Lab 6: NoSQL database models

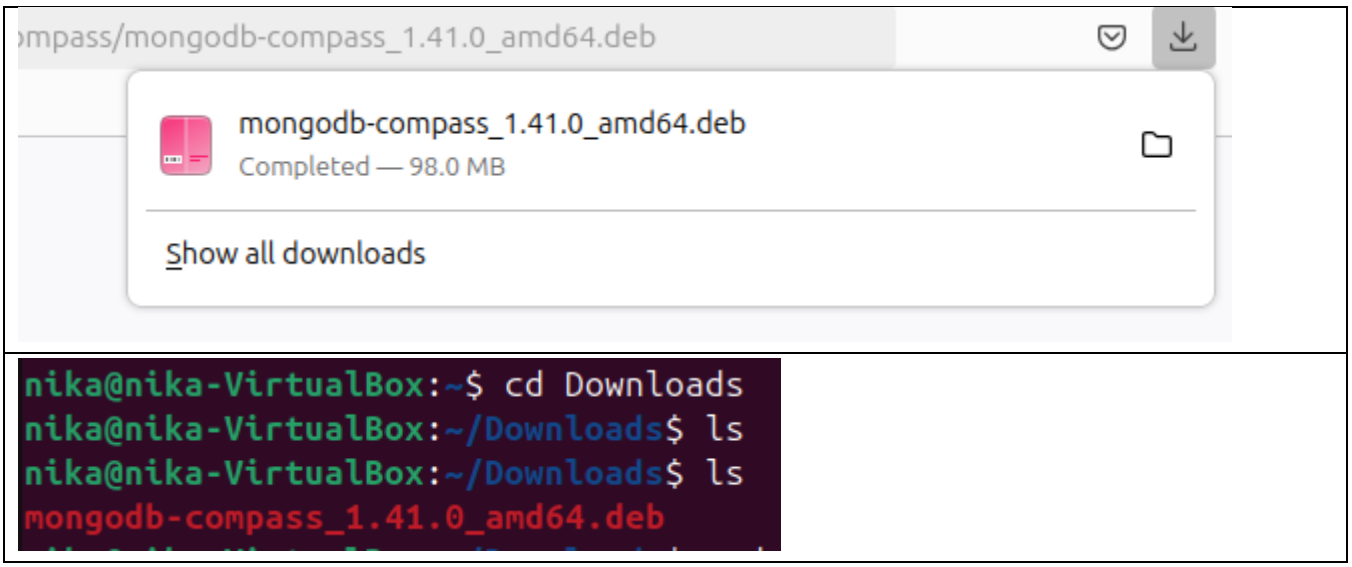
PreLab

<Tongol> (Chapter 9, Section 9.1 to 9.6) Building Cross-Platform Mobile and Web Apps for Engineers and Scientists: An Active Learning Approach
9781305855892

Chapters 9.1 to 9.6 of Building Cross-Platform Mobile and Web Apps for Engineers and Scientists by P. Lingras (2016) provide a comprehensive introduction to NoSQL databases, especially MongoDB, and their practical application in application development. Section 9.1 examines the emergence of the NoSQL database model and emphasizes its ability to manage large, unstructured and rapidly changing data more efficiently than traditional relational databases. Section 9.2 introduces MongoDB, a widely used NoSQL database, and highlights its document-oriented structure that makes data storage more flexible and scalable. Sections 9.3 and 9.4 focus on designing a NoSQL database model, and use the Thyroid App as a case study to demonstrate the design of the schema, data organization and optimization strategy. Section 9.5 provides a step-by-step guide to launch a MongoDB server for thyroid application and prepare the system for data transactions. Finally, Section 9.6 takes the process of storing new user data on the MongoDB server and shows how the NoSQL database simplifies data management in modern applications. Together, these sections provide a practical approach to understanding the NoSQL concept and its application in real-world projects, making them very valuable for engineers and scientists developing cross-platform mobile and web applications.

