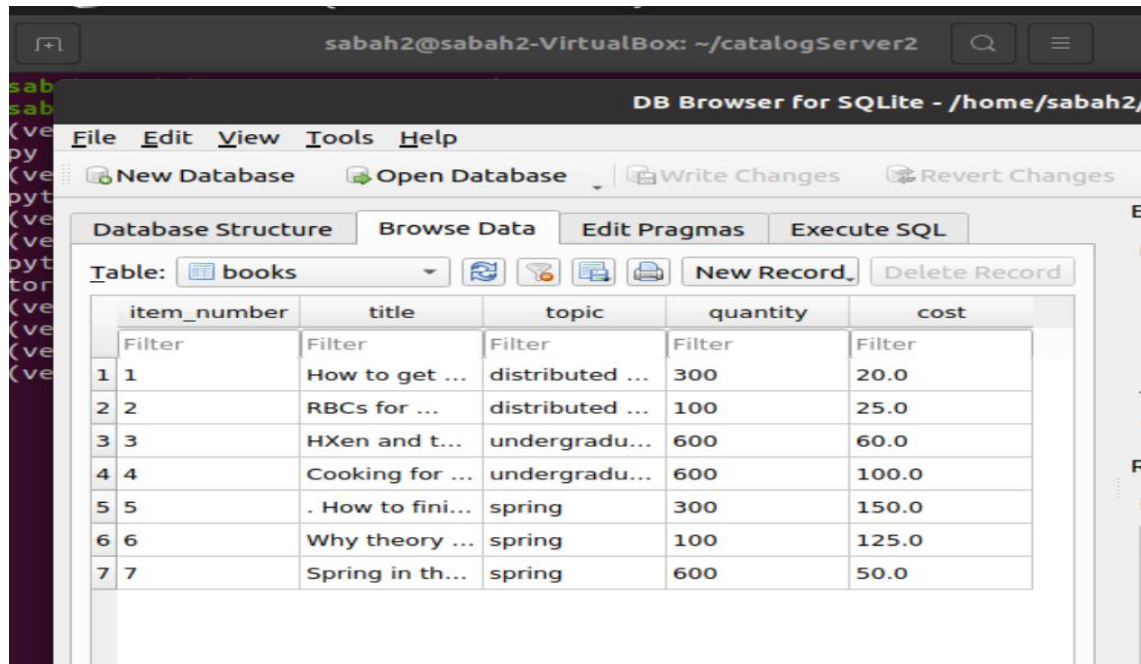


DOS PART2 PROJECT:

Database after inserting new books:



The screenshot shows the DB Browser for SQLite application. The 'Browse Data' tab is selected, displaying a table named 'books'. The table has five columns: 'item_number', 'title', 'topic', 'quantity', and 'cost'. There are 7 records in the table. The interface includes a menu bar (File, Edit, View, Tools, Help), a toolbar with buttons for 'New Database', 'Open Database', 'Write Changes', and 'Revert Changes', and a sidebar with tabs for 'Database Structure', 'Browse Data', 'Edit Pragma', and 'Execute SQL'.

