Django Proxy Model Inheritance :

- Django supports Proxy model inheritance to control the model by using another model.
- Proxy model is a model which controls or manages the non-abstract model.
- The meaning of Proxy is "Controlling".
- By using Proxy model, we can perform all CURD operations on the non-abstract model.
- We have to create Meta model to make a model as proxy model
- We have to set proxy = True in the meta model.
- Django does not creates database table for proxy model separetly. It is using existing
- model table only.

Example Structure:

```
class Employee(models.Model):
      eno   = …
      ename  = ..
      salary = …
      address = …

class EmployeeProxy(Employee):
     class Meta:
          proxy = True
```

Create a project using Proxy Model Inheritance:

Step1: Create a Django ProjectName like **Proxy_Project**

Step2: Create a Application Name like **Proxy_App**

Step3: Create Database Name like **7am_proxydb**

Step4: Goto settings.py file and configure database details under DATEBASE section.

```
DATABASES = {
   'default': {
      'ENGINE'  :  'django.db.backends.mysql',
      'NAME' : '7am_proxydb',
      'USER' : 'root',
      'PASSWORD' : 'root',
   }
}
```

Step5: Open models.py file and create required models

```
from django.db import models

class Employee(models.Model):
    eno = models.IntegerField()
    ename = models.CharField(max_length=30)
    salary = models.DecimalField(max_digits=10, decimal_places=2)
    address = models.TextField()

    class Meta:
        db_table = 'employee'

    def __str__(self):
        return self.ename

class EmployeeProxy(Employee):
    class Meta:
        proxy = True
```

Step6: Goto admin.py file and write the following code

```
from django.contrib import admin
from Proxy_App.models import Employee,EmployeeProxy

@admin.register(Employee)
class EmployeeAdmin(admin.ModelAdmin):
    list_display = ['eno', 'ename','salary','address']
    search_fields = ['ename']
    ordering = ['salary']

@admin.register(EmployeeProxy)
class EmployeeProxyAdmin(admin.ModelAdmin):
    list_display = ['eno','ename','address','salary']
    search_fields = ['address']
    ordering = ['-salary']

# admin.site.register(Employee, EmployeeAdmin)
# admin.site.register(EmployeeProxy, EmployeeProxyAdmin)
```

Step8: Execute the makemigrations command to convert model code into SQL code format

```
python manage.py makemigrations
```

Step9: Execute the migrate command to execute SQL code in database site and creating tables more models.

```
python  manage.py   migrate
```

Step10: Execute the createsuperuser command for creating admin creadentials.

```
python  manage.py   createsuperuser
```

Then it will ask like below details,

```
Username:   Virat
Email : virat@gmail.com
Password:    admin123
Password (again):  admin123
```

Step11: Now execute the runserver command for running the project

```
python  manage.py   runserver   8000
```

Step12: Now open the required browser and then send **admin/** url request from the browser then we will get admin login page response like below

Step13: Login to admin site and add some data into proxy model table.

Now goto **Employee** model and check the data, it is also must have the same data which is in **EmpProxy** model

Note: Here using **EmployeeProxy** model table we can managing the Employee model table and also to performing the all CURD operations.

In admin site, Employee model table contains the data which is given by insertion order. EmployeeProxy table we are using for dispalying the data according to admin requirements and admin easy interaction purpose.

In admin site, employee table like bellow displays

```
mysql> select  *  from  employee;

ID      ENO  ENAME        ADDRESS        SALARY
------------------------------------------------------------
1       10      Srinivas     Hyderabad       50000
2       20      Rohit        Mumbai          30000
3       30      Virat        Delhi           40000
```

In admin site, **Employee** table like bellow displays according to **salary** based **Ascending** order.

ID	ENO	ENAME	ADDRESS	SALARY
2	20	Rohit	Mumbai	30000
3	30	Virat	Delhi	40000
1	10	Srinivas	Hyderabad	50000

In admin site, **EmployeeProxy** table like bellow displays according to **salary** based **descending** order.

ID	ENO	ENAME	ADDRESS	SALARY
1	10	Srinivas	Hyderabad	50000
3	30	Virat	Delhi	40000
2	20	Rohit	Mumbai	30000

Differences between abstract and proxy models inheritances

Abstract model inheritance:

1. Abstract model is a model which has the common fields of all non-abstract models.
2. Abstract model contains the common fields of one or more non-abstract models.
3. Django will not create database table for Abstract model.

4 We set abstract = True in the Meta model.

5 Abstract model gives the fields to non-abstract models at runtime.

6 We can not manage non-abstract models through abstract model at runtime.

7 Abstract model name is the super model or parent model to all non-abstract models.

Proxy model Inheritance:

1. Proxy model is model which controls or manages non-abstract model.
2. Proxy model can control or manages only one non-abstract model.
3. Django will not create database table for Proxy model but it can access non-abstract model table.

4 We set proxy = True in the Meta model.

5 Proxy model takes the all fields from non-abstract model at runtime.

6 we can manage the non-abstract model through the proxy model.

7 non-abstract model name is the super model or parent model to the proxy model.