Lab 13

Advanced Search

Step 1: Initial Set Up

1. Go to www.github.com and log in to your GitHub account.

2. Go to Moodle and click on the link for Lab 12.

3. Once you have accepted the assignment you are asked to refresh the page, and you are then presented with a link to your repository for lab 12.

4. When you click on the repository link, you are taken to the repository where you see a starter project has already been added:

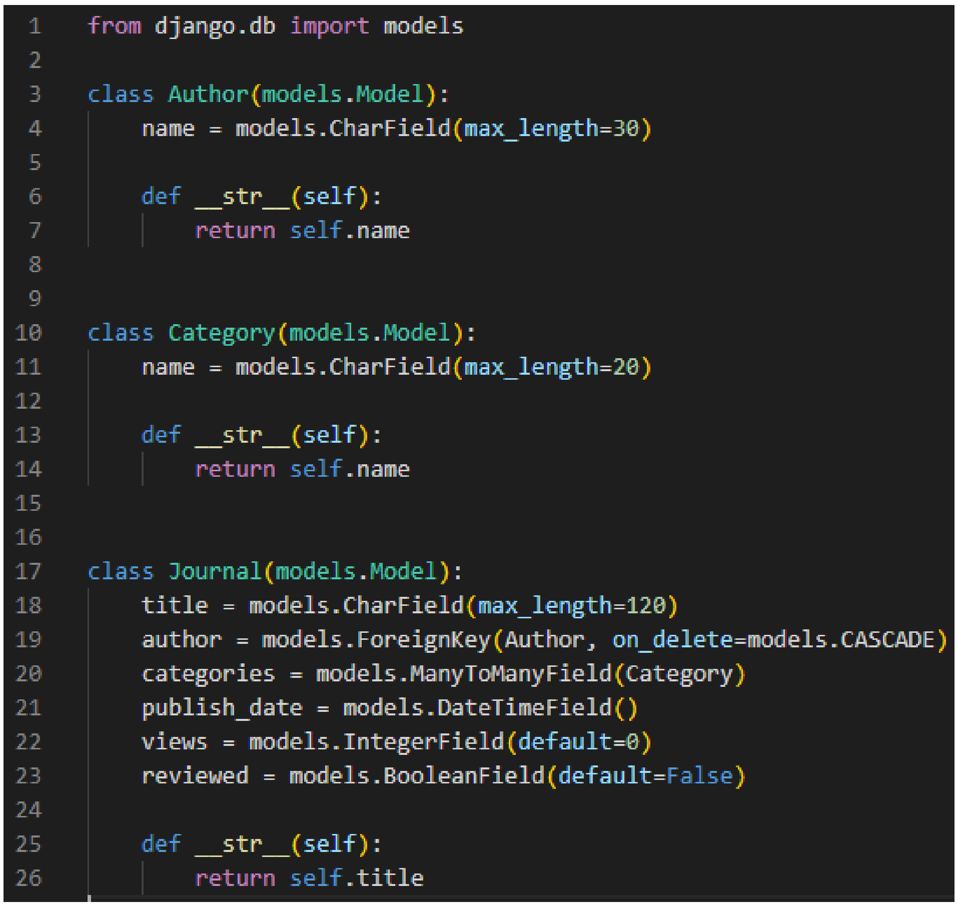
5. In Windows Command line make sure that you are in the djangoprojects folder and type the suitable command to activate the virtual environment:

6. Use the git clone command to clone the repo to your local computer:

7. Move into this lab-13-username folder using the cd command.

Step 2: Database Models

Open the project in VS Code and look at the database models in **models.py** as shown here:

