Contents

[CREATE, READ, UPDATE, DELETE 2](#_Toc28347021)

[CREATE 3](#_Toc28347022)

[READ 4](#_Toc28347023)

[UPDATE 5](#_Toc28347024)

[DELETE 6](#_Toc28347025)

[FILTER 7](#_Toc28347026)

[EXCLUDE 8](#_Toc28347027)

[LIMIT & OFFSET 9](#_Toc28347028)

[order\_by() 10](#_Toc28347029)

[count() 11](#_Toc28347030)

# CREATE, READ, UPDATE, DELETE

**models.py**

from django.db import models

class Language(models.Model):

    lang\_name = models.CharField(max\_length=128)

    def \_\_str\_\_(self):

        return self.lang\_name

class Framework(models.Model):

    fw\_name = models.CharField(max\_length=128)

    fw\_lang = models.ForeignKey(Language, on\_delete=models.CASCADE)

    def \_\_str\_\_(self):

        return self.fw\_name

class Student(models.Model):

    class YearInSchool(models.TextChoices):

        FRESHMAN = 'FR', 'Freshmman'

        SOPHOMORE = 'SO', 'Sophomore'

        JUNIOR = 'JR', 'Junior'

        SENIOR = 'SR', 'Senior'

        GRADUATE = 'GR', 'Graduate'

    full\_name = models.CharField(max\_length=70)

    email = models.CharField('User\'s Email', max\_length=180, blank=True)

    year\_in\_school = models.CharField(

        max\_length=2,

        choices=YearInSchool.choices,

        default=YearInSchool.JUNIOR,

    )

    university\_name = models.ForeignKey(University, on\_delete=models.CASCADE)

    teacher = models.ManyToManyField(

        Teacher,

        through='StudentTeacher'

    )

    class Meta:

        verbose\_name = 'Student'

        verbose\_name\_plural = "Students"

# CREATE

>>> from model2.models import Language, Framework

>>> py = Language.objects.get(lang\_name='Python')

>>> matplotlib = Framework(fw\_name="Matplotlib", fw\_lang= py) **# only creates**

>>> matplotlib.save() **# only save**

>>> Framework.objects.all()

<QuerySet [<Framework: Django>, <Framework: Flask>, <Framework: jQuery>, <Framework: Matplotlib>]>

**# create() - creates and saves an object**

>>> electrons = Framework.objects.create(fw\_name='Electrons', fw\_lang=py)

>>> Framework.objects.all()

<QuerySet [<Framework: Django>, <Framework: Flask>, <Framework: jQuery>, <Framework: Electrons>]>

# READ

>>> py = Language.objects.get(lang\_name='Python')

NOTE: get\_object\_or\_404

# [UPDATE](https://stackoverflow.com/questions/2712682/select-and-update-database-record-with-a-single-queryset)

>>> java = Language.objects.get(lang\_name='Java')

>>> matplotlib.fw\_lang= java **# only creates**

>>> matplotlib.save() **# only saves**

>>> Framework.objects.all()

<QuerySet [<Framework: Django>, <Framework: Flask>, <Framework: jQuery>, <Framework: Matplotlib>]>

**# both update and save()**

>>> Framework.objects.filter(fw\_name='Matplotlib').**update**(fw\_name='Matl')

1

>>> Framework.objects.all()

<QuerySet [<Framework: Django>, <Framework: Flask>, <Framework: jQuery>, <Framework: Matl>]>

# DELETE

>>> dlt\_flask = Framework.objects.get(fw\_name='Flask').delete()

>>> Framework.objects.all()

**# flask is now deleted**

<QuerySet [<Framework: Django>, <Framework: jQuery>, <Framework: Electrons>]>

>>> matplotlib.delete()

(1, {'model2.Framework': 1})

>>> Framework.objects.all()

**# Matl doesn’t exist anymore**

<QuerySet [<Framework: Django>, <Framework: Flask>, <Framework: jQuery>]

# FILTER

>>> py = Framework.objects.filter(fw\_lang = py)

>>> py

<QuerySet [<Framework: Django>, <Framework: Electrons>]>

**# Chaining filters**

>>> res = Student.objects.filter(year\_in\_school="JR")

>>> res

<QuerySet [<Student: Basir Payenda>, <Student: john doe>]>

>>> res = Student.objects.filter(year\_in\_school="JR").filter(full\_name='Basir Payenda')

>>> res

<QuerySet [<Student: Basir Payenda>]>

# EXCLUDE

>>> Framework.objects.all()

<QuerySet [<Framework: Django>, <Framework: jQuery>, <Framework: Electrons>]>

>>> fw\_except\_dj = Framework.objects.exclude(fw\_name='Django')

>>> fw\_except\_dj

<QuerySet [<Framework: jQuery>, <Framework: Electrons>]>

# LIMIT & OFFSET

this returns the first 5 objects (**LIMIT 5**):

>>> Entry.objects.all()[:5]

this escapes over the first 3 objects (**OFFSET 3**):

>>> Entry.objects.all()[3:]

This returns the sixth through tenth objects (OFFSET 5 LIMIT 5):

>>> Entry.objects.all()[5:10]

# order\_by()

**order\_by(\*fields)**

By default, results returned by a **QuerySet** are ordered by the ordering tuple given by the **ordering** option in the model’s **Meta**. You can override this on a per-**QuerySet** basis by using the **order\_by** method.

Example:

Entry.objects.filter(pub\_date\_\_year=2005).order\_by('-pub\_date', 'headline')

The result above will be ordered by **pub\_date** descending, then by **headline** ascending. The negative sign in front of **"-pub\_date"** indicates descending order. Ascending order is implied. To order randomly, use **"?"**, like so:

Entry.objects.order\_by('?')

Note: **order\_by('?')** queries may be expensive and slow, depending on the database backend you’re using.

To order by a field in a different model, use the same syntax as when you are querying across model relations. That is, the name of the field, followed by a double underscore (**\_\_**), followed by the name of the field in the new model, and so on for as many models as you want to join. For example:

Entry.objects.order\_by('blog\_\_name', 'headline')

# count()

# Total number of books.

>>> Book.objects.count()

2452

# Total number of books with publisher=BaloneyPress

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

73