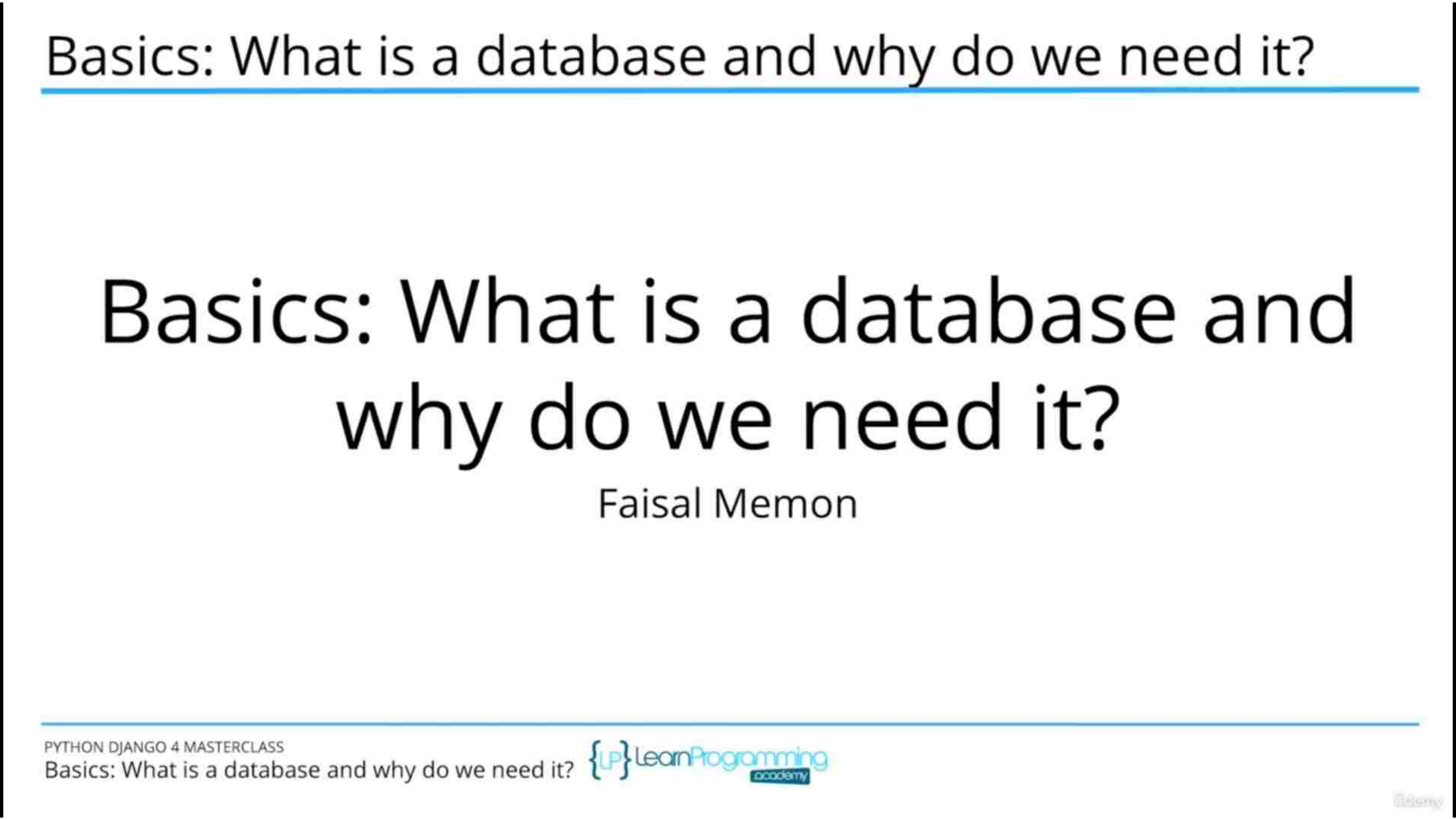


Basics: What is a database and why do we need it?

Basics: What is a database and why do we need it?

Faisal Memon

The Problem

- How do we manage things if our application scales?
- Any modern website you visit, has persistent and dynamic information that is being shown.

Databases

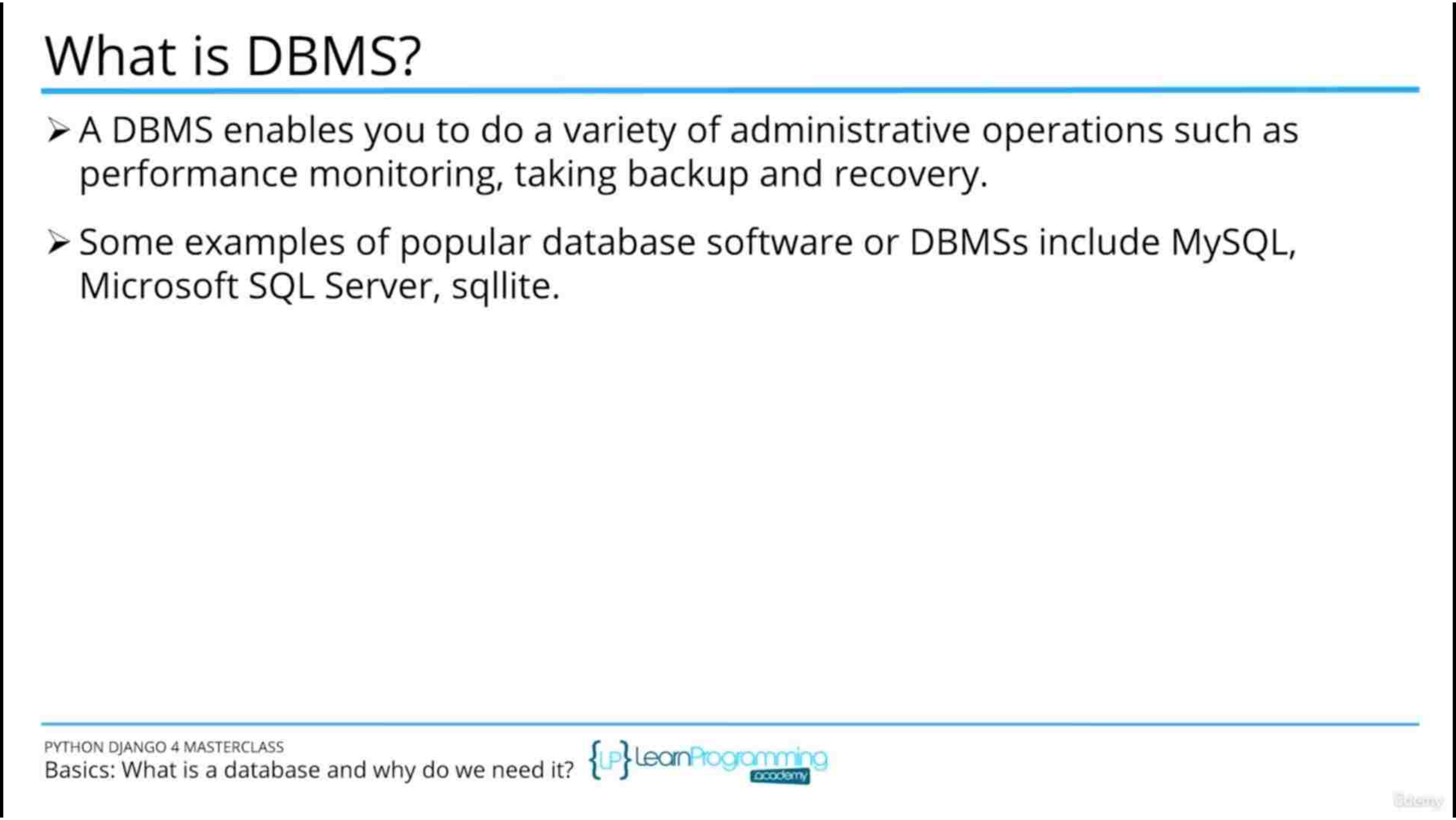
- Database is a place where data related to your users and product is stored.
- Databases are typically used to store data and maintain state of your application.

Databases in action

- Facebook example: Modern websites store your information as long as you delete it.
- On Facebook, your data is shown to others if you allow.
- Databases are used for mobile applications as well as websites.

How would you access data?

- Data stored in database can be accessed using a language called SQL.
- SQL stands for structured query language which is typically written by data analysts and programmers.



What is DBMS?

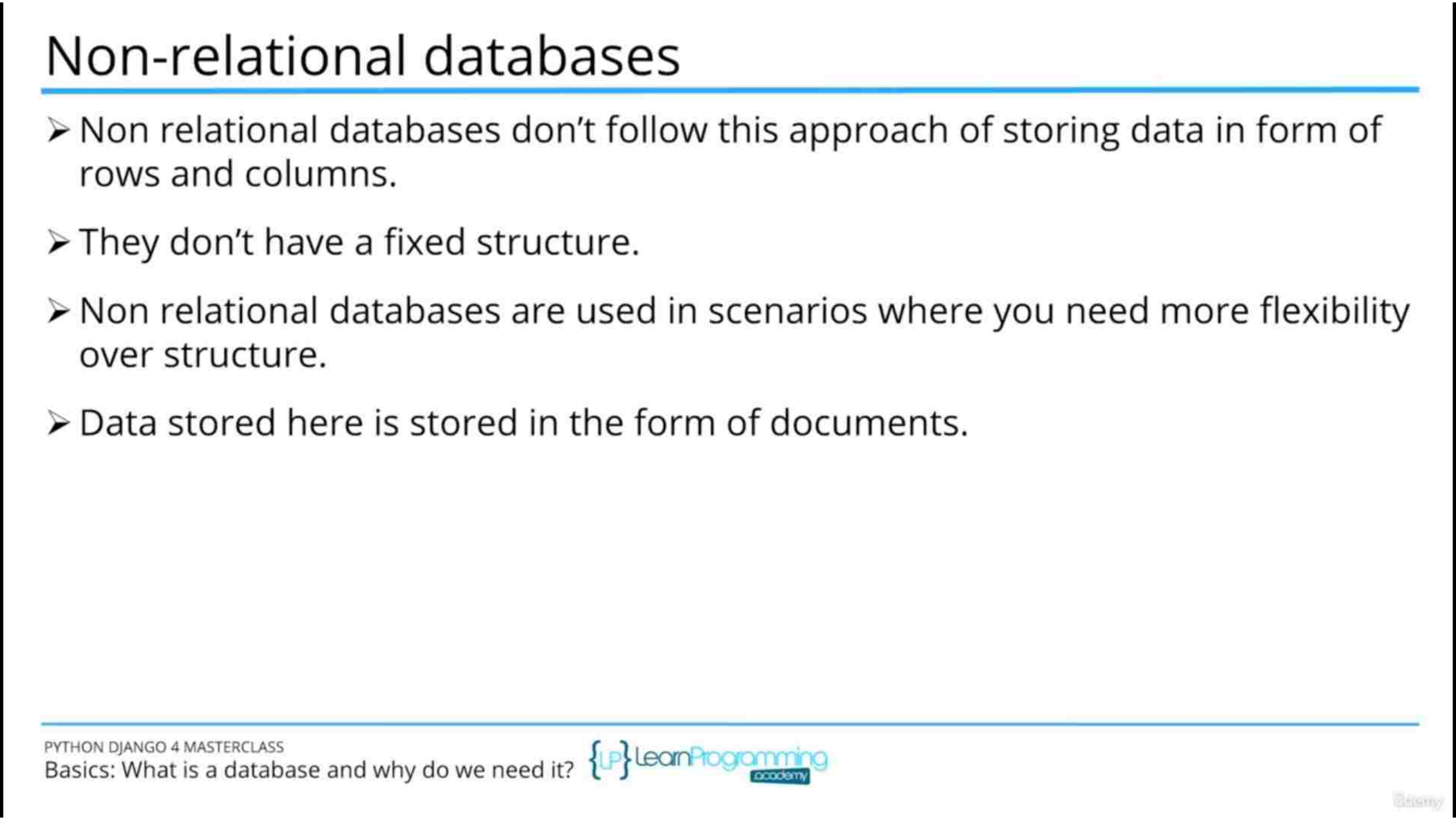
- A DBMS enables you to do a variety of administrative operations such as performance monitoring, taking backup and recovery.
- Some examples of popular database software or DBMSs include MySQL, Microsoft SQL Server, sqllite.

Database types

- Relational
- Non relational

Relational databases

- Relational databases have structure similar to that of a excel spreadsheet.
- If you have used excel, you would know you can store / structure data there in rows and column.
- Structure of the database is known as schema and one row of information is known as a record.
- Some of the popular example of relational databases are: MySQL, Microsoft Access, Microsoft SQL Server, sqlite, FileMaker Pro, Oracle Database.



Non-relational databases

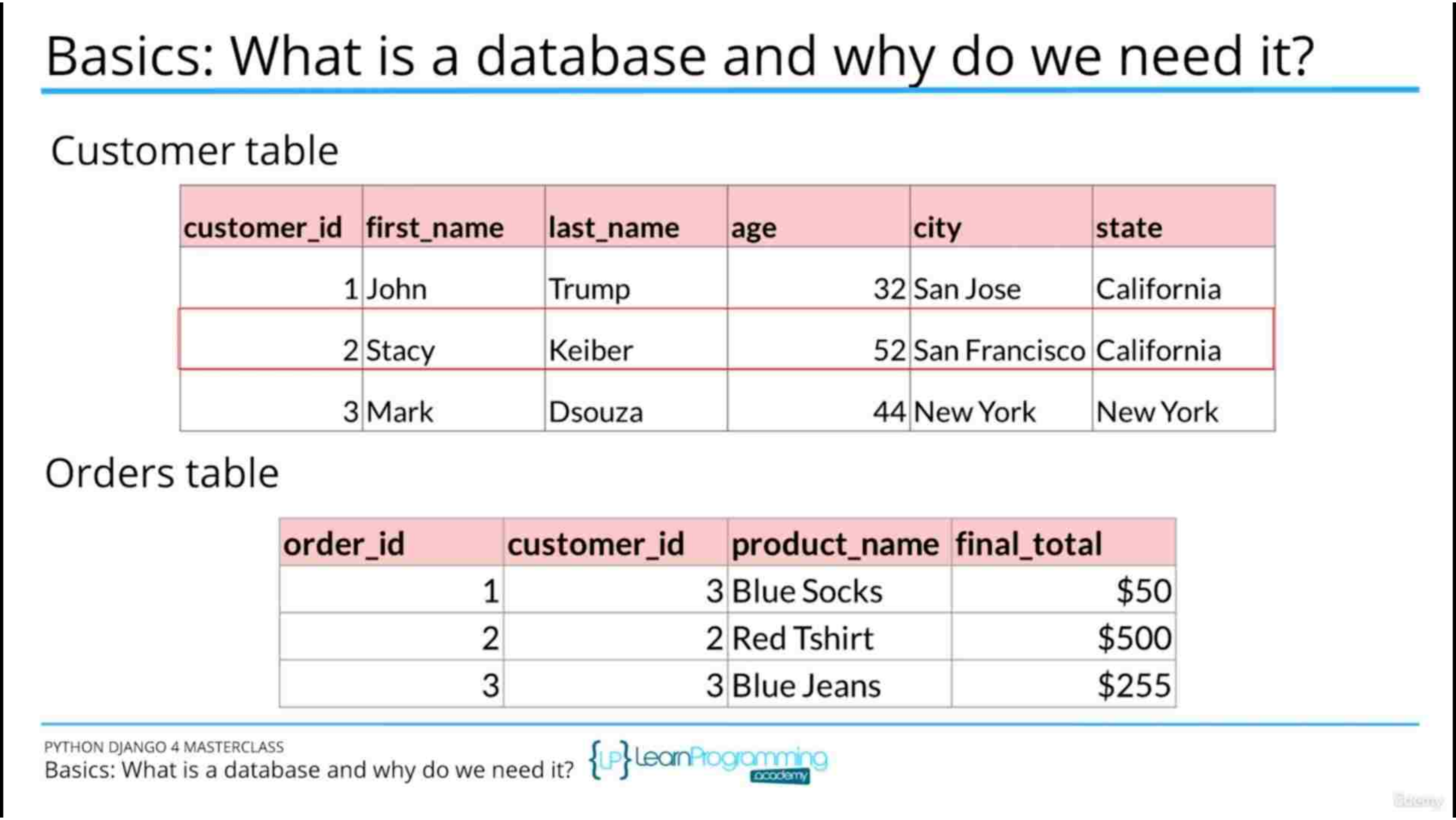
- Non relational databases don’t follow this approach of storing data in form of rows and columns.
- They don’t have a fixed structure.
- Non relational databases are used in scenarios where you need more flexibility over structure.
- Data stored here is stored in the form of documents.

Example: Store application

customer_id	first_name	last_name	age	city	state
1	John	Trump	32	San Jose	California
2	Stacy	Keiber	52	San Francisco	California
3	Mark	Dsouza	44	New York	New York

Customer table with orders

customer_id	first_name	last_name	age	city	state	product_name	final_total
1	John	Trump	32	San Jose	California		
2	Stacy	Keiber	52	San Francisco	California	Red Tshirt	$500
3	Mark	Dsouza	44	New York	New York	Blue Socks	$50
3	Mark	Dsouza	44	New York	New York	Blue Jeans	$255



Basics: What is a database and why do we need it?

Customer table

customer_id	first_name	last_name	age	city	state
1	John	Trump	32	San Jose	California
2	Stacy	Keiber	52	San Francisco	California
3	Mark	Dsouza	44	New York	New York

Orders table

order_id	customer_id	product_name	final_total
1	3	Blue Socks	$50
2	2	Red Tshirt	$500
3	3	Blue Jeans	$255

Relationships and Normalization

- Linking between customer and orders table is known as relationship between 2 tables.
- The process of eliminating redundant data from your database is known as Normalization.

Accessing and manipulating data

- You can access the data that is stored in the database using a language or a standard called SQL.
- SQL stands for structured query language.
- If you're working with relational databases, it's very important that you know SQL so that you can manipulate and access the data.
- SQL has commands like “Select”, “Insert”, “Update”, “Delete”, “Create”, and “Drop” which enables you to work with relational databases.

What is SQLite

- SQLite is a lightweight relational database.
- It is self contained and has no dependencies. Self contained meaning it does not require any external library for operating.
- It is a file based database where in all the data is stored in a files on disk.

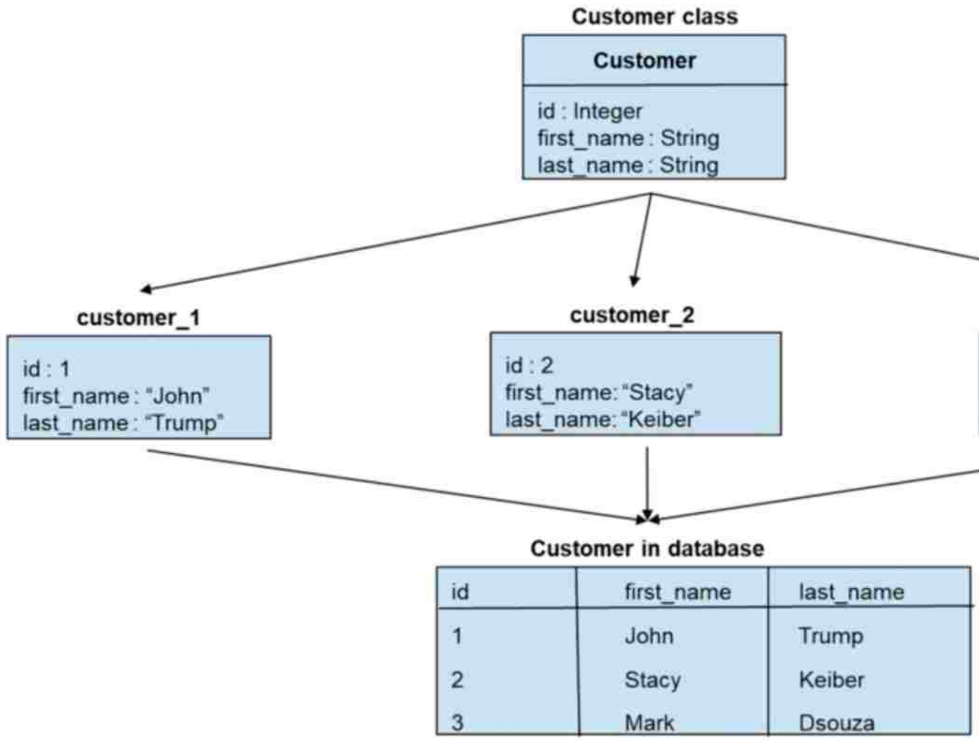
Building applications with Django

- Every application built in Python has entities which is represented in the form of classes.
- You have a class which represents that entity in real world and you have corresponding table which represents this entity.
- Every class would have attribute which would represent real world attributes like customer id, customer name, and so on.

Building applications with Django

- Class attributes are represented using columns in the table corresponding to that particular entity.
- You have Customer class, its database representation, but how do you represent multiple customers in the table.
- You can create an object of customer class with different values assigned to different attributes and then these objects become rows in tables within the database.

What is ORM?



id	first_name	last_name
1	John	Trump
2	Stacy	Keiber
3	Mark	Dsouza

ORM

- Whenever there is a class, that class can be automatically converted to a table with its attributes being converted to columns.
- So now the developer does not have to write queries for table creation, it's created automatically.
- Whenever an object is created, its data can be saved in the database as row in table, this is automatically handled by ORM.