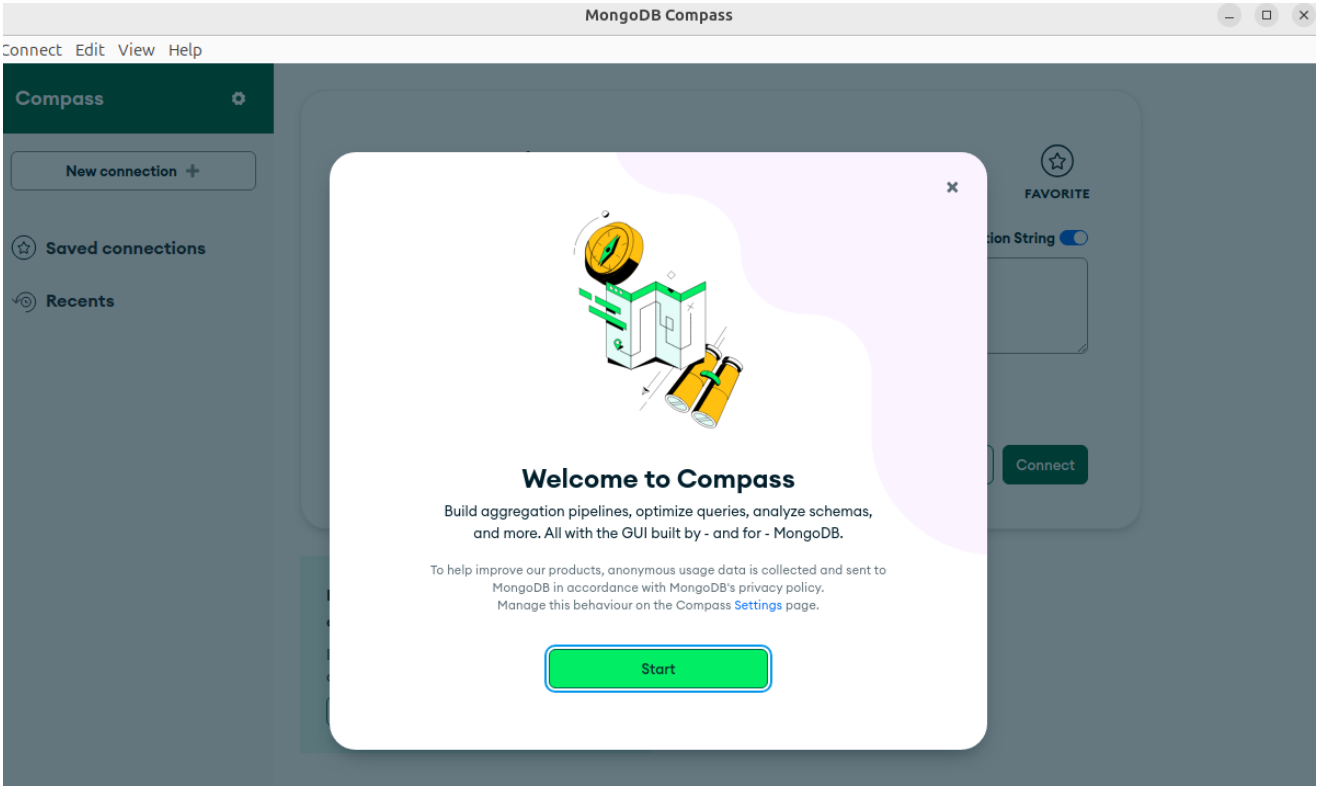
InLab

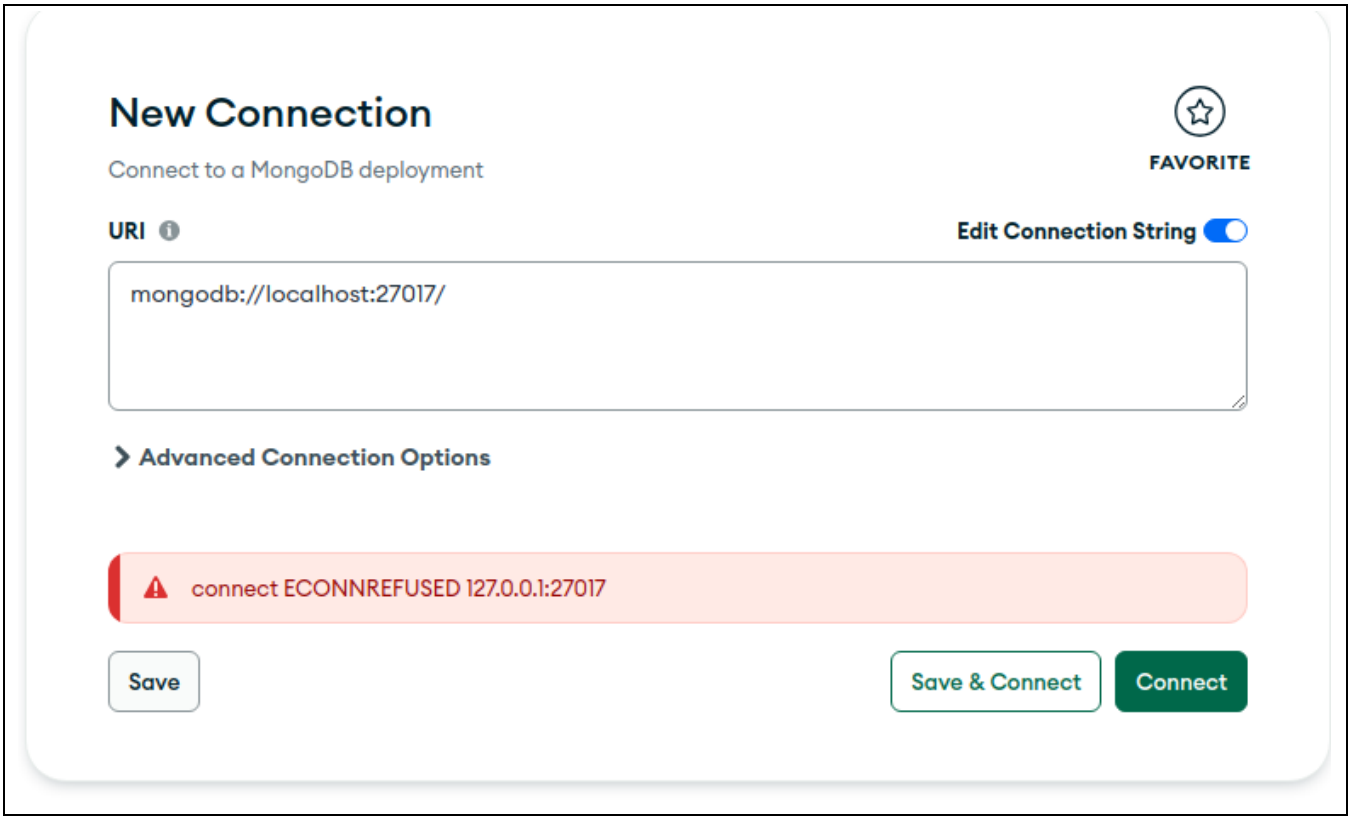
- Install MongoDB Compass. Download here: MongoDB Compass | MongoDB. For Ubuntu 64 bit OS, Download here >> https://downloads.mongodb.com/compass/mongodb-compass_1.41.0_amd64.deb
  Installation:

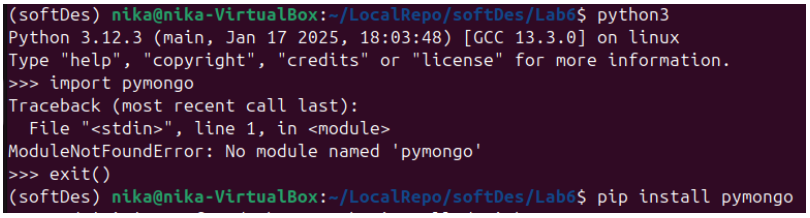
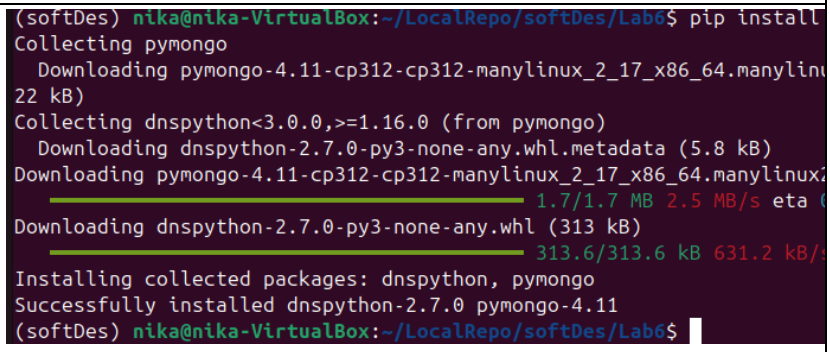
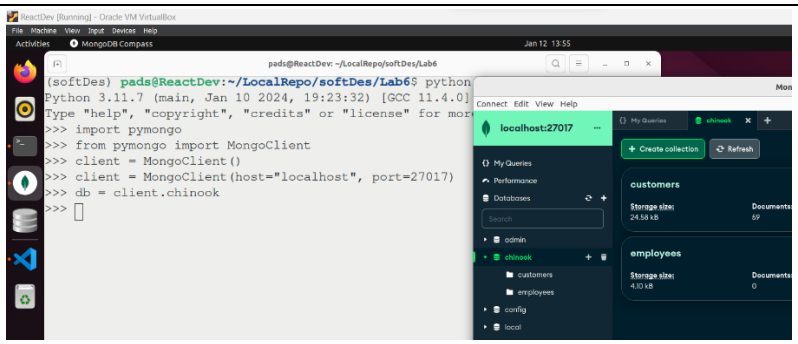


```
nika@nika-VirtualBox:~/Downloads$ sudo dpkg -i mongodb-compass_1.41.0_amd64.deb
[sudo] password for nika:
Selecting previously unselected package mongodb-compass.
(Reading database ... 148405 files and directories currently installed.)
Preparing to unpack mongodb-compass_1.41.0_amd64.deb ...
Unpacking mongodb-compass (1.41.0) ...
Setting up mongodb-compass (1.41.0) ...
Processing triggers for gnome-menus (3.36.0-1.1ubuntu3) ...
Processing triggers for desktop-file-utils (0.27-2build1) ...
```

```
nika@nika-VirtualBox:~/Downloads$ sudo apt-get install -f
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 216 not upgraded.
```





