5602201

7. MongoDB Practice: Part I

CHUTIPORN ANUTARIYA

mongoDB

Getting Started

WHAT AND WHY

mongoDB

- MongoDB was first released in **2009** and has been regularly updated since.
- It is a popular **NoSQL document-oriented database**.
- This means that data entries in the database are stored inside documents within collections.
- Sores data in **JSON-like documents** (BSON internally).
- **Fields can vary** from document to document and data structure can be changed over time
- Distributed database, high availability, horizontal scaling, and geographic distribution
- Scalability
    - Performance Scale: Sustaining 100,000+ database read and writes per second while maintaining strict latency SLAs
    - Data Scale: Storing 1 billion+ documents in the database
    - Cluster Scale: Distributing the database across 100+ nodes, in multiple data centers

MongoDB Data Modeling : Data

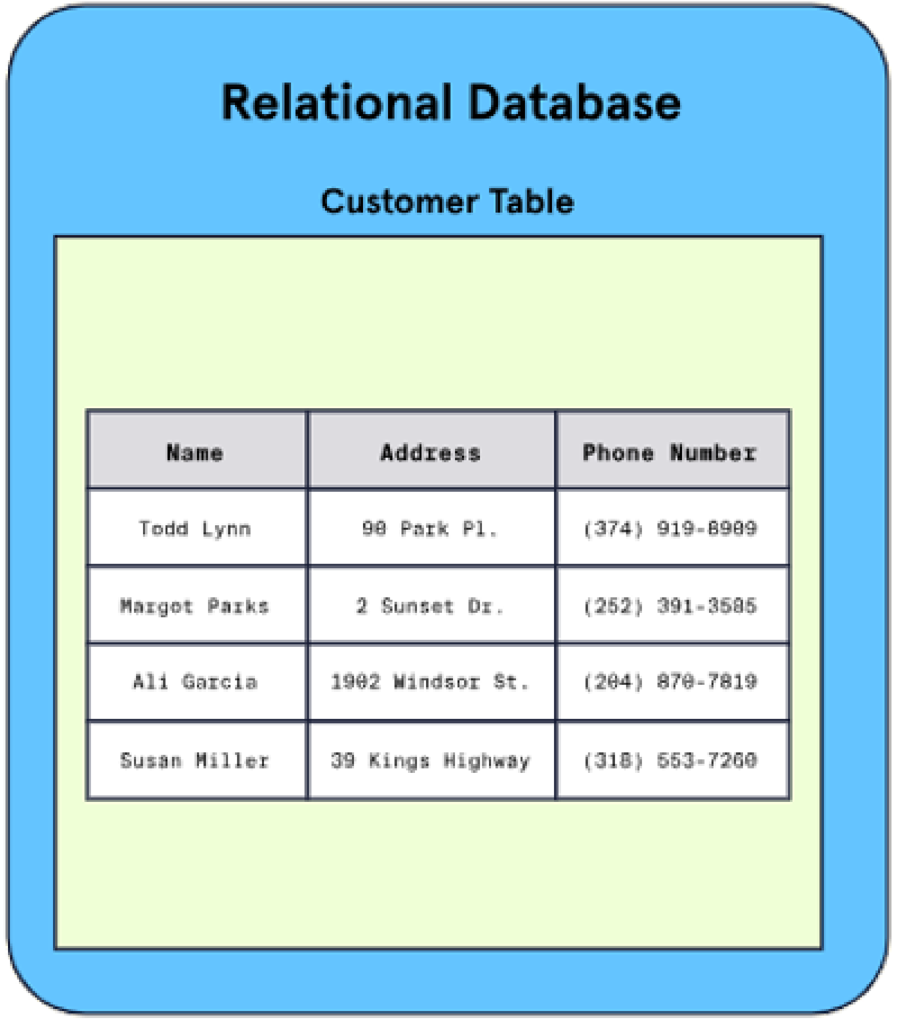
JSON

```
{
  "name": "Rodney",
  "occupation": "photographer",
  "years_of_experience": 7
}
```

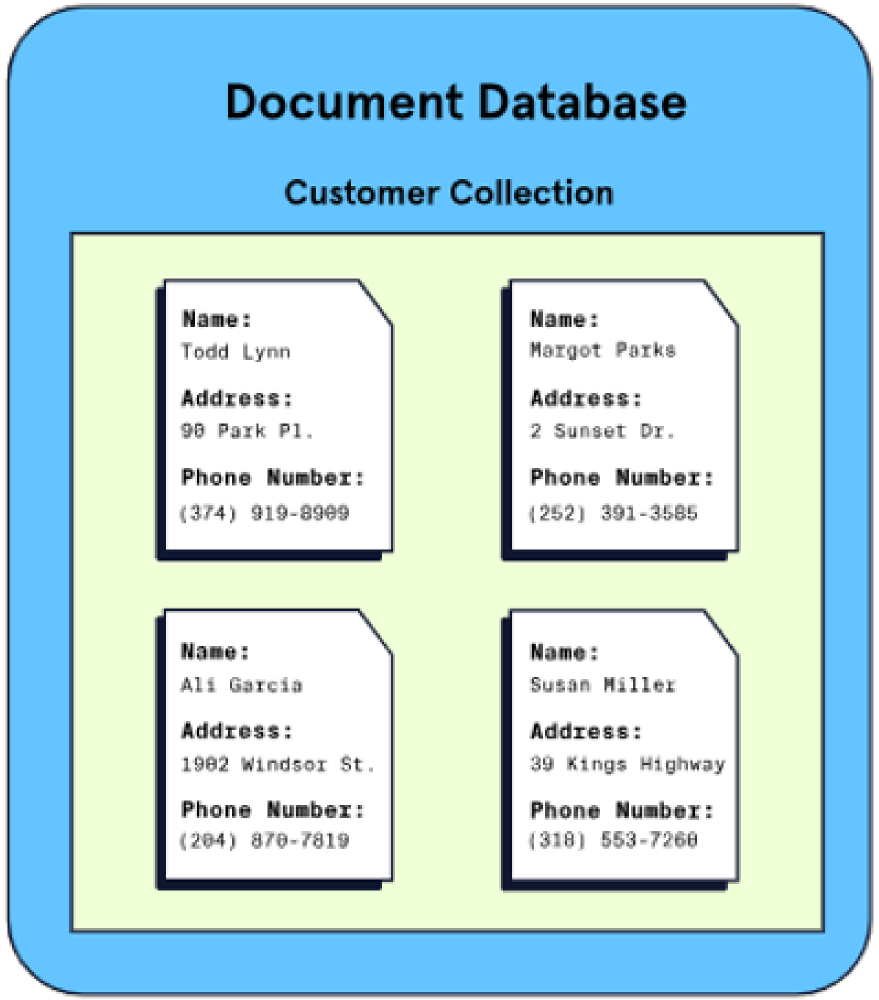
BSON

```
\x00\x00\x00\x02name\x00\a\x00\x00\x00Rodney\x00\x02occupation\x00\r\x00\x00\x
00photographer\x00\x10year_of_experience\x00\a\x00\x00\x00\x00
```

Example: Store Customer information



Name	Address	Phone Number
Todd Lynn	90 Park Pl.	(374) 919-8909
Margot Parks	2 Sunset Dr.	(252) 391-3585
Ali Garcia	1902 Windsor St.	(204) 870-7819
Susan Miller	39 Kings Highway	(318) 553-7260



Name: Todd Lynn
Address: 90 Park Pl.
Phone Number: (374) 919-8909

Name: Margot Parks
Address: 2 Sunset Dr.
Phone Number: (252) 391-3585

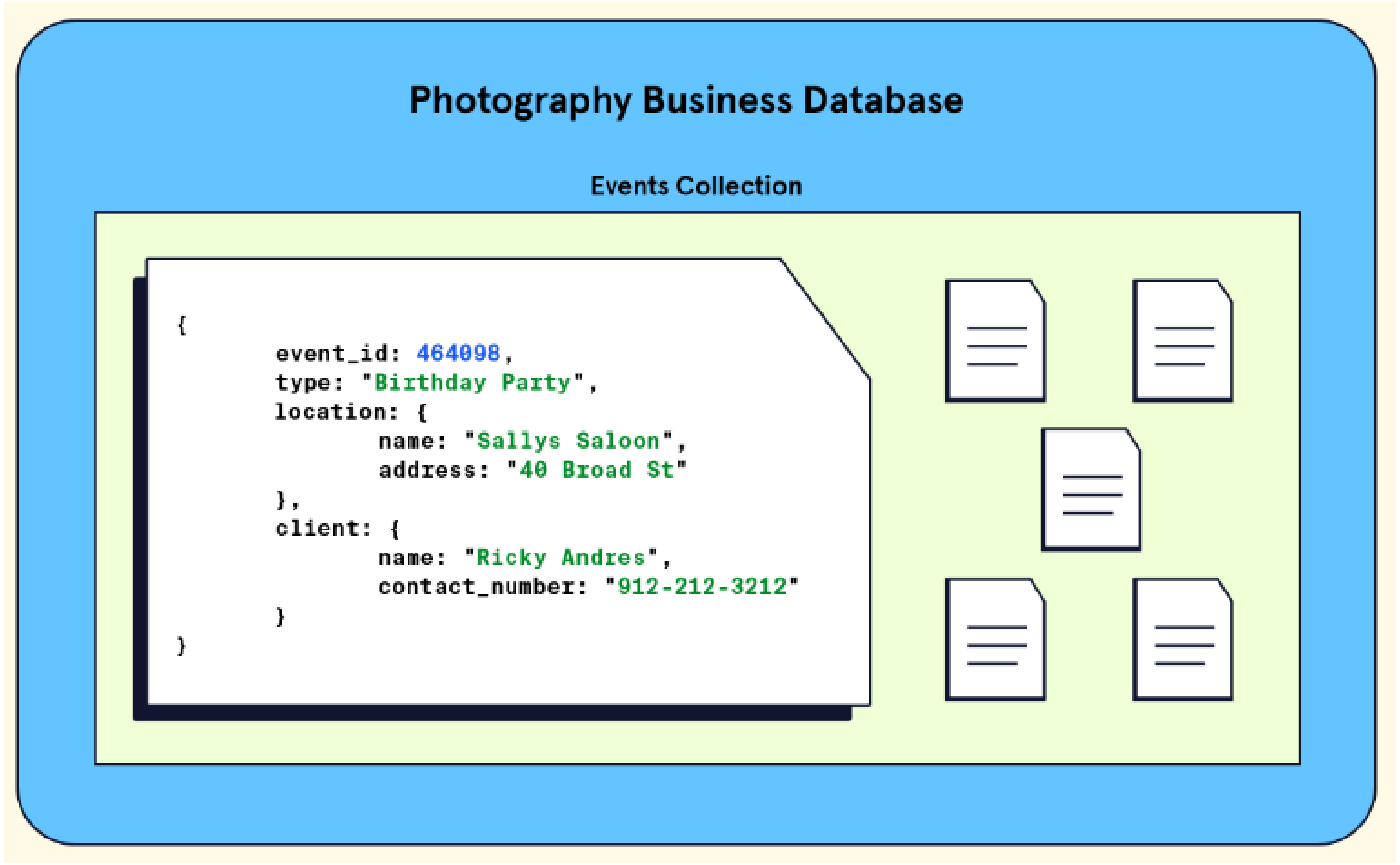
Name: Ali Garcia
Address: 1902 Windsor St.
Phone Number: (204) 870-7819

Name: Susan Miller
Address: 39 Kings Highway
Phone Number: (318) 553-7260

MongoDB Data Modeling : Relationship

- Representing relationships between data in MongoDB:
  - Embedded documents
  - References
- Example: photography business
  - *the event name*
  - *the location*
  - *the client’s name*

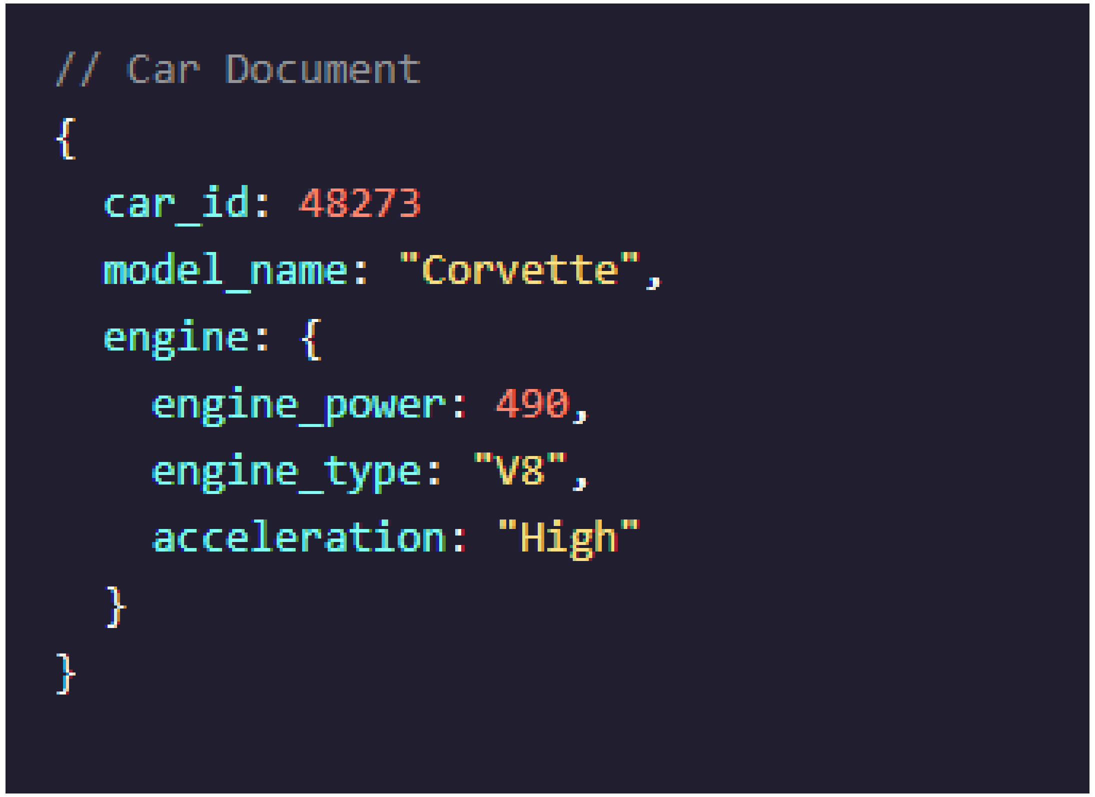
Embedded



```
{
        event_id: 464098,
        type: "Birthday Party",
        location: {
                name: "Sallys Saloon",
                address: "40 Broad St"
        },
        client: {
                name: "Ricky Andres",
                contact_number: "912-212-3212"
        }
}
```

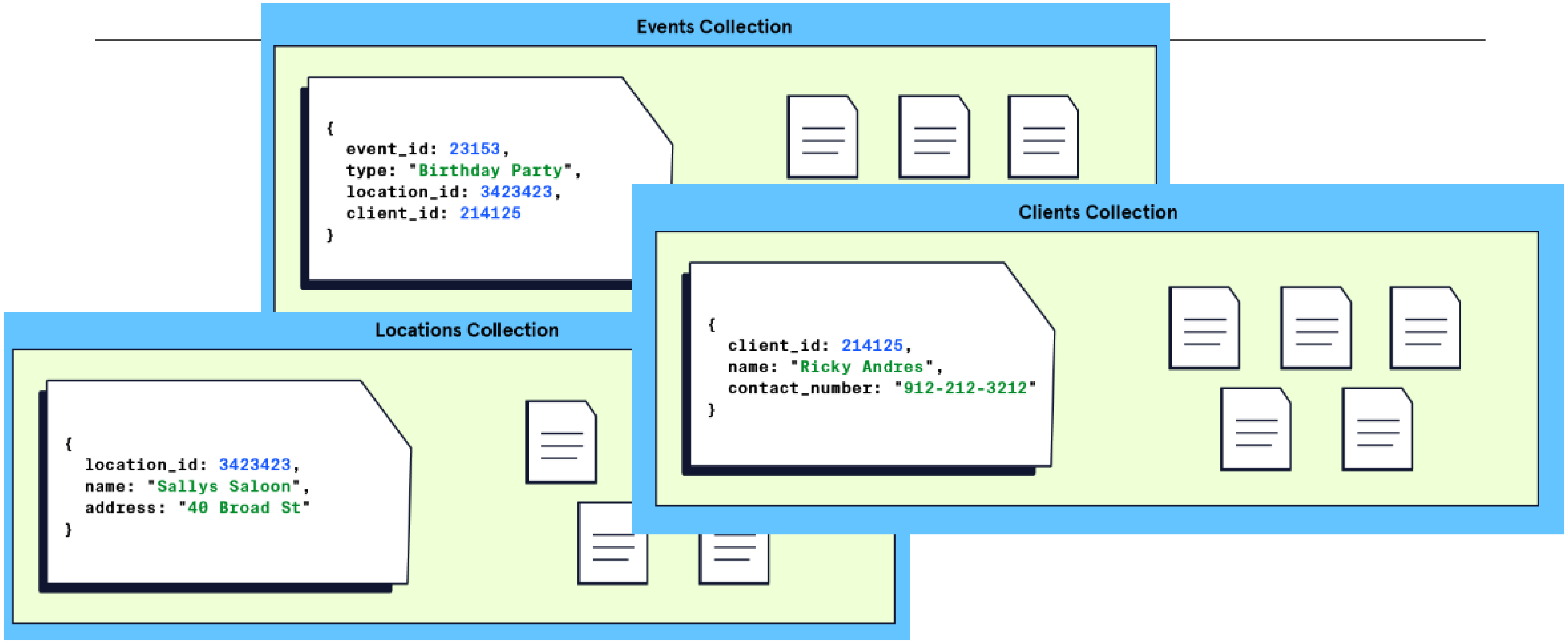
Embedded

- one-to-one relationship : a car and its unique license plate.
- one-to-many relationship: a car owner and their multiple-owned cars.



```
// Car Document
{
  car_id: 48273
  model_name: "Corvette",
  engine: {
    engine_power: 490,
    engine_type: "V8",
    acceleration: "High"
  }
}
```

Referencing



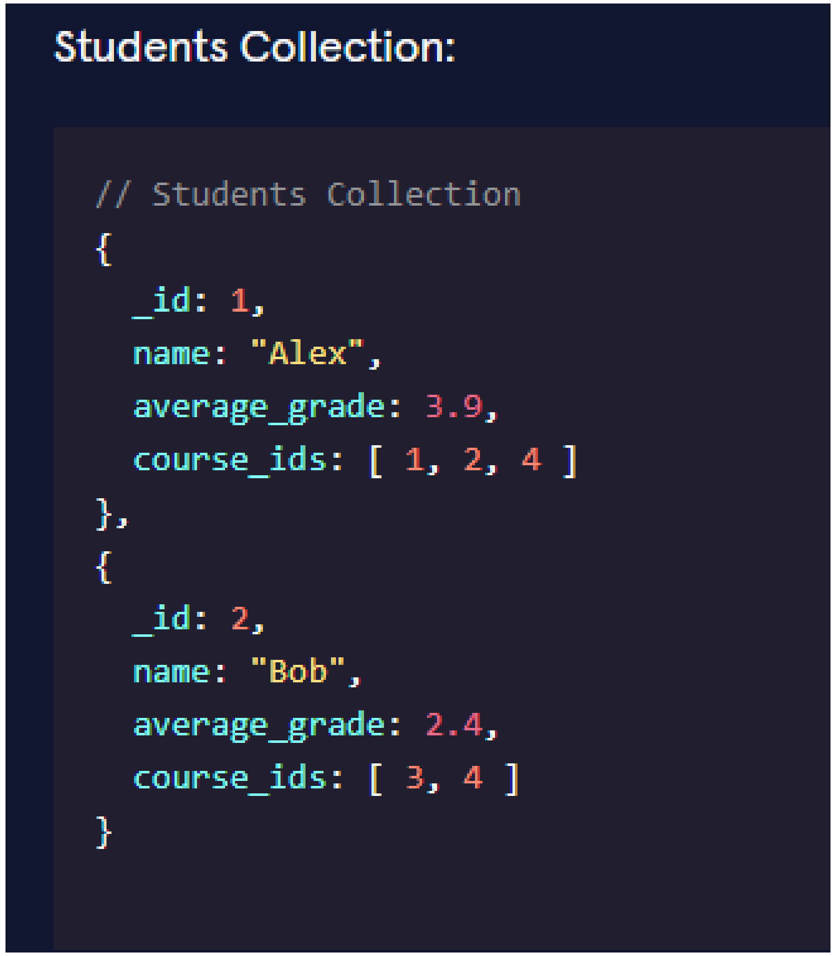
```
{
  event_id: 23153,
  type: "Birthday Party",
  location_id: 3423423,
  client_id: 214125
}
```

```
{
  client_id: 214125,
  name: "Ricky Andres",
  contact_number: "912-212-3212"
}
```

```
{
  location_id: 3423423,
  name: "Sallys Saloon",
  address: "40 Broad St"
}
```