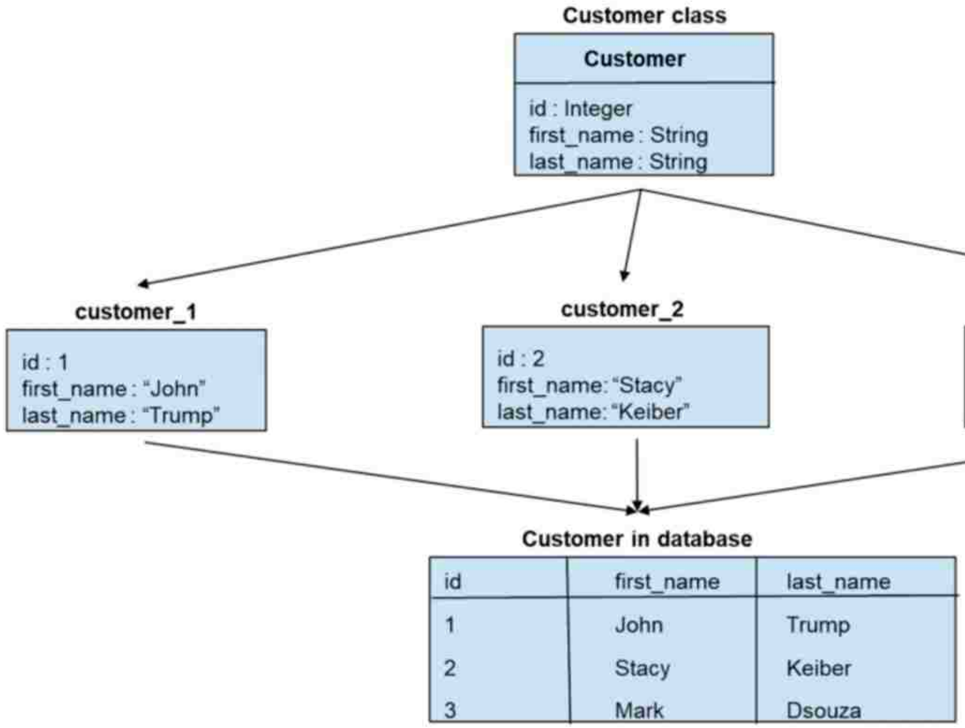
ORM

- ORM as a concept makes developers lives easier and lets developers focus on application logic rather than SQL queries.
- Because of ORM developers don't need to learn how to write SQL queries since the translation from application to SQL is handled by ORM itself.
- It's a powerful technique in programming which also minimizes mistakes since developers are not writing queries on their own.

Django Models

- A model in Django is nothing but a class which can be saved to the database.
- Every model represents a table in a database with its properties being converted to columns.

Django Models, Field, Field types, Field options



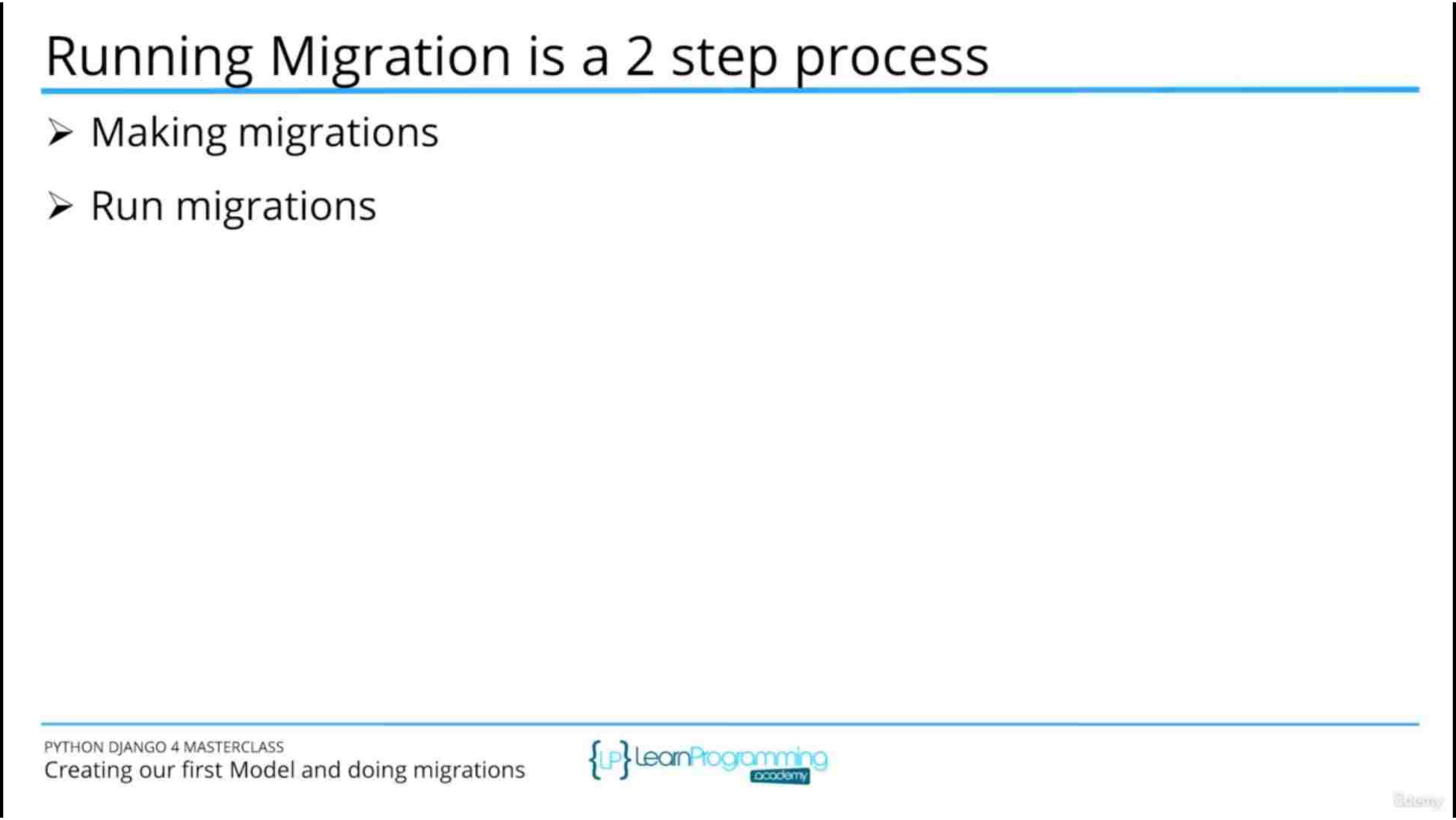
id	first_name	last_name
1	John	Trump
2	Stacy	Keiber
3	Mark	Dsouza

Defining Django Models

```
from django.db import models

class Customer(models.Model):
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
```

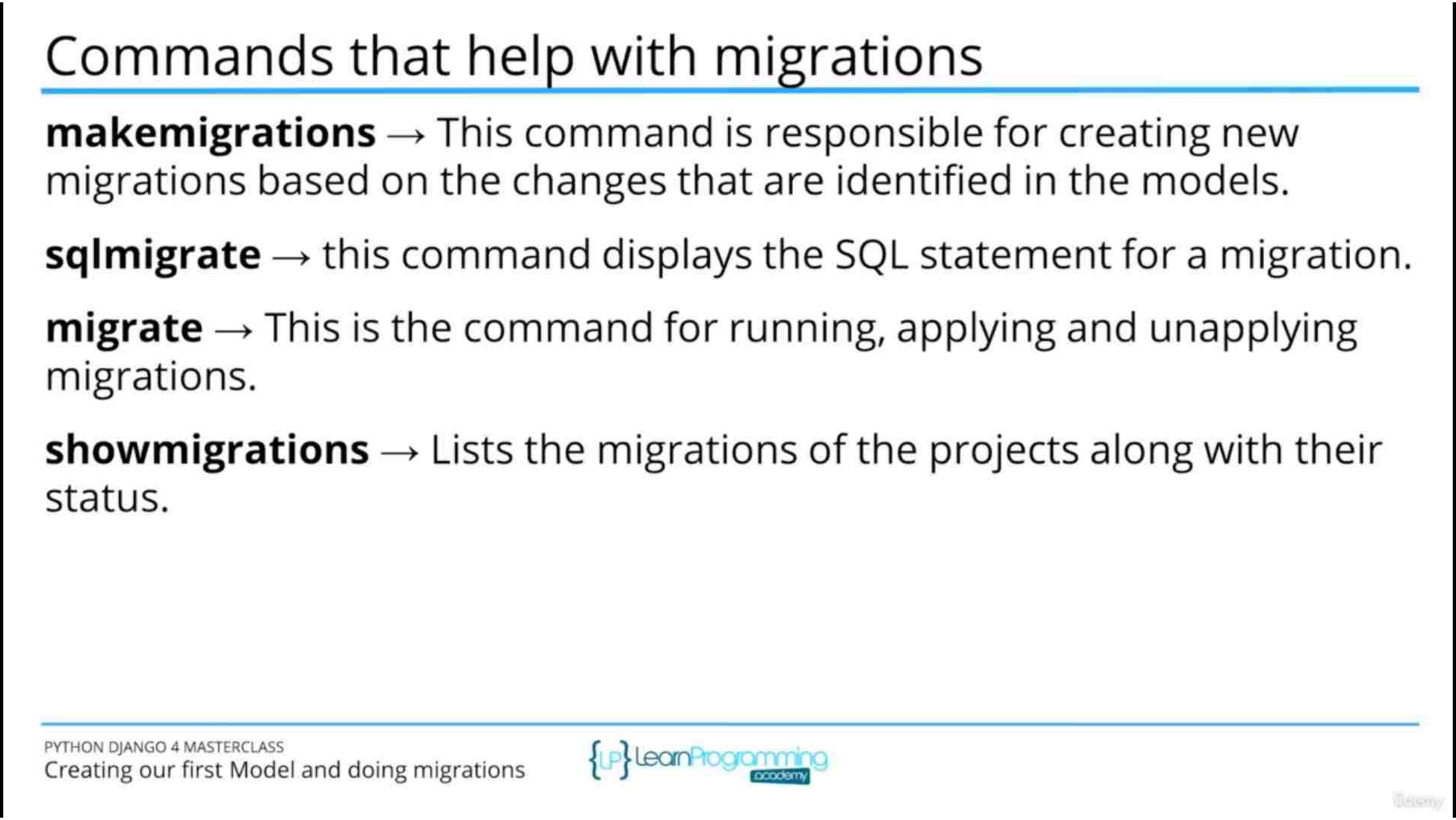
Things to note

- Every model defined in Django is a subclass of django.db.models.Model.
- Every attribute defined in model is converted to the column. These attribute of the entities are known as fields within the models. So fields are the ones that are converted to columns.
- Every field stores a particular type of data which is defined by field types.
- Every field define might also have certain restrictions like the length in this case.



Running Migration is a 2 step process

- Making migrations
- Run migrations



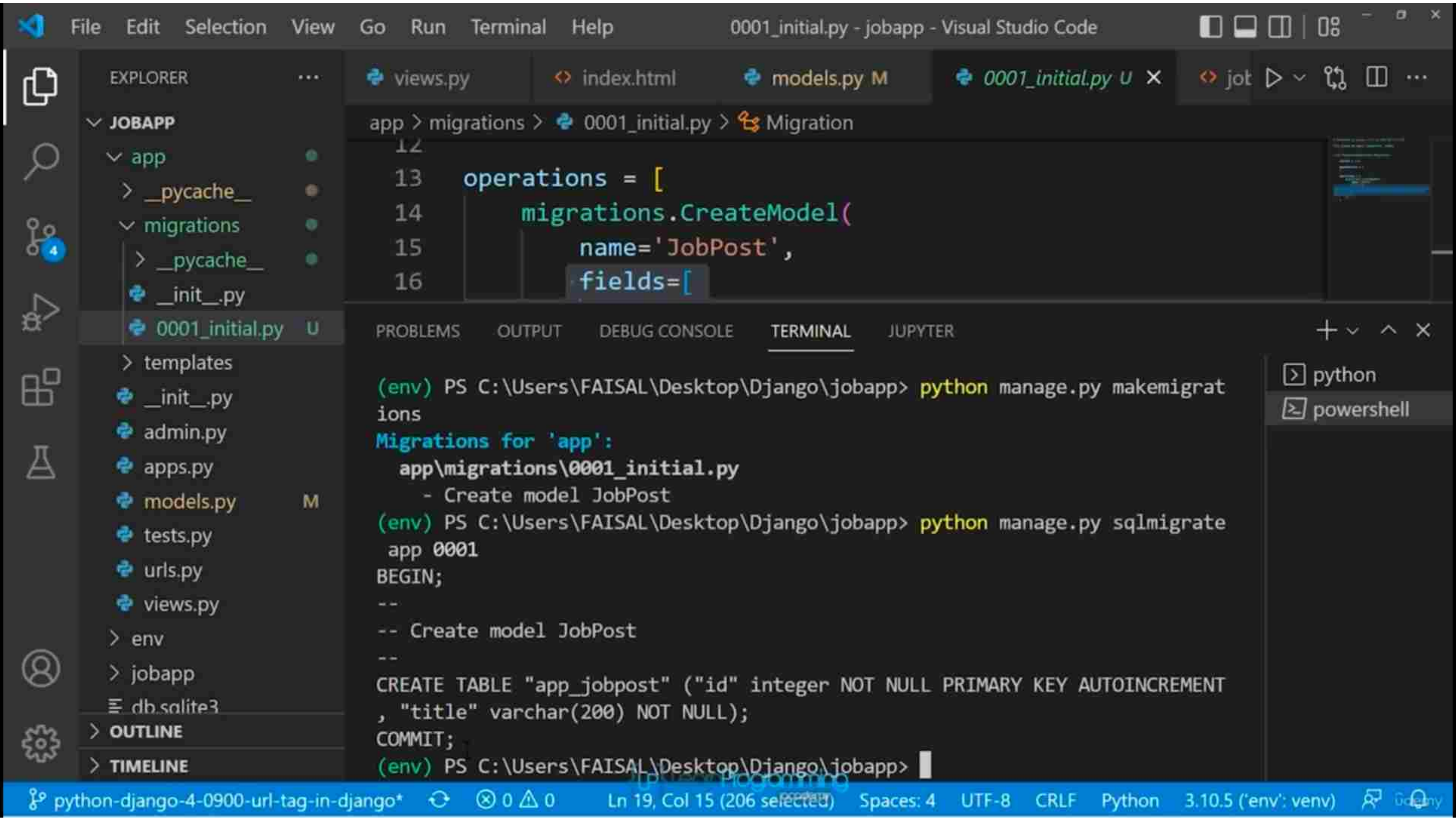
Commands that help with migrations

makemigrations → This command is responsible for creating new migrations based on the changes that are identified in the models.

sqlmigrate → this command displays the SQL statement for a migration.

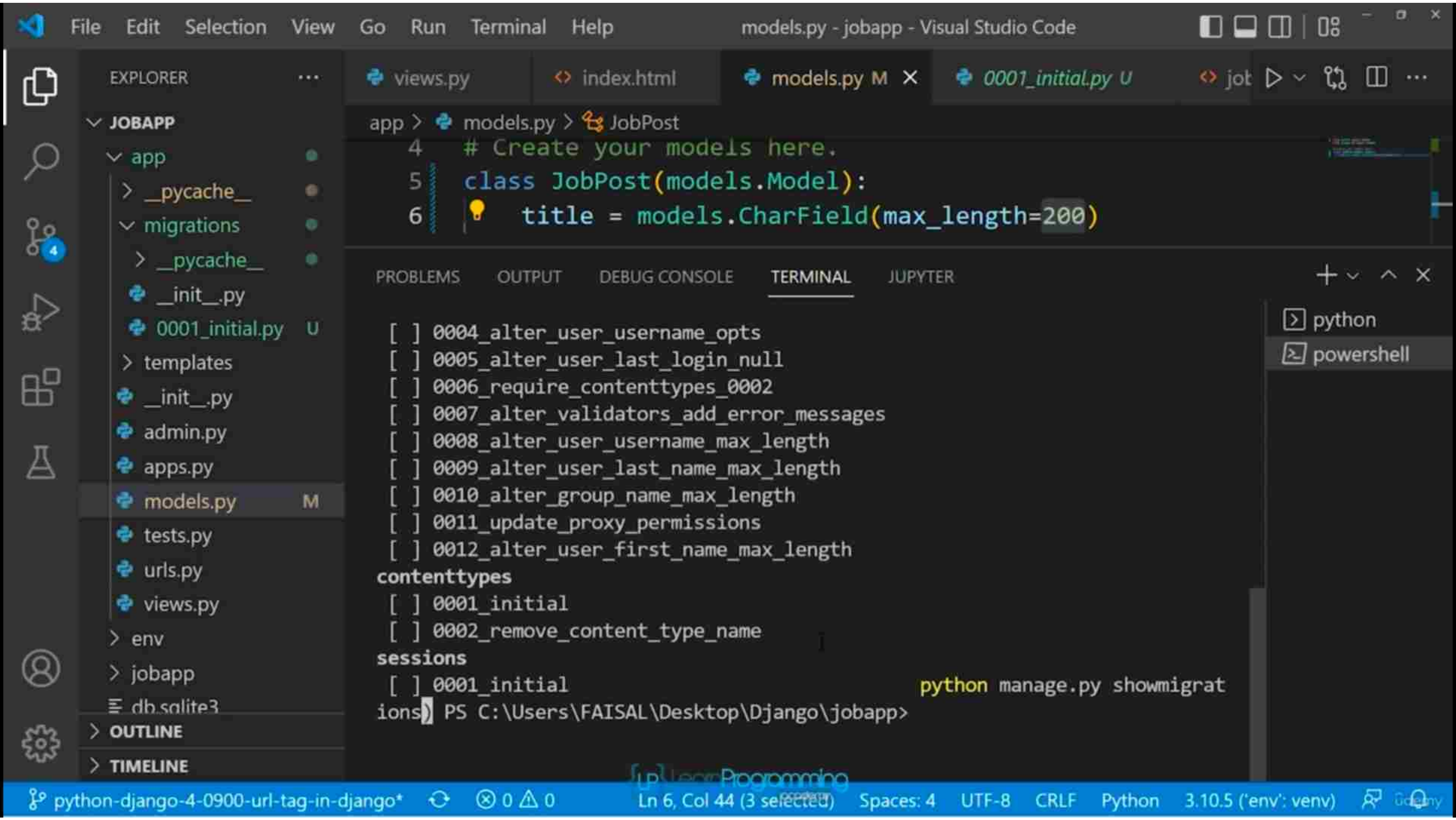
migrate → This is the command for running, applying and unapplying migrations.

showmigrations → Lists the migrations of the projects along with their status.


File Edit Selection View Go Run Terminal Help
0001_initial.py - jobapp - Visual Studio Code
EXPLORER
JOBAPP
app
pycache
migrations
pycache
init.py
0001_initial.py U
templates
init.py
admin.py
apps.py
models.py M
tests.py
urls.py
views.py
env
jobapp
db.sqlite3
OUTLINE
TIMELINE
views.py
index.html
models.py M
0001_initial.py U
app > migrations > 0001_initial.py > Migration
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
powershell
python-django-4-0900-url-tag-in-django*
Ln 19, Col 15 (206 selected) Spaces: 4 UTF-8 CRLF Python 3.10.5 ('env': venv)

```python
    operations = [
        migrations.CreateModel(
            name='JobPost',
            fields=[
```

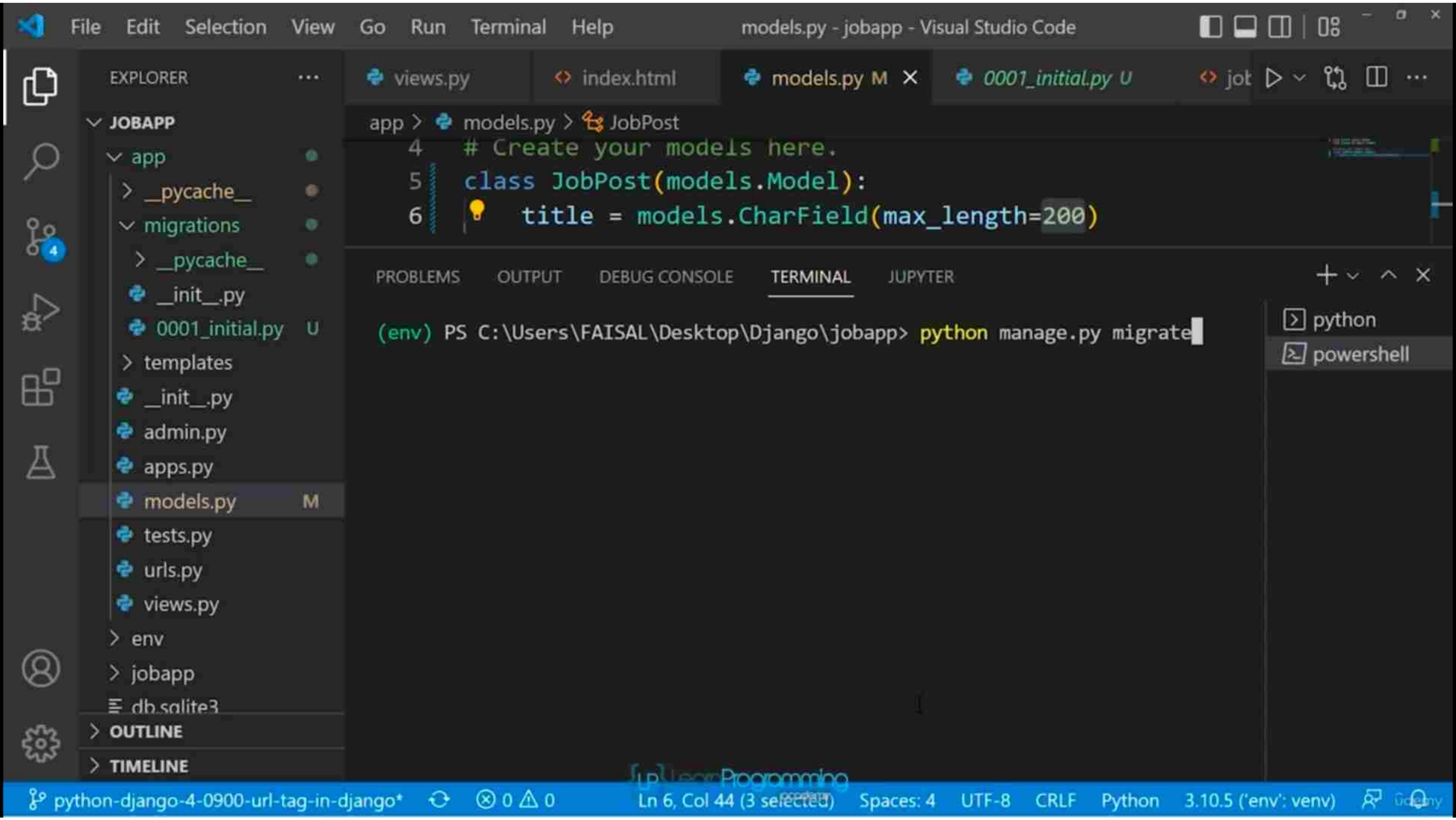
```
(env) PS C:\Users\FAISAL\Desktop\Django\jobapp> python manage.py makemigrat
ions
Migrations for 'app':
  app\migrations\0001_initial.py
    - Create model JobPost
(env) PS C:\Users\FAISAL\Desktop\Django\jobapp> python manage.py sqlmigrate
 app 0001
BEGIN;
--
-- Create model JobPost
--
CREATE TABLE "app_jobpost" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT
, "title" varchar(200) NOT NULL);
COMMIT;
(env) PS C:\Users\FAISAL\Desktop\Django\jobapp>
```



