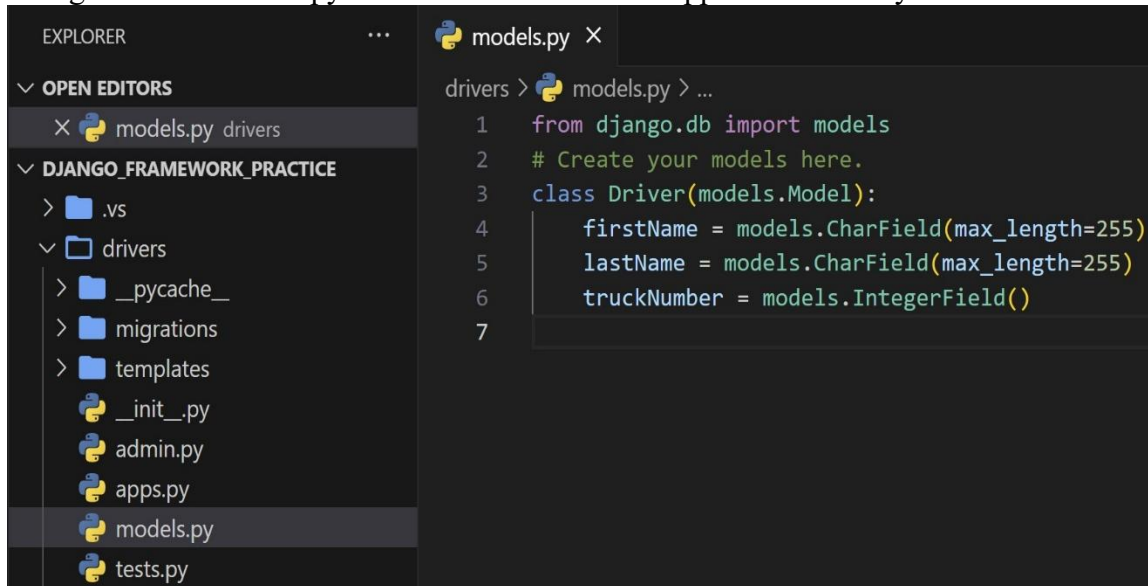


## Chapter 3 CRUD Operations

Previously, we developed a web application with a one tier architecture that render static content from our local computer. We are able to work with Django without modifying or uploading files. Django transforms the data into objects and these objects are tables in a database.

### 1. Create Model

- We use a model to retrieve data. We use a template to display the data.
- Navigate to the models.py file located in our driver app folder. Modify the code to below.

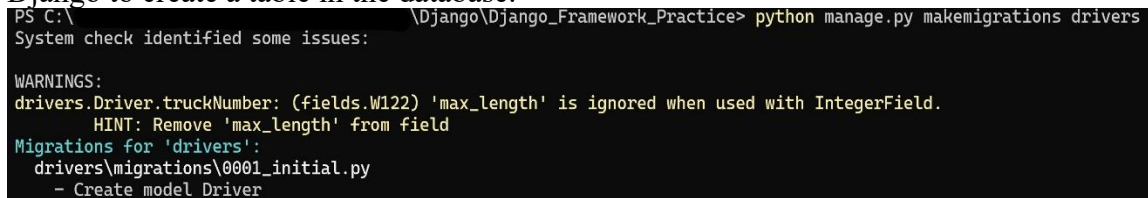


The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane displays the project structure under 'DJANGO\_FRAMEWORK\_PRACTICE'. The 'drivers' folder is expanded, showing files like \_\_init\_\_.py, admin.py, apps.py, models.py, and tests.py. The 'models.py' file is selected and open in the editor. The code in models.py is as follows:

```
1 from django.db import models
2 # Create your models here.
3 class Driver(models.Model):
4     firstName = models.CharField(max_length=255)
5     lastName = models.CharField(max_length=255)
6     truckNumber = models.IntegerField()
7
```

### 2. Create Table

- At this point we created a model in our driver app. We have to perform an operation for Django to create a table in the database.