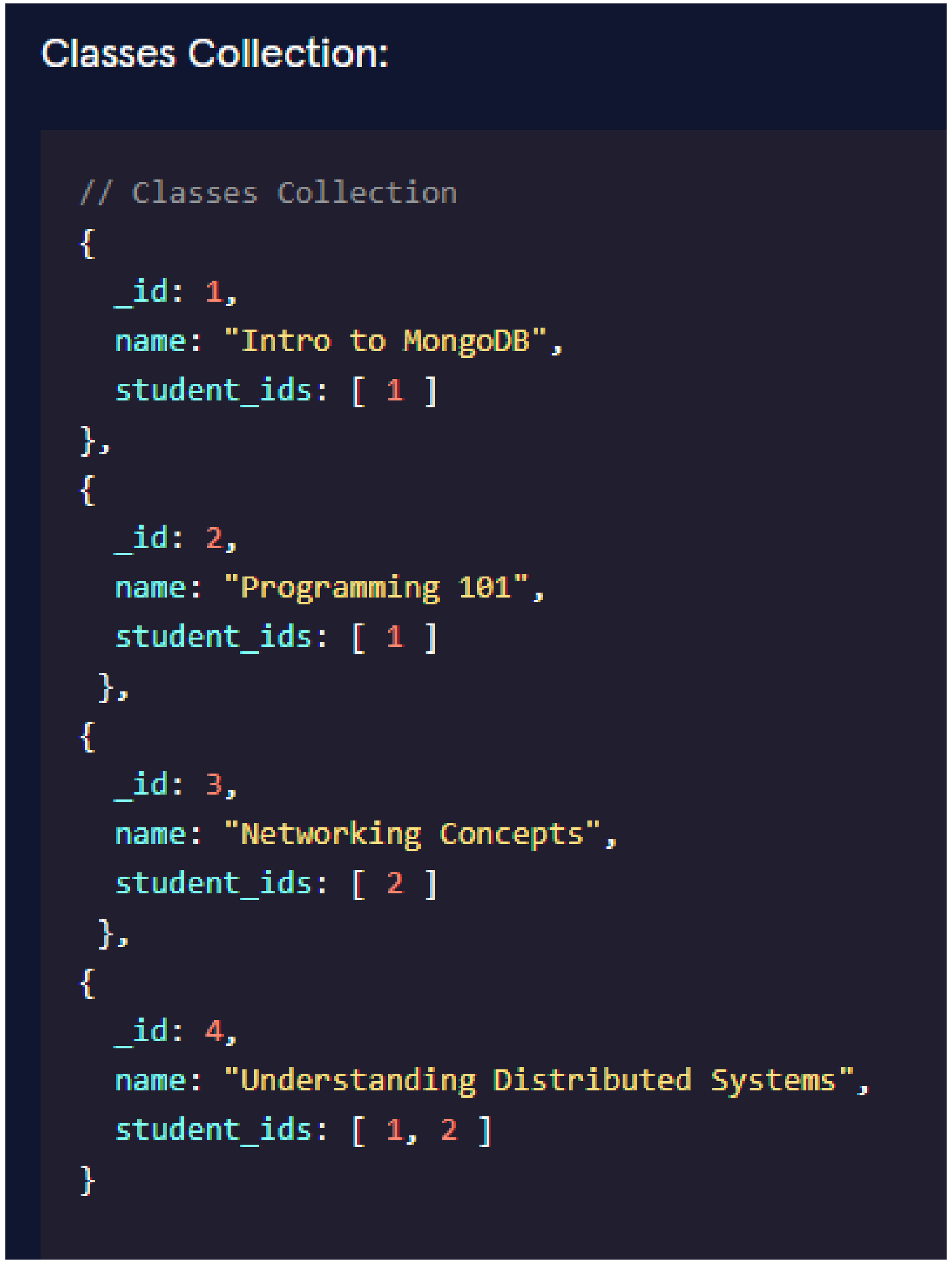
References

- many-to-many : Students and Courses



Students Collection:

```
// Students Collection
{
  _id: 1,
  name: "Alex",
  average_grade: 3.9,
  course_ids: [ 1, 2, 4 ]
},
{
  _id: 2,
  name: "Bob",
  average_grade: 2.4,
  course_ids: [ 3, 4 ]
}
```



Classes Collection:

```
// Classes Collection
{
  _id: 1,
  name: "Intro to MongoDB",
  student_ids: [ 1 ]
},
{
  _id: 2,
  name: "Programming 101",
  student_ids: [ 1 ]
 },
{
  _id: 3,
  name: "Networking Concepts",
  student_ids: [ 2 ]
 },
{
  _id: 4,
  name: "Understanding Distributed Systems",
  student_ids: [ 1, 2 ]
}
```

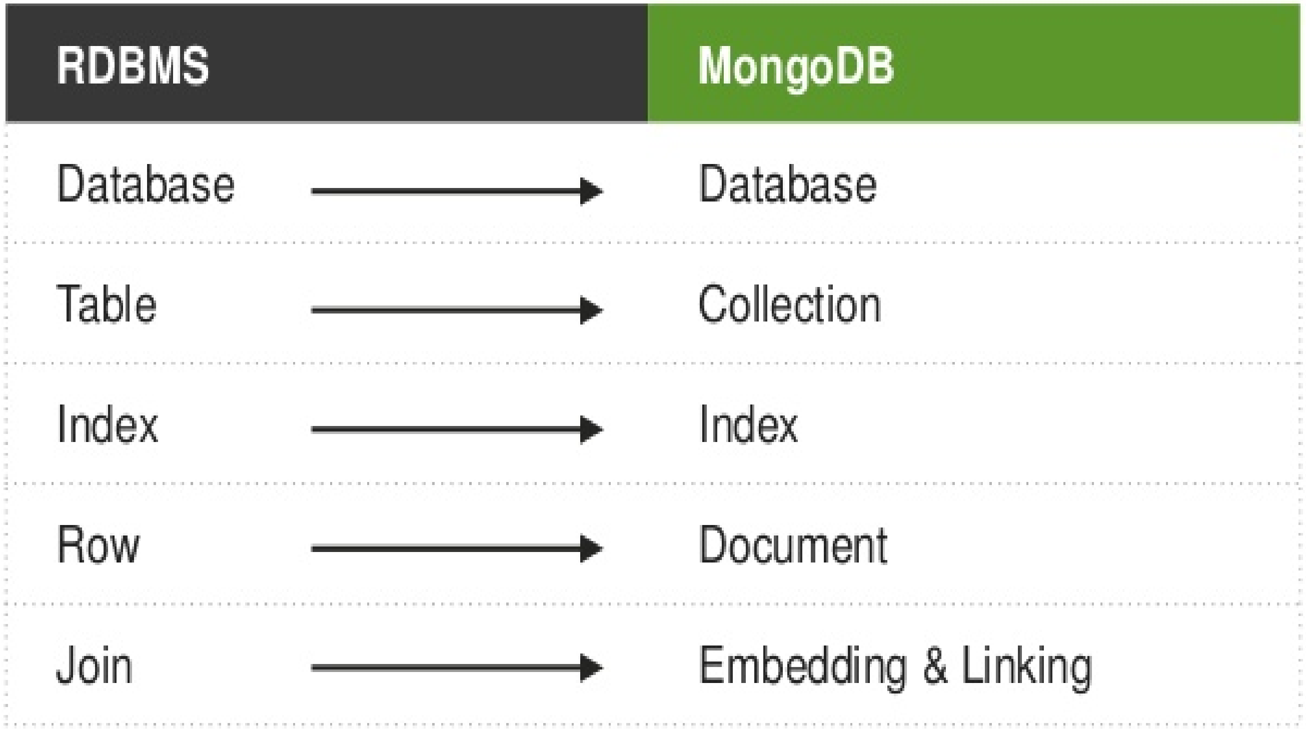
Embedded or References ??

Trade-off between performance and data integrity.

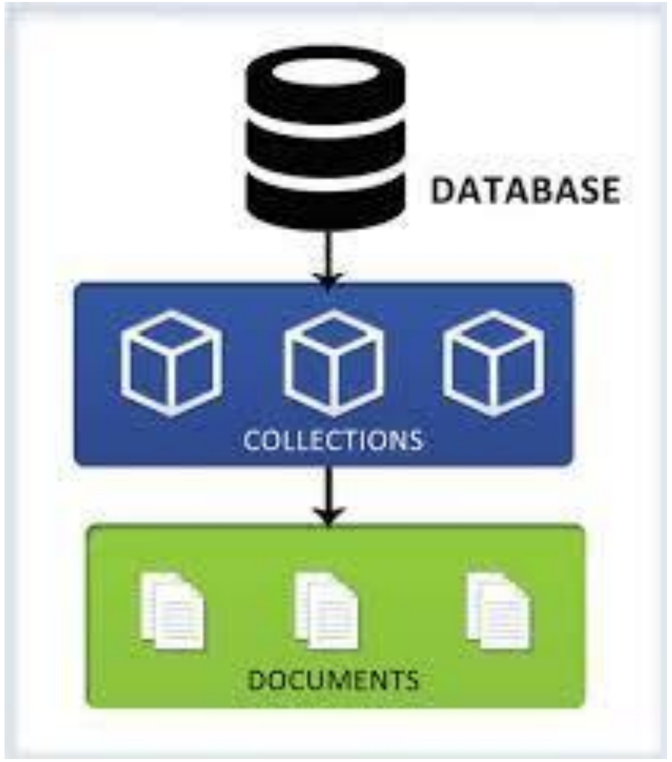
- References (Normalized Model):
  - Better **data integrity**.
  - Longer query times (due to needing data from multiple collections).
- **Embedded (DENormalized Model):**
  - Better **performance.**
  - Weaker **data integrity** because of Duplicated data

When to Use:

- **Embed** when data is accessed together frequently.
- **Reference** when documents are large or accessed separately.



RDBMS		MongoDB
Database	→	Database
Table	→	Collection
Index	→	Index
Row	→	Document
Join	→	Embedding & Linking



MongoDB Terminology

MongoDB Hierarchy

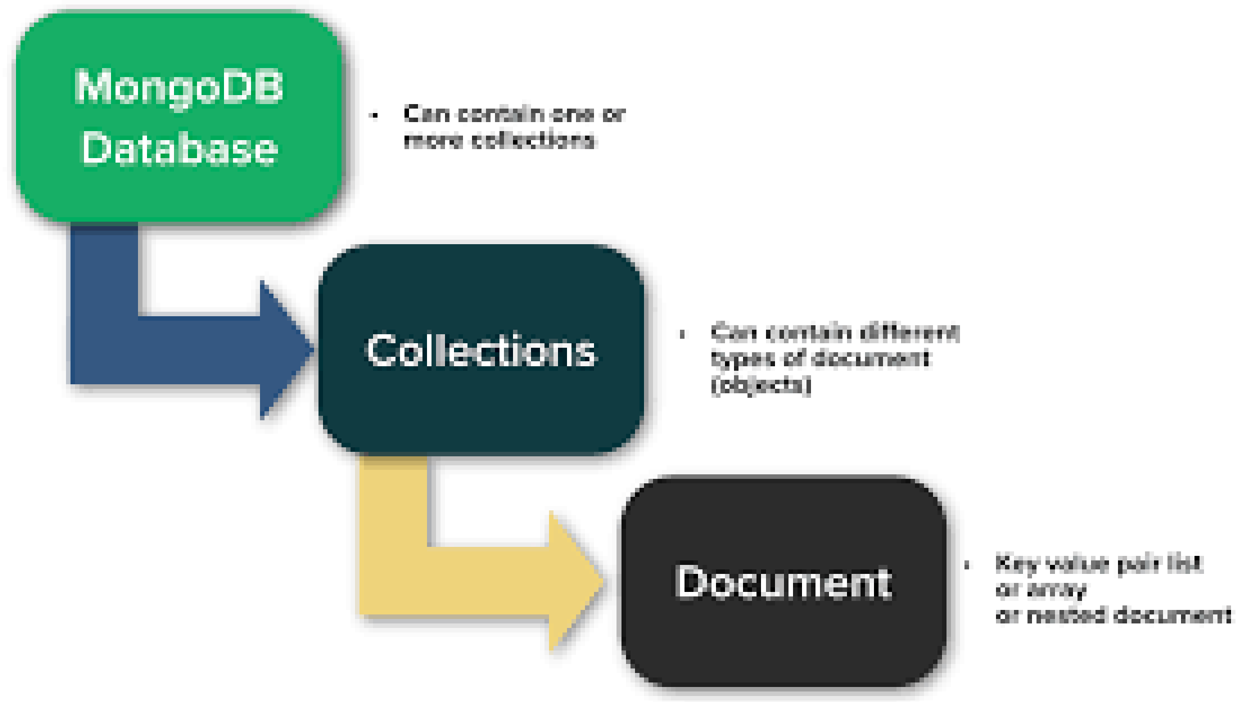
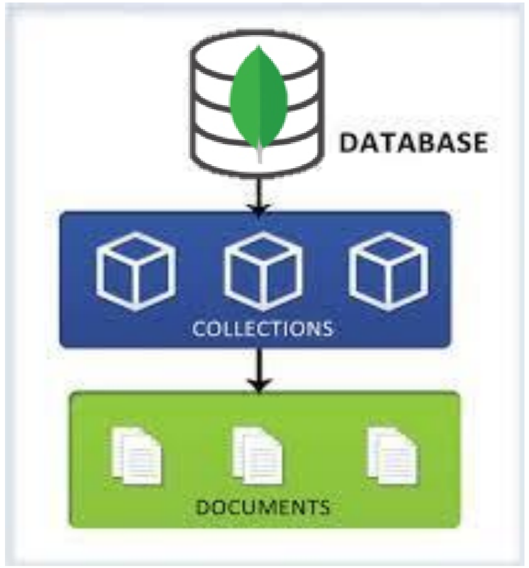

MongoDB Database
• Can contain one or more collections
Collections
• Can contain different types of document (objects)
Document
• Key value pair list or array or nested document

Figure: https://studio3t.com/academy/lessons/mongodb-basics/

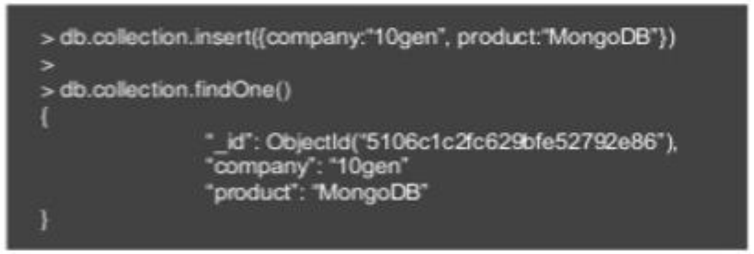
MongoDB Architecture

MongoDB Cloud : Atlas



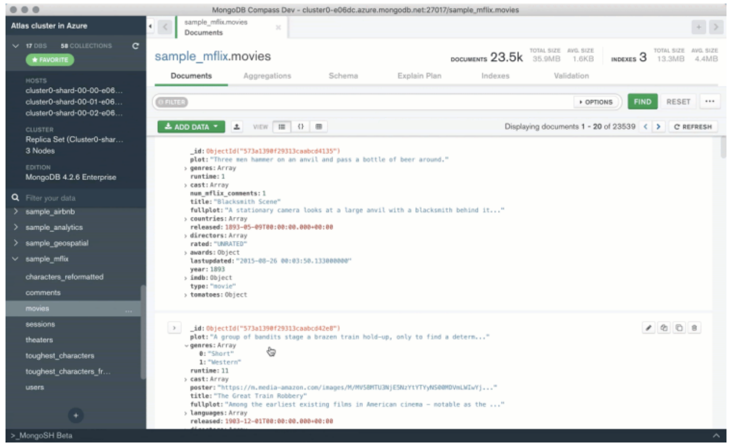
MongoDB Client

- MongoDB Shell



```
> db.collection.insert({company:"10gen", product:"MongoDB"})
>
> db.collection.findOne()
{
                "_id": ObjectId("5106c1c2fc629bfe52792e86"),
                "company": "10gen"
                "product": "MongoDB"
}
```

- MongoDB Compass

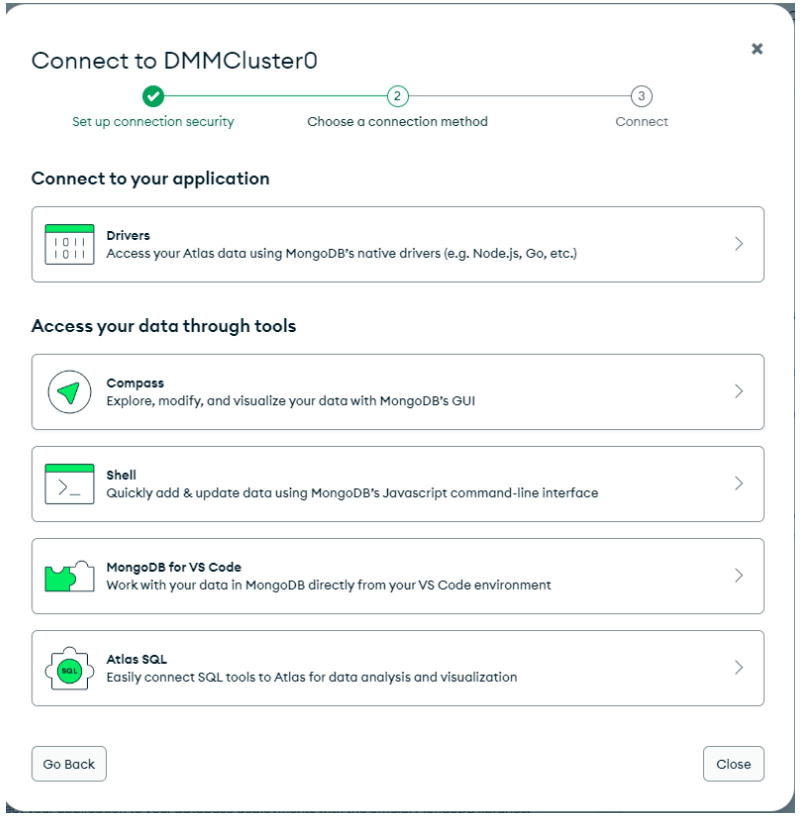


MongoDB Compass Dev - cluster0-e06dc.azure.mongodb.net:27017/sample_mflix.movies

```
_id: ObjectId("573a1390f29313caabcd4135")
plot: "Three men hammer on an anvil and pass a bottle of beer around."
> genres: Array
runtime: 1
> cast: Array
num_mflix_comments: 1
title: "Blacksmith Scene"
fullplot: "A stationary camera looks at a large anvil with a blacksmith behind it..."
> countries: Array
released: 1893-05-09T00:00:00.000+00:00
> directors: Array
rated: "UNRATED"
> awards: Object
lastupdated: "2015-08-26 00:03:50.133000000"
year: 1893
> imdb: Object
type: "movie"
> tomatoes: Object

_id: ObjectId("573a1390f29313caabcd42e8")
plot: "A group of bandits stage a brazen train hold-up, only to find a determ..."
v genres: Array
    0: "Short"
    1: "Western"
runtime: 11
> cast: Array
poster: "https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYyNS00MDVmLWIwYj..."
title: "The Great Train Robbery"
fullplot: "Among the earliest existing films in American cinema - notable as the ..."
> languages: Array
released: 1903-12-01T00:00:00.000+00:00
```

- etc.