File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
EXPLORER
JOBAPP
app
pycache
migrations
pycache
init.py
0001_initial.py U
templates
init.py
admin.py
apps.py
models.py M
tests.py
urls.py
views.py
env
jobapp
db.sqlite3
OUTLINE
TIMELINE
views.py
index.html
models.py M
0001_initial.py U
app > models.py > JobPost
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
powershell
python-django-4-0900-url-tag-in-django*
Ln 6, Col 44 (3 selected) Spaces: 4 UTF-8 CRLF Python 3.10.5 ('env': venv)

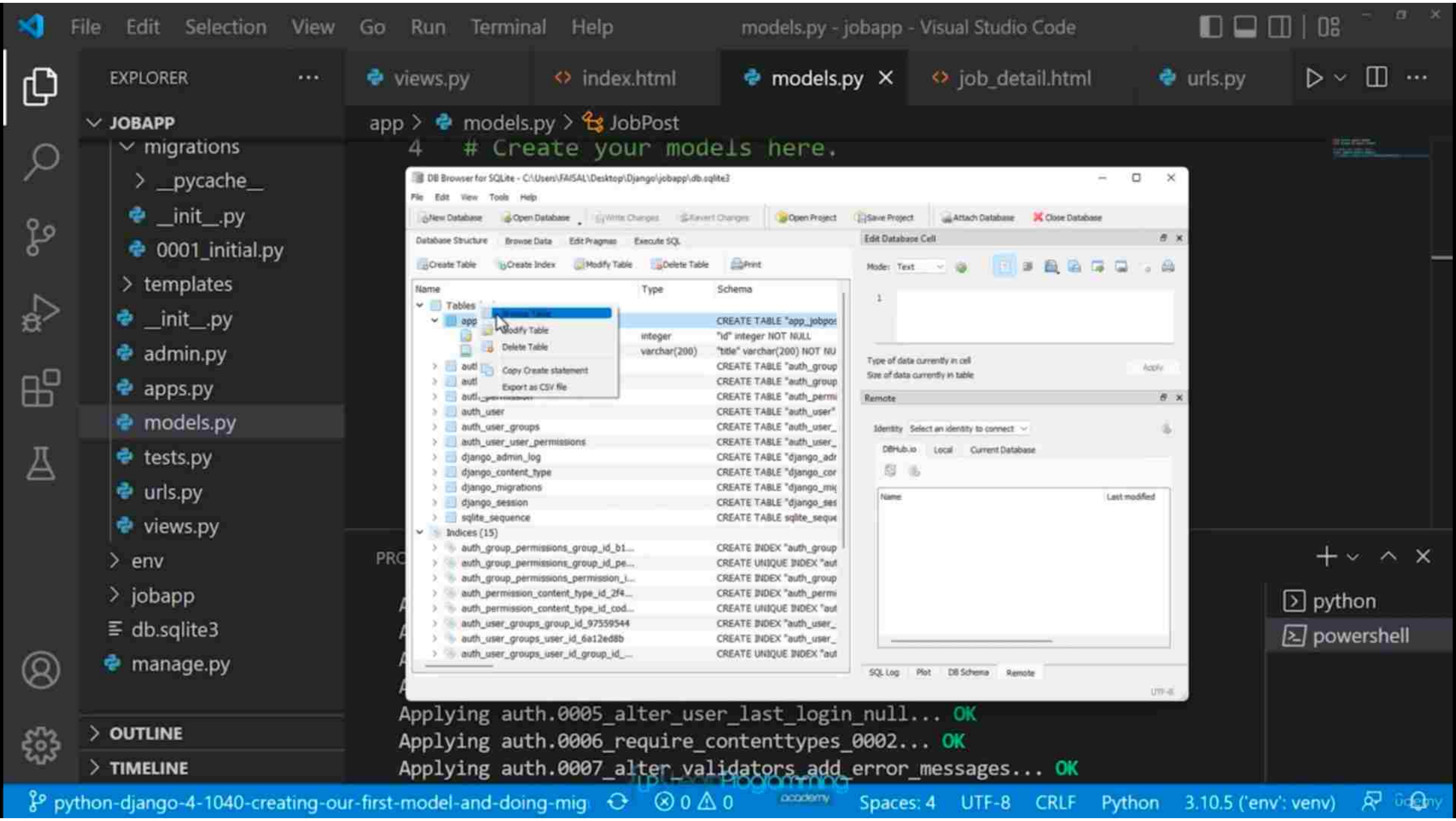
```python
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
```

```
 [ ] 0004_alter_user_username_opts
 [ ] 0005_alter_user_last_login_null
 [ ] 0006_require_contenttypes_0002
 [ ] 0007_alter_validators_add_error_messages
 [ ] 0008_alter_user_username_max_length
 [ ] 0009_alter_user_last_name_max_length
 [ ] 0010_alter_group_name_max_length
 [ ] 0011_update_proxy_permissions
 [ ] 0012_alter_user_first_name_max_length
contenttypes
 [ ] 0001_initial
 [ ] 0002_remove_content_type_name
sessions
 [ ] 0001_initial                          python manage.py showmigrat
ions PS C:\Users\FAISAL\Desktop\Django\jobapp>
```



```python
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
```

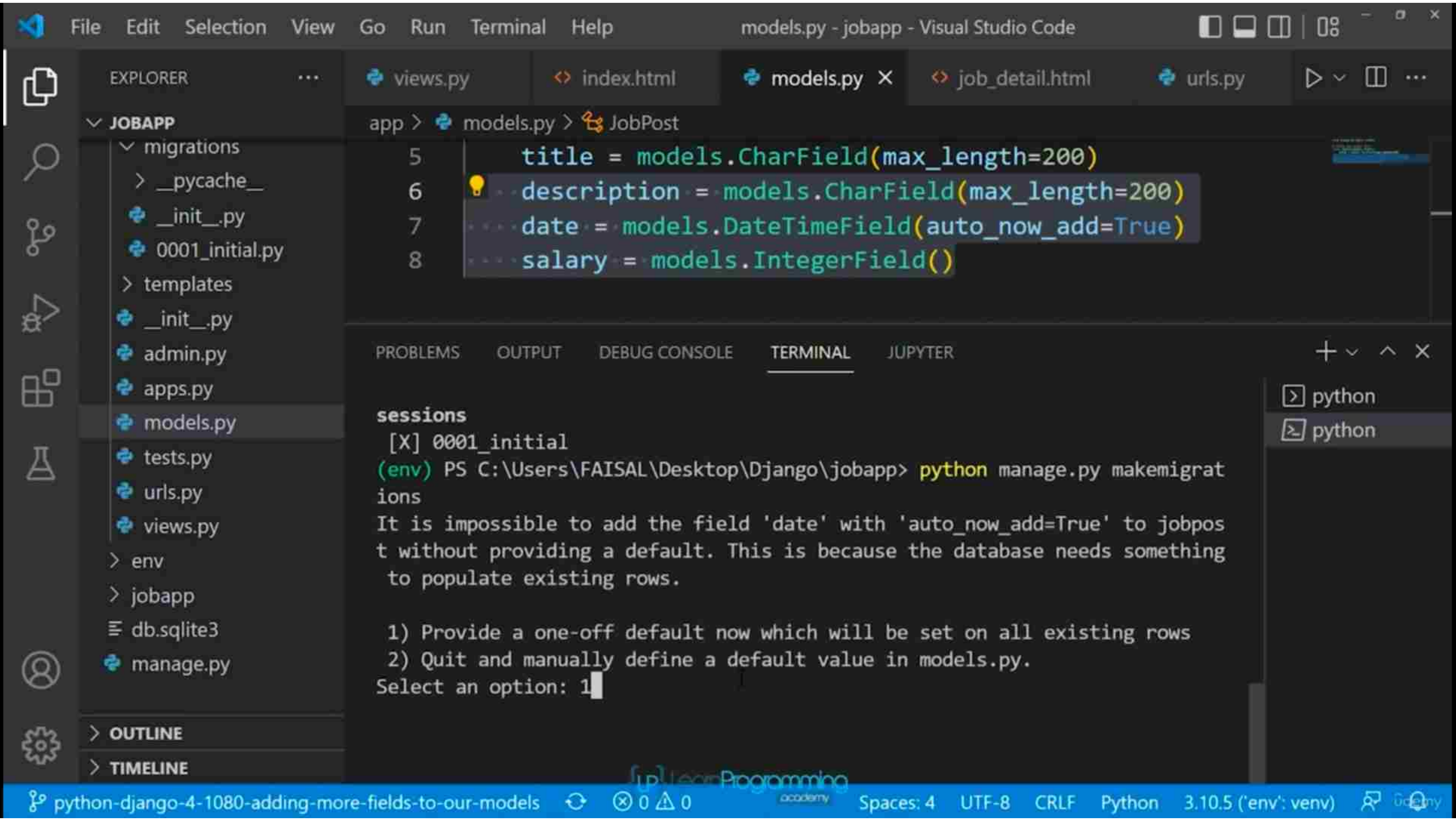
```
(env) PS C:\Users\FAISAL\Desktop\Django\jobapp> python manage.py migrate
```



```
# Create your models here.
```

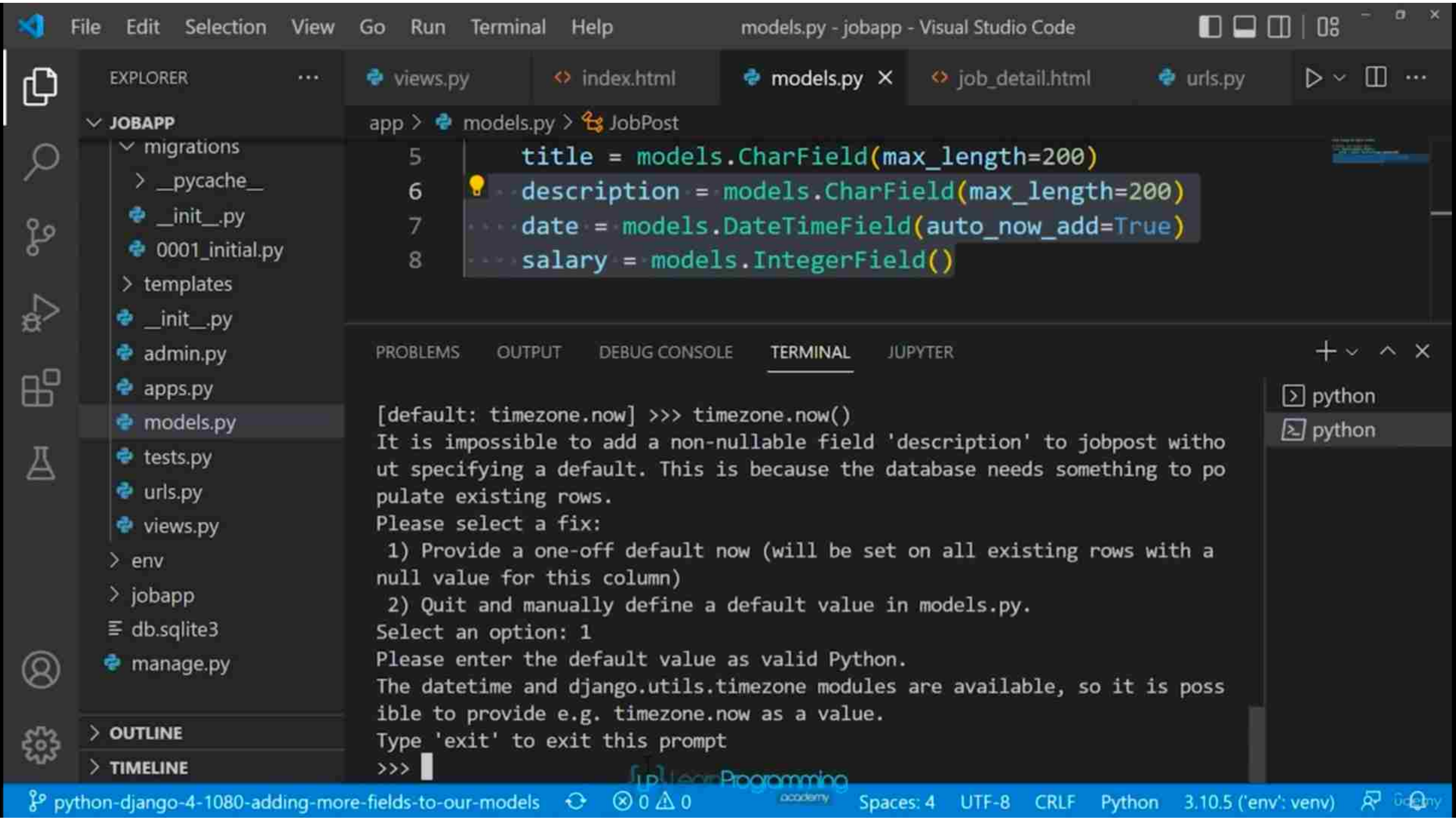
Name	Type	Schema
Tables		
app		CREATE TABLE "app_jobpos
	integer	"id" integer NOT NULL
	varchar(200)	"title" varchar(200) NOT NU
autl		CREATE TABLE "auth_group
autl		CREATE TABLE "auth_group
autl		CREATE TABLE "auth_permi
auth_user		CREATE TABLE "auth_user"
auth_user_groups		CREATE TABLE "auth_user_
auth_user_user_permissions		CREATE TABLE "auth_user_
django_admin_log		CREATE TABLE "django_adr
django_content_type		CREATE TABLE "django_cor
django_migrations		CREATE TABLE "django_mi(
django_session		CREATE TABLE "django_ses
sqlite_sequence		CREATE TABLE sqlite_seque
Indices (15)		
auth_group_permissions_group_id_b1...		CREATE INDEX "auth_group
auth_group_permissions_group_id_pe...		CREATE UNIQUE INDEX "aut
auth_group_permissions_permission_i...		CREATE INDEX "auth_group
auth_permission_content_type_id_2f4...		CREATE INDEX "auth_permi
auth_permission_content_type_id_cod...		CREATE UNIQUE INDEX "aut
auth_user_groups_group_id_97559544		CREATE INDEX "auth_user_
auth_user_groups_user_id_6a12ed8b		CREATE INDEX "auth_user_
auth_user_groups_user_id_group_id_...		CREATE UNIQUE INDEX "aut

```
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
```


File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
EXPLORER
JOBAPP
migrations
__pycache__
__init__.py
0001_initial.py
templates
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
env
jobapp
db.sqlite3
manage.py
OUTLINE
TIMELINE
views.py
index.html
models.py
job_detail.html
urls.py
app > models.py > JobPost
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
python
python-django-4-1080-adding-more-fields-to-our-models
Spaces: 4 UTF-8 CRLF Python 3.10.5 ('env': venv)

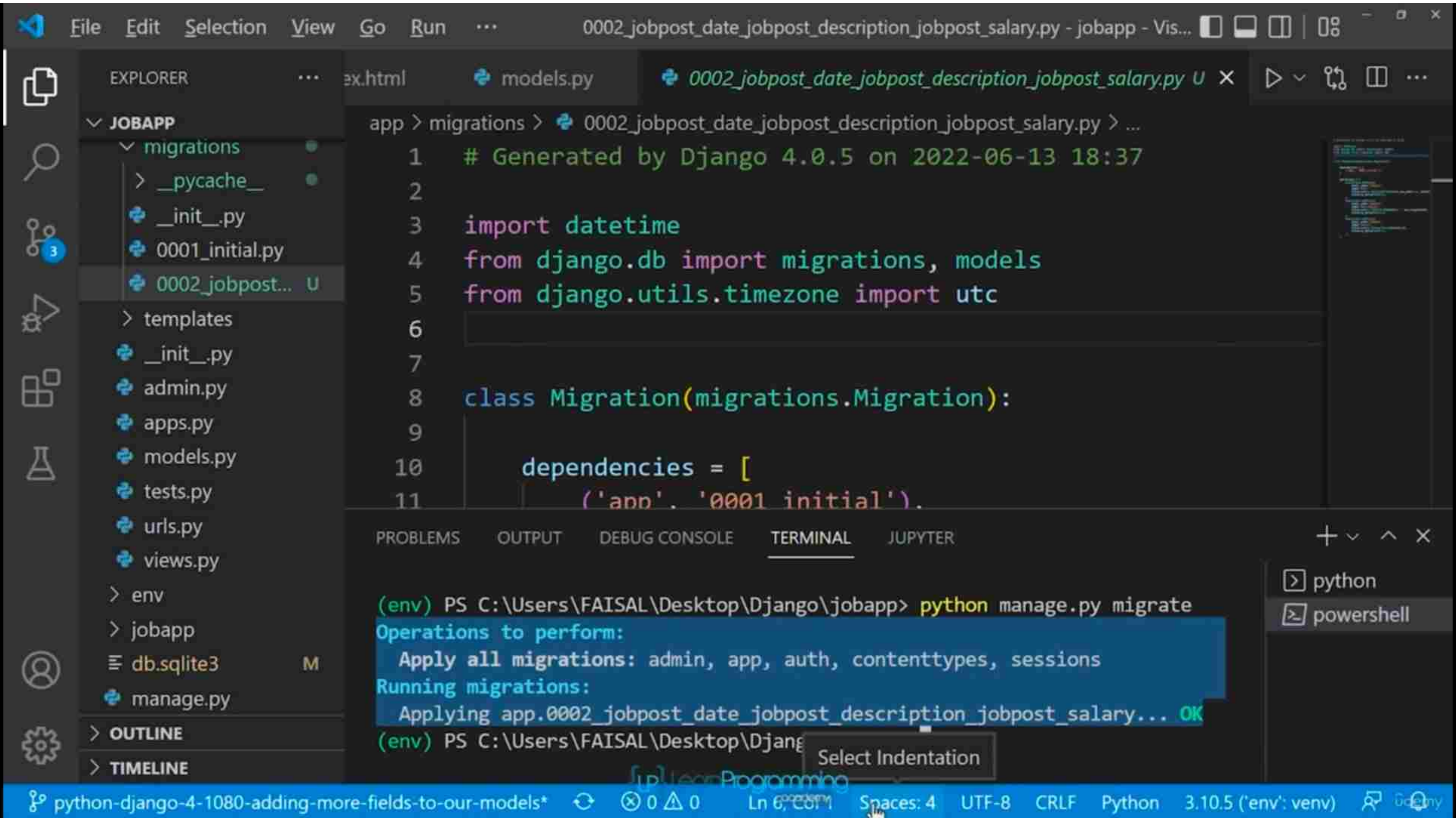
```python
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
    salary = models.IntegerField()
```

```
sessions
 [X] 0001_initial
(env) PS C:\Users\FAISAL\Desktop\Django\jobapp> python manage.py makemigrat
ions
It is impossible to add the field 'date' with 'auto_now_add=True' to jobpos
t without providing a default. This is because the database needs something
 to populate existing rows.

 1) Provide a one-off default now which will be set on all existing rows
 2) Quit and manually define a default value in models.py.
Select an option: 1
```



