# CHAPTER - IV

# INDEX PAGE

**1. Model Queries using django shell**

1.1. Creating objects

1.1.1. Creating Objects With save()
1.1.2. Creating Objects With create()

1.2. Retrieving objects

1.2.1. Retrieving all objects
1.2.2. Retrieving specific objects with filters
1.2.3. Retrieving a single object with get()
1.2.4. Retrieving objects with exclude()
1.2.5. Retrieving objects with Chaining filters
1.2.6. Retrieving objects Using order_by()

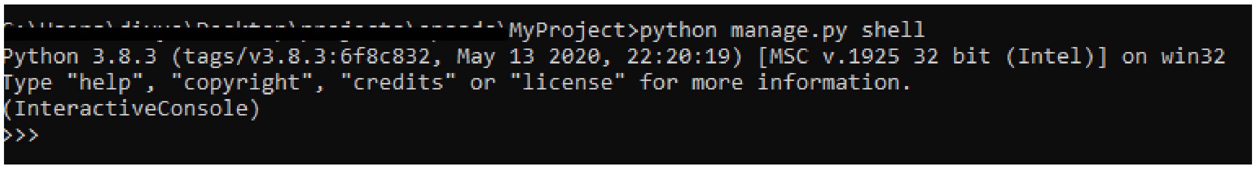
1.3. Saving changes to objects
1.4. Updating multiple objects at once
1.5. Deleting objects in 2 simple steps
1.6. Limiting QuerySets
1.7. Field lookups
1.7.1. Exact
1.7.2. Iexact
1.7.3. Contains

**2. Superuser Creation**

# 1. Model Queries using django shell

**step-4**

open the command prompt from the location of project's manage.py, run the command “python manage.py shell” to get interactive shell to work



```
[redacted]MyProject>python manage.py shell
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>>
```

**End-step-4**

Now we will practice CRUD operations from the command prompt.

## 1.1. Creating objects

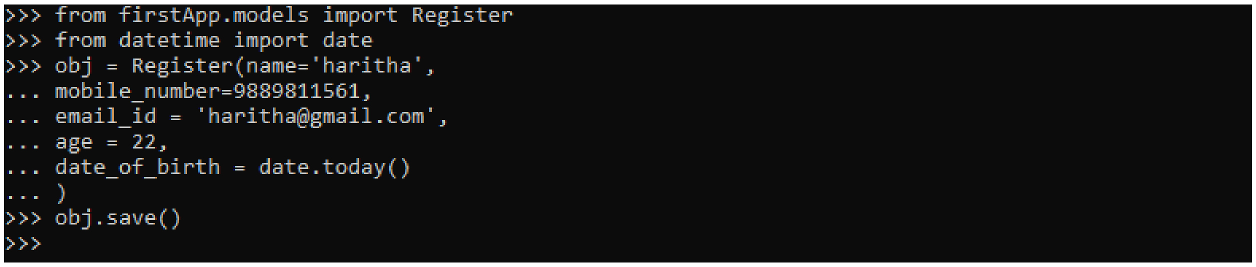
To represent database-table data in Python objects, Django uses an intuitive system: A model class represents a database table, and an instance of that class represents a particular record in the database table.

### 1.1.1. **Creating Objects With save()**

To create an object, instantiate it using keyword arguments to the model class, then call save() to save it to the database.

Assuming models live in a file YOUR_PROJECT_NAME/YOUR_APP_NAME/models.py

step-5 do from current shell(already we done in step-4)