The screenshot shows a terminal window with the following output:

```
PS C:\Django\Django_Framework_Practice> python manage.py makemigrations drivers
System check identified some issues:

WARNINGS:
drivers.Driver.truckNumber: (fields.W122) 'max_length' is ignored when used with IntegerField.
HINT: Remove 'max_length' from field
Migrations for 'drivers':
  drivers\migrations\0001_initial.py
    - Create model Driver
```

- The changes to the are stored in the top-level migrations folder.

```

1 # Generated by Django 3.2.6 on 2024-10-25 16:11
2
3 from django.db import migrations, models
4
5
6 class Migration(migrations.Migration):
7
8     initial = True
9
10    dependencies = [
11    ]
12
13    operations = [
14        migrations.CreateModel(
15            name='Driver',
16            fields=[
17                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
18                ('firstName', models.CharField(max_length=255)),
19                ('lastName', models.CharField(max_length=255)),
20                ('truckNumber', models.IntegerField(max_length=5)),
21            ],
22        ),
23    ]
24

```

- d.
  - i. Note that Django added an 'id' field as a primary key. The variable is autoincremented with each entry. However, the table has not been created in SQL.

- e. Run: `python manage.py migrate`

```

PS C:\Django\Django_Framework_Practice> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, drivers, sessions
Running migrations:
  Applying drivers.0001_initial... OK

```

- f.
- g. We can see the SQL statement execute from the migration. Execute the below statement.

- h. Run: `python manage.py sqlmigrate drivers 0001`

```

PS C:\Django\Django_Framework_Practice> python manage.py sqlmigrate drivers 0001
BEGIN;
--
-- Create model Driver
--
CREATE TABLE "drivers_driver" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "firstName" varchar(255) NOT NULL, "last
Name" varchar(255) NOT NULL, "truckNumber" integer NOT NULL);
COMMIT;

```

- i.
3. Create Document
  - a. The current table/database is empty. We need to use the python shell to make a new entry in the database.

- b. Run: `python manage.py shell`

```

PS C:\Django\Django_Framework_Practice> python manage.py shell
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr 6 2021, 13:40:21) [MSC v.1928 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.26.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from drivers.models import Driver

In [2]: Driver.objects.all()
Out[2]: <QuerySet []>

In [3]: driver = Driver(firstName='Maurice', lastName='Wesley', truckNumber=44503)

In [4]: driver.save()

In [5]: Driver.objects.all().values()
Out[5]: <QuerySet [{'id': 1, 'firstName': 'Maurice', 'lastName': 'Wesley', 'truckNumber': 44503}]>

```

- c.
  - i. Line 1: We have to import the model of our table schema from our driver application.
  - ii. Line 2: Test execution statement to check that we indeed do have an empty database
  - iii. Line 3: Our first entry. Declare a variable assigned to an object instance of our model. We passed the attributes and values.

- iv. Line 4: The object instance has a 'save' member method.
- v. Line 5: Statement executed to list of the records in the database.

#### 4. Creating Multiple Records

```
In [6]: driver1 = Driver(firstName='John', lastName='Stamos', truckNumber=44601)
In [7]: driver2 = Driver(firstName='Crystal', lastName='Livermor', truckNumber=97845)
In [8]: driver3 = Driver(firstName='Andrew', lastName='Mason', truckNumber=45314)
In [9]: driver4 = Driver(firstName='Pete', lastName='Hach', truckNumber=78415)
In [10]: allDrivers = [driver1, driver2, driver3, driver4]
In [11]: for x in allDrivers:
...:     x.save()
...:
In [12]: |
```

