



MAPÚA UNIVERSITY

SCHOOL OF ELECTRICAL, ELECTRONICS, AND COMPUTER ENGINEERING

Experiment 6:

NoSQL Database Models

CPE106L (Software Design Laboratory)

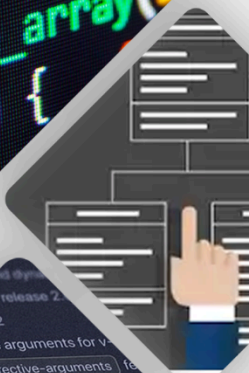
Member 1: ASIDAO, Shane Eowyn R.

Member 2: GONZALES, Benedick B.

Member 3: LULU, Neal Axel M.

Member 4: VARGAS, Imar Colt S.

Group No.: 4
Section: E04



PreLab

Readings, Insights, and Reflection

- [Chapter 9, Section 9.1 to 9.6] Lingras, P. (2016-01-01). Building Cross-Platform Mobile and Web Apps for Engineers and Scientists: An Active Learning Approach. [[VitalSource Bookshelf version]]. Retrieved from vbk://9781305855892
- <https://www.mongodb.com/docs/manual/introduction/>
- <https://www.sqlitetutorial.net/wp-content/uploads/2018/03/sqlite-sample-database-diagram-color.pdf>

Reflecting on the materials from *Building Cross-Platform Mobile and Web Apps for Engineers and Scientists: An Active Learning Approach* by P. Lingras (2016), along with the documentation from MongoDB and the SQLite sample database diagram, I've gained valuable insights into mobile and web application development, especially from an engineering and scientific standpoint.

Cross-platform development has become an essential aspect of modern engineering and scientific applications. Lingras highlights the importance of ensuring that web and mobile applications function seamlessly across various operating systems. As mobile technology continues to advance, I recognize the necessity of engaging with cross-platform solutions that enhance accessibility and usability. The book's active learning approach resonated with me—it reinforced how hands-on experience with coding and application development is key to mastering modern computational tools.

One of the biggest takeaways for me was the role of database management in application development. Comparing SQL and NoSQL databases helped me understand their unique strengths and limitations. MongoDB, as a NoSQL database, stands out for its flexibility and scalability, making it perfect for handling unstructured or semi-structured data. Its dynamic schema is a game-changer for applications requiring real-time updates and large-scale data management. On the other hand, SQLite

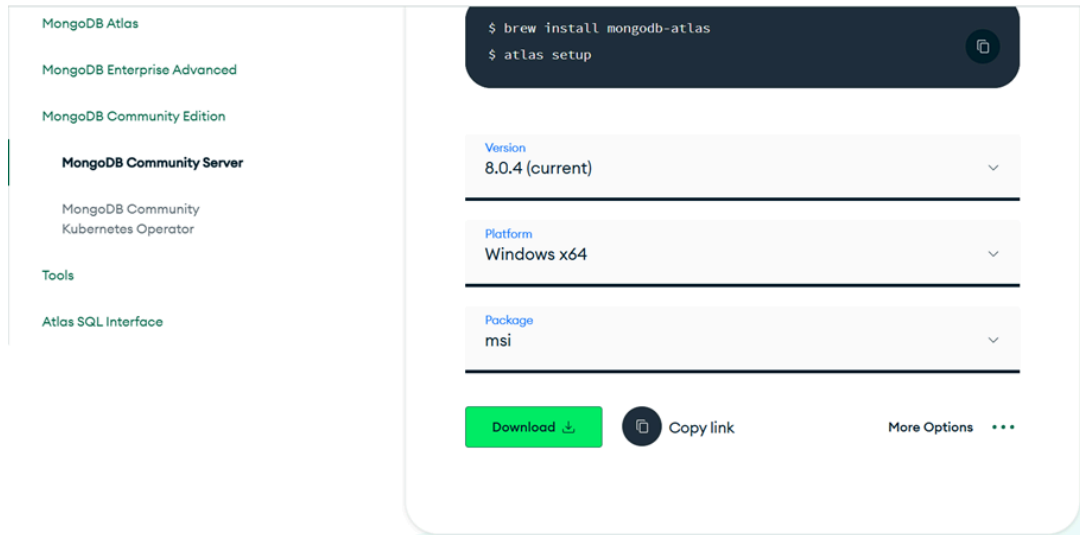
follows a structured, relational database model that is lightweight and ideal for mobile applications needing offline functionality. Looking at the SQLite sample database diagram, I could see how efficiently it organizes structured data with minimal setup—making it a solid choice for applications prioritizing local storage.

Understanding both NoSQL and SQL databases has been eye-opening for me as I think about mobile and web application development. Choosing between MongoDB and SQLite really depends on the specific needs of an application—whether it requires a scalable, flexible solution or a lightweight, structured database. Additionally, cross-platform development demands careful attention to database compatibility, performance, security, and data synchronization across different environments.

