

# MongoDB Data Analytics Portfolio

---

Name: Daniel Cyril Obon

Student ID: C2650218

Course: Computer Science (Year 2)

Assignment: Relational and NO SQL Databases (Task 2)

---

## Topic 1: Start Here - Intro to MongoDB

The screenshot shows a browser window with the URL [https://tinstrukt.com/?track=https://play.instrukt.com/embed/mongodb/tracks/introduction-to-the-ide?token=em\\_zLoa1VDlqE\\_Glb9nB&jwt=eyJhbGciOiUzIiNlsInR5C16lkpXVC...](https://tinstrukt.com/?track=https://play.instrukt.com/embed/mongodb/tracks/introduction-to-the-ide?token=em_zLoa1VDlqE_Glb9nB&jwt=eyJhbGciOiUzIiNlsInR5C16lkpXVC...). The main area is a terminal window titled 'terminal' showing MongoDB logs and help output. To the right is a sidebar with tabs for 'Progress', 'Hints', 'Review and Solved Code', and 'Solved Code'. The 'Review and Solved Code' tab is active, containing text and bullet points about MongoDB commands.

Current Mongosh Log ID: 659f58fe3c3c918080ee810  
Connecting to: mongod+srv://<credentials>@instrukttest.3xfvk.mongodb.net/?appName=mongosh+1.10.6  
Using Mongosh: 6.0.12  
Using Mongosh: 1.10.6

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

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (<https://www.mongodb.com/legal/privacy-policy>). You can opt-out by running the `disableTelemetry()` command.

Atlas atlas-d3opcw-shard-0 [primary] test> db.help()

**database Class:**

getMongo	Returns the current database connection
getDB	Returns the name of the DB
getCollectionNames	Returns an array containing the names of all collections in the current database.
getCollectionInfos	Returns an array of documents with collection information, i.e. collection name and options
, for the current database.	
runCommand	Runs an arbitrary command on the database.
adminCommand	Runs an arbitrary command against the admin database.
aggregate	Runs a specified admin/diagnostic pipeline which does not require an underlying collection.
getSiblingDB	Returns another database without modifying the db variable in the shell environment.
getCollection	Returns a collection or a view object that is functionally equivalent to using the db.collectionName.
dropDatabase	Removes the current database, deleting the associated data files.
createUser	Creates a new user for the database on which the method is run. db.createUser() returns a document on the database.
updateUser	Updates the users profile on the database on which you run the method. An update to a field completely replaces the previous fields values. This includes updates to the users roles array.
changeUserPassword	Updates a users password. Run the method in the database where the user is defined, i.e. the database you created the user.
logout	Ends the current authentication session. This function has no effect if the current session is not authenticated.
dropUser	Removes the user from the current database.
dropAllUsers	Removes all users from the current database.
auth	Allows a user to authenticate to the database from within the shell.
grantRolesToUser	Grants additional roles to a user.

This is the Review and Solved Code tab. Use this tab to compare your answer to the correct answer and review the code. For example:

- You are connected to an Atlas cluster. The connection information is shown in the terminal window.
- Use the `db.help()` to return a list of commonly used commands and a brief description of each command.
- In the terminal, there is a list of commands such as `getDB`, which returns the name of the database, and `getUser`, which returns information about the specified user.

Solved Code

```
db.help()
```

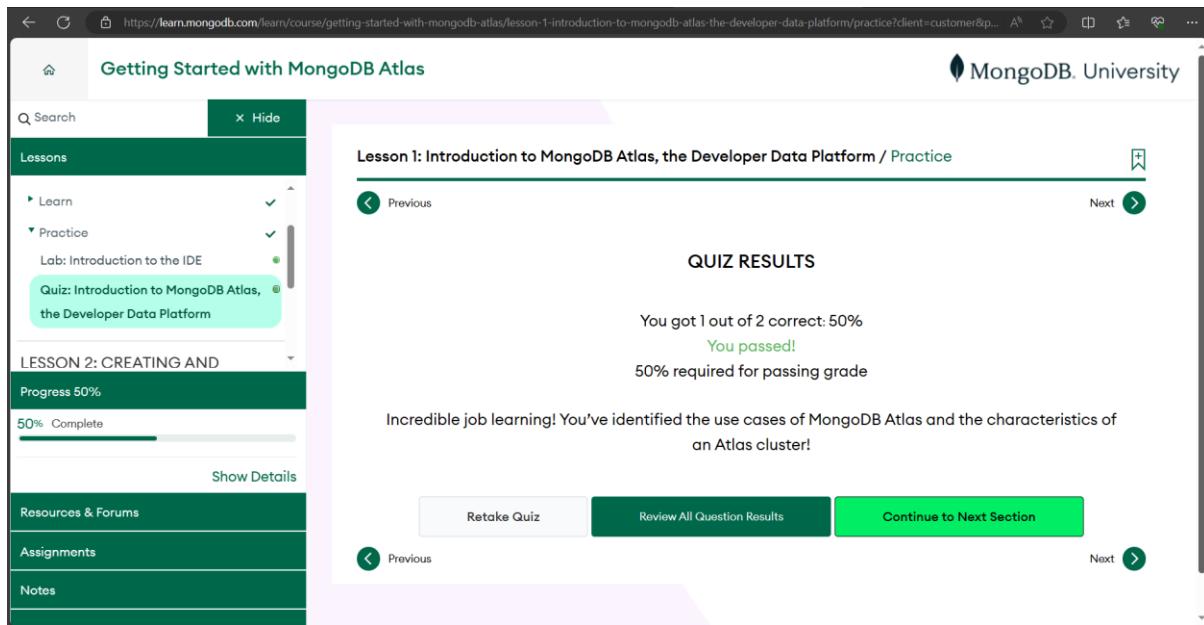
Troubleshooting

Next

Figure 1: Introducing MongoDB Lab

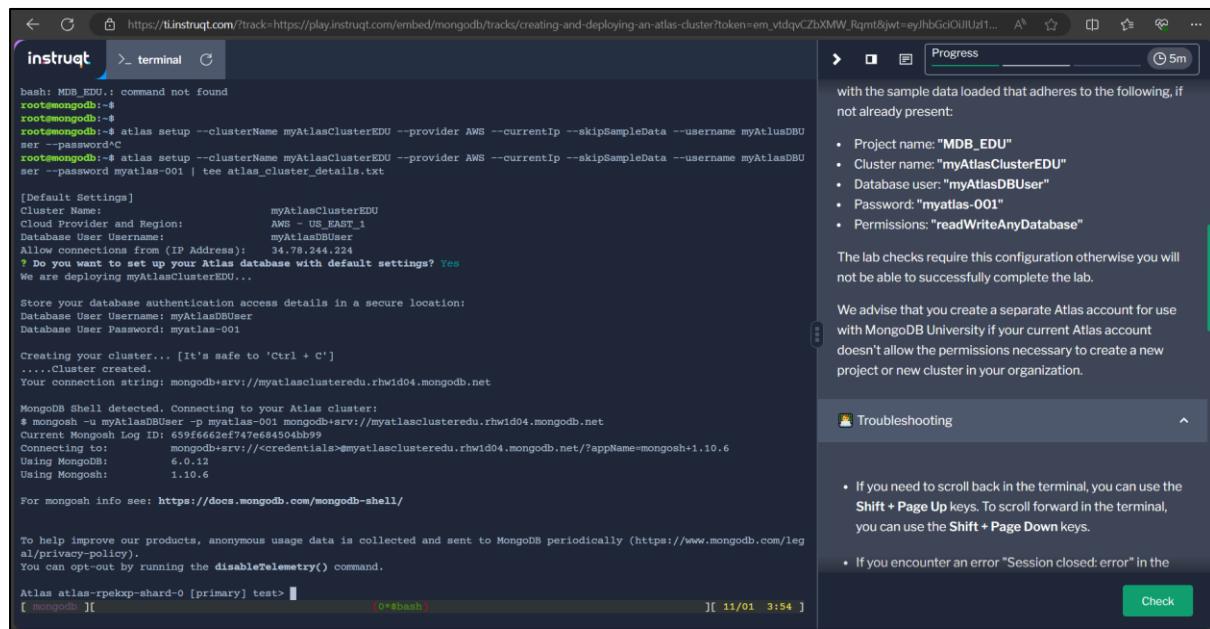
Using the online MongoDB lab and using the 'db.help()' function to view the list of commands available.

## Topic 2: Getting Started with MongoDB Atlas



The screenshot shows a web browser window for MongoDB University. The URL is https://learn.mongodb.com/learn/course/getting-started-with-mongodb-atlas/lesson-1-introduction-to-mongodb-atlas-the-developer-data-platform/practice?client=customer&p... . The main content area displays 'Lesson 1: Introduction to MongoDB Atlas, the Developer Data Platform / Practice'. It shows a 'QUIZ RESULTS' section with the message: 'You got 1 out of 2 correct: 50% You passed! 50% required for passing grade'. Below this, a message says: 'Incredible job learning! You've identified the use cases of MongoDB Atlas and the characteristics of an Atlas cluster!'. At the bottom are three buttons: 'Retake Quiz' (gray), 'Review All Question Results' (white), and 'Continue to Next Section' (green). On the left sidebar, under 'Lessons', there are sections for 'Learn' (selected), 'Practice' (selected), and 'Lab: Introduction to the IDE'. A green box highlights 'Quiz: Introduction to MongoDB Atlas, the Developer Data Platform'. Under 'LESSON 2: CREATING AND DEPLOYING AN ATLAS CLUSTER', it shows 'Progress 50%' and '50% Complete'. Other sidebar sections include 'Resources & Forums', 'Assignments', and 'Notes'.

Figure 2.1: Quiz 1 results in Topic 2



The screenshot shows a terminal interface on Instructruct. The URL is https://instructruct.com/?track=https://play.instructruct.com/embed/mongodb/tracks/creating-and-deploying-an-atlas-cluster?token=em\_vtdqvCzbXMW\_Rqmt&jwt=eyJhbGciOIJUzI... . The terminal window shows a series of commands being run to set up an Atlas cluster. The output includes configuration details like cluster name (myAtlasClusterEDU), provider (AWS), region (US EAST 1), and user credentials (myAtlasDBUser, myAtlasDBUser, password myatlas-001). It also shows the creation of a database and the connection string. To the right of the terminal, there's a 'Progress' bar at 5m, a note about sample data, a list of project requirements (Project name: "MDB\_EDU", Cluster name: "myAtlasClusterEDU", Database user: "myAtlasDBUser", Password: "myatlas-001", Permissions: "readWriteAnyDatabase"), a note about lab checks, and a 'Troubleshooting' section with tips for navigating the terminal and handling errors.

Figure 2.2: Lab on Creating a Cluster

The screenshot shows the MongoDB Atlas Data Explorer interface. On the left, there's a sidebar with sections for Project 0, Data Services, Deployment, Services, and Security. Under Data Services, the 'sample\_analytics' database is selected, and its 'accounts' collection is shown. The main panel displays the 'sample\_analytics.accounts' collection with various tabs like Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A search bar at the top right says 'VISUALIZE YOUR DATA' and 'REFRESH'. Below the search bar, there's a 'Filter' section with the query '{account\_id: 794875}' and buttons for 'Reset', 'Apply', and 'Options'. The 'QUERY RESULTS' section shows 1-1 OF 1 document with the following content:

```
_id: ObjectId('5ca4bbc7a2dd94ee58162392')
account_id: 794875
limit: 9999
products: Array (2)
```

Figure 2.3: Explore sample data with Atlas Data Explorer



Figure 2.4: Topic 2 Proof of Completion

## Topic 3: MongoDB and the Document Model

The screenshot shows the MongoDB Learn interface. On the left, a sidebar displays navigation links like 'Lessons', 'Lab: Introduction to the IDE', 'Quiz: The MongoDB Database' (which is highlighted), and 'Assignments'. The main content area is titled 'Lesson 1: Introduction to MongoDB / Practice'. It shows a 'QUIZ RESULTS' section with the message: 'You got 3 out of 3 correct: 100%', 'You passed!', and '50% required for passing grade'. Below this, a text block states: 'Incredible job learning! The MongoDB database is a general purpose database at the core of MongoDB Atlas. A document is the basic unit of data in MongoDB. Within a database, documents are grouped into collections. A database can have multiple collections.' At the bottom, there are three buttons: 'Retake Quiz', 'Review All Question Results', and a green 'Continue to Next Section' button.

Figure 3.1: Quiz 1 results for Topic 3

The screenshot shows the MongoDB Learn interface. The sidebar is identical to Figure 3.1. The main content area is titled 'Lesson 2: The MongoDB Document Model / Practice'. It shows a 'QUIZ RESULTS' section with the message: 'You got 3 out of 3 correct: 100%', 'You passed!', and '50% required for passing grade'. Below this, a text block states: 'Great job learning! In this activity, you've identified common BSON data types and defined the characteristics of the `_id` field, which is a unique identifier in MongoDB. You also reviewed multiple documents that can exist in the same collection due to MongoDB's flexible schema model.' At the bottom, there are three buttons: 'Retake Quiz', 'Review All Question Results', and a green 'Continue to Next Section' button.

Figure 3.2: Quiz 2 results for Topic 3

The screenshot shows a web browser window for the MongoDB Learn platform at <https://learn.mongodb.com/learn/course/mongodb-document-model/lesson-3-managing-databases-collections-and-documents-in-atlas-data-explore/practice?client=customer...>. The main content area displays 'Lesson 3: Managing Databases, Collections, and Documents in Atlas Data Explorer / Practice'. Below this, under 'QUIZ RESULTS', it says 'You got 2 out of 2 correct: 100%' and 'You passed!'. It also notes '50% required for passing grade'. A message below states: 'Nice job! You've identified how to view databases and collections in the Atlas Data Explorer. You also identified how to add a collection to an existing database.' At the bottom, there are three buttons: 'Retake Quiz' (disabled), 'Review All Question Results' (disabled), and 'Continue to Next Section'.

Figure 3.3: Quiz 3 results for Topic 3

The screenshot shows a web browser window for the MongoDB Cloud interface at [https://cloud.mongodb.com/v2/659f64ca0703de74c49594ac#/metrics/replicaSet/659f6725777c34521988f997/explorer/mdbuser\\_test\\_db/users/find](https://cloud.mongodb.com/v2/659f64ca0703de74c49594ac#/metrics/replicaSet/659f6725777c34521988f997/explorer/mdbuser_test_db/users/find). The left sidebar shows 'Atlas' with sections like 'MDB\_EDU', 'Data Services' (selected), 'App Services', and 'Charts'. Under 'Data Services', 'Database' is selected, showing 'mdbuser\_test\_db' with 'users' as the active collection. The main area is titled 'Insert Document' with the sub-instruction 'To collection users'. A code editor window displays the following JSON document:

```
1 > [ {  
2   "name": "Parker",  
3   "age": 28  
4 } ]
```

Below the code editor are 'VIEW' and 'SEARCH INDEXES' buttons, and a large 'INSERT DOCUMENT' button. At the bottom right of the dialog are 'Cancel' and 'Insert' buttons.

