* Question.objects.filter(question\_text\_\_startswith='what')
* Question.objects.filter(id=1)
* Question.objects.all()
* q.question\_text
* q.question\_text = "Welcome"
* q.save()
* q.pub\_date

from django.utils import timezone

* q=Question(question\_text="what's new?", pub\_date=timezone.now())
* q.save()
* q.id
* Question.objects.get(pk=1)
* <Question: what's new?>
* Question.objects.get(pk=2)
* <Question: what's up????>
* Question.objects.get(pk=3)
* <Question: Welcome>
* q.was\_published\_recently()
* True

**Create**

* q.choice\_set.create(choice\_text='Not Much', votes=0)
* q.choice\_set.create(choice\_text=’The Sky', votes=0)
* q.choice\_set.create(choice\_text=’The hacking again', votes=0)
* c.question

**all questions**

* q.choice\_set.all()

**Count**

* q.choice\_set.count()

**Starts With**

* c=q.choice\_set.filter(choice\_text\_\_startswith='Just hacking')

**Delete**

* c.delete()

**Admin Part**

* In Django webpages and other content are delivered by views.
* View is represented by a function or method in the case of class based views.

**Select\_related():**follows foreign-key relationship, selecting additional related-object data when it executes its query.

1.Select\_related when the object that selecting is a single object, so OneToOneFIeld or a ForeignKey.

2.Does not work either select\_related before prefetch\_related.

**Prefetch\_related():** does a separate lookup for each relationship and does the joining in python.

ManyToMany fields or reverse ForeignKeys.

* Class A(models.Model):

Pass

Class B(models.Model):

A=models.ForeignKey(ModelA, on\_delete=models.CASCADE)

ModelB.objects.select\_Related(‘a’).all()

ModelA.objects.prefetch\_related(‘modelb\_set’).all()

* Select\_related when you have a one-to-one relationship.OnetoOneField and ForeignKey which is known as many-to-one.
* Use prefetch\_related when you have a manytomany releationship.manytomany field and reverse foreignkey which is known as one-to-many.
* **Types of views in Django**

**A view is a callable which takes a request and return a response.**

1.function-based views

2.class-based views -inherit the method as\_view() which uses a dispatch() method to call the appropriate method depending on the HTTP methods(GET, POST).

They must accept an HttpRequest object as its first positional argument.they must return an HttpResponse object or raise an exception.

* Many-To-Many relationship, the different books can be related to different autors.three tables schema I described is the basic design for it.
* One-To-One relationship are like FOreignKey but with a unique =True.you can only link between one object to another object.it is usually used when you want to expand a certain model without changing the original .

**Django Get() and Filter()**

* Get() when you want to get a single unique object.
* Filter()get all objects that match with lookup parameters.

Modelname.objects.get(name=’nagur’)

Modelname.objects.filter(name=’nagur’)

**Django Seralizers:**

* Seralizers in Django rest framework are responsible to converting objects into data types understandable by JavaScript and front-end framework.
* Serializers also provide deserialization, allowing parsed data to be converted back into complex types, after first validating the ncoming data.

**Django Normalization:**

* It is designing a relational database schema such that you remove duplication and eliminate redundancy.
* In Django we have one model, might have multiple models which represent the relationship between database tables via ForeignKey and ManyToMany fields provided by the Model API.

1. ManyToMany RelationShip (models.ManyToManyField(fieldname)
2. ManyToOne Relationship

(models.ForeignKey(on\_delete=models.CASCADE)

1. OneToOneRelationship(models.FOreignKey(fieldname, on\_delete=models.CASCADE))

**OneToOne :**

* A one-to-one relationship is similar to many-to-one relationship.
* It restricts two objects to having a unique relationship.

**ManyToMany:**

* A many-to-many relationship between two mdoels defines that zero, one or more objects of the first model may be related to zero,one or more objects of the second model.