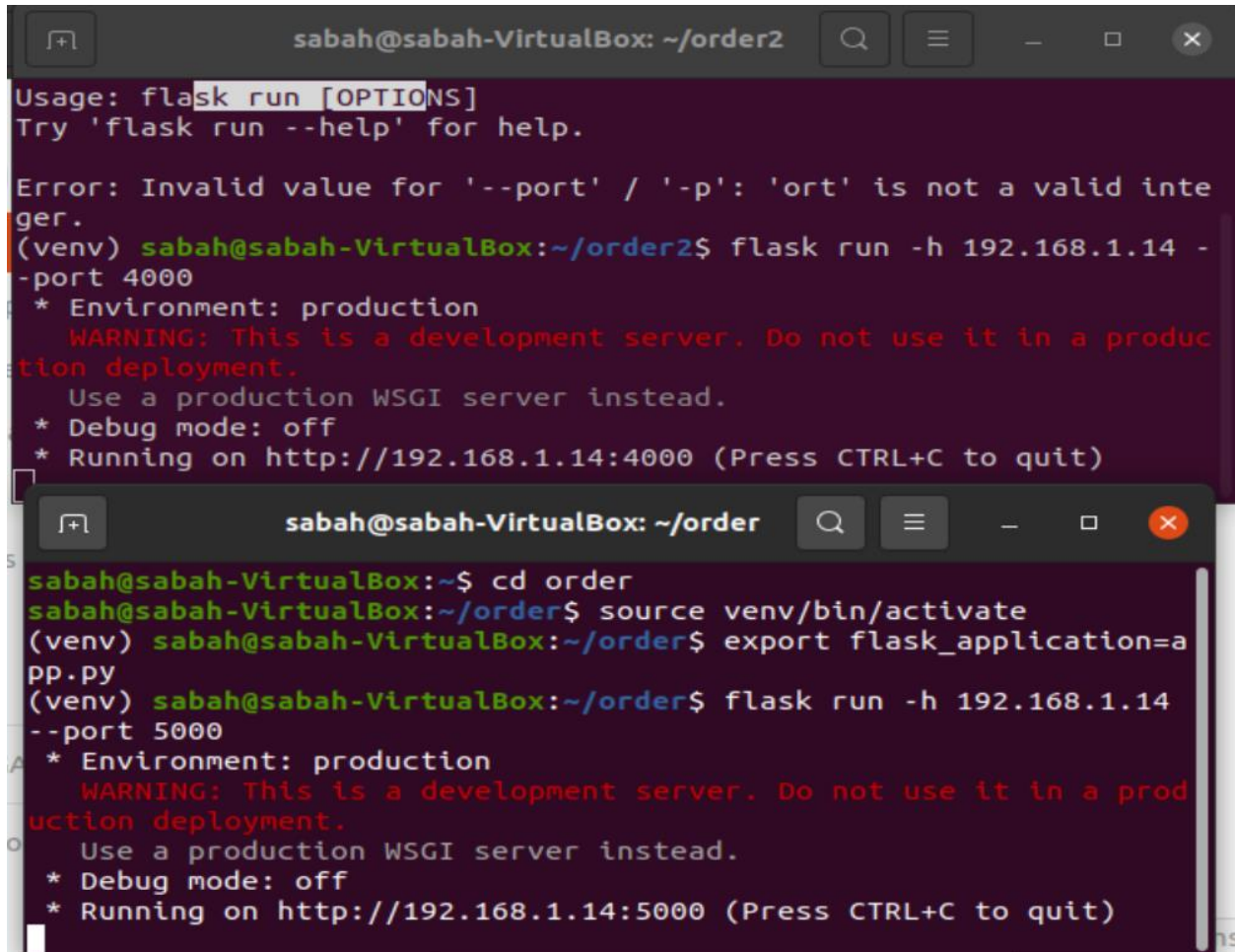
	item_number	title	topic	quantity	cost
	Filter	Filter	Filter	Filter	Filter
1	1	How to get ...	distributed ...	300	20.0
2	2	RBCs for ...	distributed ...	100	25.0
3	3	HXen and t...	undergradu...	600	60.0
4	4	Cooking for ...	undergradu...	600	100.0
5	5	. How to fini...	spring	300	150.0
6	6	Why theory ...	spring	100	125.0
7	7	Spring in th...	spring	600	50.0

Catalog server 1 with port 4000 and Catalog server 2 with port 5000:

```
sabah2@sabah2-VirtualBox: ~$ cd catalogServer2
sabah2@sabah2-VirtualBox: ~/catalogServer2$ source venv/bin/activate
(venv) sabah2@sabah2-VirtualBox: ~/catalogServer2$ export flask_application=app.py
(venv) sabah2@sabah2-VirtualBox: ~/catalogServer2$ flask run -h 192.168.1.70 --port 4000
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://192.168.1.70:4000 (Press CTRL+C to quit)

sabah2@sabah2-VirtualBox: ~/catalogServer$ source venv/bin/activate
(venv) sabah2@sabah2-VirtualBox: ~/catalogServer$ export flask_application=app.py
(venv) sabah2@sabah2-VirtualBox: ~/catalogServer$ flask run -h 192.168.1.70 --port 5000
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://192.168.1.70:5000 (Press CTRL+C to quit)
```