Figure 3.4: Inserting document into a database

# Proof of Completion

Congratulations to

Daniel Obon

For successfully completing

MongoDB and the Document Model

On 01-10-2024



Sahir Azam  
CPO  
MongoDB, Inc



MDBj895djoj89

Figure 3.5: Topic 3 Proof of Completion

## Topic 4: Connecting to a MongoDB Database

The screenshot shows the MongoDB University learning platform interface. The left sidebar has sections for Lessons, Practice, Resources & Forums, Assignments, and Notes. The main content area is titled "Lesson 1: Using MongoDB Connection Strings / Practice". It displays a "QUIZ RESULTS" section with the message: "You got 1 out of 1 correct: 100% You passed! 50% required for passing grade". Below this, a message says "Incredible job learning!". At the bottom are buttons for "Retake Quiz", "Review All Question Results", and a large green "Continue to Next Page". Navigation arrows are visible on both sides of the content area.

Figure 4.1: Quiz 1 results for Topic 4

The screenshot shows the MongoDB University learning platform interface. The left sidebar has sections for Lessons, Practice, Resources & Forums, Assignments, Notes, and Get Support. The main content area is titled "Lesson 2: Connecting to a MongoDB Atlas Cluster with the Shell / Practice". It displays a "QUIZ RESULTS" section with the message: "You got 2 out of 2 correct: 100% You passed! 50% required for passing grade". Below this, a message says "Incredible job learning!". At the bottom are buttons for "Retake Quiz", "Review All Question Results", and a large green "Continue to Next Section". Navigation arrows are visible on both sides of the content area.

Figure 4.2: Quiz 2 results for Topic 4

The screenshot shows the MongoDB University interface. The left sidebar has sections for Lessons, Progress (75% complete), Resources & Forums, Assignments, and Notes. The main content area is titled "Lesson 3: Connecting to a MongoDB Atlas Cluster with Compass / Practice". It displays a "QUIZ RESULTS" section with the message: "You got 2 out of 2 correct: 100% You passed! 50% required for passing grade Incredible job learning!". Below this are buttons for "Retake Quiz", "Review All Question Results", and "Continue to Next Section". Navigation arrows for "Previous" and "Next" are also present.

Figure 4.3: Quiz 3 results for Topic 4

The screenshot shows the MongoDB University interface. The left sidebar has sections for Lessons, Practice (Quiz: Connecting to a MongoDB Atlas Cluster from an Application selected), Progress (100% complete), Resources & Forums, Assignments, and Notes. The main content area is titled "Lesson 4: Connecting to a MongoDB Atlas Cluster from an Application / Practice". It displays a "QUIZ RESULTS" section with the message: "You got 2 out of 2 correct: 100% You passed! 50% required for passing grade Great job learning!". Below this are buttons for "Retake Quiz", "Review All Question Results", and "Continue to Next Section". Navigation arrows for "Previous" and "Next" are also present.

Figure 4.4: Quiz 4 results for Topic 4

The screenshot shows the MongoDB Learn platform interface. On the left, there's a sidebar with 'Search' and sections for 'Lessons' (Learn, Practice), 'Quiz: Troubleshooting MongoDB Atlas Connection Errors', and 'CONCLUSION' (Learn). The main area is titled 'Lesson 5: Troubleshooting MongoDB Atlas Connection Errors / Practice'. It displays 'QUIZ RESULTS' with the message 'You got 2 out of 2 correct: 100% You passed! 50% required for passing grade'. Below this, it says 'Great job learning!' with three buttons: 'Retake Quiz', 'Review All Question Results', and 'Continue to Next Section'. Navigation arrows ('Previous', 'Next') are at the top and bottom of the main content area.

Figure 4.5: Quiz 5 results for Topic 4

The screenshot shows an Instruqt lab session titled 'Connecting to your Atlas Cluster'. The terminal window on the left shows a MongoDB shell session with the following command and output:

```
setVersion: 12,
isWritablePrimary: true,
secondary: false,
primary: 'ac-zxbj0sw-shard-00-01.rhwld04.mongodb.net:27017',
tags: {
  nodeType: 'ELECTABLE',
  availabilityZone: 'use1-az4',
  provider: 'AWS',
  workloadType: 'OPERATIONAL',
  region: 'US_EAST_1'
},
me: 'ac-zxbj0sw-shard-00-01.rhwld04.mongodb.net:27017',
electionId: ObjectId("ffffffff0000000000000000"),
lastWriteFsyncTime: {
  ts: Timestamp({ t: 1704950832, i: 1 }), t: Long("456")
},
lastWriteTime: ISODate("2024-01-11T05:27:12.000Z"),
majorityOpTime: { ts: Timestamp({ t: 1704950832, i: 1 }), t: Long("456") },
majorityWriteTime: ISODate("2024-01-11T05:27:12.000Z"),
maxBsonObjectSize: 16777216,
maxMessageSizeBytes: 48000000,
maxWriteBatchSize: 100000,
localTime: ISODate("2024-01-11T05:27:12.409Z"),
logicalSessionTimeoutMinutes: 30,
connectionId: 96014,
minWireVersion: 0,
maxWireVersion: 17,
readOnly: false,
ok: 1,
clusterTime: {
  clusterTime: Timestamp({ t: 1704950832, i: 1 }),
  signature: {
    hash: Binary(Buffer.from("726380b9d9ddbe69195ff11b0183acbb3ab5e", "hex"), 0),
    keyid: Long("7261351622226214914")
  }
},
operationTime: Timestamp({ t: 1704950832, i: 1 })
}
Atlas atlas-rpekxp-shard-00 [primary] test> [ mongoDB ] [ 0*#bash ]
```

The right-hand panel contains instructions and a code editor. It says: 'In this lab, you will get your connection string and then use it to connect to your Atlas cluster by using `mongosh`. For these steps you will use the `terminal` tab.' It lists three steps:

- In the `terminal` tab, enter the command to get the connection string for your Atlas cluster. The cluster name is `myAtlasCluster001`. This is for your information only, we have already saved this to a bash variable which we will use in the next step. (Forgot the command? Check the hint below!)
- Copy and paste the following commands to get the connection string and store it in an environment variable. Use this variable with `mongosh` to connect to your Atlas cluster.
- Copy and paste the following command to the `mongosh` that's running in the `terminal` tab and run it. The `hello` or `helper db.hello()` command will output a document that describes the role of the `mongod` instance that you have connected to by using the `mongosh`:

```
MY_ATLAS_CONNECTION_STRING=$(atlas clusters c
mongosh -u myAtlasDBUser -p myAtlas-001 #MY_A'
```

Figure 4.6: Lab on Connecting to your Atlas Cluster

Entering the command ‘db.hello()’ will then output a document that describes the role of the instance that you have just connected to.

The screenshot shows the MongoDB Compass interface connected to a cluster named 'Cluster0'. The left sidebar lists databases: admin, local, sample\_airbnb, sample\_analytics, and sample\_geospatial. The main area displays detailed information for each database, including storage size, number of collections, and indexes.

Database	Storage size:	Collections:	Indexes:
admin	0 B	0	0
local	-	-	-
sample_airbnb	54.41 MB	1	4
sample_analytics	9.87 MB	3	3
sample_geospatial	770.05 kB	1	2

Figure 4.7: Download MongoDB Compass and Connect Cluster to it



Figure 4.8: Topic 4 Proof of Completion

## Topic 5: MongoDB CRUD Operations: Insert and Find Documents

The screenshot shows the MongoDB shell (mongosh) and the Instruct lab interface. The shell shows a series of commands being run in the sample\_analytics database. The first few commands attempt to use the `insertOne()` method on the `accounts` collection, which results in `ReferenceError` because the collection does not exist. Subsequent commands show the creation of the `accounts` collection and the execution of the `insertOne()` command successfully, inserting a document with `account_id: 111333`. The Instruct lab interface on the right provides instructions for inserting a single document into the `accounts` collection, including a code template and a "Check" button.

```
al/privacy-policy.
You can opt-out by running the disableTelemetry() command.

Atlas atlas-d3opcw-shard-0 [primary] sample_analytics> insertOne()
ReferenceError: insertOne is not defined
Atlas atlas-d3opcw-shard-0 [primary] sample_analytics> db.sample_analytics.insertOne()
MongoServerError: [COMMON-10001] Missing required argument at position 0 (Collection.insertOne)
Atlas atlas-d3opcw-shard-0 [primary] sample_analytics> db.account.insertOne()
MongoServerError: [COMMON-10001] Missing required argument at position 0 (Collection.insertOne)
Atlas atlas-d3opcw-shard-0 [primary] sample_analytics> db.account.insertOne({
...   "account_id": 111333,
...   "limit": 12000,
...   "products": [
...     "Commodity",
...     "Brokerage"
...   ],
...   "last_updated": new Date()
... })
MongoServerError: not authorized on sample_analytics to execute command { insert: "account", documents: [ { account_id: 111333, limit: 12000, products: [ "Commodity", "Brokerage" ], last_updated: new Date(1704953723019), _id: ObjectId('659f877bd83afaf389f75fa5e') } ], ordered: true, lsid: { id: UUID("49a3e64-99dd-4f47-a21b-1535bd048e92") }, txnNumber: 1, cClusterTime: { clusterTime: Timestamp(1704953581, 6), signature: { hash: BinData(0, C6733D18C2951043CF735BFA9BB64D365C5484D), keyId: 7286845783591092292 } }, $db: "sample_analytics" }
Atlas atlas-d3opcw-shard-0 [primary] sample_analytics> db.accounts.insertOne(
...   {
...     account_id: 111333,
...     limit: 12000,
...     products: [
...       "Commodity",
...       "Brokerage"
...     ],
...     "last_updated": new Date()
...   }
...
{
  acknowledged: true,
  insertedId: ObjectId("659f879583a3fa389f75fa5f")
}
Atlas atlas-d3opcw-shard-0 [primary] sample_analytics> []
[ mongosh ]| 0-* mongodb | 1*#mongosh mongodb+srv://<credentials>@instructtest.3xfv8.mongodb.net/sample_analytics |[ 11/01
```

Figure 5.1: Inserting One Document with `insertOne()`

The screenshot shows the MongoDB shell (mongosh) and the Instruct lab interface. The shell shows a series of commands attempting to use the `insertMany()` method on the `accounts` collection. The first few commands fail with `MongoServerError` because the collection does not exist. Subsequent commands show the creation of the `accounts` collection and the execution of the `insertMany()` command successfully, inserting three documents with `account_id: 111333`, `account_id: 678943`, and `account_id: 321654`. The Instruct lab interface on the right provides instructions for inserting multiple documents into the `accounts` collection, including a code template and a "Check" button.

```
...
MongoServerError: Argument "docs" must be an array of documents
Atlas atlas-d3opcw-shard-0 [primary] sample_analytics> db.accounts.insertMany({
...   "account_id": 111333,
...   "limit": 12000,
...   "products": [
...     "Commodity",
...     "Brokerage"
...   ],
...   "last_updated": new Date()
... },
...
{
  "account_id": 678943,
  "limit": 8000,
  "products": [
    "CurrencyService",
    "Brokerage",
    "InvestmentStock"
  ],
  "last_updated": new Date()
},
{
  "account_id": 321654,
  "limit": 10000,
  "products": [
    "Commodity",
    "CurrencyService"
  ],
  "last_updated": new Date()
})
{
  acknowledged: true,
  insertedIds: [
    "$0": ObjectId("659f886b70e246ed7028550b"),
    "$1": ObjectId("659f886b70e246ed7028550c"),
    "$2": ObjectId("659f886b70e246ed7028550d")
  ]
}
Atlas atlas-d3opcw-shard-0 [primary] sample_analytics>
[ mongosh ]| 0-* mongodb | 1*#mongosh mongodb+srv://<credentials>@instructtest.3xfv8.mongodb.net/sample_analytics |[ 11/01
```

Figure 5.2: Inserting Many Documents with `insertMany()`

Lesson 1: Inserting Documents in a MongoDB Collection / Practice

QUIZ RESULTS

You got 2 out of 2 correct: 100%  
You passed!  
50% required for passing grade

Great work! You've correctly identified that the `insertOne()` method is used to insert a single document and the `insertMany()` method is used to insert multiple documents.

Retake Quiz    Review All Question Results    Continue to Next Section

Figure 5.3: Quiz 1 Results for Topic 5

```
atlas atlas-d3opcw-shard-0 [primary] sample_supplies> db.sales.find({ "_id": ObjectId("5bd761dcae323e45a93ccff4") })
[ { _id: ObjectId("5bd761dcae323e45a93ccff4"),
  saleDate: ISODate("2014-08-18T10:42:13.952Z"),
  items: [
    {
      name: "backpack",
      tags: [ "school", "travel", "kids" ],
      price: Decimal128("187.16"),
      quantity: 2
    },
    {
      name: "printer paper",
      tags: [ "office", "stationary" ],
      price: Decimal128("20.61"),
      quantity: 10
    },
    {
      name: "notepad",
      tags: [ "office", "writing", "school" ],
      price: Decimal128("23.75"),
      quantity: 5
    },
    {
      name: "envelopes",
      tags: [ "stationary", "office", "general" ],
      price: Decimal128("9.44"),
      quantity: 5
    }
  ],
  storeLocation: "San Diego",
  customer: { gender: "F", age: 59, email: "lamecevam.tj", satisfaction: 4 },
  couponUsed: false,
  purchaseMethod: "in store"
} ]
```

Instructions

**Finding a Document Using Equality**

In this lab, you will find a document in the `sales` collection using the `_id`.

You are now connected to an Atlas cluster and to the `sample_supplies` database. Use the `sales` collection in this lab.

Lab Instructions

- Return the document with the `_id` value of `ObjectId("5bd761dcae323e45a93ccff4")`. (Forgot the command? Check the hint below!)

Hints

Once you complete this lab, compare your answer to the correct answer in the Review and Solved Code section, then select Next.

Figure 5.4: Finding a Document Using Equality

The screenshot shows a browser window with two main panes. The left pane is a terminal window titled 'mongosh' displaying a MongoDB query. The right pane is a sidebar titled 'Instructions' for a 'Lab Instructions' section.

```

{
  name: 'notepad',
  tags: [ 'office', 'writing', 'school' ],
  price: Decimal128("7.44"),
  quantity: 4
},
{
  name: 'pens',
  tags: [ 'writing', 'office', 'school', 'stationary' ],
  price: Decimal128("33.36"),
  quantity: 5
},
{
  name: 'envelopes',
  tags: [ 'stationary', 'office', 'general' ],
  price: Decimal128("12.85"),
  quantity: 10
},
{
  name: 'backpack',
  tags: [ 'school', 'travel', 'kids' ],
  price: Decimal128("143.51"),
  quantity: 4
},
{
  name: 'notepad',
  tags: [ 'office', 'writing', 'school' ],
  price: Decimal128("25.05"),
  quantity: 3
}
]
storeLocation: 'London',
customer: { gender: 'F', age: 48, email: 'wir@osibova.aq', satisfaction: 2 },
couponUsed: true,
purchaseMethod: 'In store'
}

```

**Instructions**

**Find Documents that Match a Value in a Specified Array**

Next, you will find all the sales that are located in particular cities in the `sales` collection.

You are now connected to an Atlas cluster and to the `sample_supplies` database. Use the `sales` collection in this lab.