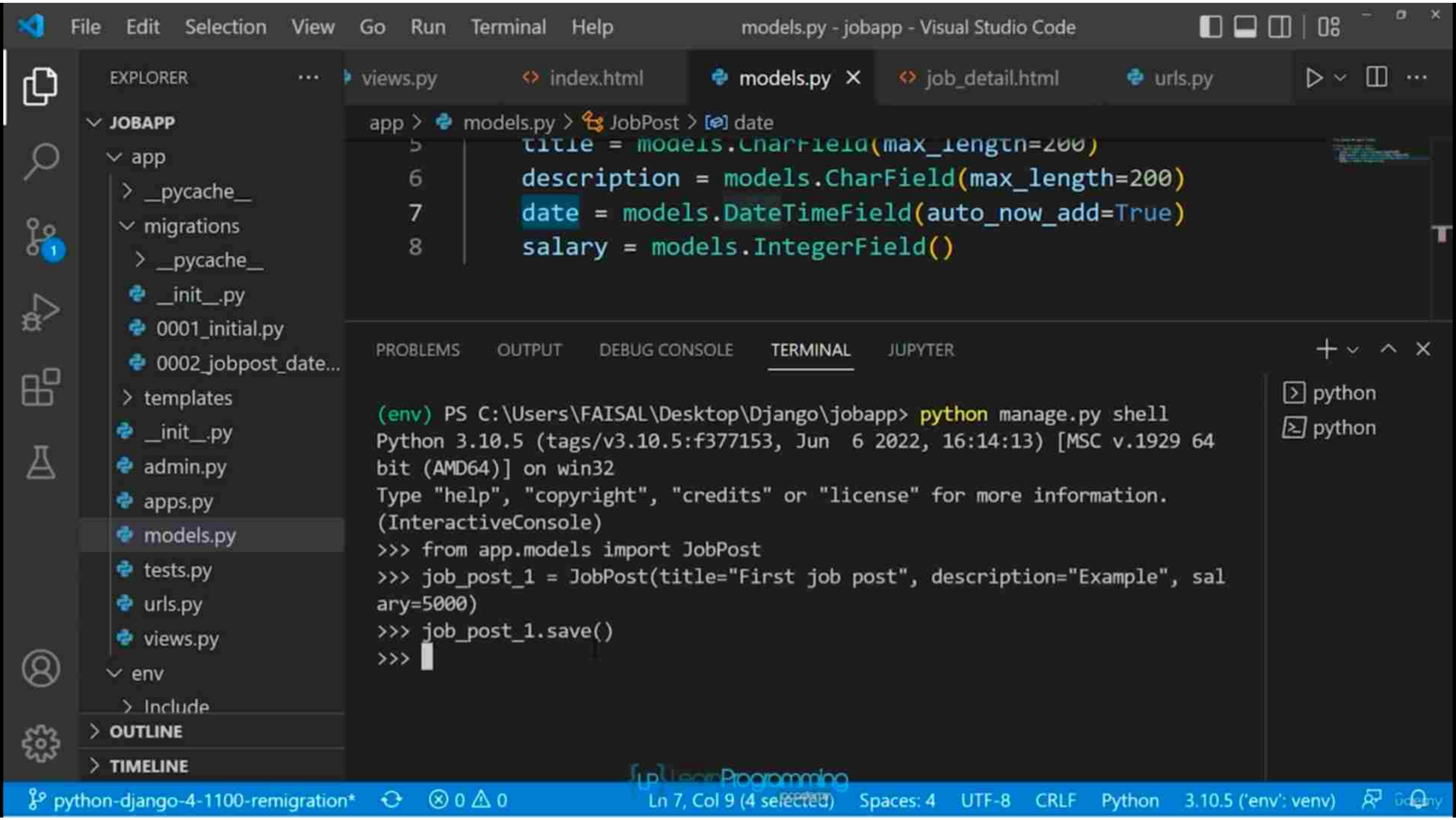
```python
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
    salary = models.IntegerField()
```

```
[default: timezone.now] >>> timezone.now()
It is impossible to add a non-nullable field 'description' to jobpost witho
ut specifying a default. This is because the database needs something to po
pulate 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 manually define a default value in models.py.
Select an option: 1
Please enter the default value as valid Python.
The datetime and django.utils.timezone modules are available, so it is poss
ible to provide e.g. timezone.now as a value.
Type 'exit' to exit this prompt
>>> 
```



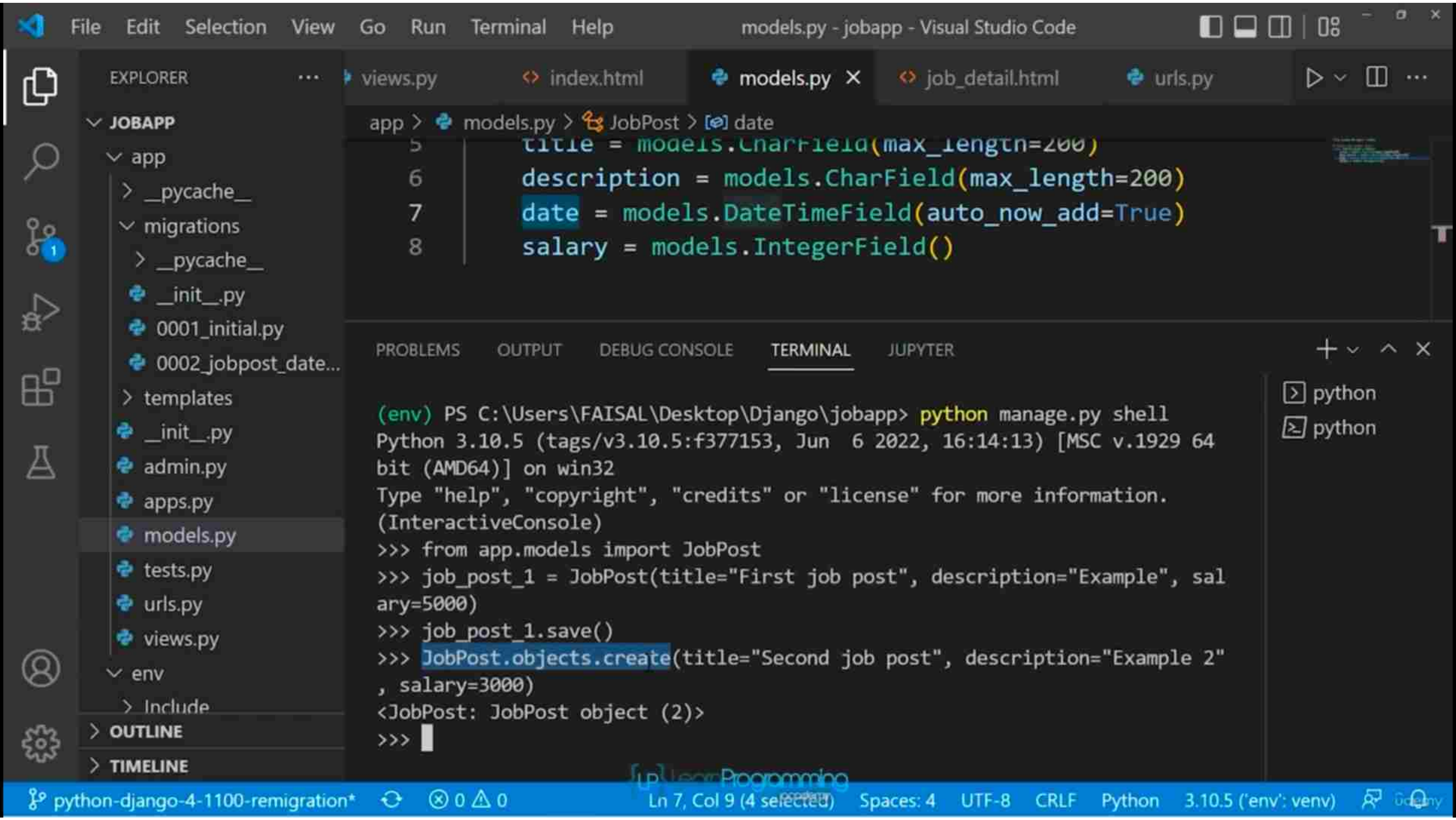
```python
# Generated by Django 4.0.5 on 2022-06-13 18:37

import datetime
from django.db import migrations, models
from django.utils.timezone import utc


class Migration(migrations.Migration):

    dependencies = [
        ('app', '0001_initial'),
```

```
(env) PS C:\Users\FAISAL\Desktop\Django\jobapp> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, app, auth, contenttypes, sessions
Running migrations:
  Applying app.0002_jobpost_date_jobpost_description_jobpost_salary... OK
(env) PS C:\Users\FAISAL\Desktop\Djang
```



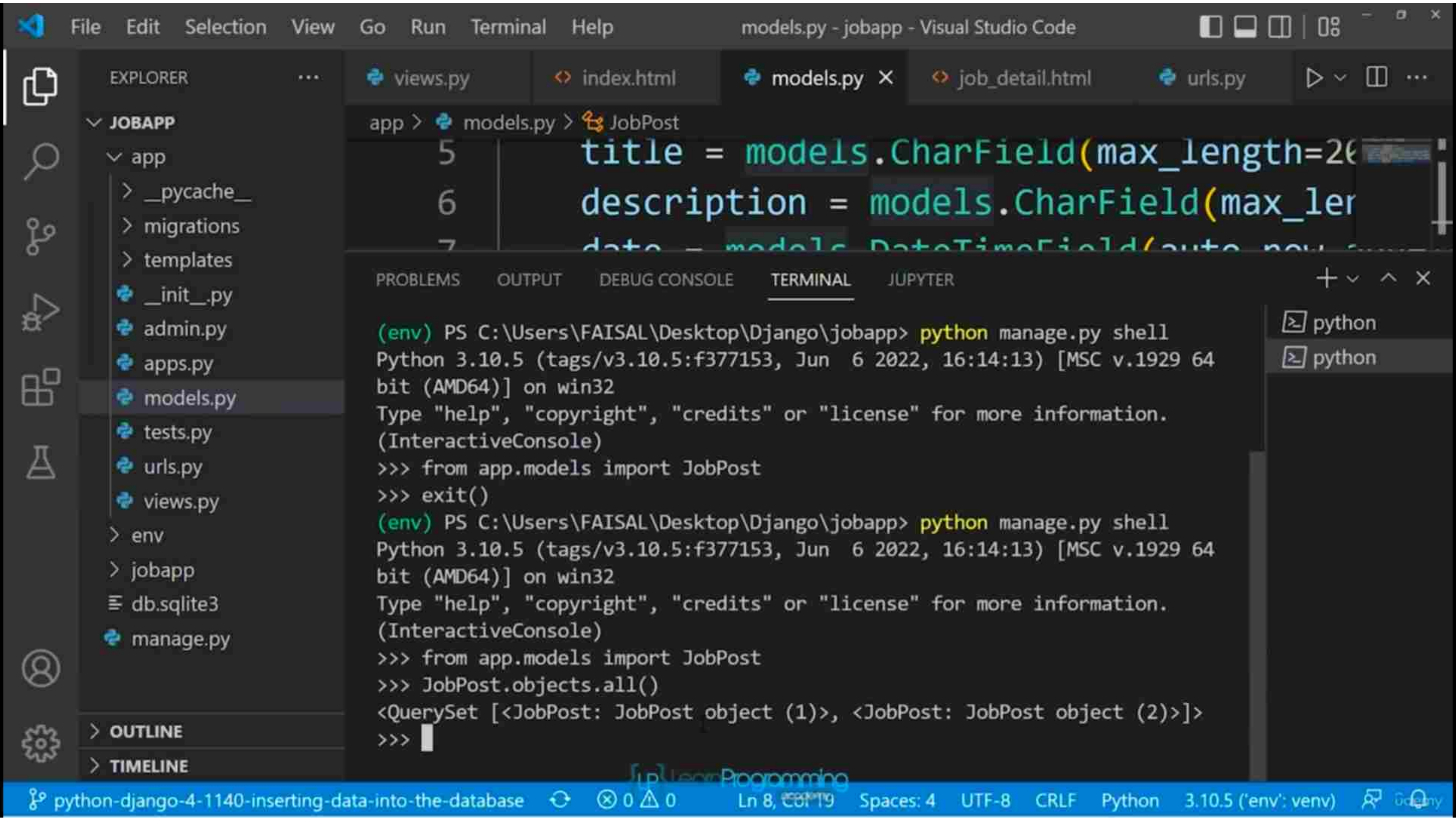
```python
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
    salary = models.IntegerField()
```

```
(env) PS C:\Users\FAISAL\Desktop\Django\jobapp> python manage.py shell
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from app.models import JobPost
>>> job_post_1 = JobPost(title="First job post", description="Example", sal
ary=5000)
>>> job_post_1.save()
>>>
```



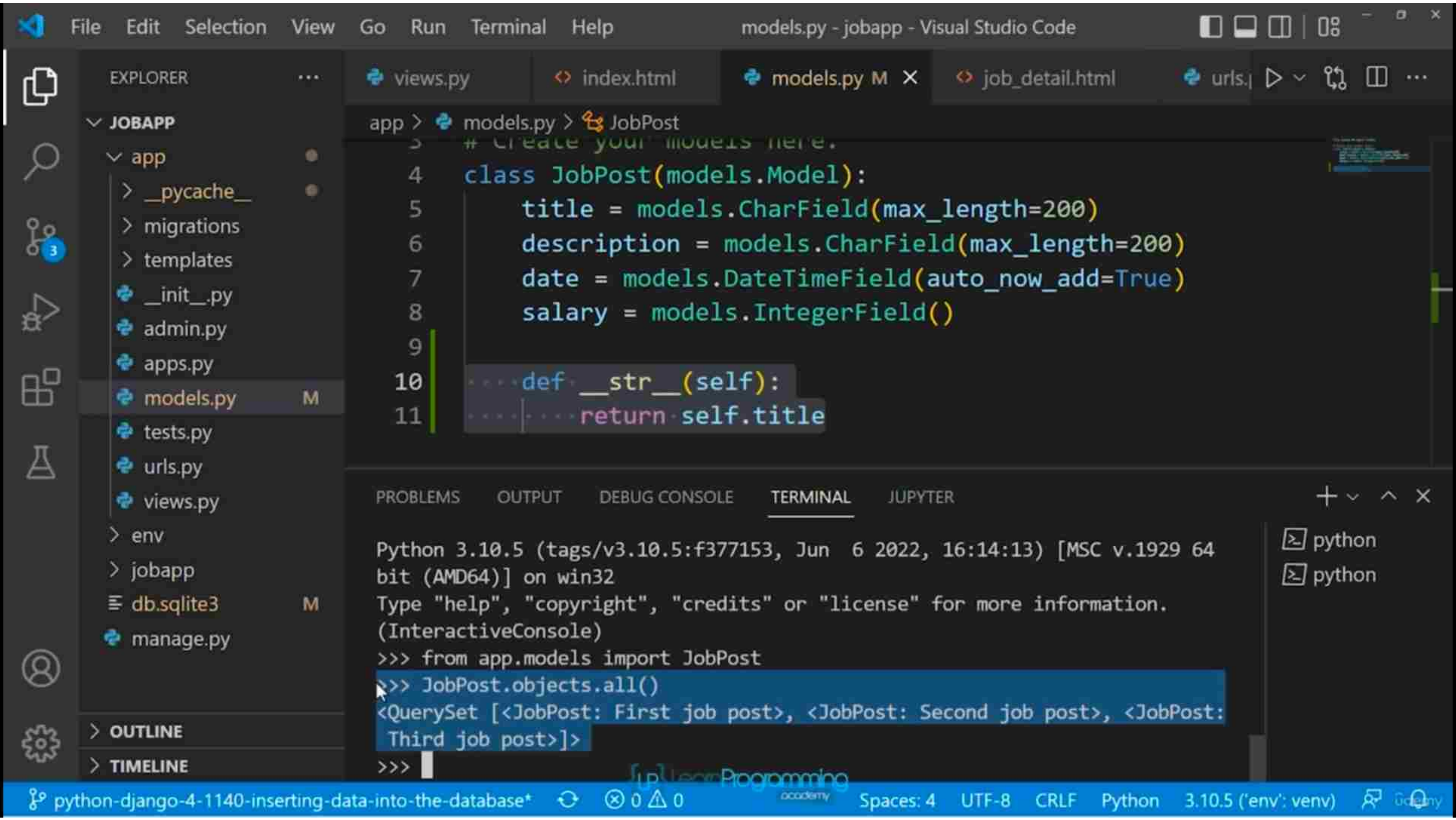
```
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
    salary = models.IntegerField()
```

```
(env) PS C:\Users\FAISAL\Desktop\Django\jobapp> python manage.py shell
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from app.models import JobPost
>>> job_post_1 = JobPost(title="First job post", description="Example", sal
ary=5000)
>>> job_post_1.save()
>>> JobPost.objects.create(title="Second job post", description="Example 2"
, salary=3000)
<JobPost: JobPost object (2)>
>>>
```



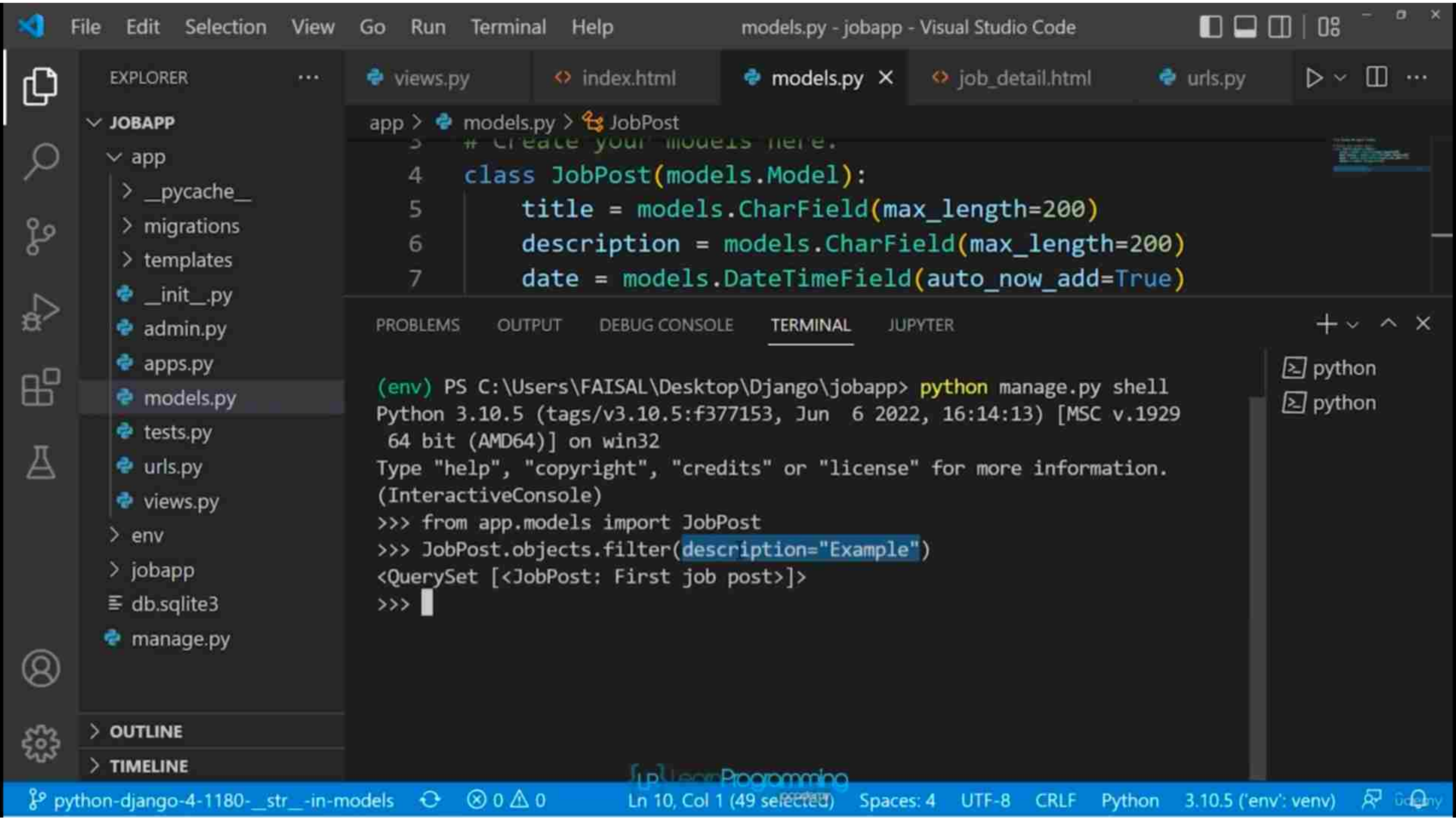
```
    title = models.CharField(max_length=2
    description = models.CharField(max_ler
```

```
(env) PS C:\Users\FAISAL\Desktop\Django\jobapp> python manage.py shell
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from app.models import JobPost
>>> exit()
(env) PS C:\Users\FAISAL\Desktop\Django\jobapp> python manage.py shell
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from app.models import JobPost
>>> JobPost.objects.all()
<QuerySet [<JobPost: JobPost object (1)>, <JobPost: JobPost object (2)>]>
>>>
```



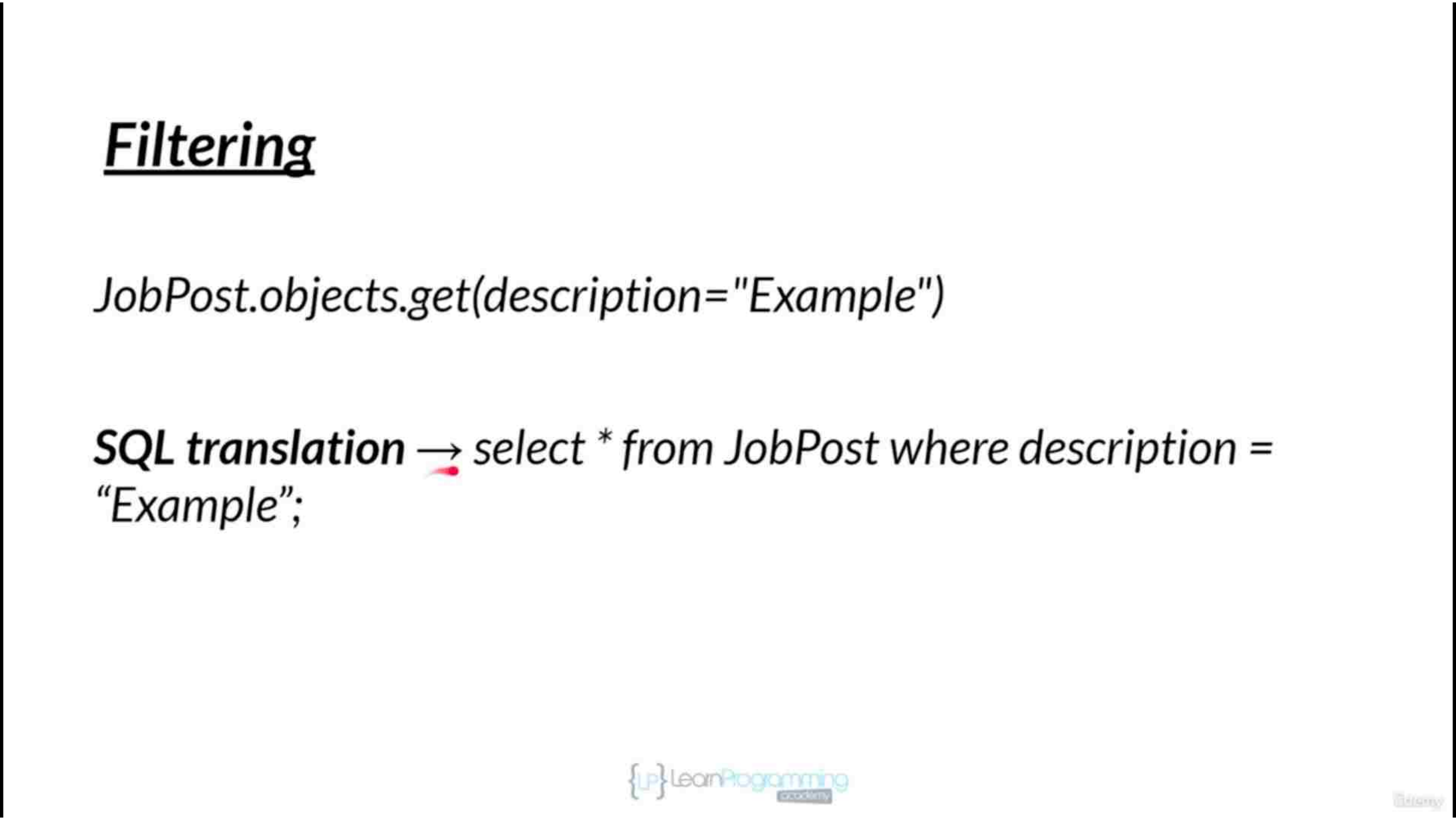
```python
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
    salary = models.IntegerField()

    def __str__(self):
        return self.title
```