Order server 1 run at port 4000 and Order server 2 run at port 5000 :



```
sabah@sabah-VirtualBox: ~/order2
Usage: flask run [OPTIONS]
Try 'flask run --help' for help.

Error: Invalid value for '--port' / '-p': 'ort' is not a valid integer.
(venv) sabah@sabah-VirtualBox:~/order2$ flask run -h 192.168.1.14 -p 4000
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://192.168.1.14:4000 (Press CTRL+C to quit)

sabah@sabah-VirtualBox: ~/order
sabah@sabah-VirtualBox:~$ cd order
sabah@sabah-VirtualBox:~/order$ source venv/bin/activate
(venv) sabah@sabah-VirtualBox:~/order$ export flask_application=app.py
(venv) sabah@sabah-VirtualBox:~/order$ flask run -h 192.168.1.14 --port 5000
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://192.168.1.14:5000 (Press CTRL+C to quit)
```

Search 1 : first time (no cache) 57ms

The screenshot shows a REST client interface with the following details:

- URL:** `http://192.168.1.135:5000/search/spring`
- Method:** GET
- Status:** 200 OK
- Time:** 57 ms
- Size:** 347 B

The response body is displayed in JSON format:

```
{
  "result": [
    {
      "item_number": 5,
      "title": ". How to finish Project 3 on time"
    },
    {
      "item_number": 6,
      "title": "Why theory classes are so hard"
    },
    {
      "item_number": 7,
      "title": "Spring in the Pioneer Valley"
    }
  ]
}
```

After caching : 6m

The screenshot shows the same REST client interface as above, but with a significantly faster response time:

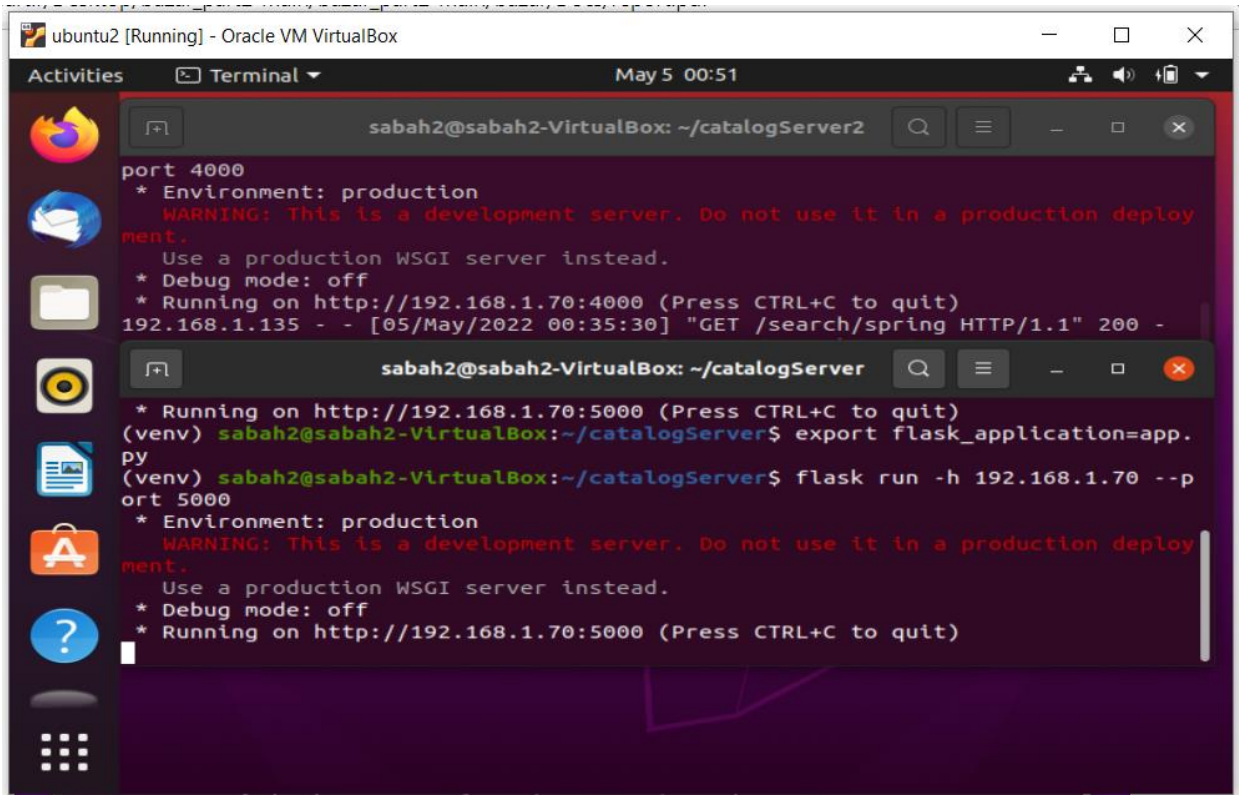
- URL:** `http://192.168.1.135:5000/search/spring`
- Method:** GET
- Status:** 200 OK
- Time:** 6 ms
- Size:** 347 B