Activate softDes virtual environment	
Pip install pymongo	
Inside the python interactive terminal, run the following commands shown in the screengrab. Note: You need to run the mongodb process before you can connect to your mongodb database. Command: sudo systemctl start mongod	

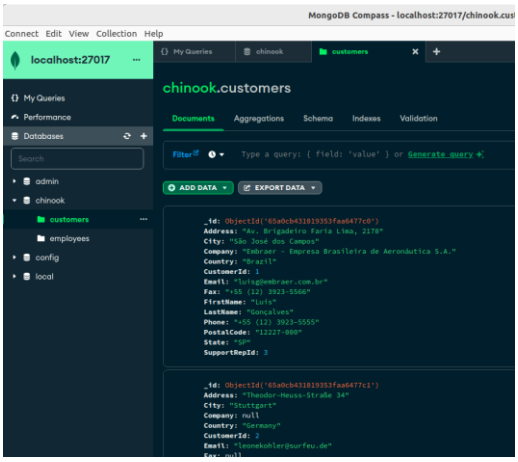
```
(softDes) nika@nika-VirtualBox:~/LocalRepo/softDes/Lab6$ python3
Python 3.12.3 (main, Jan 17 2025, 18:03:48) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pymongo
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'pymongo'
>>> exit()
(softDes) nika@nika-VirtualBox:~/LocalRepo/softDes/Lab6$ pip install pymongo
```

```
(softDes) nika@nika-VirtualBox:~/LocalRepo/softDes/Lab6$ pip install
Collecting pymongo
  Downloading pymongo-4.11-cp312-cp312-manylinux_2_17_x86_64.manylinu
22 kB)
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Downloading dnspython-2.7.0-py3-none-any.whl.metadata (5.8 kB)
Downloading pymongo-4.11-cp312-cp312-manylinux_2_17_x86_64.manylinux2
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 2.5 MB/s eta (
Downloading dnspython-2.7.0-py3-none-any.whl (313 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 313.6/313.6 kB 631.2 kB/s
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.7.0 pymongo-4.11
(softDes) nika@nika-VirtualBox:~/LocalRepo/softDes/Lab6$
```

```
(softDes) pads@ReactDev:~/LocalRepo/softDes/Lab6$ python
Python 3.11.7 (main, Jan 10 2024, 19:23:32) [GCC 11.4.0]
Type "help", "copyright", "credits" or "license" for mor
>>> import pymongo
>>> from pymongo import MongoClient
>>> client = MongoClient()
>>> client = MongoClient(host="localhost", port=27017)
>>> db = client.chinook
>>>
```

