

# Lab6: MongoDB

---

**Objectives:** Study NoSQL Document Model using MongoDB

**Estimated Time :** 3 hours

## Lab Instruction

---

### Outline

1. Mongo DB Overview
2. Setup Database on cloud
  - Create database
  - Create database user
  - Connect to database
3. CRUD Operations
  - Insert Document
  - Update Document
  - Delete Document
  - Query Data
    - Basic Query
    - Query Nested Field
    - Aggregate Function
    - Join

## 1. MongoDB Overview

### 1.1. About MongoDB

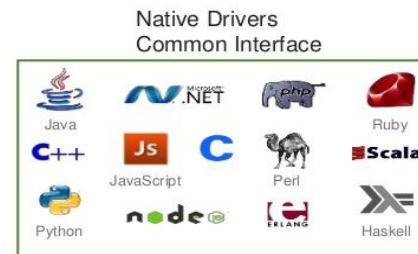
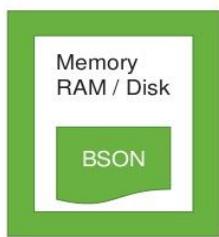
- Stores data in flexible, JSON-like documents
- Fields can vary from document to document and data structure can be changed over time
- Distributed database, high availability, horizontal scaling, and geographic distribution
- Free to use
- Scalability
  - Performance Scale: Sustaining 100,000+ database 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, often in multiple data centers

Ref. Examples of MongoDB users: <https://www.mongodb.com/mongodb-scale>



### 1.2. MongoDB Architecture

## So what is the MongoDB Architecture?



```

> db.collection.insert({company:"10gen", product:"MongoDB"})
>
> db.collection.findOne()
{
  "_id": ObjectId("506c1c2fc629bfe52792e86"),
  "company": "10gen",
  "product": "MongoDB"
}

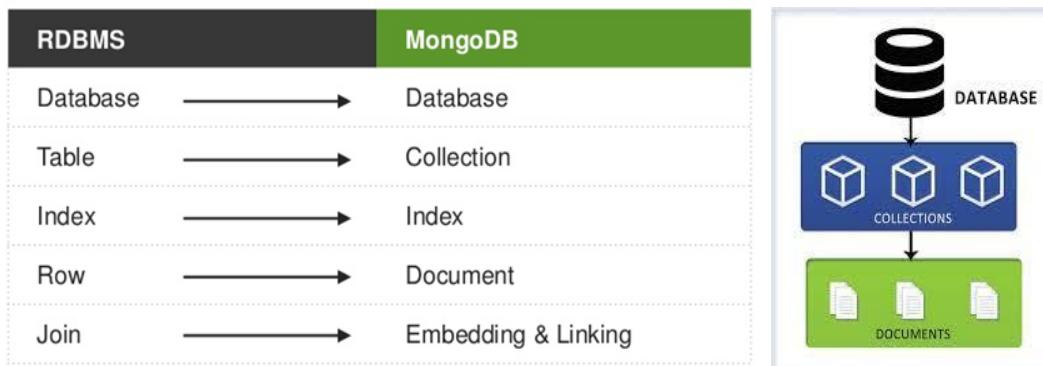
```

Command Line Shell  
Javascript Interpreter

[MongoDB Atlas](#)

[MongoDB Shell /](#)  
[MongoDB Compass](#)

## 1.3. MongoDB Terminology



## 2. Set up our cluster (= Database Server) in Mongo DB cloud

<https://cloud.mongodb.com/>

### 2.1 Sign up

We've launched a unified login experience that gives you access to MongoDB Cloud, Support, and Jira with a single identity.  
[LEARN MORE](#)

**Login**

Usually, this is the email you used to register.

[Next](#)

**Have an existing MongoDB deployment?**

Check out the Atlas Live Migration Service.

[See how it works](#)

[Register for a new account](#)



**Sign Up**

**Email Address**

**Password**

✓ 8 characters minimum

**First Name**

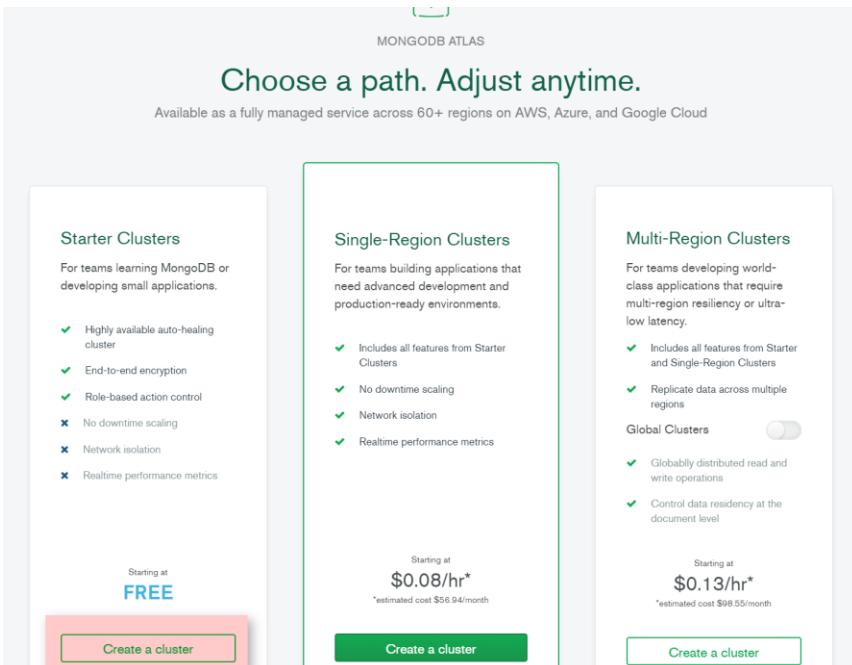
**Last Name**

**Phone Number**

**Company Name**

**Job Function**

## 2.2 Create a Cluster



**MONGODB ATLAS**

**Choose a path. Adjust anytime.**

Available as a fully managed service across 60+ regions on AWS, Azure, and Google Cloud

**Starter Clusters**  
For teams learning MongoDB or developing small applications.

- ✓ Highly available auto-healing cluster
- ✓ End-to-end encryption
- ✓ Role-based access control
- ✗ No downtime scaling
- ✗ Network isolation
- ✗ Realtime performance metrics

Starting at **FREE**

**Create a cluster**

**Single-Region Clusters**  
For teams building applications that need advanced development and production-ready environments.

- ✓ Includes all features from Starter Clusters
- ✓ No downtime scaling
- ✓ Network isolation
- ✓ Realtime performance metrics

Starting at **\$0.08/hr\***

\*estimated cost \$56.94/month

**Create a cluster**

**Multi-Region Clusters**  
For teams developing world-class applications that require multi-region resiliency or ultra-low latency.