The response body is identical to the first screenshot:

```
{
  "result": [
    {
      "item_number": 5,
      "title": ". How to finish Project 3 on time"
    },
    {
      "item_number": 6,
      "title": "Why theory classes are so hard"
    },
    {
      "item_number": 7,
      "title": "Spring in the Pioneer Valley"
    }
  ]
}
```

Making Search request for 2 times:

one for the first time and the other the second time which was taken from cache
(no request reached to the server because it was taken from cache)



```
ubuntu2 [Running] - Oracle VM VirtualBox
Activities Terminal May 5 00:51
sabah2@sabah2-VirtualBox: ~/catalogServer2
port 4000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://192.168.1.70:4000 (Press CTRL+C to quit)
192.168.1.135 - - [05/May/2022 00:35:30] "GET /search/spring HTTP/1.1" 200 -

sabah2@sabah2-VirtualBox: ~/catalogServer
* Running on http://192.168.1.70:5000 (Press CTRL+C to quit)
(venv) sabah2@sabah2-VirtualBox:~/catalogServer$ export flask_application=app.py
(venv) sabah2@sabah2-VirtualBox:~/catalogServer$ flask run -h 192.168.1.70 --port 5000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://192.168.1.70:5000 (Press CTRL+C to quit)
```

TRIAL2 :

INFO REQ: 56ms

This request was handled by server 2 as after applying load balancing .

Catalog 2 respond : 62m before caching , with load

Postman interface showing a GET request to `http://192.168.1.135:5000/info/6`. The response is a 200 OK with a 62 ms latency and 273 B body. The response body is a JSON object:

```
{
  "result": [
    {
      "cost": 125.0,
      "item_number": 6,
      "quantity": 100,
      "title": "Why theory classes are so hard",
      "topic": "spring"
    }
  ]
}
```

With caching :

Postman interface showing a GET request to `http://192.168.1.135:5000/info/6`. The response is a 200 OK with a 5 ms latency and 273 B body. The response body is a JSON object:

