Rakesh KR

1. Django Setup / Installation

You will need to install Django and mysqlclient if not already installed.

```
pip install django mysqlclient
```

2. Django Project Setup

To create a Django project called myproject, run:

```
django-admin startproject myproject
```

3. Django App Setup

To create the employees app inside your project:

```
cd myproject

python manage.py startapp employees
```

4. Department and Employee Models

In **employees/models.py**, define the Department and Employee models, incorporating the ForeignKey relationship between them:

```
# Create your models here.
from django.db import models

class Department(models.Model):
    name = models.CharField(max_length=255)
    def __str__(self):
        return self.name
class Employee(models.Model):
    name = models.CharField(max_length=255)
    dept = models.ForeignKey(Department, on_delete=models.CASCADE)
    job_title = models.CharField(max_length=255)
    salary = models.DecimalField(max_digits=10, decimal_places=2)
    bonus = models.DecimalField(max_digits=10, decimal_places=2, null=True, blank=True)
    def __str__(self):
        return f'{self.name} - {self.job_title}'
```

Explanation of Changes:

Department Model:

Contains only a name field (CharField) for the department's name.

Employee Model:

name: The employee's name.

dept: A foreign key pointing to the Department model. When a department is deleted, all associated employees will be deleted (on_delete=models.CASCADE).

job_title: The employee's job title.

salary: The employee's salary using a DecimalField.

bonus: An optional bonus field (null=True, blank=True).

5. Database Setup with MySQL

Configure your MySQL database in myproject/settings.py:

```
INSTALLED_APPS = [
   . . . .
    'employees',
]

DATABASES = {
'default': {
'ENGINE': 'django.db.backends.mysql',
'NAME': 'djangotest', # Your MySQL database name
'USER': 'root',
'PASSWORD': 'password',
'HOST': 'localhost',
'PORT': '3306',
}
}
```

After setting up the models and MySQL, run the migrations to create the necessary database tables:

D:\Training\Django\myproject> **python manage.py makemigrations**
D:\Training\Django\myproject> **python manage.py migrate**

6. Django Python Shell
The Django shell allows you to interact with models directly. You can access it using:

```
python manage.py shell
```

7. Employee and Department CRUD Operations
You can perform CRUD operations on the Department and Employee models using Django's ORM. **Execute the following in Shell:**

```
Create Department:

from employees.models import Department

hr_dept = Department(name="Human Resources")
hr_dept.save()

it_dept = Department(name="Information Technology")
it_dept.save()
```

Create Employee:

```
from employees.models import Employee, Department

# Assigning an employee to the HR department
hr_dept = Department.objects.get(name="Human Resources")
employee1 = Employee(name="John Doe", dept=hr_dept, job_title="HR Manager", salary=50000.00)
employee1.save()

# Assigning an employee to the IT department
it_dept = Department.objects.get(name="Information Technology")
employee2 = Employee(name="Jane Smith", dept=it_dept, job_title="Software Engineer", salary=75000.00,
bonus=5000.00)
employee2.save()
```

Read Employees and Their Departments:

```
employees = Employee.objects.all()
for employee in employees:
    print(f"Name: {employee.name}, Job Title: {employee.job_title}, Department: {employee.dept.name}")
```

Update Employee's Department:

```
employee = Employee.objects.get(name="John Doe")
it_dept = Department.objects.get(name="Information Technology")
employee.dept = it_dept
employee.save()
```

Delete Employee:

```
employee = Employee.objects.get(name="Jane Smith")
employee.delete()
```

8. Employee ORM Operations

Here are a few more ORM operations that can be performed with the new models:

Filter Employees by Department

```
it_employees = Employee.objects.filter(dept__name="Information Technology")
for employee in it_employees:
    print(employee)
```

Calculate Average Salary in a Department

```
from django.db.models import Avg

avg_salary = Employee.objects.filter(job_title="HR Manager").aggregate(Avg('salary'))
print(f"Average Salary in HR: {avg_salary['salary__avg']}")
```

Get Employees with a Salary Greater than $60,000

```
high_paid_employees = Employee.objects.filter(salary__gt=40000)

if high_paid_employees.exists():  # Check if any results were returned
    for employee in high_paid_employees:
        print(f'Employee Name: {employee.name}, Salary: {employee.salary}')
else:
    print("No employees found with salary greater than 40,000.")
```