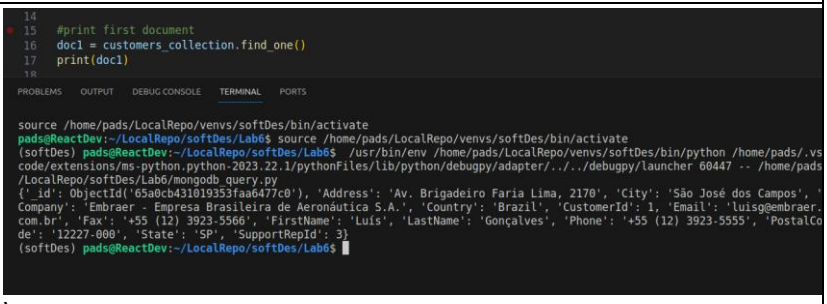
Navigate to Lab6 directory and open visual studio code

Check if you have the chinook database and the customers collections and documents. If you don't have, open the chinook database and "export as JSON file" the customers table



Enter the following lines in VS Code:

```
from pymongo import MongoClient
import pprint
import re

# client = MongoClient(host="localhost",
port=27017)
client =
MongoClient("mongodb://localhost:27017/")

# Get reference to 'chinook' database
db = client["chinook"]

# Get a reference to the 'customers'
collection
customers_collection = db["customers"]
# print(customers_collection)

#print first document
doc1 = customers_collection.find_one()
print(doc1)

client.close()
```



\

Print all documents of the customers collection

Comment this

```
#print first document
# doc1 =
customers_collection.find_one()
# print(doc1)
```

And add the following line

```
#print all documents
for all_doc in
customers_collection.find():
    print(all_doc)
```

```
13  # print(customers_collection)
14
15  #print first document
16  # doc1 = customers_collection.find_one()
17  # print(doc1)
18
19  #print all documents
20  for all_doc in  customers_collection.find():
21      print(all_doc)
22
23  client.close()
24
```

```
18
19  #print all documents
20  for all_doc in  customers_collection.find():
21      print(all_doc)
22
23  client.close()
24
25  #return only the LastName
26  # for rec in customers_collection.find({},{"_id":0,"LastName": 1, "FirstName": 1}):
27  #     print(rec)
28
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

ne': '+61 (02) 9332 3633', 'PostalCode': '2010', 'State': 'NSW', 'SupportRepId': 4}
{'_id': ObjectId('65a0cb431019353faa6477f7'), 'Address': '307 Macacha Güemes', 'City': 'Buenos Aires', 'Company': Nor
ntry': 'Argentina', 'CustomerId': 56, 'Email': 'diego.gutierrez@yahoo.ar', 'Fax': None, 'FirstName': 'Diego', 'LastNa
utiérrez', 'Phone': '+54 (0)11 4311 4333', 'PostalCode': '1106', 'State': None, 'SupportRepId': 4}
{'_id': ObjectId('65a0cb431019353faa6477f8'), 'Address': 'Calle Lira, 198', 'City': 'Santiago', 'Company': None, 'Cou
Chile', 'CustomerId': 57, 'Email': 'luisrojas@yahoo.cl', 'Fax': None, 'FirstName': 'Luis', 'LastName': 'Rojas', 'Phc
56 (0)2 635 4444', 'PostalCode': None, 'State': None, 'SupportRepId': 5}
{'_id': ObjectId('65a0cb431019353faa6477f9'), 'Address': '12,Community Centre', 'City': 'Delhi', 'Company': None, 'Co
'India', 'CustomerId': 58, 'Email': 'manoj.pareek@rediff.com', 'Fax': None, 'FirstName': 'Manoj', 'LastName': 'Paree
one': '+91 0124 39883988', 'PostalCode': '110017', 'State': None, 'SupportRepId': 3}
{'_id': ObjectId('65a0cb431019353faa6477fa'), 'Address': '3,Raj Bhavan Road', 'City': 'Bangalore', 'Company': None, '
: 'India', 'CustomerId': 59, 'Email': 'puja_srivastava@yahoo.in', 'Fax': None, 'FirstName': 'Puja', 'LastName': 'Sri
, 'Phone': '+91 080 22289999', 'PostalCode': '560001', 'State': None, 'SupportRepId': 3}
(softDes) pads@ReactDev:~/LocalRepo/softDes/Lab6$
```

```
#return only the LastName and
FirstName
for rec in
customers_collection.find({},{"_id
":0,"LastName": 1, "FirstName":
1}):
    print(rec)
```

```
22
23  #return only the LastName and FirstName
24  for rec in customers_collection.find({},{"_id":0,"LastName": 1, "FirstName": 1}):
25      print(rec)
26
27
28  client.close()
29
30  #return only the LastName
31  # for rec in customers_collection.find({},{" id":0,"LastName": 1, "FirstName": 1}):
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

(softDes) pads@ReactDev:~/LocalRepo/softDes/Lab6$  cd /home/pads/LocalRepo/softDes/Lab6 ; /
venvs/softDes/bin/python /home/pads/.vscode/extensions/ms-python.python-2023.22.1/pythonFil
../debugpy/launcher 38043 -- /home/pads/LocalRepo/softDes/Lab6/mongodb_query.py
{'FirstName': 'Luís', 'LastName': 'Gonçalves'}
{'FirstName': 'Leonie', 'LastName': 'Köhler'}
{'FirstName': 'François', 'LastName': 'Tremblay'}
{'FirstName': 'Bjørn', 'LastName': 'Hansen'}
{'FirstName': 'František', 'LastName': 'Wichterlová'}
{'FirstName': 'Helena', 'LastName': 'Holý'}
{'FirstName': 'Astrid', 'LastName': 'Gruber'}
{'FirstName': 'Daan', 'LastName': 'Peeters'}
```

```
Print all customers with LastName
that starts with "G"

rgx = re.compile('^G.*?$',
re.IGNORECASE)  # compile the
regex
cursor =
customers_collection.find({"LastNa
me":rgx })
num_docs = 0
for document in cursor:
    num_docs += 1
    pprint.pprint(document)
    print()
print("# of documents found: " +
str(num_docs))
```

```
# rgx = re.compile('.*G.*', re.IGNORECASE)
# bcd% → ^bcd.*?$
rgx = re.compile('^G.*?$', re.IGNORECASE)
cursor = customers_collection.find({"LastNa
num_docs = 0
for document in cursor:
    num_docs += 1
    pprint.pprint(document)
    print()
print("# of documents found: " + str(num_do

client.close()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
 'Country': 'Argentina',
 'CustomerId': 56,
 'Email': 'diego.gutierrez@yahoo.ar',
 'Fax': None,
 'FirstName': 'Diego',
 'LastName': 'Gutiérrez',
 'Phone': '+54 (0)11 4311 4333',
 'PostalCode': '1106',
 'State': None,
 'SupportRepId': 4,
 '_id': ObjectId('65a0cb431019353faa6477f7')}

# of documents found: 7
(softDes) pads@ReactDev:~/LocalRepo/softDes/Lab6$
```