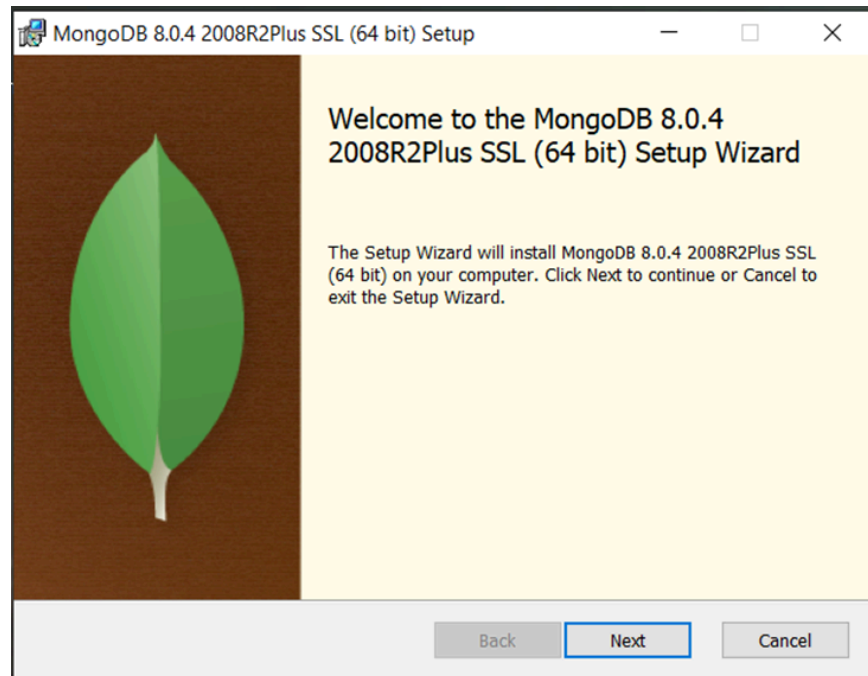
Ultimately, integrating mobile and web applications into engineering and scientific work significantly enhances data accessibility and computational efficiency. Lingras' emphasis on active learning, along with a deep dive into database structures, has given me a strong foundation for developing robust applications. By weighing the advantages of both MongoDB and SQLite, I feel more equipped to make informed decisions about database implementation, ensuring efficiency and adaptability in real-world scenarios

InLab

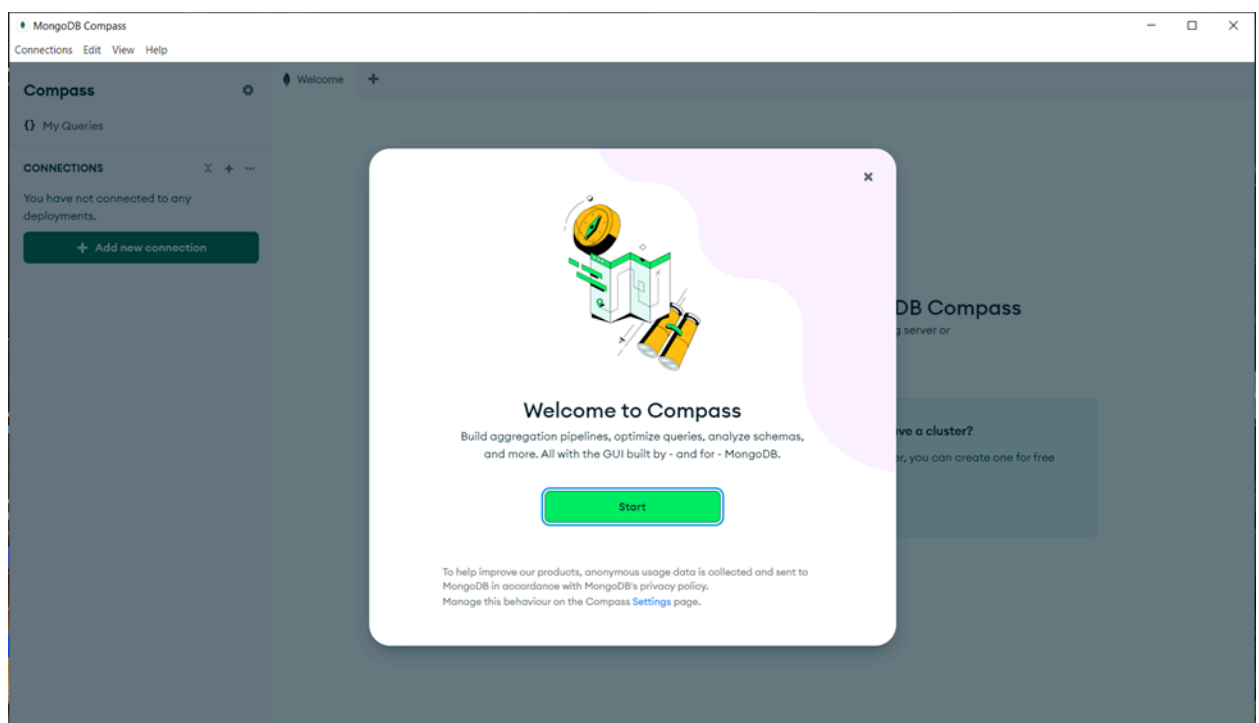
- Install MongoDB Compass. Download here: MongoDB Compass | MongoDB. For Ubuntu 64 bit OS, Download here >>
https://downloads.mongodb.com/compass/mongodb-compass_1.41.0_amd64.deb
Installation:

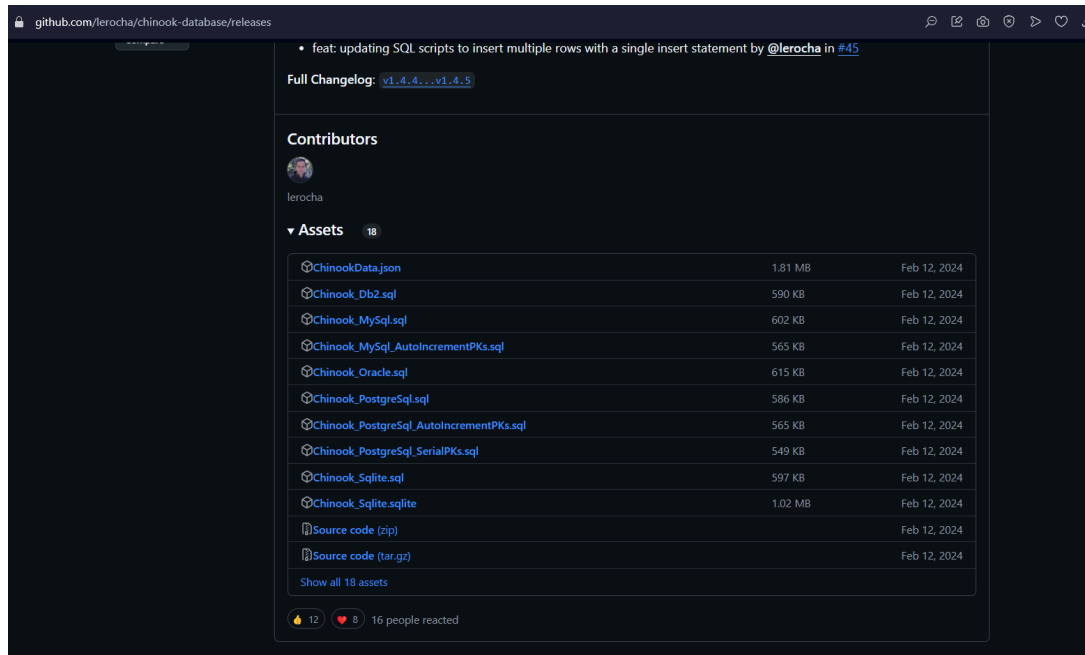


We downloaded MongoDB first. Once MongoDB had been downloaded, we installed it.

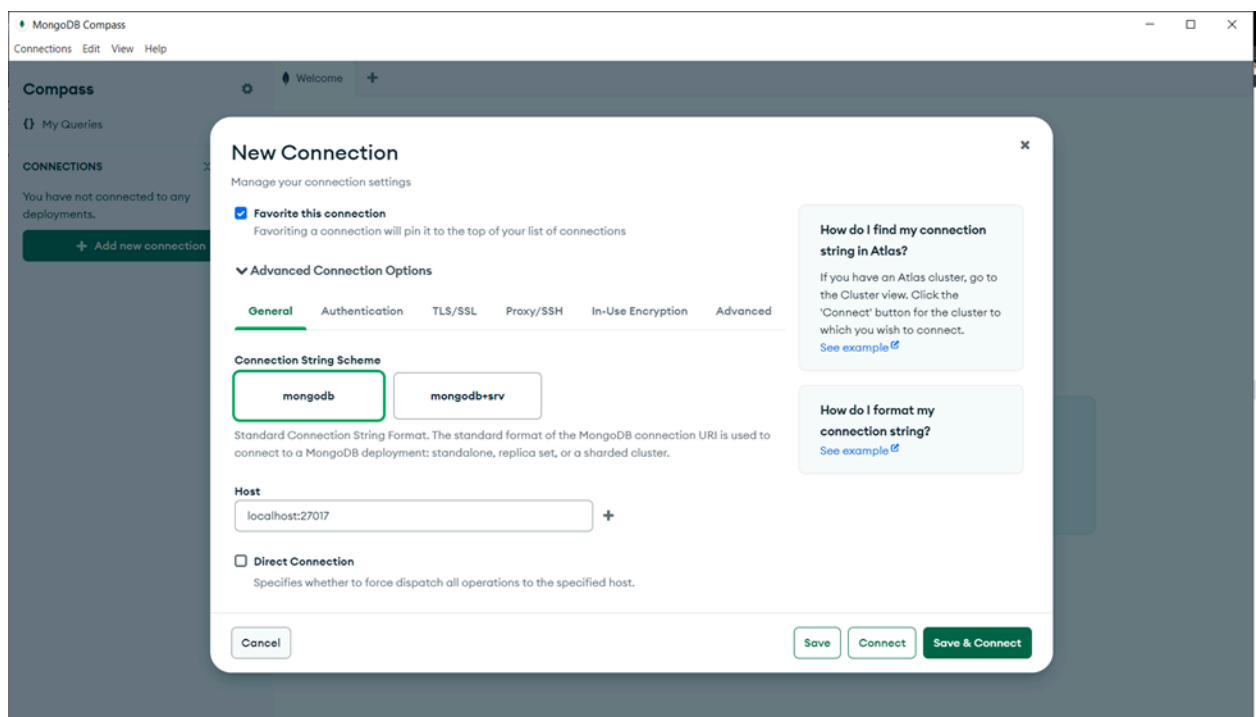


To verify the configuration, we connected to MongoDB, installed the appropriate Python version, and imported *chinook.db* into MongoDB Compass using SQLite.

