.filter(name='Canada').annotate(min\_pop = Min('city\_\_population'))

>>> res[0].min\_pop

8485

Example: find which city of countries among list of countries has less population .

>>> res = Country.objects.annotate(max\_pop=Max('city\_\_population'))

>>> vars(res[0])

{'\_state': <django.db.models.base.ModelState object at 0x000001A54528DF28>, 'id': 1, 'name': 'Canada', 'max\_pop': 2930000}

>>> res[0].max\_pop

2930000