[redacted]

```
41  # cursor = customers_collection.find(doc
42
43  # rgx = re.compile('.*G.*', re.IGNORECAS
44  # bcd% → ^bcd.*?$
45  rgx = re.compile('^Go.*?$', re.IGNORECAS
46  cursor = customers_collection.find({"Las
47  num_docs = 0
48  for document in cursor:
49      num_docs += 1
50      pprint.pprint(document)
51      print()
52  print("# of documents found: " + str(num
53
54  client.close()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
 'Country': 'USA',
 'CustomerId': 23,
 'Email': 'johngordon22@yahoo.com',
 'Fax': None,
 'FirstName': 'John',
 'LastName': 'Gordon',
 'Phone': '+1 (617) 522-1333',
 'PostalCode': '2113',
 'State': 'MA',
 'SupportRepId': 4,
 '_id': ObjectId('65a0cb431019353faa6477d6')}

# of documents found: 3
```

[redacted]

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: customers Filter in any column

	CustomerId	FirstName	LastName	Company	Address
	Filter	Filter	Filter	Filter	Filter
1	12	Roberto	Almeida	Riotur	Praça Pio X, 119
2	28	Julia	Barnett	NULL	302 S 700 E
3	39	Camille	Bernard	NULL	4, Rue Milton
4	18	Michelle	Brooks	NULL	627 Broadway
5	29	Robert	Brown	NULL	796 Dundas Street West
6	21	Kathy	Chase	NULL	801 W 4th Street
7	26	Richard	Cunningham	NULL	2211 W Berry Street
8	41	Marc	Dubois	NULL	11, Place Bellecour
9	34	João	Fernandes	NULL	Rua da Assunção 53
10	30	Edward	Francis	NULL	230 Elgin Street
11	42	Wyatt	Girard	NULL	9, Place Louis Barthou
12	1	Luís	Gonçalves	Embraer - Empresa Brasileira de …	Av. Brigadeiro Faria Lima
13	23	John	Gordon	NULL	69 Salem Street
14	19	Tim	Goyer	Apple Inc.	1 Infinite Loop
15	27	Patrick	Gray	NULL	1033 N Park Ave
16	7	Astrid	Gruber	NULL	Rotenturmstraße 4, 1010 I