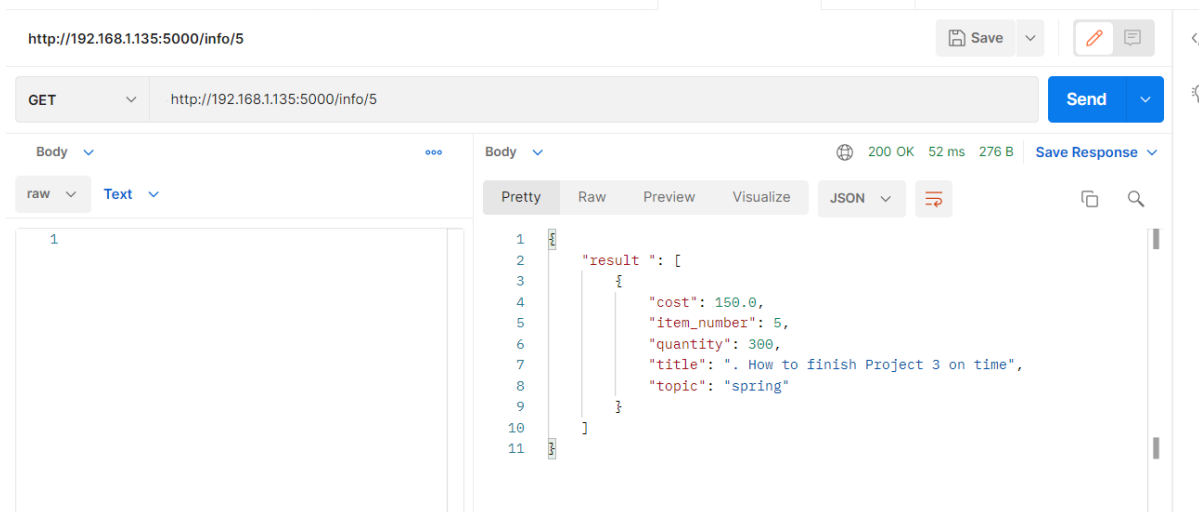
```
{
  "result": [
    {
      "cost": 125.0,
      "item_number": 6,
      "quantity": 100,
      "title": "Why theory classes are so hard",
      "topic": "spring"
    }
  ]
}
```


TRIAL3 :

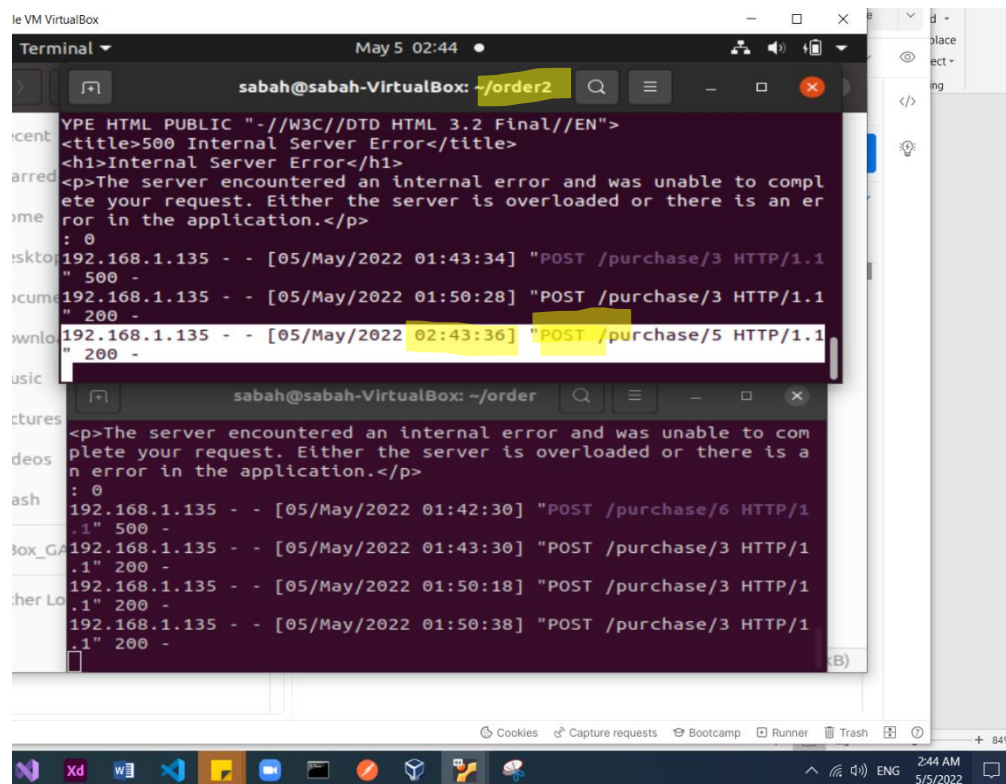
Data before purchase :

Checking consistency:

- Get information for item with id =6 :