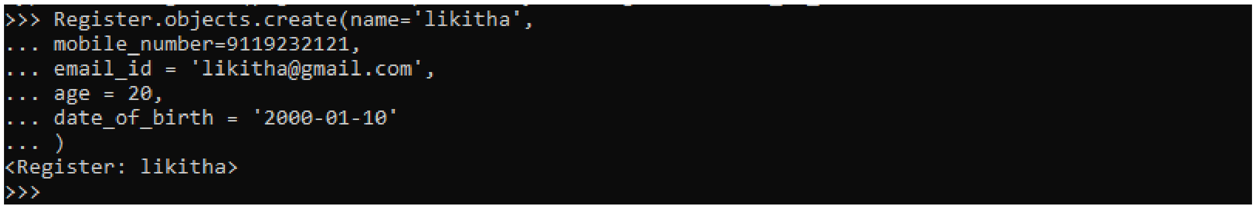


```
>>> from firstApp.models import Register
>>> from datetime import date
>>> obj = Register(name='haritha',
... mobile_number=9889811561,
... email_id = 'haritha@gmail.com',
... age = 22,
... date_of_birth = date.today()
... )
>>> obj.save()
>>>
```

This performs an INSERT SQL statement behind the scenes. Django doesn’t hit the database until you explicitly call save().

### 1.1.2. **Creating Objects With create()**

We can directly create instances without assigning objects by using the create method.



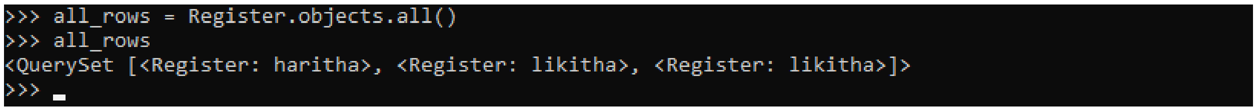
```
>>> Register.objects.create(name='likitha',
... mobile_number=9119232121,
... email_id = 'likitha@gmail.com',
... age = 20,
... date_of_birth = '2000-01-10'
... )
<Register: likitha>
>>>
```

## 1.2. Retrieving objects

To retrieve objects from your database, construct a QuerySet

### 1.2.1. Retrieving all objects

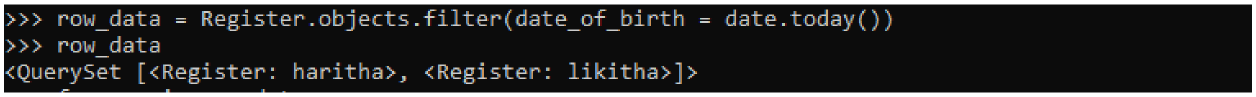
The all() method returns a QuerySet of all the objects in the database.



```
>>> all_rows = Register.objects.all()
>>> all_rows
<QuerySet [<Register: haritha>, <Register: likitha>, <Register: likitha>]>
>>>
```

### 1.2.2. Retrieving specific objects with filters

example to get a QuerySet of date_of_birth from the year 2020

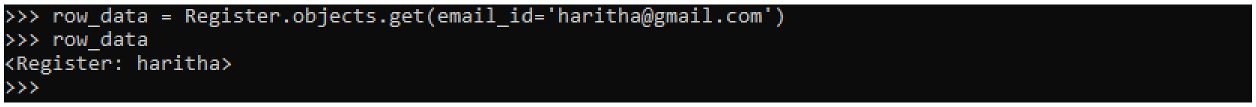


```
>>> row_data = Register.objects.filter(date_of_birth = date.today())
>>> row_data
<QuerySet [<Register: haritha>, <Register: likitha>]>
```

### 1.2.3. Retrieving a single object with get()

filter() will always give you a QuerySet, even if only a single object matches the query - in this case, it will be a QuerySet containing a single element.

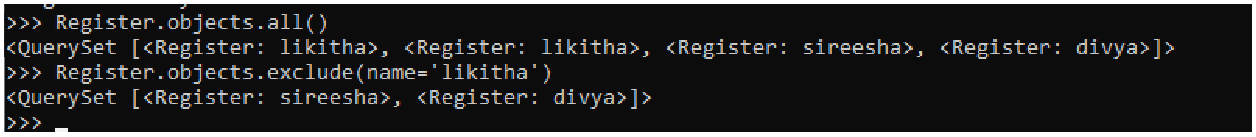
If you know there is only one object that matches your query, you can use the get() method



```
>>> row_data = Register.objects.get(email_id='haritha@gmail.com')
>>> row_data
<Register: haritha>
>>>
```