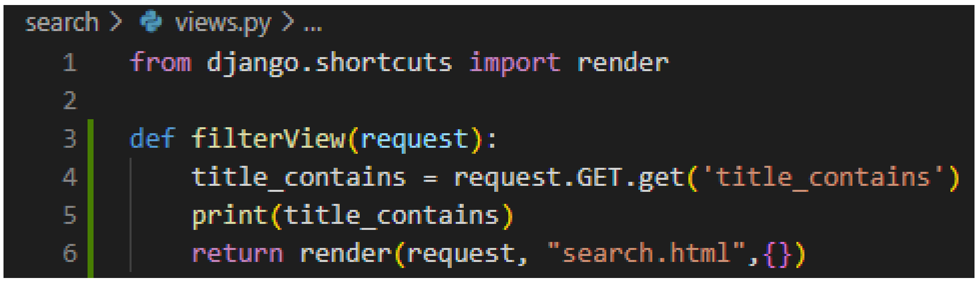


```python
from django.db import models

class Author(models.Model):
    name = models.CharField(max_length=30)

    def __str__(self):
        return self.name


class Category(models.Model):
    name = models.CharField(max_length=20)

    def __str__(self):
        return self.name


class Journal(models.Model):
    title = models.CharField(max_length=120)
    author = models.ForeignKey(Author, on_delete=models.CASCADE)
    categories = models.ManyToManyField(Category)
    publish_date = models.DateTimeField()
    views = models.IntegerField(default=0)
    reviewed = models.BooleanField(default=False)

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

There are three tables in the database and as you can see from the code above there is a One To Many relationship between Author and Journal and a Many To Many relationship between Journal and Category i.e. the same journal can feature in more than one category. The database has been populated with test data in order to show the advanced search working.

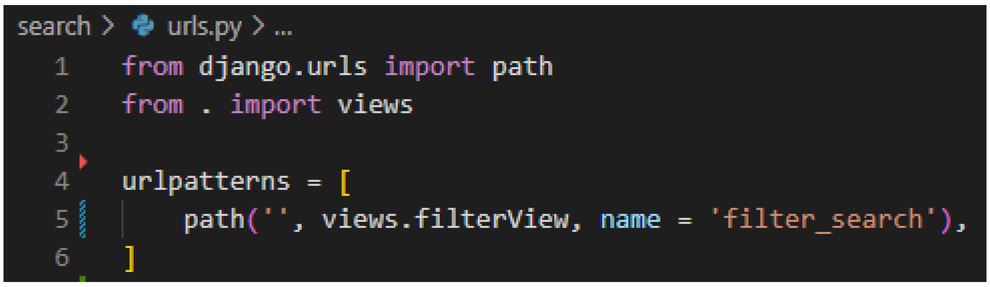
Step 3: Test out the Form Widgets

Open **search/views.py** and add in the following code:



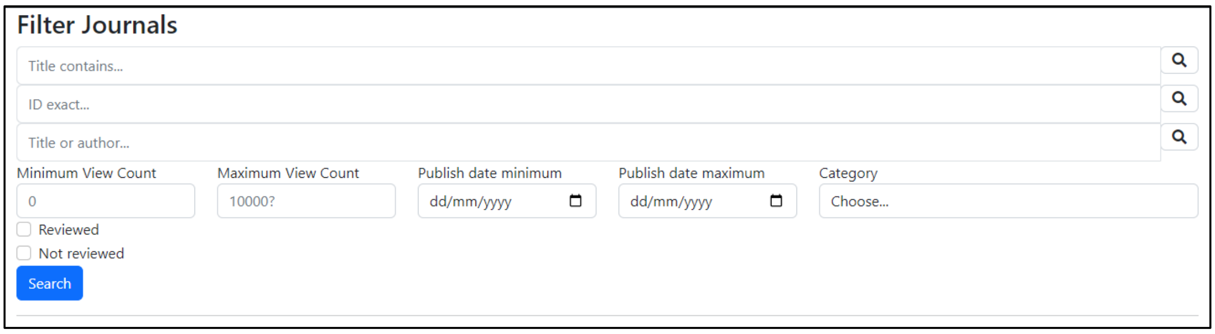
```
search > views.py > ...
from django.shortcuts import render

def filterView(request):
    title_contains = request.GET.get('title_contains')
    print(title_contains)
    return render(request, "search.html",{})
```

Create a new **urls.py** file inside the **search** app and add in the following code:



```
search > urls.py > ...
from django.urls import path
from . import views

urlpatterns = [
    path('', views.filterView, name = 'filter_search'),
]
```

Run the server and open the home page at http://localhost:8080/. The search page is loaded and should look like the following:



Filter Journals

Title contains...

ID exact...

Title or author...

Minimum View Count	Maximum View Count	Publish date minimum	Publish date maximum	Category
0	10000?	dd/mm/yyyy	dd/mm/yyyy	Choose...

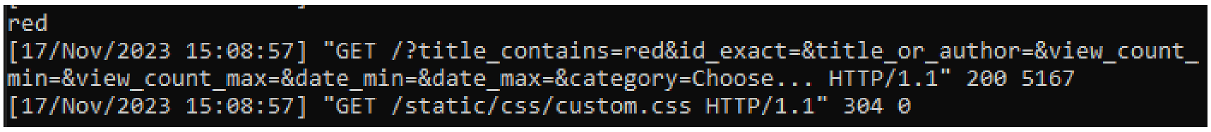
☐ Reviewed

☐ Not reviewed

Search

The code that we have just written in **views.py** is getting the data from the first input box in the form which is called **title_contains**. Take a look at the template **search.html** in VS Code and you will see that all the form widgets have been assigned names.

Type the search term **red** into the first input box and press the **Search** button. Check the output in the console and you will see that the value red has been printed out as shown below:



```
red
[17/Nov/2023 15:08:57] "GET /?title_contains=red&id_exact=&title_or_author=&view_count_
min=&view_count_max=&date_min=&date_max=&category=Choose... HTTP/1.1" 200 5167
[17/Nov/2023 15:08:57] "GET /static/css/custom.css HTTP/1.1" 304 0
```

If you look in the address bar of the browser, you will also see that the search term has been appended on to the url as shown below. The other input boxes have no values as yet:

http://localhost:8080/?title_contains=red&id_exact=&title_or_author=&view_count_min=&view_count_max=&date_min=&date_max=&category=Choose...

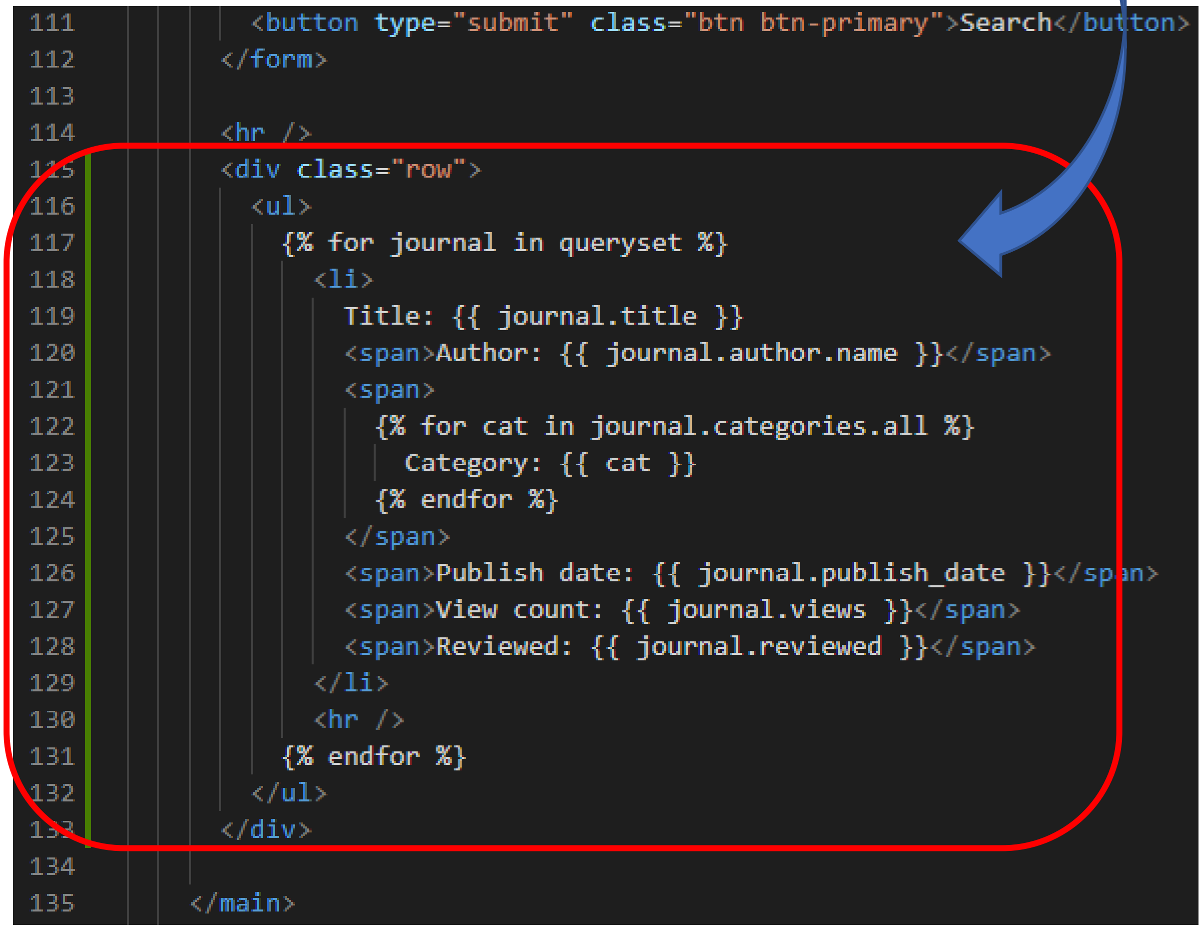
Step 4: Filter by Title

In **views.py** delete the first three lines of code from the function and copy-paste the following code. Using this code, we can search based on the **title** of the journal.

```python
from django.shortcuts import render
from .models import Journal

def filterView(request):
    qs = Journal.objects.all()
    title_contains_query = request.GET.get('title_contains')

    if title_contains_query != '' and title_contains_query is not None:
        qs = qs.filter(title__icontains=title_contains_query)