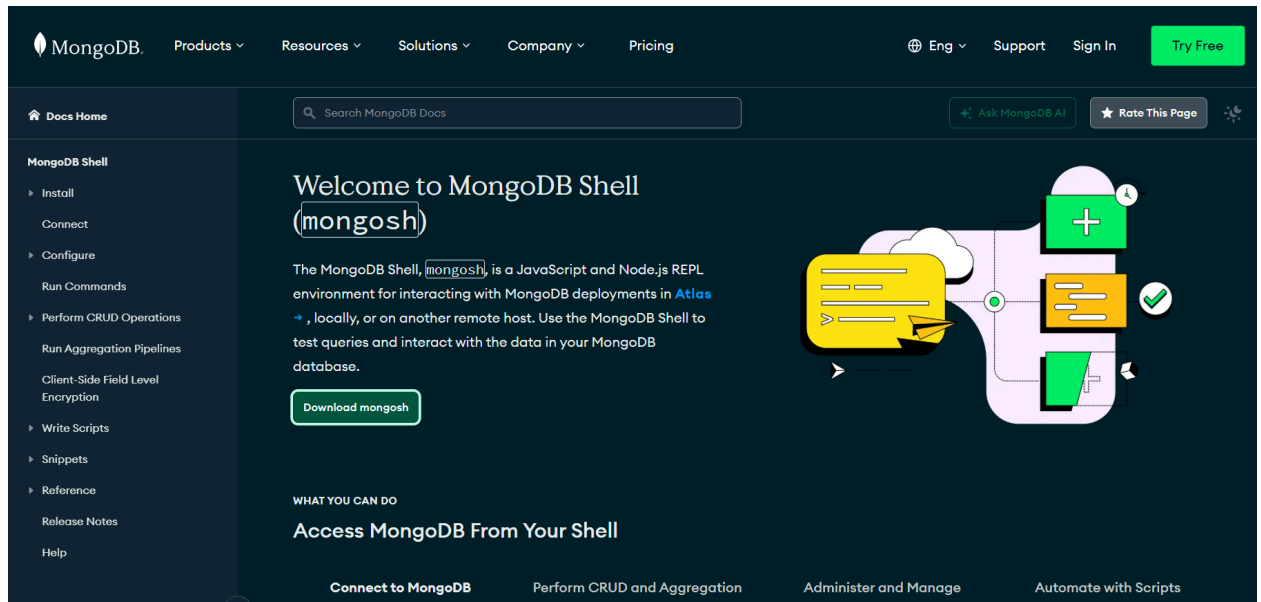




Install Chinook_Sqlite.sqlite



Creating a database connection



Install mongosh and extract the zip file to your desired directory

```
cmd mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Microsoft Windows [Version 10.0.22631.4890]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>mongosh
Current Mongosh Log ID: 67aded725fdc8812654d7941
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.9
Using MongoDB:      8.0.4
Using Mongosh:       2.3.9

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

-----
  The server generated these startup warnings when booting
  2025-02-13T20:27:33.819+08:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
  -----
```

Check if MongoDB and Mongosh are installed

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

-----
  The server generated these startup warnings when booting
  2025-02-13T20:27:33.819+08:00: Access control is not enabled for the database. Read and write access to data and conf
  uration is unrestricted
  -----

test> use ChinookDB
switched to db ChinookDB
ChinookDB> show collections
Album
Artist
Customer
Employee
Genre
Invoice
InvoiceLine
MediaType
Playlist
PlaylistTrack
Track
```

Check the tables inside of the chinook database

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

ChinookDB> show collections
Album
Artist
Customer
Employee
Genre
Invoice
InvoiceLine
MediaType
Playlist
PlaylistTrack
Track
ChinookDB> db.Customer.findOne()
{
  _id: ObjectId('67ade67f82db91f635550da2'),
  CustomerId: 1,
  FirstName: 'Luís',
  LastName: 'Gonçalves',
  Company: 'Embraer - Empresa Brasileira de Aeronáutica S.A.',
  Address: 'Av. Brigadeiro Faria Lima, 2170',
  City: 'São José dos Campos',
  State: 'SP',
  Country: 'Brazil',
  PostalCode: '12227-000',
  Phone: '+55 (12) 3923-5555',
  Fax: '+55 (12) 3923-5566',
  Email: 'luisg@embraer.com.br',
  SupportRepId: 3
}
ChinookDB>
```



```
pymongo_test.py • mongo_test2.py X
D: > LR5 > chinookdb > mongo_test2.py > ...
1  import sqlite3
2  from pymongo import MongoClient
3
4  # Connect to SQLite database
5  sqlite_conn = sqlite3.connect("C:\\chinookdb\\chinook.db") # Update path if needed
6  sqlite_cursor = sqlite_conn.cursor()
7
8  # Connect to MongoDB
9  mongo_client = MongoClient("mongodb://localhost:27017/")
10 mongo_db = mongo_client["ChinookDB"]
11
12 # Get all table names from SQLite
13 sqlite_cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
14 tables = [table[0] for table in sqlite_cursor.fetchall()]
15
16 for table in tables:
17     print(f"Importing {table}...")
18
19     # Fetch all rows from the table
20     sqlite_cursor.execute(f"SELECT * FROM {table}")
21     rows = sqlite_cursor.fetchall()
22
23     # Get column names
24     column_names = [desc[0] for desc in sqlite_cursor.description]
25
26     # Convert rows to dictionaries
27     documents = [dict(zip(column_names, row)) for row in rows]
28
29     # Insert into MongoDB
30     if documents:
31         mongo_db[table].insert_many(documents)
32         print(f"Inserted {len(documents)} records into {table}")
33
34 # Close connections
35 sqlite_conn.close()
36 mongo_client.close()
37
```

Importing chinook sqlite database file using VS Code

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS (PREVIEW) SQL CONSOLE

Inserted 8 records into Employee
Importing Genre...
Inserted 25 records into Genre
Importing Invoice...
Inserted 412 records into Invoice
Importing InvoiceLine...
Inserted 2240 records into InvoiceLine
Importing MediaType...
Inserted 5 records into MediaType
Importing Playlist...
Inserted 18 records into Playlist
Importing PlaylistTrack...
Inserted 8715 records into PlaylistTrack
Importing Track...
Inserted 3503 records into Track
Data import completed successfully!
PS C:\Users\itski>
0 Δ 0 Connect
```

```
pymongo_test.py X mongo_test2.py
C:\Users\itski> pymongo_test.py > ...
1 from pymongo import MongoClient
2
3 import pprint
4
5 import re
6
7 # client = MongoClient(host="localhost", port=27017)
8 client = MongoClient("mongodb://localhost:27017/")
9
10 # Get reference to 'chinook' database
11 db = client["chinookDB"]
12 |
13 # Get a reference to the 'customers' collection
14 customers_collection = db["customer"]
15 # print(customers_collection)
16
17 #print all documents
18 doc1 = customers_collection.find_one()
19 print(doc1)
20
21 client.close()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS (PREVIEW) SQL CONSOLE

