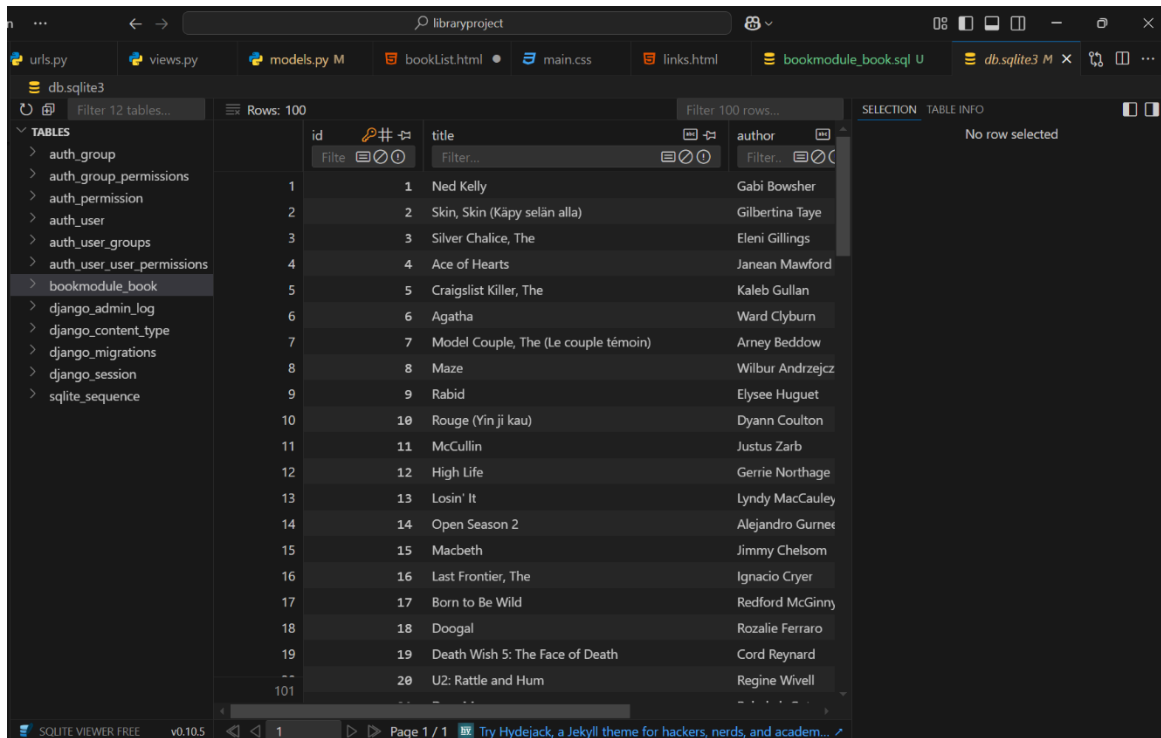


Task 2:



The screenshot shows a SQLite database viewer with a table named 'bookmodule_book'. The table contains 20 rows of book data. The columns are id, title, and author. The data is as follows:

id	title	author
1	Ned Kelly	Gabi Bowsher
2	Skin, Skin (Käpy selän alla)	Gilbertina Tåye
3	Silver Chalice, The	Eleni Gillings
4	Ace of Hearts	Janean Mawford
5	Craigslist Killer, The	Kaleb Gullan
6	Agatha	Ward Clyburn
7	Model Couple, The (Le couple témoin)	Arney Beddow
8	Maze	Wilbur Andrzejcz
9	Rabid	Elysee Huguet
10	Rouge (Yin ji kau)	Dyann Coulton
11	McCullin	Justus Zarb
12	High Life	Gerrie Northage
13	Losin' It	Lyndy MacCauley
14	Open Season 2	Alejandro Gurnes
15	Macbeth	Jimmy Chelsom
16	Last Frontier, The	Ignacio Cryer
17	Born to Be Wild	Redford McGinny
18	Doogal	Rozalie Ferraro
19	Death Wish 5: The Face of Death	Cord Reynard
20	U2: Rattle and Hum	Regine Wivell

Step 4:

```
def __getBooksList():  
    book1 = Book(title = 'Continuous Delivery', author = 'J.Humble and D. Farley',price=120, edition = 3)  
    book1.save()  
    book2 = Book(title = 'Reversing: Secrets of Reverse Engineer', author = 'E.Eilam',price=97, edition = 2)  
    book2.save()  
    book3 = Book(title = 'The Hundred-Page Machine Learning Book', author = 'Andriy Burkov',price=100, edition = 4)  
    book3.save()  
    return [book1, book2, book3]
```