    context = {
        'queryset':qs,
    }
    return render(request, 'search.html', context)
```

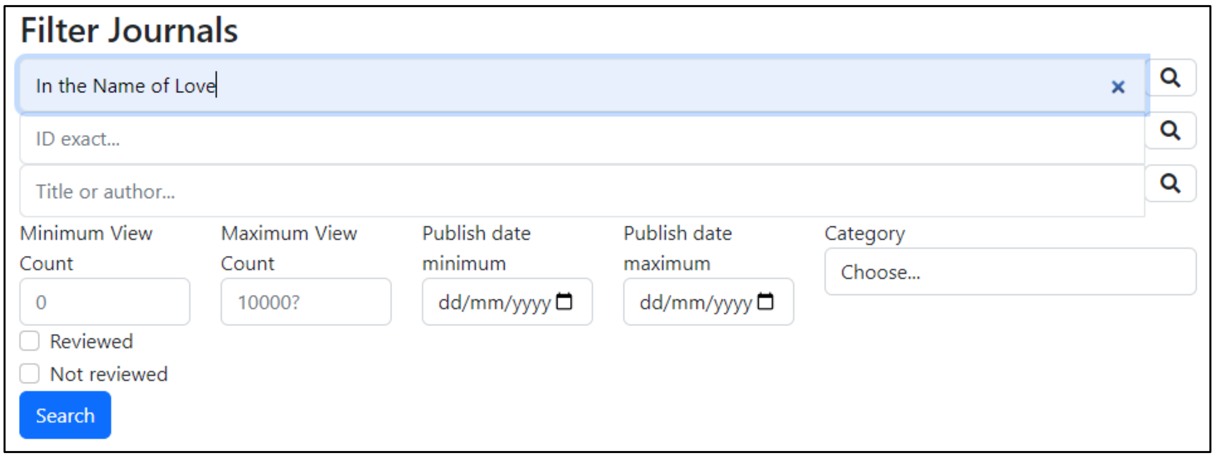
The search results will be displayed on the same web page just below the search form. Open the **search.html** page and copy-paste the following code to the location shown below:

```
<div class="row">
  <ul>
    {% for journal in queryset %}
      <li>
        Title: {{ journal.title }}
        <span>Author: {{ journal.author.name }}</span>
        <span>
          {% for cat in journal.categories.all %}
            Category: {{ cat }}
          {% endfor %}
        </span>
        <span>Publish date: {{ journal.publish_date }}</span>
        <span>View count: {{ journal.views }}</span>
        <span>Reviewed: {{ journal.reviewed }}</span>
      </li>
      <hr />
    {% endfor %}
  </ul>
</div>
```



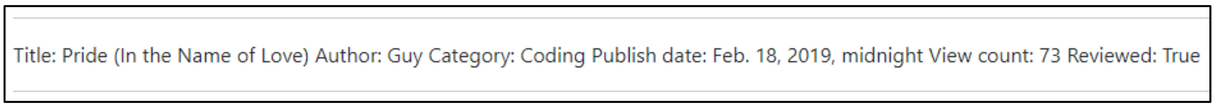
```
          <button type="submit" class="btn btn-primary">Search</button>
        </form>

        <hr />
        <div class="row">
          <ul>
            {% for journal in queryset %}
              <li>
                Title: {{ journal.title }}
                <span>Author: {{ journal.author.name }}</span>
                <span>
                  {% for cat in journal.categories.all %}
                    Category: {{ cat }}
                  {% endfor %}
                </span>
                <span>Publish date: {{ journal.publish_date }}</span>
                <span>View count: {{ journal.views }}</span>
                <span>Reviewed: {{ journal.reviewed }}</span>
              </li>
              <hr />
            {% endfor %}
          </ul>
        </div>