**Lab Instructions**

- Create a query that returns all sales from the London and New York stores. (Forgot the command? Check the hint below)

```
{ storeLocation: { $in: ["London", "New York"] } }
```

Once you complete this lab, compare your answer to the correct answer in the Review and Solved Code section, then select Next.

Next

Figure 5.5: Finding a Document with Matching Values Using `$in`

The screenshot shows a web page titled 'MongoDB CRUD Operations: Insert and Find Documents' under the 'MongoDB University' header. On the left, there's a sidebar with navigation links like 'Lessons', 'Quiz: Finding Documents in a MongoDB Collection', 'Progress 40%', 'Resources & Forums', 'Assignments', 'Notes', and 'Get Support'. The main content area is titled 'Lesson 2: Finding Documents in a MongoDB Collection / Practice'. It displays the results of a quiz:

**QUIZ RESULTS**

You got 2 out of 2 correct: 100%  
You passed!  
50% required for passing grade

Great work! You've correctly identified that the `find()` method is used to find documents. You also identified that the `$in` operator is used to select all documents that have a field value equal to any of the values that are specified in the array.

[Retake Quiz](#) [Review All Question Results](#) [Continue to Next Section](#)

Figure 5.6: Quiz 2 Results for Topic 5

The screenshot shows the MongoDB Compass interface. On the left, the mongo shell window displays a query to find items in the sample\_supplies collection where the price is greater than \$200. On the right, the 'Instructions' panel provides context for the task: finding items priced above \$200.

```

{
  name: 'envelopes',
  tags: [ 'stationary', 'office', 'general' ],
  price: Decimal128("20.84"),
  quantity: 7
},
{
  name: 'pens',
  tags: [ 'writing', 'office', 'school', 'stationary' ],
  price: Decimal128("73.25"),
  quantity: 3
},
{
  name: 'binders',
  tags: [ 'school', 'general', 'organization' ],
  price: Decimal128("25.6"),
  quantity: 9
},
{
  name: 'notepad',
  tags: [ 'office', 'writing', 'school' ],
  price: Decimal128("24.55"),
  quantity: 2
},
{
  name: 'laptop',
  tags: [ 'electronics', 'school', 'office' ],
  price: Decimal128("1094.18"),
  quantity: 4
}
]
storeLocation: 'London',
customer: { gender: 'M', age: 46, email: 'an@tiv.co', satisfaction: 3 },
couponUsed: false,
purchaseMethod: 'In store'
}
]
Type "it" for more
atlas-atlas-d3opcw-shard-0 [primary] sample_supplies>
[ mongoDB ](0*mongosh mongoDB:sh://<credentials>@instructtest.3xfvk.mongodb.net/sample_supplies) ][ 11/01 6:28 ]

```

**Using Comparison Operators: Greater Than**

In this lab, you will find documents that contain an item priced greater than \$200 in the `sales` collection using the relevant comparison operator.

You are now connected to an Atlas cluster and to the `sample_supplies` database. Use the `sales` collection in this lab.

**Lab Instructions**

- Find sales that included an item that costs more than \$200. (Forgot the command? Check the hint below)

```
{ "item.price": { $gt: 200 }}
```

Once you complete the lab, compare your answer to the correct answer in the Review and Solved Code tab, then select Next.

**Hints**

**Next**

Figure 5.7: Using Greater Than Operator \$gt

The screenshot shows the MongoDB Compass interface. On the left, the mongo shell window displays a query to find items in the sample\_supplies collection where the price is less than \$25. On the right, the 'Instructions' panel provides context for the task: finding items priced below \$25.

```

{
  name: 'envelopes',
  tags: [ 'stationary', 'office', 'general' ],
  price: Decimal128("5.46"),
  quantity: 9
},
{
  name: 'notepad',
  tags: [ 'office', 'writing', 'school' ],
  price: Decimal128("20.5"),
  quantity: 1
},
{
  name: 'notepad',
  tags: [ 'office', 'writing', 'school' ],
  price: Decimal128("27.74"),
  quantity: 3
},
{
  name: 'notepad',
  tags: [ 'office', 'writing', 'school' ],
  price: Decimal128("26.03"),
  quantity: 3
},
{
  name: 'pens',
  tags: [ 'writing', 'office', 'school', 'stationary' ],
  price: Decimal128("70.56"),
  quantity: 2
}
],
storeLocation: 'Seattle',
customer: { gender: 'M', age: 39, email: 'defzienonja.sx', satisfaction: 5 },
couponUsed: false,
purchaseMethod: 'Online'
}
]
Type "it" for more
atlas-atlas-d3opcw-shard-0 [primary] sample_supplies>
[ mongoDB ](0*mongosh mongoDB:sh://<credentials>@instructtest.3xfvk.mongodb.net/sample_supplies) ][ 11/01 6:28 ]

```

**Using Comparison Operators: Less Than**

In this lab, you will find the documents contain items priced less than \$25 in the `sales` collection by using the relevant comparison operator.

You are now connected to an Atlas cluster and to the `sample_supplies` database. Use the `sales` collection in this lab.

**Lab Instructions**

- Find all documents that contain an item that is priced less than \$25. (Forgot the command? Check the hint below)

```
{ "item.price": { $lt: 25 }}
```

Once you complete the lab, compare your answer to the correct answer in the Review and Solved Code tab, then select Next.

**Hints**

**Next**

Figure 5.8: Using Less Than Operator \$lt

The screenshot shows the MongoDB shell interface in a browser window. The URL is [https://ti.instruct.com/?track=https://play.instruct.com/embed/mongodb/tracks/03-comparison-operators-new?token=em\\_0STZKzErZS82Grsa&jwt=eyJhbGciOiUzI1NlslnR5cCl6l...](https://ti.instruct.com/?track=https://play.instruct.com/embed/mongodb/tracks/03-comparison-operators-new?token=em_0STZKzErZS82Grsa&jwt=eyJhbGciOiUzI1NlslnR5cCl6l...). The command entered is:

```
db.sample_supplies.find({ "items.quantity": { $gte: 10 } })
```

The results show several documents from the `sample_supplies` collection, each containing an array of items. One document is highlighted, showing items like envelopes, notepad, pens, binder, backpack, and a customer record.

Figure 5.9: Using Greater Than or Equal Operator `$gte`

The screenshot shows the MongoDB shell interface in a browser window. The URL is [https://ti.instruct.com/?track=https://play.instruct.com/embed/mongodb/tracks/03-comparison-operators-new?token=em\\_0STZKzErZS82Grsa&jwt=eyJhbGciOiUzI1NlslnR5cCl6l...](https://ti.instruct.com/?track=https://play.instruct.com/embed/mongodb/tracks/03-comparison-operators-new?token=em_0STZKzErZS82Grsa&jwt=eyJhbGciOiUzI1NlslnR5cCl6l...). The command entered is:

```
db.sales.find({ "customer.age": { $lte: 45 } })
```

The results show documents from the `sales` collection, specifically focusing on the `customer` field which contains age information. One document is highlighted, showing a customer record with an age of 45.

Figure 5.10: Using Less Than or Equal Operator `$lte`

The screenshot shows the MongoDB Learn course interface. On the left, there's a sidebar with a search bar and sections for 'Lessons', 'Assignments', 'Notes', and 'Get Support'. The main area is titled 'Lesson 3: Finding Documents by Using Comparison Operators / Practice'. It displays the quiz results: 'You got 2 out of 2 correct: 100%', 'You passed!', and '50% required for passing grade'. Below this, a message says 'Nicely done! You've demonstrated that you know how to use comparison operators in MongoDB queries.' At the bottom, there are buttons for 'Retake Quiz', 'Review All Question Results', and 'Continue to Next Section'.

Figure 5.11: Quiz 3 Result for Topic 5

The screenshot shows the Instruqt platform interface. On the left, there's a terminal window titled 'mongosh' showing a MongoDB query. The query retrieves documents from a collection where the 'products' array contains the value 'CurrencyService'. The results show several documents with different account IDs and product lists. To the right of the terminal is a 'Progress' bar at 19m. The main area is titled 'Instructions' and contains a section for 'Querying for an Array Value'. It asks to find all 'CurrencyService' accounts within the products field. It also states that the user is connected to an Atlas cluster and the sample\_analytics database, and uses the accounts collection. A 'Lab Instructions' section provides a step: 'Create a query that matches all the documents with a products array that contains the value CurrencyService'. A 'Hints' section is available for assistance. A 'Next' button is located at the bottom right.

Figure 5.12: Querying for an Array Value

The screenshot shows the MongoDB shell (mongosh) running on an Atlas cluster. The command `db.transactions.find().pretty()` is executed, displaying a list of documents from the transactions collection. Each document contains a subdocument named 'transactions' which includes fields like 'date', 'amount', 'transaction\_code', 'symbol', and 'price'. The Instruct platform interface is overlaid on the right, titled 'Instructions' and 'Querying Subdocuments'. It provides a task description, lab instructions, and a 'Next' button.

```

{
  total: '6528.648013190223441171156082',
},
{
  date: ISODate("2016-06-23T00:00:00.000Z"),
  amount: 9672,
  transaction_code: 'buy',
  symbol: 'sap',
  price: '80.302549805269478611080558039247989654541015625',
  total: '776686.2617165663971263711574',
},
{
  date: ISODate("1997-06-03T00:00:00.000Z"),
  amount: 9155,
  transaction_code: 'buy',
  symbol: 'adbe',
  price: '5.18666945667976531098020132048986852169036865234309',
  total: '47403.9587590335142202374309',
},
{
  date: ISODate("2016-10-24T00:00:00.000Z"),
  amount: 7184,
  transaction_code: 'buy',
  symbol: 'znga',
  price: '2.50451961762374552336041233502328395843505859375',
  total: '20866.5886533089078398212022',
},
{
  date: ISODate("2007-06-13T00:00:00.000Z"),
  amount: 7687,
  transaction_code: 'sell',
  symbol: 'ibm',
  price: '86.19128123927652040947577916085720062255859375',
  total: '66258.3788863186123876403144'
}
]

```

Figure 5.13: Querying Subdocuments

The screenshot shows the MongoDB Learn platform. On the left, there's a sidebar with navigation links like 'Lessons', 'MongoDB', 'Quiz: Querying on Array Elements in MongoDB', 'Progress 80%', 'Show Details', 'Resources & Forums', 'Assignments', 'Notes', and 'Get Support'. The main content area is titled 'Lesson 4: Querying on Array Elements in MongoDB / Practice'. It displays the results of a quiz: 'You got 2 out of 2 correct: 100% You passed! 50% required for passing grade'. Below this, a message says 'Excellent! You know how to use \$elemMatch and implicit equality to query arrays in MongoDB.' There are three buttons at the bottom: 'Retake Quiz', 'Review All Question Results', and a large green 'Continue to Next Section' button.

Figure 5.14: Quiz 4 Results for Topic 5

The screenshot shows the MongoDB shell (mongosh) running on an Atlas cluster. The command entered is:

```
items: [
  {
    name: 'notepad',
    tags: [ 'office', 'writing', 'school' ],
    price: Decimal128("7.32"),
    quantity: 4
  },
  {
    name: 'notepad',
    tags: [ 'office', 'writing', 'school' ],
    price: Decimal128("39.08"),
    quantity: 2
  },
  {
    name: 'envelopes',
    tags: [ 'stationery', 'office', 'general' ],
    price: Decimal128("7.4"),
    quantity: 4
  },
  {
    name: 'pens',
    tags: [ 'writing', 'office', 'school', 'stationary' ],
    price: Decimal128("52.79"),
    quantity: 5
  },
  {
    name: 'printer paper',
    tags: [ 'office', 'stationary' ],
    price: Decimal128("13.49"),
    quantity: 6
  }
],
storeLocation: 'Austin',
customer: { gender: 'M', age: 25, email: 'ar@babjug.dk', satisfaction: 5 },
couponUsed: true,
purchaseMethod: 'Online'
```

The results show a list of items from the sample\_supplies collection. On the right, the "Instructions" panel for "Using Logical Operators: And" provides the task: "Find every document in the sales collection that meets the following criteria:" with three bullet points: "Purchased online", "Used a coupon", and "Purchased by a customer 25 years old or younger". A note at the bottom says "(Forgot the command? Check the hint below!)"

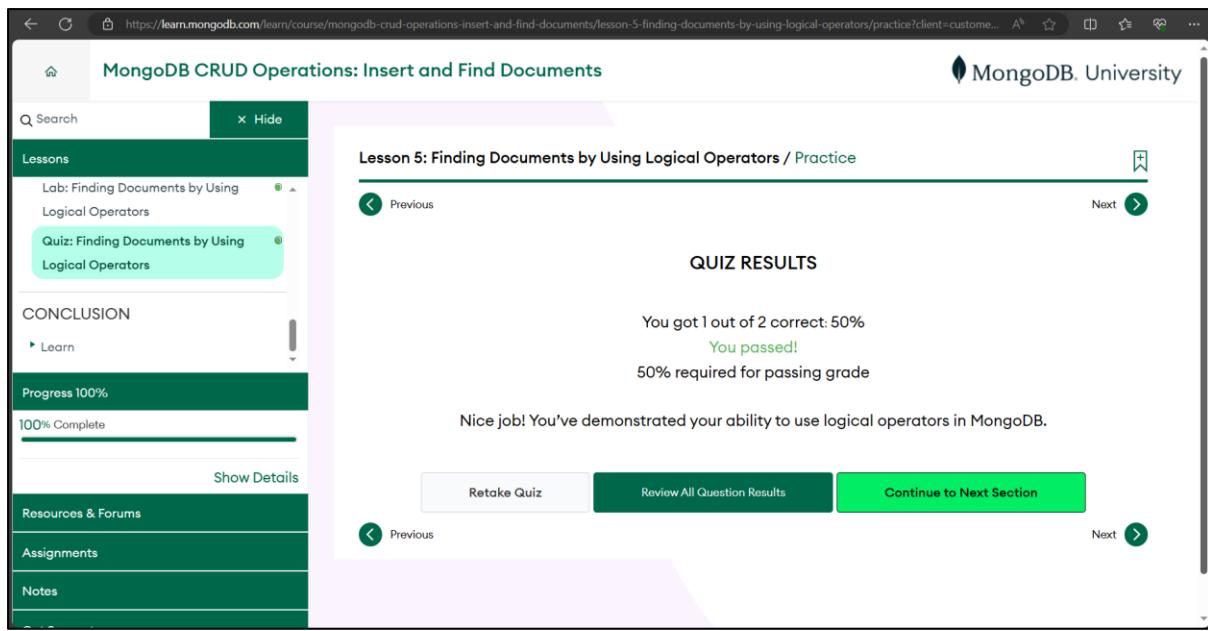
Figure 5.15: Using And Logical Operator \$and

The screenshot shows the MongoDB shell (mongosh) running on an Atlas cluster. The command entered is:

```
items: [
  {
    name: 'backpack',
    tags: [ 'school', 'travel', 'kids' ],
    price: Decimal128("157.33"),
    quantity: 5
  },
  {
    name: 'printer paper',
    tags: [ 'office', 'stationary' ],
    price: Decimal128("31.99"),
    quantity: 7
  },
  {
    name: 'binder',
    tags: [ 'school', 'general', 'organization' ],
    price: Decimal128("28.22"),
    quantity: 4
  },
  {
    name: 'binder',
    tags: [ 'school', 'general', 'organization' ],
    price: Decimal128("24.39"),
    quantity: 6
  },
  {
    name: 'notepad',
    tags: [ 'office', 'writing', 'school' ],
    price: Decimal128("13.44"),
    quantity: 2
  }
],
storeLocation: 'Denver',
customer: { gender: 'F', age: 68, email: 'medju@humdo.mj', satisfaction: 2 },
couponUsed: false,
purchaseMethod: 'Online'
```

The results show a list of items from the sample\_supplies collection. On the right, the "Instructions" panel for "Using Logical Operators: Or" provides the task: "Return every document in the sales collection that meets one of the following criteria:" with two bullet points: "Item with the name of pens" and "Item with a writing tag". A note at the bottom says "(Forgot the command? Check the hint below!)"