- ✓ Includes all features from Starter and Single-Region Clusters
- ✓ Replicate data across multiple regions
- ✓ Global Clusters
- ✓ Globally distributed read and write operations
- ✓ Control data residency at the document level

Starting at **\$0.13/hr\***

\*estimated cost \$98.55/month

**Create a cluster**

Select Cloud Provider and Region and Click “Create Cluster”

CLUSTERS > CREATE A STARTER CLUSTER

Choose your cloud provider and region

**Next**

recommended some of our most popular options, but feel free to customize your cluster to check our documentation.

**Cloud Provider & Region**

AWS, N. Virginia (us-east-1) ▾

**aws** Google CloudPlatform Azure

Create a **free tier cluster** by selecting a region with **FREE TIER AVAILABLE** and choosing the M0 cluster tier below.

★ Recommended region ⓘ

NORTH AMERICA	EUROPE	ASIA
N. Virginia (us-east-1) ★ FREE TIER AVAILABLE	Ireland (eu-west-1) ★ FREE TIER AVAILABLE	Singapore (ap-southeast-1) ★ FREE TIER AVAILABLE
Oregon (us-west-2) ★ FREE TIER AVAILABLE	Frankfurt (eu-central-1) ★ FREE TIER AVAILABLE	Mumbai (ap-south-1) FREE TIER AVAILABLE
AUSTRALIA		
Sydney (ap-southeast-2) ★		

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

Back Create Cluster

Cloud Provider & Region

GCP, Singapore (asia-southeast1) ▾

**aws** Google CloudPlatform Azure

Create a **free tier cluster** by selecting a region with **FREE TIER AVAILABLE** and choosing the M0 cluster tier below.

★ Recommended region ⓘ

NORTH AMERICA / SOUTH AMERICA	EUROPE / MIDDLE EAST / AFRICA	ASIA PACIFIC
Iowa (us-central1) ★ FREE TIER AVAILABLE	Belgium (europe-west1) ★ FREE TIER AVAILABLE	Taiwan (asia-east1) ★ FREE TIER AVAILABLE
Singapore (asia-southeast1) ★ FREE TIER AVAILABLE		

**Cluster Tier** M0 Sandbox (Shared RAM, 512 MB Storage) >  
Encrypted

**Additional Settings** MongoDB 4.0, No Backup >

**Cluster Name** Cluster0 >

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

Back Create Cluster

The screenshot shows the MongoDB Atlas Clusters page. At the top, a blue banner displays the message: "We are deploying your changes: 0 of 3 servers complete (current action: provisioning 3 servers)". Below this, the "Clusters" section shows a cluster named "Cluster0" (Version 4.0.12) in the "SANDBOX" tier. A "CONNECT" button is visible. On the left, a sidebar lists "ATLAS", "Clusters" (selected), "Data Lake BETA", "SECURITY", "Database Access", "Network Access", "Advanced", "PROJECT", "Access Management", and "Connect to Atlas". Under "Connect to Atlas", there is a checklist with one item completed: "Build your first cluster". Other items include "Create your first database user", "Whitelist your IP address", "Load Sample Data (Optional)", and "Connect to your cluster". A progress bar indicates 20% completion. A "Get Started" button is at the bottom.

## 2.3 Create your first database user

This screenshot is identical to the one above, but the "Create your first database user" checkbox in the checklist has been highlighted with a red rectangle. All other elements, including the deployment progress bar, cluster details, and sidebar, remain the same.

The screenshot shows the MongoDB Atlas interface. In the top left, there's a sidebar with 'ATLAS' at the top, followed by 'Clusters' (which is highlighted), 'Data Lake BETA', 'SECURITY' (with 'Database Access' and 'Network Access' options), and 'Advanced'. To the right of the sidebar is a search bar with the placeholder 'Find a cluster...'. Below the search bar is a 'SANDBOX' section containing a green dot next to 'Cluster0' and 'Version 4.0.12'. A callout box with a green border and rounded corners points to the 'Database Access' link in the sidebar, containing the text 'Click here to manage your project's database users'. Another callout box points to the '+ ADD NEW USER' button in the 'User Management' section below, containing the text 'Click here to add your first database user'. The 'User Management' section also includes a 'MongoDB Roles' link.

**Add New User**

**SCRAM Authentication**  
SCRAM is MongoDB's default authentication method.

User Name: ying  
e.g. new-user\_31

User Password: ying HIDE  
 Autogenerate Secure Password

**User Privileges**

Atlas admin  Read and write to any database  Only read any database  Select Custom Role

Add Default Privileges

Save as temporary user Cancel **Add User**

**Database Access**

User Name	Authentication Method	MongoDB Roles
d_ying	SCRAM	readWriteAnyDatabase@admin

**PROJECT**

**Access Management**

**Connect to Atlas**

Follow this checklist to get started.

- Build your first cluster
- Create your first database user
- Whitelist your IP address
- Load Sample Data (Optional)
- Connect to your cluster

40%

No thanks

## 1.4 Allow Hosts that can access the Cluster

**Network Access**

Advanced

Click here to manage IP address configuration and VPC peering

+ ADD IP ADDRESS

Add Whitelist Entry

Add a whitelist entry using either CIDR notation or a single IP address. [Learn more.](#)

Whitelist Entry:	0.0.0.0/0
Comment:	Optional comment describing this entry

Save as temporary whitelist

## 1.5 (Optional) Load Sample DB

mongoDB. Atlas All Clusters

CONTEXT Project 0 AIT > PROJECT 0

**Clusters**

Find a cluster...

**SANDBOX**

Cluster0 Version 4.0.12

...

Operations R: 0 W: 0

Click "... " and then press Load Sample Dataset.

CLUSTER TIER M0 Sandbox (General)

REGION GCP / Singapore (asia-southeast1)

TYPE Replica Set - 3 nodes

LINKED STITCH APP None Linked

Last 6 Hours

Connections 0

Last 6 Hours

Load Sample Dataset

We've created a sample dataset to help you test features on Cluster0.

**Sample Dataset**  
Size: ~350 MB

Please confirm that you want to load this sample dataset.

## 1.6 Create Database in Mongo Atlas

The screenshot shows the MongoDB Atlas Clusters overview page for Project 0. The left sidebar has 'Clusters' selected under 'ATLAS'. The main area shows a cluster named 'Cluster0' (Version 4.0.10) in a 'SANDBOX' environment. The 'COLLECTIONS' tab is highlighted with a pink box. Below it, the 'INSTANCE SIZE' is listed as 'M0 Sandbox (General)'.