      </main>
```

Run the server and type **In the Name of Love** into the first text box as shown below:


Filter Journals

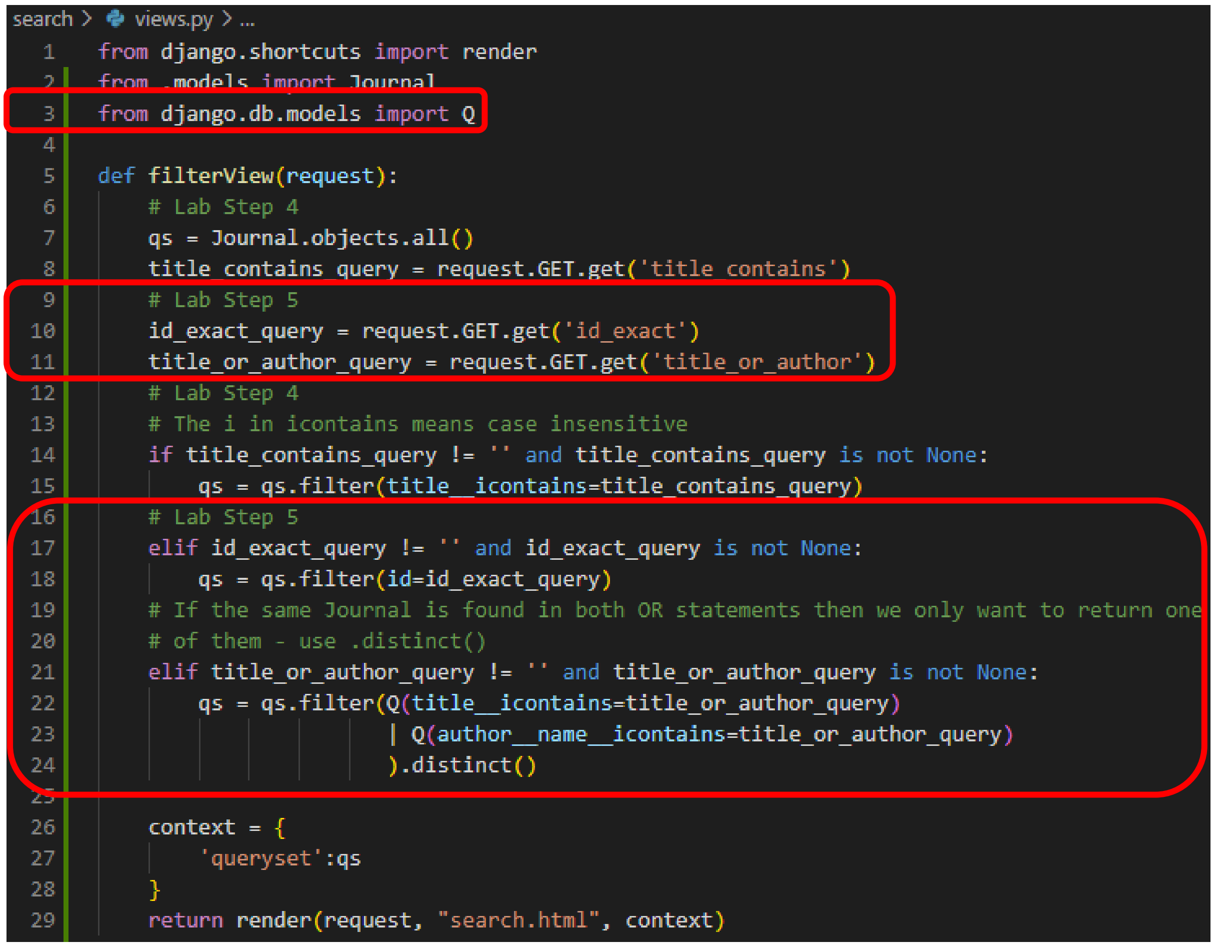
```
In the Name of Love
ID exact...
Title or author...
Minimum View Count: 0
Maximum View Count: 10000?
Publish date minimum: dd/mm/yyyy
Publish date maximum: dd/mm/yyyy
Category: Choose...
Reviewed
Not reviewed
Search
```

When you click the **Search** button, just one Journal entry is displayed in the results:


Title: Pride (In the Name of Love) Author: Guy Category: Coding Publish date: Feb. 18, 2019, midnight View count: 73 Reviewed: True

Step 5: Filter by Id & Title or Author

In **views.py** add the following code to filter the Journal entries based on the id & Title or Author.

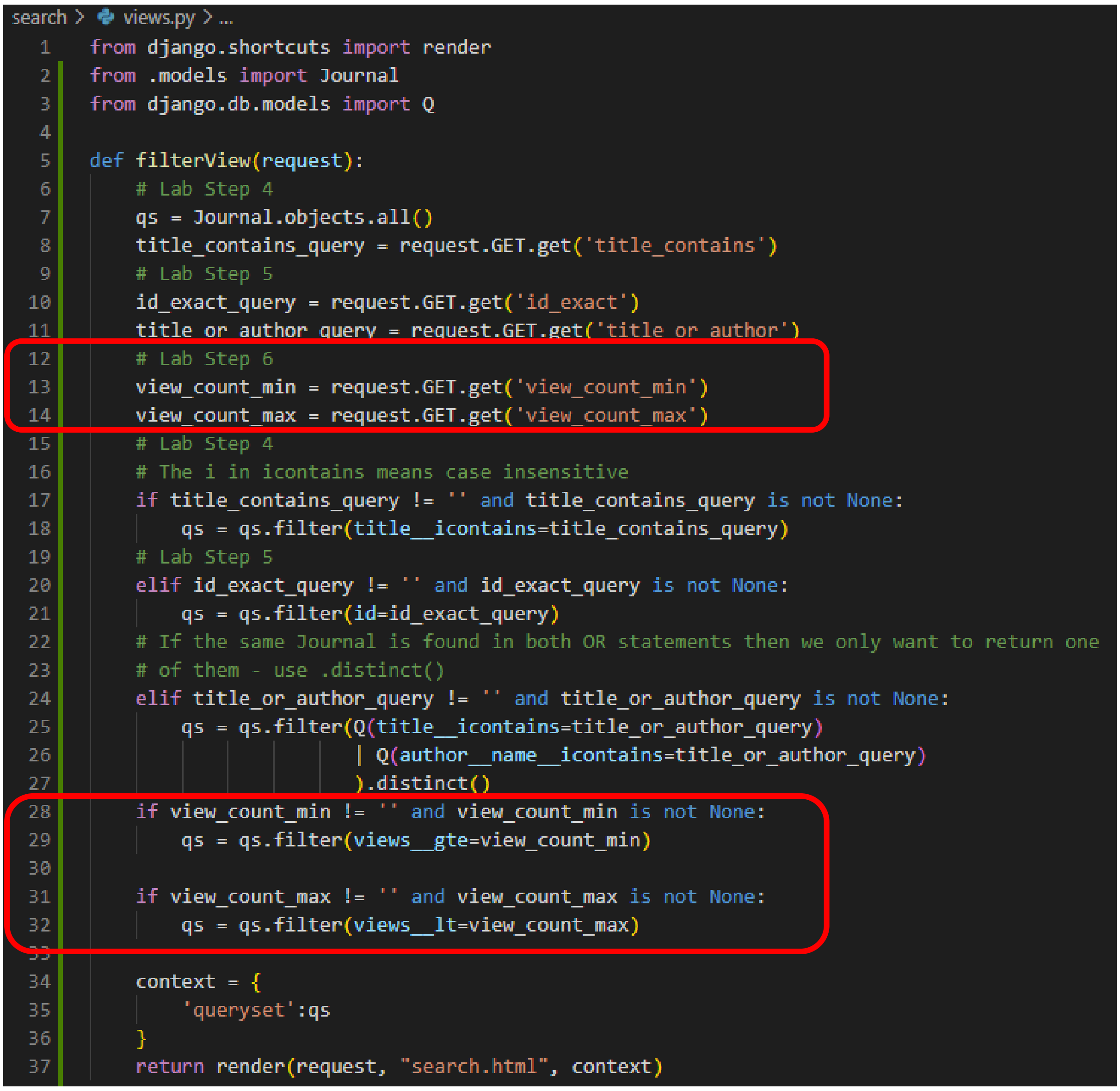

search > views.py > ...