Figure 5.16: Using Or Logical Operator \$or



The screenshot shows the MongoDB University quiz results page for 'Lesson 5: Finding Documents by Using Logical Operators / Practice'. The left sidebar includes a search bar, a 'Lessons' section with 'Lab: Finding Documents by Using Logical Operators' and 'Quiz: Finding Documents by Using Logical Operators' (which is highlighted in green), a 'CONCLUSION' section with a 'Learn' button, a progress bar at 100% complete, and 'Resources & Forums', 'Assignments', and 'Notes' sections. The main content area displays the quiz results: 'You got 1 out of 2 correct: 50%', 'You passed!', and '50% required for passing grade'. It also says 'Nice job! You've demonstrated your ability to use logical operators in MongoDB.' Below the results are three buttons: 'Retake Quiz', 'Review All Question Results', and 'Continue to Next Section'. Navigation arrows for 'Previous' and 'Next' are also present.

Figure 5.17: Quiz 5 Results for Topic 5



Figure 5.18: Topic 5 Proof of Completion

## Topic 6: MongoDB CRUD Operations: Replace and Delete Documents

The screenshot shows a MongoDB shell session on Instructr. The terminal window displays the following command:

```
atlas-atlas-d1opcw-shard-0 [primary] bird_data> db.birds.replaceOne({ common_name: "Northern Cardinal" }, { _id: ObjectId("6286809e2fffa87b7d8edcc0") })
```

The response shows the replaced document:

```
{ "_id": "6286809e2fffa87b7d8edcc0", "common_name": "Northern Cardinal", "scientific_name": "Cardinalis cardinalis", "wingspan_cm": 25.32, "habitat": "woodlands", "diet": ["grain", "seeds", "fruit"], "last_seen": ISODate("2022-05-19T20:24:08Z") }
```

Below the terminal, there is a sidebar titled "Replace a Single Document" with instructions and a "Check" button.

Figure 6.1: Replacing a Single Document Using replaceOne()

The screenshot shows the MongoDB Learn platform. On the left, there is a sidebar with navigation links like "Lessons", "Progress", "Resources & Forums", "Assignments", and "Notes". The main area is titled "Lesson 1: Replacing a Document in MongoDB / Practice". It shows the following quiz results:

You got 1 out of 2 correct: 50%  
You passed!  
50% required for passing grade

Great job! You've demonstrated that you know how to use `replaceOne()` to replace documents in MongoDB.

At the bottom, there are buttons for "Retake Quiz", "Review All Question Results", and "Continue to Next Section".

Figure 6.2: Quiz 1 Results for Topic 6

The screenshot shows a browser window with the URL [https://instructt.com/?track=https://play.instructt.com/embed/mongodb/tracks/02-update-operators?token=em\\_EmGMB16-x\\_fDlQV&jwt=eyJhbGciOJUz1NilslnRScI6lkpXVCJ9...](https://instructt.com/?track=https://play.instructt.com/embed/mongodb/tracks/02-update-operators?token=em_EmGMB16-x_fDlQV&jwt=eyJhbGciOJUz1NilslnRScI6lkpXVCJ9...). The title bar says "instructt > mongosh". The main area contains the MongoDB shell code:

```

atlas atlas-xbzsi3-shard-0 [primary] bird_data> db.birds.findOne({common_name:"Canada Goose"})
{
  _id: ObjectId("6268413c613e55b82d7065d2"),
  common_name: 'Canada Goose',
  scientific_name: 'Branta canadensis',
  wingspan_cm: 152.4,
  habitat: 'wetlands',
  diet: [ 'grass', 'algae' ],
  last_seen: ISODate("2022-05-19T20:20:44.083Z")
}
atlas atlas-xbzsi3-shard-0 [primary] bird_data> db.birds.updateOne({
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
})
MongoInvalidArgumentError: Update document requires atomic operators
atlas atlas-xbzsi3-shard-0 [primary] bird_data> db.birds.updateOne(
  {
    common_name: "Canada Goose",
    ...
    {
      $set: {
        tags: [ "geese", "herbivore", "migration" ],
        ...
      }
    }
  }
)
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
atlas atlas-xbzsi3-shard-0 [primary] bird_data> [
  {mongodbs:[{"0":#mongosh mongodb+srv://<credentials>@instructt-test-shared-0.twlipo2.mongodb.net/bird_data"}], [11/01 7:32]}
]

```

The right side of the screen displays a "Progress" bar at 27m and a "Lab Instructions" section:

**Updating a Document Field**

In this lab, you'll add a new field to a document.

You are now connected to an Atlas cluster and the `bird_data` database. Use the `birds` collection for this lab.

**Lab Instructions**

- Run a `findOne()` query for a document with a `common_name` of `Canada Goose` and examine its contents:

```
{
  _id: ObjectId("6268413c613e55b82d7065d2"),
  common_name: 'Canada Goose',
  scientific_name: 'Branta canadensis',
  wingspan_cm: 152.4,
  habitat: 'wetlands',
  diet: [ 'grass', 'algae' ]
},
```

- Update the document by adding a new field titled `tags` and set it to an array containing the following strings: `geese`, `herbivore`, and `migration` using the `updateOne()` command.

**Skip** **Check**

Figure 6.3: Updating a Document Field with `updateOne()`

The screenshot shows a browser window with the URL [https://instructt.com/?track=https://play.instructt.com/embed/mongodb/tracks/02-update-operators?token=em\\_EmGMB16-x\\_fDlQV&jwt=eyJhbGciOJUz1NilslnRScI6lkpXVCJ9...](https://instructt.com/?track=https://play.instructt.com/embed/mongodb/tracks/02-update-operators?token=em_EmGMB16-x_fDlQV&jwt=eyJhbGciOJUz1NilslnRScI6lkpXVCJ9...). The title bar says "instructt > mongosh". The main area contains the MongoDB shell code:

```

Current Mongosh Log ID: 659f999e8845ab86956463ca
Connecting to: mongodb+srv://<credentials>@instructt-test-shared-0.twlipo2.mongodb.net/bird_data?appName=mongosh+1.10
.6
Using MongodB: 6.0.12
Using Mongosh: 1.10.6

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

atlas atlas-xbzsi3-shard-0 [primary] bird_data> db.birds.updateOne(
  ...
  {
    ...
    $push: {
      ...
      diet: { $each: ["newts", "opossum", "skunks", "squirrels"] },
      ...
    }
  }
)
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
atlas atlas-xbzsi3-shard-0 [primary] bird_data> [
  {mongodbs:[{"0":#mongosh mongodb+srv://<credentials>@instructt-test-shared-0.twlipo2.mongodb.net/bird_data"}], [11/01 7:33]}
]

```

The right side of the screen displays a "Progress" bar at 26m and a "Lab Instructions" section:

**Adding Values to an Array**

In this activity, we will add an element to an array field within a document.

You are now connected to an Atlas cluster and the `bird_data` database. Use the `birds` collection for this lab.

**Lab Instructions**

- Write a query that will update a document with an `_id` of `objectId("6268471e613e55b82d7065d7")` and add the following to the `diet` array without removing any existing values:

```
diet: ["newts", "opossum", "skunks", "squirrels"]
```

A successful update will return the following:

```
{
  ...
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
}
```

**Skip** **Check**

Figure 6.4: Updating a Document Field by Adding Values to an Array

The screenshot shows the MongoDB Mongosh shell running on an Atlas cluster. A query is being executed to update a document in the `birds` collection. The command uses `updateOne()` with filters for `common_name` and `$inc` operator to increment `sightings`. The output shows the updated document with an additional `last_updated` field.

**Return, Update, and Add a Document**

In this activity, we will perform an operation that can return, edit, and optionally add a new document to a collection in one step.

You are now connected to an Atlas cluster and the `bird_data` database. Use the `birds` collection for this lab.

**Lab Instructions**

1. Write a query to update a document in the `birds` collection.
  - In the filter document, use the `common_name` field with a value of `Robin Redbreast` to search for the document.
  - In the update document, use the `$inc` operator to increment the `sightings` field by 1. Additionally, your query should set a new field called `last_updated` to the current date and time, using `new Date()`.
  - Add an option to the `updateOne()` method to create a new document if no documents match your query.

**Note**

If you get an unexpected result of `upsertedCount: 0`, where

Skip Check

Figure 6.5: Return, Update and Add a Document

The screenshot shows the MongoDB Learning Platform interface. On the left, a sidebar displays navigation links for lessons, resources, assignments, and notes. The main content area is titled "Lesson 2: Updating MongoDB Documents by Using updateOne() / Practice". It shows a quiz results summary: 2 out of 2 correct (100%), passed, and 50% required for passing grade. A message congratulates the user on using the `$set` and `$push` operators correctly. Navigation buttons include "Retake Quiz", "Review All Question Results", and "Continue to Next Section".

Figure 6.6: Quiz 2 Results for Topic 6

The screenshot shows a browser window with the URL [https://instruct.com/?track=https://play.instruct.com/embed/mongodb/tracks/03-findandmodify?token=em\\_2PB7pc0UnrEtGJaf&jwt=eyJhbGciOiUzI1NlslnR5cCI6IkpXVCJ9eyJj...](https://instruct.com/?track=https://play.instruct.com/embed/mongodb/tracks/03-findandmodify?token=em_2PB7pc0UnrEtGJaf&jwt=eyJhbGciOiUzI1NlslnR5cCI6IkpXVCJ9eyJj...). The left pane displays the MongoDB shell with the following command and its output:

```

Current Mongosh Log ID: 659f9a596fe77c775e3c59de
Connecting to: mongodb://<credentials>@instructtest.3xfvk.mongodb.net/bird_data?appName=mongosh+1.10.6
Using MongoDB: 6.0.12
Using Mongosh: 1.10.6

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

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

Atlas atlas-d3opcw-shard-0 [primary] bird_data> db.birds.findAndModify({
  ... query: { common_name: "Blue Jay" },
  ... update: { $inc: { sightings_count: 1 } },
  ... new: true,
  ...
})
{
  _id: ObjectId("628682d92f3fa87b7d86dcce"),
  common_name: 'Blue Jay',
  scientific_name: 'Cyanocitta cristata',
  wingspan_cm: 34.17,
  habitat: 'forests',
  diet: ['vegetables', 'nuts', 'grains'],
  sightings_count: 5,
  last_seen: ISODate("2022-05-19T20:20:44.083Z")
}
Atlas atlas-d3opcw-shard-0 [primary] bird_data>

```

The right pane contains instructions for the 'Find and Modify' activity, including a list of tasks and a sample document to verify the update.

Figure 6.7: Find and Modify Data using `findAndModify()`

The screenshot shows a browser window with the URL <https://learn.mongodb.com/learn/course/mongodb-crud-operations-replace-and-delete/documents/lesson-3-updating-mongodb-documents-by-using-findandmodify/practice?d...>. The left sidebar shows a navigation menu with 'Lessons' selected, leading to 'Lab: Updating MongoDB Documents by Using findAndModify()' and 'Quiz: Updating MongoDB Documents by Using findAndModify()'. The main content area displays the results of a quiz titled 'Lesson 3: Updating MongoDB Documents by Using `findAndModify()` / Practice'.

**QUIZ RESULTS**

You got 2 out of 2 correct: 100%  
You passed!  
50% required for passing grade

Excellent work! You've identified that `findAndModify()` is used to update and return an existing document. When the new option is set to true, `findAndModify()` returns the updated version of the document. When the upsert option is set to true, `findAndModify()` inserts a new document if one does not already exist.

Buttons at the bottom include 'Retake Quiz', 'Review All Question Results', and 'Continue to Next Section'.

Figure 6.8: Quiz 3 Results for Topic 6

The screenshot shows a terminal window titled 'mongosh' running on an Atlas cluster. The command executed was:

```
db.birds.updateMany({
  common_name: {
    $in: ["Blue Jay", "Grackle"]
  },
  {
    $set: {
      last_seen: ISODate("2022-01-01")
    }
  }
})
```

The output shows the operation was acknowledged with true, and it updated 2 documents, resulting in 2 modified and 0 upserted documents.

To the right of the terminal, there is a 'Progress' bar at 10m and a sidebar titled 'Updating Multiple Documents' with instructions and lab instructions.

Figure 6.9: Updating Multiple Documents using updateMany()

The screenshot shows a quiz result page from the MongoDB Learn platform. The quiz title is 'Lesson 4: Updating MongoDB Documents by Using updateMany() / Practice'. The results section shows:

- You got 1 out of 1 correct: 100%
- You passed!
- 50% required for passing grade

A message states: 'Great work! You've identified that `db.collection.updateMany()` is used to update multiple documents.'

At the bottom, there are buttons for 'Retake Quiz', 'Review All Question Results', and 'Continue to Next Page'.

Figure 6.10: Quiz 4 Results for Topic 6

The screenshot shows a browser-based MongoDB shell interface. On the left, the terminal window displays a command to delete a document from the `birds` collection:

```
mongodbsrv://<credentials>@instruct-test-shared-0.twlio2.mongodb.net/bird_data?appName=mongosh+1.10
db.birds.deleteOne({ _id: ObjectId("62cddf53c1d62bc45439beb") })

```

On the right, the UI provides instructions for the task:

- Delete a Single Document**
- In this activity we will delete a document from the `birds` collection.
- Before you begin, please note that you are now connected to an Atlas cluster and the `bird_data` database. Use the `birds` collection for this lab.
- Lab Instructions**

  - Delete the document in the `birds` collection with the following `_id`:

```
ObjectId("62cddf53c1d62bc45439beb")
```

- The output from your command will look like the following:

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

- Note**
- If you get an unexpected result of `deletedCount: 0`, where the output of the command is as follows:

Figure 6.11: Deleting a Single Document using `deleteOne()`

The screenshot shows a browser-based MongoDB shell interface. On the left, the terminal window displays a command to delete multiple documents from the `birds` collection:

```
mongodbsrv://<credentials>@instruct-test-shared-0.twlio2.mongodb.net/bird_data?appName=mongosh+1.10
db.birds.deleteMany({ sightings_count: { $lte: 10 } })

```

On the right, the UI provides instructions for the task:

- Delete Multiple Documents**
- In this activity we will delete multiple documents from the `birds` collection.
- Before you begin, please note that you are now connected to an Atlas cluster and the `bird_data` database. Use the `birds` collection for this lab.
- Lab Instructions**

  - Delete all documents in the `birds` collection that have a `sightings_count` of less than or equal to `10`.