### 1.2.4. Retrieving objects with exclude()

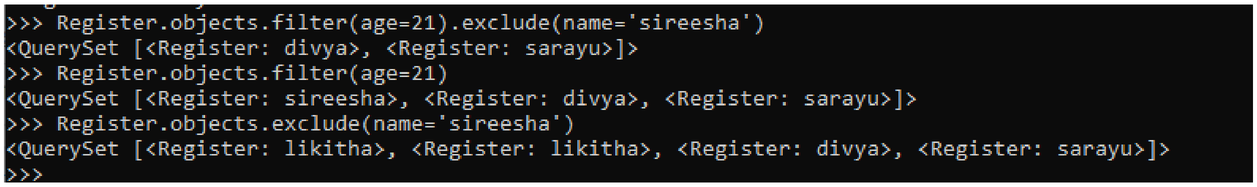
Returns a new QuerySet containing objects that do not match the given lookup parameters.



```
>>> Register.objects.all()
<QuerySet [<Register: likitha>, <Register: likitha>, <Register: sireesha>, <Register: divya>]>
>>> Register.objects.exclude(name='likitha')
<QuerySet [<Register: sireesha>, <Register: divya>]>
>>>
```

### 1.2.5. Retrieving objects with Chaining filters

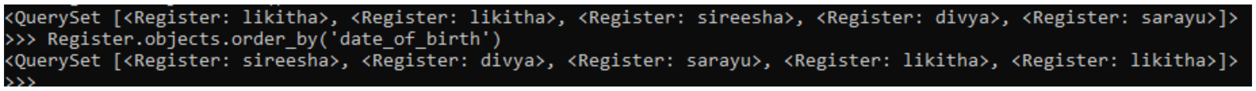
The result of refining a QuerySet is itself a QuerySet, so it's possible to chain refinements together.



```
>>> Register.objects.filter(age=21).exclude(name='sireesha')
<QuerySet [<Register: divya>, <Register: sarayu>]>
>>> Register.objects.filter(age=21)
<QuerySet [<Register: sireesha>, <Register: divya>, <Register: sarayu>]>
>>> Register.objects.exclude(name='sireesha')
<QuerySet [<Register: likitha>, <Register: likitha>, <Register: divya>, <Register: sarayu>]>
>>>
```

### 1.2.6. Retrieving objects Using order_by()

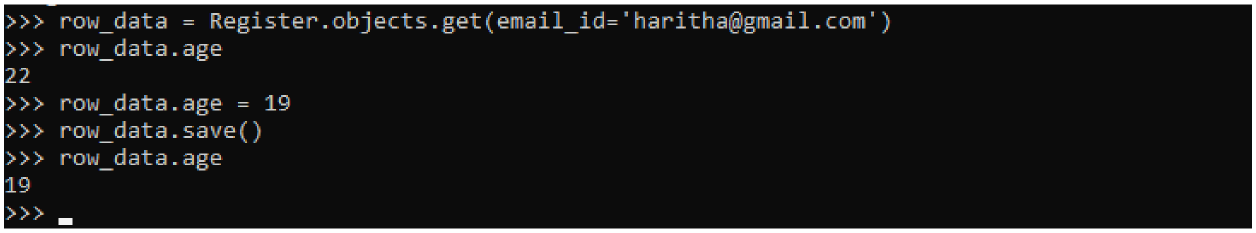
By default, results returned by a QuerySet are ordered by the ordering tuple given by the ordering option in the model's Meta. You can override this on a per-QuerySet basis by using the order_by method.



```
<QuerySet [<Register: likitha>, <Register: likitha>, <Register: sireesha>, <Register: divya>, <Register: sarayu>]>
>>> Register.objects.order_by('date_of_birth')
<QuerySet [<Register: sireesha>, <Register: divya>, <Register: sarayu>, <Register: likitha>, <Register: likitha>]>
>>>
```