SoftwareFoundation.Python.3.13_qbz5n2kfra8p0/python.exe c:/Users/itski/pymongo_test.py
None
PS C:\Users\itski> & C:/Users/itski/AppData/Local/Microsoft/WindowsApps/PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0/python.exe c:/Users/itski/pymongo_test.py
{'_id': ObjectId('67ade67f82db91f635550da2'), 'CustomerId': 1, 'FirstName': 'Luis', 'LastName': 'Gonçalves', 'Company': 'Embraer - Empresa Brasileira de Aeronáutica S.A.', 'Address': 'A
v. Brigadeiro Faria Lima, 2170', 'City': 'São José dos Campos', 'State': 'SP', 'Country': 'Brazil', 'PostalCode': '12227-000', 'Phone': '+55 (12) 3923-5555', 'Fax': '+55 (12) 3923-5566'
, 'Email': 'luisg@embraer.com.br', 'SupportRepId': 3}
PS C:\Users\itski>
```

Print all documents of the customer's collection

```
pymongo_test.py • mongo_test2.py
C:\Users\itski > pymongo_test.py > ...

4
5 import re
6
7 # client = MongoClient(host="localhost", port=27017)
8 client = MongoClient("mongodb://localhost:27017/")
9
10 # Get reference to 'chinook' database
11 db = client["chinookDB"]
12
13 # Get a reference to the 'customers' collection
14 customers_collection = db["Customer"]
15 # print(customers_collection)
16
17 #print first document
18 # doc1 = customers_collection.find_one()
19 # print(doc1)
20
21 #print all documents
22 for all_doc in customers_collection.find():
23     print(all_doc)
24
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS QUERY RESULTS (PREVIEW) SQL CONSOLE

```
{ '_id': ObjectId('67ade67f82db91f635550dd0'), 'CustomerId': 47, 'FirstName': 'Lucas', 'LastName': 'Mancini', 'Company': None, 'Address': 'Via Degli Scipioni, 43', 'City': 'Rome', 'State': 'RM', 'Country': 'Italy', 'PostalCode': '00192', 'Phone': '+39 06 39733434', 'Fax': None, 'Email': 'lucas.mancini@yahoo.it', 'SupportRepId': 5}
{'_id': ObjectId('67ade67f82db91f635550dd1'), 'CustomerId': 48, 'FirstName': 'Johannes', 'LastName': 'Van der Berg', 'Company': None, 'Address': 'Lijnbaansgracht 120bg', 'City': 'Amsterdam', 'State': 'VW', 'Country': 'Netherlands', 'PostalCode': '1016', 'Phone': '+31 020 6223130', 'Fax': None, 'Email': 'johavanderberg@yahoo.nl', 'SupportRepId': 5}
{'_id': ObjectId('67ade67f82db91f635550dd2'), 'CustomerId': 49, 'FirstName': 'Stanisław', 'LastName': 'Wójcik', 'Company': None, 'Address': 'Ordynacka 10', 'City': 'Warsaw', 'State': 'No', 'Country': 'Poland', 'PostalCode': '00-358', 'Phone': '+48 22 828 37 39', 'Fax': None, 'Email': 'stanislaw.wojcik@wp.pl', 'SupportRepId': 4}
{'_id': ObjectId('67ade67f82db91f635550dd3'), 'CustomerId': 50, 'FirstName': 'Enrique', 'LastName': 'Muñoz', 'Company': None, 'Address': 'C/ San Bernardo 85', 'City': 'Madrid', 'State': 'None', 'Country': 'Spain', 'PostalCode': '28015', 'Phone': '+34 914 454 454', 'Fax': None, 'Email': 'enrique.munoz@yahoo.es', 'SupportRepId': 5}
{'_id': ObjectId('67ade67f82db91f635550dd4'), 'CustomerId': 51, 'FirstName': 'Joakim', 'LastName': 'Johansson', 'Company': None, 'Address': 'Celsiusg. 9', 'City': 'Stockholm', 'State': 'None', 'Country': 'Sweden', 'PostalCode': '11230', 'Phone': '+46 08-651 52 52', 'Fax': None, 'Email': 'joakim.johansson@yahoo.se', 'SupportRepId': 5}
{'_id': ObjectId('67ade67f82db91f635550ddb'), 'CustomerId': 58, 'FirstName': 'Manoj', 'LastName': 'Pareek', 'Company': None, 'Address': '12,Community Centre', 'City': 'Delhi', 'State': 'None', 'Country': 'India', 'PostalCode': '110017', 'Phone': '+91 0124 39883988', 'Fax': None, 'Email': 'manoj.pareek@rediff.com', 'SupportRepId': 3}
{'_id': ObjectId('67ade67f82db91f635550ddc'), 'CustomerId': 59, 'FirstName': 'Puja', 'LastName': 'Srivastava', 'Company': None, 'Address': '3,Raj Bhavan Road', 'City': 'Bangalore', 'State': 'None', 'Country': 'India', 'PostalCode': '560001', 'Phone': '+91 080 22289999', 'Fax': None, 'Email': 'puja.srivastava@yahoo.in', 'SupportRepId': 3}
```

Printing all documents

```
pymongo_test.py X mongo_test2.py
C: > Users > itski > pymongo_test.py > ...
14 customers_collection = db[ 'customer' ]
15 # print(customers_collection)
16
17 #print first document
18 # doc1 = customers_collection.find_one()
19 # print(doc1)
20
21 #print all documents
22 for all_doc in customers_collection.find():
23     print(all_doc)
24
25 #return only the LastName and FirstName
26 for rec in customers_collection.find({}, {"_id":0,"LastName": 1, "FirstName": 1}):
27     print(rec)
28
29 client.close()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS (PREVIEW) SQL CONSOLE