- The output from your command will look like the following:

```
{ acknowledged: true, deletedCount: 3 }
```

- Note**
- If you get an unexpected result of `deletedCount: 0`, where the output of the command is as follows:

Figure 6.12: Deleting a Multiple Document using `deleteMany()`

The screenshot shows the MongoDB Learn platform interface. On the left, there's a sidebar with a search bar, a 'Lessons' section containing 'Practice' (selected), 'Lab: Deleting Documents in MongoDB' (green dot), and 'Quiz: Deleting Documents in MongoDB' (green dot); a 'CONCLUSION' section with a 'Learn' button; a progress bar at 100% complete; and sections for 'Resources & Forums', 'Assignments', and 'Notes'. The main content area is titled 'Lesson 5: Deleting Documents in MongoDB / Practice'. It displays 'QUIZ RESULTS' with the message: 'You got 2 out of 2 correct: 100%', 'You passed!', and '50% required for passing grade'. Below this, a note says: 'Nice job! You've identified that `deleteOne()` is used to delete a single document, and that `deleteMany()` deletes multiple documents.' At the bottom are buttons for 'Retake Quiz', 'Review All Question Results', and 'Continue to Next Section'.

Figure 6.13: Quiz 5 Results for Topic 6



Figure 6.14: Topic 6 Proof of Completion

## Topic 7: MongoDB CRUD Operations: Modifying Query Results

The screenshot shows the MongoDB shell (mongosh) running on an Atlas cluster. The command `db.sales.find().sort({date: 1})` is executed, resulting in the following document:

```
{ "_id": "5f371f3a0000000000000001", "date": "2020-01-01T00:00:00.000Z", "customer": { "name": "John Doe", "email": "john.doe@example.com", "address": "123 Main St", "city": "Anytown", "state": "CA", "zip": "90210", "phone": "(555) 123-4567", "gender": "M", "age": 30, "satisfaction": 4.5 }, "product": { "name": "Binder", "tags": ["school", "general", "organization"], "price": Decimal128("17.22"), "quantity": 1 }, "purchaseMethod": "In-store", "storeLocation": "Anytown Superstore", "couponsUsed": false, "orderStatus": "Shipped", "orderNotes": "Please include a note for delivery." }, { "_id": "5f371f3a0000000000000002", "date": "2020-01-02T00:00:00.000Z", "customer": { "name": "Jane Doe", "email": "jane.doe@example.com", "address": "456 Elm St", "city": "Anytown", "state": "CA", "zip": "90210", "phone": "(555) 123-4567", "gender": "F", "age": 25, "satisfaction": 4.8 }, "product": { "name": "Envelopes", "tags": ["stationery", "office", "general"], "price": Decimal128("23.33"), "quantity": 3 }, "purchaseMethod": "Online", "storeLocation": "Anytown Superstore", "couponsUsed": true, "orderStatus": "Shipped", "orderNotes": "Please include a note for delivery." }, { "_id": "5f371f3a0000000000000003", "date": "2020-01-03T00:00:00.000Z", "customer": { "name": "Jane Doe", "email": "jane.doe@example.com", "address": "456 Elm St", "city": "Anytown", "state": "CA", "zip": "90210", "phone": "(555) 123-4567", "gender": "F", "age": 25, "satisfaction": 4.8 }, "product": { "name": "Envelopes", "tags": ["stationery", "office", "general"], "price": Decimal128("22.06"), "quantity": 7 }, "purchaseMethod": "Online", "storeLocation": "Anytown Superstore", "couponsUsed": true, "orderStatus": "Shipped", "orderNotes": "Please include a note for delivery." }, { "_id": "5f371f3a0000000000000004", "date": "2020-01-04T00:00:00.000Z", "customer": { "name": "John Doe", "email": "john.doe@example.com", "address": "123 Main St", "city": "Anytown", "state": "CA", "zip": "90210", "phone": "(555) 123-4567", "gender": "M", "age": 30, "satisfaction": 4.5 }, "product": { "name": "Notebook", "tags": ["office", "writing", "school"], "price": Decimal128("8.06"), "quantity": 1 }, "purchaseMethod": "Online", "storeLocation": "Anytown Superstore", "couponsUsed": false, "orderStatus": "Shipped", "orderNotes": "Please include a note for delivery." }
```

The right panel displays the challenge instructions for "Return Query Results in Ascending Order". It states: "In this activity, you will return query results in ascending order." and "Lab Instructions: You are now connected to an Atlas cluster and the sample\_supplies database. You will use the sales collection. (Suggestion: run db.sales.findOne() to return a sample document and review the structure of the data in this collection.)". It also includes hints and a "Review and Solved Code" section.

Figure 7.1: Return Query in Ascending Order using sort() where sort() is 1

The screenshot shows the MongoDB shell (mongosh) running on an Atlas cluster. The command `db.sales.find().sort({date: -1})` is executed, resulting in the following document:

```
{ "_id": "5f371f3a0000000000000004", "date": "2020-01-04T00:00:00.000Z", "customer": { "name": "John Doe", "email": "john.doe@example.com", "address": "123 Main St", "city": "Anytown", "state": "CA", "zip": "90210", "phone": "(555) 123-4567", "gender": "M", "age": 30, "satisfaction": 4.5 }, "product": { "name": "Notebook", "tags": ["office", "writing", "school"], "price": Decimal128("8.06"), "quantity": 1 }, "purchaseMethod": "Online", "storeLocation": "Anytown Superstore", "couponsUsed": false, "orderStatus": "Shipped", "orderNotes": "Please include a note for delivery." }, { "_id": "5f371f3a0000000000000003", "date": "2020-01-03T00:00:00.000Z", "customer": { "name": "Jane Doe", "email": "jane.doe@example.com", "address": "456 Elm St", "city": "Anytown", "state": "CA", "zip": "90210", "phone": "(555) 123-4567", "gender": "F", "age": 25, "satisfaction": 4.8 }, "product": { "name": "Envelopes", "tags": ["stationery", "office", "general"], "price": Decimal128("22.06"), "quantity": 7 }, "purchaseMethod": "Online", "storeLocation": "Anytown Superstore", "couponsUsed": true, "orderStatus": "Shipped", "orderNotes": "Please include a note for delivery." }, { "_id": "5f371f3a0000000000000002", "date": "2020-01-02T00:00:00.000Z", "customer": { "name": "Jane Doe", "email": "jane.doe@example.com", "address": "456 Elm St", "city": "Anytown", "state": "CA", "zip": "90210", "phone": "(555) 123-4567", "gender": "F", "age": 25, "satisfaction": 4.8 }, "product": { "name": "Envelopes", "tags": ["stationery", "office", "general"], "price": Decimal128("23.33"), "quantity": 3 }, "purchaseMethod": "Online", "storeLocation": "Anytown Superstore", "couponsUsed": true, "orderStatus": "Shipped", "orderNotes": "Please include a note for delivery." }, { "_id": "5f371f3a0000000000000001", "date": "2020-01-01T00:00:00.000Z", "customer": { "name": "John Doe", "email": "john.doe@example.com", "address": "123 Main St", "city": "Anytown", "state": "CA", "zip": "90210", "phone": "(555) 123-4567", "gender": "M", "age": 30, "satisfaction": 4.5 }, "product": { "name": "Binder", "tags": ["school", "general", "organization"], "price": Decimal128("17.22"), "quantity": 1 }, "purchaseMethod": "In-store", "storeLocation": "Anytown Superstore", "couponsUsed": false, "orderStatus": "Shipped", "orderNotes": "Please include a note for delivery." }
```

The right panel displays the challenge instructions for "Return Query Results in Descending Order". It states: "In this activity, you will return query results in descending order." and "Lab Instructions: You are now connected to an Atlas cluster and the sample\_supplies database. You will use the sales collection. (Suggestion: run db.sales.findOne() to return a sample document and review the structure of the data in this collection.)". It also includes hints and a "Review and Solved Code" section.

Figure 7.2: Return Query in Descending Order using sort() where sort() is -1

The screenshot shows a browser window with the URL [https://instruct.com/track=https://play.instruct.com/embed/mongodb/tracks/01-organizing-query-results?token=em\\_1ZGKFWLiw5rhlze&jwt=eyJhbGciOiJIUzI1NiJnR5cC16lk...](https://instruct.com/track=https://play.instruct.com/embed/mongodb/tracks/01-organizing-query-results?token=em_1ZGKFWLiw5rhlze&jwt=eyJhbGciOiJIUzI1NiJnR5cC16lk...). The main content area displays a JSON array of items from the `sample_supplies` collection. The items include various school supplies like printer paper, pens, backpacks, and notebooks, each with a name, price, quantity, tags, and a customer object containing gender, age, email, and satisfaction. To the right of the code, there is a challenge card titled "Return a Limited Number of Sorted Results". It instructs the user to return the three most recent sales from the London store for a laptop, backpack, or printer paper. Below the challenge are sections for "Lab Instructions", "Hints", and "Review and Solved Code". A progress bar at the top indicates 26% completion.

```

        {
            quantity: 4
        },
        {
            name: 'printer paper',
            tags: [ 'office', 'stationary' ],
            price: Decimal128("12.3"),
            quantity: 3
        },
        {
            name: 'binder',
            tags: [ 'school', 'general', 'organization' ],
            price: Decimal128("29.85"),
            quantity: 2
        },
        {
            name: 'pens',
            tags: [ 'writing', 'office', 'school', 'stationary' ],
            price: Decimal128("29.56"),
            quantity: 3
        },
        {
            name: 'backpack',
            tags: [ 'school', 'travel', 'kids' ],
            price: Decimal128("68.59"),
            quantity: 5
        }
    ],
    storeLocation: 'London',
    customer: {
        gender: 'M',
        age: 12,
        email: 'egeorgeidupke.tj',
        satisfaction: 5
    },
    couponUsed: false,
    purchaseMethod: 'Online'
}
]
Atlas atlas-d3opcw-shard-0 [primary] sample_supplies>
[ mongoDB ](0*#mongosh mongoDB:srv://<credentials>@instruct-test.0xfvkk.mongodb.net/sample_supplies) [ 11/01 8:14 ]

```

Figure 7.3: Return a Limited Number of Sorted Results using `top()`

The screenshot shows a browser window with the URL <https://learn.mongodb.com/learn/course/mongodb-crud-operations-modifying-query-results/lesson-1-sorting-and-limiting-query-results-in-mongodb/practice?client=customer&...>. The page is titled "MongoDB CRUD Operations: Modifying Query Results" and "Lesson 1: Sorting and Limiting Query Results in MongoDB / Practice". On the left, there is a sidebar with "Lessons", "Progress 33%", "Resources & Forums", "Assignments", and "Notes". The main content area shows the results of a quiz: "You got 2 out of 2 correct: 100% You passed! 50% required for passing grade". It also includes a message: "Great work! You know how to use `cursor.sort()` and `cursor.limit()` to modify the results returned from your query." At the bottom, there are buttons for "Retake Quiz", "Review All Question Results", and "Continue to Next Section".

Figure 7.4: Quiz 1 Results for Topic 7

The screenshot shows the MongoDB shell (mongosh) running on an Atlas cluster. The command entered is:

```
_id: ObjectId("5bd761dcae323e45a93cd0b?"),
saleDate: ISODate("2013-07-26T18:06:11.111Z"),
storeLocation: "Denver",
purchaseMethod: "Online"
},
{
_id: ObjectId("5bd761dcae323e45a93cd0cd"),
saleDate: ISODate("2015-12-10T02:10:31.924Z"),
storeLocation: "Denver",
purchaseMethod: "Online"
},
{
_id: ObjectId("5bd761dcae323e45a93cd0d5"),
saleDate: ISODate("2013-09-04T17:35:29.831Z"),
storeLocation: "Denver",
purchaseMethod: "Online"
},
{
_id: ObjectId("5bd761dcae323e45a93cd0d7"),
saleDate: ISODate("2015-05-14T19:23:19.887Z"),
storeLocation: "Denver",
purchaseMethod: "In store"
},
{
_id: ObjectId("5bd761dcae323e45a93cd0da"),
saleDate: ISODate("2017-04-14T18:53:34.341Z"),
storeLocation: "Denver",
purchaseMethod: "Online"
},
{
_id: ObjectId("5bd761dcae323e45a93ccfca"),
saleDate: ISODate("2017-06-22T09:54:14.185Z"),
storeLocation: "Denver",
purchaseMethod: "In store"
}
]
```

Type "it" for more

Atlas atlas-d3opcw-shard-0 [primary] sample\_supplies> [ mongodbs ] [ 0\*#mongosh mongoDB+srv://<credentials>@instructortest.3xfvk.mongodb.net/sample\_supplies ] [ 11/01 8:17 ]

The right panel displays the activity details:

### Return Selected Fields, Including the \_id Field

In this activity, you will write a query that returns only selected fields from matching documents, including the `_id` field.

#### Lab Instructions

You are now connected to an Atlas cluster and the `sample_supplies` database.

You will use the `sales` collection for this activity. (Suggestion: run `db.sales.find()` to return a sample document and review the structure of the data in this collection.)

- Query for all sales at the Denver store. Return only the sale date, store location, purchase method and `_id` fields. (Forgot the command? Check the hint below!)

Once you complete the challenge, compare your answer to the correct answer in the Review & Solved Code section, then select 'Next'.

Hints

Review and Solved Code

Next

Figure 7.5: Return Selected Fields, Including the `_id` Field using `{<field> : 1}`

The screenshot shows the MongoDB shell (mongosh) running on an Atlas cluster. The command entered is:

```
_id: ObjectId("5bd761dcae323e45a93cd0b?"),
saleDate: ISODate("2013-07-25T08:45:36.267Z"),
storeLocation: "Austin",
customer: { age: 29, satisfaction: 5 }
},
{
saleDate: ISODate("2017-07-13T08:36:21.692Z"),
storeLocation: "San Diego",
customer: { age: 29, satisfaction: 5 }
},
{
saleDate: ISODate("2017-12-16T05:25:53.099Z"),
storeLocation: "London",
customer: { age: 29, satisfaction: 5 }
},
{
saleDate: ISODate("2014-08-24T09:56:23.437Z"),
storeLocation: "San Diego",
customer: { age: 29, satisfaction: 5 }
},
{
saleDate: ISODate("2013-10-15T18:38:25.706Z"),
storeLocation: "Austin",
customer: { age: 29, satisfaction: 5 }
},
{
saleDate: ISODate("2015-11-27T20:46:04.462Z"),
storeLocation: "Denver",
customer: { age: 29, satisfaction: 5 }
},
{
saleDate: ISODate("2015-06-04T17:53:18.246Z"),
storeLocation: "Seattle",
customer: { age: 29, satisfaction: 5 }
}
]
```

Type "it" for more

Atlas atlas-d3opcw-shard-0 [primary] sample\_supplies> [ mongodbs ] [ 0\*#mongosh mongoDB+srv://<credentials>@instructortest.3xfvk.mongodb.net/sample\_supplies ] [ 11/01 8:18 ]

The right panel displays the activity details:

### Return Selected Fields, Excluding the `_id` Field

In this activity, you will write a query that returns only selected fields from matching documents, excluding the `_id` field.

#### Lab Instructions

You are now connected to an Atlas cluster and the `sample_supplies` database.

You will use the `sales` collection for this activity.