References:

- Python and MongoDB: Connecting to NoSQL Databases – Real Python
- Installation: Install MongoDB — MongoDB Manual
- https://www.mongodb.com/basics/get-started
- Perform MongoDB CRUD Operations CRUD operations: MongoDB CRUD Operations — MongoDB Manual
- MongoDB CRUD Operations in Python >> MongoDB CRUD Operations in Python - Learn | MongoDB

PostLab

Using the ERD shown here >> Chinook DB ERD , create the artists-albums-tracks database in MongoDB compass.

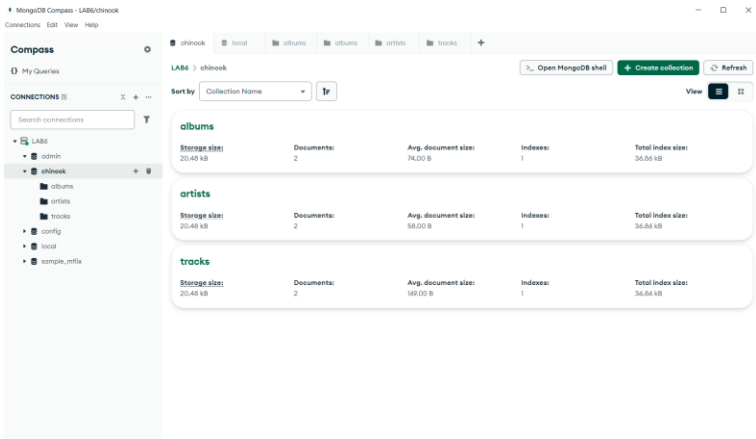


Figure 1. MongoDB Compass View

Figure 1 presents the home screen of MongoDB Compass with the implementation of the three databases, namely the artists, albums, and tracks.

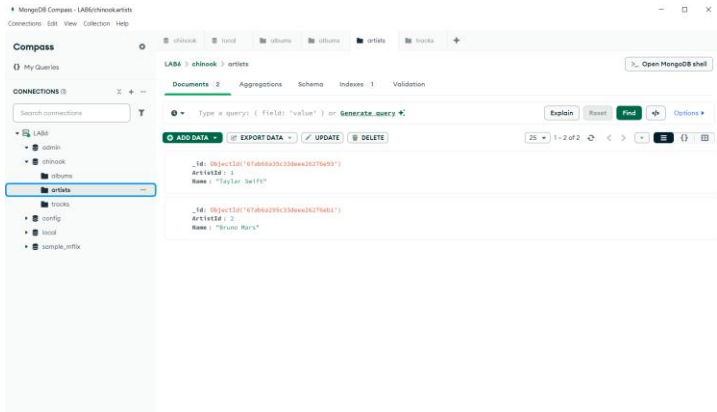


Figure 2. Artists Database

Figure 2 presents the database of the artists with the attributes of artist ID and artist name. The database also contains data of two artists, Taylar Swift, with an artist ID of 1, and Bruno Mars, with an artist ID of 2.