- Applying purchase: it reached only to order server #2



- Applying Purchase request again: only in order1 and not in order 2

The image shows two terminal windows from a VirtualBox environment. The top window, titled 'sabah@sabah-VirtualBox: ~/order2', displays an HTTP 500 Internal Server Error response. The response body is HTML: <title>500 Internal Server Error</title>, <h1>Internal Server Error</h1>, and <p>The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.</p>. Below this, log entries show three POST requests to /purchase/3 and one to /purchase/5, with status codes 500 and 200 respectively. The bottom window, titled 'sabah@sabah-VirtualBox: ~/order', shows a continuation of the logs, including a message 'n error in the application.</p>' and several more POST requests to /purchase/6 and /purchase/3, with status codes 500 and 200. The last log entry is a POST request to /purchase/5 with a 200 status code.

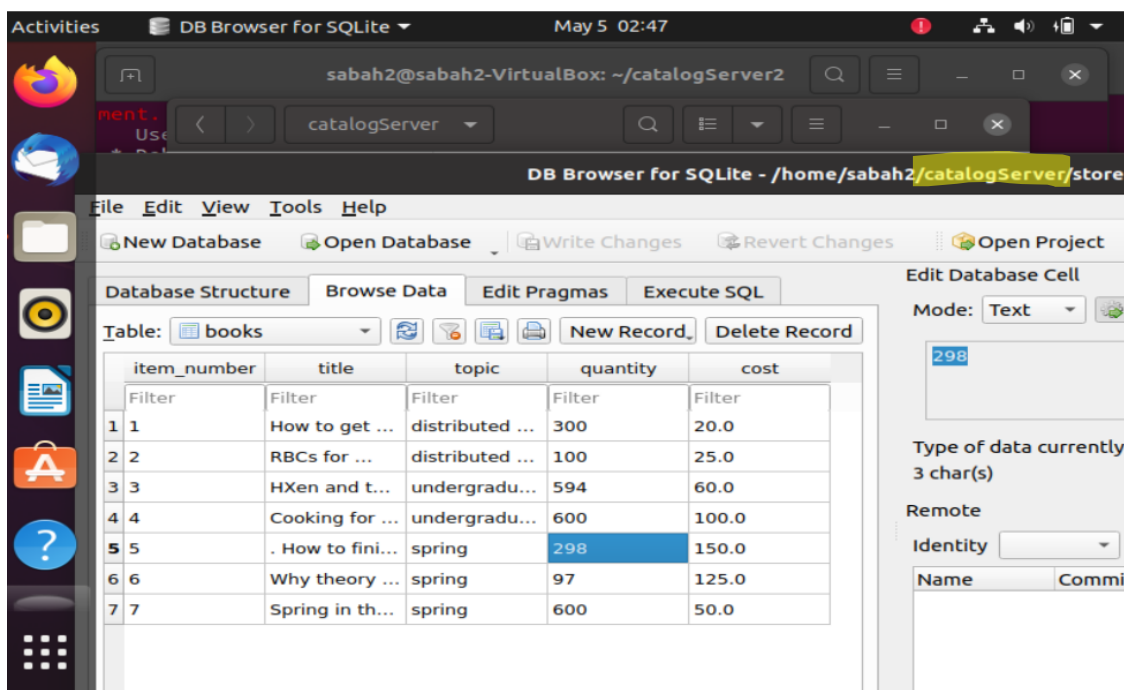
```

TYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.</p>
: 0
192.168.1.135 - - [05/May/2022 01:43:34] "POST /purchase/3 HTTP/1.1" 500 -
192.168.1.135 - - [05/May/2022 01:50:28] "POST /purchase/3 HTTP/1.1" 200 -
192.168.1.135 - - [05/May/2022 02:43:36] "POST /purchase/5 HTTP/1.1" 200 -

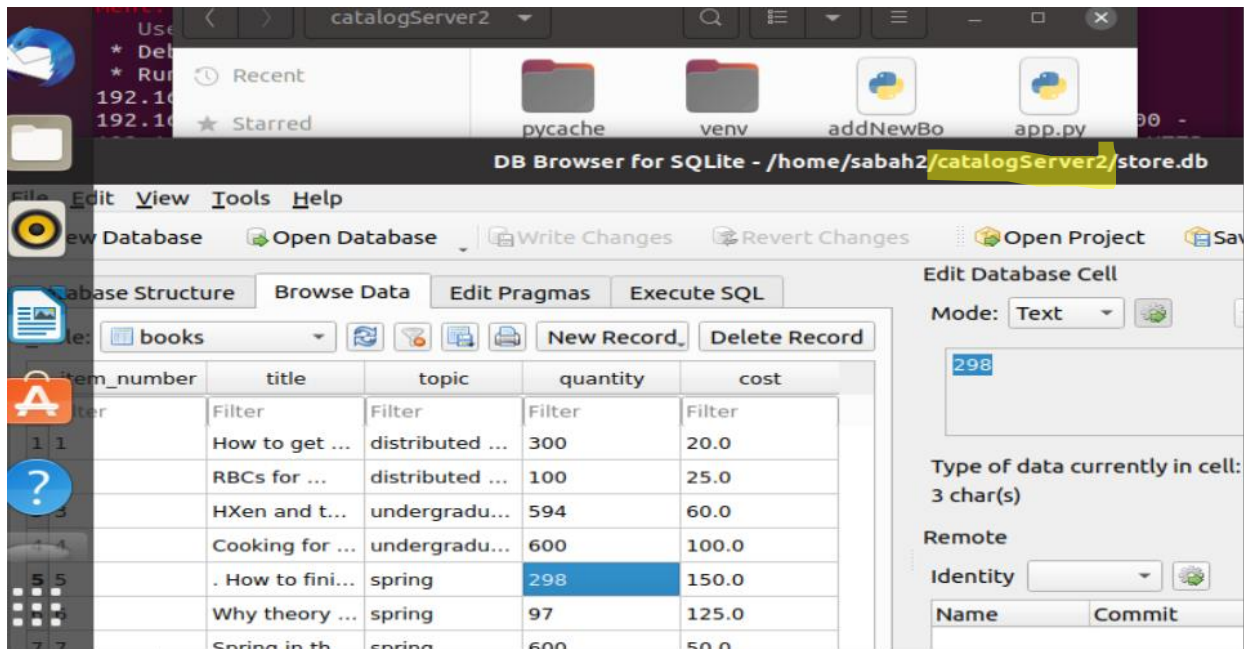
sabah@sabah-VirtualBox: ~/order
n error in the application.</p>
: 0
192.168.1.135 - - [05/May/2022 01:42:30] "POST /purchase/6 HTTP/1.1" 500 -