- Find the data on sales to customers less than 30 years old in which the customer satisfaction rating was greater than three. Return only the customer's age and satisfaction rating, the sale date and store location. Do not include the `_id` field. (Forgot the command? Check the hint below!)

Once you complete the challenge, compare your answer to the correct answer in the Review & Solved Code section, then select 'Next'.

Hints

Review and Solved Code

Next

Figure 7.6: Return Selected Fields, Excluding the `_id` Field using `{<field> : 0}`

The screenshot shows a MongoDB playground interface. On the left, the terminal window displays a query in MongoDB shell:

```
_id: ObjectId("5bd761dcae323e45a93cd08e"),
saleDate: ISODate("2017-09-02T15:42:22.167Z"),
items: [
  {
    name: 'backpack',
    tags: [ 'school', 'travel', 'kids' ],
    price: Decimal128("132.21"),
    quantity: 3
  },
  {
    name: 'binder',
    tags: [ 'school', 'general', 'organization' ],
    price: Decimal128("25.87"),
    quantity: 7
  },
  {
    name: 'pens',
    tags: [ 'writing', 'office', 'school', 'stationary' ],
    price: Decimal128("74.79"),
    quantity: 5
  },
  {
    name: 'pens',
    tags: [ 'writing', 'office' ],
    price: Decimal128("54.44"),
    quantity: 4
  },
  {
    name: 'envelopes',
    tags: [ 'stationary', 'office', 'general' ],
    price: Decimal128("8.16"),
    quantity: 3
  }
],
storeLocation: 'New York'
]
```

The right side of the interface contains a challenge titled "Return All Fields Except Those Explicitly Excluded". It includes lab instructions, hints, and a "Review and Solved Code" section.

Figure 7.7: Return Selected Fields Except Those Explicitly Excluded

The screenshot shows a MongoDB learning platform interface. The left sidebar displays a navigation menu with sections like "Lessons", "Progress", "Resources & Forums", "Assignments", "Notes", and "Get Support". The main content area is titled "Lesson 2: Returning Specific Data from a Query in MongoDB / Practice". It shows quiz results:

**QUIZ RESULTS**

You got 1 out of 2 correct: 50%  
You passed!  
50% required for passing grade

Great work! You've identified how to use the projection document to include or exclude specific fields in query results.

Buttons at the bottom include "Retake Quiz", "Review All Question Results", and a large green "Continue to Next Section".

Figure 7.8: Quiz 2 Results for Topic 7

Current Mongosh Log ID: 659fa52c232b474575cf04c3  
 Connecting to: mongodbsrv://<credentials>@instructtest.3xfvk.mongodb.net/sample\_supplies?appName=mongosh+1.10.6  
 Using MongoDB: 6.0.12  
 Using Mongosh: 1.10.6

```
For mongosh info see: https://docs.mongodb.com/mongosh-shell/  
  
To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy). You can opt-out by running the disableTelemetry() command.  
  
Atlas atlas-d3opcw-shard-0 [primary] sample_supplies> db.sales.countDocuments({});  
5000  
Atlas atlas-d3opcw-shard-0 [primary] sample_supplies>
```

Instructions  
**Count All Documents in a Collection**  
 In this activity, you will count all documents in a collection.  
 Lab Instructions  
 You are now connected to an Atlas cluster and the `sample_supplies` database.  
 You will use the `sales` collection for this activity.  
 (Suggestion: run `db.sales.findOne()` to return a sample document and review the structure of the data in this collection.)  
 1. Find the total number of documents in the `sales` collection.(Forgot the command? Check the hint below!)

Once you complete the challenge, compare your answer to the correct answer in the Review & Solved Code section, then select 'Next.'

Hints  
 Review and Solved Code

Figure 7.9: Count All Documents in a Collection using `countDocument({})`

Current Mongosh Log ID: 659fa54d166fdd97370dfc15  
 Connecting to: mongodbsrv://<credentials>@instructtest.3xfvk.mongodb.net/sample\_supplies?appName=mongosh+1.10.6  
 Using Mongosh: 1.10.6

```
For mongosh info see: https://docs.mongodb.com/mongosh-shell/  
  
Atlas atlas-d3opcw-shard-0 [primary] sample_supplies> db.sales.countDocuments({ storeLocation: "Denver", couponUsed: true })  
157  
Atlas atlas-d3opcw-shard-0 [primary] sample_supplies>
```

Instructions  
**Count All Documents That Match a Query (1)**  
 In this activity, you will count all documents that match a query made with an equality constraint.  
 Lab Instructions  
 You are now connected to an Atlas cluster and the `sample_supplies` database.  
 You will use the `sales` collection for this activity.  
 1. Find the number of sales made using a coupon at the Denver location.(Forgot the command? Check the hint below!)

Once you complete the challenge, compare your answer to the correct answer in the Review & Solved Code section, then select 'Next.'

Hints  
 Review and Solved Code

Figure 7.10: Count All Documents That Match a Query 1

Current Mongosh Log ID: 659fa5768ad8b80910f7de69  
Connecting to: mongodbsrv://<credentials>@instructortest.3xfvk.mongodb.net/sample\_supplies?appName=mongosh+1.10.6  
Using MongoDB: 6.0.12  
Using Mongosh: 1.10.6

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

```
Atlas atlas-d3opcw-shard-0 [primary] sample_supplies> db.sales.countDocuments({ items: { $elemMatch: { name: "laptop", price: { $lt: 600 } } } })
397
Atlas atlas-d3opcw-shard-0 [primary] sample_supplies>
```

In this activity, you will count all documents that match a query made with an array operator and a comparison operator.

**Lab Instructions**

You are now connected to an Atlas cluster and the `sample_supplies` database.

You will use the `sales` collection for this activity.

- Find the number of sales that included a laptop that cost less than \$600. (Forgot the command? Check the hint below)

Once you complete the challenge, compare your answer to the correct answer in the Review & Solved Code section, then select 'Next'.

**Hints**

**Review and Solved Code**

**Troubleshooting**

Next

Figure 7.11: Count All Documents That Match a Query 2

The screenshot shows a browser window for the MongoDB Learn platform. The URL is <https://learn.mongodb.com/learn/course/mongodb-crud-operations-modifying-query-results/lesson-3-counting-documents-in-a-mongodb-collection/practice?client=customer&...>. The main content area displays "Lesson 3: Counting Documents in a MongoDB Collection / Practice". On the left, a sidebar lists "Lessons" (Lab: Counting Documents in a MongoDB Collection, Quiz: Counting Documents in a MongoDB Collection), "CONCLUSION" (Learn, Progress 100%, 100% Complete), "Resources & Forums", "Assignments", "Notes", "Get Support", and "FAQ'S" (with a "SIGN OUT" link). The right side shows "QUIZ RESULTS" with the message "You got 2 out of 2 correct: 100%", "You passed!", and "50% required for passing grade". Below this, a message says "Great work! You've identified how to return the total number of documents in a MongoDB collection by using the `countDocuments()` collection method." At the bottom are buttons for "Retake Quiz", "Review All Question Results", and a large green "Continue to Next Section". Navigation arrows ("Previous", "Next") are also present.

Figure 7.12: Quiz 3 Results for Topic 7

# Proof of Completion

Congratulations to

Daniel Obon

For successfully completing

MongoDB CRUD Operations: Modifying Query Results

On 01-11-2024



Sahir Azam  
CPO  
MongoDB, Inc



MDBm7yg06d9gd

Figure 7.13: Topic 7 Proof of Completion

## Topic 8: MongoDB Aggregation

The screenshot shows a browser window for the MongoDB Learn website at <https://learn.mongodb.com/learn/course/mongodb-aggregation/lesson-1-introduction-to-mongodb-aggregation/practice?page=1>. The left sidebar displays a navigation menu with 'Lessons' selected, showing 'LESSON 1: INTRODUCTION TO MONGODB AGGREGATION' and 'Quiz: Introduction to Aggregation'. A progress bar indicates '20% Complete'. The main content area is titled 'Lesson 1: Introduction to MongoDB Aggregation / Practice'. It shows 'QUIZ RESULTS' with the message 'You got 2 out of 2 correct: 100%' and 'You passed!'. It also states '50% required for passing grade'. Below this, a note says 'Excellent! An aggregation pipeline can be used to filter for relevant data, to group documents, and to calculate total values from a field across many documents. An aggregation function is written as db.collection.aggregate()'.

Figure 8.1: Quiz 1 Results for Topic 8

The screenshot shows a browser window for the Instruqt MongoDB track at [https://ti.instruqt.com/?track=https://play.instruqt.com/embed/mongodb/tracks/match-and-group-stages?token=em\\_1xW3YQBEBVvX1Cw8jwI=ey/hbGcOIUu1NilsInR5cCl6lkp...](https://ti.instruqt.com/?track=https://play.instruqt.com/embed/mongodb/tracks/match-and-group-stages?token=em_1xW3YQBEBVvX1Cw8jwI=ey/hbGcOIUu1NilsInR5cCl6lkp...). The left pane shows a list of MongoDB documents with fields like '\_id', 'species\_common', 'species\_scientific', 'date', and 'location'. The right pane contains instructions for creating an aggregation pipeline. It starts with '#match and \$group Stages' and provides a hint: 'You have a database called bird\_data with a collection of sightings. We want to use this data to find out where we should go to see our favorite bird, Eastern Bluebirds. We'll use the location coordinates (latitude and longitude) and the number of sightings in each location to identify the best places to view the Eastern Bluebirds.' It then lists three steps under 'Instructions': 1. Create an aggregation pipeline with two stages. 2. Create a \$match stage filtering by species\_common. 3. Create a \$group stage grouping by location.coordinates. A 'Next' button is visible at the bottom right.

Figure 8.2: Aggregation using \$match and \$group

The screenshot shows a MongoDB University quiz results page. The main title is "Lesson 2: Using \$match and \$group Stages in a MongoDB Aggregation Pipeline / Practice". Below it, the "QUIZ RESULTS" section displays: "You got 2 out of 2 correct: 100%", "You passed!", and "50% required for passing grade". A message states: "Excellent! You've identified that the \$match & \$group stages are used to find and group documents in the aggregation pipeline." At the bottom are three buttons: "Retake Quiz", "Review All Question Results", and a green "Continue to Next Section" button.

Figure 8.3: Quiz 2 Results for Topic 8

The screenshot shows an Instruqt MongoDB aggregation lab interface. On the left, the terminal window shows a command being run: `db.sightings.aggregate([{\$sort: {location.coordinates: -1}}, {"\$limit: 4}])`. The output lists four bird sightings from the "bird\_data" collection. On the right, the "Instructions" panel provides context: "You have a database called bird\_data with a collection of sightings. We want to use this data to find the birds that are sighted furthest North." It also includes a list of tasks: 1. Create an aggregation pipeline. (Forgot the command or aggregation stages? Check the hint below!) 2. Use a \$sort stage to sort the data from North to South. To do this, use location.coordinates.i, noting the schema is { location: { coordinates: [x, y] } }, where the highest latitude value is the furthest North. The y in the schema represents latitude.

Figure 8.4: Aggregation using \$sort and \$limit

The screenshot shows the MongoDB University quiz results page for 'Lesson 3: Using \$sort and \$limit Stages in a MongoDB Aggregation Pipeline / Practice'. The sidebar on the left displays the 'MongoDB Aggregation' course structure, including lessons like 'Using \$sort and \$limit stages in a MongoDB Aggregation Pipeline' and 'Quiz: Using \$sort and \$limit Stages in a MongoDB Aggregation Pipeline'. A progress bar indicates 60% completion. The main content area shows the quiz results: 'You got 2 out of 2 correct: 100%', 'You passed!', and '50% required for passing grade'. It also includes a message: 'Congratulations! You learned what the \$sort and \$limit stages do in an aggregation pipeline!' Below the message are three buttons: 'Retake Quiz', 'Review All Question Results', and a green 'Continue to Next Section' button.

Figure 8.5: Quiz 3 Results for Topic 8

The screenshot shows an aggregation pipeline exercise on the Instruct platform. On the left, a terminal window titled 'mongosh' shows the command: 'db.sightings.aggregate([{"\$project": {"\_id": 0, "species\_common": 1, "date": 1}}])'. The right side of the screen displays a 'Progress' bar at 29m. The 'Instructions' section provides a hint: 'You have a database called bird\_data with a collection of sightings. There is a lot of data in each document, but we want to return only a list of the time of the sighting and the common name of the bird that was sighted.' Below the hint are four numbered steps: 1. Create an aggregation pipeline on the sightings collection. (Forgot the command or aggregation stages? Check the hint below!) 2. Create a \$project stage that just shows us the "date" and "species\_common" field. Project out the "\_id" field. 3. Run your aggregation pipeline, and see the list of sightings! 4. Once you complete this lab, compare your answer to the correct answer in the Review and Solved Code section, then select Next.

Figure 8.6: Aggregation using \$project

```

instruct >_ mongosh
{
  habitat: 'woodlands',
  diet: [ 'grain', 'seeds', 'fruit' ],
  last_seen: ISODate("2022-05-19T20:20:44.083Z"),
  class: 'bird'
},
{
  _id: ObjectId("6288e82d92f3fa87b7d86dcce"),
  common_name: 'Blue Jay',
  scientific_name: 'Cyanocitta cristata',
  wingspan_cm: 34.17,
  habitat: 'forests',
  diet: [ 'vegetables', 'nuts', 'grains' ],
  sightings_count: 4,
  last_seen: ISODate("2022-05-19T20:20:44.083Z"),
  class: 'bird'
},
{
  _id: ObjectId("6288e82d92f3fa87b7d86dcdd2"),
  common_name: 'Grackle',
  scientific_name: 'Quiscalus quiscula',
  wingspan_cm: 28.04,
  habitat: 'pine trees',
  diet: [ 'insects', 'minnows', 'eggs' ],
  sightings_count: 7,
  last_seen: ISODate("2022-05-19T20:20:44.083Z"),
  class: 'bird'
},
{
  _id: ObjectId("62cdaf53c1d62bc45439beb"),
  common_name: 'Grey Catbird',
  scientific_name: 'Dumetella carolinensis',
  wingspan_cm: 26.04,
  habitat: [ 'vegetation', 'scrublands', 'woodlands' ],
  diet: [ 'fruit', 'berries', 'worms' ],
  sightings_count: 3,
  class: 'bird'
}
]
Atlas atlas-d3opcw-shard-0 [primary] bird_data> []
[ mongoDB ](0*#mongosh mongoDB+srv://<credentials>@instructtest.xxfvk.mongodb.net/bird_data) [ 11/01 10:24 ]

```

**#set Stage**

You have a database called `bird_data` with a collection of information about bird species. In the future, we'll add new animals to the database and collection. Before we do that, we have to add a `class` field and set that field to `'bird'` in all of the existing documents in the collection

**Instructions**

You are now connected to an Atlas cluster and to the `bird_data` database. Use the `birds` collection in this lab.

1. Create an aggregation pipeline on the `birds` collection. (Forgot the command or aggregation stages? Check the hint below)
2. Create a new field called `class` populated with the class of each of these animals, `bird`.
3. Run your aggregation pipeline, and see that the `class` field has been added and set to "bird".
4. Once you complete this lab, compare your answer to the correct answer in the Review and Solved Code section, then select Next.