The screenshot shows the MongoDB Atlas Cluster details page for 'Cluster0'. The left sidebar has 'Clusters' selected under 'ATLAS'. The main area shows the 'Collections' tab is selected. A red box highlights the '+ Create Database' button. On the right, the 'sample\_airbnb' database is listed with its details: DATABASE SIZE: 89.99MB, INDEX SIZE: 472KB, TOTAL COLLECTIONS: 1. Below it is a table showing collection details:

Collection Name	Documents	Documents Size	Documents Avg
listingsAndReviews	5555	89.99MB	16.59KB

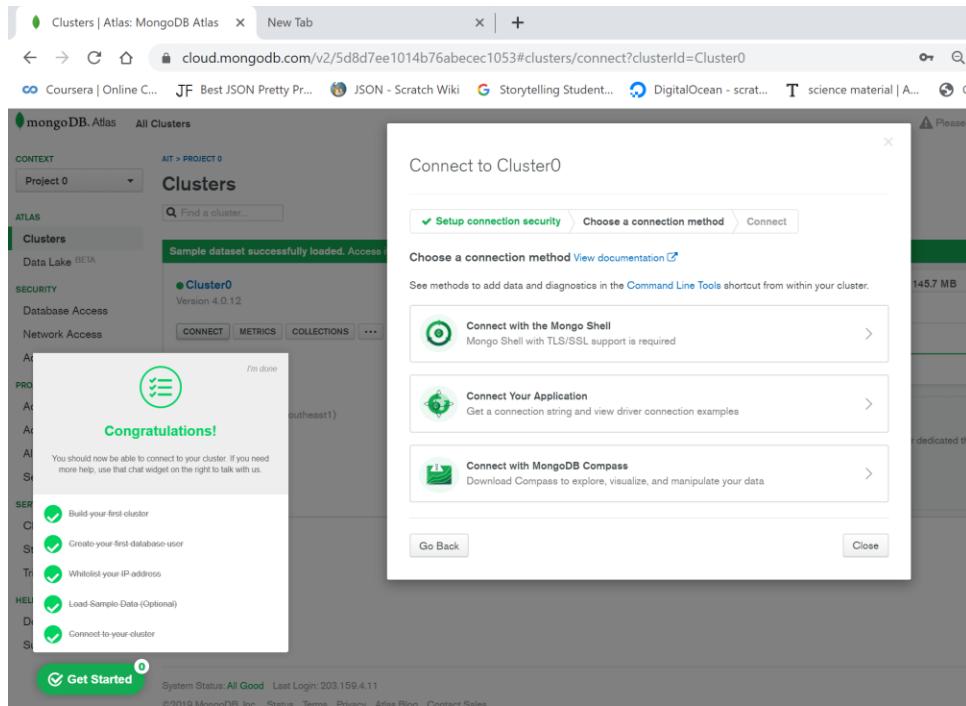
Create Database name =shop101 and Collection name = customer.

The screenshot shows the 'Create Database' form. It has two input fields: 'DATABASE NAME' containing 'shop101' and 'COLLECTION NAME' containing 'customer'. Below the fields is a note about capped collections. At the bottom are 'Cancel' and 'Create' buttons.

1.7 Connect to the Cluster using the User. There are 3 methods to connect: Mongo Shell, Your App and Mongo Compass.

The screenshot shows the MongoDB Atlas Clusters page for Project 0. A prominent blue banner at the top says "We are deploying your changes (current action: configuring MongoDB)". On the left, there's a sidebar with sections like ATLAS (Clusters, Data Lake BETA), SECURITY (Database Access, Network Access, Advanced), PROJECT (Access Management, Connect to Atlas), and HELP (Build your first cluster, Create your first database user, Whitelist your IP address, Lead Sample Data Optional). A red box highlights the "CONNECT" button under the "Connect to Atlas" section. Below it, a green box highlights the "Get Started" button. The main area shows a cluster named "Cluster0" (Version 4.0.12) with a status of "OPERATIONAL". It includes tabs for CONNECT, METRICS, and COLLECTIONS. To the right, there are sections for Operations (R: 0 W: 0) and Connections (0). A message at the bottom says "Once your cluster is created, click here to connect".

This screenshot is similar to the previous one but includes a callout box around the "CONNECT" button. The callout contains the text "Once your cluster is created, click here to connect". The rest of the interface is identical, showing the "Cluster0" cluster details and monitoring sections.



Install Shell or Compass and using the connection string to connect. Example of Connection string: `mongodb+srv://ying:<password>@cluster0-dfi1d.gcp.mongodb.net/<database name>`  
In this tutorial we will use Shell

## Connect to Cluster0

✓ Setup connection security > Choose a connection method > Connect

Choose a connection method [View documentation](#)

See methods to add data and diagnostics in the [Command Line Tools](#) shortcut from within your cluster.

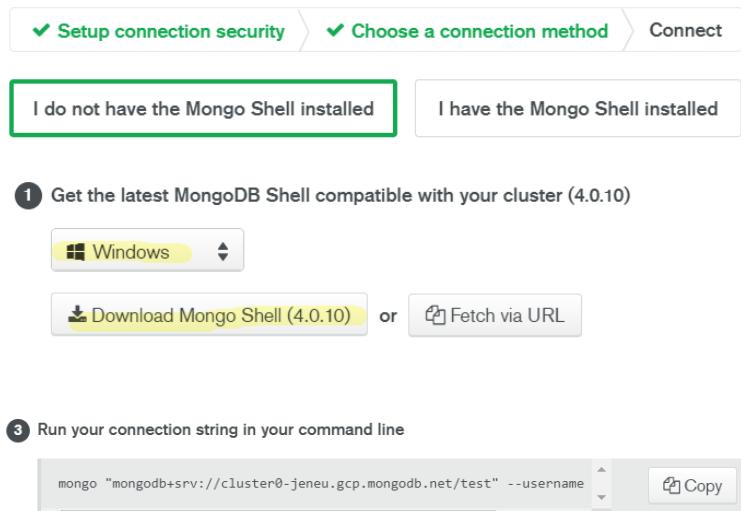
**Connect with the Mongo Shell**  
Mongo Shell with TLS/SSL support is required >

**Connect Your Application**  
Get a connection string and view driver connection examples >

**Connect with MongoDB Compass**  
Download Compass to explore, visualize, and manipulate your data >

[Go Back](#) [Close](#)

## Connect to Cluster0



### ③ Run your connection string in your command line

```
mongo "mongodb+srv://cluster0-jeneu.gcp.mongodb.net/test" --username
      ^ Copy
```