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from app.models import JobPost
>>> JobPost.objects.all()
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobPost:
 Third job post>]>
>>>
```



```
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
```

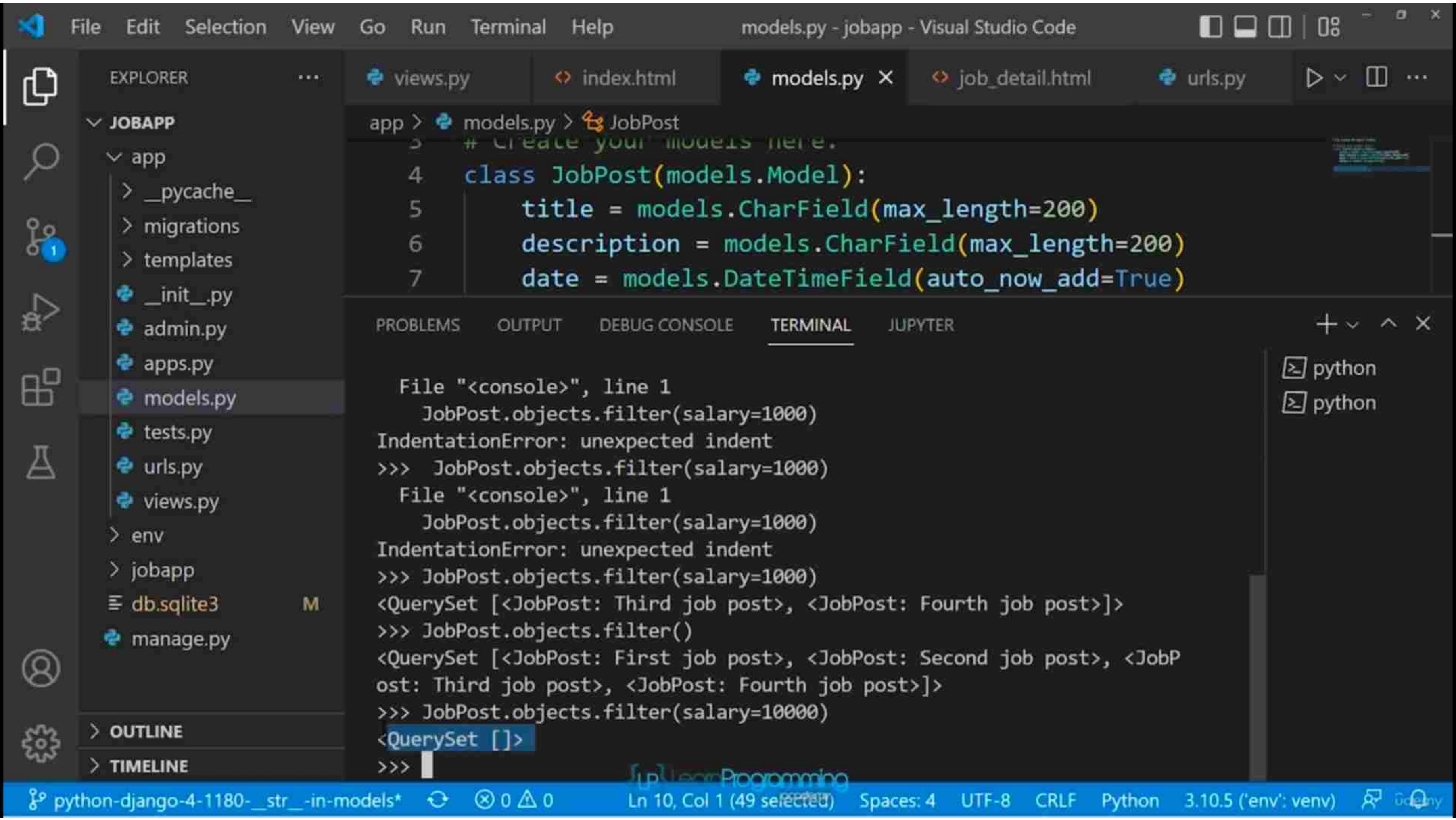
```
(env) PS C:\Users\FAISAL\Desktop\Django\jobapp> python manage.py shell
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929
 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from app.models import JobPost
>>> JobPost.objects.filter(description="Example")
<QuerySet [<JobPost: First job post>]>
>>>
```



Filtering

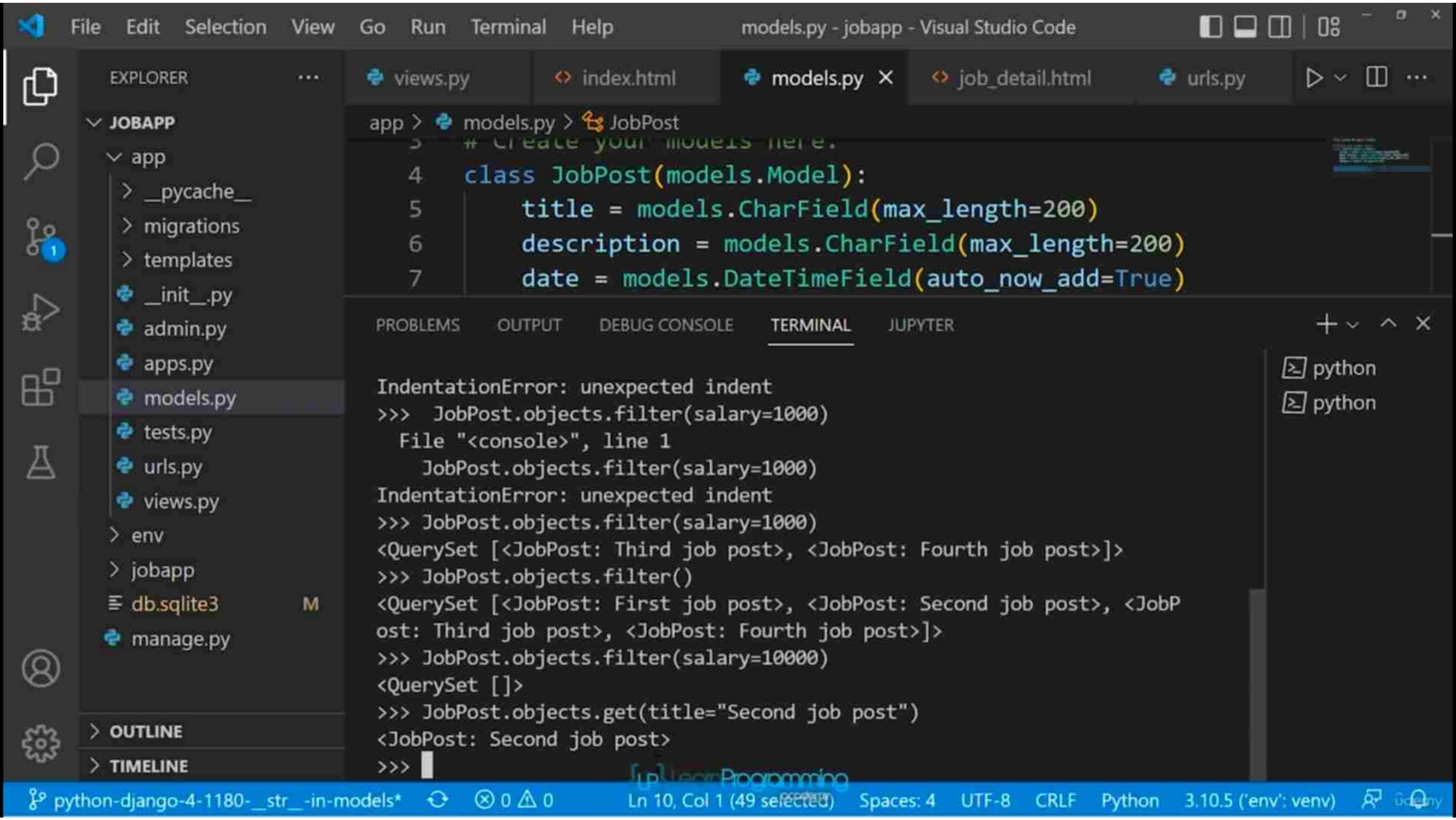
JobPost.objects.get(description="Example")

SQL translation $\rightarrow$ *select * from JobPost where description = "Example";*


File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
EXPLORER
JOBAPP
app
pycache
migrations
templates
init.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
env
jobapp
db.sqlite3 M
manage.py
OUTLINE
TIMELINE
views.py
index.html
models.py
job_detail.html
urls.py
app > models.py > JobPost
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
python
python-django-4-1180-__str__-in-models*
Ln 10, Col 1 (49 selected) Spaces: 4 UTF-8 CRLF Python 3.10.5 ('env': venv)

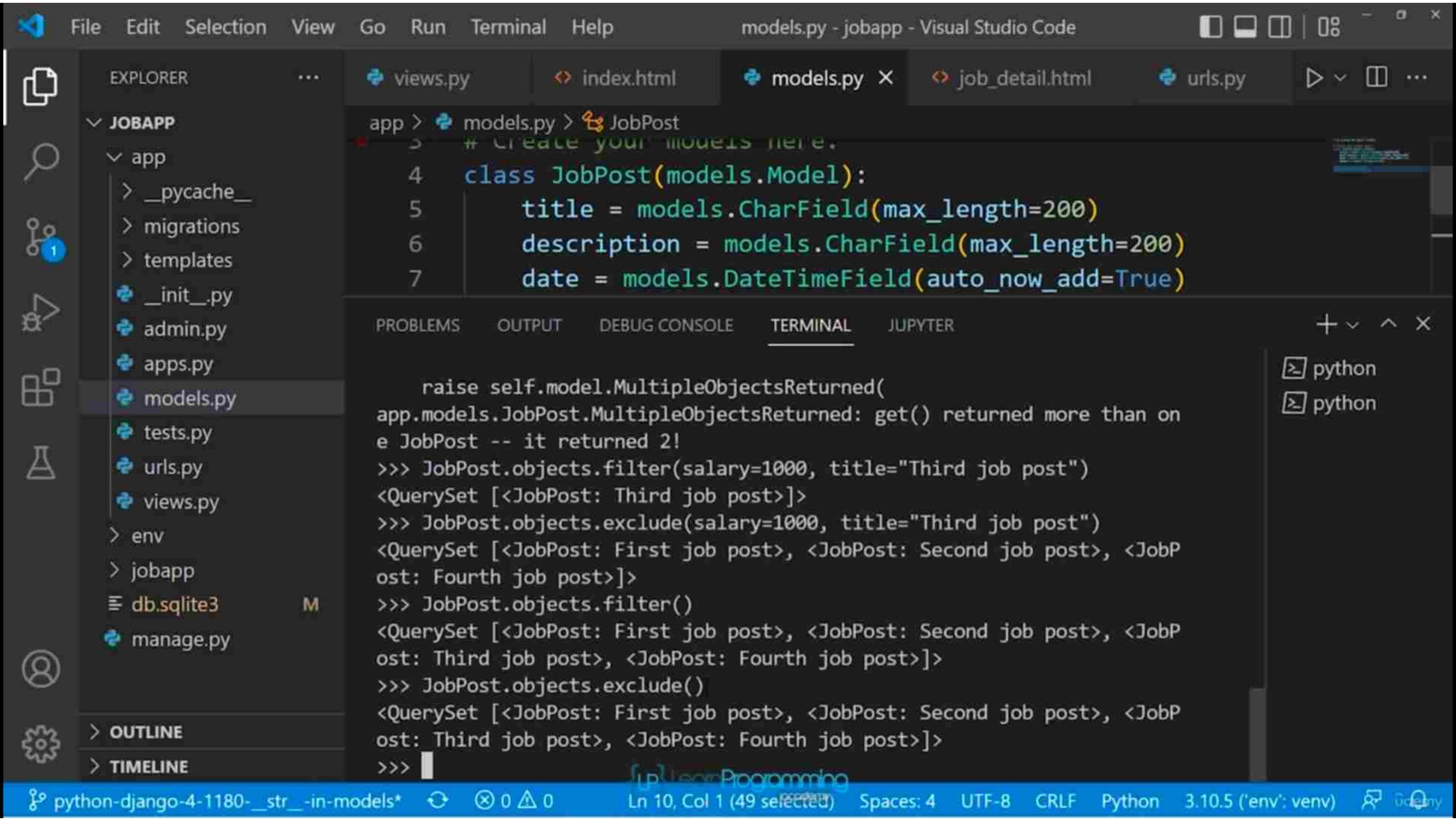
```python
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
```

```
  File "<console>", line 1
    JobPost.objects.filter(salary=1000)
IndentationError: unexpected indent
>>>  JobPost.objects.filter(salary=1000)
  File "<console>", line 1
    JobPost.objects.filter(salary=1000)
IndentationError: unexpected indent
>>> JobPost.objects.filter(salary=1000)
<QuerySet [<JobPost: Third job post>, <JobPost: Fourth job post>]>
>>> JobPost.objects.filter()
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Third job post>, <JobPost: Fourth job post>]>
>>> JobPost.objects.filter(salary=10000)
<QuerySet []>
>>>
```



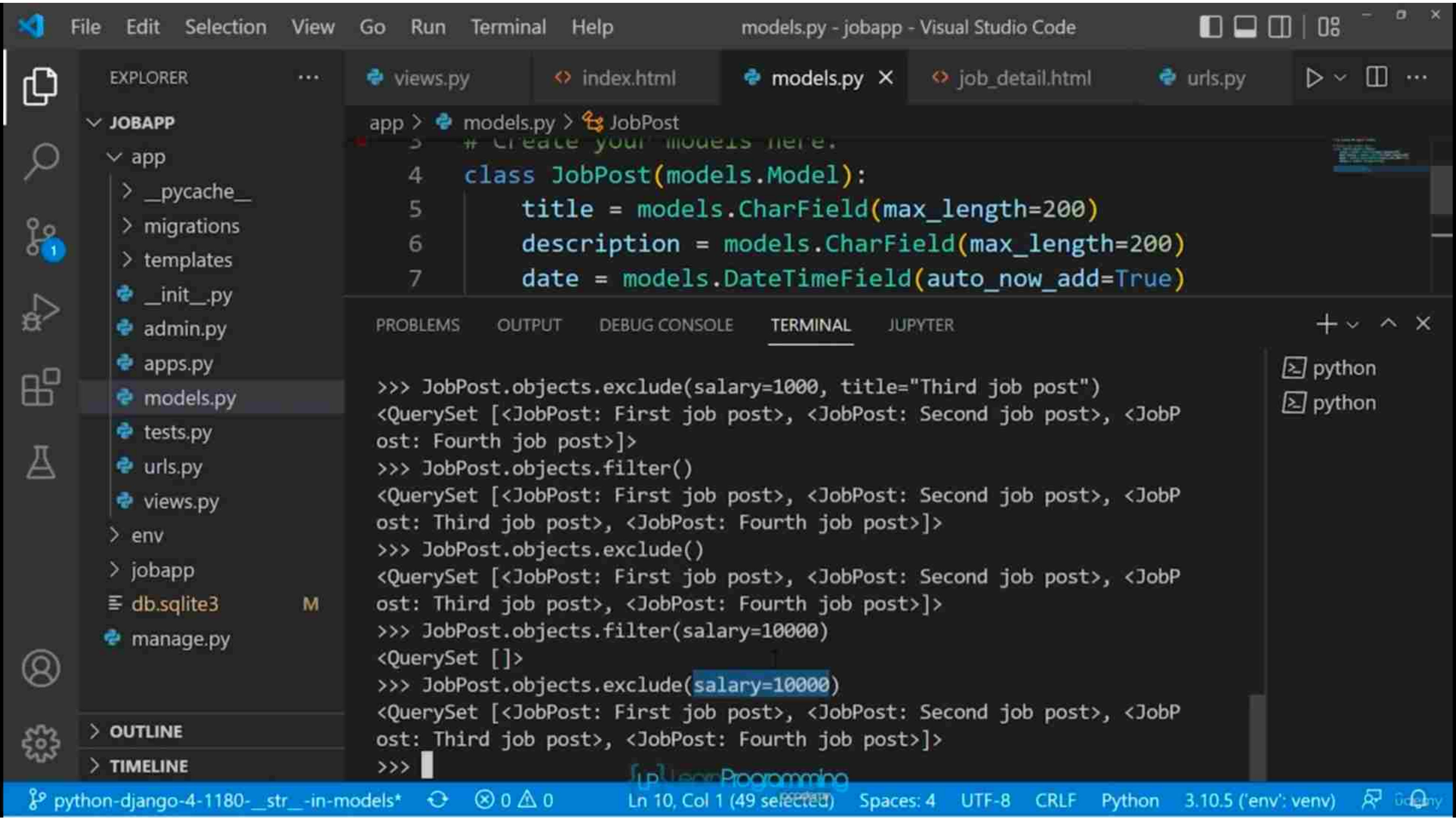
```python
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
```

```
IndentationError: unexpected indent
>>>  JobPost.objects.filter(salary=1000)
  File "<console>", line 1
    JobPost.objects.filter(salary=1000)
IndentationError: unexpected indent
>>> JobPost.objects.filter(salary=1000)
<QuerySet [<JobPost: Third job post>, <JobPost: Fourth job post>]>
>>> JobPost.objects.filter()
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Third job post>, <JobPost: Fourth job post>]>
>>> JobPost.objects.filter(salary=10000)
<QuerySet []>
>>> JobPost.objects.get(title="Second job post")
<JobPost: Second job post>
>>> 
```



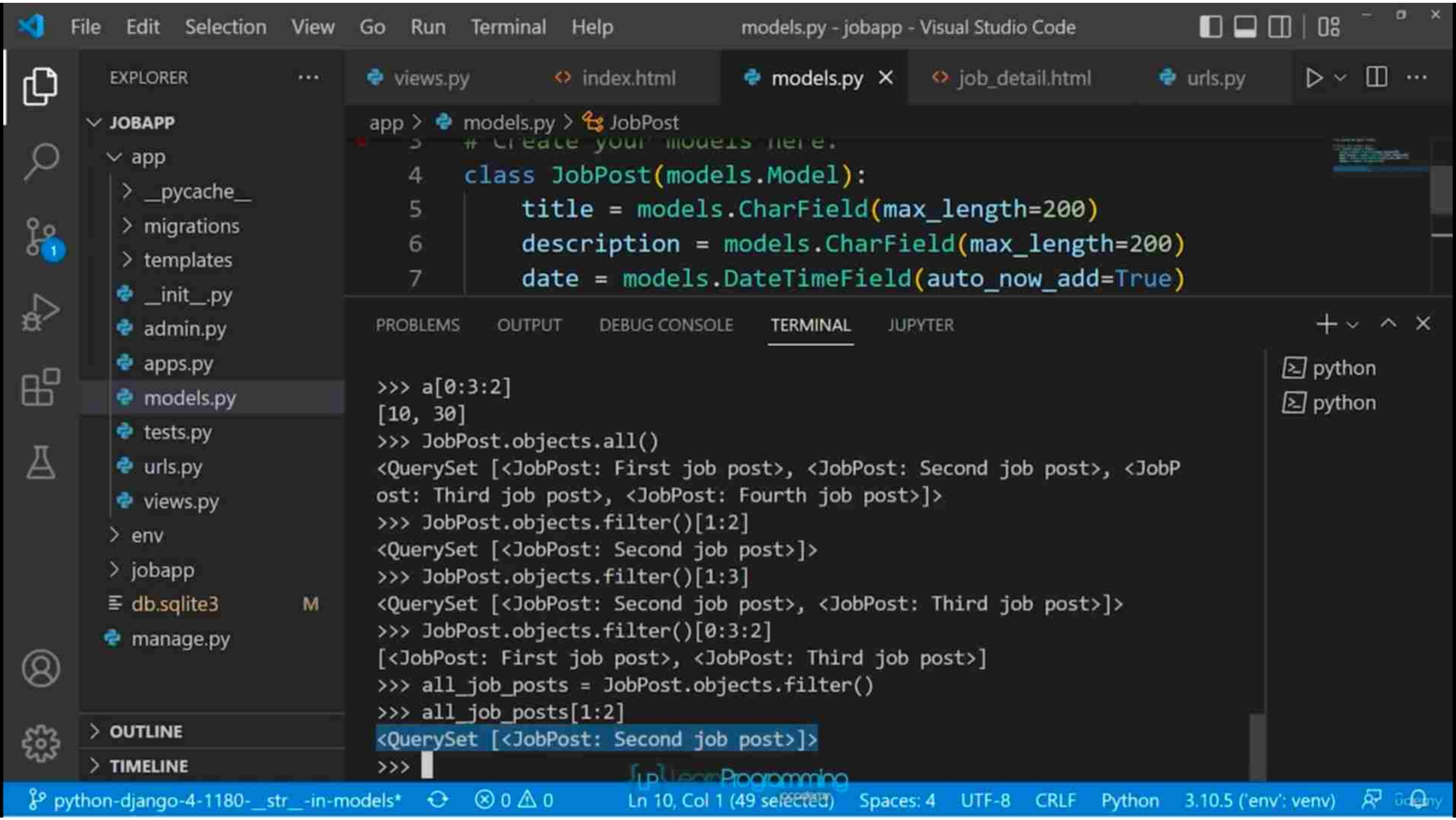
```python
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
```

