**CRUD OPERATIONS IN DJANGO**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***CREATE***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

WAY TO ENTER DATA IN PYTHON DJANGO SHELL

**CLASS MODEL**

class Car(models.Model):

car\_name = models.CharField(max\_length=100)

    speed = models.IntegerField(default=50)

    def \_\_str\_\_(self) -> str:

        return self.car\_name

1.

car= Car(car\_name= 'Tata Nexon', speed= 50)

car.save()

2.

Car.objects.create(car\_name = 'XUV700', speed=200)

unlike 1st no need to save()

3.

car\_dict = {"car\_name":"Renault Kiger", "speed":"300"}

Car.objects.create(\*\*car\_dict)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***READ***

WAY TO READ DATA IN PYTHON DJANGO SHELL

1.

Car.objects.all()[0].car\_name

2.

By using for loop

cars = Car.objects.all()

for car in cars:

print(f"the car name is {car.car\_name} and speed is {car.speed}")

3.

BY USING THE GET() METHOD

car = Car.objects.get(id =2)

***IF THE GIVEN ID DOES NOT EXISTS THE IT WILL THROW AN ERROR***

4.

BY USING THE FILTER() METHOD

car = Car.objects.filter(id=1000)

IF DATA EXISTS FOR THE GIVEN ID THEN IT WILL RETURN A QUESRYSET WITH DATA IN IT BUT IF THE ID IS NOT AVAILABLE OR NO MATCH FOUND THEN IT WILL RETURN AN EMPTY QUERYSET []

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***UPDATE***

WAY TO UPDATE DATA IN PYTHON DJANGO SHELL

1.

car = Car.objects.get(id=1)

car.car\_name = "CREATA"

car.speed= 400

car.save()

2.

car = Car.objects.filter(id=1).update(car\_name= 'Dark edition')

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***DELETE***

WAY TO DELETE DATA IN PYTHON DJANGO SHELL

1.

car = Car.objects.filter(id=2).delete()

car

2.

car = Car.objects.filter(id=2).delete()

car

TO DELETE EVERYTHING

car = Car.objects.all().delete()

***FETCHING DATA BY ORDER AND LIMITING IT***

**Receipe.objects.all().order\_by('receipe\_view\_count')[0:3]**

Here after ‘receipe\_view\_count’ if we put a ‘-’ then this will generate the result in descending order.

**Receipe.objects.all().order\_by('-receipe\_view\_count')[0:3]**

**Here [0:3] describes the total number ot output after the query like in this example it will give output of only initial 3 results only (from 0 to 3).**

**Receipe.objects.filter(receipe\_view\_count\_\_gte=80)**

**This will generate the result in which receipe\_view\_count is greater than 80.**

**Receipe.objects.filter(receipe\_view\_count\_\_lte=80)**

**This will generate the result in which receipe\_view\_count is less than 80.**

**------------------------------------------*RETRIEVING THE DESIRED DATA FROM DJANGO***

queryset= Student.objects.filter(department\_\_department\_\_in=d)

Here the first department is of the model name and second department is of the field inside it.

d is the list of values to be retrieves ie. Computer science , bio technical etc.

Student.objects.exclude(department\_\_department\_\_in=d)

This will exclude the given list from the database in the results. D can be a String or it can be a list of different strings.

queryset = Student.objects.values\_list('student\_name', 'student\_age')

Student.objects.aggregate(Avg('student\_age'))

This will give result of student\_name and student\_age only.

It return data in tuple only.

**------------------------------------------*AGGREGATE FUNCTIONS IN DJANGO***

Student.objects.aggregate(Avg('student\_age'))

Student.objects.aggregate(Max('student\_age'))

This will give the avg age of student from the list .

Student.objects.aggregate(Max('student\_age'))

This will give the max age of student from the list .