You will be prompted for the password for the <username> user's (MongoDB User) username.  
When entering your password, make sure that any special characters are [URL encoded](#).

```
C:\ Command Prompt
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\KK>d:

D:>cd D:\Program\shell\bin

D:\Program\shell\bin>dir
 Volume in drive D is Data
 Volume Serial Number is CCE2-1D5D

 Directory of D:\Program\shell\bin

07/11/2019  04:04 PM    <DIR>        .
07/11/2019  04:04 PM    <DIR>        ..
05/28/2019  09:59 PM           18,461,184 mongo.exe
                           1 File(s)   18,461,184 bytes
                           2 Dir(s)  37,391,085,568 bytes free

D:\Program\shell\bin>mongo "mongodb+srv://cluster0-jeneu.gcp.mongodb.net/MyFirstDB" --username user1
```

```

D:\Program\shell\bin>mongo "mongodb+srv://cluster0-jeneu.gcp.mongodb.net/MyFirstDB" --username user1
MongoDB shell version v4.0.10
Enter password:
connecting to: mongodb://cluster0-shard-00-01-jeneu.gcp.mongodb.net:27017,cluster0-shard-00-02-jeneu.gcp.mongodb.net:27017,cluster0-shard-00-00-jeneu.gcp.mongodb.net:27017/MyFirstDB?authSource=admin&gssapiServiceName=mongodb&replicaSet=Cluster0-shard-0-ssl=true
2019-07-12T16:16:46.200+0700 I NETWORK [js] Starting new replica set monitor for Cluster0-shard-0/cluster0-shard-0-01-jeneu.gcp.mongodb.net.:27017,cluster0-shard-00-02-jeneu.gcp.mongodb.net:27017,cluster0-shard-00-00-jeneu.gcp.mongodb.net.:27017
2019-07-12T16:16:46.581+0700 I NETWORK [ReplicaSetMonitor-TaskExecutor] Successfully connected to cluster0-shard-0-00-jeneu.gcp.mongodb.net.:27017 (1 connections now open to cluster0-shard-00-00-jeneu.gcp.mongodb.net:27017 with a 5 second timeout)
2019-07-12T16:16:46.581+0700 I NETWORK [js] Successfully connected to cluster0-shard-00-02-jeneu.gcp.mongodb.net.:27017 (1 connections now open to cluster0-shard-00-02-jeneu.gcp.mongodb.net:27017 with a 5 second timeout)
2019-07-12T16:16:46.961+0700 I NETWORK [ReplicaSetMonitor-TaskExecutor] Successfully connected to cluster0-shard-0-01-jeneu.gcp.mongodb.net:27017 (1 connections now open to cluster0-shard-00-01-jeneu.gcp.mongodb.net:27017 with a 5 second timeout)
2019-07-12T16:16:46.964+0700 I NETWORK [js] Successfully connected to cluster0-shard-00-01-jeneu.gcp.mongodb.net.:27017 (1 connections now open to cluster0-shard-00-01-jeneu.gcp.mongodb.net:27017 with a 5 second timeout)
2019-07-12T16:16:46.995+0700 I NETWORK [ReplicaSetMonitor-TaskExecutor] changing hosts to Cluster0-shard-0/cluster0-shard-00-jeneu.gcp.mongodb.net:27017,cluster0-shard-00-01-jeneu.gcp.mongodb.net:27017,cluster0-shard-00-00-jeneu.gcp.mongodb.net:27017 from Cluster0-shard-0/cluster0-shard-00-00-jeneu.gcp.mongodb.net:27017,cluster0-shard-00-01-jeneu.gcp.mongodb.net:27017,cluster0-shard-00-02-jeneu.gcp.mongodb.net:27017
2019-07-12T16:16:47.609+0700 I NETWORK [js] Successfully connected to cluster0-shard-00-01-jeneu.gcp.mongodb.net:27017 (1 connections now open to cluster0-shard-00-01-jeneu.gcp.mongodb.net:27017 with a 0 second timeout)
2019-07-12T16:16:47.889+0700 I NETWORK [js] Successfully connected to cluster0-shard-00-02-jeneu.gcp.mongodb.net:27017 (1 connections now open to cluster0-shard-00-02-jeneu.gcp.mongodb.net:27017 with a 5 second timeout)
2019-07-12T16:16:50.162+0700 I NETWORK [ReplicaSetMonitor-TaskExecutor] Successfully connected to cluster0-shard-0-00-jeneu.gcp.mongodb.net:27017 (1 connections now open to cluster0-shard-00-00-jeneu.gcp.mongodb.net:27017 with a 5 second timeout)
Implicit session: session { "id" : UUID("a4b38900-8ff1-4a4a-aa9c-176fa290f2c4") }
MongoDB server version: 4.0.10
MongoDB Enterprise Cluster0-shard-0:PRIMARY <-->
```

mongo "mongodb+srv://cluster0-jeneu.gcp.mongodb.net/<database name=shop101>" --username <username>

### 3. Insert, Update, Delete Document

You can do DDL operation in Mongo Atlas, Mongo Compass or Mongo Shell.

#### 3.1 Insert customers in “shop101” database using **Mongo Atlas**

Click “Collection” -> click “customer” -> click “Insert Document”

The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with 'ATLAS' selected, showing 'Clusters' and 'Data Lake (BETA)'. Under 'Clusters', 'Project 0' is selected, and 'Cluster0' is shown. In the main area, 'Collections' is selected under 'PROJECT > PROJECT 0 > CLUSTERS'. The 'shop101.customer' collection is listed, with its size (0B), documents (0), and index size (4KB) displayed. Below the collection list, there are 'Find', 'Indexes', and 'Aggregation' buttons. At the bottom right, there's a 'FILTER {"filter": "example"}' input field and 'Find' and 'Reset' buttons. A red box highlights the 'customer' collection in the list and the 'INSERT DOCUMENT' button at the top right.

## Insert Document

---

```

1   _id : ObjectId("5d8e32391c9d440000ea487d")
2   cid : "001"
3   name : "Lisa"
4   address : Object
5     no : "99"
6     road : "main street"
7     province : "BKK"
8   age : 30

```

ObjectId
String
String
Object
String
String
String
Int32

---

## Insert Document

---

```

1   _id : ObjectId("5d9074bb1c9d4400004170b2")
2   cid : "002"
3   name : "Bob"
4   age : 20
5   address : Object
6     no : "1"
7     road : "TU street"
8     province : "Phatum"
9     zipcode : "12020"

```

ObjectId
String
String
Int32
Object
String
String
String
String

---

### 3.2 Insert inventory using Mongo Shell

Get connection string from Atlas = (mongo "mongodb+srv://cluster0-dfi1d.gcp.mongodb.net/shop101" --username ying)

```
Command Prompt - mongo "mongodb+srv://cluster0-dfi1d.gcp.mongodb.net/shop101" --username ying
D:\_Phd_Candidate\AIT-4Aug19\DMM-TA\MongoDB\shell\bin>mongo "mongodb+srv://cluster0-dfi1d.gcp.mongodb.net/shop101" --username ying
MongoDB shell version v4.2.0
Enter password:
```