Connect to DMMCluster0

Connect to your application

Drivers
Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)

Access your data through tools

Compass
Explore, modify, and visualize your data with MongoDB's GUI

Shell
Quickly add & update data using MongoDB's Javascript command-line interface

MongoDB for VS Code
Work with your data in MongoDB directly from your VS Code environment

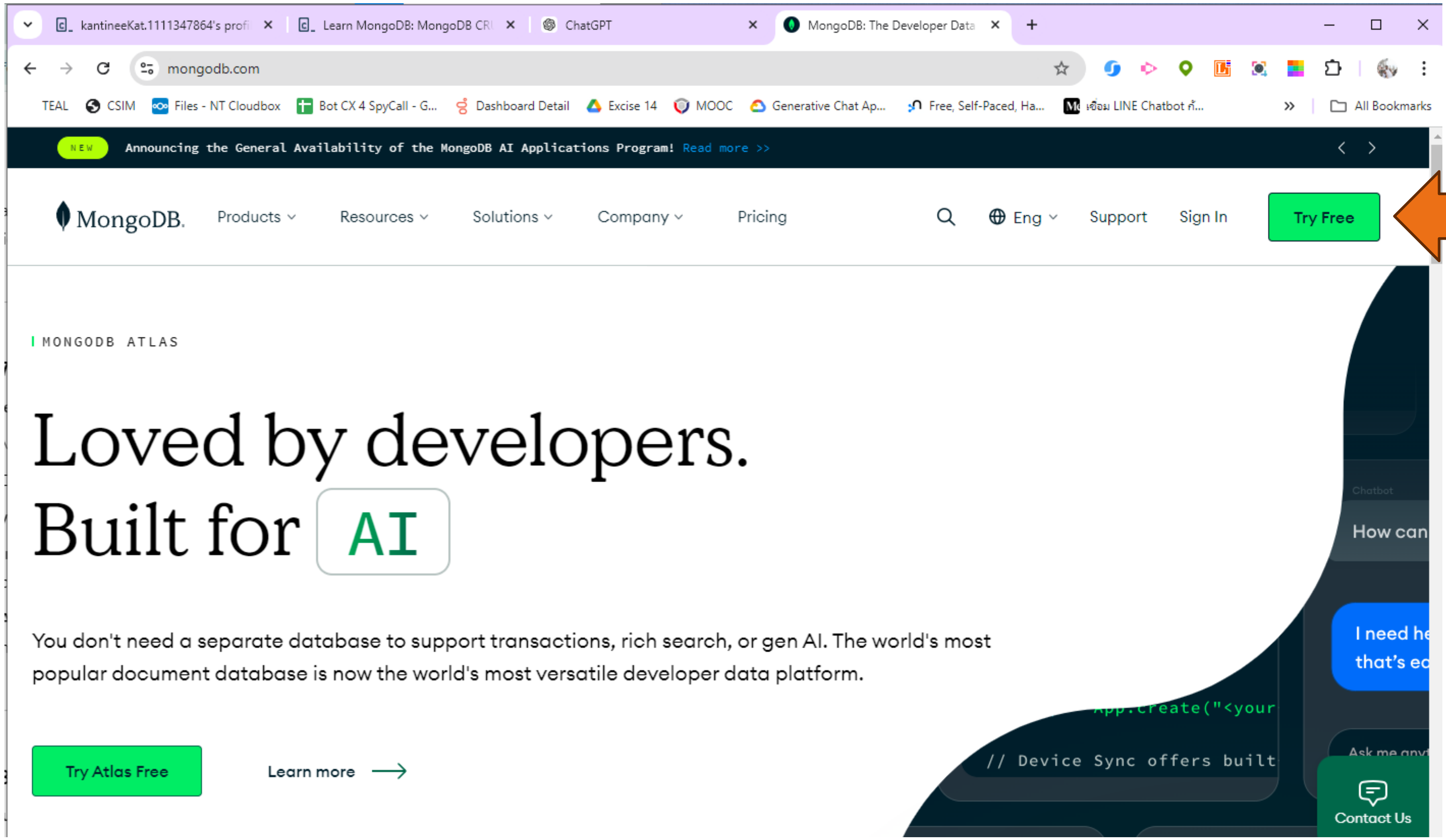
Atlas SQL
Easily connect SQL tools to Atlas for data analysis and visualization

MongoDB Practice:

Setting Up

- Managing Product Information
  - INSERT
  - UPDATE
  - DELETE

Step 1. Set up Mongo Atlas (Server)

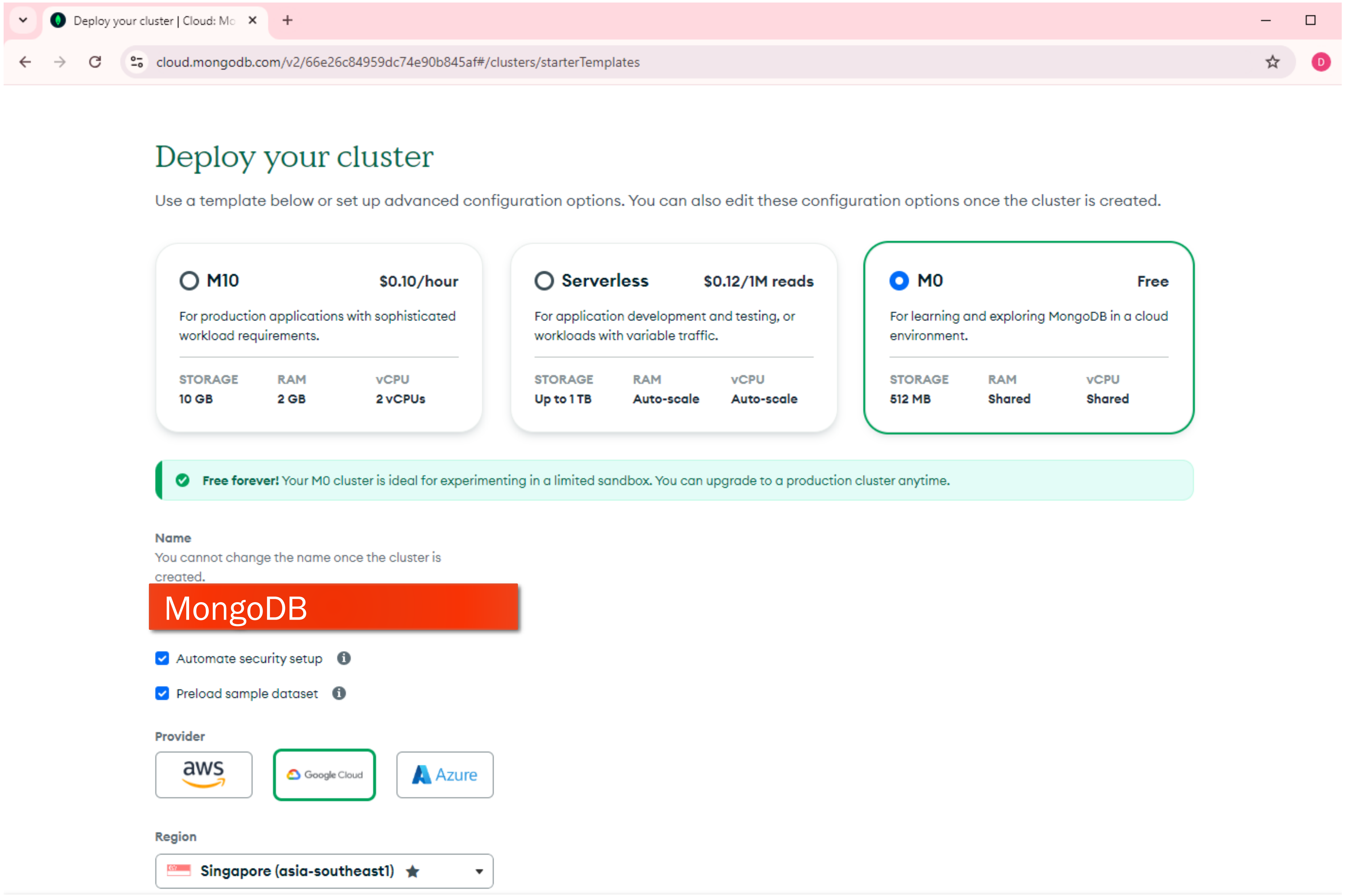


MONGODB ATLAS

Loved by developers.
Built for AI

You don't need a separate database to support transactions, rich search, or gen AI. The world's most popular document database is now the world's most versatile developer data platform.

Use your
chula account
xxx@chula.ac.th



Deploy your cluster | Cloud: Mo

cloud.mongodb.com/v2/66e26c84959dc74e90b845af#/clusters/starterTemplates

Deploy your cluster

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

○ M10	$0.10/hour		
For production applications with sophisticated workload requirements.			
STORAGE 10 GB	RAM 2 GB	vCPU 2 vCPUs	

○ Serverless	$0.12/1M reads		
For application development and testing, or workloads with variable traffic.			
STORAGE Up to 1 TB	RAM Auto-scale	vCPU Auto-scale	

● M0	Free		
For learning and exploring MongoDB in a cloud environment.			
STORAGE 512 MB	RAM Shared	vCPU Shared	

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Name
You cannot change the name once the cluster is created.

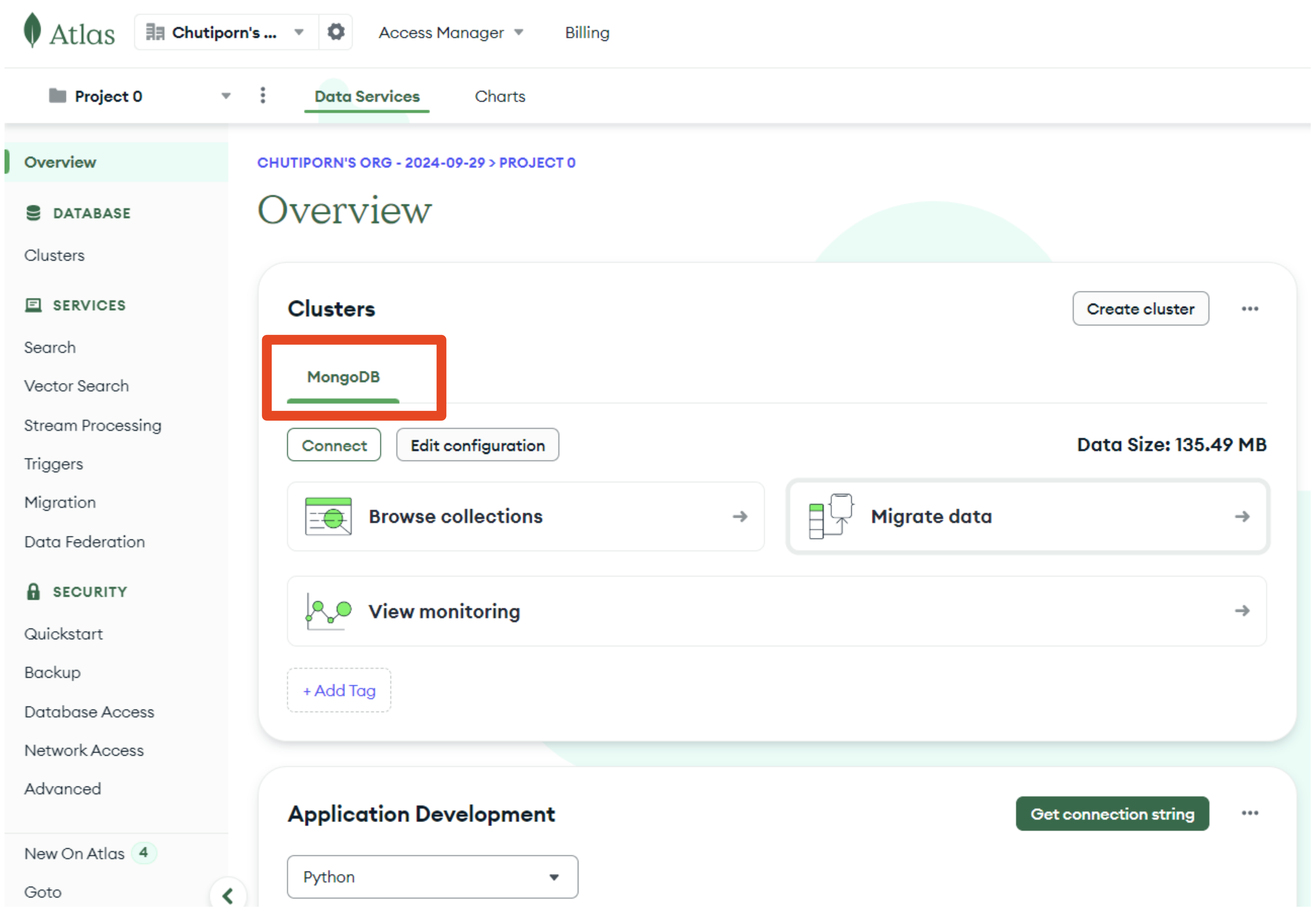
MongoDB

[✓] Automate security setup

[✓] Preload sample dataset

Provider
aws [✓ Google Cloud] Azure

Region
Singapore (asia-southeast1)



Atlas
Chutiporn's ...
Access Manager
Billing

Project 0
Data Services
Charts

Overview

DATABASE
Clusters

SERVICES
Search
Vector Search
Stream Processing
Triggers
Migration
Data Federation

SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced

New On Atlas 4
Goto

CHUTIPORN'S ORG - 2024-09-29 > PROJECT 0

Overview

Clusters
Create cluster

MongoDB

Connect
Edit configuration
Data Size: 135.49 MB

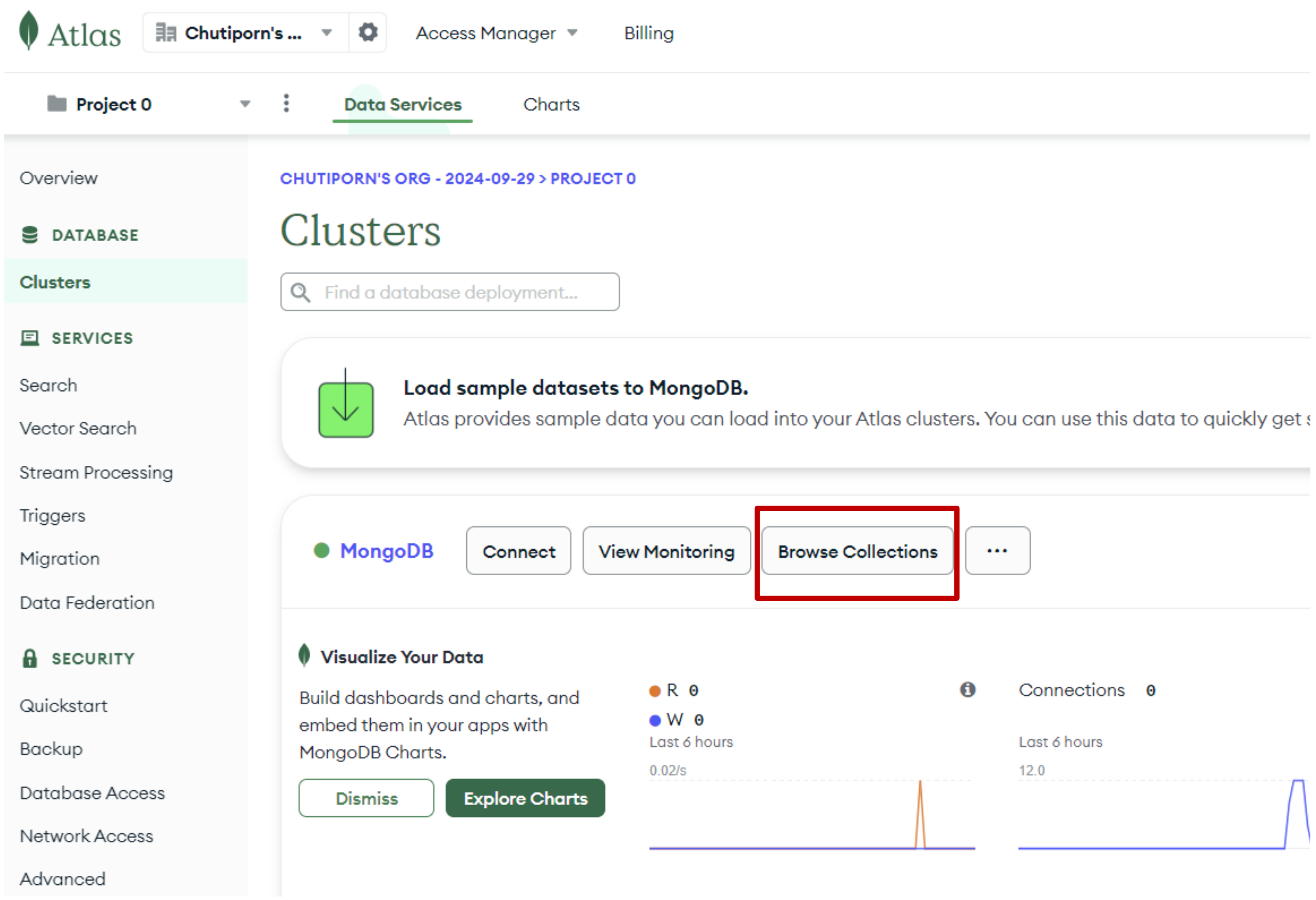
Browse collections
Migrate data
View monitoring

+ Add Tag

Application Development
Get connection string

Python

Step 2. Create a MongoDB Database


Atlas | Chutiporn's ... | Access Manager | Billing

Project 0 | Data Services | Charts

Overview

DATABASE

Clusters

SERVICES

Search

Vector Search

Stream Processing

Triggers

Migration

Data Federation

SECURITY

Quickstart

Backup

Database Access

Network Access

Advanced

CHUTIPORN'S ORG - 2024-09-29 > PROJECT 0

Clusters

Find a database deployment...

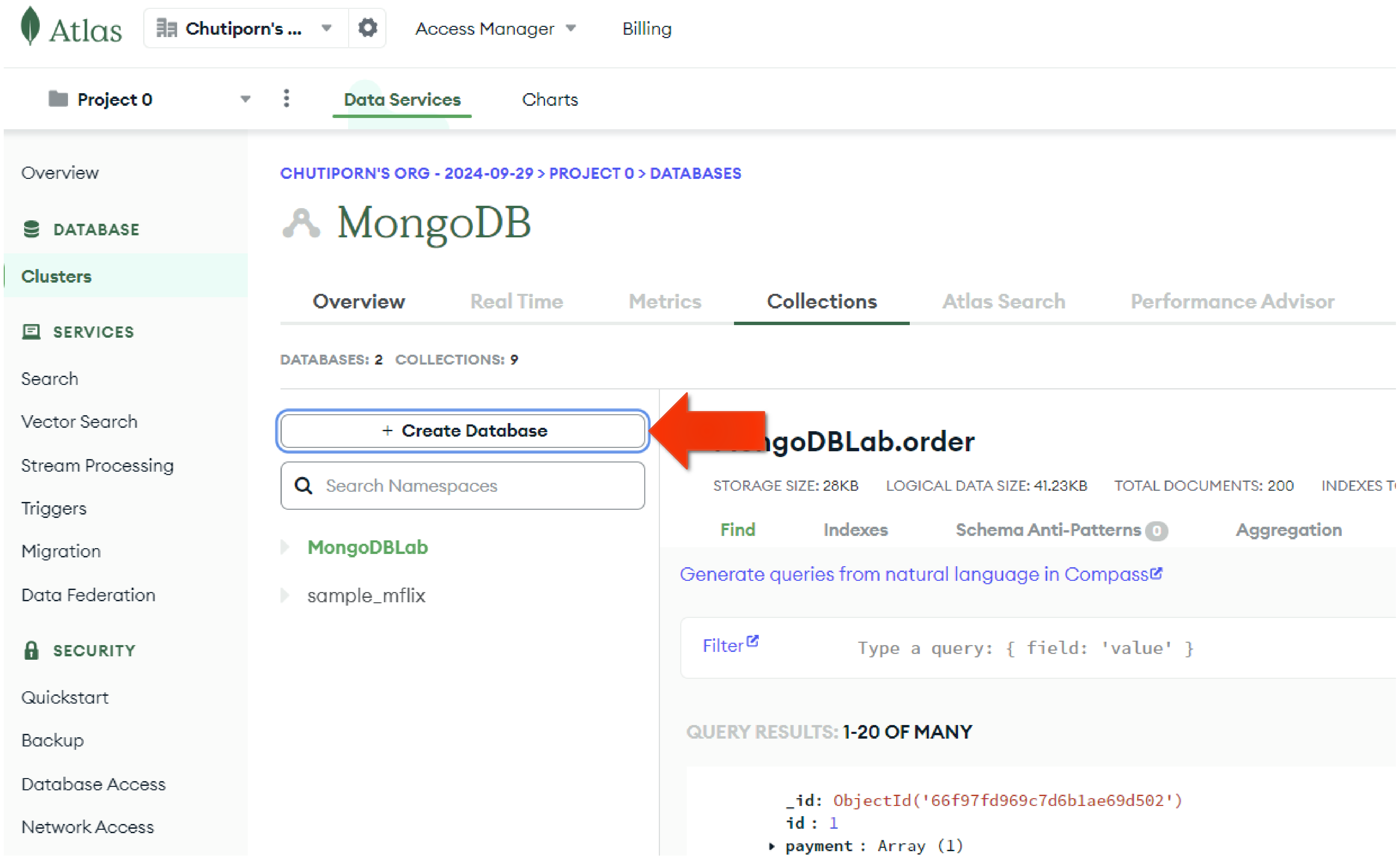
Load sample datasets to MongoDB.
Atlas provides sample data you can load into your Atlas clusters. You can use this data to quickly get

MongoDB | Connect | View Monitoring | Browse Collections | ···

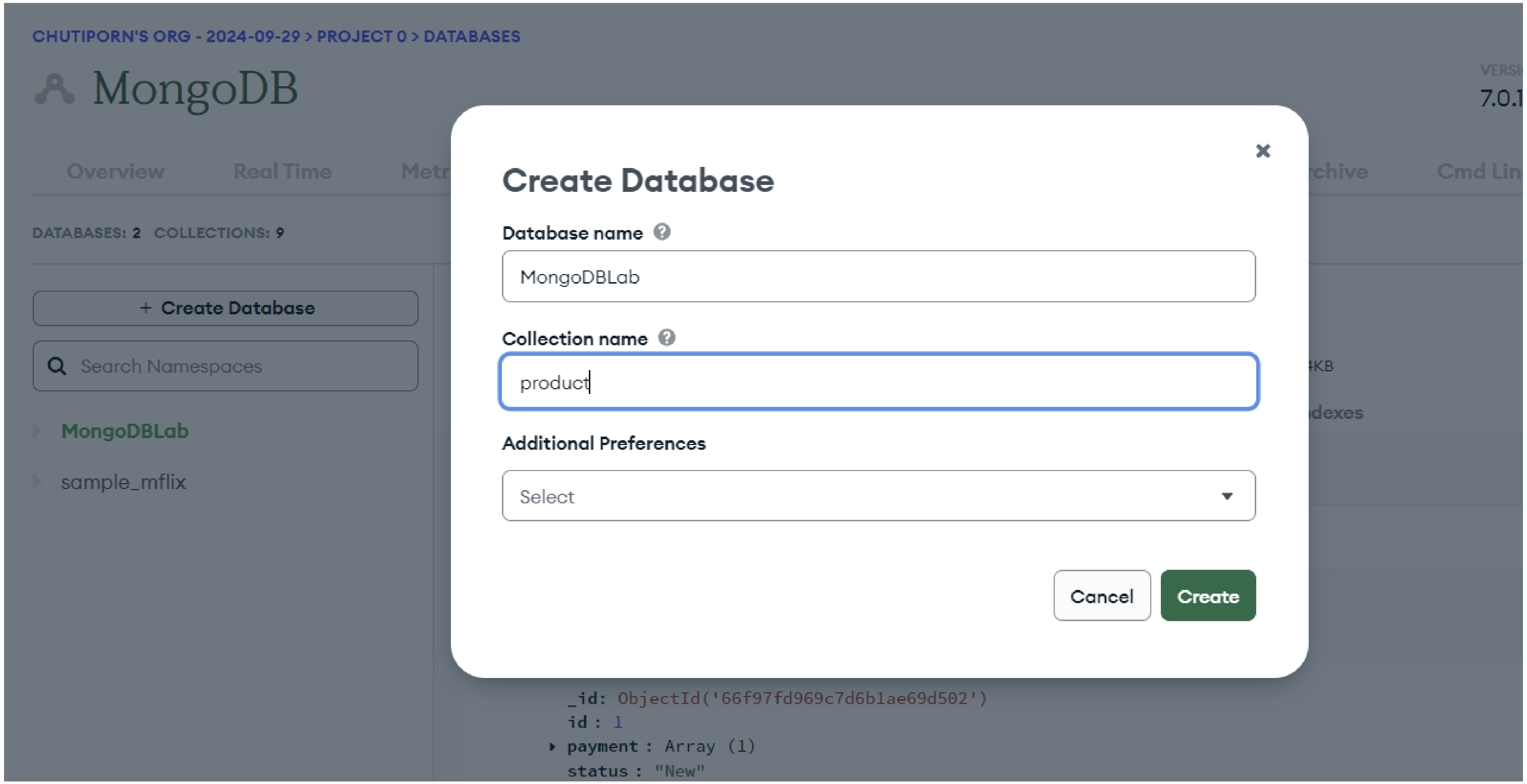
Visualize Your Data

Build dashboards and charts, and embed them in your apps with MongoDB Charts.

