**ORM(**Object Relational Mapping**)**

The Django web framework includes a default object-relational mapping layer (ORM) that can be used to interact with data from various relational databases such as SQLite, PostgreSQL, and MySQL. Django allows us to add, delete, modify, and query objects, using an API called ORM. ORM stands for Object Relational Mapping. An object-relational mapper provides an object-oriented layer between relational databases and object-oriented programming languages without having to write SQL queries.

**Step 1:- Defining Models**

You create Python classes that subclass **django.db.models.Model**. Each class represents a database table, and attributes represent fields in the table.

A screen shot of a computer

Description automatically generated

**Step 2:-** **Creating Migrations**

Django's migration system allows you to create, apply, and manage changes to your models and database schema. After defining a model, you create a migration file that captures these changes:

**python manage.py makemigrations**

**python manage.py migrate**

**Step 3:-** **Create interactive console**

By using the shell, we can easily test and make changes to a project's models.

**Py manage.py shell**

A black screen with white text

Description automatically generated

**Step 4:- To import model from the database**

**from APPNAME.models import MODELCLASSNAME**

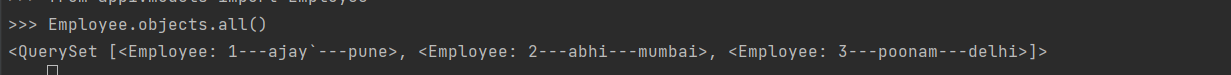


**Querying the Database:**

You can perform database operations using the model classes and methods provided by Django's ORM.

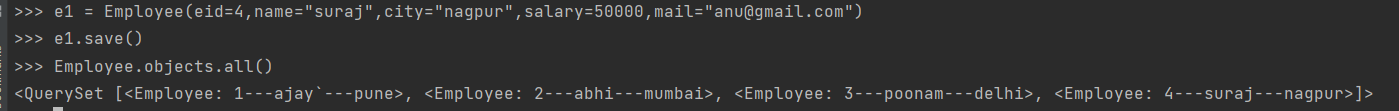
1. **How to get all records from table(Model)**

We have a model called **Employee.** To get all records from model, we will use the **Employee.objects.all().**



1. **Add new Employee record in Employee Model:**

Create a new instance instance = YourModel(name='John', age=25) instance.save()

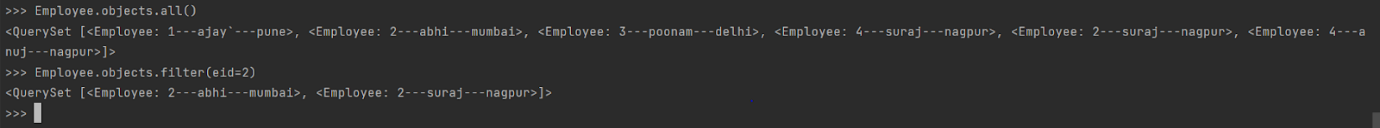


1. **Fetch particular record**



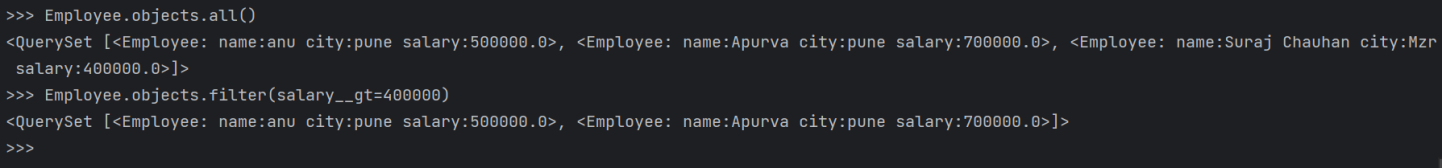
1. **Fetch all record depend on condition**

YourModel.objects.filter(condition)



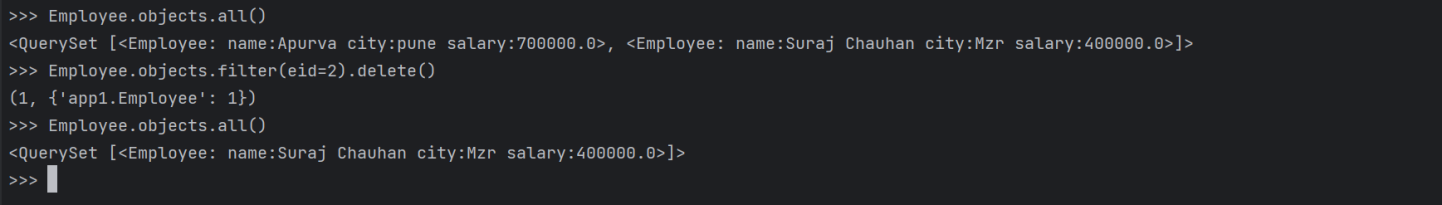
1. **Filtering records where salary is greater than 400000**

Querying results = YourModel.objects.filter(salary\_\_gt=400000)



1. **Delete particular Record**

Delete instance = YourModel.objects.get(name='John') instance.delete()



1. **Update Particular Record**



Update instance = YourModel.objects.get(fetch record)

instance.name = “update name”

instance.save()

The Django ORM abstracts away many complexities of database interaction, allowing you to focus on your application's logic. It supports various databases and provides a high level of flexibility in handling database operations.

**Querys**

1. objs = MODEL.objects.all()

2. obj = MODEL.objects.get(col=val) -->OBJ

3. objs = MODEL.objects.filter(col=val) -->QS

4. objs = MODEL.objects.exclude(col=val)

5. objs = MODEL.objects.filter(col\_\_lt = val)

6. objs = MODEL.objects.filter(col\_\_gt = val)

7. objs = MODEL.objects.filter(col\_\_lte = val)

8. objs = MODEL.objects.filter(col\_\_gte = val)

9. objs = MODEL.objects.filter(col\_\_startswith = 'val')

10. objs = MODEL.objects.filter(col\_\_endswith = 'val')

11. objs = MODEL.objects.filter(col\_\_contains = 'val')

12. objs = MODEL.objects.order\_by('col')

12. objs = MODEL.objects.order\_by('-col')

1. **Manipulating Data**: You can update, delete, or retrieve data using ORM methods:

#Update

13. obj = MODEL.objects.get(col=val)

obj.COL = NEWVAL

obj.save()

#Delete

14. obj = MODEL.objects.get(col=val)

obj.delete()

#Insert

15. obj = MODEL(col1=val,col2=val,col3=val..)

obj.save()

OR

obj = MODEL()

obj.col1 = val

obj.col2 = val

obj.col3 = val

obj.save()

**Aggregation Funs**

from django.db.models import Max,Min,Avg,Sum,Count

16. mx = MODEL.objects.all().aggregate(Max('col'))

17. mi = MODEL.objects.all().aggregate(Min('col'))

18. sm = MODEL.objects.all().aggregate(Sum('col'))

19. av = MODEL.objects.all().aggregate(Avg('col'))

20. ct = MODEL.objects.all().aggregate(Count('col'))

#And

21. objs = MODEL.objects.filter(condition1 , condition2)

#Or

22. objs = MODEL.objects.filter(condition1) | MODEL.objects.filter(condition2)

from django.db.models import Q

23. objs = MODEL.objects.filter(Q(condition1) | Q(condition2))

a]Student.objects.filter(marks=Student.objects.all().aggregate(mm=Max('marks'))['mm']).first()

b]max\_marks = Student.objects.all().aggregate(mm=Max('marks'))['mm']

Student.objects.filter(marks=max\_marks).first()