## Insert Command

[db.collection.insertMany\(\)](#)

Syntax:

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

```
db.inventory.insertOne(
  { item: "canvas", qty: 100, price: 500, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } })
db.inventory.insertMany([
  { item: "pen", qty: 25, price: 200, tags: ["red"], size: { h: 14, w: 1, uom: "cm" } },
  { item: "mat", qty: 85, price: 99, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },
  { item: "mousepad", qty: 25, price: 29, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "in" } }
])
```

```
Command Prompt - mongo "mongodb+srv://cluster0-dfi1d.gcp.mongodb.net/shop101" --username ying
...
return result;
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.inventory.insertOne(
...   { item: "canvas", qty: 100, price: 500, tags: [ "cotton" ], size: { h: 28, w: 35.5, uom: "cm" } })
{
  "acknowledged" : true,
  "insertedId" : ObjectId("5d8e3f03ba0c8c019a09414e")
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.inventory.insertMany([
...   { item: "journal", qty: 25, price: 200, tags: [ "blank", "red" ], size: { h: 14, w: 21, uom: "cm" } },
...   { item: "mat", qty: 85, price: 99, tags: [ "gray" ], size: { h: 27.9, w: 35.5, uom: "cm" } },
...   { item: "mousepad", qty: 25, price: 29, tags: [ "gel", "blue" ], size: { h: 19, w: 22.85, uom: "cm" } }
...
])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("5d8e3f03ba0c8c019a09414f"),
    ObjectId("5d8e3f03ba0c8c019a094150"),
    ObjectId("5d8e3f03ba0c8c019a094151")
  ]
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

## Verify

The screenshot shows the MongoDB Atlas interface. On the left, the sidebar includes sections for ATLAS (Clusters, SECURITY, PROJECT, SERVICES, HELP), a 'Get Started' button, and a 'Project 0' dropdown. The main area shows 'Databases: 7 Collections: 20'. The 'shop101' database is selected. The 'customer' and 'inventory' collections are highlighted with a red box. The right panel shows the 'shop101.inventory' collection with a total size of 505B, 4 documents, and 16KB indexes. A query result for the filter '{ "filter": "example" }' is shown, with 1-4 of 4 results:

```

{
  "_id": ObjectId("5d8e3ae9ba0c0c019a09414a"),
  "item": "Canvas",
  "qty": 100,
  "tags": [
    "cotton"
  ],
  "size": {
    "h": 28,
    "w": 35.5,
    " uom": "cm"
  }
},
{
  "_id": ObjectId("5d8e3bb0ba0c0c019a09414b"),
  "item": "Journal",
  "qty": 25,
  "tags": [
    "paper"
  ],
  "size": {
    "h": 20,
    "w": 30,
    " uom": "cm"
  }
},
{
  "_id": ObjectId("5d8e3bb0ba0c0c019a09414c"),
  "item": "mousepad",
  "qty": 25,
  "tags": [
    "mousepad"
  ],
  "size": {
    "h": 15,
    "w": 22,
    " uom": "cm"
  }
}

```

Note: if you want to UPDATE or DELETE:

@Atlas you can click the document and click small buttons on the right corner.

The screenshot shows the 'shop101.customer' collection in MongoDB Atlas. The left sidebar shows the 'customer' collection under the 'shop101' database. The right panel shows a single document result for a filter ('example') with one result:

```

{
  "_id": ObjectId("5d8e32391c9d40000ea087d"),
  "cid": "001",
  "name": "Lisa",
  "address": {
    "city": "London"
  },
  "age": 30
}

```

On the right side of the document preview, there are several small icons for document management, including edit, delete, and copy, which are circled in red in the screenshot.

(@Shell see more command in <https://docs.mongodb.com/manual/crud/>.)

### 3.4 Insert sales in “shop101” database using Mongo Shell

```

db.sales.insertOne(
  { item: [{pid: "canvas", qty: 1, price: 500}, {pid: "pen", qty: 1, price: 200}], customer:"Lisa",
  total:700, couponused:true })
db.sales.insertOne(
  { item: [{pid: "mousepad", qty: 1, price: 29}], customer:"Bob", total:29, couponused:true })
db.sales.insertOne(
  { item: [{pid: "mat", qty: 1, price: 99}], customer:"Bob", total:99, couponused:false })

```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.find().pretty()
{
    "_id" : ObjectId("5d8f62648fb57f941738209b"),
    "item" : [
        {
            "pid" : "canvas",
            "qty" : 1,
            "price" : 500
        },
        {
            "pid" : "pen",
            "qty" : 1,
            "price" : 200
        }
    ],
    "customer" : "Lisa",
    "total" : 700,
    "couponused" : true
}
{
    "_id" : ObjectId("5d8f63588fb57f941738209c"),
    "item" : [
        {
            "pid" : "mousepad",
            "qty" : 1,
            "price" : 29
        }
    ],
    "customer" : "Bob",
    "total" : 29,
    "couponused" : true
}
{
    "_id" : ObjectId("5d9027dd8fb57f941738209d"),
    "item" : [
        {
            "pid" : "mat",
            "qty" : 1,
            "price" : 99
        }
    ],
    "customer" : "Bob",
    "total" : 99,
    "couponused" : false
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

### 3.5 Update customer add status field

Alter table customer add status char(1)  + Update customer set status ="A"	db.customer.updateMany( { }, {\$set: {status: "A"} } )
Update customer set status ="I"  where name ="Lisa"	db.customer.updateMany( {name:"Lisa" }, {\$set: {status: "I"} } )

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.updateMany( { }, {$set: {status: "A"})  
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }  
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.updateMany( {name:"Lisa" }, {$set: {status: "I"})  
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }  
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

## 4. Query Data

### [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>

*Note: pretty() - displays the results in a formatted way.*

### 4.1 Select

SQL SELECT Statements	MongoDB find() Statements
<pre>SELECT * FROM Customer</pre>	<pre>db.customer.find().pretty()</pre>
<pre>MongoDB Enterprise Cluster0-shard-0:PRIMARY&gt; db.customer.find().pretty() {     "_id" : ObjectId("5d8e32391c9d440000ea487d"),     "cid" : "001",     "name" : "Lisa",     "address" : {         "no" : "99",         "road" : "main street",         "province" : "BKK"     },     "age" : 30,     "status" : "I" } {     "_id" : ObjectId("5d8e34411c9d440000ea487f"),     "cid" : "002",     "name" : "Bob",     "address" : {         "no" : "1",         "road" : "TU street",         "province" : "Phatum",         "zipcode" : "12020"     },     "age" : 20,     "status" : "A" } MongoDB Enterprise Cluster0-shard-0:PRIMARY&gt;</pre>	

- Projection (return some fields)