Dismiss | Explore Charts



```
_id: ObjectId('66f97fd969c7d6b1ae69d502')
id : 1
▸ payment : Array (1)
```



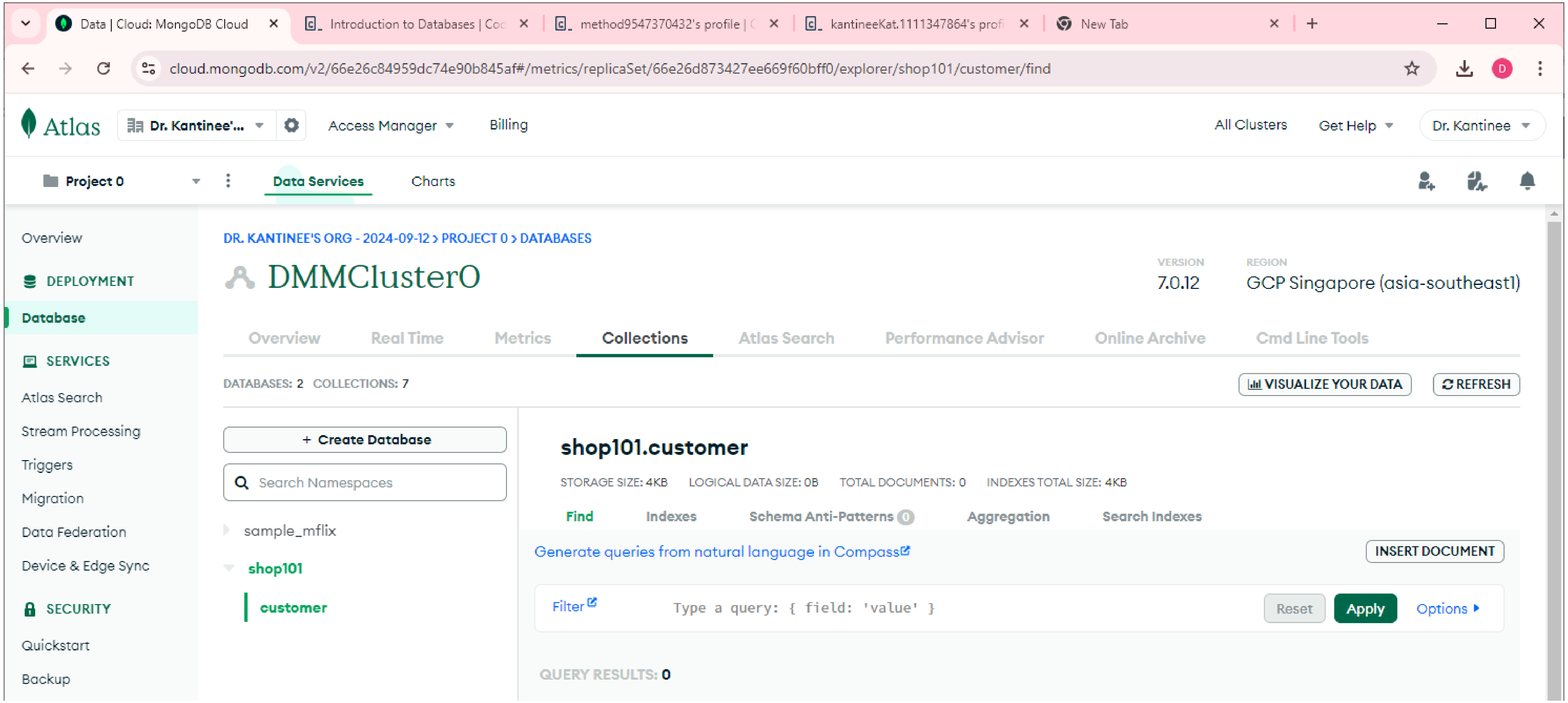
Create Database

Database name: MongoDBLab

Collection name: product

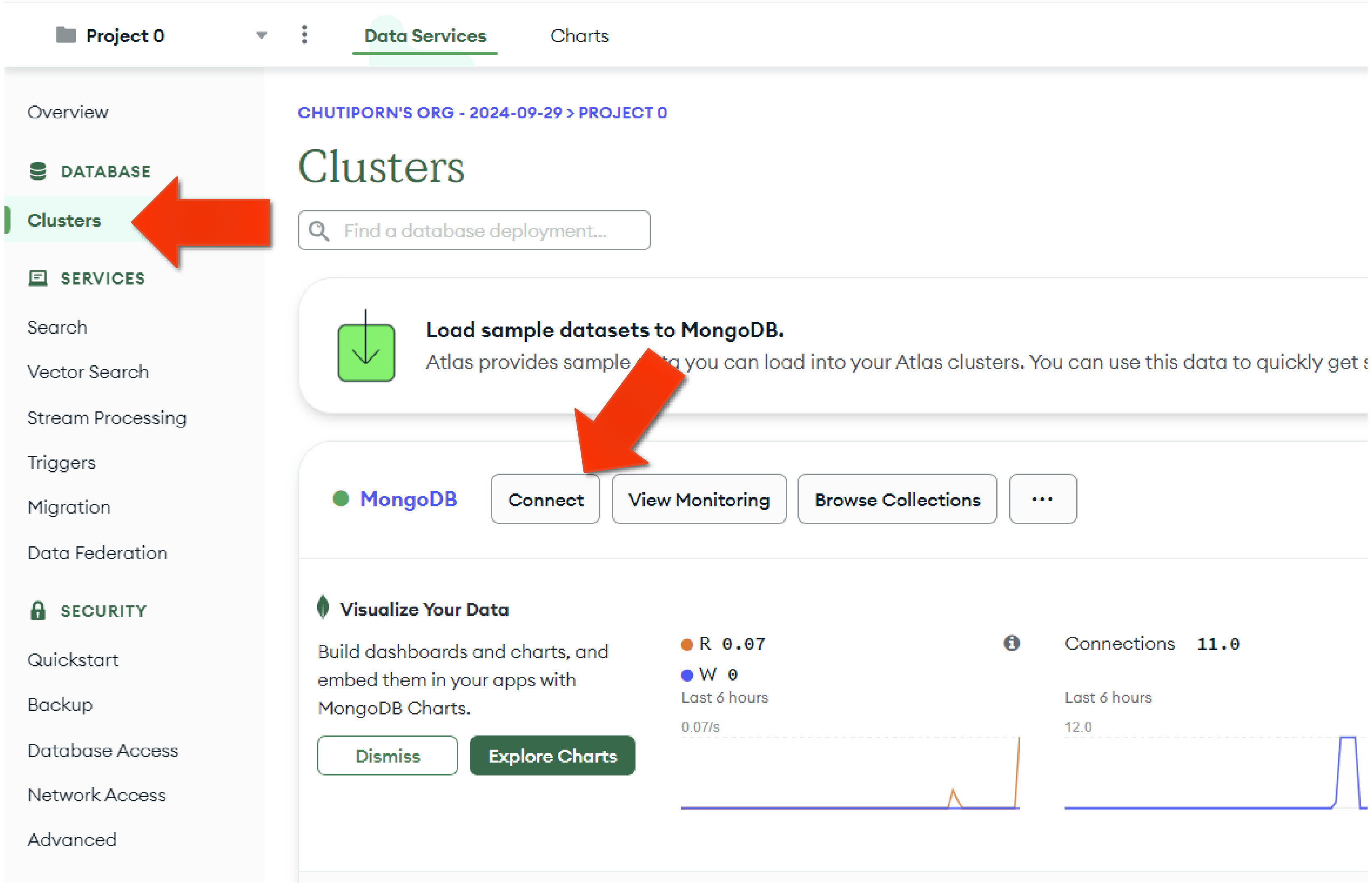
Additional Preferences: Select

```
_id: ObjectId('66f97fd969c7d6b1ae69d502')
id : 1
payment : Array (1)
status : "New"
```



Data | Cloud: MongoDB Cloud
Introduction to Databases | Coc
method9547370432's profile | C
kantineeKat.1111347864's profi
New Tab
cloud.mongodb.com/v2/66e26c84959dc74e90b845af#/metrics/replicaSet/66e26d873427ee669f60bff0/explorer/shop101/customer/find
Atlas
Dr. Kantinee'...
Access Manager
Billing
All Clusters
Get Help
Dr. Kantinee
Project 0
Data Services
Charts
Overview
DEPLOYMENT
Database
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
Device & Edge Sync
SECURITY
Quickstart
Backup
DR. KANTINEE'S ORG - 2024-09-12 > PROJECT 0 > DATABASES
DMMCluster0
VERSION
7.0.12
REGION
GCP Singapore (asia-southeast1)
Overview
Real Time
Metrics
Collections
Atlas Search
Performance Advisor
Online Archive
Cmd Line Tools
DATABASES: 2 COLLECTIONS: 7
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
sample_mflix
shop101
customer
shop101.customer
STORAGE SIZE: 4KB LOGICAL DATA SIZE: 0B TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE: 4KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
Generate queries from natural language in Compass
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
QUERY RESULTS: 0

Step 3. Connect to the Database



Project 0

Data Services

Charts

Overview

DATABASE

Clusters

SERVICES

Search

Vector Search

Stream Processing

Triggers

Migration

Data Federation

SECURITY

Quickstart

Backup

Database Access

Network Access

Advanced

CHUTIPORN'S ORG - 2024-09-29 > PROJECT 0

Clusters

Find a database deployment...

Load sample datasets to MongoDB.

Atlas provides sample data you can load into your Atlas clusters. You can use this data to quickly get s

MongoDB | Connect | View Monitoring | Browse Collections | ···

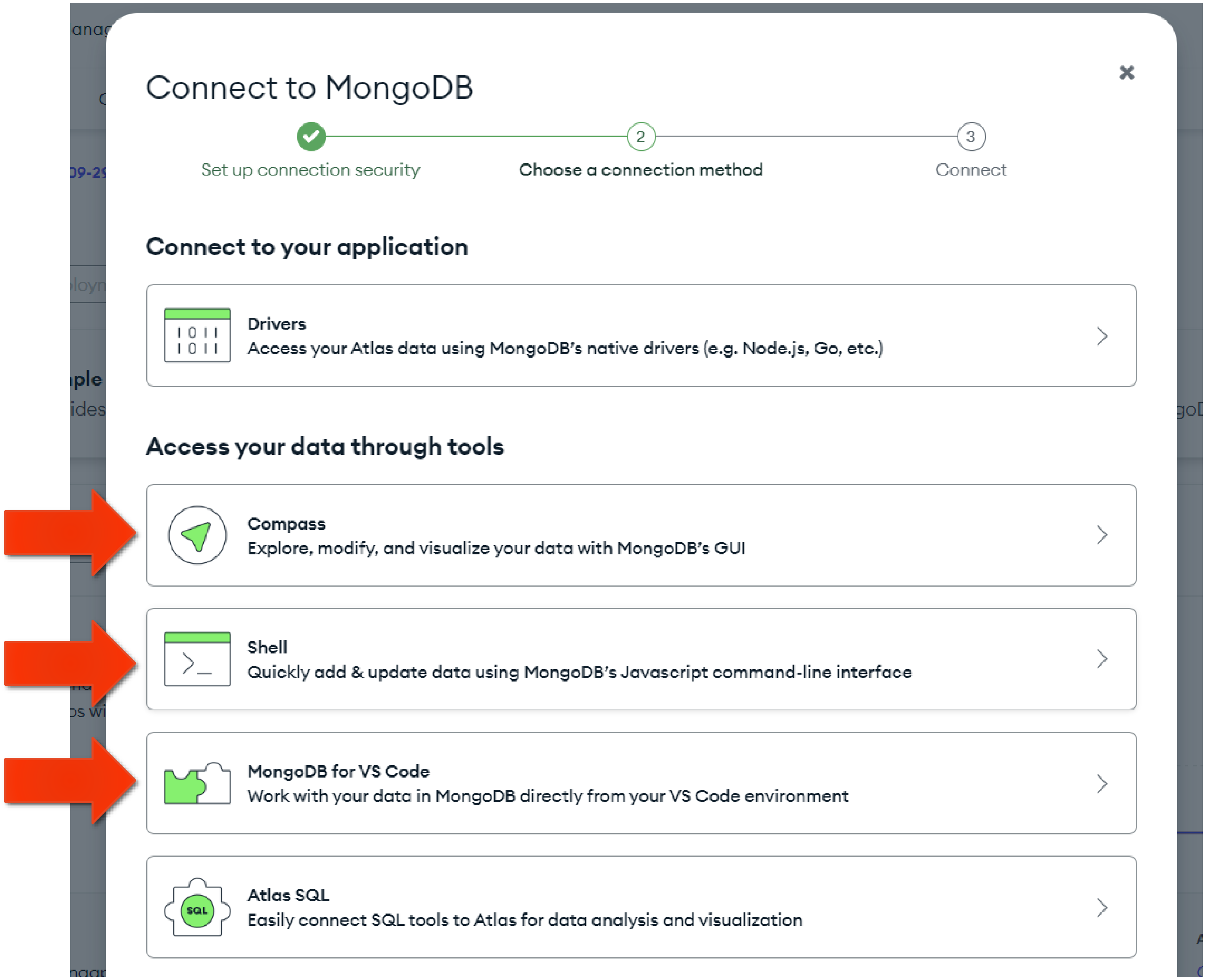
Visualize Your Data

Build dashboards and charts, and embed them in your apps with MongoDB Charts.

Dismiss | Explore Charts

● R 0.07
● W 0
Last 6 hours
0.07/s

Connections 11.0
Last 6 hours
12.0



Connect to MongoDB

Set up connection security — Choose a connection method — Connect

Connect to your application

Drivers
Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)

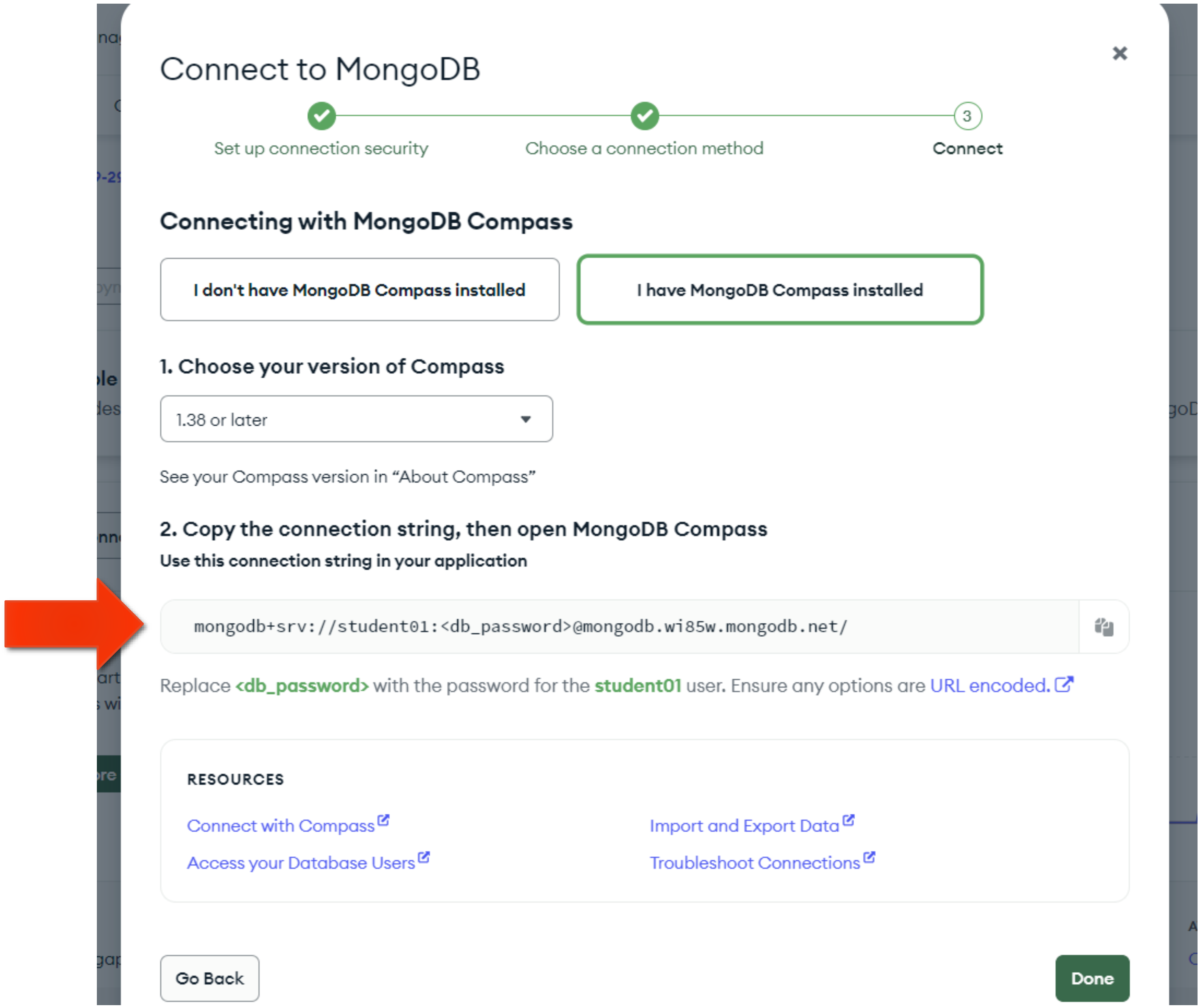
Access your data through tools

Compass
Explore, modify, and visualize your data with MongoDB's GUI

Shell
Quickly add & update data using MongoDB's Javascript command-line interface

MongoDB for VS Code
Work with your data in MongoDB directly from your VS Code environment

Atlas SQL
Easily connect SQL tools to Atlas for data analysis and visualization



Connect to MongoDB

Set up connection security
Choose a connection method
3
Connect

Connecting with MongoDB Compass

I don't have MongoDB Compass installed
I have MongoDB Compass installed

1. Choose your version of Compass

1.38 or later

See your Compass version in “About Compass”

2. Copy the connection string, then open MongoDB Compass

Use this connection string in your application

```
mongodb+srv://student01:<db_password>@mongodb.wi85w.mongodb.net/
```

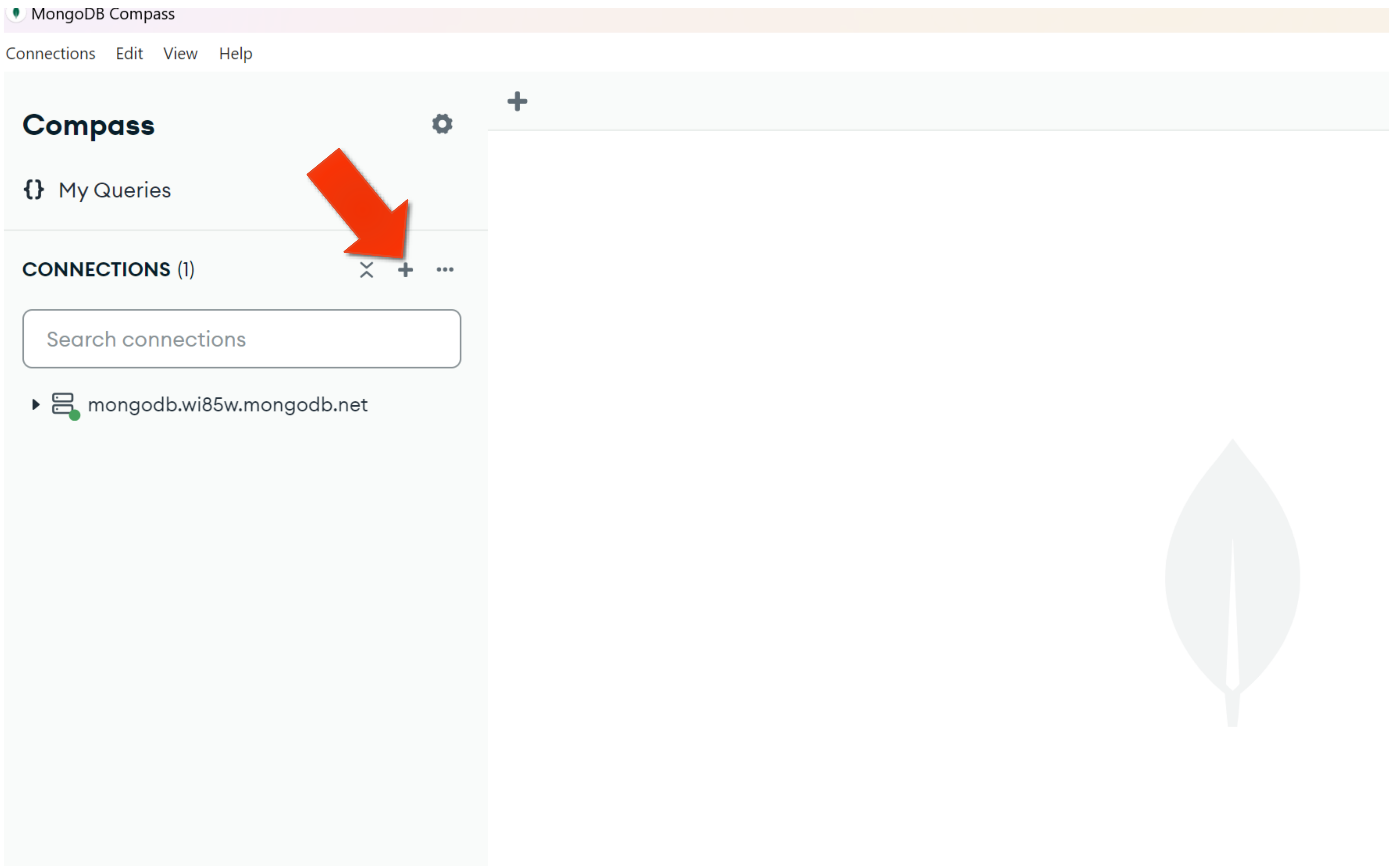
Replace <db_password> with the password for the student01 user. Ensure any options are URL encoded.