```
{'FirstName': 'Kara', 'LastName': 'Nielsen'}
{'FirstName': 'Eduardo', 'LastName': 'Martins'}
{'FirstName': 'Alexandre', 'LastName': 'Rocha'}
{'FirstName': 'Roberto', 'LastName': 'Almeida'}
{'FirstName': 'Fernanda', 'LastName': 'Ramos'}
{'FirstName': 'Mark', 'LastName': 'Philips'}
{'FirstName': 'Jennifer', 'LastName': 'Peterson'}
{'FirstName': 'Frank', 'LastName': 'Harris'}
{'FirstName': 'Jack', 'LastName': 'Smith'}
{'FirstName': 'Michelle', 'LastName': 'Brooks'}
{'FirstName': 'Tim', 'LastName': 'Goyer'}
{'FirstName': 'Dan', 'LastName': 'Miller'}
{'FirstName': 'Kathy', 'LastName': 'Chase'}
{'FirstName': 'Heather', 'LastName': 'Leacock'}
{'FirstName': 'John', 'LastName': 'Gordon'}
{'FirstName': 'Frank', 'LastName': 'Ralston'}
{'FirstName': 'Victor', 'LastName': 'Stevens'}
{'FirstName': 'Richard', 'LastName': 'Cunningham'}
{'FirstName': 'Patrick', 'LastName': 'Gray'}
{'FirstName': 'Julia', 'LastName': 'Barnett'}
```

0 ^ 0 Connect

Return only the last name and first name

```
pymongo_test.py X mongo_test2.py
C: > Users > itski > pymongo_test.py > ...
22 for all_doc in customers_collection.find():
23     print(all_doc)
24
25 #return only the LastName and FirstName
26 for rec in customers_collection.find({}, {"_id":0, "LastName": 1, "FirstName": 1}):
27     print(rec)
28
29 #Print all customers with LastName that starts with "G"
30
31 rgx = re.compile('^G.*?$', re.IGNORECASE) # compile the regex
32 cursor = customers_collection.find({"LastName":rgx})
33 num_docs = 0
34 for document in cursor:
35     num_docs += 1
36     pprint.pprint(document)
37     print()
38 print("# of documents found: " + str(num_docs))
39
40 client.close()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS (PREVIEW) SQL CONSOLE

{'Address': '307 Macacha Güemes',
 'City': 'Buenos Aires',
 'Company': None,
 '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('67ade67f82db91f635550dd9')}

# of documents found: 7
PS C:\Users\itski>
```

Print all customers with LastName that starts with “G”

```
C: > Users > itski > py mongo_test.py > ...
22 for all_doc in customers_collection.find():
23     print(all_doc)
24
25 #return only the LastName and FirstName
26 for rec in customers_collection.find({}, {"_id":0,"LastName": 1, "FirstName": 1}):
27     print(rec)
28
29 #Print all customers with LastName that starts with "G"
30
31 rgx = re.compile('^Go.*?$', re.IGNORECASE) # compile the regex
32 cursor = customers_collection.find({"LastName":rgx})
33 num_docs = 0
34 for document in cursor:
35     num_docs += 1
36     pprint.pprint(document)
37     print()
38 print("# of documents found: " + str(num_docs))
39
40 client.close()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS (PREVIEW) SQL CONSOLE

```
{'Address': '69 Salem Street',
'City': 'Boston',
'Company': None,
'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('67ade67f82db91f635550db8')}
```

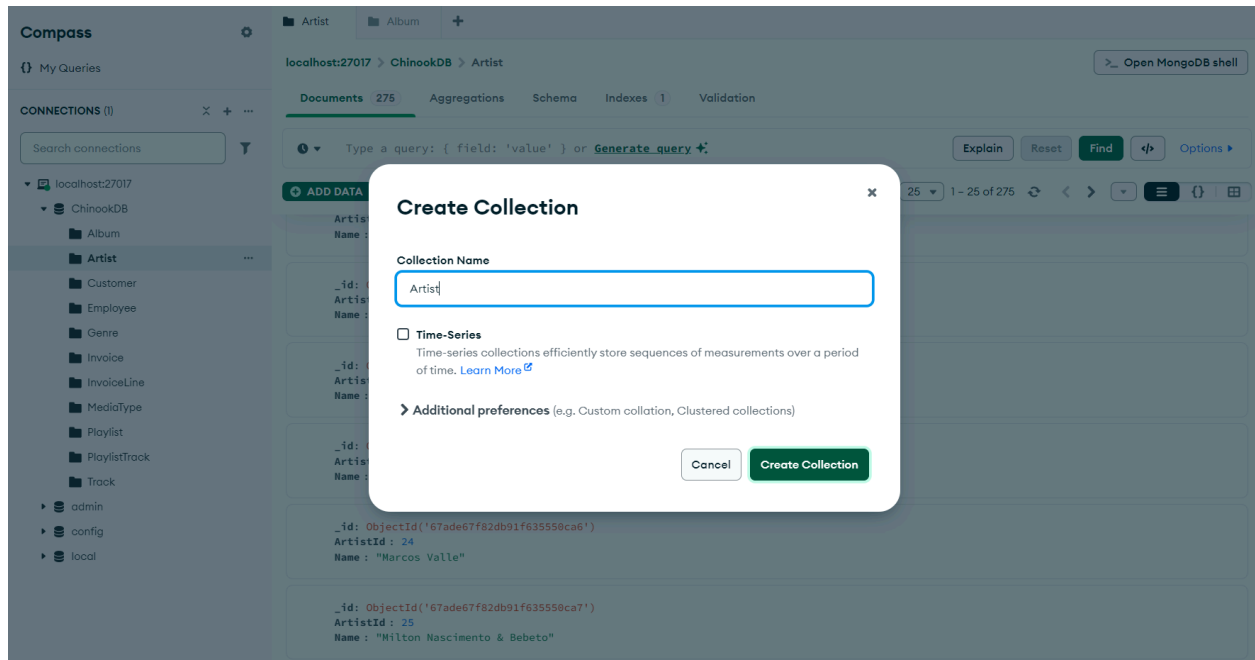
```
# of documents found: 3
PS C:\Users\itski>
```

