DJANGO SHELL

Django Shell is a shell that provides an interactive console to execute the queries on the database created using Django. In other words we can say that it is a Django shell is a Python shell that gives you access to the database API included with Django.

Let us consider the tables which we created in models.py file of our app named appDemo. On these tables we will perform CRUD operations in Django Shell.

```
from django.db import models

# Create your models here.

class bankDetails(models.Model):

        bankname = models.CharField(max_length=40)

        address = models.CharField(max_length=40, blank=True, null=True)

        DoE = models.DateTimeField() #Date of Establishment

        bankID = models.IntegerField(null=False)

        url = models.URLField(default='www.rbi.co.in')


        def __str__(self):

                return (self.bankname)


class bankAccount(models.Model):

        username = models.CharField(max_length=40)

        mybankname = models.ForeignKey(bankDetails,
on_delete=models.CASCADE)

        surname = models.CharField(max_length=40, blank=True, null=True)
```

```
        emailid = models.EmailField(max_length=40)

        DoI = models.DateTimeField() #Date of Initiation

        accnumber = models.IntegerField()

        url = models.URLField(default='www.google.com')



        def __str__(self):

                return (self.username+" account details")
```

Starting a Shell

(MYENV) E:\000_WorkingDemo\projectDemo>python manage.py shell

Importing models

>>> *from appDemo.models import bankDetails, bankAccount*

>>> bankDetails.objects.all()

OUTPUT: <QuerySet [<bankDetails: SBI>, <bankDetails: SIB>, <bankDetails: HDFC>, <bankDetails: Kotak>]>

Create

NOTE: Table should be created in the model.py file or in database directly. Shell doesn't allow you to create a new table which is not configured with Django.

Inserting Values & Saving

>>> var1= bankDetails (bankname="ICICI", address="Paris", DoE = '10-09-2020', bankID = 128, url ='www.rbi.co.in')

>>> var1.save

OUTPUT: <bound method Model.save of <bankDetails: ICICI>>

>>> bankDetails.objects.all()

<QuerySet [<bankDetails: SIB>, <bankDetails: HDFC>, <bankDetails: Kotak>, <bankDetails: ICICI>]>

>>>

Accessing individual attributes

>>> var1.bankname

'ICICI'

>>> var1.address

'Paris'

>>> var1.DoE

'10-09-2020'

>>> var1.bankID

128

>>> var1.url

'www.rbi.co.in

Updating the attribute directly

>>> var1.address='Norway'

>>> var1.address

'Norway'

>>> var1.save

Updating through query

```
from appDemo.models import bankDetails
obj1 = bankDetails.objects.get(id=2) ###Condition
obj1.bankname = "ABC" ##Changes to be updated
obj1.save()
```

```
from appDemo.models import bankAccount
```

```
obj2= bankAccount.objects.get(url='www.google.com')

obj2.url='www.google.co.in'

obj2.save()
```

Batch Update

```
bankAccount.objects.select_for_update().filter(url='www.google.com').upd
ate(url='www.google.co.in')
```

OutPUT: 2

Delete query

```
from appDemo.models import bankDetails
# query1
obj1 = bankDetails.objects.get(id=1)
obj1.delete()
# query2
bankDetails.objects.filter(id=1).delete()
```

OUTPUT: (2, {'appDemo.bankAccount': 1, 'appDemo.bankDetails': 1})

Batch Delete

```
bankDetails.objects.select_for_update().filter(url='www.google.com').d
elete()
```

OUTPUT: (0, {})

Read All objects

```
bankAccount.objects.all()
```

```
OUTPUT: <QuerySet [<bankAccount: kashi account details>,
<bankAccount: Sripati account details>]>
```

Filtering

```
From appDemo.models import bankAccount
```

```
bankAccount.objects.filter(id=1, surname__contains='Patil')

OUTPUT: <QuerySet []>
```

```
From appDemo.models import bankAccount
```

```
bankAccount.objects.filter(id=1, surname='Patil').exclude(id=1)

OUTPUT: <QuerySet []>
```