```python
from django.shortcuts import render
from .models import Journal
from django.db.models import Q

def filterView(request):
    # Lab Step 4
    qs = Journal.objects.all()
    title_contains_query = request.GET.get('title_contains')
    # Lab Step 5
    id_exact_query = request.GET.get('id_exact')
    title_or_author_query = request.GET.get('title_or_author')
    # Lab Step 4
    # The i in icontains means case insensitive
    if title_contains_query != '' and title_contains_query is not None:
        qs = qs.filter(title__icontains=title_contains_query)
    # Lab Step 5
    elif id_exact_query != '' and id_exact_query is not None:
        qs = qs.filter(id=id_exact_query)
    # If the same Journal is found in both OR statements then we only want to return one
    # of them - use .distinct()
    elif title_or_author_query != '' and title_or_author_query is not None:
        qs = qs.filter(Q(title__icontains=title_or_author_query)
                       | Q(author__name__icontains=title_or_author_query)
                       ).distinct()

    context = {
        'queryset':qs
    }
    return render(request, "search.html", context)
```

When you run the server, you can now do a search on the title, the id or the title or author (Open the database in DB Browser and look at the data in the tables). The code for these 3 filters is placed in an if-elif block because only one of the filters can be applied so the first if statement to evaluate to true will be executed.

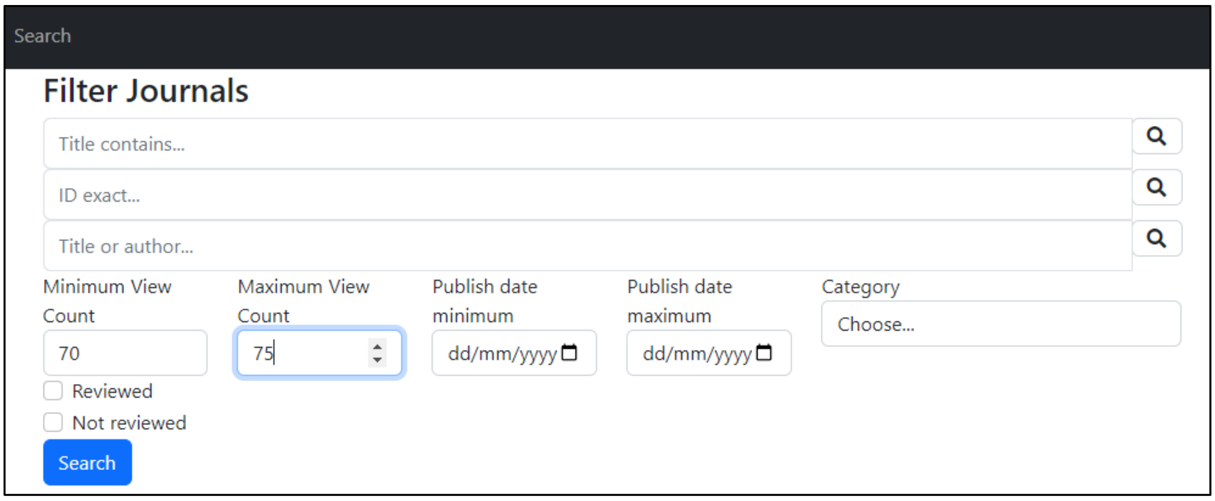
Step 6: Filter by View Count

In **views.py** add the following code to filter the Journal entries based on the view count