## 1.3. Saving changes to objects in 3 simple steps(Update)

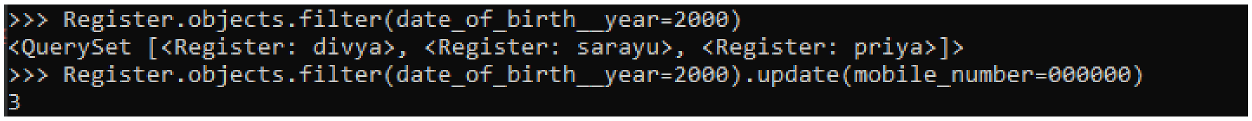
1.get the existing object
2.do the modification
3.save the object



```
>>> row_data = Register.objects.get(email_id='haritha@gmail.com')
>>> row_data.age
22
>>> row_data.age = 19
>>> row_data.save()
>>> row_data.age
19
>>>
```

## 1.4. Updating multiple objects at once

Sometimes you want to set a field to a particular value for all the objects in a QuerySet. You can do this with the update() method. For example:

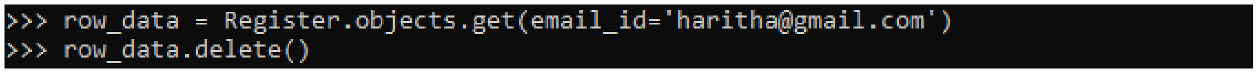


```
>>> Register.objects.filter(date_of_birth__year=2000)
<QuerySet [<Register: divya>, <Register: sarayu>, <Register: priya>]>
>>> Register.objects.filter(date_of_birth__year=2000).update(mobile_number=000000)
3
```

## 1.5. Deleting objects in 2 simple steps

1. get the existing object
2.delete the object

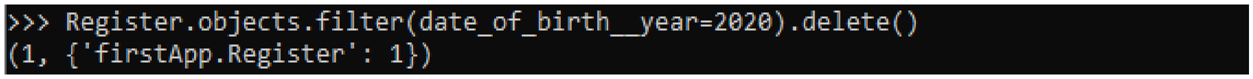
Delete() method directly deletes the object from our DataBase.



```
>>> row_data = Register.objects.get(email_id='haritha@gmail.com')
>>> row_data.delete()
```

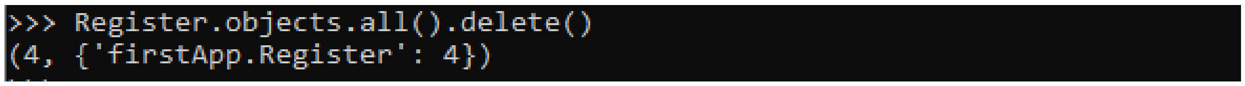
You can also delete objects in bulk. Every QuerySet has a delete() method, which deletes all members of that QuerySet.

For example, this deletes all Entry objects with a pub_date year of 2005:



```
>>> Register.objects.filter(date_of_birth__year=2020).delete()
(1, {'firstApp.Register': 1})
```

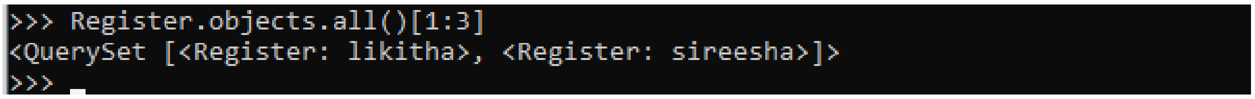
deleting all the entries. If you do want to delete all the objects, then you have to explicitly request a complete query set



```
>>> Register.objects.all().delete()
(4, {'firstApp.Register': 4})
```

## 1.6. Limiting QuerySets

Use a subset of Python's array-slicing syntax to limit your QuerySet to a certain number of results. This is the equivalent of SQL's LIMIT and OFFSET clauses.



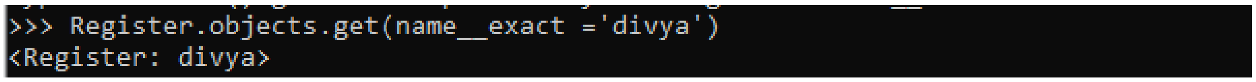
```
>>> Register.objects.all()[1:3]
<QuerySet [<Register: likitha>, <Register: sireesha>]>
>>> _
```

## 1.7. Field lookups

Field lookups are how you specify the meat of an SQL WHERE clause. They're specified as keyword arguments to the QuerySet methods filter(), exclude() and get().

