Many to One Model Relationship

- If we want Many to One relationaship then multiple child records can be depended on single parent record.
- For example, multiple Courses like Python, Django are joined by single Student like Virat.
- We use **ForeignKey()** field type to implement Many-To-One Relationship.

commoninfo = models.ForeignKey(Parent, on_delete=models.CASCADE)

Q. Create a Django Project using ManyToOne Model Relationship

Step1: Create a Django ProjectName like ManyToOne_Project

Step2: Create a Application Name like ManyToOne_App

Step3: Create Database Name like 7am_mtodb

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

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

Step6: Open **models.py** file and create required models

```
from django.db import models

class Student(models.Model):
   sno = models.IntegerField()
   sname = models.CharField(max_length=30)
   marks = models.IntegerField()
   def __str__(self):
      return self.sname
```

```
class Course(models.Model):
    cno = models.IntegerField()
    cname = models.CharField(max_length=30)
    cfee = models.FloatField()

    student = models.ForeignKey(Student, on_delete=models.CASCADE)

    def __str__(self):
        return self.cname
```

Step7: open **admin.py** file and create required admin logics

```
from django.contrib import admin
from OneToOne_App.models import Student, Course

class StudentAdmin(admin.ModelAdmin):
    list_display = ['sno', 'sname', 'marks']

class CourseAdmin(admin.ModelAdmin):
    list_display = ['cno', 'cname', 'cfee', 'student_id']

admin.site.register(Student, StudentAdmin)
admin.site.register(Course,CourseAdmin)
```

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

Now we will see our Student and Course model related tables in Admin site.

Now add some records into both Student and course tables and open them.

mysql> select * from onetoone_app_student;

```
+----+-----+--------+-------------+
| id | sno | sname  | marks |
+----+-----+--------+-------------+
|  1 | 101 | Virat  |    85     |
|  2 | 102 | Rohit  |    90     |
|  3 | 103 | Surya  |    75     |
|  4 | 104 | Dravid |    95     |
+----+-----+--------+------------+
```

mysql> select * from onetoone_app_course;

```
+----+-----+--------+------+--------------------+
| id | cno | cname  | cfee | student_id |
+----+-----+--------+------+--------------------+
|  1 | 201 | Python | 5000 |          1 |
|  2 | 202 | Django | 6000 |          1 |
|  3 | 203 | RESTAPI| 4000|          2 |
| 4  | 204 | MySQL |  3000 |          3 |
+----+-----+--------+------+----------------+
```

Now you can Perform all required CURD operations using this admin site presentation.

Here, Python and Django child records are depended on single Virat parent record.
REST API child records is depended on single Rohit parent record.

Add cascading rule to manage the child records when the corresponding Parent records are deleted or updated, like One-to-One relationaship.