```python
from django.shortcuts import render
from .models import Journal
from django.db.models import Q

def filterView(request):
    # Lab Step 4
    qs = Journal.objects.all()
    title_contains_query = request.GET.get('title_contains')
    # Lab Step 5
    id_exact_query = request.GET.get('id_exact')
    title_or_author_query = request.GET.get('title_or_author')
    # Lab Step 6
    view_count_min = request.GET.get('view_count_min')
    view_count_max = request.GET.get('view_count_max')
    # Lab Step 4
    # The i in icontains means case insensitive
    if title_contains_query != '' and title_contains_query is not None:
        qs = qs.filter(title__icontains=title_contains_query)
    # Lab Step 5
    elif id_exact_query != '' and id_exact_query is not None:
        qs = qs.filter(id=id_exact_query)
    # If the same Journal is found in both OR statements then we only want to return one
    # of them - use .distinct()
    elif title_or_author_query != '' and title_or_author_query is not None:
        qs = qs.filter(Q(title__icontains=title_or_author_query)
                       | Q(author__name__icontains=title_or_author_query)
                       ).distinct()
    if view_count_min != '' and view_count_min is not None:
        qs = qs.filter(views__gte=view_count_min)

    if view_count_max != '' and view_count_max is not None:
        qs = qs.filter(views__lt=view_count_max)

    context = {
        'queryset':qs
    }
    return render(request, "search.html", context)
```

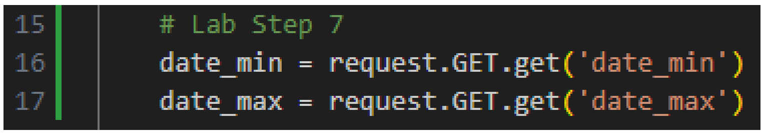
When you run the server, you can now do a search on the view count. In the output here the values 70 and 75 were used which gives just 15 results. We used two separate if statements used for the min and max count which means that both get executed.



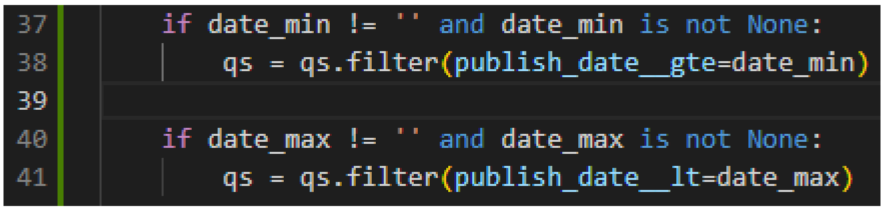
Minimum View Count: 70
Maximum View Count: 75
Publish date minimum: dd/mm/yyyy
Publish date maximum: dd/mm/yyyy

Step 7: Filter by Date & View Count

In **views.py** add the following code to filter the Journal entries based on the publish date. These will be written in separate if statements which means that we can filter by data and view count.

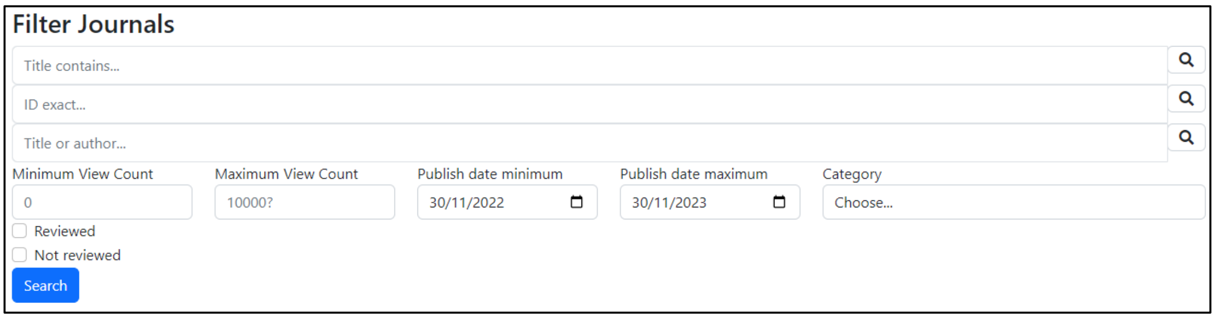


```python
    # Lab Step 7
    date_min = request.GET.get('date_min')
    date_max = request.GET.get('date_max')
```



```python
    if date_min != '' and date_min is not None:
        qs = qs.filter(publish_date__gte=date_min)