```
    raise self.model.MultipleObjectsReturned(
app.models.JobPost.MultipleObjectsReturned: get() returned more than on
e JobPost -- it returned 2!
>>> JobPost.objects.filter(salary=1000, title="Third job post")
<QuerySet [<JobPost: Third job post>]>
>>> JobPost.objects.exclude(salary=1000, title="Third job post")
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Fourth job post>]>
>>> JobPost.objects.filter()
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Third job post>, <JobPost: Fourth job post>]>
>>> JobPost.objects.exclude()
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Third job post>, <JobPost: Fourth job post>]>
>>> 
```



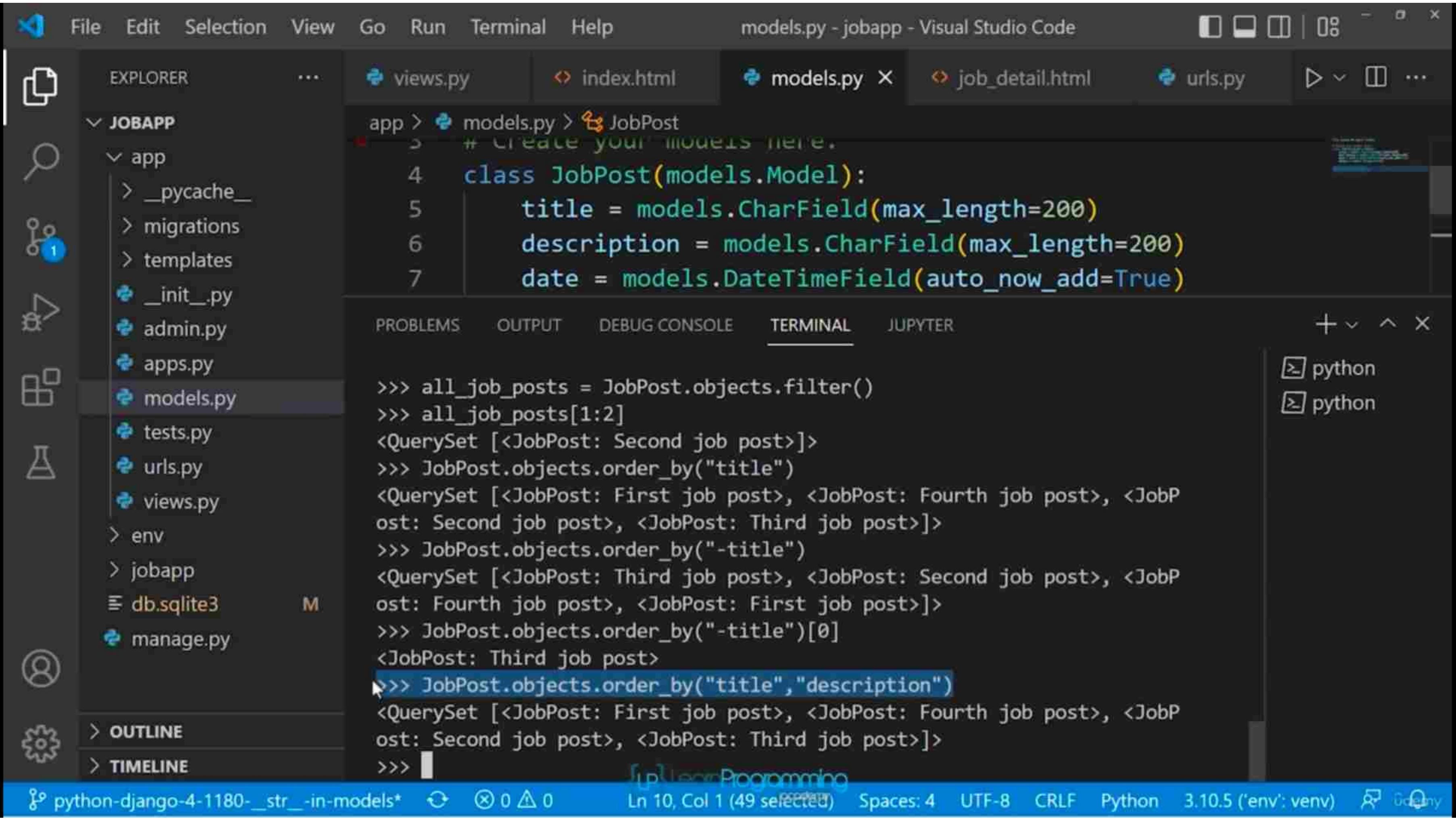
```python
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
```

```
>>> JobPost.objects.exclude(salary=1000, title="Third job post")
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Fourth job post>]>
>>> JobPost.objects.filter()
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Third job post>, <JobPost: Fourth job post>]>
>>> JobPost.objects.exclude()
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Third job post>, <JobPost: Fourth job post>]>
>>> JobPost.objects.filter(salary=10000)
<QuerySet []>
>>> JobPost.objects.exclude(salary=10000)
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Third job post>, <JobPost: Fourth job post>]>
>>>
```


File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
EXPLORER
JOBAPP
app
__pycache__
migrations
templates
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
env
jobapp
db.sqlite3 M
manage.py
OUTLINE
TIMELINE
views.py index.html models.py job_detail.html urls.py
app > models.py > JobPost
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
python
python-django-4-1180-__str__-in-models* 0 0 Ln 10, Col 1 (49 selected) Spaces: 4 UTF-8 CRLF Python 3.10.5 ('env': venv)

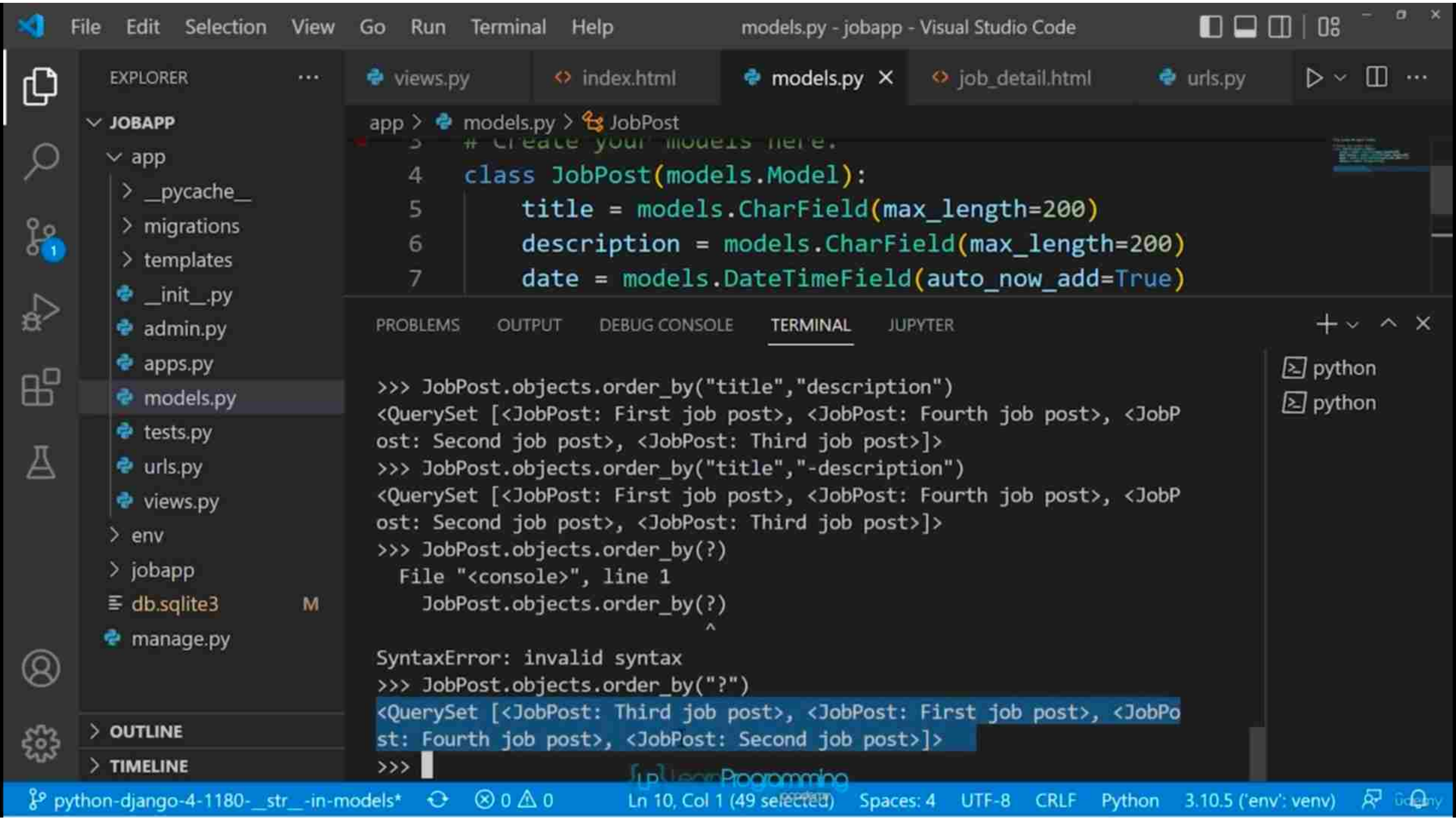
```python
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
```

```
>>> a[0:3:2]
[10, 30]
>>> JobPost.objects.all()
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Third job post>, <JobPost: Fourth job post>]>
>>> JobPost.objects.filter()[1:2]
<QuerySet [<JobPost: Second job post>]>
>>> JobPost.objects.filter()[1:3]
<QuerySet [<JobPost: Second job post>, <JobPost: Third job post>]>
>>> JobPost.objects.filter()[0:3:2]
[<JobPost: First job post>, <JobPost: Third job post>]
>>> all_job_posts = JobPost.objects.filter()
>>> all_job_posts[1:2]
<QuerySet [<JobPost: Second job post>]>
>>>
```


File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
EXPLORER
JOBAPP
app
pycache
migrations
templates
init.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
env
jobapp
db.sqlite3 M
manage.py
OUTLINE
TIMELINE
views.py
index.html
models.py
job_detail.html
urls.py
app > models.py > JobPost
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
python
python-django-4-1180-_str_-in-models*
Ln 10, Col 1 (49 selected) Spaces: 4 UTF-8 CRLF Python 3.10.5 ('env': venv)

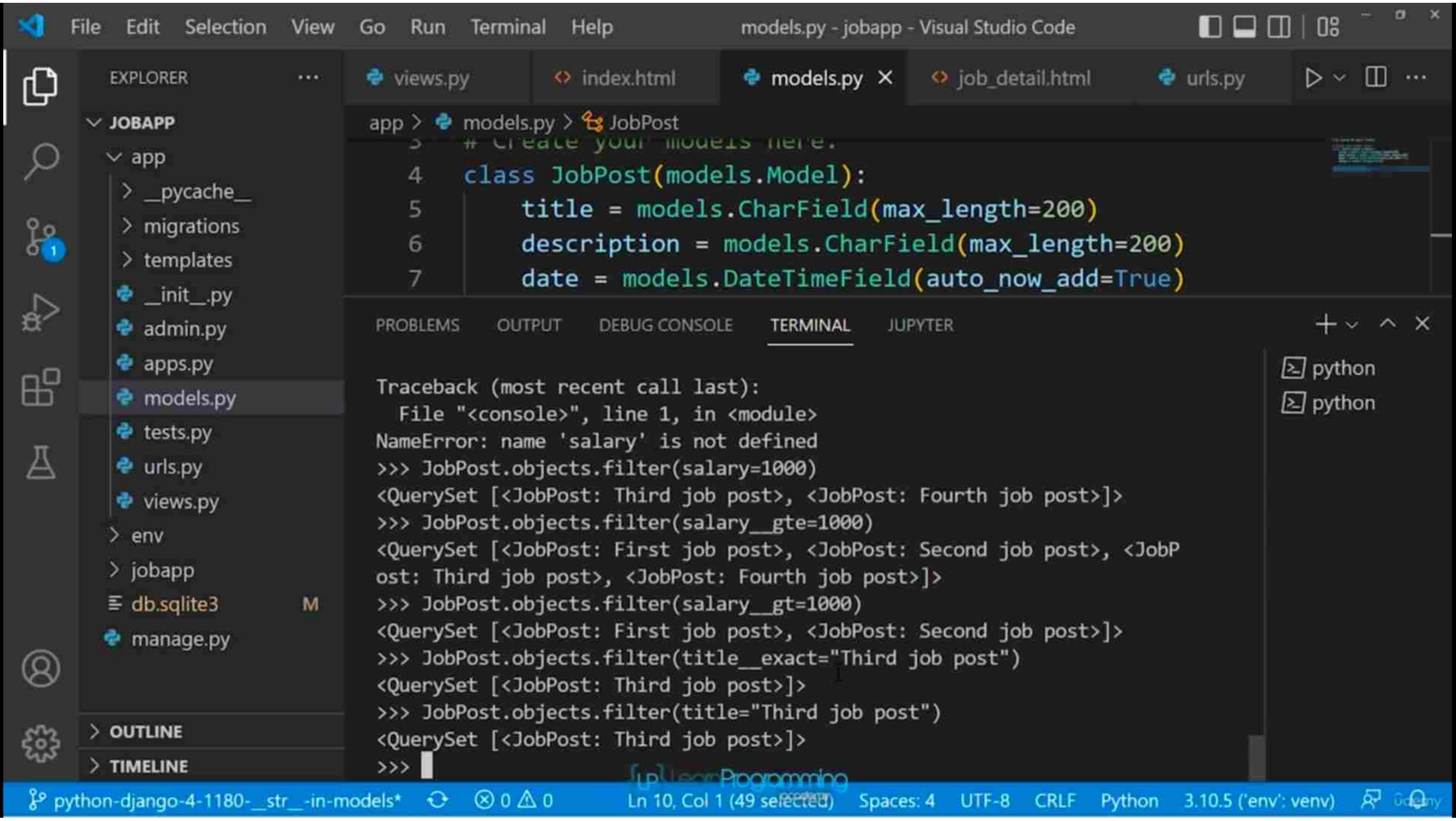
```python
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
```

```
>>> all_job_posts = JobPost.objects.filter()
>>> all_job_posts[1:2]
<QuerySet [<JobPost: Second job post>]>
>>> JobPost.objects.order_by("title")
<QuerySet [<JobPost: First job post>, <JobPost: Fourth job post>, <JobP
ost: Second job post>, <JobPost: Third job post>]>
>>> JobPost.objects.order_by("-title")
<QuerySet [<JobPost: Third job post>, <JobPost: Second job post>, <JobP
ost: Fourth job post>, <JobPost: First job post>]>
>>> JobPost.objects.order_by("-title")[0]
<JobPost: Third job post>
>>> JobPost.objects.order_by("title","description")
<QuerySet [<JobPost: First job post>, <JobPost: Fourth job post>, <JobP
ost: Second job post>, <JobPost: Third job post>]>
>>>
```



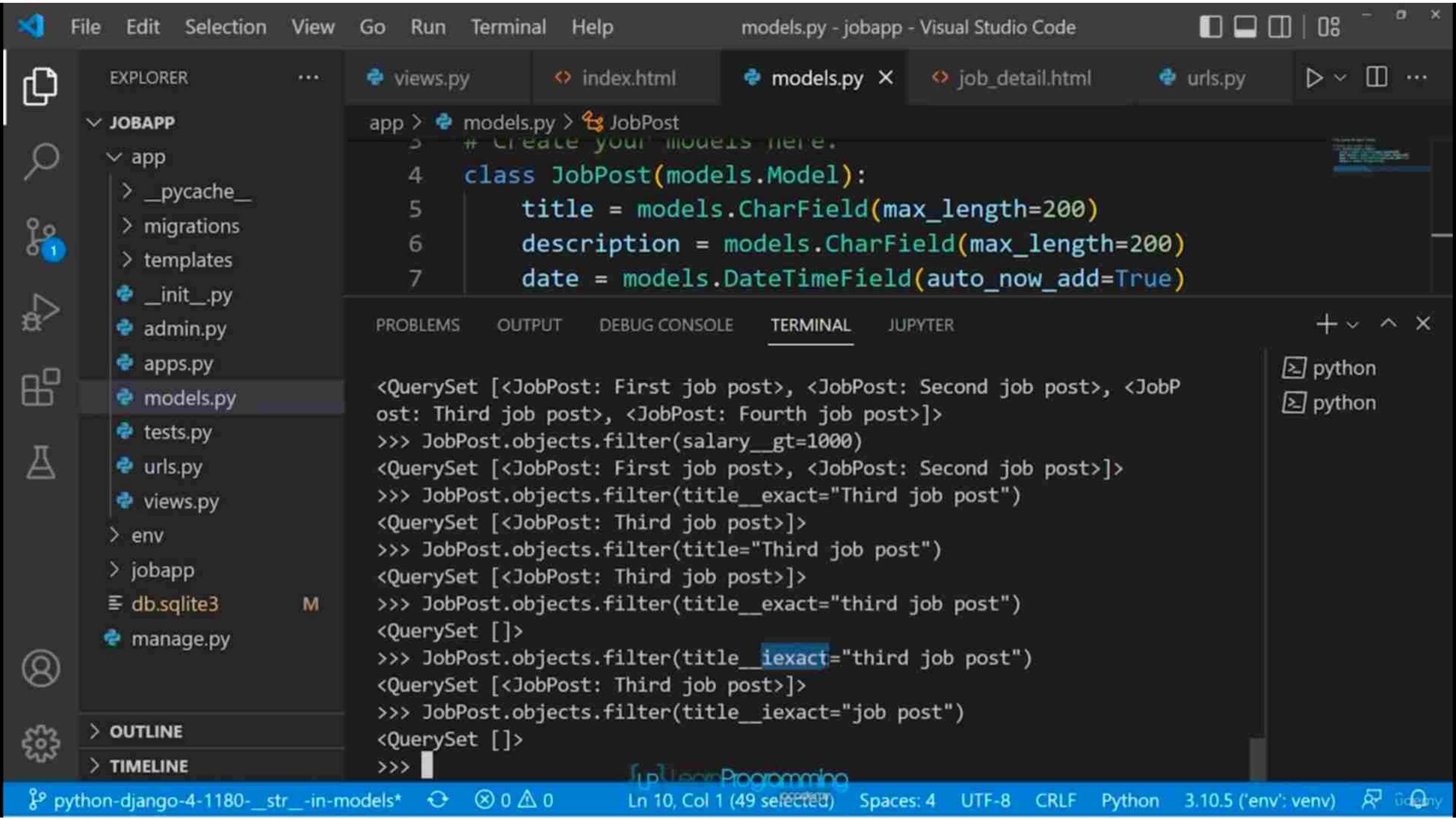
```python
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
```

```
>>> JobPost.objects.order_by("title","description")
<QuerySet [<JobPost: First job post>, <JobPost: Fourth job post>, <JobP
ost: Second job post>, <JobPost: Third job post>]>
>>> JobPost.objects.order_by("title","-description")
<QuerySet [<JobPost: First job post>, <JobPost: Fourth job post>, <JobP
ost: Second job post>, <JobPost: Third job post>]>
>>> JobPost.objects.order_by(?)
  File "<console>", line 1
    JobPost.objects.order_by(?)
                             ^
SyntaxError: invalid syntax
>>> JobPost.objects.order_by("?")
<QuerySet [<JobPost: Third job post>, <JobPost: First job post>, <JobPo
st: Fourth job post>, <JobPost: Second job post>]>
>>> 
```


File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
EXPLORER
JOBAPP
app
pycache
migrations
templates
init.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
env
jobapp
db.sqlite3 M
manage.py
OUTLINE
TIMELINE
views.py index.html models.py job_detail.html urls.py
app > models.py > JobPost
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
python
python-django-4-1180-__str__-in-models*
Ln 10, Col 1 (49 selected) Spaces: 4 UTF-8 CRLF Python 3.10.5 ('env': venv)