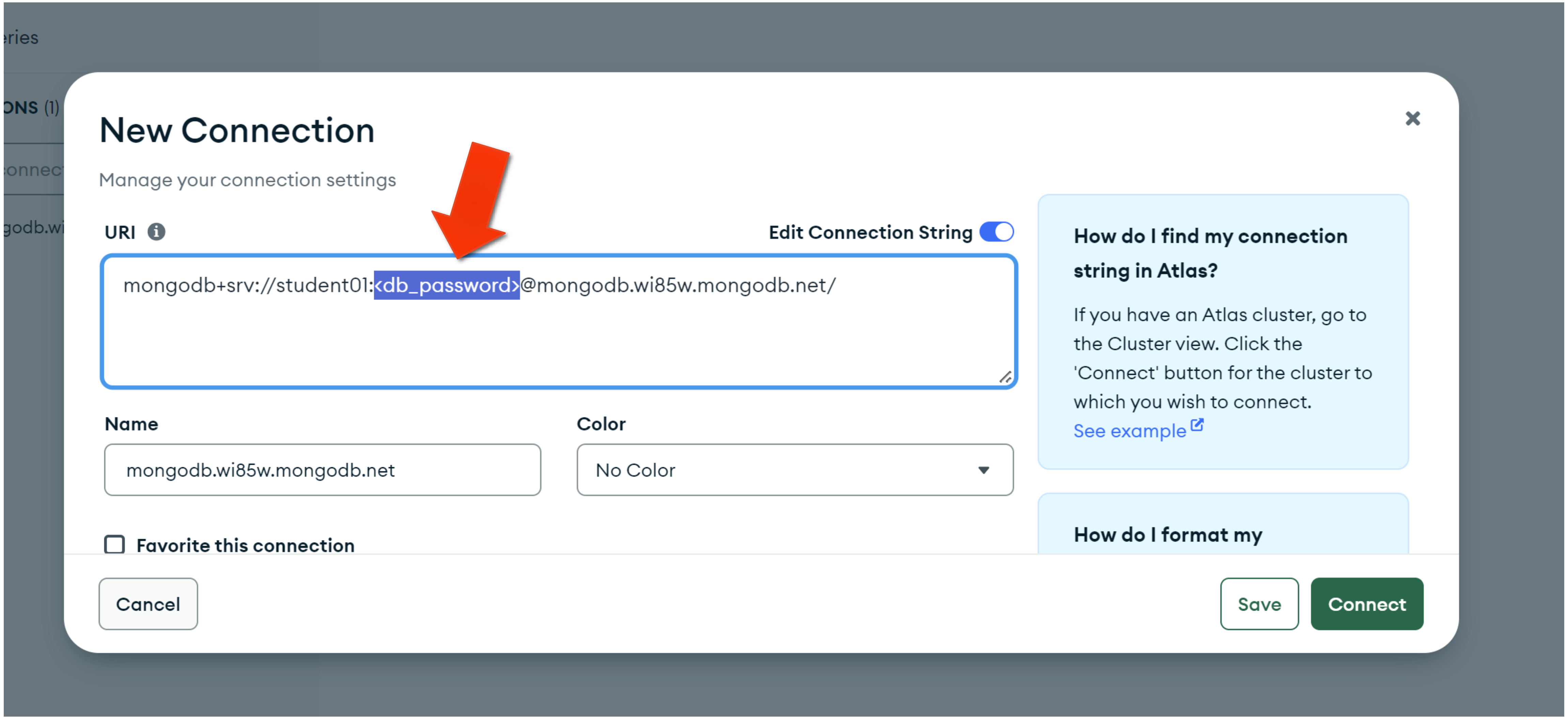
RESOURCES

Step 4. Install and Set-up MongoDB Compass

MongoDB Compass

- Download and Install MongoDB Compass
  https://www.mongodb.com/try/download/compass
- Create a new connection


MongoDB Compass
Connections
Edit
View
Help
Compass
My Queries
CONNECTIONS (1)
Search connections
mongodb.wi85w.mongodb.net



eries

ONS (1)

onnec

godb.wi

New Connection

Manage your connection settings

URI

Edit Connection String

```
mongodb+srv://student01:<db_password>@mongodb.wi85w.mongodb.net/
```

Name: mongodb.wi85w.mongodb.net

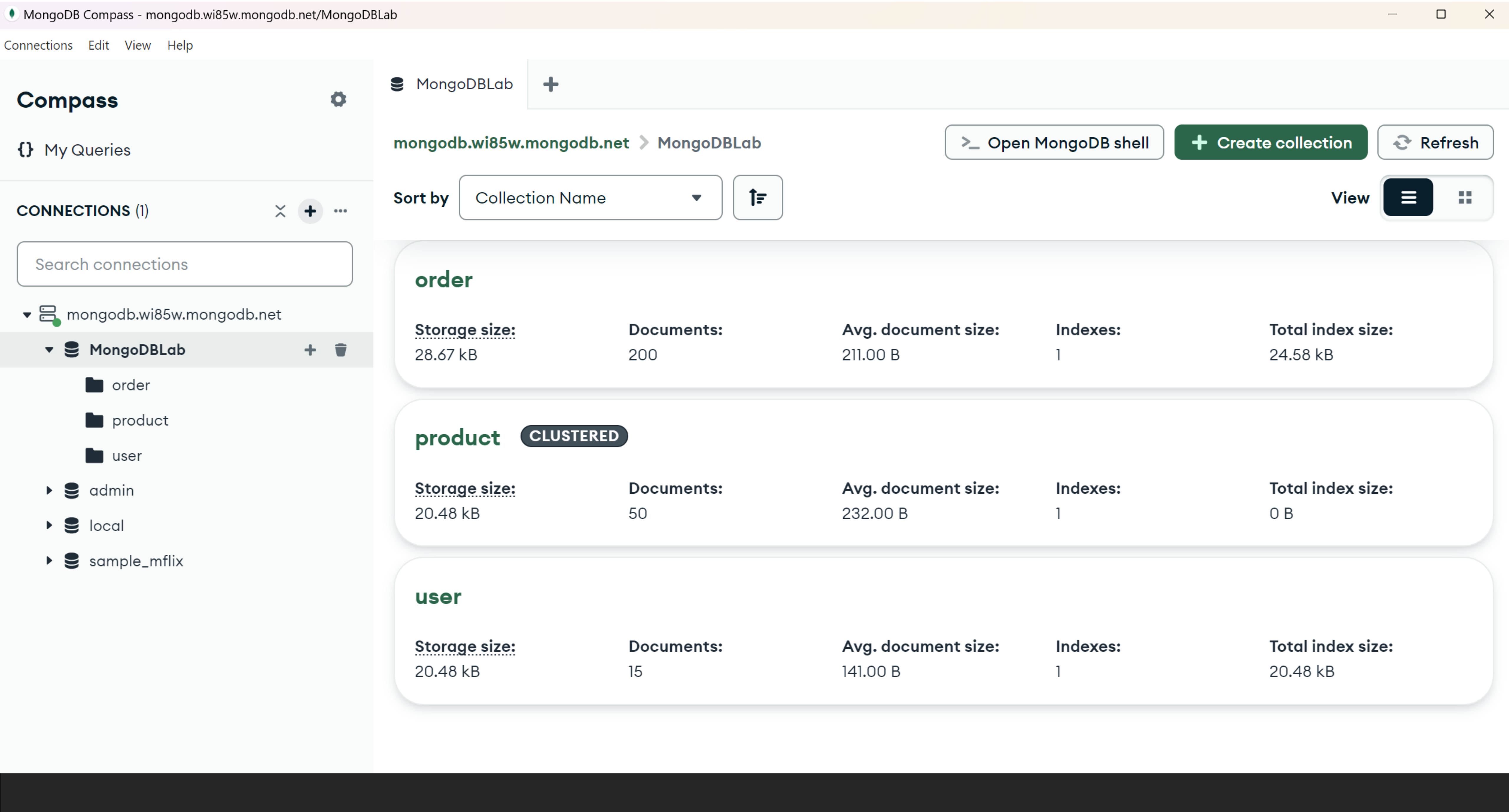
Color: No Color

☐ Favorite this connection

How do I find my connection string in Atlas?

If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.
See example

How do I format my



order

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
28.67 kB	200	211.00 B	1	24.58 kB

product CLUSTERED

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	50	232.00 B	1	0 B

user

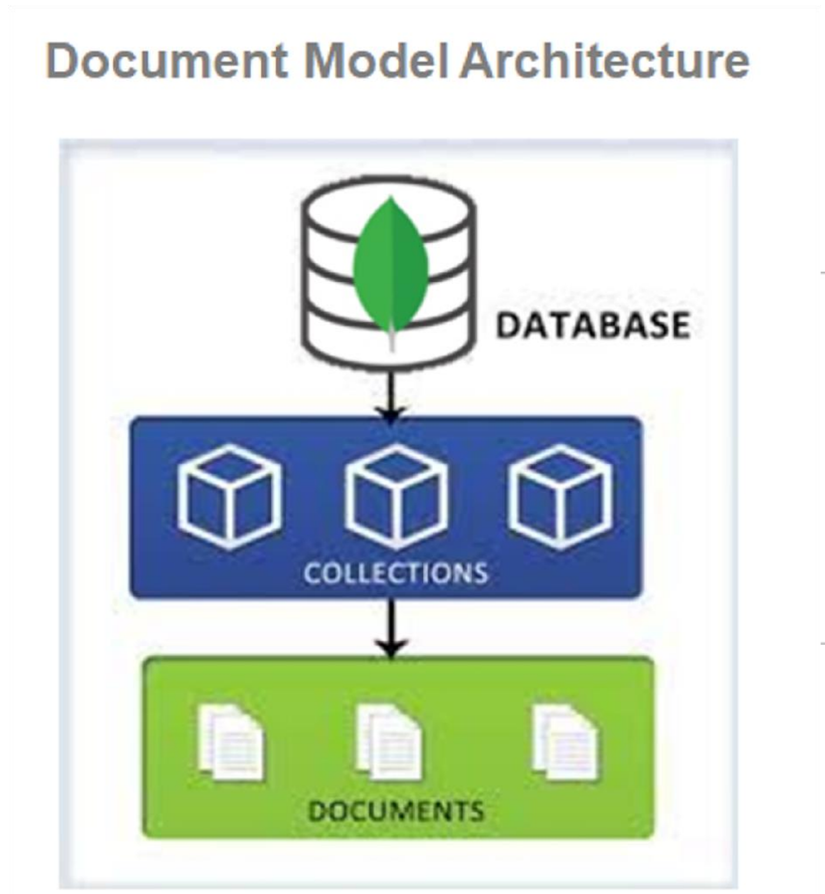
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	15	141.00 B	1	20.48 kB

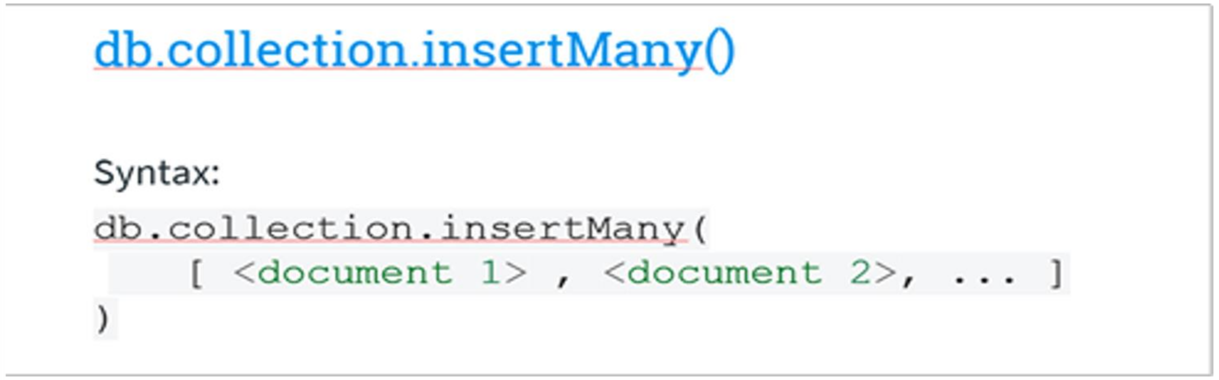
MongoDB Practice:

Part I

- Managing Product Information
  - INSERT
  - UPDATE
  - DELETE

Insert Documents



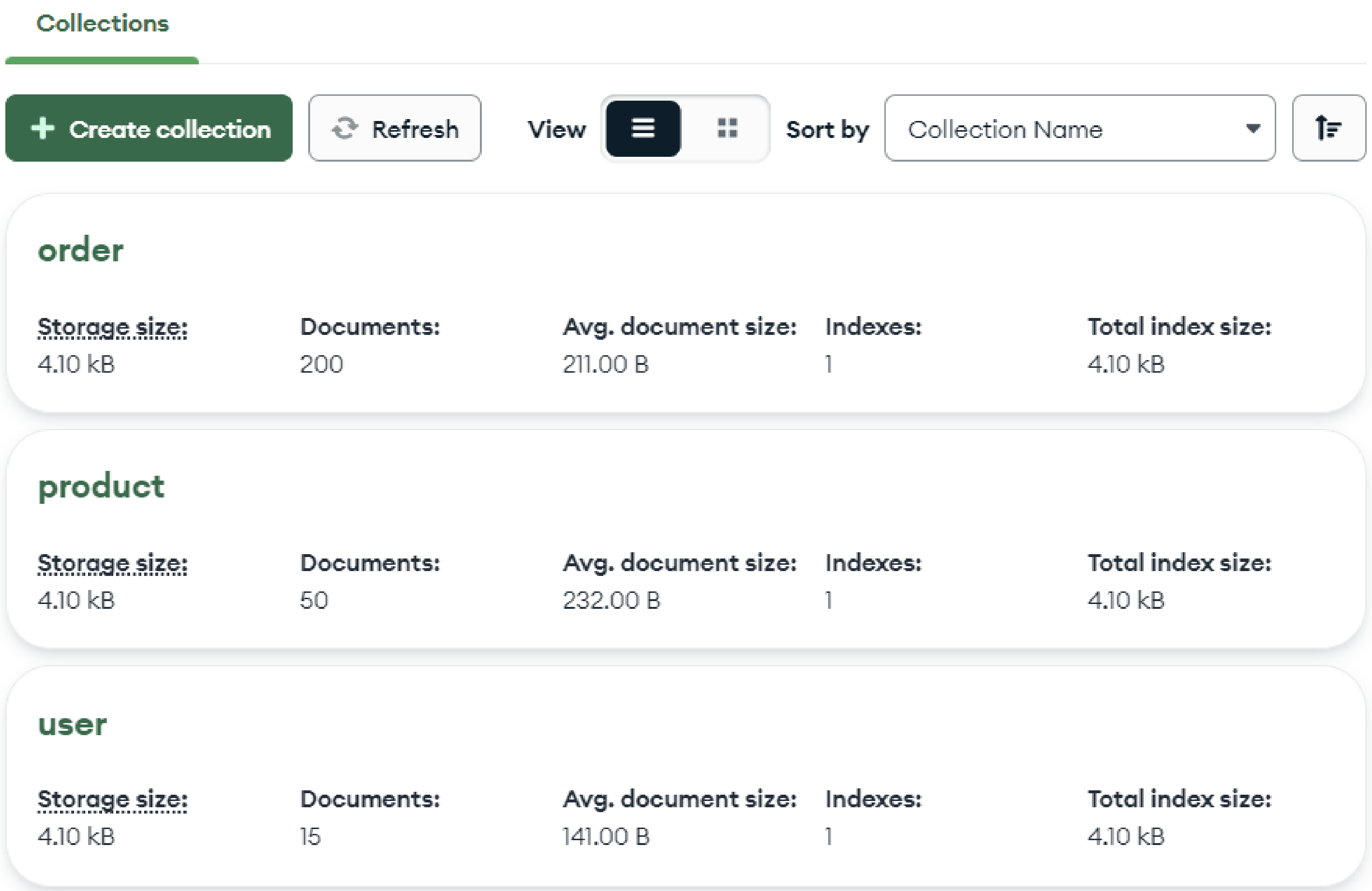

db.collection.insertMany()

Syntax:

```
db.collection.insertMany(
    [ <document 1> , <document 2>, ... ]
)
```

Practice 01: Create Database and Document Collections

- Connect to your MongoDB Cluster
- Create a new Database "MongoDBLab"
- Create new 3 collections and load the provided json documents:
  - product
  - user
  - order



order

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
4.10 kB	200	211.00 B	1	4.10 kB

product

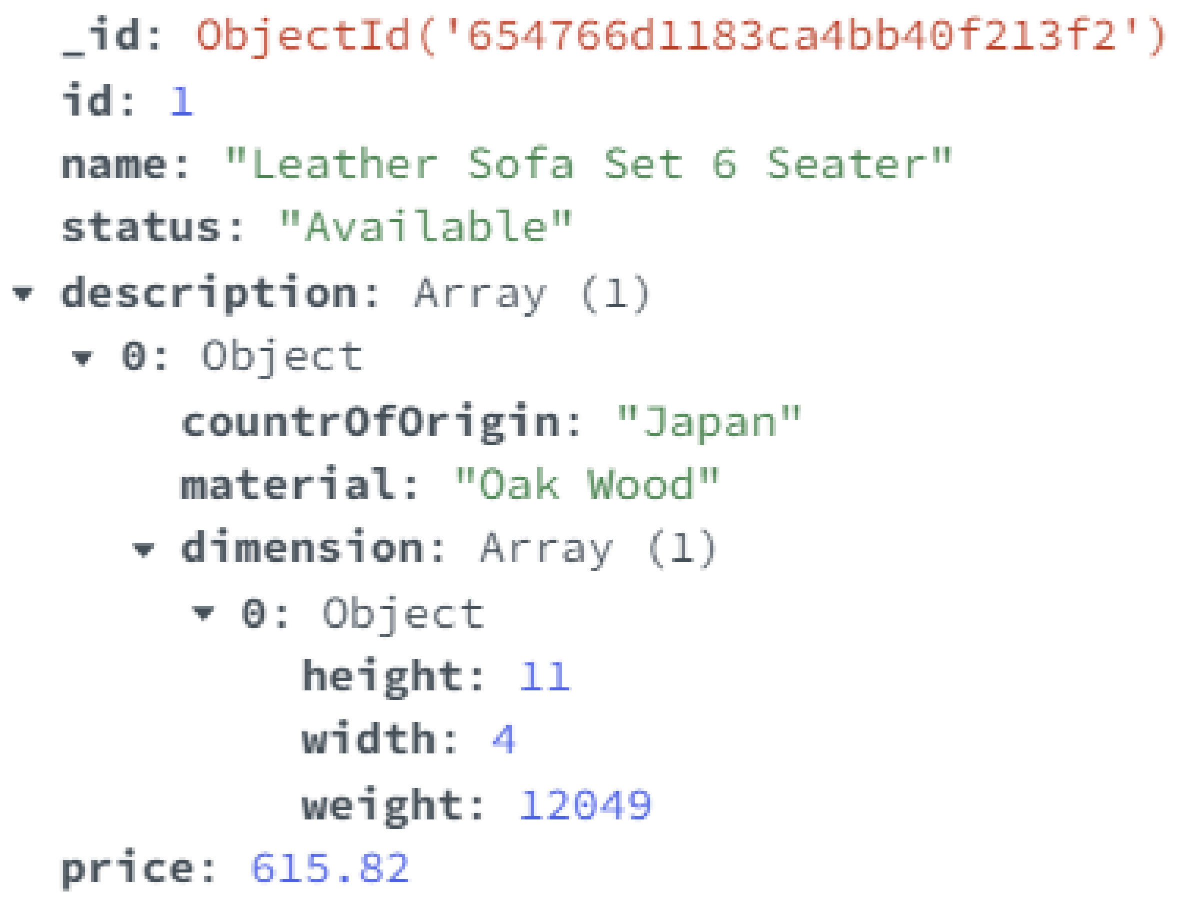
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
4.10 kB	50	232.00 B	1	4.10 kB

user

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
4.10 kB	15	141.00 B	1	4.10 kB

Sample data

product document



```
  _id: ObjectId('654766d1183ca4bb40f213f2')
  id: 1
  name: "Leather Sofa Set 6 Seater"
  status: "Available"
▾ description: Array (1)
  ▾ 0: Object
      countrOfOrigin: "Japan"
      material: "Oak Wood"
    ▾ dimension: Array (1)
      ▾ 0: Object
          height: 11
          width: 4
          weight: 12049
  price: 615.82
```

Insert Documents

There are 2 methods to insert documents into a MongoDB database.