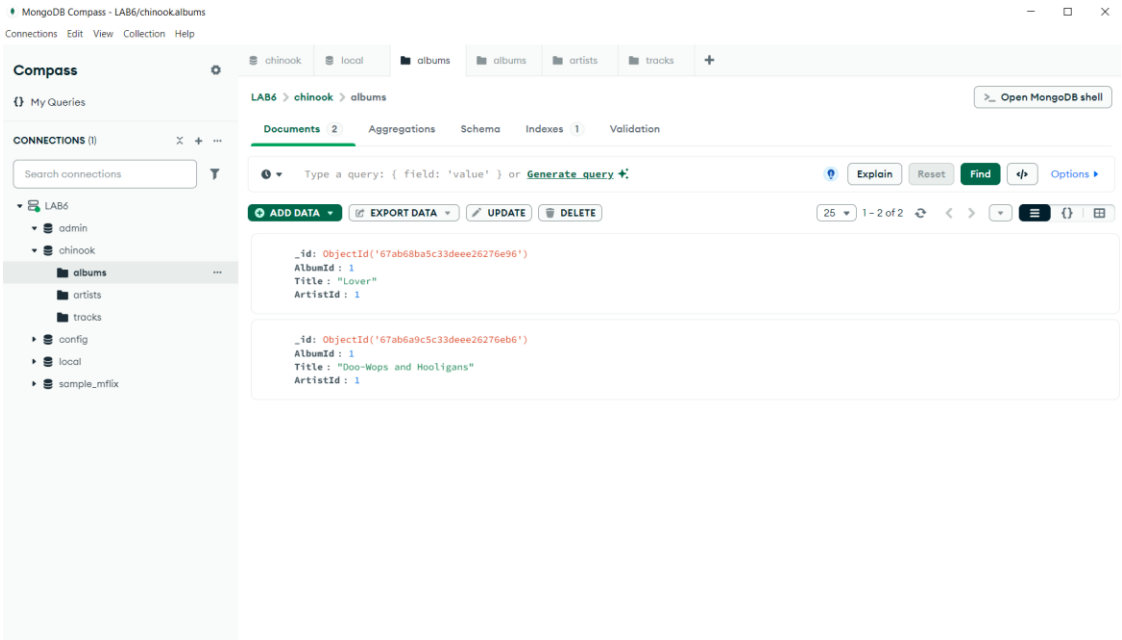


Figure 3. Albums Database

Figure 3 presents the album database with the attributes album ID, album name, and the associated artist ID. The database contains two album data: the "Lover" album with an album ID and artist ID 1, and the "Doo-Wops and Hooligans" with an album ID and artist ID of 2.

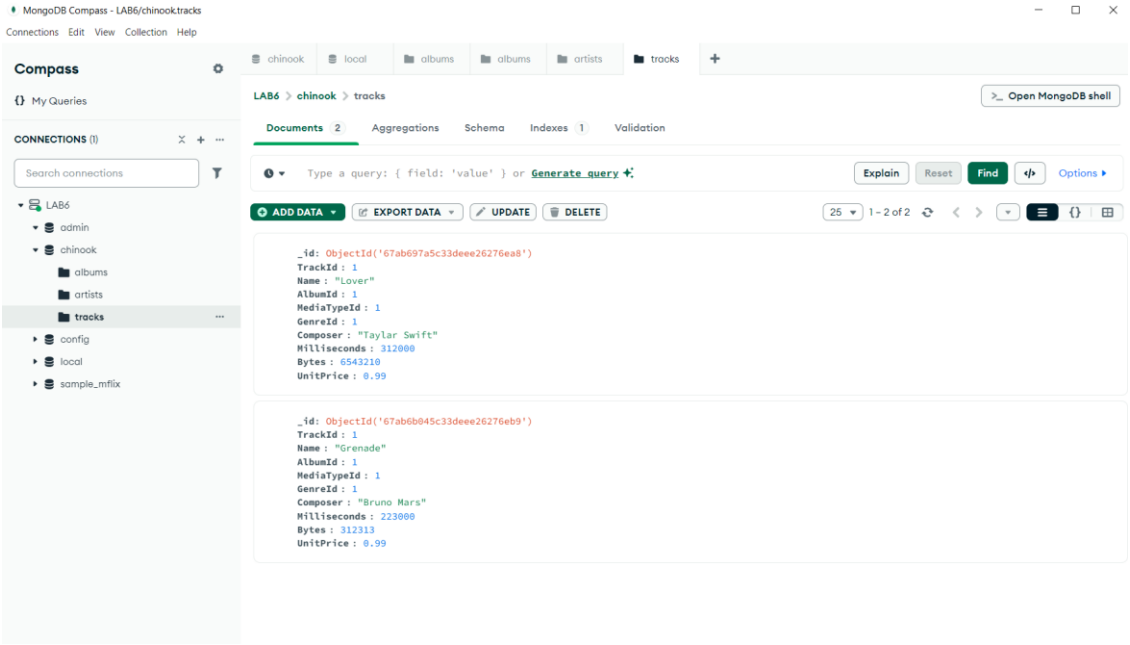


Figure 4. Tracks Database

Figure 4 presents the track database containing the attributes track ID, track name, the album ID associated with it, media type ID, genre ID, its composer, time in milliseconds, file size in bytes, and unit price.