```
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
```

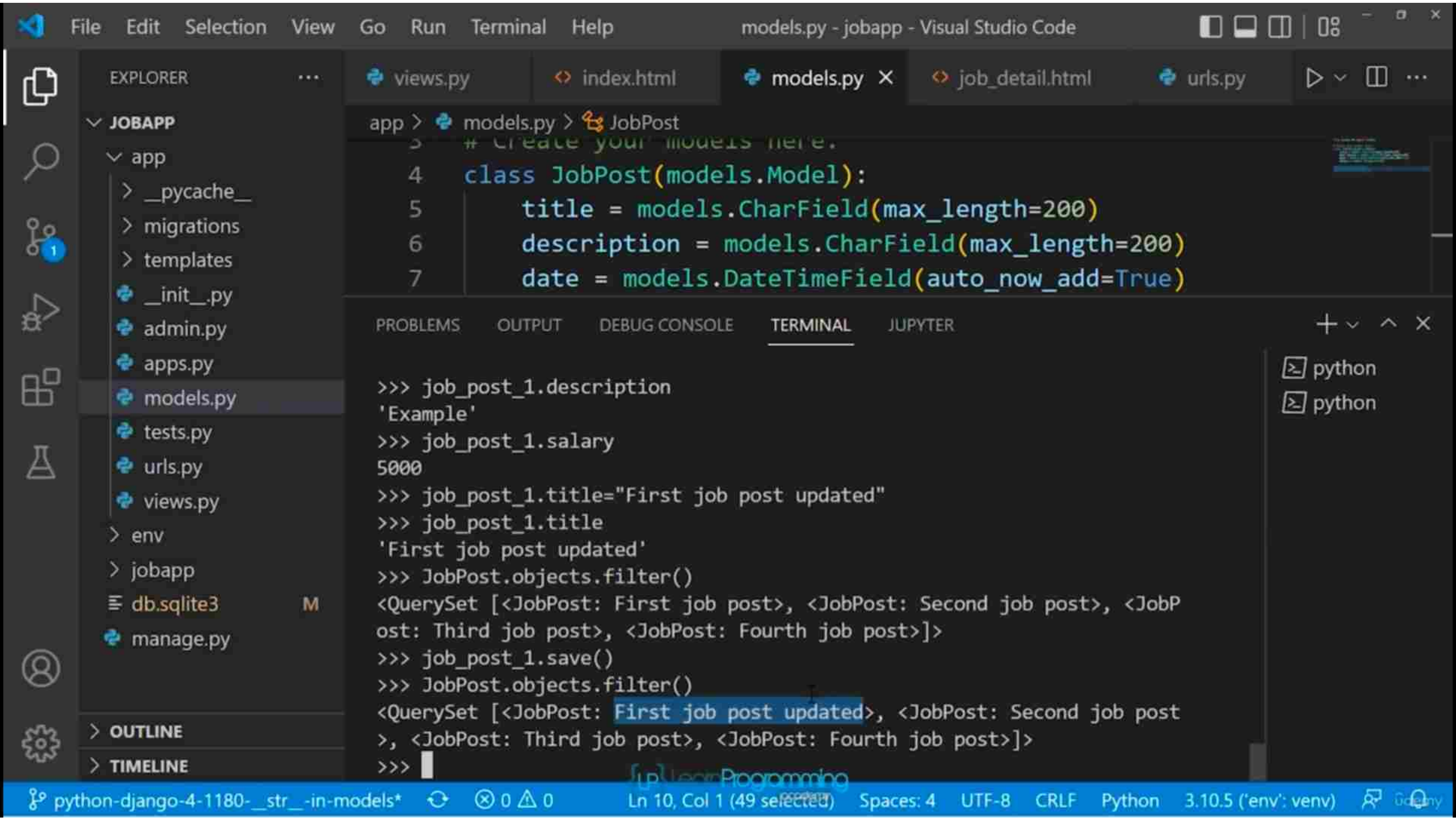
```
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'salary' is not defined
>>> JobPost.objects.filter(salary=1000)
<QuerySet [<JobPost: Third job post>, <JobPost: Fourth job post>]>
>>> JobPost.objects.filter(salary__gte=1000)
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Third job post>, <JobPost: Fourth job post>]>
>>> JobPost.objects.filter(salary__gt=1000)
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>]>
>>> JobPost.objects.filter(title__exact="Third job post")
<QuerySet [<JobPost: Third job post>]>
>>> JobPost.objects.filter(title="Third job post")
<QuerySet [<JobPost: Third job post>]>
>>>
```



File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
EXPLORER
JOBAPP
app
pycache
migrations
templates
init.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
env
jobapp
db.sqlite3 M
manage.py
OUTLINE
TIMELINE
views.py index.html models.py job_detail.html urls.py
app > models.py > JobPost
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
python
python-django-4-1180-__str__-in-models*
Ln 10, Col 1 (49 selected) Spaces: 4 UTF-8 CRLF Python 3.10.5 ('env': venv)

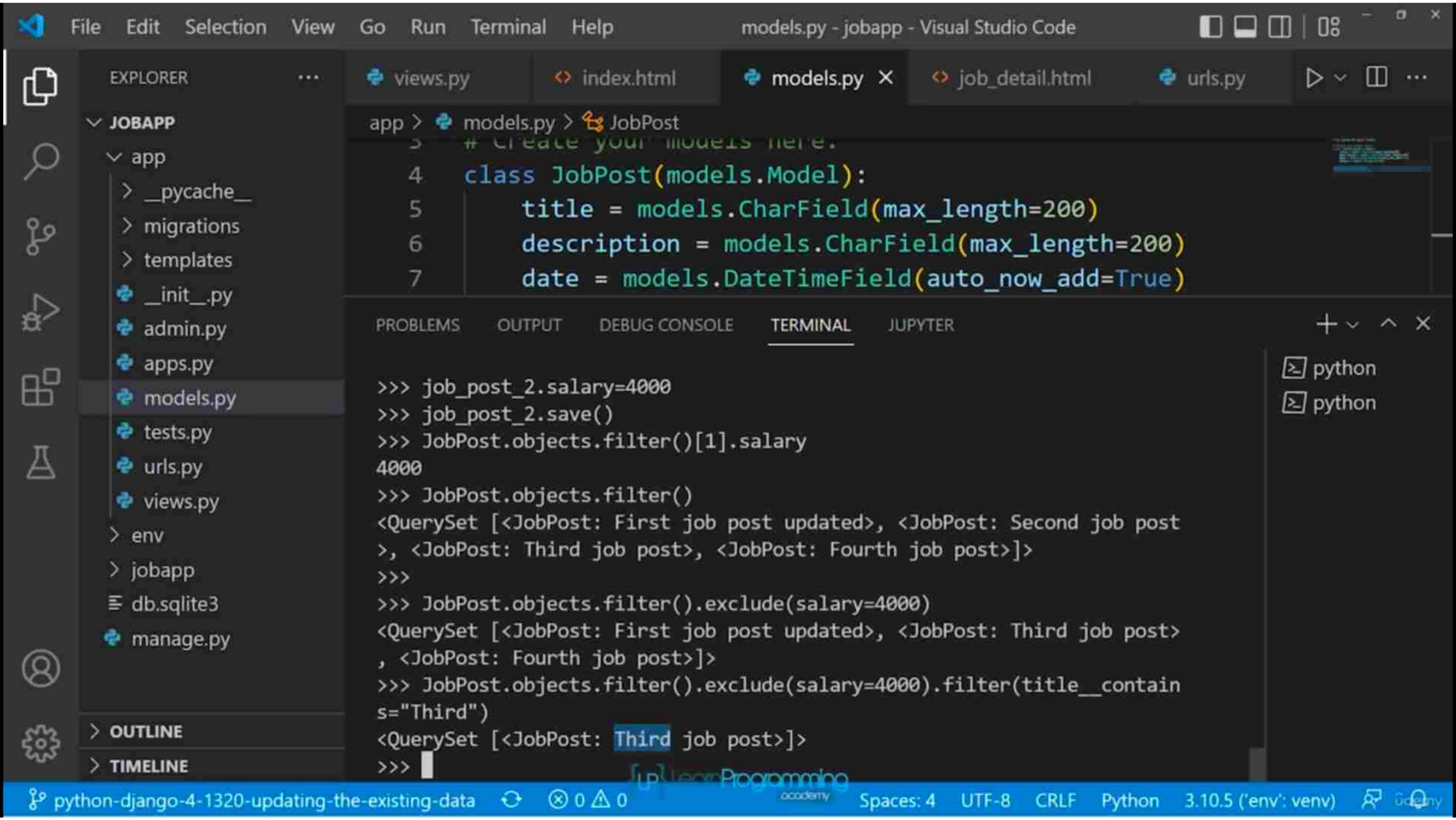
```python
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
```

```
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Third job post>, <JobPost: Fourth job post>]>
>>> JobPost.objects.filter(salary__gt=1000)
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>]>
>>> JobPost.objects.filter(title__exact="Third job post")
<QuerySet [<JobPost: Third job post>]>
>>> JobPost.objects.filter(title="Third job post")
<QuerySet [<JobPost: Third job post>]>
>>> JobPost.objects.filter(title__exact="third job post")
<QuerySet []>
>>> JobPost.objects.filter(title__iexact="third job post")
<QuerySet [<JobPost: Third job post>]>
>>> JobPost.objects.filter(title__iexact="job post")
<QuerySet []>
>>> 
```



```python
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
```

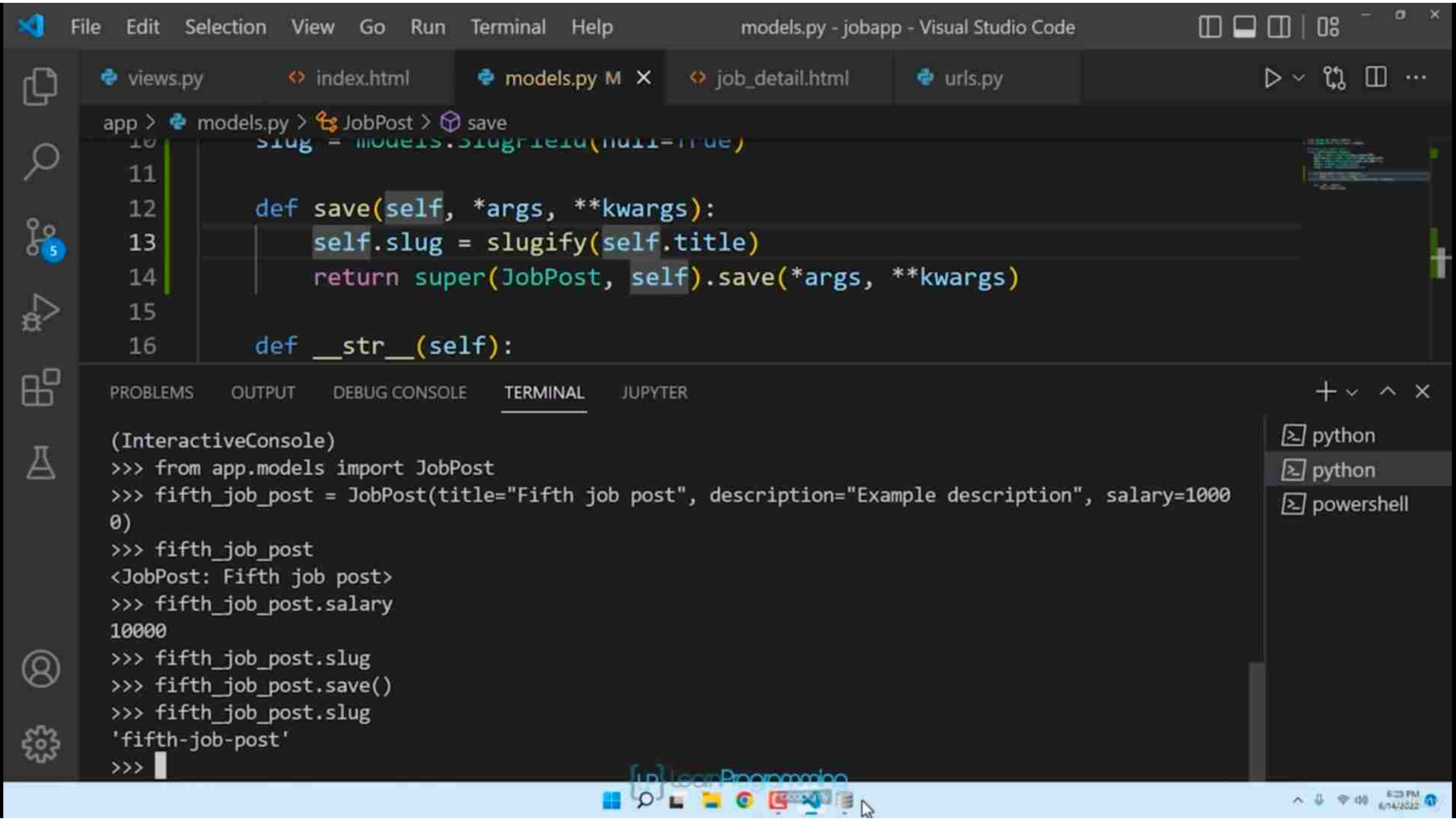
```
>>> job_post_1.description
'Example'
>>> job_post_1.salary
5000
>>> job_post_1.title="First job post updated"
>>> job_post_1.title
'First job post updated'
>>> JobPost.objects.filter()
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobP
ost: Third job post>, <JobPost: Fourth job post>]>
>>> job_post_1.save()
>>> JobPost.objects.filter()
<QuerySet [<JobPost: First job post updated>, <JobPost: Second job post
>, <JobPost: Third job post>, <JobPost: Fourth job post>]>
>>>
```



File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
EXPLORER
JOBAPP
app
pycache
migrations
templates
init.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
env
jobapp
db.sqlite3
manage.py
OUTLINE
TIMELINE
views.py
index.html
models.py
job_detail.html
urls.py
app > models.py > JobPost
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
python
python-django-4-1320-updating-the-existing-data
Spaces: 4 UTF-8 CRLF Python 3.10.5 ('env': venv)

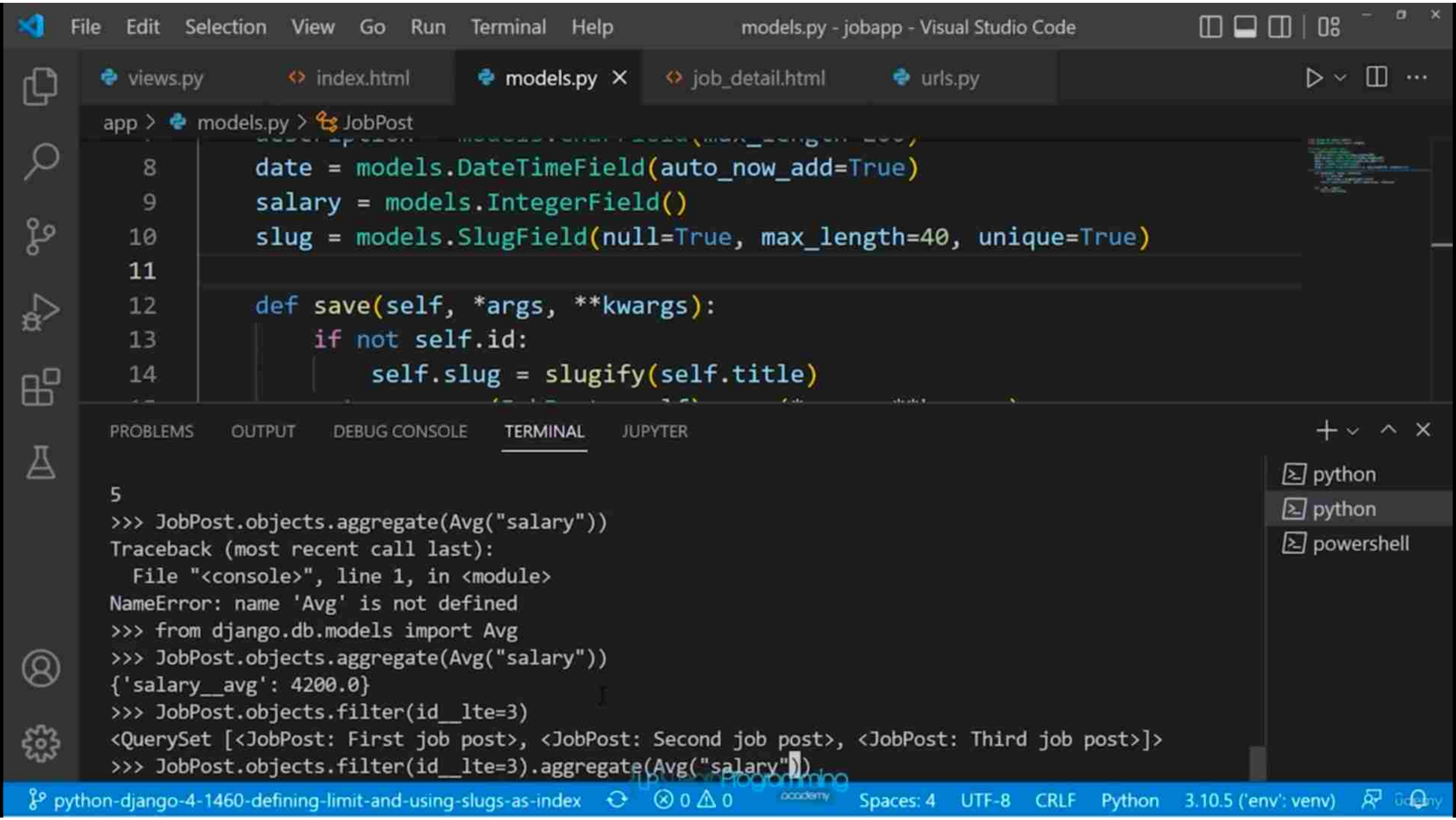
```python
# Create your models here.
class JobPost(models.Model):
    title = models.CharField(max_length=200)
    description = models.CharField(max_length=200)
    date = models.DateTimeField(auto_now_add=True)
```

```
>>> job_post_2.salary=4000
>>> job_post_2.save()
>>> JobPost.objects.filter()[1].salary
4000
>>> JobPost.objects.filter()
<QuerySet [<JobPost: First job post updated>, <JobPost: Second job post
>, <JobPost: Third job post>, <JobPost: Fourth job post>]>
>>>
>>> JobPost.objects.filter().exclude(salary=4000)
<QuerySet [<JobPost: First job post updated>, <JobPost: Third job post>
, <JobPost: Fourth job post>]>
>>> JobPost.objects.filter().exclude(salary=4000).filter(title__contain
s="Third")
<QuerySet [<JobPost: Third job post>]>
>>>
```


File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
views.py
index.html
models.py M
job_detail.html
urls.py
app > models.py > JobPost > save
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
python
powershell

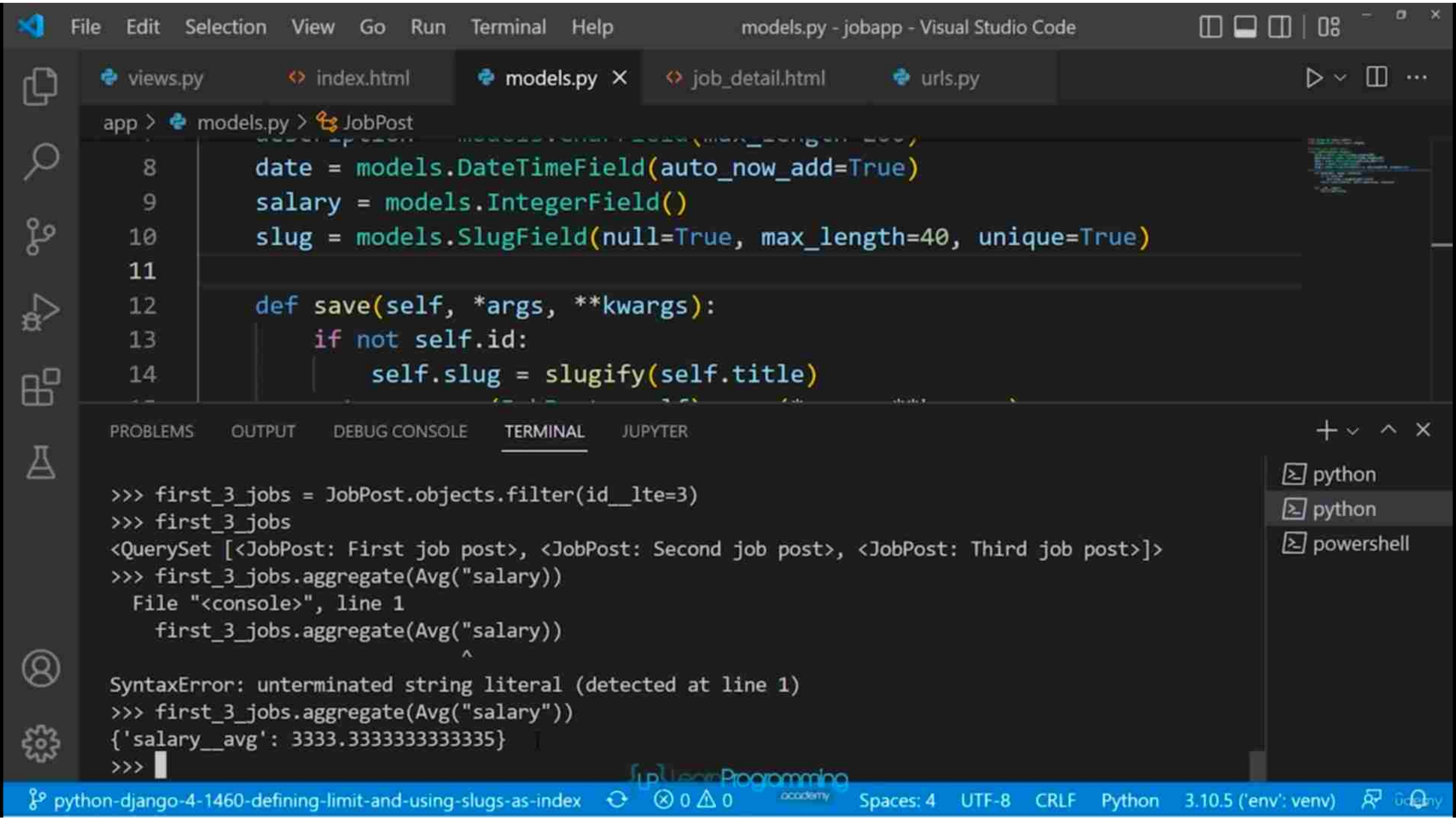
```python
    def save(self, *args, **kwargs):
        self.slug = slugify(self.title)
        return super(JobPost, self).save(*args, **kwargs)