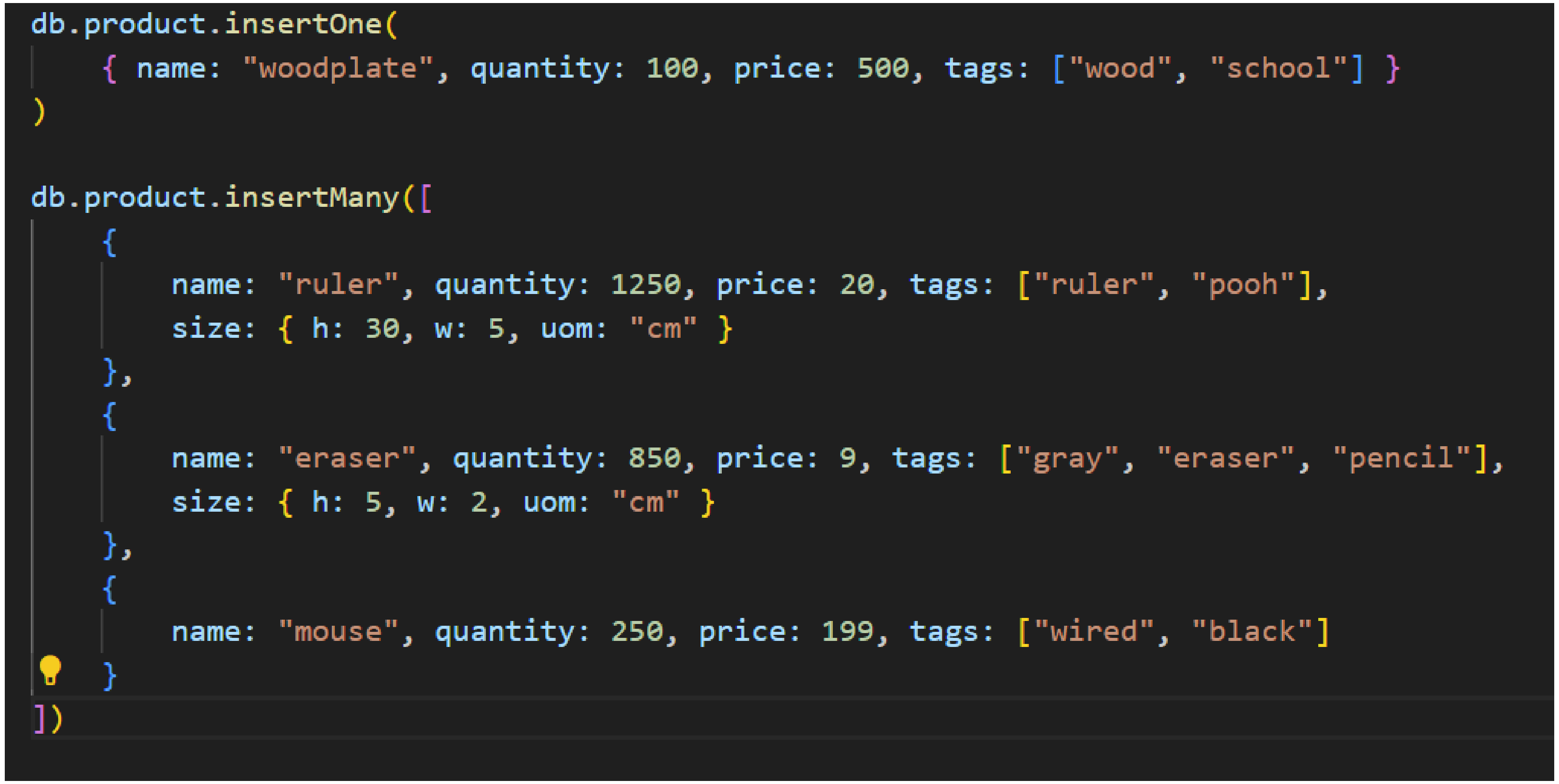
- insertOne()

  This method inserts a single object into the database.

- insertMany()

  This method inserts an array of objects into the database.

Practice 02: Insert product documents



```
db.product.insertOne(
    { name: "woodplate", quantity: 100, price: 500, tags: ["wood", "school"] }
)

db.product.insertMany([
    {
        name: "ruler", quantity: 1250, price: 20, tags: ["ruler", "pooh"],
        size: { h: 30, w: 5, uom: "cm" }
    },
    {
        name: "eraser", quantity: 850, price: 9, tags: ["gray", "eraser", "pencil"],
        size: { h: 5, w: 2, uom: "cm" }
    },
    {
        name: "mouse", quantity: 250, price: 199, tags: ["wired", "black"]
    }
])
```

Update Document

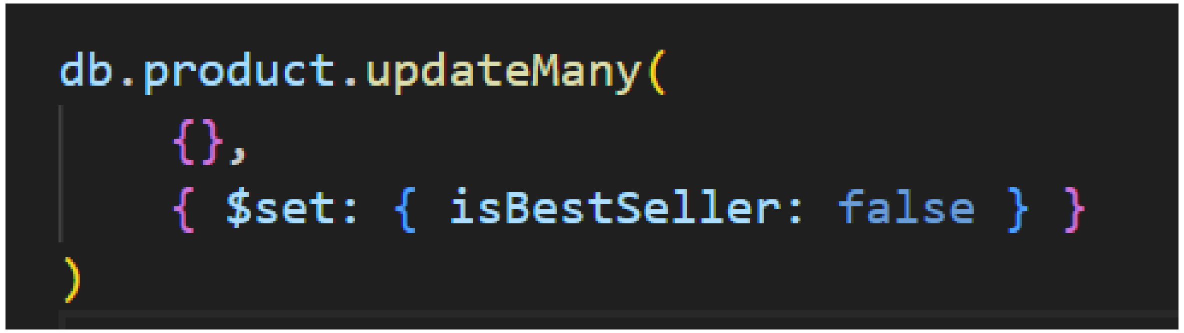
To update an existing document we can use

- updateOne()
- updateMany()

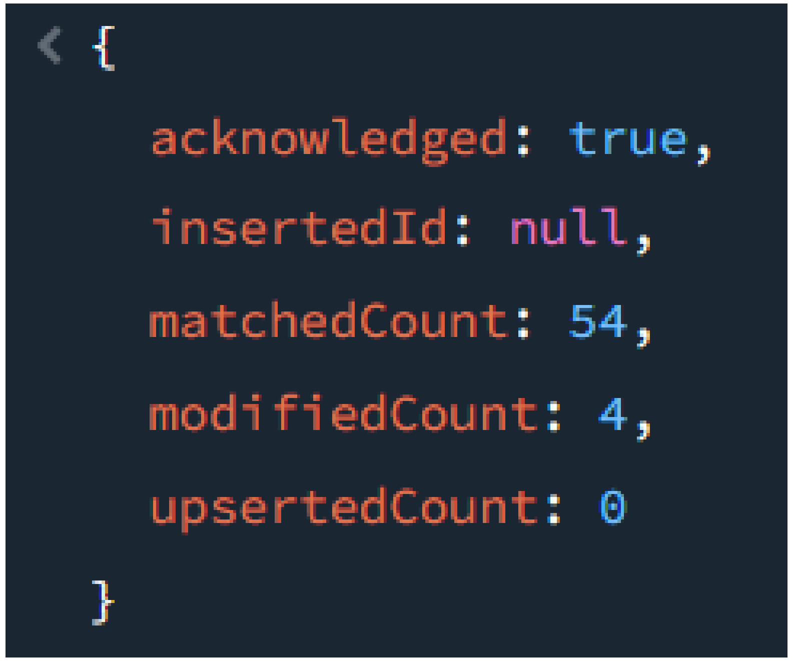
The first parameter is a query object to define which document or documents should be updated.

The second parameter is an object defining the updated data.

Practice 03: UpdateMany

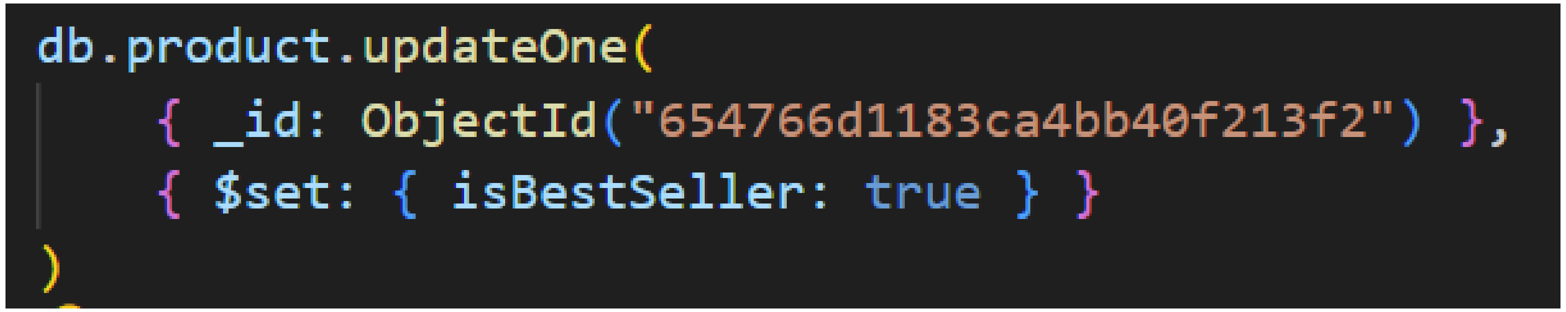


```
db.product.updateMany(
    {},
    { $set: { isBestSeller: false } }
)
```

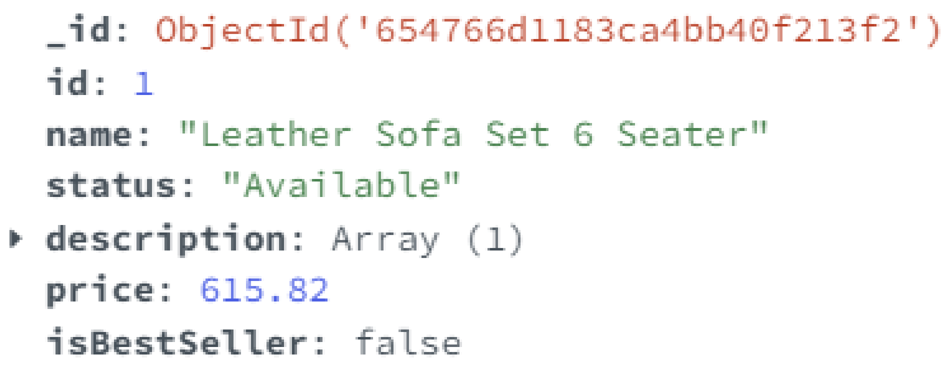


```
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 54,
    modifiedCount: 4,
    upsertedCount: 0
  }
```

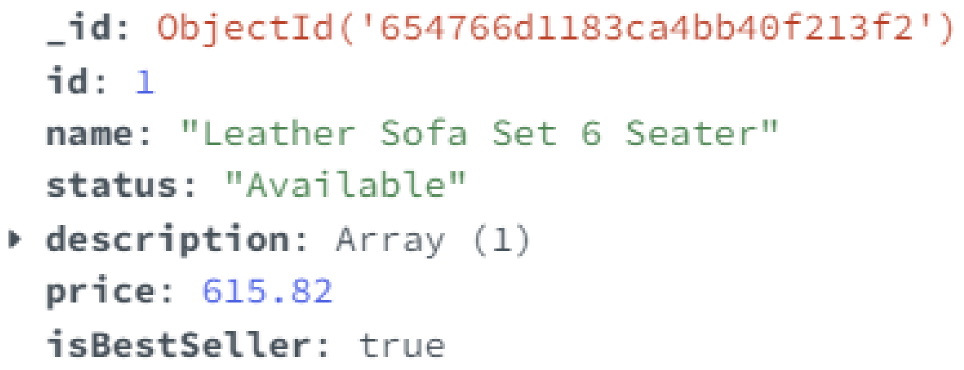
Practice 04: UpdateOne



```
db.product.updateOne(
    { _id: ObjectId("654766d1183ca4bb40f213f2") },
    { $set: { isBestSeller: true } }
)
```

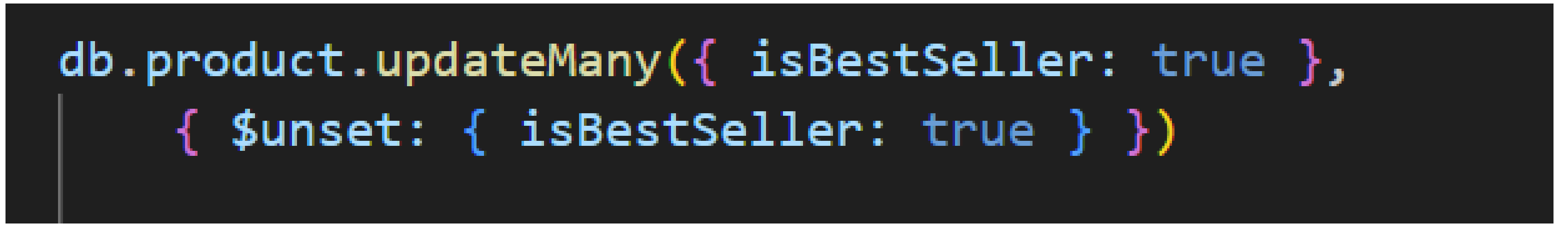


```
  _id: ObjectId('654766d1183ca4bb40f213f2')
  id: 1
  name: "Leather Sofa Set 6 Seater"
  status: "Available"
▸ description: Array (1)
  price: 615.82
  isBestSeller: false
```

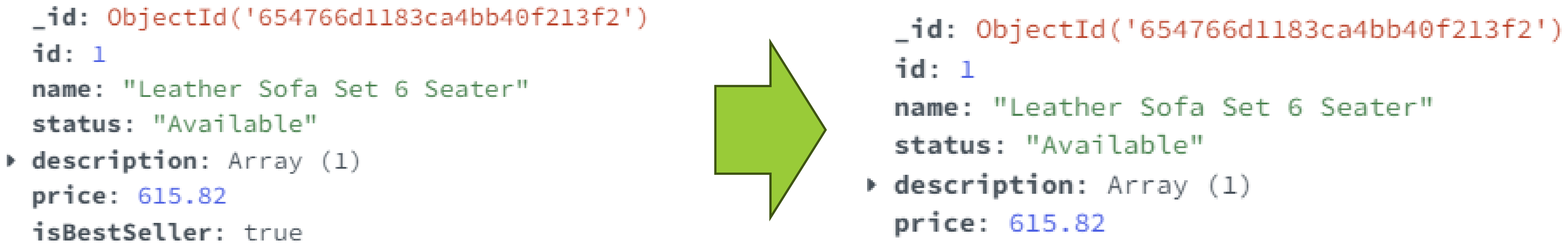


```
  _id: ObjectId('654766d1183ca4bb40f213f2')
  id: 1
  name: "Leather Sofa Set 6 Seater"
  status: "Available"
▸ description: Array (1)
  price: 615.82
  isBestSeller: true
```

Practice 05: $unset



```
db.product.updateMany({ isBestSeller: true },
    { $unset: { isBestSeller: true } })
```



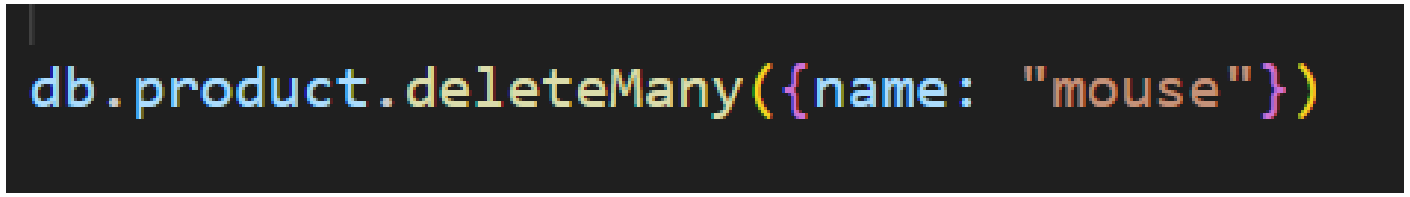
```
  _id: ObjectId('654766d1183ca4bb40f213f2')
  id: 1
  name: "Leather Sofa Set 6 Seater"
  status: "Available"
▸ description: Array (1)
  price: 615.82
  isBestSeller: true
```

```
  _id: ObjectId('654766d1183ca4bb40f213f2')
  id: 1
  name: "Leather Sofa Set 6 Seater"
  status: "Available"
▸ description: Array (1)
  price: 615.82
```

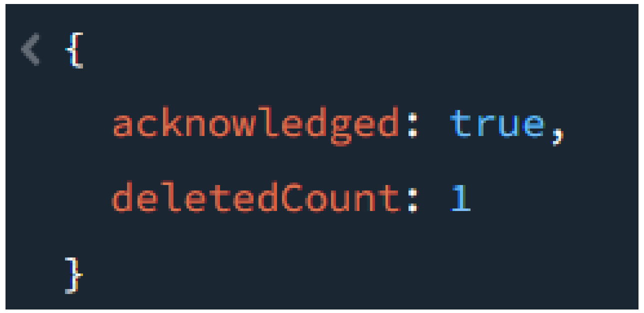
Delete Documents

We can delete documents by using the methods deleteOne() or deleteMany().

These methods accept a query object. The matching documents will be deleted.



```
db.product.deleteMany({name: "mouse"})
```

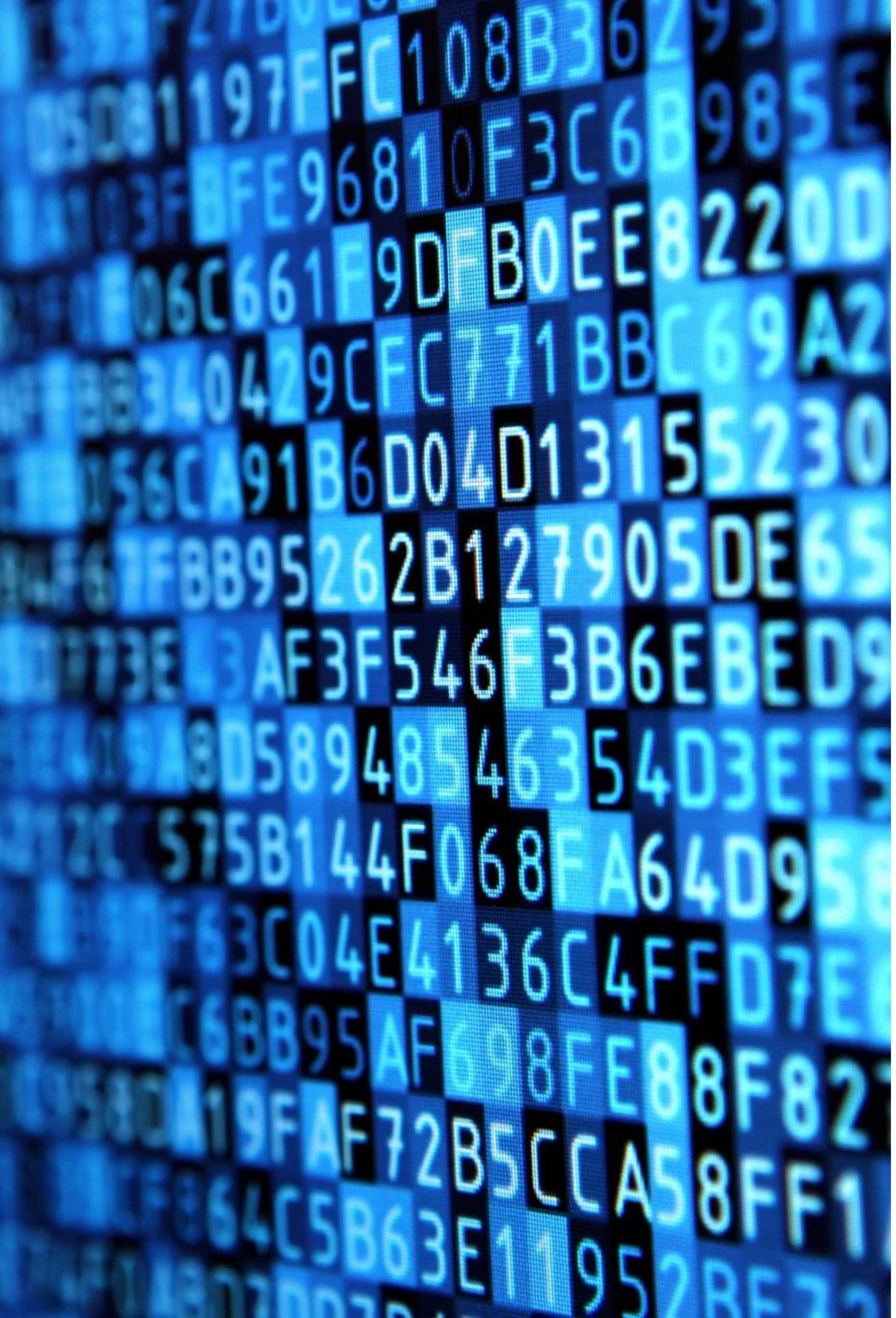


```
< {
    acknowledged: true,
    deletedCount: 1
  }
```

Your Turn

1. Add your own 4 products using insertMany()
2. Try to update your product information by adding a quantity field. Define different quantity for each product.

