====================================================================================

# Django (Models) with Mysql (Relations)

ORM - Object Relation Mapping
- Django Setup / Installation
- Django Project Setup
- Django App Setup
- Django Python Shell
- Employee Model
- Employee CRUD operations
- Employee ORM operations
- Run Django Project

====================================================================================

## 1. Django Project Setup

### 1.1. Install Django and MySQL Client

First, ensure that you have `django` and `mysqlclient` installed:

```
pip install django mysqlclient
```

### 1.2. Create a Django Project

Create a Django project called `myproject`:

```
django-admin startproject myproject
cd myproject
```

### 1.3. Create a Django App

Inside the project, create an app called `employees`:

```
python manage.py startapp employees
```

## 2. Database Setup (MySQL)

In the `myproject/settings.py` file, configure the database settings for MySQL:

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

3. Employee Model (Python OOP Principles for Django Models)
In the `employees/models.py` file, define the `Employee` model:

```
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}'
```

4. Database Migrations
After defining the models, apply migrations to update the database schema:

4.1. Make Migrations

```
python manage.py makemigrations
```

4.2. Migrate

```
python manage.py migrate
```

5. Django Shell for CRUD Operations
To perform CRUD operations in Django using the shell:
5.1. Enter the Django Shell

```
python manage.py shell
```

5.2. Import the Models

```
from employees.models import Employee, Department
```

5.3. CRUD Operations Using the Shell

5.3.1. Create a new Department and Employee:

```
it_dept = Department.objects.create(name='IT Department')
employee = Employee.objects.create(name='John Doe', dept=it_dept, job_title='Software Engineer', salary=70000, bonus=5000)
```

5.3.2. Read all Employees:

```
employees = Employee.objects.all()
for emp in employees:
    print(emp.name, emp.job_title)
```

5.3.3. Read an Employee by ID:

```
emp = Employee.objects.get(id=1)
print(emp.name, emp.job_title)
```

5.3.4. Update an Employee's salary:

```
emp = Employee.objects.get(id=1)
emp.salary = 80000
emp.save()
```

5.3.5.

- Delete an Employee:

```
emp = Employee.objects.get(id=1)
emp.delete()
```

6. ORM Operations from Python and Django

6.1. Filtering employees by department:

```
it_employees = Employee.objects.filter(dept__name='IT Department')
for emp in it_employees:
    print(emp.name)
```

6.2. Sorting employees by salary:

```
sorted_employees = Employee.objects.all().order_by('salary')
for emp in sorted_employees:
    print(emp.name, emp.salary)
```

6.3. Aggregating salaries:

```
from django.db.models import Avg
average_salary = Employee.objects.all().aggregate(Avg('salary'))
print(average_salary)
```

6.4. Joining with Department (similar to SQL JOIN):

```
employees_with_depts = Employee.objects.select_related('dept').all()
for emp in employees_with_depts:
    print(emp.name, emp.dept.name)
```

7. Running the Django Development Server
After setting up the models and performing migrations,
start the Django development server
to view the project in a browser:

```
python manage.py runserver
```

Visit `http://127.0.0.1:8000/` to access the Django app.

8. Summary
- Django Project Setup:
  You created a project and app.
- Database Setup:
  Configured MySQL as the backend.
- CRUD Operations:
  Performed create, read, update, and delete operations on the `Employee` model.
- Django Shell:
  Used Django ORM through the shell to manipulate data.
- ORM Operations:
  Filtered, sorted, and joined data from the `Employee` and `Department` models.