Create Views for CRUD and ORM Operations

Open employees/views.py and add the following view functions:

```python
from django.shortcuts import render, get_object_or_404, redirect
from django.http import HttpResponse
from .models import Employee, Department
from django.db.models import Avg

# Create Department and Employee
def create_employee(request):
    hr_dept = Department.objects.get_or_create(name="Human Resources")[0]
    it_dept = Department.objects.get_or_create(name="Information Technology")[0]

    employee1 = Employee.objects.get_or_create(name="Keerthi", dept=hr_dept, job_title="CEO",
salary=50000.00)[0]
    employee2 = Employee.objects.get_or_create(name="Test", dept=it_dept, job_title="Software Engineer",
salary=75000.00, bonus=5000.00)[0]

    return HttpResponse("Employees created successfully!")

# Read Employees and Departments
def read_employees(request):
    employees = Employee.objects.all()
    employee_info = [
        f"Name: {employee.name}, Job Title: {employee.job_title}, Department: {employee.dept.name}"
        for employee in employees
    ]
    return HttpResponse("<br>".join(employee_info))
```

```
# Update Employee's Department
def update_employee_department(request, employee_name):
    employee = get_object_or_404(Employee, name=employee_name)
    it_dept = Department.objects.get(name="Information Technology")
    employee.dept = it_dept
    employee.save()
    return HttpResponse(f"{employee_name}'s department updated successfully!")

# Delete Employee
def delete_employee(request, employee_name):
    employee = get_object_or_404(Employee, name=employee_name)
    employee.delete()
    return HttpResponse(f"{employee_name} deleted successfully!")

# ORM Query: Filter Employees by Department
def filter_employees_by_department(request):
    it_employees = Employee.objects.filter(dept__name="Information Technology")
    employee_list = [employee.name for employee in it_employees]
    return HttpResponse("<br>".join(employee_list))

# ORM Query: Calculate Average Salary for Job Title
def average_salary(request):
    avg_salary = Employee.objects.filter(job_title="HR Manager").aggregate(Avg('salary'))
    return HttpResponse(f"Average Salary for HR Manager: {avg_salary['salary__avg']}")

# ORM Query: Get Employees with a Salary Greater than $40,000
def high_paid_employees(request):
    high_paid_employees = Employee.objects.filter(salary__gt=40000)
    if high_paid_employees.exists():
        employee_info = [
            f'Employee Name: {employee.name}, Salary: {employee.salary}'
            for employee in high_paid_employees
        ]
        return HttpResponse("<br>".join(employee_info))
    else:
        return HttpResponse("No employees found with salary greater than 40,000.")
```

---

3. Setup URLs for the Views

Now, we need to link these views to URLs. Create a file employees/urls.py if it doesn't already exist, and add the following:

```
from django.urls import path
from . import views

urlpatterns = [
    path('create/', views.create_employee, name='create_employee'),
    path('read/', views.read_employees, name='read_employees'),
    path('update/<str:employee_name>/', views.update_employee_department,
name='update_employee_department'),
    path('delete/<str:employee_name>/', views.delete_employee, name='delete_employee'),
    path('filter/', views.filter_employees_by_department, name='filter_employees_by_department'),
    path('average_salary/', views.average_salary, name='average_salary'),
    path('high_paid/', views.high_paid_employees, name='high_paid_employees'),
]
```

Also, link this employees/urls.py file to your main urls.py file located at myproject/urls.py:

```
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
     path('employees/', include('employees.urls')),
]
```

---

4. Running the Django Project

1. Start the Django server:

   python manage.py runserver

2. Access the URLs in your browser to trigger the operations:
   - **Create Employees**: http://127.0.0.1:8000/employees/create/
   - **Read Employees**: http://127.0.0.1:8000/employees/read/
   - **Update Employee's Department**: http://127.0.0.1:8000/employees/update/John%20Doe/
   - **Delete Employee**: http://127.0.0.1:8000/employees/delete/Jane%20Smith/
   - **Filter Employees by Department**: http://127.0.0.1:8000/employees/filter/
   - **Calculate Average Salary**: http://127.0.0.1:8000/employees/average_salary/
   - **Get High Paid Employees**: http://127.0.0.1:8000/employees/high_paid/