Figure 8.7: Aggregation using \$set

```

instruct >_ mongosh
{
  class: 'bird'
},
{
  _id: ObjectId("6288e82d92f3fa87b7d86dcdd2"),
  common_name: 'Grackle',
  scientific_name: 'Quiscalus quiscula',
  wingspan_cm: 28.04,
  habitat: 'pine trees',
  diet: [ 'insects', 'minnows', 'eggs' ],
  sightings_count: 7,
  last_seen: ISODate("2022-05-19T20:20:44.083Z"),
  class: 'bird'
},
{
  _id: ObjectId("62cdaf53c1d62bc45439beb"),
  common_name: 'Grey Catbird',
  scientific_name: 'Dumetella carolinensis',
  wingspan_cm: 26.04,
  habitat: [ 'vegetation', 'scrublands', 'woodlands' ],
  diet: [ 'fruit', 'berries', 'worms' ],
  sightings_count: 3,
  class: 'bird'
}
]
Atlas atlas-d3opcw-shard-0 [primary] bird_data> db.sightings.aggregate([
  {
    $match: {
      date: {
        $gt: ISODate('2022-01-01T00:00:00.000Z'),
        $lt: ISODate('2023-01-01T00:00:00.000Z')
      }
    },
    species_common: 'Eastern Bluebird'
  },
  {
    $count: 'bluebird_sightings_2022'
  }
])
[ { bluebird_sightings_2022: 5 } ]
Atlas atlas-d3opcw-shard-0 [primary] bird_data> []
[ mongoDB ](0*#mongosh mongoDB+srv://<credentials>@instructtest.xxfvk.mongodb.net/bird_data) [ 11/01 10:26 ]

```

**\$count stage**

You have a database called `bird_data` with a collection of sightings. We want to see the total number of sightings of Eastern Bluebirds in 2022.

**Instructions**

You are now connected to an Atlas cluster and to the `bird_data` database. Use the `sightings` collection in this lab.

1. Create an aggregation pipeline on the `sightings` collection. (Forgot the command or aggregation stages? Check the hint below)
2. Create a stage that finds matches where `species_common` is "Eastern Bluebird" and where the `date` field is a value between January 1, 2022 0:00, and January 1, 2023 0:00.
3. Create a stage to count how many sightings that includes.
4. Run your aggregation pipeline, and see how many sightings of Eastern Bluebirds took place in 2022.
5. Once you complete this lab, compare your answer to the correct answer in the Review and Solved Code section, then select Next.

Figure 8.8: Aggregation using \$count

The screenshot shows a browser window for MongoDB University. The main content area displays the results of a quiz titled "Lesson 4: Using \$project, \$count, and \$set Stages in a MongoDB Aggregation Pipeline / Practice". The results indicate that the user got 2 out of 2 correct, achieving 100% and passing the quiz. A message states: "Excellent! The \$count stage usually comes at the end of an aggregation pipeline, because it returns one document with a field named in the parameters with the value set to the number of documents at this stage in the aggregation pipeline." Below the results are three buttons: "Retake Quiz", "Review All Question Results", and "Continue to Next Section". On the left side of the screen, there is a sidebar titled "MongoDB Aggregation" with sections for "Lessons", "Quiz: Using \$project, \$count, and \$set", and "LESSON 5: USING THE \$OUT STAGE IN A MONGODB AGGREGATION". A progress bar shows 80% complete.

Figure 8.9: Quiz 4 Results for Topic 8

The screenshot shows a browser window for the Instruqt platform. The main area displays a MongoDB shell session with the following aggregation pipeline:

```

{
  ...
  {
    $match: {
      ...
      date: {
        $gte: ISODate('2022-01-01T00:00:00.0Z'),
        $lt: ISODate('2023-01-01T00:00:00.0Z')
      }
    }
  },
  {
    $out: 'sightings_2022'
  }
}

```

Below the shell session, the output shows a single document from the "bird\_data" collection. The right side of the screen contains a "Progress" bar (9m) and a "Instructions" section. The instructions guide the user through creating an aggregation pipeline to filter sightings from 2022 and output them to a new collection named "sightings\_2022".

Figure 8.10: Aggregation using \$out

The screenshot shows a web browser window for the MongoDB learning platform. The URL is [https://learn.mongodb.com/learn/course/mongodb-aggregation/lesson-5-using-the-\\$out-stage-in-a-mongodb-aggregation-pipeline/practice?client=customer&page=2](https://learn.mongodb.com/learn/course/mongodb-aggregation/lesson-5-using-the-$out-stage-in-a-mongodb-aggregation-pipeline/practice?client=customer&page=2). The main content area displays the results of a quiz titled "Lesson 5: Using the \$out Stage in a MongoDB Aggregation Pipeline / Practice". The results show a score of 2 out of 2 correct, with a message stating "You passed!". It also indicates that 50% was required for a passing grade. Below the results, a congratulatory message says "Congratulations! You learned how to use the \$out stage in aggregation". At the bottom, there are three buttons: "Retake Quiz", "Review All Question Results", and a green "Continue to Next Section" button. On the left side, there is a sidebar with sections for "Lessons", "CONCLUSION", "Progress 100%", "Resources & Forums", "Assignments", "Notes", and "Get Support". The "Progress 100%" section shows "100% Complete". The "SIGN OUT" link is located at the bottom right of the sidebar.

Figure 8.11: Quiz 5 Results for Topic 8



Figure 8.12: Topic 8 Proof of Completion

## Topic 9: MongoDB Indexes

The screenshot shows the MongoDB University interface. On the left, the navigation bar includes 'MongoDB Indexes' and a search bar. The main content area is titled 'Lesson 1: Using MongoDB Indexes in Collections / Practice'. It displays the 'QUIZ RESULTS' section with the message: 'You got 2 out of 2 correct: 100%', 'You passed!', '50% required for passing grade', and 'Incredible job learning!'. Below this are three buttons: 'Retake Quiz', 'Review All Question Results', and a green 'Continue to Next Page' button. Navigation arrows for 'Previous' and 'Next' are also present.

Figure 9.1: Quiz 1 Results for Topic 9

The screenshot shows the MongoDB University interface. On the left, the navigation bar includes 'MongoDB Indexes' and a search bar. The main content area is titled 'Lesson 2: Creating a Single Field Index in MongoDB / Practice'. It displays the 'QUIZ RESULTS' section with the message: 'You got 2 out of 2 correct: 100%', 'You passed!', '50% required for passing grade', and a detailed note: 'Great work! You learned that a single field index is an index that supports efficient querying against one field. You also learned that when a unique constraint is set to true, the indexed field does not store duplicate values.' Below this are three buttons: 'Retake Quiz', 'Review All Question Results', and a green 'Continue to Next Page' button. Navigation arrows for 'Previous' and 'Next' are also present.

Figure 9.2: Quiz 2 Results for Topic 9

The screenshot shows a browser-based MongoDB shell interface on the left and a corresponding lab exercise on the right.

**MongoDB Shell:**

```

Current Mongosh Log ID: 659fc07cd444e2405d858ee67
Connecting to: mongodb+srv://<credentials>@instructortest.3xfvk.mongodb.net/bank?appName=mongosh+1.10.6
Using MongoDB: 6.0.12
Using Mongosh: 1.10.6

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

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

Atlas atlas-d3opcw-shard-0 [primary] bank> db.transfers.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
Atlas atlas-d3opcw-shard-0 [primary] bank>

```

**Lab Exercise (Right):**

### Get Indexes for a MongoDB Collection

In this lab, you will retrieve an array of indexes that hold information about the indexes on the `transfers` collection.

Before you begin, please note that you are now connected to an Atlas cluster and the `bank` database. Use the `transfers` collection for this lab.

**Lab Instructions:**

- Run the command that returns an array of indexes on the `transfers` collection. *Forgot the command? Check the hints below!*
- After running the command, notice that the indexes are returned in an array.

```
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
```

Once you complete this lab, compare your answer to the correct answer in the Review and Solved Code section, then select Next.

**Buttons:** Skip | Next

Figure 9.3: Get Indexes for a MongoDB Collection using `getIndexes()`

The screenshot shows a browser-based MongoDB shell interface on the left and a corresponding lab exercise on the right.

**MongoDB Shell:**

```

Current Mongosh Log ID: 659fc80a5539af6501a7117d
Connecting to: mongodb+srv://<credentials>@instructortest.3xfvk.mongodb.net/bank?appName=mongosh+1.10.6
Using MongoDB: 6.0.12
Using Mongosh: 1.10.6

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

Atlas atlas-d3opcw-shard-0 [primary] bank> db.accounts.createIndex({ account_id: 1 }, { unique: true })
account_id_1
Atlas atlas-d3opcw-shard-0 [primary] bank> db.accounts.findOne()
{
  _id: ObjectId("62d6e04ecab6d8e130497482"),
  account_id: 'MDB829000996',
  account_holder: 'Kasper Serensen',
  account_type: 'checking',
  balance: Decimal128("4688"),
  transfers_completed: [
    'TR854612948',
    'TR413008451',
    'TR328708274',
    'TR193714910',
    'TR263422717'
  ]
}
Atlas atlas-d3opcw-shard-0 [primary] bank>

```

**Lab Exercise (Right):**

### Create an Index with an Equality Constraint

In this lab, we will create a single field index on the `accounts` collection of the `bank` database. We will create this single field index with a unique constraint on a field to ensure that there can only exist one document in the collection with a given value for the indexed field.

Before you begin, please note that you are now connected to an Atlas cluster and the `bank` database. Use the `accounts` collection for this lab.

**Lab Instructions:**

- In this lab, you will be working with the `accounts` collection. Once you are using the `accounts` collection, run `db.accounts.findOne()` to view the first document in the collection to understand the structure of the document. You should see a document similar to the following:

```
{
  _id: ObjectId("62d6e04ecab6d8e130497482"),
  account_id: 'MDB829000996',
  account_holder: 'Kasper Serensen',
```

**Buttons:** Skip | Check

Figure 9.4: Create an Index with an Equality Constraint using `createIndex()`

```

instruct >_ mongosh
{
  indexVersion: 2,
  direction: "forward",
  indexBounds: { account_id: [ ["MDB829000996", "MDB829000996"] ] }
},
{
  rejectedPlans: []
},
{
  command: {
    find: 'accounts',
    filter: { account_id: 'MDB829000996' },
    'db': 'bank'
  },
  serverInfo: {
    host: 'Atlas-d3opcw-shard-00-02.3xfvk.mongodb.net',
    port: 27017,
    version: '4.0.12',
    gitVersion: '2e6e8e11a45dfbdb7ca6cf95fa8c5f859e2b118'
  },
  serverParameters: {
    internalQueryFaceBufferSizeBytes: 104857600,
    internalQueryFaceMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentsSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxLockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongos: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
  },
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1704970452, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("32f250f0b2d0e34056dc44678bc11988d4f756f", "hex"), 0),
      keyId: Long("266845783591092292")
    }
  },
  operationTime: Timestamp({ t: 1704970452, i: 1 })
}
Atlas atlas-d3opcw-shard-0 [primary] bank>
[ mongodb ]( 0=> mongod ) 1*#mongosh mongod@srv://<credentials>@instructtest.3xfvk.mongodb.net/bank [ 11/01 10:54 ]

```

Use `explain()` to Verify that Indexes are Working

In this lab, we will use a command to view the `winningPlan` for a query that uses an index. A `winningPlan` is a document that contains information about the query and the method that was used to execute the query.

Before you begin, please note that you are now connected to an Atlas cluster and the `bank` database. Use the `accounts` collection for this lab.

**Lab Instructions**

1. In this lab, you will use the `accounts` collection. First, create a query that finds a document with the `account_id` field equal to `MDB829000996`.
2. Add the `explain()` method to your query to view the `winningPlan` (Forgot the command? Check the hints below):

```
db.accounts.explain().find({ account_id: "MDB829000996" })
```

Once you complete this lab, compare your answer to the correct answer in the Review and Solved Code section, then

Skip Next

Figure 9.5: Use `explain()` to Verify that Indexes are Working

Q Search X Hide

Lessons

▼ Practice

Quiz: Multikey Indexes in MongoDB

Lab: Creating a Multikey Single Field Index

LESSON 4: WORKING WITH COMPOUND INDEXES IN

Progress 60%

60% Complete

Show Details

Resources & Forums

Assignments

Notes

Get Support

FAQ'S SIGN OUT

Lesson 3: Creating a Multikey Index in MongoDB / Practice

Previous Next

QUIZ RESULTS

You got 1 out of 2 correct: 50%

You passed!

50% required for passing grade

Incredible job learning!

Retake Quiz Review All Question Results Continue to Next Page

Previous Next

Figure 9.6: Quiz 3 Results for Topic 9

```

instruct >_ mongosh
{
  indexVersion: 3,
  direction: 'forward',
  indexBounds: { transfers_complete: [ '^TR617907396', 'TR617907396' ] }
},
rejectedPlans: [],
command: {
  find: 'accounts',
  filter: { transfers_complete: { '$in': [ 'TR617907396' ] } },
  'db': 'bank'
},
serverInfo: {
  host: 'atlas-d3opcw-shard-00-02.3xfvk.mongodb.net',
  port: 27017,
  version: '4.0.12',
  gitVersion: '21ee8e11a45dfbd7ca6cf95fa8c5f859e2b118'
},
serverParameters: {
  internalQueryFacetBufferSizeBytes: 104857600,
  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalDocumentsSourceGroupMaxMemoryBytes: 104857600,
  internalQueryMaxLockingSortMemoryUsageBytes: 104857600,
  internalQueryProhibitBlockingMergeOnMongos: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentsSourceSetWindowFieldsMaxMemoryBytes: 104857600
},
ok: 1,
'@clusterTime': {
  clusterTime: Timestamp({ t: 1704970585, i: 7 }),
  signature: {
    hash: Binary(Buffer.from("010d53b6ce3b91f926471cc1c1c9c06cda5a1df3", "hex"), 0),
    keyId: Long("726845783591092292")
  }
},
operationTime: Timestamp({ t: 1704970585, i: 7 })
}
Atlas atlas-d3opcw-shard-0 [primary] bank> [
  mongosh ][ 0-> mongodb | 1*#mongosh mongodbserv://@instructtest,3xfvk.mongodb.net/bank | ] [ 11/01 10:56 ]

```

Figure 9.7: Creating Multikey Indexes and Using `explain()`

Lesson 4: Working with Compound Indexes in MongoDB / Practice

You got 2 out of 2 correct: 100%

You passed!

50% required for passing grade

Incredible job learning!

