




## Complete Guide to Django, MongoDB, and Your Project

### ♦ What is Django?

Django is a **Python web framework** that helps build web applications. It allows handling **databases**, creating **APIs**, and managing **data easily**. Django follows **MVT (Model-View-Template)** architecture:

- **Model**  → Defines database structure.
- **View**  → Handles user requests and returns data.
- **Template**  → Displays data (not needed for your assignment).

For your project, you'll mostly work with:

- **Models (MongoDB structure)**
- **Views (APIs to fetch and modify data)**
- **Serializers (to format data for APIs)**
- **Django Admin (to manage data easily)**

---

## Folder & File Structure in Your Project

Your Django project contains **two main parts**:

1. **data\_platform/ (Django Project)**
2. **analytics/ (Django App)**

### data\_platform/ (Django Project)

This is the **main Django project folder**.

File	Purpose
<code>__init__.py</code>	Marks this as a <b>Python package</b> .
<code>settings.py</code>	<b>Main configuration file</b> (database, apps, security settings, etc.).
<code>urls.py</code>	<b>Manages URL routing</b> (connects APIs and pages).
<code>wsgi.py</code>	Helps <b>deploy Django on web servers</b> .
<code>asgi.py</code>	Alternative to <code>wsgi.py</code> (for async applications).

### 📁 analytics/ (Django App)

This is where **your main code lives**.

Django apps are like “modules” that **handle different features**.

This is where **your main code lives**.

Django apps are like “modules” that **handle different features**.

File	Purpose
<code>__init__.py</code>	Marks this as a <b>Python package</b> .
<code>admin.py</code>	Registers models for the <b>Django Admin Panel</b> .
<code>apps.py</code>	Basic <b>configurations for the app</b> .
<code>models.py</code>	Defines <b>database structure</b> (MongoDB models).
<code>views.py</code>	Contains <b>functions to process API requests</b> .
<code>tests.py</code>	Used for <b>testing the application</b> (not needed for now).
<code>migrations/</code>	Stores <b>database changes</b> (not needed for MongoDB).

---

## ♦ How Will You Use Django & MongoDB for Your Assignment?

### Step 1: Store Data in MongoDB

- The CSV files (amazon\_sale\_report.csv, etc.) contain **e-commerce sales data**.

- **Goal:** Import this data into MongoDB so Django can access it.

---

## Step 2: Define a MongoDB Model

Edit analytics/models.py and define a model:

```
python Copy

from django import models

class AmazonSaleReport(models.Model):
    order_id = models.CharField(max_length=100)
    date = models.DateField()
    sales_channel = models.CharField(max_length=100)
    sku = models.CharField(max_length=100)
    category = models.CharField(max_length=100)
    amount = models.FloatField()

    objects = models.DjangoManager()

    def __str__(self):
        return f"Order {self.order_id} - {self.category}"
```

💡 This creates a table in MongoDB!

---

## Step 3: Connect Django to MongoDB

Modify data\_platform/settings.py:

```
python Copy

DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'data_analysis_db',
        'CLIENT': {
            'host': 'mongodb://127.0.0.1:27017/',
        }
    }
}
```

✅ **Start MongoDB on macOS:**

```
bash
```

Copy

```
brew services start mongodb-community@7.0
```

Check MongoDB connection:

```
bash
```

Copy

```
mongo --eval "db.runCommand({ connectionStatus: 1 })"
```

If it prints { "ok" : 1 }, MongoDB is ready. 🚀

**Step 4: Create an API to Fetch Data**

Edit analytics/views.py:

python

 Copy

```
from rest_framework import viewsets
from .models import AmazonSaleReport
from .serializers import AmazonSaleReportSerializer

class AmazonSaleReportViewSet(viewsets.ModelViewSet):
    queryset = AmazonSaleReport.objects.all()
    serializer_class = AmazonSaleReportSerializer
```

Create analytics/serializers.py:

python

 Copy

```
from rest_framework import serializers
from .models import AmazonSaleReport

class AmazonSaleReportSerializer(serializers.ModelSerializer):
    class Meta:
        model = AmazonSaleReport
        fields = '__all__'
```

Define API URLs in analytics/urls.py:

python

 Copy

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import AmazonSaleReportViewSet

router = DefaultRouter()
router.register(r'amazon_sales', AmazonSaleReportViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

Connect APIs in data\_platform/urls.py:

python

 Copy

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

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

---

## Step 5: Run the Server and Test the API

Start Django:

```
bash
```

 Copy

```
python manage.py runserver
```

✓ Open your browser and go to:

👉 [http://127.0.0.1:8000/api/amazon\\_sales/](http://127.0.0.1:8000/api/amazon_sales/)

You should see JSON data from MongoDB!

## ◆ Summary: How Everything Works Together

Component	Role in Your Project
Django Project (data_platform)	Main project, settings, URLs.
Django App (analytics)	Handles database and APIs.
MongoDB	Stores sales data.
Models (models.py)	Defines data structure in MongoDB.
Views (views.py)	Creates API endpoints.
Serializers (serializers.py)	Formats data for API.
Admin (admin.py)	Manages data via Django admin.

## 🎯 Next Steps

- 1 Complete all the above steps.
- 2 Verify MongoDB is connected (`mongo --eval "db.runCommand({ connectionStatus: 1 })"`).
- 3 Run Django server and test API (`http://127.0.0.1:8000/api/amazon_sales/`).
- 4 Import CSV data into MongoDB (I'll guide you on this next!).