Print all customers with LastName that starts with "Go"

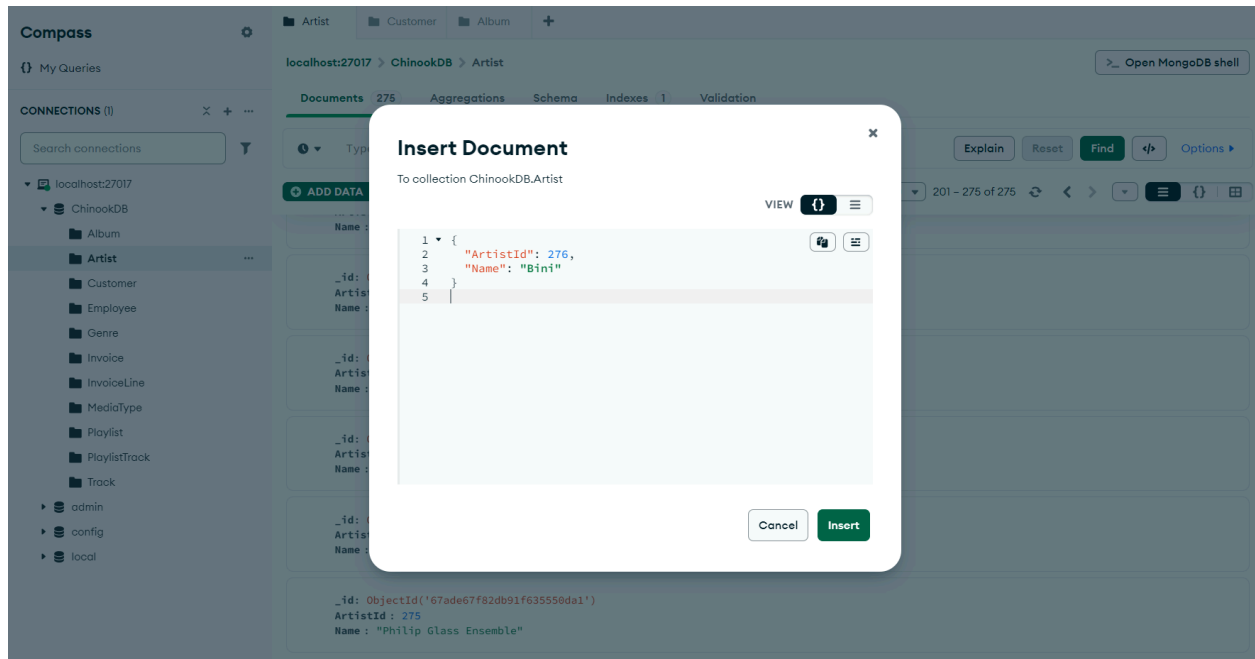
Database Structure Browse Data Edit Pragmas Execute SQL				
Table: Customer				
	CustomerId	FirstName	LastName ▲	Company
	Filter	Filter	Filter	Filter
4	18	Michelle	Brooks	NULL
5	29	Robert	Brown	NULL
6	21	Kathy	Chase	NULL
7	26	Richard	Cunningham	NULL
8	41	Marc	Dubois	NULL
9	34	João	Fernandes	NULL
10	30	Edward	Francis	NULL
11	42	Wyatt	Girard	NULL
12	1	Luís	Gonçalves	Embraer - Empresa Brasileira
13	23	John	Gordon	NULL
14	19	Tim	Goyer	Apple Inc.
15	27	Patrick	Gray	NULL
16	7	Astrid	Gruber	NULL
17	56	Diego	Gutiérrez	NULL
18	4	Bjørn	Hansen	NULL
19	16	Frank	Harris	Google Inc.
20	6	Helena	Holý	NULL
21	53	Phil	Hughes	NULL
22	44	Terhi	Hämäläinen	NULL

PostLab

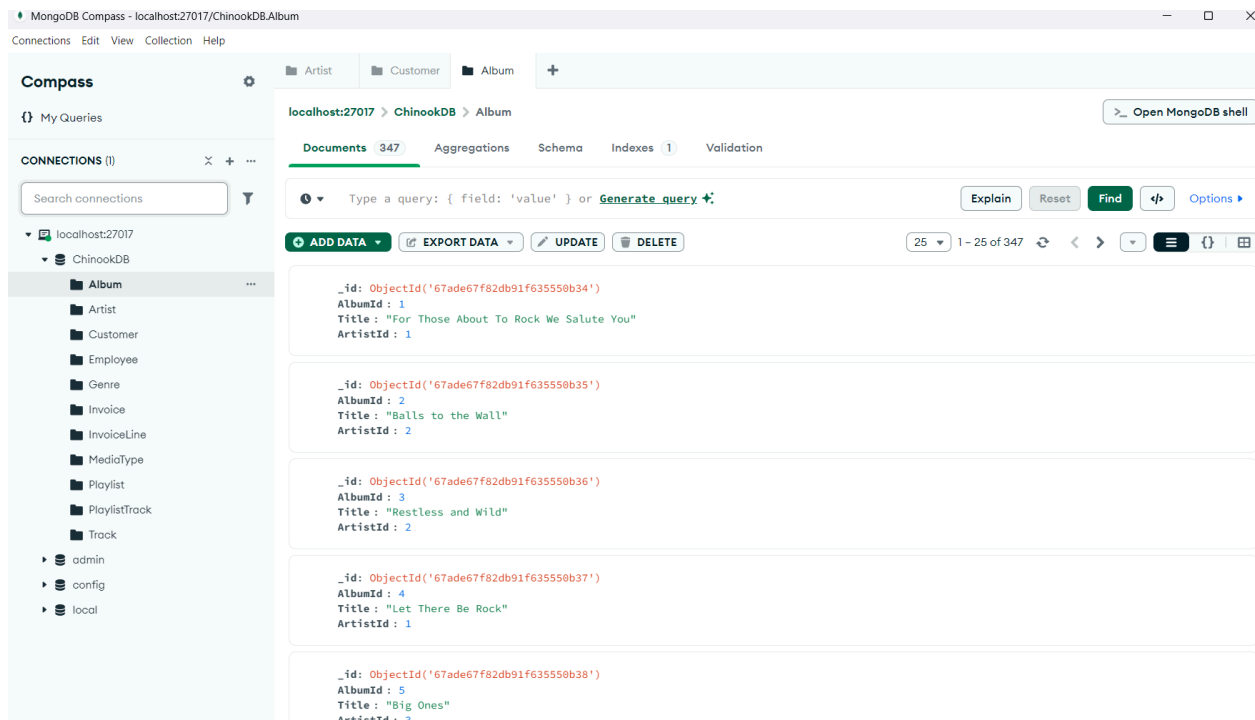
Using the ERD shown here >> Chinook DB ERD , create the artists-albums-tracks database in MongoDB compass



