**May 02**

**How to implement AND queries:**

**AND:**all conditions should be satisfied.

**3-ways:**

1).queryset1 & queryset2

2).filter(Q(condition1) & Q(condition2))

3).filter(condition1,condition2)

**Ex:**

select all employees where ename starts with 'S' and esal < 15000.

1).emp\_list = Employee.objects.filter(ename\_\_startswith='S') & Employee.objects.filter(esal\_\_lt=15000)

2).emp\_list = Employee.objects.filter(Q(ename\_\_startswith='A') & Q(esal\_\_lt=18000))

3).emp\_list = Employee.objects.filter(ename\_\_startswith='S',esal\_\_lt=18000)

**How to implement Not queries in Django ORM:**

**all()** -->To get all records.

filter(condition)-->To get records where condition is satisfied.

**We can implement NOT queries in 2-ways:**

**1st way:** exclude(condition)--->To get records where condition is failed.

**2nd way:** filter(~Q(condition))

**Ex:**To select all employees whose name not starts with 'S'

emp\_list = Employee.objects.exclude(ename\_\_startswith='S')

emp\_list = Employee.objects.filter(~Q(ename\_\_startswith='D'))

**How to select only required columns in the query set:**

select \* from employee;

select ename,esal from employee;

**3-ways**

**1).By using values\_list():**

* **views.py**

]

emp\_list = Employee.objects.all().values\_list('ename','esal')

return render(request,'testapp/specificcolumns.html', {'emp\_list':emp\_list})

* **specificcolumns.html**

<!DOCTYPE html>

{% extends 'testapp/base.html' %}

{% block body\_block %}

<h1>Employee Information DashBoard</h1>

<table border="3">

<thead>

<th>Employee Name</th>

<th>Employee Salary</th>

</thead>

{% for emp in emp\_list %}

<tr>

<td>{{emp}}</td>

<td>{{emp}}</td>

</tr>

{% endfor %}

</table>

<br>

{% endblock %}

* **changes**

<!DOCTYPE html>

{% extends 'testapp/base.html' %}

{% block body\_block %}

<h1>Employee Information DashBoard</h1>

<table border="3">

<thead>

<th>Employee Name</th>

<th>Employee Salary</th>

</thead>

{% for emp in emp\_list %}

<tr>

{% for v in emp %}

<td>{{v}}</td>

{% endfor %}

</tr>

{% endfor %}

</table>

<br>

{% endblock %}

**2).By using values():**

emp\_list = Employee.objects.all().values('ename','esal')

* **html file**

<!DOCTYPE html>

{% extends 'testapp/base.html' %}

{% block body\_block %}

<h1>Employee Information DashBoard</h1>

<table border="3">

<thead>

<th>Employee Name</th>

<th>Employee Salary</th>

</thead>

{% for emp in emp\_list %}

<tr>

{% for k,v in emp.items %}

<td>{{v}}</td>

{% endfor %}

</tr>

{% endfor %}

</table>

<br>

{% endblock %}

**3).By using only():**

emp\_list = Employee.objects.all().only('ename','esal')

* **html file**

<!DOCTYPE html>

{% extends 'testapp/base.html' %}

{% block body\_block %}

<h1>Employee Information DashBoard</h1>

<table border="3">

<thead>

<th>Employee Name</th>

<th>Employee Salary</th>

</thead>

{% for emp in emp\_list %}

<tr>

<td>{{emp.ename}}</td>

<td>{{emp.esal}}</td>

</tr>

{% endfor %}

</table>

<br>

{% endblock %}

**Note:**

values\_list()--->QuerySet contains tuple.

values()--->QuerySet contains dict objects

only()--->QuerySet contains Employee objects

**-->**Hence values() method is recommended to use when compared with others.

**Aggregate Functions:**

Django ORM defines several functions to perform aggregate operations.

Avg(), Max(),Min(),Sum(),Count()...etc.......

* **views.py**

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

def aggregate\_view(request):

avg = Employee.objects.all().aggregate(Avg('esal'))

max = Employee.objects.all().aggregate(Max('esal'))

min = Employee.objects.all().aggregate(Min('esal'))

sum = Employee.objects.all().aggregate(Sum('esal'))

count = Employee.objects.all().aggregate(Count('esal'))

my\_dict = {'avg':avg['esal\_\_avg'], 'max':max['esal\_\_max'], 'min':min['esal\_\_min'],'sum':sum['esal\_\_sum'], 'count':count['esal\_\_count']}

return render(request,'testapp/aggregate.html',my\_dict)

* **aggregate.html**

<!DOCTYPE html>

{% extends 'testapp/base.html' %}

{% block body\_block %}

<h1>Employee Aggregate Information </h1>

<ul>

<h2><li>Average Salary:{{avg}}</li></h2>

<h2><li>Maximum Salary:{{max}}</li></h2>

<h2><li>Minimum Salary:{{min}}</li></h2>

<h2><li>Total Salary:{{sum}}</li></h2>

<h2><li>Number of Employees:{{count}}</li></h2>

</ul>

{% endblock %}

* **urls.py**

path('agg/', views.aggregate\_view),