Task 3:

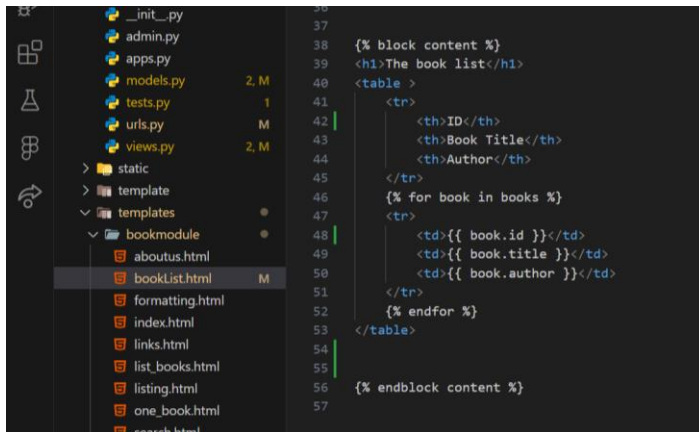
Step 1:

Step 2:



```
def simple_query(request):  
    mybooks=Book.objects.filter(title__icontains='and') # <- multiple objects  
    return render(request, 'bookmodule/bookList.html', {'books':mybooks})
```

Step 3:



The screenshot shows a file explorer on the left with a tree view of the project files. The 'templates' directory is expanded, showing 'bookmodule' and 'bookList.html'. The 'bookList.html' file is selected. The main editor shows the content of 'bookList.html', which is a Django template for displaying a list of books. It includes a table with columns for ID, Book Title, and Author, and a loop to iterate over a list of books.

```
37 {% block content %}
38 <h1>The book list</h1>
39 <table>
40 <tr>
41 <th>ID</th>
42 <th>Book Title</th>
43 <th>Author</th>
44 </tr>
45 <tr>
46 <td>{{ book.id }}</td>
47 <td>{{ book.title }}</td>
48 <td>{{ book.author }}</td>
49 </tr>
50 <tr>
51 <td>{{ book.id }}</td>
52 <td>{{ book.title }}</td>
53 <td>{{ book.author }}</td>
54 </tr>
55 <tr>
56 <td>{{ book.id }}</td>
57 <td>{{ book.title }}</td>
58 <td>{{ book.author }}</td>
59 </tr>
60 </table>
61 {% endblock content %}
```

Step 4:



The book list

ID	Book Title	Author
20	U2: Rattle and Hum	Regine Wivell
64	Friends and Family	Karlen Bleasdale
93	Tarzan and His Mate	Cassandra Kimmings

Task 4:

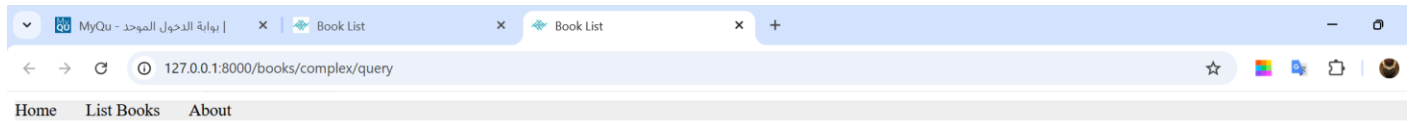
Step 1:



The screenshot shows a Django view function named 'complex_query' in a file named 'views.py'. The function takes a 'request' object as an argument. It filters a queryset of books based on several criteria: 'author_isnull' is False, 'title' contains 'and', 'edition' is greater than or equal to 2, and 'price' is less than or equal to 100. It then returns a render call to 'bookmodule/bookList.html' with the filtered books as context.

```
84 def complex_query(request):
85     mybooks=books=Book.objects.filter(author_isnull = False).filter(title__icontains='and').filter(edition__gte = 2).exclude(price__lte = 100)[:10]
86     if len(mybooks)>=1:
87         return render(request, 'bookmodule/bookList.html', {'books':mybooks})
88     else:
89         return render(request, 'bookmodule/index.html')
```

Step 2:



The book list

ID	Book Title	Author
20	U2: Rattle and Hum	Regine Wivell
64	Friends and Family	Karlen Bleasdale
93	Tarzan and His Mate	Cassandra Kimmings