**Django ORM(Object Relational Mapping)**

* One of the most powerful features of Django is its Object-Relational Mapper (ORM), which enables you to interact with your database, like you would with SQL.
* In fact, Django's ORM is just a pythonical way to create SQL to query and manipulate your database and get results in a pythonic fashion.

Go to project location and type the below command

python manage.py shell

It will opens interactive console

**Retrieving Data in SQL**

Select \* from admissions\_student;

**ORM**

from admissions.models import student

s=student.objects.all()

print(s)

**To select particular id in SQL**

select \* from admissions\_student where id=3;

**ORM**

s=student.objects.get(id=3)

print(s)

s=student.objects.get(name="abc")

print(s)

**To select the particular columns in SQL**

select id,name from admissions\_student;

**Selective Columns in ORM (values\_list or values or only)**

**values\_list**

s=student.objects.values\_list('name', 'contact') #output will be displayed in tuple format

for s1 in s:

print(s1)

**values**

s=student.objects.values('name', 'contact') #output will be displayed in dictionary format

for s1 in s:

print(s1)

**only**

s=student.objects.only('name', 'contact') #return rows as objects and adds id field.

for s1 in s:

print(s1)

**To select the records based on condition using SQL**

Select id,name,contact from admissions\_student where id>3;

**ORM**

**Greater than condition**

s=student.objects.filter(id\_\_gt=3)

print(s)

**Greater than equal to condition**

s=student.objects.filter(id\_\_gte=3)

print(s)

**To select the records into the table by selecting the middle character using LIKE Operator SQL**

select \* from admissions\_student where name LIKE '%bc%'

**ORM**

s=student.objects.filter(name\_\_contains= 'bc')

for s1 in s:

print(s1.name)

Here **contains** is **case sensitive** it will checks the record into the table must be in same case only

s=student.objects.filter(name\_\_icontains= 'bc')

for s1 in s:

print(s1.name)

Here **icontains** is **case insensitive** it will checks the record into the table must be same

**To select the records into the table by selecting the first character using LIKE Operator SQL**

select \* from admissions\_student where name LIKE 'b%'

**ORM**

s=student.objects.filter(name\_\_startswith= 'b')

for s1 in s:

print(s1.name)

**To select the records into the table by selecting the first character using LIKE Operator SQL**

select \* from admissions\_student where name LIKE 'b%'

**ORM**

s=student.objects.filter(name\_\_startswith= 'b')

for s1 in s:

print(s1.name)

**To select the records into the table by selecting the lastcharacter using LIKE Operator SQL**

select \* from admissions\_student where name LIKE '%c'

**ORM**

s=student.objects.filter(name\_\_endswith= 'c')

for s1 in s:

print(s1.name)

**In Operator in SQL**

select \* from admissions\_student where id in(1,2,5);

**ORM**

s=student.objects.filter(id\_\_in=[1,2,5])

for s1 in s:

print(s1.id)

**Logical Operator in SQL**

**And in SQL**

select \* from admissions\_student where id>2 and classname>6;

**ORM**

s=student.objects.filter(id\_\_gt=2)& student.objects.filter(classname\_\_gt=6)

**Or in SQL**

select \* from admissions\_student where id>2 or classname>6;

**ORM**

s=student.objects.filter(id\_\_gt=2)| student.objects.filter(classname\_\_gt=6)

**Not in SQL**

select \* from admissions\_student where not id>2

**ORM**

s=student.objects.exclude(id\_\_gt=2)

**Inserting data into tables**

**SQL**

Insert into admissions\_student values(1, 'abc','bb',6,'8976547897');

**ORM**

**To insert single record**

s=student(name="sam",fathername="arun",classname=9,contact="8976567890")

s.save()

print(s)

**To insert multiple records**

s=student.objects.bulk\_create(

[

student(name="divya",fathername="vijay",classname=8,contact="9876509876"),

student(name="ajay",fathername="sam",classname=7,contact="8765789080")

])

s.save()

print(s)

**Delete Record in SQL**

delete from admissions\_student where id=1;

**ORM**

s=student.objects.get(id=1)

s.delete() #row will be deleted

**To delete multiple records**

s=student.objects.filter(id\_\_in=[1,3,5])

s.delete() #rows will be deleted

**Updating Records in SQL**

update admissions\_student set classname=6,contact= '8562314567' where id=1;

**ORM**

s=student.objects.get(id=1)

s.classname=6

s.contact= "8562314567"

s.save() # now 1st record will be updated

**Sorting**

**SQL using order by asc**

select \* from admissions\_student orderby classname asc;

**ORM**

s=student.objects.all().order\_by('classname')

for s1 in s:

print(s1.name)

**SQL using order by desc**

select \* from admissions\_student orderby classname desc;

ORM

s=student.objects.all().order\_by('-classname')

for s1 in s:

print(s1.name)

**Aggregate Functions**

**In SQL**

**Avg**

select avg(id) from admissions\_student;

**Sum**

select sum(id) from admissions\_student;

**Count**

select count(id) from admissions\_student;

**Min**

select min(id) from admissions\_student;

**Max**

select max(id) from admissions\_student;

**In ORM**

**Avg**

python manage.py shell

from admissions.models import student

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

s=student.objects.all().aggregate(Avg('id'))

print(s)

**Sum**

s=student.objects.all().aggregate(Sum('id'))

print(s)

**Count**

s=student.objects.all().aggregate(Count('id'))

print(s)

**Min**

s=student.objects.all().aggregate(Min('id'))

print(s)

**Max**

s=student.objects.all().aggregate(Max('id'))

print(s)