- a.
- b. Now that we have entered the new records, we can use a for loop to retrieve the entries.

```
In [13]: for x in Driver.objects.all().values():
...:     print(x)
...:
{'id': 1, 'firstName': 'Maurice', 'lastName': 'Wesley', 'truckNumber': 44503}
{'id': 2, 'firstName': 'John', 'lastName': 'Stamos', 'truckNumber': 44601}
{'id': 3, 'firstName': 'Crystal', 'lastName': 'Livermor', 'truckNumber': 97845}
{'id': 4, 'firstName': 'Andrew', 'lastName': 'Mason', 'truckNumber': 45314}
{'id': 5, 'firstName': 'Pete', 'lastName': 'Hach', 'truckNumber': 78415}
In [14]: |
```

#### 5. Update Record

```
In [18]: x = Driver.objects.all()[2]
In [19]: x.firstName
Out[19]: 'Crystal'
In [20]: |
```

- a.
- i. Line 18: We can access a single record and assign it to a variable
- ii. Line 19: Not that we have an object instance, we can access an attribute
- b. The object attribute can be access and mutated. Assign the attribute to new value.

```
In [19]: x.firstName
Out[19]: 'Crystal'
In [20]: x.firstName = 'Carol'
In [21]: x.save()
In [22]: print(x)
Driver object (3)
In [23]: x.firstName
Out[23]: 'Carol'
```

- c.
- i. Note the print statement on line 22. We are able to print out the index/id of the object instance.

- d. Now we can output the outdated records.

```
In [24]: for x in Driver.objects.all().values():
...:     print(x)
...:
{'id': 1, 'firstName': 'Maurice', 'lastName': 'Wesley', 'truckNumber': 44503}
{'id': 2, 'firstName': 'John', 'lastName': 'Stamos', 'truckNumber': 44601}
{'id': 3, 'firstName': 'Carol', 'lastName': 'Livermor', 'truckNumber': 97845}
{'id': 4, 'firstName': 'Andrew', 'lastName': 'Mason', 'truckNumber': 45314}
{'id': 5, 'firstName': 'Pete', 'lastName': 'Hach', 'truckNumber': 78415}

In [25]: |
```

- e. i. Reflected above, the third record ('id': 3) has been updated.

## 6. Delete Record

- a. We recently have a driver quit and cleaned out their truck. We need to remove them from our drivers list. We can use a loop, do a check, and delete the record. We will output the new list. Please see below.

```
In [25]: for x in Driver.objects.all():
...:     if x.truckNumber == 78415:
...:         x.delete()
...:

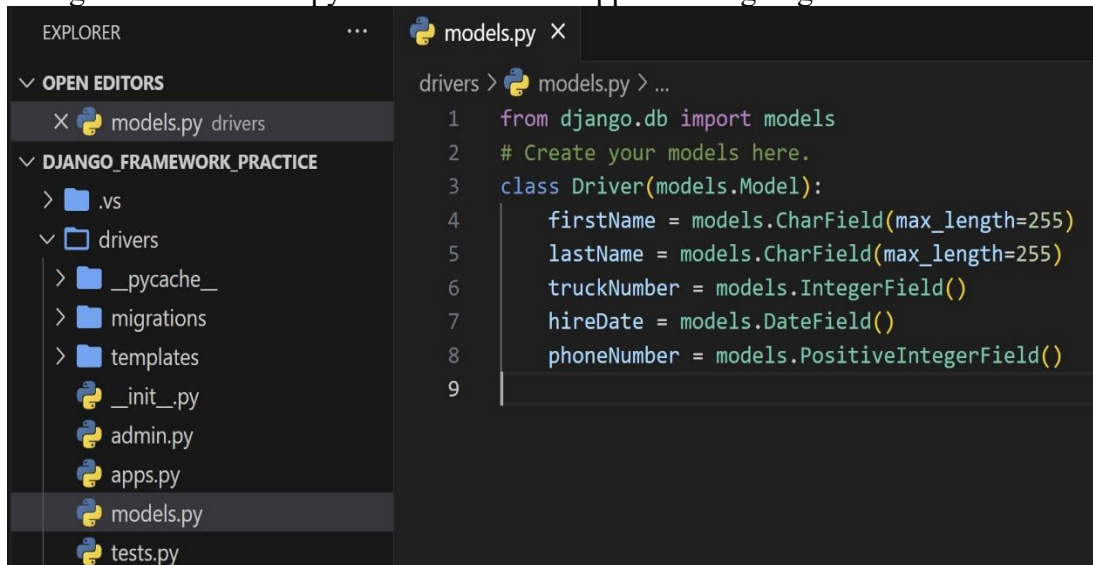
In [26]: for x in Driver.objects.all().values():
...:     print(x)
...:
{'id': 1, 'firstName': 'Maurice', 'lastName': 'Wesley', 'truckNumber': 44503}
{'id': 2, 'firstName': 'John', 'lastName': 'Stamos', 'truckNumber': 44601}
{'id': 3, 'firstName': 'Carol', 'lastName': 'Livermor', 'truckNumber': 97845}
{'id': 4, 'firstName': 'Andrew', 'lastName': 'Mason', 'truckNumber': 45314}

In [27]: |
```