192.168.1.135 - - [05/May/2022 01:43:30] "POST /purchase/3 HTTP/1.1" 200 -
192.168.1.135 - - [05/May/2022 01:50:18] "POST /purchase/3 HTTP/1.1" 200 -
192.168.1.135 - - [05/May/2022 01:50:38] "POST /purchase/3 HTTP/1.1" 200 -
192.168.1.135 - - [05/May/2022 02:45:30] "POST /purchase/5 HTTP/1.1" 200 -

```


- Database result *in catalog1* after these s2 purchase : (the amount decreased 2)



- Result from database in catalog 2 : (it decreased here too automatically)



Try get info for item 5 after making changes in the database from the purchase process:

From front: (cache doesn't work here) it give the long time

Overview | GET https://restcount | GET status | Mock server | GET http://192.168.1.135:5000/info/5 | Sabah app status checker

http://192.168.1.135:5000/info/5

GET http://192.168.1.135:5000/info/5

Body (Text): 1

Body (Pretty): 200 OK 51 ms 276 B

```
1 { "result": [{"cost": 150.0, "item_number": 5, "quantity": 298, "title": ". How to finish Project 3 on time", "topic": "spring"}]}
```

Trying again to test cache for info 5:

http://192.168.1.135:5000/info/5

GET http://192.168.1.135:5000/info/5

Body (Text): 1

Body (JSON): 200 OK 5 ms 276 B

```
1 {
2   "result": [
3     {
4       "cost": 150.0,
5       "item_number": 5,
6       "quantity": 298,
7       "title": ". How to finish Project 3 on time",
8       "topic": "spring"
9     }
10  ]
11 }
```