### 1.7.1. Exact

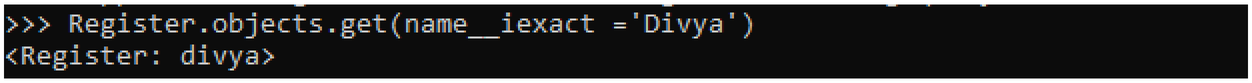
An "exact" match. For example:



```
>>> Register.objects.get(name__exact ='divya')
<Register: divya>
```

### 1.7.2. Iexact

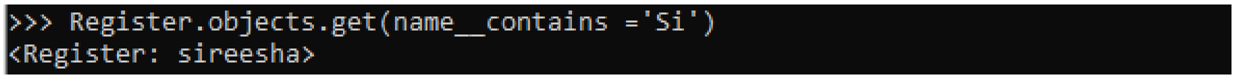
A case-insensitive match. So, the query:



```
>>> Register.objects.get(name__iexact ='Divya')
<Register: divya>
```

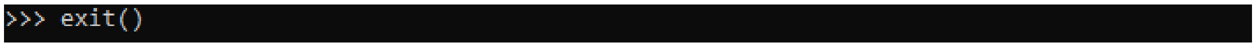
### 1.7.3. Contains

Case-sensitive containment test. For example:



```
>>> Register.objects.get(name__contains ='Si')
<Register: sireesha>
```

if we want to exit from shell use below command



```
>>> exit()
```

**end-step-5**

## 2. Superuser Creation

Giving all permissions to particular user (admin).

Generating admin sites for your staff or clients to add, change, and delete content is tedious work that doesn't require much creativity. For that reason, Django entirely automates creation of admin interfaces for models

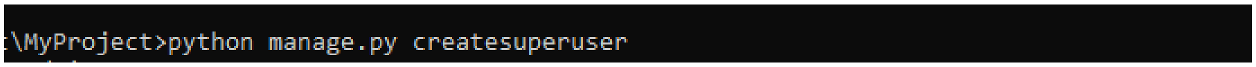
Note:

- The admin isn't intended to be used by  site visitors. It's for site managers.

**step-6**

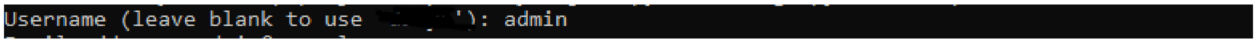
open the command prompt from the location of project's manage.py

First we'll need to create a user who can login to the admin site. Run the following command:



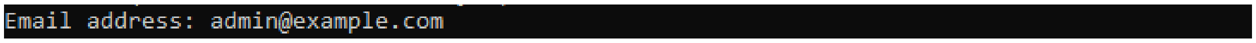
```
:\MyProject>python manage.py createsuperuser
```

Enter your desired username and press enter



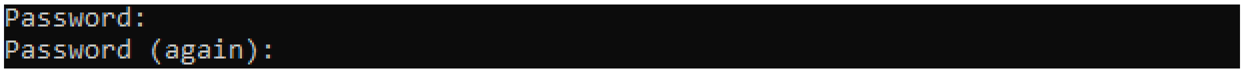
```
Username (leave blank to use '[redacted]'): admin
```

You will then be prompted for your desired email address:



```
Email address: admin@example.com
```

The final step is to enter your password. You will be asked to enter your password twice, the second time as a confirmation of the first

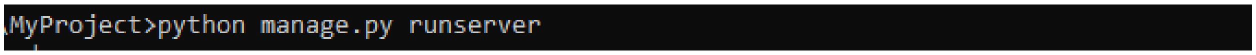


```
Password:
Password (again):
```

**end-step-6**

Finally you will get a Superuser created successfully message.

**Start the development server**



```
MyProject>python manage.py runserver
```

Now, open a Web browser and go to "/admin/" on your local domain – e.g., http://127.0.0.1:8000/admin/