    def __str__(self):
```

```
(InteractiveConsole)
>>> from app.models import JobPost
>>> fifth_job_post = JobPost(title="Fifth job post", description="Example description", salary=1000
0)
>>> fifth_job_post
<JobPost: Fifth job post>
>>> fifth_job_post.salary
10000
>>> fifth_job_post.slug
>>> fifth_job_post.save()
>>> fifth_job_post.slug
'fifth-job-post'
>>>
```


File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
views.py
index.html
models.py
job_detail.html
urls.py
app > models.py > JobPost
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
python
powershell
python-django-4-1460-defining-limit-and-using-slugs-as-index
Spaces: 4 UTF-8 CRLF Python 3.10.5 ('env': venv)

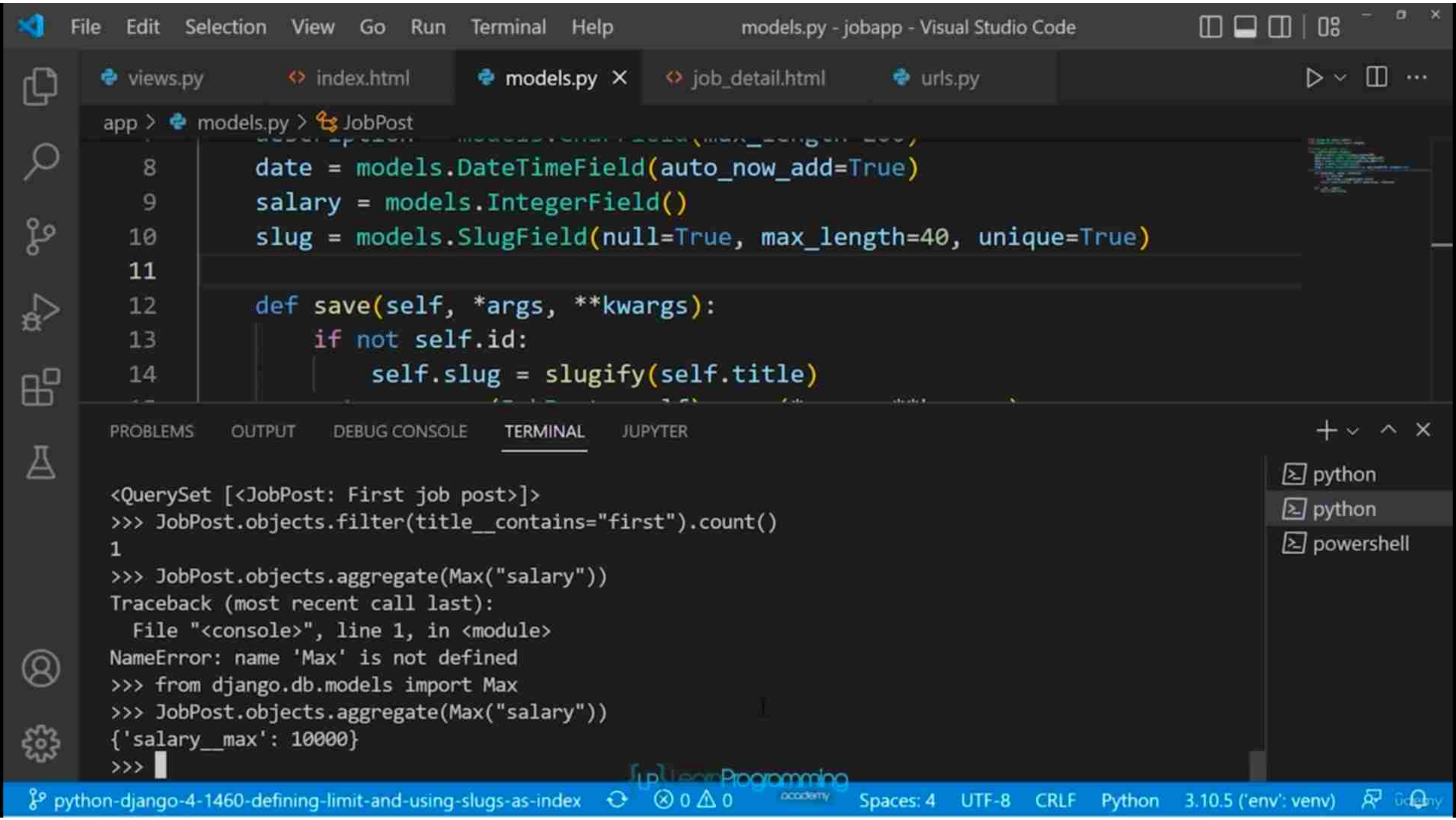
```python
    date = models.DateTimeField(auto_now_add=True)
    salary = models.IntegerField()
    slug = models.SlugField(null=True, max_length=40, unique=True)

    def save(self, *args, **kwargs):
        if not self.id:
            self.slug = slugify(self.title)
```

```
5
>>> JobPost.objects.aggregate(Avg("salary"))
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'Avg' is not defined
>>> from django.db.models import Avg
>>> JobPost.objects.aggregate(Avg("salary"))
{'salary__avg': 4200.0}
>>> JobPost.objects.filter(id__lte=3)
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobPost: Third job post>]>
>>> JobPost.objects.filter(id__lte=3).aggregate(Avg("salary"))
```


File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
views.py
index.html
models.py
job_detail.html
urls.py
app > models.py > JobPost
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
python
python
powershell
python-django-4-1460-defining-limit-and-using-slugs-as-index
Spaces: 4
UTF-8
CRLF
Python
3.10.5 ('env': venv)

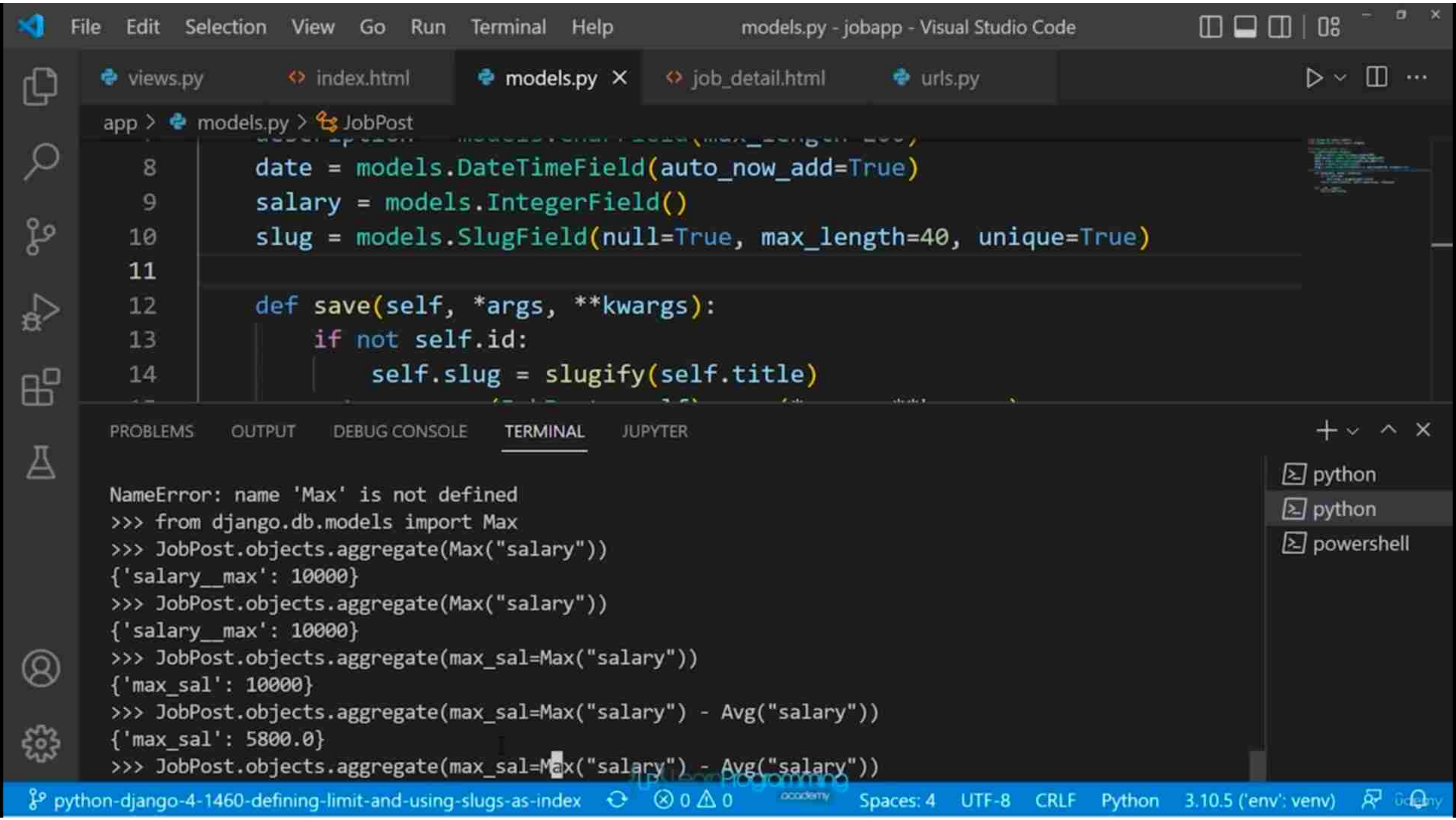
```python
    date = models.DateTimeField(auto_now_add=True)
    salary = models.IntegerField()
    slug = models.SlugField(null=True, max_length=40, unique=True)

    def save(self, *args, **kwargs):
        if not self.id:
            self.slug = slugify(self.title)
```

```
>>> first_3_jobs = JobPost.objects.filter(id__lte=3)
>>> first_3_jobs
<QuerySet [<JobPost: First job post>, <JobPost: Second job post>, <JobPost: Third job post>]>
>>> first_3_jobs.aggregate(Avg("salary))
  File "<console>", line 1
    first_3_jobs.aggregate(Avg("salary))
                               ^
SyntaxError: unterminated string literal (detected at line 1)
>>> first_3_jobs.aggregate(Avg("salary"))
{'salary__avg': 3333.3333333333335}
>>>
```


File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
views.py
index.html
models.py
job_detail.html
urls.py
app > models.py > JobPost
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
python
powershell
python-django-4-1460-defining-limit-and-using-slugs-as-index
Spaces: 4 UTF-8 CRLF Python 3.10.5 ('env': venv)

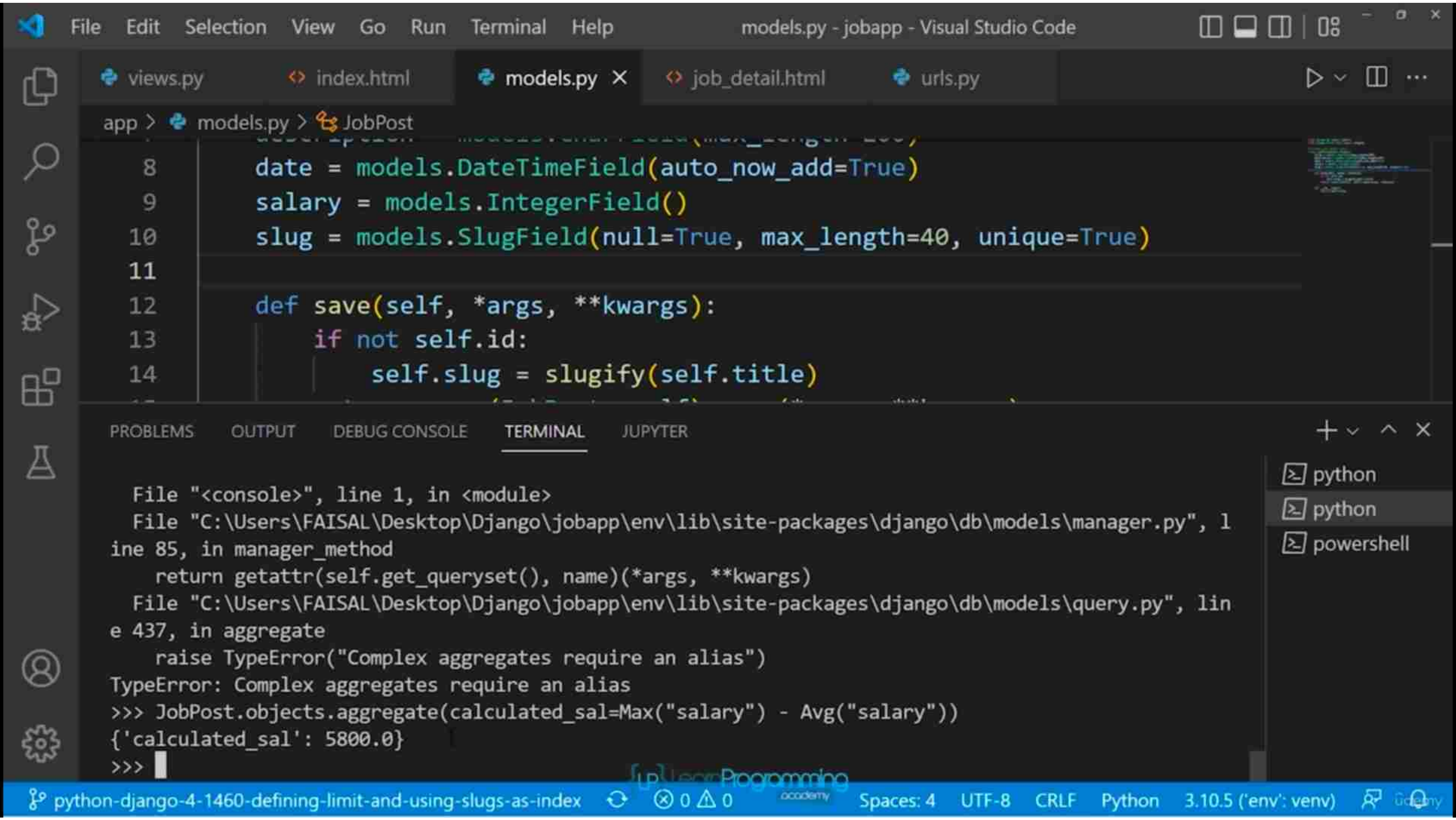
```python
    date = models.DateTimeField(auto_now_add=True)
    salary = models.IntegerField()
    slug = models.SlugField(null=True, max_length=40, unique=True)

    def save(self, *args, **kwargs):
        if not self.id:
            self.slug = slugify(self.title)
```

```
<QuerySet [<JobPost: First job post>]>
>>> JobPost.objects.filter(title__contains="first").count()
1
>>> JobPost.objects.aggregate(Max("salary"))
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'Max' is not defined
>>> from django.db.models import Max
>>> JobPost.objects.aggregate(Max("salary"))
{'salary__max': 10000}
>>>
```


File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
views.py
index.html
models.py
job_detail.html
urls.py
app > models.py > JobPost
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
python
python
powershell
python-django-4-1460-defining-limit-and-using-slugs-as-index
Spaces: 4
UTF-8
CRLF
Python
3.10.5 ('env': venv)

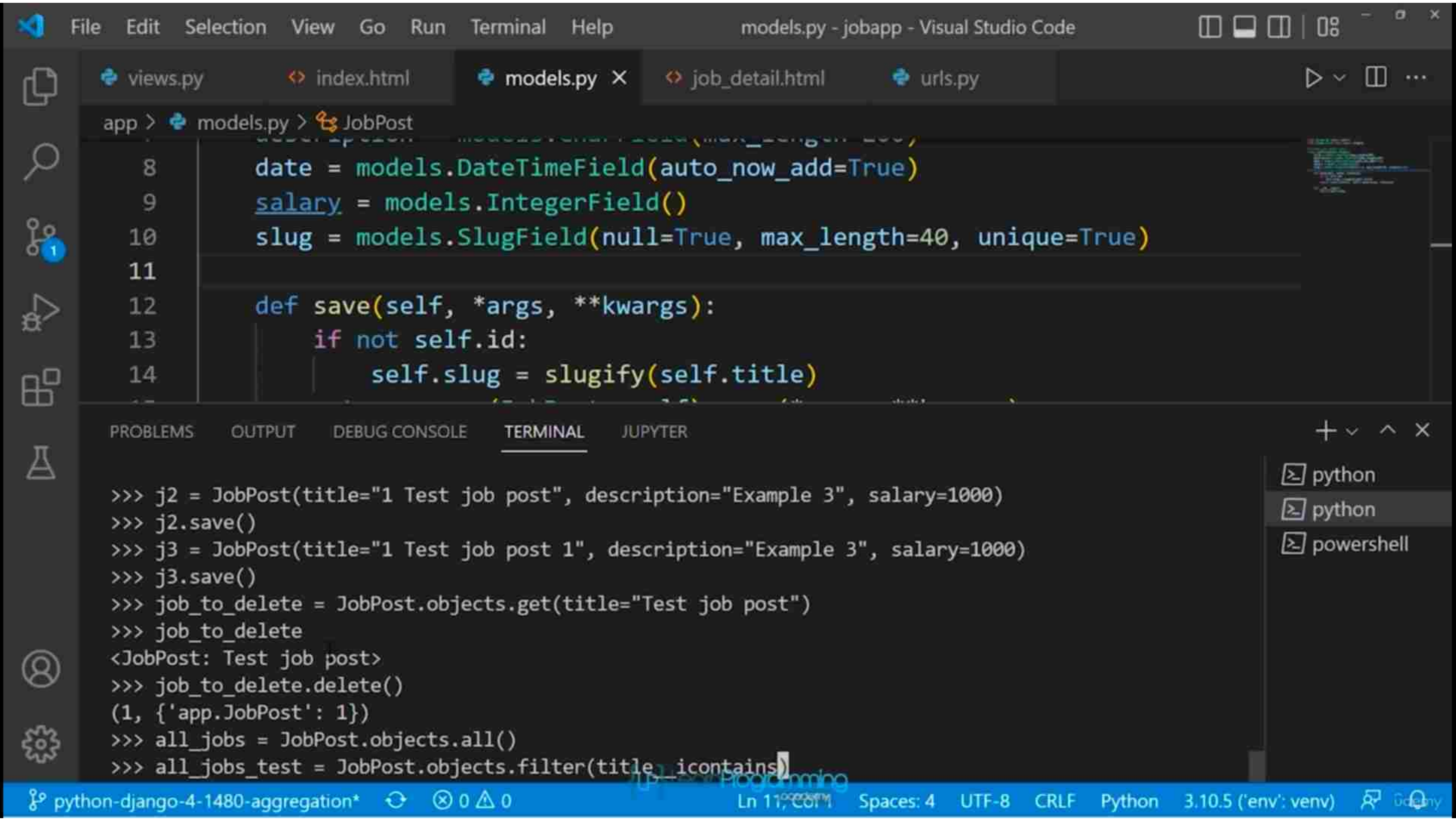
```python
    date = models.DateTimeField(auto_now_add=True)
    salary = models.IntegerField()
    slug = models.SlugField(null=True, max_length=40, unique=True)

    def save(self, *args, **kwargs):
        if not self.id:
            self.slug = slugify(self.title)
```

```
NameError: name 'Max' is not defined
>>> from django.db.models import Max
>>> JobPost.objects.aggregate(Max("salary"))
{'salary__max': 10000}
>>> JobPost.objects.aggregate(Max("salary"))
{'salary__max': 10000}
>>> JobPost.objects.aggregate(max_sal=Max("salary"))
{'max_sal': 10000}
>>> JobPost.objects.aggregate(max_sal=Max("salary") - Avg("salary"))
{'max_sal': 5800.0}
>>> JobPost.objects.aggregate(max_sal=Max("salary") - Avg("salary"))
```


File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
views.py
index.html
models.py
job_detail.html
urls.py
app > models.py > JobPost
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
python
python
powershell
python-django-4-1460-defining-limit-and-using-slugs-as-index
Spaces: 4
UTF-8
CRLF
Python
3.10.5 ('env': venv)

```python
    date = models.DateTimeField(auto_now_add=True)
    salary = models.IntegerField()
    slug = models.SlugField(null=True, max_length=40, unique=True)

    def save(self, *args, **kwargs):
        if not self.id:
            self.slug = slugify(self.title)
```

```
  File "<console>", line 1, in <module>
  File "C:\Users\FAISAL\Desktop\Django\jobapp\env\lib\site-packages\django\db\models\manager.py", l
ine 85, in manager_method
    return getattr(self.get_queryset(), name)(*args, **kwargs)
  File "C:\Users\FAISAL\Desktop\Django\jobapp\env\lib\site-packages\django\db\models\query.py", lin
e 437, in aggregate
    raise TypeError("Complex aggregates require an alias")
TypeError: Complex aggregates require an alias
>>> JobPost.objects.aggregate(calculated_sal=Max("salary") - Avg("salary"))
{'calculated_sal': 5800.0}
>>>
```


File Edit Selection View Go Run Terminal Help
models.py - jobapp - Visual Studio Code
views.py
index.html
models.py
job_detail.html
urls.py
app > models.py > JobPost
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
python
powershell
python-django-4-1480-aggregation*
Ln 11, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.5 ('env': venv)

```python
    date = models.DateTimeField(auto_now_add=True)
    salary = models.IntegerField()
    slug = models.SlugField(null=True, max_length=40, unique=True)

    def save(self, *args, **kwargs):
        if not self.id:
            self.slug = slugify(self.title)
```

```
>>> j2 = JobPost(title="1 Test job post", description="Example 3", salary=1000)
>>> j2.save()
>>> j3 = JobPost(title="1 Test job post 1", description="Example 3", salary=1000)
>>> j3.save()
>>> job_to_delete = JobPost.objects.get(title="Test job post")
>>> job_to_delete
<JobPost: Test job post>
>>> job_to_delete.delete()
(1, {'app.JobPost': 1})
>>> all_jobs = JobPost.objects.all()
>>> all_jobs_test = JobPost.objects.filter(title__icontains)
```