Creating the database “Artist.” The same process was used with creating the collection “Albums” and “Tracks”



Adding sample data to the collection “Artist”



Database “Album”

MongoDB Compass - localhost:27017/ChinookDB.Track

Connections Edit View Collection Help

Compass

{ My Queries

CONNECTIONS (1)

Search connections

localhost:27017

ChinookDB

- Album
- Artist
- Customer
- Employee
- Genre
- Invoice
- InvoiceLine
- MediaType
- Playlist
- PlaylistTrack
- Track

admin

config

local

localhost:27017 > ChinookDB > Track

Documents 3.5K Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

Explain Reset Find Options

ADD DATA EXPORT DATA UPDATE DELETE

100 1 - 100 of 3503

```

_id: ObjectId('67ade67f82db91f635553a7c')
TrackId: 1
Name: "For Those About To Rock (We Salute You)"
AlbumId: 1
MediaTypeId: 1
GenreId: 1
Composer: "Angus Young, Malcolm Young, Brian Johnson"
Milliseconds: 343719
Bytes: 11176334
UnitPrice: 0.99

```

```

_id: ObjectId('67ade67f82db91f635553a7d')
TrackId: 2
Name: "Balls to the Wall"
AlbumId: 2
MediaTypeId: 2
GenreId: 1
Composer: null
Milliseconds: 342562
Bytes: 5510424
UnitPrice: 0.99

```

```

_id: ObjectId('67ade67f82db91f635553a7e')
TrackId: 3
Name: "Fast As a Shark"
AlbumId: 3

```

Database "Track"

Connections Edit View Collection Help

Compass

{ My Queries

CONNECTIONS (1)

Search connections

localhost:27017

ChinookDB

- Album
- Artist
- Customer
- Employee
- Genre
- Invoice
- InvoiceLine
- MediaType
- Playlist
- PlaylistTrack
- Track

admin

config

local

localhost:27017 > ChinookDB > Artist

Documents 276 Aggregations Schema Indexes 1 Validation

{ "ArtistId": 276 }

[Generate query](#) Explain Reset Find Options

ADD DATA EXPORT DATA UPDATE DELETE

100 1 - 1 of 1

```

_id: ObjectId('67adf543e4df5e976b997a8f')
ArtistId: 276
Name: "Bini"

```

Sample Query

Connections Edit View Collection Help

Compass

{ My Queries

CONNECTIONS (1)

Search connections

localhost:27017

ChinookDB

Album

Artist

Customer

Employee

Genre

Invoice

InvoiceLine

MediaType

Playlist

PlaylistTrack

Track

admin

config

local

Artist

Customer

Track

localhost:27017 > ChinookDB > Artist

Documents 276 Aggregations Schema Indexes 1 Validation

Generate aggregation Explain Export Run Options

Untitled - modified SAVE CREATE NEW EXPORT TO LANGUAGE PREVIEW STAGES TEXT WIZARD

Stage 1 \$match

```
1 { "Name": "Bini" }
```

Output after \$match stage (Sample of 1 document)

```
{
  "_id": ObjectId('67adf543e4df5e976b997a8f'),
  "ArtistId": 276,
  "Name": "Bini"
}
```

+ Add Stage

[Learn more about aggregation pipeline stages](#)

Data relationship verification: Aggregation query (Stage 1)

Connections Edit View Collection Help

Compass

{ My Queries

CONNECTIONS (1)

Search connections

localhost:27017

ChinookDB

Album

Artist

Customer

Employee

Genre

Invoice

InvoiceLine

MediaType

Playlist

PlaylistTrack

Track

admin

config

local

Artist

Customer

Track

localhost:27017 > ChinookDB > Artist

Documents 276 Aggregations Schema Indexes 1 Validation

Generate aggregation Explain Export Run Options

Untitled - modified SAVE CREATE NEW EXPORT TO LANGUAGE PREVIEW STAGES TEXT WIZARD

Stage 2 \$lookup

```
1 {
2   $from: "albums",
3   $localField: "ArtistId",
4   $foreignField: "ArtistId",
5   $as: "Albums"
6 }
7
```

Output after \$lookup stage (Sample of 1 document)

```
{
  "_id": ObjectId('67adf543e4df5e976b997a8f'),
  "ArtistId": 276,
  "Name": "Bini",
  "Albums": Array (empty)
}
```

+ Add Stage

[Learn more about aggregation pipeline stages](#)

Data relationship verification: Aggregation query (Stage 2)