THANK YOU



5602201

8. MongoDB Practice: Part II

CHUTIPORN ANUTARIYA

MongoDB Practice:

Part II

- Searching Products
  - Basic Query

Find Data

There are 2 methods to find and select data from a MongoDB collection, find() and findOne().

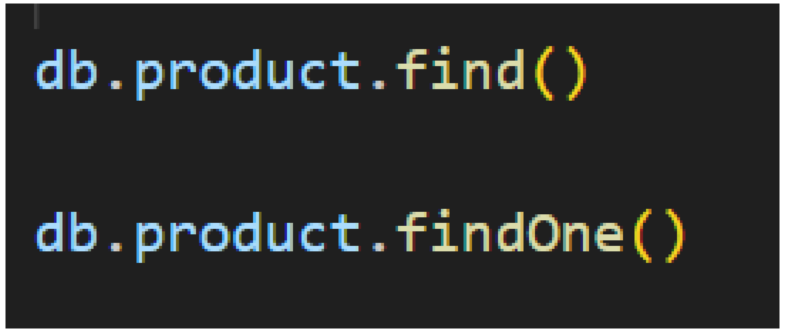
find()

- To select data from a collection in MongoDB, we can use the find() method.
- This method accepts a query object. If left empty, all documents will be returned.

findOne()

- To select only one document, we can use the findOne() method.
- This method accepts a query object. If left empty, it will return the first document it finds.

Practice 06: find() and findOne()


```
db.product.find()

db.product.findOne()
```

Basic Query

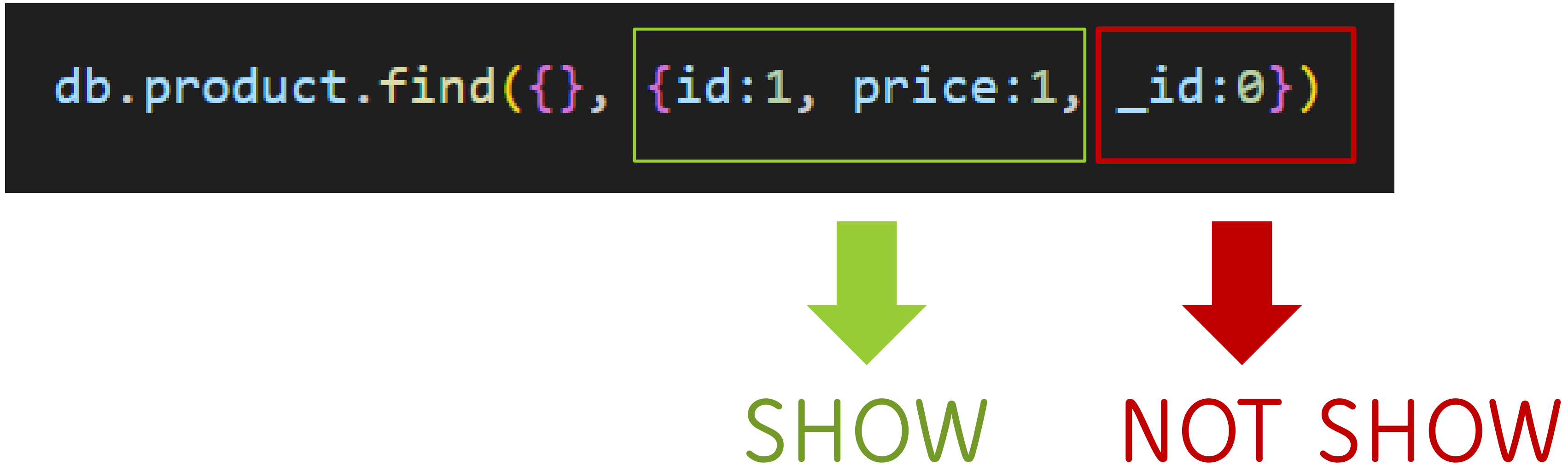
db.collection.find()

`db.collection.find(`**`query, projection`**`)`

- **query**: document
  - Optional. Specifies selection filter using query operators. To return all documents in a collection, omit this parameter or pass an empty document ({}).
- **projection**: document
  - Optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter.

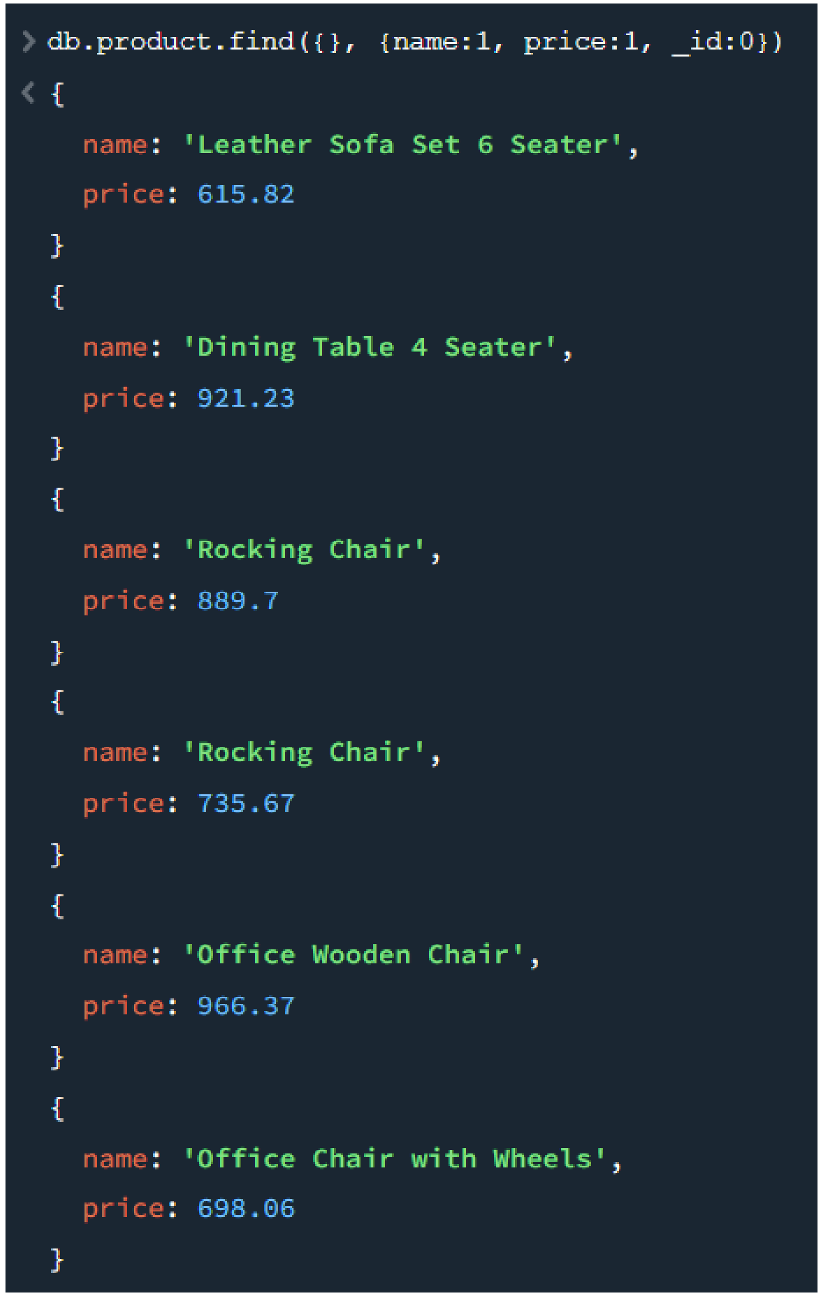
https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find

Practice 07: Projection



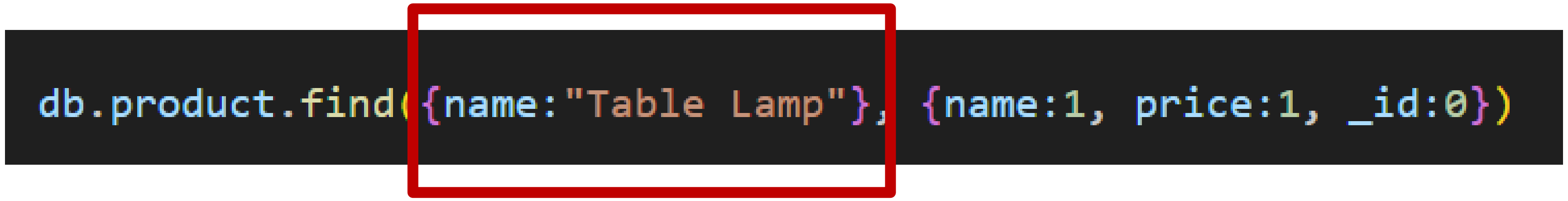
```
db.product.find({}, {id:1, price:1, _id:0})
```

Results

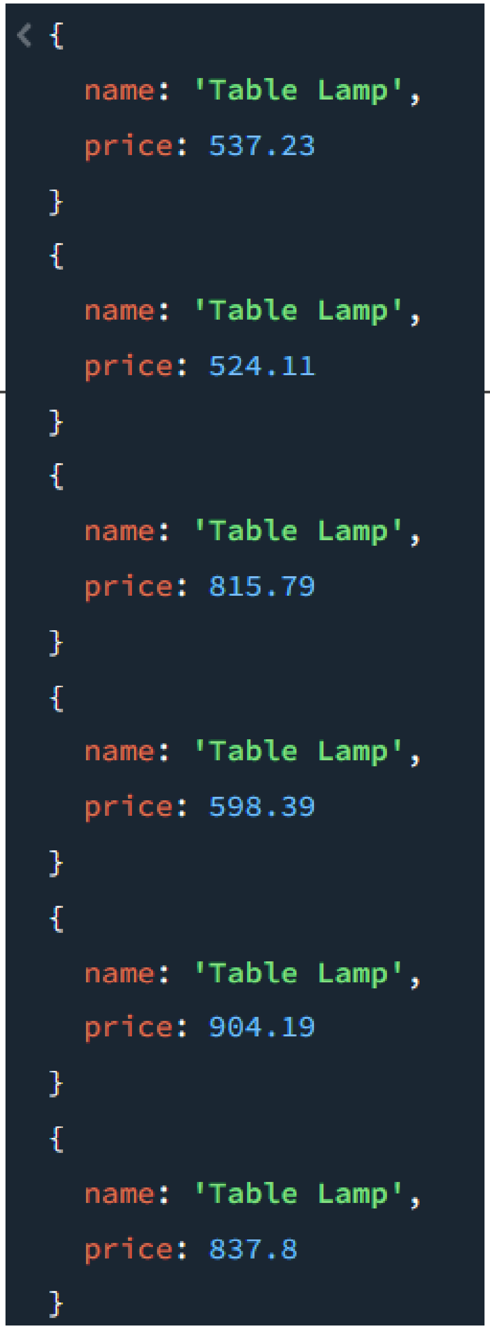


```
> db.product.find({}, {name:1, price:1, _id:0})
< {
    name: 'Leather Sofa Set 6 Seater',
    price: 615.82
  }
  {
    name: 'Dining Table 4 Seater',
    price: 921.23
  }
  {
    name: 'Rocking Chair',
    price: 889.7
  }
  {
    name: 'Rocking Chair',
    price: 735.67
  }
  {
    name: 'Office Wooden Chair',
    price: 966.37
  }
  {
    name: 'Office Chair with Wheels',
    price: 698.06
  }
```

Practice 08: Query Conditions



```
db.product.find({name:"Table Lamp"}, {name:1, price:1, _id:0})
```



```
< {
    name: 'Table Lamp',
    price: 537.23
  }
  {
    name: 'Table Lamp',
    price: 524.11
  }
  {
    name: 'Table Lamp',
    price: 815.79
  }
  {
    name: 'Table Lamp',
    price: 598.39
  }
  {
    name: 'Table Lamp',
    price: 904.19
  }
  {
    name: 'Table Lamp',
    price: 837.8
  }
```

Query Operators

Comparison

The following operators can be used in queries to compare values:

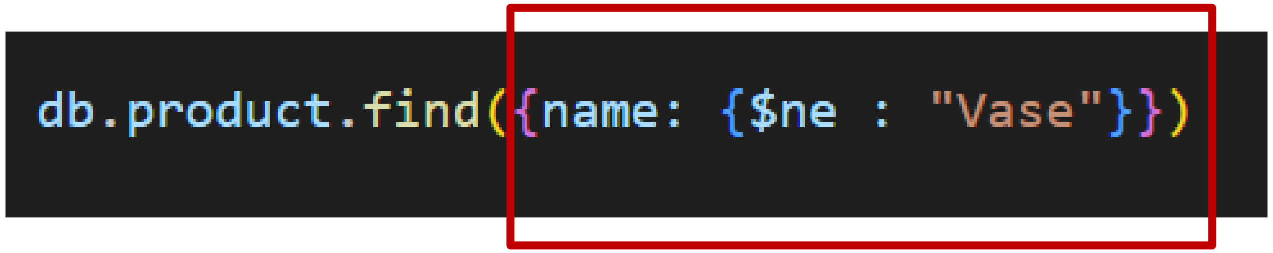
- `$eq`: Values are equal
- `$ne`: Values are not equal
- `$gt`: Value is greater than another value
- `$gte`: Value is greater than or equal to another value
- `$lt`: Value is less than another value
- `$lte`: Value is less than or equal to another value
- `$in`: Value is matched within an array

Logical

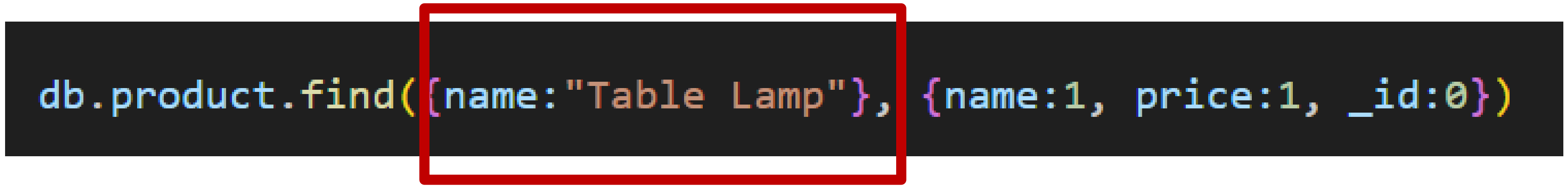
The following operators can logically compare multiple queries.

- `$and`: Returns documents where both queries match
- `$or`: Returns documents where either query matches
- `$nor`: Returns documents where both queries fail to match
- `$not`: Returns documents where the query does not match

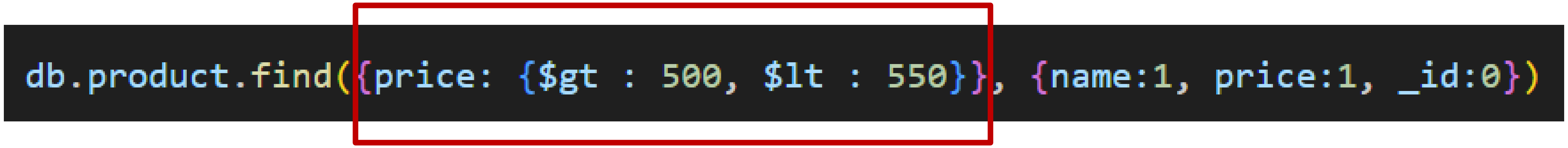
Practice 09: Query Operators



```
db.product.find({name: {$ne : "Vase"}})
```

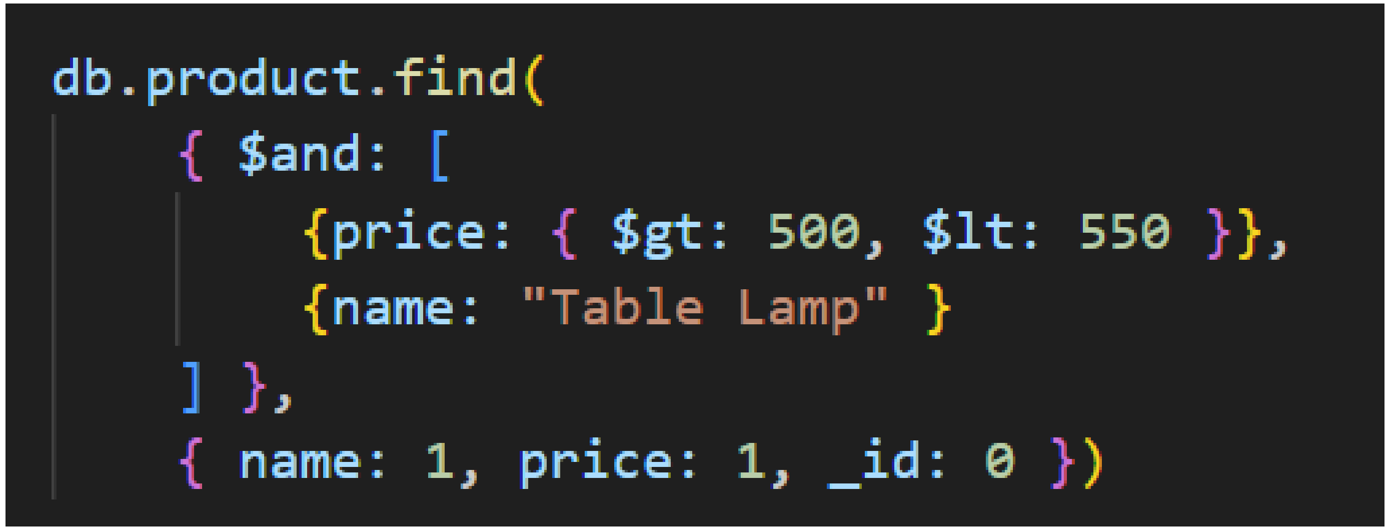


```
db.product.find({name:"Table Lamp"}, {name:1, price:1, _id:0})
```

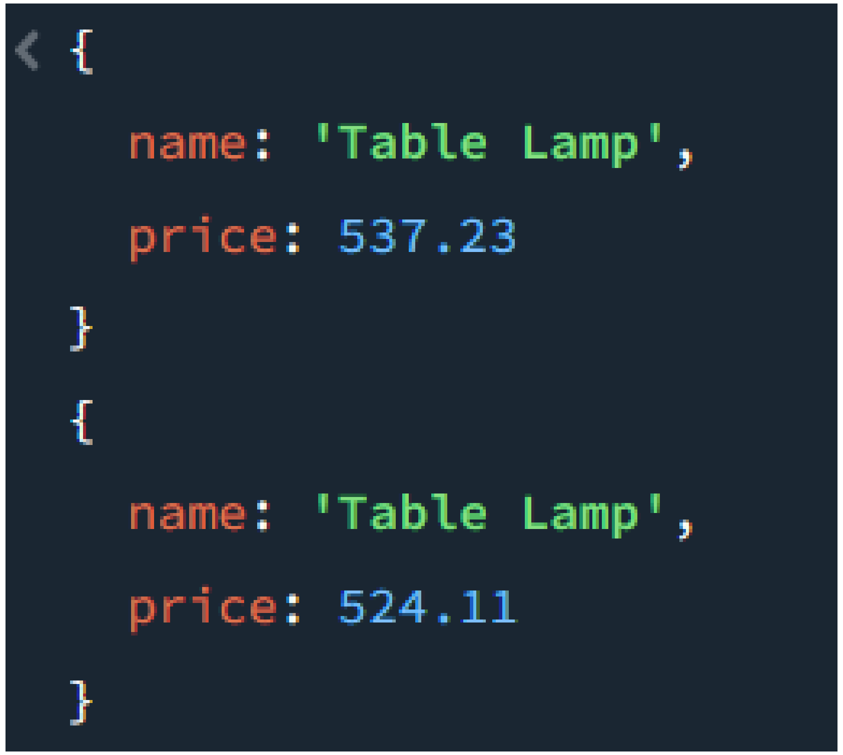


```
db.product.find({price: {$gt : 500, $lt : 550}}, {name:1, price:1, _id:0})
```