- b. i. We successfully deleted the record from the database.

## 7. Update Model

- a. Navigate to the models.py file in our drivers app. We are going to add two new fields.



- b.
- c. Similar to earlier, we will have to run a migration to update our Driver app model.



```
PS C:\Django\Django_Framework_Practice> python manage.py makemigrations drivers
You are trying to add a non-nullable field 'hireDate' to driver without a default; we can't do that (the database needs something to populate existing rows).
Please select a fix:
1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
2) Quit, and let me add a default in models.py
Select an option:
```

- d. i. When we add the new field, we did not pass a min, max, or default value. Select option 2 and navigate back to the models.py file. Make the following adjustments.

```
EXPLORER
...
models.py X
drivers > models.py > ...
1 from django.db import models
2 # Create your models here.
3 class Driver(models.Model):
4     firstName = models.CharField(max_length=255)
5     lastName = models.CharField(max_length=255)
6     truckNumber = models.IntegerField()
7     hireDate = models.DateField(null=True)
8     phoneNumber = models.PositiveIntegerField(null=True)
9
```

- ii. e. Run: python manage.py makemigrations drivers

```
Select an option: 2
PS C:\Django\Django_Framework_Practice> python manage.py makemigrations drivers
Migrations for 'drivers':
  drivers\migrations\0002_auto_20241025_1434.py
    - Add field hireDate to driver
    - Add field phoneNumber to driver
    - Alter field truckNumber on driver
```

- f. g. Run: python manage.py migrate

```
PS C:\Django\Django_Framework_Practice> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, drivers, sessions
Running migrations:
  Applying drivers.0002_auto_20241025_1434... OK
PS C:\Django\Django_Framework_Practice>
```

- h. 8. Update Entry

- a. Import and access a record in the database. Then update the values for the new fields.

```
In [1]: from drivers.models import Driver

In [2]: x = Driver.objects.all()[0]

In [3]: x.firstName
Out[3]: 'Maurice'

In [4]: x.phoneNumber = 8008887755

In [5]: x.hireDate = '2025-10-25'

In [6]: x.save()

In [7]: for x in Driver.objects.all().values():
...:     print(x)
...:
{'id': 1, 'firstName': 'Maurice', 'lastName': 'Wesley', 'truckNumber': 44503, 'hireDate': datetime.date(2025, 10, 25), 'phoneNumber': 8008887755}
{'id': 2, 'firstName': 'John', 'lastName': 'Stamos', 'truckNumber': 44601, 'hireDate': None, 'phoneNumber': None}
{'id': 3, 'firstName': 'Carol', 'lastName': 'Livermor', 'truckNumber': 97845, 'hireDate': None, 'phoneNumber': None}
{'id': 4, 'firstName': 'Andrew', 'lastName': 'Mason', 'truckNumber': 45314, 'hireDate': None, 'phoneNumber': None}
```

- b. i. Success!!!