```
SELECT cid,name
FROM Customer
db.Customer.find(
    { },
    {cid:1, name:1,_id:0}
)
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find({}, {cid:1, name:1, _id:0})
{ "cid" : "001", "name" : "Lisa" }
{ "cid" : "002", "name" : "Bob" }
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

## 4.2 Order by

Select * from customer order by name	db.customer.find().sort({name:1})
Select * from customer order by name desc	db.customer.find().sort({name:-1})

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find().sort({name:1})
{ "_id" : ObjectId("5d8e34411c9d440000ea487f"), "cid" : "002", "name" : "Bob", "address" : { "no" : "1" , "road" : "TU street", "province" : "Phathum", "zipcode" : "12020" }, "age" : 20, "status" : "A" }
{ "_id" : ObjectId("5d8e32391c9d440000ea487d"), "cid" : "001", "name" : "Lisa", "address" : { "no" : "9" , "road" : "main street", "province" : "BKK" }, "age" : 30, "status" : "I" }
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find().sort({name:-1})
{ "_id" : ObjectId("5d8e32391c9d440000ea487d"), "cid" : "001", "name" : "Lisa", "address" : { "no" : "9" , "road" : "main street", "province" : "BKK" }, "age" : 30, "status" : "I" }
{ "_id" : ObjectId("5d8e34411c9d440000ea487f"), "cid" : "002", "name" : "Bob", "address" : { "no" : "1" , "road" : "TU street", "province" : "Phathum", "zipcode" : "12020" }, "age" : 20, "status" : "A" }
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

## 4.3 Where condition

### 4.3.1 =, !=, >, <, <=, >=

```
SELECT *
FROM Customer
WHERE Name ="Lisa"
db.Customer.find(
    {name :"Lisa"}
)
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find({name:"Lisa"}).pretty()
{
    "_id" : ObjectId("5d8e32391c9d440000ea487d"),
    "cid" : "001",
    "name" : "Lisa",
    "address" : {
        "no" : "99",
        "road" : "main street",
        "province" : "BKK"
    },
    "age" : 30
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

```
SELECT *
```

```
FROM Customer
```

```
WHERE name != "Lisa"
```

```
db.Customer.find(
```

```
    { name : { $ne: "Lisa" } }
```

```
)
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find({name:{$ne:"Lisa"} }).pretty()
{
    "_id" : ObjectId("5d8e34411c9d440000ea487f"),
    "cid" : "002",
    "name" : "Bob",
    "address" : {
        "no" : "1",
        "road" : "TU street",
        "province" : "Phatum",
        "zipcode" : "12020"
    },
    "age" : 20
}
```

```
SELECT *
```

```
FROM Customer
```

```
WHERE age > 25
```

```
db.Customer.find(
```

```
    { age: { $gt: 25 } }
```

```
)
```

```
SELECT *
```

```
FROM Customer
```

```
WHERE age < 25
```

```
db.Customer.find(
```

```
    { age: { $lt: 25 } }
```

```
)
```

## Query Operators

- ◎ [\\$eq](#): Matches values that are equal to a specified value.
- ◎ [\\$gt](#): Matches values that are greater than a specified value.
- ◎ [\\$gte](#): Matches values that are greater than or equal to a specified value.
- ◎ [\\$in](#): Matches any of the values specified in an array.
- ◎ [\\$lt](#): Matches values that are less than a specified value.
- ◎ [\\$lte](#): Matches values that are less than or equal to a specified value.
- ◎ [\\$ne](#): Matches all values that are not equal to a specified value.
- ◎ [\\$nin](#): Matches none of the values specified in an array.

### 4.3.2 AND

<pre><b>SELECT</b> * <b>FROM</b> Customer <b>WHERE</b> age &gt; 25 <b>AND</b>   age &lt;= 50</pre>	<pre>db.Customer.find(   { age: { \$gt: 25, \$lte: 50 } } )</pre>
--	---

```
db.customer.find({$and : [{status: "A"}, {age:20}]})
```

#### 4.3.3 OR

<pre><b>SELECT</b> * <b>FROM</b> Customer <b>WHERE</b> status = "A" <b>OR</b>   age = 20</pre>	<pre>db.Customer.find(   { \$or: [ { status: "A" } , { age: 20 } ] } )</pre>
--	--

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find({$or:[{status:"A"},{age: 20}]}).pretty()
{
  "_id" : ObjectId("5d8e34411c9d44000ea487f"),
  "cid" : "002",
  "name" : "Bob",
  "address" : {
    "no" : "1",
    "road" : "TU street",
    "province" : "Phatum",
    "zipcode" : "12020"
  },
  "age" : 20,
  "status" : "A"
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

#### 4.3.4 LIKE

<pre><b>SELECT</b> * <b>FROM</b> Customer <b>WHERE</b> name <b>like</b> "%is%"</pre>	<pre>db.Customer.find( { name : /is/ } )</pre>
--	--

-or-

	<pre>db.Customer.find( { name : { \$regex: /is/ } } )</pre>
--	---

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find({name :/is/}).pretty()
{
  "_id" : ObjectId("5d8e32391c9d44000ea487d"),
  "cid" : "001",
  "name" : "Lissa",
  "address" : {
    "no" : "99",
    "road" : "main street",
    "province" : "BKK"
  },
  "age" : 30,
  "status" : "I"
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

### 4.3.5 IN

```
db.Customer .find( { status: { $in: [ "A", "D" ] } } )
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find({status:{$in: ["A","D"]}}).pretty()
{
    "_id" : ObjectId("5d8e34411c9d44000ea487f"),
    "cid" : "002",
    "name" : "Bob",
    "address" : {
        "no" : "1",
        "road" : "TU street",
        "province" : "Phatum",
        "zipcode" : "12020"
    },
    "age" : 20,
    "status" : "A"
}
```

### 4.3.6 Exists

Returns all the documents from the customer collection where zipcode field does not exists:

```
db.customer.find( {
    address.zipcode: { $exists: false }
} )
```

## 4.4 Query on Nested field

Uses **dot notation** to access fields in an embedded document:

- Select all documents where uom of size is in “in” unit