Practice 10: $and



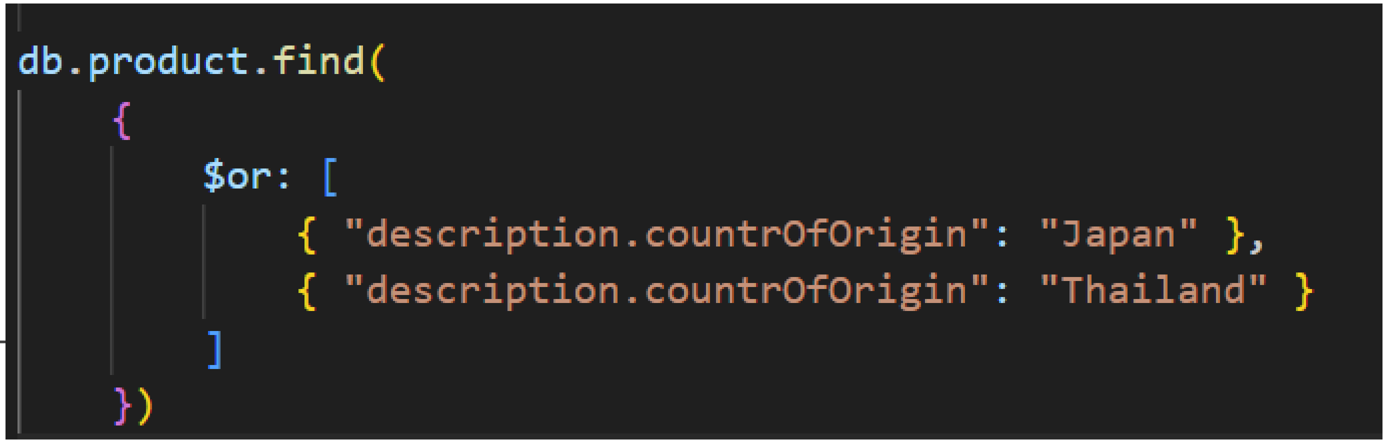
```
db.product.find(
    { $and: [
        {price: { $gt: 500, $lt: 550 }},
        {name: "Table Lamp" }
    ] },
    { name: 1, price: 1, _id: 0 })
```



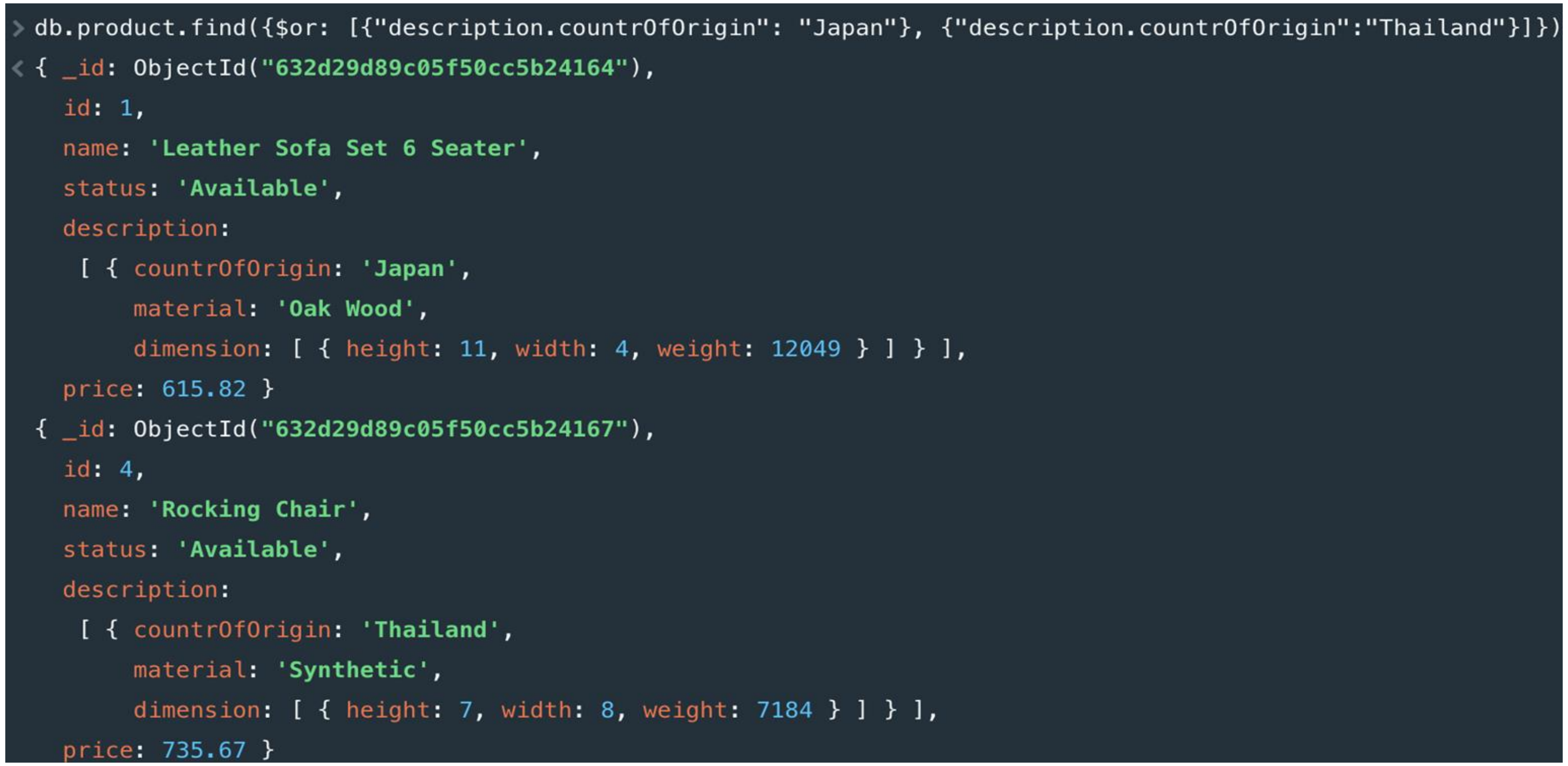
```
{
  name: 'Table Lamp',
  price: 537.23
}
{
  name: 'Table Lamp',
  price: 524.11
}
```

Practice 11: $or

Search product from Japan or Thailand



```
db.product.find(
    {
        $or: [
            { "description.countrOfOrigin": "Japan" },
            { "description.countrOfOrigin": "Thailand" }
        ]
    })
```

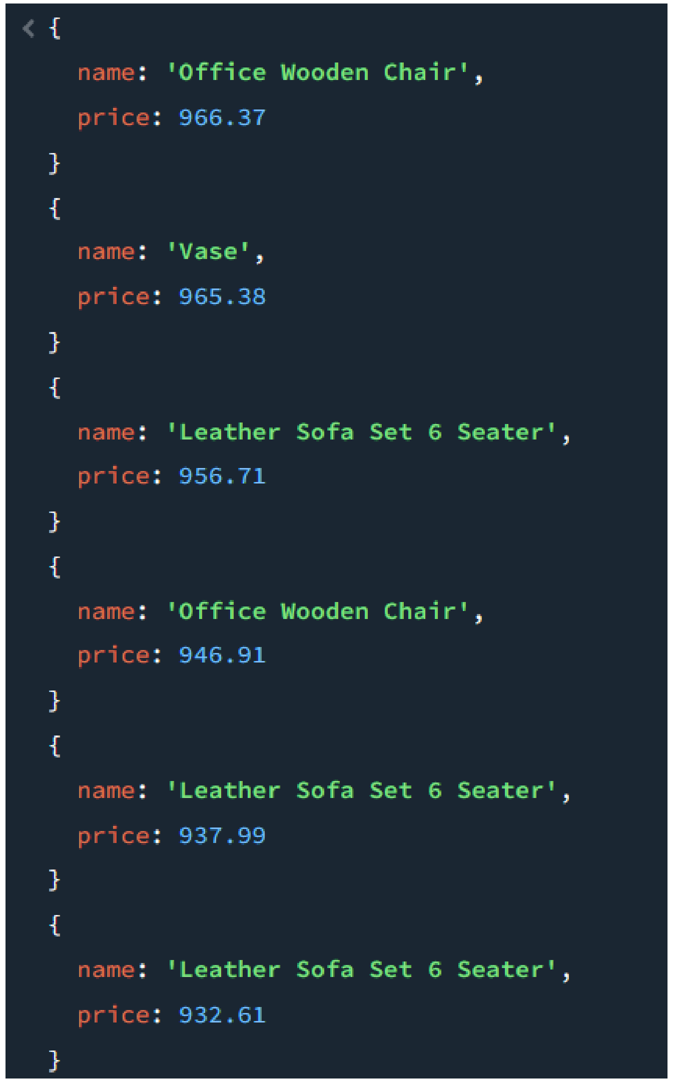


```
> db.product.find({$or: [{"description.countrOfOrigin": "Japan"}, {"description.countrOfOrigin":"Thailand"}]})
< { _id: ObjectId("632d29d89c05f50cc5b24164"),
    id: 1,
    name: 'Leather Sofa Set 6 Seater',
    status: 'Available',
    description:
     [ { countrOfOrigin: 'Japan',
         material: 'Oak Wood',
         dimension: [ { height: 11, width: 4, weight: 12049 } ] } ],
    price: 615.82 }
  { _id: ObjectId("632d29d89c05f50cc5b24167"),
    id: 4,
    name: 'Rocking Chair',
    status: 'Available',
    description:
     [ { countrOfOrigin: 'Thailand',
         material: 'Synthetic',
         dimension: [ { height: 7, width: 8, weight: 7184 } ] } ],
    price: 735.67 }
```

Practice 12: Sort()

```
db.product.find(
    {},
    {name:1, price:1, _id:0}
).sort({price: -1})
```

1 = ASC
-1 = DESC



```
{
  name: 'Office Wooden Chair',
  price: 966.37
}
{
  name: 'Vase',
  price: 965.38
}
{
  name: 'Leather Sofa Set 6 Seater',
  price: 956.71
}
{
  name: 'Office Wooden Chair',
  price: 946.91
}
{
  name: 'Leather Sofa Set 6 Seater',
  price: 937.99
}
{
  name: 'Leather Sofa Set 6 Seater',
  price: 932.61
}
```

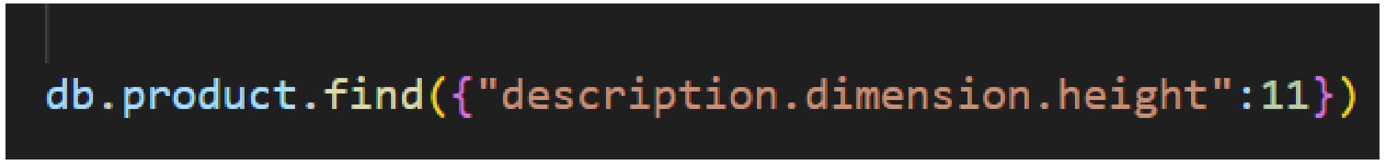
MongoDB Practice:

Part III

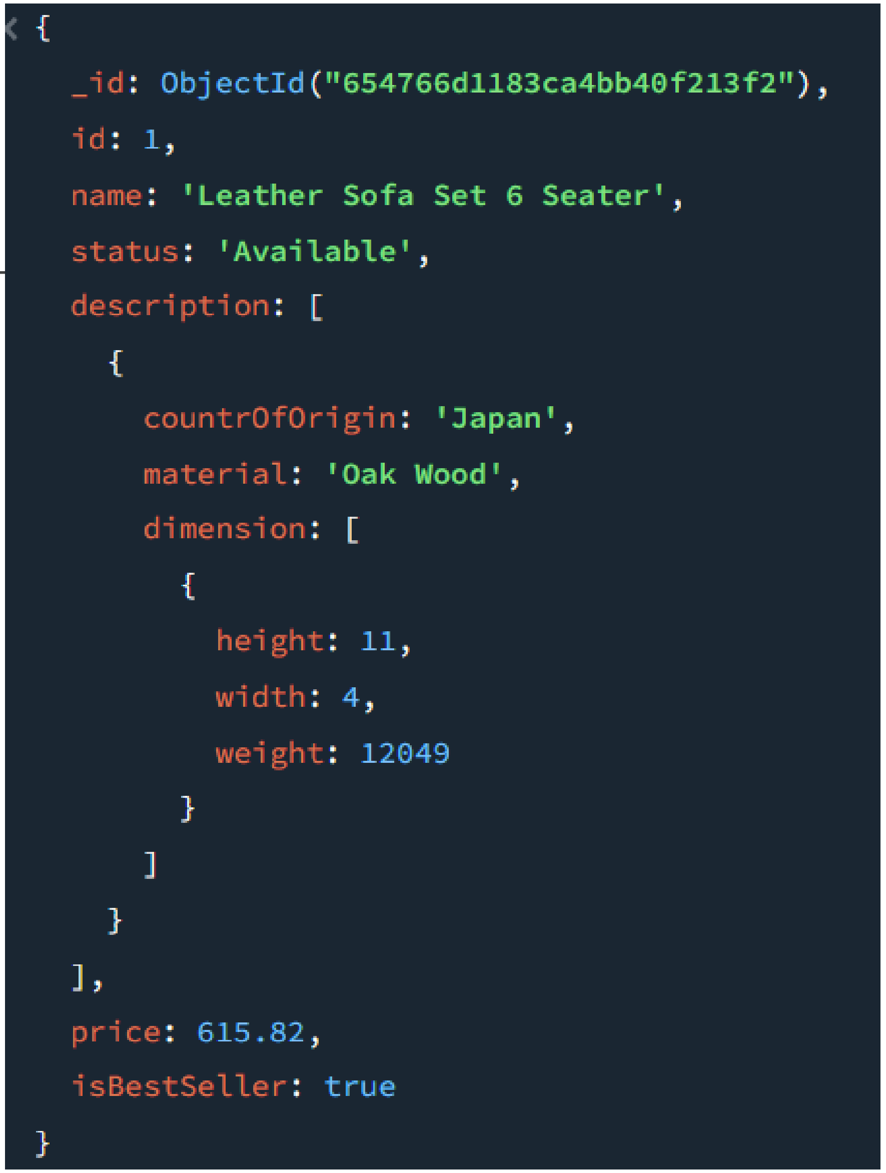
- Searching Products
  - Advanced Query
- Aggregate Query

Practice 13: Nested Field

Uses dot notation to access fields in an embedded document:



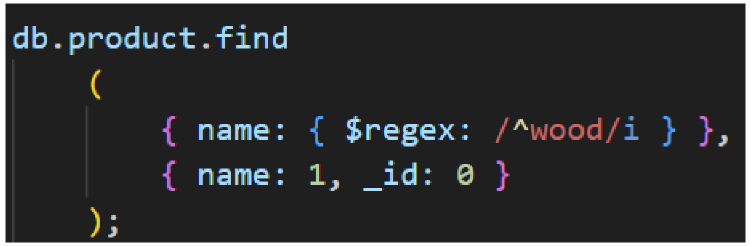
```
db.product.find({"description.dimension.height":11})
```



```
{
  _id: ObjectId("654766d1183ca4bb40f213f2"),
  id: 1,
  name: 'Leather Sofa Set 6 Seater',
  status: 'Available',
  description: [
    {
      countrOfOrigin: 'Japan',
      material: 'Oak Wood',
      dimension: [
        {
          height: 11,
          width: 4,
          weight: 12049
        }
      ]
    }
  ],
  price: 615.82,
  isBestSeller: true
}
```

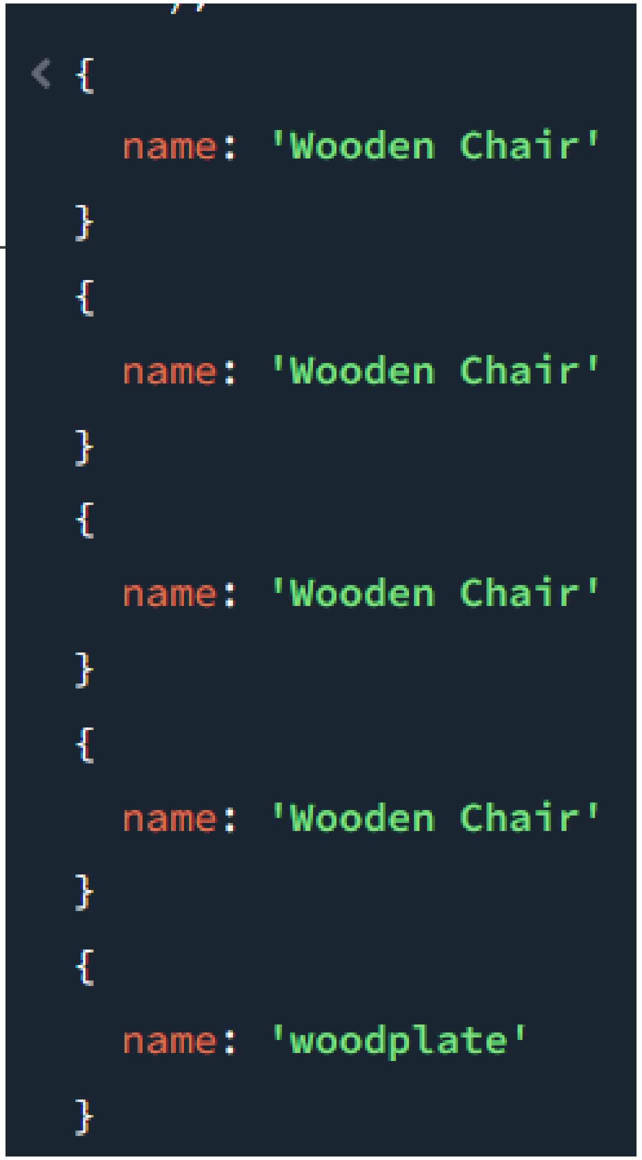
Practice 14: Pattern Matching

Pattern Matching using Regular Expression ($regex):



```
db.product.find
    (
        { name: { $regex: /^wood/i } },
        { name: 1, _id: 0 }
    );
```

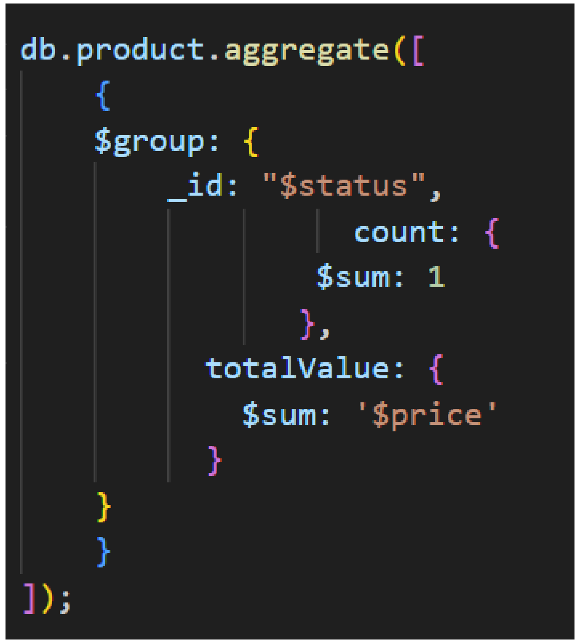
Start with the word "wood", match with any cases.



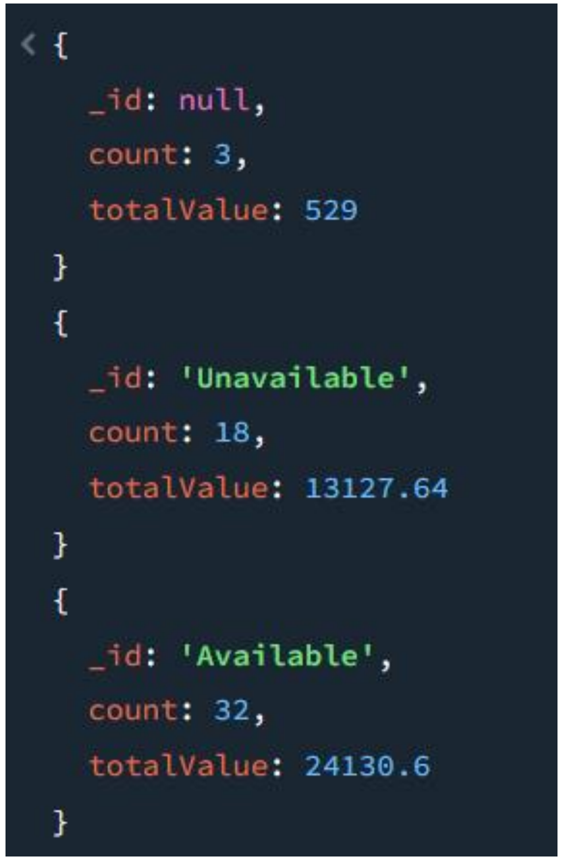
```
< {
    name: 'Wooden Chair'
  }
  {
    name: 'Wooden Chair'
  }
  {
    name: 'Wooden Chair'
  }
  {
    name: 'Wooden Chair'
  }
  {
    name: 'woodplate'
  }
```

Practice 15: Aggregate Query

Group products by their status, count them and sum their prices



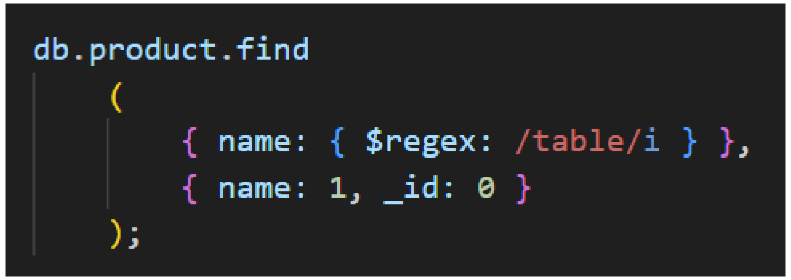
```
db.product.aggregate([
    {
    $group: {
        _id: "$status",
                count: {
            $sum: 1
        },
          totalValue: {
            $sum: '$price'
          }
    }
    }
]);
```



```
< {
    _id: null,
    count: 3,
    totalValue: 529
  }
  {
    _id: 'Unavailable',
    count: 18,
    totalValue: 13127.64
  }
  {
    _id: 'Available',
    count: 32,
    totalValue: 24130.6
  }
```

Bonus: Your Turn

1. Select products where their height less than 13 cm and the material is “Oak Wood”.
2. Group products by their status, count them, show the average, min, max and sum of their prices
3. What does the following query return:



```
db.product.find
    (
        { name: { $regex: /table/i } },
        { name: 1, _id: 0 }
    );
```

THANK YOU