    if date_max != '' and date_max is not None:
        qs = qs.filter(publish_date__lt=date_max)
```

Run the server, and do a search using values for the dates only as shown below:



Minimum View Count: 0
Maximum View Count: 10000?
Publish date minimum: 30/11/2022
Publish date maximum: 30/11/2023

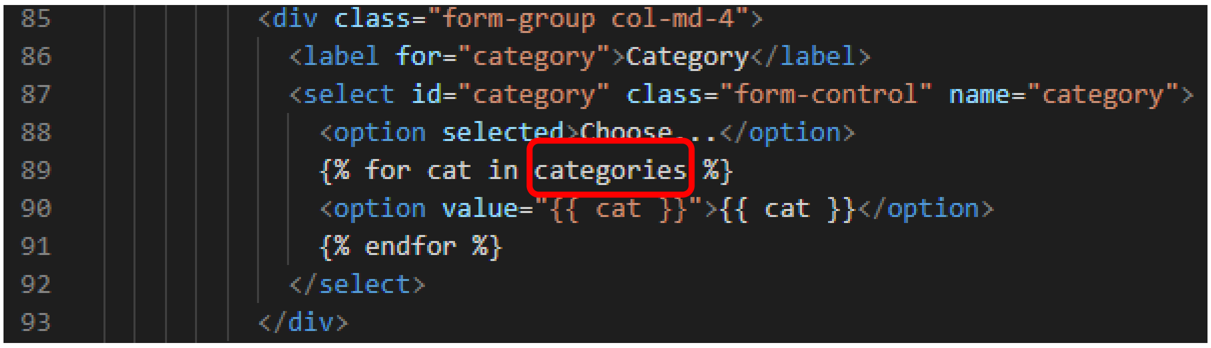
We get four results based on the dates entered.

Now use the same dates and also specify values for the min and max view count e.g., 10 for min and 30 for max. You will see a smaller set of results (2 Journal entries) as we have narrowed our search

Step 8: Filter by Category

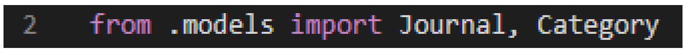
If you look at the form on the website, the list of categories is empty.

The code is already in the template to populate the list, but it doesn’t work because we have not yet passed categories into the template using the context

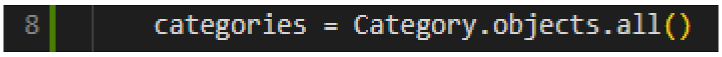


```
<div class="form-group col-md-4">
  <label for="category">Category</label>
  <select id="category" class="form-control" name="category">
    <option selected>Choose...</option>
    {% for cat in categories %}
    <option value="{{ cat }}">{{ cat }}</option>
    {% endfor %}
  </select>
</div>
```

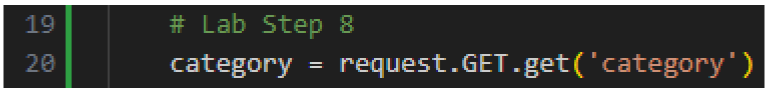
In **views.py** add the following code:



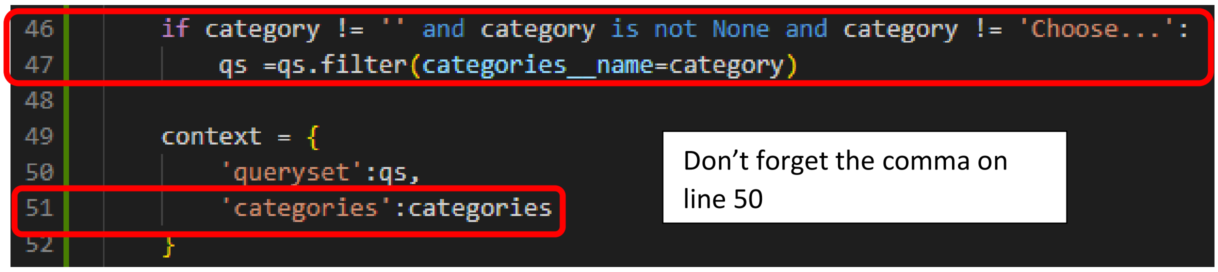
```
2   from .models import Journal, Category
```



```
8       categories = Category.objects.all()
```



```
19      # Lab Step 8
20      category = request.GET.get('category')
```



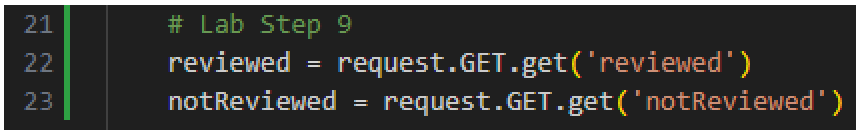
```
46      if category != '' and category is not None and category != 'Choose...':
47          qs =qs.filter(categories__name=category)
48
49      context = {
50          'queryset':qs,
51          'categories':categories
52      }
```

Don’t forget the comma on line 50

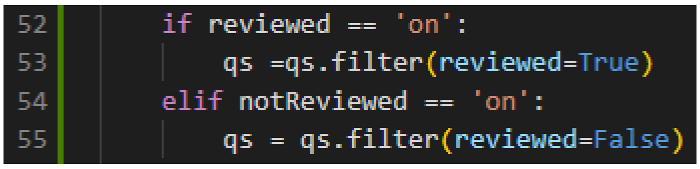
Run the server, and do a search based on the category. You can also narrow the search by using the view count and the publish date.

Step 9: Filter By Reviewed (CheckBox)

In **views.py** add the following code:



```python
    # Lab Step 9
    reviewed = request.GET.get('reviewed')
    notReviewed = request.GET.get('notReviewed')
```



```python
    if reviewed == 'on':
        qs =qs.filter(reviewed=True)
    elif notReviewed == 'on':
        qs = qs.filter(reviewed=False)
```

Run the server and tick the Reviewed checkbox and note the results and then tick the Not Reviewed checkbox and note the results.

Stop the server and run the following git commands to update the local and remote repositories:

git add -A

git commit -m “lab 13 commit”

git push -u origin main