```
db.inventory.find( { "size.uom": "in" } )
```

- selects items where their height less than 15 CM.

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.inventory.find( { "size.h": { $lt: 15 }, "size.uom": "cm" } ).pretty()
{
    "_id" : ObjectId("5d8e3f03ba0c8c019a09414f"),
    "item" : "pen",
    "qty" : 25,
    "price" : 200,
    "tags" : [
        "red"
    ],
    "size" : {
        "h" : 14,
        "w" : 1,
        "uom" : "cm"
    }
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

## .AGGREGATE

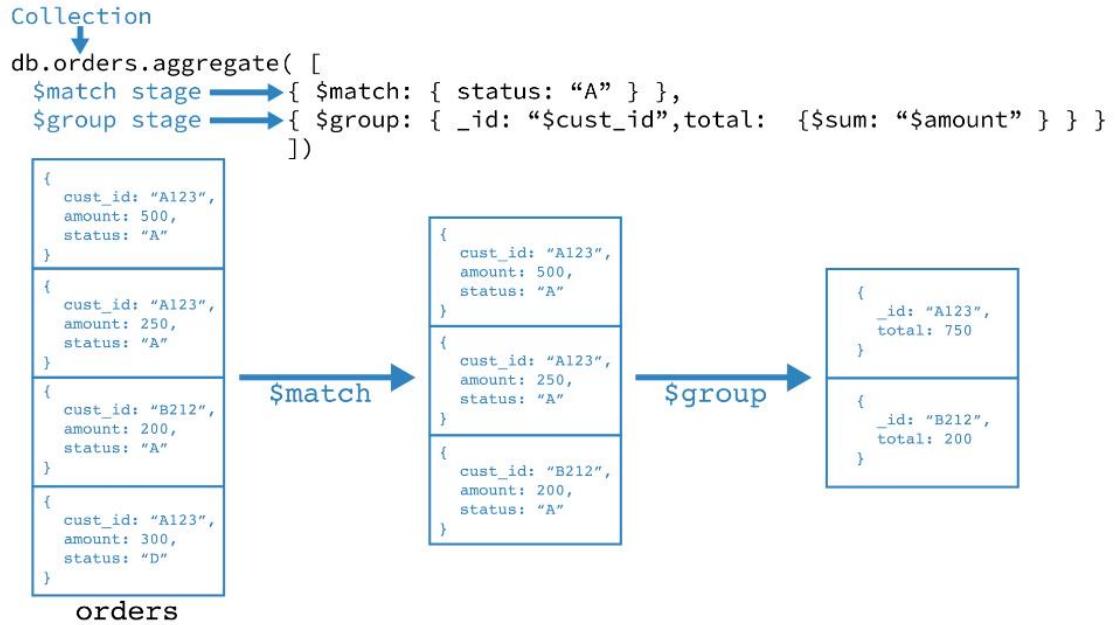
### 4.5 Aggregate

Aggregation operations group values from multiple documents together and can perform a variety of operations on the grouped data to return a single result.

[SQL Terms, Functions, and Concepts](#) [MongoDB Aggregation Operators](#)

WHERE	\$match
GROUP BY	\$group
HAVING	\$match
SELECT	\$project
ORDER BY	\$sort
LIMIT	\$limit
SUM()	\$sum
COUNT()	\$sum \$sortByCount
join	\$lookup

### Example



#### 4.5.1 Sum

```

db.sales.aggregate([
  { $match: { couponused:true } },
  { $group: { _id: "$customer", total: { $sum: "$total" } } }
])
  
```

```

MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([
  { $match: { couponused:true } },
  { $group: { _id: "$customer", total: { $sum: "$total" } } } ])
{ "_id" : "Bob", "total" : 29 }
{ "_id" : "Lisa", "total" : 700 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
  
```

All sales record:

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.find().pretty()
{
    "_id" : ObjectId("5d8f62648fb57f941738209b"),
    "item" : [
        {
            "pid" : "canvas",
            "qty" : 1,
            "price" : 500
        },
        {
            "pid" : "pen",
            "qty" : 1,
            "price" : 200
        }
    ],
    "customer" : "Lisa",
    "total" : 700,
    "couponused" : true
}
{
    "_id" : ObjectId("5d8f63588fb57f941738209c"),
    "item" : [
        {
            "pid" : "mousepad",
            "qty" : 1,
            "price" : 29
        }
    ],
    "customer" : "Bob",
    "total" : 29,
    "couponused" : true
}
{
    "_id" : ObjectId("5d9027dd8fb57f941738209d"),
    "item" : [
        {
            "pid" : "mat",
            "qty" : 1,
            "price" : 99
        }
    ],
    "customer" : "Bob",
    "total" : 99,
    "couponused" : false
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

```
db.sales.aggregate([
    { $group: { _id: "$customer", total: { $sum: "$total" } } }
])
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([ { $group: { _id: "$customer", total: { $sum: "$total" } } } ])
{ "_id" : "Bob", "total" : 128 }
{ "_id" : "Lisa", "total" : 700 }
```

#### 4.5.2 Count

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([ { $group: { _id: "$customer", total: { $sum: 1 } } } ])
{ "_id" : "Bob", "total" : 2 }
{ "_id" : "Lisa", "total" : 1 }
```

Note .FIND() also has COUNT() function

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.find({customer:"Lisa"}).count()
1
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.find({customer:"Bob"}).count()
2
```

## 4.5.3 Average, Min, Max

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.find().pretty()
{
    "_id" : ObjectId("5d8f62648fb57f941738209b"),
    "item" : [
        {
            "pid" : "canvas",
            "qty" : 1,
            "price" : 500
        },
        {
            "pid" : "pen",
            "qty" : 1,
            "price" : 200
        }
    ],
    "customer" : "Lisa",
    "total" : 700,
    "couponused" : true
}
{
    "_id" : ObjectId("5d8f63588fb57f941738209c"),
    "item" : [
        {
            "pid" : "mousepad",
            "qty" : 1,
            "price" : 29
        }
    ],
    "customer" : "Bob",
    "total" : 29,
    "couponused" : true
}
{
    "_id" : ObjectId("5d9027dd8fb57f941738209d"),
    "item" : [
        {
            "pid" : "mat",
            "qty" : 1,
            "price" : 99
        }
    ],
    "customer" : "Bob",
    "total" : 99,
    "couponused" : false
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

```
db.sales.aggregate([
    { $group: { _id: "$customer", average: { $avg: "$total" } } }
])

db.sales.aggregate([
    { $group: { _id: "$customer", minimum: { $min: "$total" } } }
])

db.sales.aggregate([
    { $group: { _id: "$customer", maximum: { $max: "$total" } } }
])
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([
  { $group: { _id: "$customer", average: { $avg: "$total" } } }
])
{ "_id" : "Bob", "average" : 64 }
{ "_id" : "Lisa", "average" : 700 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([
  { $group: { _id: "$customer", minimum: { $min: "$total" } } }
])
{ "_id" : "Bob", "minimum" : 29 }
{ "_id" : "Lisa", "minimum" : 700 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([
  { $group: { _id: "$customer", maximum: { $max: "$total" } } }
])
{ "_id" : "Bob", "maximum" : 99 }
{ "_id" : "Lisa", "maximum" : 700 }
```

#### 4.5.4 Having

```
db.sales.aggregate([
  { $group: { _id: "$customer", total: { $sum: "$total" } } },
  { $match: {total: { $gt: 150 } } }
])
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([
  { $group: { _id: "$customer", total: { $sum: "$total" } } },
  { $match: {total: { $gt: 150 } } }
])
{ "_id" : "Lisa", "total" : 700 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

#### 4.5.5 Sort and Limit

```
db.sales.aggregate([
  { $group: { _id: "$customer", total: { $sum: "$total" } } },
  { $sort: { total: -1 } }
])
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([
  { $group: { _id: "$customer", total: { $sum: "$total" } } },
  { $sort: { total: -1 } }
])
{ "_id" : "Lisa", "total" : 700 }
{ "_id" : "Bob", "total" : 128 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([
  { $group: { _id: "$customer", total: { $sum: "$total" } } },
  { $sort: { total: 1 } }
])
{ "_id" : "Bob", "total" : 128 }
{ "_id" : "Lisa", "total" : 700 }
```

```
db.sales.aggregate([
  { $group: { _id: "$customer", total: { $sum: "$total" } } },
  { $sort: { total: -1 } },
  { $limit: 1 }
])
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([
  { $group: { _id: "$customer", total: { $sum: "$total" } } },
  { $sort: { total: -1 } },
  { $limit: 1 }
])
{ "_id" : "Lisa", "total" : 700 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

#### 4.6 Join

- Syntax

```
{
  $lookup:
    {
      from: <collection to join>,
      localField: <field from the input documents>,
      foreignField: <field from the documents of the "from" collection>,
      as: <output array field>
    }
}
```

Example from

<https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/index.html#lookup-single-equality> :

```

db.orders.insert([
  { "_id" : 1, "item" : "almonds", "price" : 12, "quantity" : 2 },
  { "_id" : 2, "item" : "pecans", "price" : 20, "quantity" : 1 },
  { "_id" : 3 }
])

db.inventory.insert([
  { "_id" : 1, "sku" : "almonds", description: "product 1", "instock" : 120 },
  { "_id" : 2, "sku" : "bread", description: "product 2", "instock" : 80 },
  { "_id" : 3, "sku" : "cashews", description: "product 3", "instock" : 60 },
  { "_id" : 4, "sku" : "pecans", description: "product 4", "instock" : 70 },
  { "_id" : 5, "sku": null, description: "Incomplete" },
  { "_id" : 6 }
])

```

Command:

Join orders กับ Inventory

```

db.orders.aggregate([
  {
    $lookup:
      {
        from: "inventory",
        localField: "item",
        foreignField: "sku",
        as: "inventory_docs"
      }
  }
])

```

- Insert Inventory info into Orders
- Orders.item = Inventory.sku
- New Field = Inventory\_docs

Result:

The operation returns the following documents:

```
{  
  "_id" : 1,  
  "item" : "almonds",  
  "price" : 12,  
  "quantity" : 2,  
  "inventory_docs" : [  
    { "_id" : 1, "sku" : "almonds", "description" : "product 1", "instock" : 120 }  
  ]  
}
```

Try our database:

- Include customer info into sale

```
db.sales.aggregate([  
  {  
    $lookup:  
      {  
        from: "customer",  
        localField: "customer",  
        foreignField: "name",  
        as: "cust_info"  
      }  
  }  
]).pretty()
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([
...   {
...     $lookup:
...       {
...         from: "customer",
...         localField: "customer",
...         foreignField: "name",
...         as: "cust_info"
...       }
...   }
... ]).pretty()
{
  "_id" : ObjectId("5d8f62648fb57f941738209b"),
  "item" : [
    {
      "pid" : "canvas",
      "qty" : 1,
      "price" : 500
    },
    {
      "pid" : "pen",
      "qty" : 1,
      "price" : 200
    }
  ],
  "customer" : "Lisa",
  "total" : 700,
  "couponused" : true,
  "cust_info" : [
    {
      "_id" : ObjectId("5d8e32391c9d440000ea487d"),
      "cid" : "001",
      "name" : "Lisa",
      "address" : {
        "no" : "99",
        "road" : "main street",
        "province" : "BKK"
      },
      "age" : 30,
      "status" : "I"
    }
  ]
}
```

- Include the info under customer in sale

```
db.sales.aggregate([
  {
    $lookup:
      {
        from: "customer",
        localField: "customer",
        foreignField: "name",
        as: "customer.info"
      }
  }
])
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([
  { $lookup:
    {
      from: "customer",
      localField: "customer",
      foreignField: "name",
      as: "customer.info"
    }
  }
])
[{"_id": ObjectId("5d8f62648fb57f941738209b"), "item": [{"pid": "canvas", "qty": 1, "price": 500}, {"pid": "pen", "qty": 1, "price": 200}], "customer": [{"info": [{"_id": ObjectId("5d8e32391c9d440000ea487d"), "cid": "001", "name": "Lisa", "address": {"no": "99", "road": "main street", "province": "BKK"}, "age": 30, "status": "I"}]}, {"total": 700, "couponused": true}]}
```

## Reference:

<https://docs.mongodb.com/manual/crud/>  
<https://docs.mongodb.com/manual/>  
<https://docs.mongodb.com/manual/reference/sql-comparison/#examples>  
<https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/index.html#lookup-single-equality>

## Lab Assignment

---

1. Install Mongo Shell and connect to RDBProject database using the below command

***mongo "mongodb+srv://dmmcluster.f1uo1.gcp.mongodb.net/RDBProject" --username st\_dmm***

2. Convert your Peer-review feedback into the below JSON format and insert into feedback collection

	A	B	C	D	E	F	G	H	
1	<b>Peer Feedback Form</b>								
2	Reviewer Name	xx	xx	xx					
3	Meeting Room#	xx							
4									
5									
6	Team#	Project Title	Member#	Presenter Name	Time Taken (min)	Summary of Presentation	Strengths	Suggestions	
7	19	Bike sharing	19-1	Mr. XYZ	15	good presentation	precise and easy to understand	practice presentation skills	
8	20								
9									
10									
11									
12									
13									

```
{
  "member_id": "19-1",
  "presenter_name": "Mr. XYZ",
  "project_id" : "19",
  "project_name": "Bike sharing",
  "summary_of_presentation": "the presentation was well prepared and_",
  "strengths": [ "precise", "easy to understand" ],
  "suggestions": ["practice presentation skills" ]
  ...
}
```

... means you can add any other information because NOSQL is schemaless.

3. Insert the peer review into the feedback table in the database.
4. Write MongoDB command to retrieve the following information, capture the running screen and the returned result.

[TASK1] List your presentation feedback reviewed by friends.

[TASK2] How many strengths are provided in total?

[TASK3] What is the average number of suggestion for a reviewer?

Submission System: [Google Classroom](#)

Total TASKS: 3