Retake Quiz      Review All Question Results      Continue to Next Page

Figure 9.8: Quiz 4 Results for Topic 9

```

        }
    ],
    command: {
        find: 'accounts',
        filter: { account_holder: 'Andrea', balance: { '$gt': 0 } },
        sort: { balance: 1 },
        projection: { account_holder: 1, balance: 1, account_type: 1, _id: 0 },
        '$db': 'bank'
    },
    serverInfo: {
        host: 'atlas-d3opcw-shard-00-02.3xfvk.mongodb.net',
        port: 27017,
        version: '6.0.12',
        gitVersion: '21ee8e11a45dfbdb7ca6cf5fa8c5f859e2b118'
    },
    serverParameters: {
        internalQueryFacetBufferSizeBytes: 104857600,
        internalQueryFacetMaxOutputDocSizeBytes: 104857600,
        internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
        internalDocumentsSourceGroupMaxMemoryBytes: 104857600,
        internalQueryMaxLockingSortMemoryUsageBytes: 104857600,
        internalQueryProhibitBlockingMergeOnMongoS: 0,
        internalQueryMaxAddToSetBytes: 104857600,
        internalDocumentsSourceSetWindowFieldsMaxMemoryBytes: 104857600
    },
    ok: 1,
    'clusterTime': {
        clusterTime: Timestamp({ t: 1704970772, i: 1 }),
        signature: {
            hash: Binary(Buffer.from("d1228d028faf67d25f4023c790b14815f2086fba", "hex"), 0),
            keyId: Long("7286845783591092292")
        }
    },
    operationTime: Timestamp({ t: 1704970772, i: 1 })
}
Atlas atlas-d3opcw-shard-0 [primary] bank> []
[ mongoDB ](0-> mongosh mongod+srv://<credentials>@instructtest.3xfvk.mongodb.net/bank) [ 11/01 10:59 ]

```

**Creating a Compound Index and Using explain()**

In this lab, we will create a compound index on the `accounts` collection of the `bank` database. Additionally, we will verify that the index was created and actively used in queries using the `explain()` method.

Before you begin, please note that you are now connected to an Atlas cluster and the `bank` database. Use the `accounts` collection for this lab.

**Lab Instructions**

1. In the `accounts` collection, create a compound index using the `account_holder`, `balance` and `account_type` fields. The `account_holder` field is the first field in the index and should be sorted in ascending order. The `balance` field is the second field in the index and should also be sorted in ascending order. Finally, the third field should be `account_type`, also sorted in ascending order. *Forgot the command? Check the hints below!*
2. Use the `explain` method to confirm that the index is being used. Be sure to look at the `winningPlan` field in the output as this contains information about the index

Skip | Check

Figure 9.9: Creating a Compound Index and Using `explain()`

MongoDB Indexes

Lessons

Practice

Quiz: Deleting MongoDB Indexes

Lab: Delete an Index

CONCLUSION

Learn

Progress 100%

100% Complete

Show Details

Resources & Forums

Assignments

Notes

Lesson 5: Deleting MongoDB Indexes / Practice

QUIZ RESULTS

You got 2 out of 2 correct: 100%

You passed!

50% required for passing grade

Excellent! You learned that deleting the only index that is supporting a query will negatively impact the performance of that query. You also learned that you can use the `hideIndex()` command to hide an index and assess if removing the index affects the query performance.

Retake Quiz | Review All Question Results | Continue to Next Page

Figure 9.10: Quiz 5 Results for Topic 9

The screenshot shows a browser window with two main panes. The left pane is a MongoDB shell session titled 'mongosh'. It displays a command to drop an index on the 'accounts' collection in the 'bank' database, followed by the execution of the command and its output. The right pane is a 'Delete an Index' lab interface from 'instruct'. It includes a progress bar at 9m, a 'Delete an Index' section with instructions, a 'Lab Instructions' section with numbered steps, and a code editor showing an 'explain()' command. At the bottom are 'Skip' and 'Check' buttons.

```

Current Mongosh Log ID: 659fca8ae557f36f44b781ed
Connecting to: mongodb+srv://<credentials>@instruct-test-shared-1.3xfvk.mongodb.net/bank?appName=mongosh+1.10.6
Using MongoDB: 6.0.12
Using Mongosh: 1.10.6

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

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

Atlas atlas-nmat34-shard-0 [primary] bank> db.accounts.dropIndex("account_holder_1")
{
  "IndexesWas": 2,
  "ok": 1,
  "@clusterTime": {
    "clusterTime": Timestamp({ t: 1704970911, i: 1 }),
    "signature": {
      "hash": Binary(Buffer.from("bc44c7bd1e2d9a9844eabc1b25be0356566f410c", "hex"), 0),
      "keyId": Long("7268662726010863618")
    }
  },
  "operationTime": Timestamp({ t: 1704970911, i: 1 })
}
Atlas atlas-nmat34-shard-0 [primary] bank>

```

Figure 9.11: Deleting an Index using dropIndex()



Figure 9.12: Topic 9 Proof of Completion

## Topic 10: MongoDB Atlas Search

The screenshot shows a browser window for MongoDB University. On the left, a sidebar titled 'MongoDB Atlas Search' lists 'Lessons' (Quiz: Using Relevance-Based Search and Search Indexes), 'Highlight Zone: Using Relevance-Based Search and Search Indexes', and 'LESSON 2: CREATING A SEARCH' (Progress 20%, 20% Complete). The main content area is titled 'Lesson 1: Using Relevance-Based Search and Search Indexes / Practice'. It displays 'QUIZ RESULTS' with a message: 'You got 2 out of 2 correct: 100%', 'You passed!', and '50% required for passing grade'. Below this, a message says: 'Great work! You've identified an example of a relevance-based search and a search index. You've also identified the analyzer, field mappings, and specific fields within a search index. You are now ready to build a search index!' At the bottom are buttons for 'Retake Quiz', 'Review All Question Results', and 'Continue to Next Page'.

Figure 10.1: Quiz 1 for Topic 10

The screenshot shows a browser window for Instruqt. On the left, a terminal window shows the command: 'root@mongodbs:~\$ atlas clusters search indexes create --clusterName myAtlasClusterEDU -f /app/search\_index.json'. The response indicates: 'Index sample\_supplies-sales-dynamic created.' On the right, a 'Progress' bar is at 100%. The main content area is titled 'Create a Atlas Search Index With Dynamic Mapping'. It contains a text block: 'In this lab, you will use the `atlas cli`, specifically the command `atlas clusters search indexes create` to create it.' Below this is a 'Lab Instructions' section with two steps: '1. Open the `search_index.json` file in the IDE by clicking the file name in the file explorer to the left.' and '2. Edit the JSON file in the IDE changing the value for the field `dynamic` to `true` to reflect the sample below.' A code snippet of the JSON file is shown: { "name": "sample\_supplies-sales-dynamic", "searchable": true }. There is a 'Check' button at the bottom right.

Figure 10.2: Creating an Atlas Search Index With Dynamic Mapping using atlas cli

The screenshot shows a terminal window within a browser tab. The URL is [https://instruct.instrqt.com/track=https://play.instrqt.com/embed/mongodb/tracks/creating-a-search-index-with-dynamic-mapping?token=en\\_9fMc3LosC4PIT-Dl8jvt-eyJhbGciOUI...](https://instruct.instrqt.com/track=https://play.instrqt.com/embed/mongodb/tracks/creating-a-search-index-with-dynamic-mapping?token=en_9fMc3LosC4PIT-Dl8jvt-eyJhbGciOUI...). The terminal output displays a MongoDB query result for a document in the `sample_supplies` collection. The document includes fields like `customer`, `purchaseMethod`, `score`, and an array of `items`. Each item has properties such as `name`, `tags`, `price`, and `quantity`. The terminal prompt shows the user is connected to a primary shard of the `atlas-atlas-rpekxp-shard-0` cluster.

**Using a Atlas Search Index With Dynamic Mapping**

In this lab, you will use the `mongosh` and the Aggregation Framework, specifically the `$search` stage to use your newly created Atlas Search index.

**Lab Instructions**

You will be connected to your Atlas cluster and to the `sample_supplies` database. Use the `sales` collection in this lab.

- Copy and paste the following command to use `mongosh` to connect to your Atlas cluster. For simplicity, we have included your connection string in the command by using the bash variable `MY_ATLAS_CONNECTION_STRING` and appended the correct database for this lab.

```
mongosh -u myAtlasDBUser -p myatlas-001 $MY_ATLAS_CONNECTION_STRING sample_supplies
```

- Next, create an aggregation pipeline, which will contain two stages. (Forgot the method for aggregation? Check the hint below!)
- Create a `$search` stage that uses the index

**Next**

Figure 10.3: Using an Atlas Search Index with Dynamic Mapping using `$search`

The screenshot shows a web-based MongoDB learning interface. On the left, there's a sidebar with navigation links for "Lessons", "Assignments", "Notes", and "Get Support". The main content area is titled "Lesson 2: Creating a Search Index with Dynamic Mapping / Practice". It displays the following information:

- QUIZ RESULTS:** You got 2 out of 2 correct: 100% (You passed!). 50% required for passing grade.
- Feedback:** You have created your first Atlas Search Index! In this activity, you learned how to create a dynamic search index which goes through every field. Next, you may want to learn how to make your search more specific and efficient by only indexing certain fields with static mapping.
- Buttons:** Retake Quiz, Review All Question Results, Continue to Next Section.

Figure 10.4: Quiz 2 for Topic 10

The screenshot shows a terminal window titled 'instruct' with a 'terminal' tab selected. The command `atlas clusters search indexes create --clusterName myAtlasClusterEDU -f /app/search\_index.json` is run, followed by `Index sample\_supplies-sales-static created.` Then, `atlas clusters search indexes list --clusterName myAtlasClusterEDU --db sample\_supplies --collection sales` is run, displaying the following table:

ID	NAME	DATABASE	COLLECTION	TYPE
659fd0d11824b72a0653ceb	sample_supplies-sales-dynamic	sample_supplies	sales	search
659fd20bf90d653df1b2dc9e	sample_supplies-sales-static	sample_supplies	sales	search

The right side of the interface contains a 'Create a Atlas Search Index With Static Mapping' section with instructions and a 'Lab Instructions' section with step 1.

Figure 10.5: Creating an Atlas Search Index with Static Mapping using `atlas cli`

The screenshot shows a terminal window titled 'instruct' with a 'terminal' tab selected. A complex JSON document is being inserted into the 'sales' collection. The document includes fields like '\_id', 'name', 'tags', 'price', 'quantity', 'storeLocation', 'customer', and 'score'. The right side of the interface contains a 'Using a Atlas Search Index With Static Mapping' section with instructions and a 'Lab Instructions' section with step 1.

Figure 10.6: Using an Atlas Search Index with Static Mapping with `$search`

The screenshot shows a quiz results page from MongoDB University. The left sidebar displays navigation links for 'MongoDB Atlas Search' and 'Lessons'. Under 'Lessons', it lists 'Lab: Creating a Search Index with Static Field Mapping' and 'Quiz: Dynamic vs. Static Field Mapping'. A progress bar indicates '60% Complete'. The main content area is titled 'Lesson 3: Creating a Search Index with Static Field Mapping / Practice'. It shows a 'QUIZ RESULTS' section with the message: 'You got 1 out of 2 correct: 50%', 'You passed!', and '50% required for passing grade'. Below this, a success message states: 'Great work! You've created a search index that maps specific fields. You've also correctly identified the syntax for dynamic and static mapping, after using them to build search indexes. With dynamic mapping, all fields are weighted equally. With static mapping, you can focus the search on specific fields which make your search more fast and efficient.' At the bottom, there are three buttons: 'Retake Quiz', 'Review All Question Results', and a green 'Continue to Next Section' button.

Figure 10.7: Quiz 3 for Topic 10

The screenshot shows an Instruqt terminal session titled 'terminal'. The terminal window displays a JSON document representing a collection of items. The document includes fields like '\_id', 'name', 'purchaseMethod', 'score', and 'items'. The 'items' field contains sub-documents with fields such as 'name' and 'purchaseMethod'. To the right of the terminal, a 'Progress' bar is shown, and the title 'Using a Atlas Search Index With Dynamic Mapping' is displayed. Below the title, instructions state: 'In this lab, you will use the `mongosh` and the Aggregation Framework, specifically the `$search` stage to use your newly created Atlas Search Index.' A 'Lab Instructions' section provides further details about connecting to the cluster and database. At the bottom of the terminal window, a command line interface shows the connection string: 'mongosh -u myAtlasDBUser -p mys'. A 'Next' button is visible at the bottom right.

Figure 10.8: Using an Atlas Search Index with Dynamic Mapping with \$search

**Lesson 4: Using \$search and Compound Operators / Practice**

**QUIZ RESULTS**

You got 2 out of 2 correct: 100%

**You passed!**

50% required for passing grade

Excellent! You have now created a relevance-based search function within an Atlas Search index! The \$search aggregation stage is very important for specifying how you want your application's search to work behind the scenes. In this activity, you have learned how to use a dynamic search index to create a \$search stage that uses the compound operator to filter based on a field and to weight a specific field. Next, you may want to explore other ways to modify how your relevance-based search works, such as including facets!

**Resources & Forums**

**Assignments**

**Notes**

**Retake Quiz**   **Review All Question Results**   **Continue to Next Section**

Figure 10.9: Quiz 4 for Topic 10

```
root@mongodbs:~$ atlas clusters search indexes create --clusterName myAtlasClusterEDU -f /app/search_index.json
Index sample_supplies-sales-facets created.
root@mongodbs:~$ atlas clusters search indexes list --clusterName myAtlasClusterEDU --db sample_supplies --collection sales
ID          NAME           DATABASE      COLLECTION   TYPE
659fd0d911824b72a0653ceb  sample_supplies-sales-dynamic  sample_supplies  sales    search
659fd20bf90d653df1b2dc9e  sample_supplies-sales-static   sample_supplies  sales    search
659fd3eff90d653df1b3669b  sample_supplies-sales-facets    sample_supplies  sales    search
root@mongodbs:~$
```

**Create a Atlas Search Index With Static Mapping**

In this lab, you will use the `atlas cli`, specifically the command `atlas clusters search indexes create` to create an static mapped Atlas Search index. In the next steps, we will use this index and query the dataset using a faceted search, specifically the field mapping will create a bucket for each store location so sales results can be presented by store location.

We have a dataset that contains records from an office supply company. The records contain information about orders, products, and customers. Imagine we are working on an application with a dashboard for administrators keeping track of sales at our office supply company. We want to add search functionality so that administrators can easily find sales matching their criteria. The administrators like to look at the sales by store and would like to organize search results by the store's location. We can

**Check**

Figure 10.10: Creating an Atlas Search Index with Static Mapping using atlas cli

```

Atlas atlas-rpekxp-shard-0 [primary] sample_supplies> db.sales.aggregate([
...   {
...     $searchMeta: {
...       index: 'sample_supplies-sales-facets',
...       "facet": {
...         "operator": {
...           "text": {
...             "query": "In store",
...             "path": "purchaseMethod"
...           }
...         },
...         "facets": {
...           "locationFacet": {
...             "type": "string",
...             "path": "storeLocation",
...           }
...         }
...       }
...     }
...   ],
...   [
...     {
...       count: { lowerBound: Long("2819") },
...       facets: {
...         locationFacet: {
...           buckets: [
...             { _id: 'Denver', count: Long("964") },
...             { _id: 'Seattle', count: Long("648") },
...             { _id: 'London', count: Long("455") },
...             { _id: 'Austin', count: Long("378") },
...             { _id: 'New York', count: Long("289") },
...             { _id: 'San Diego', count: Long("185") }
...           ]
...         }
...       }
...     }
...   ]
... )
Atlas atlas-rpekxp-shard-0 [primary] sample_supplies> []
[ mongoDB ] [ 0*@bash ] [ 11/01 11:42 ]

```

Using a Atlas Search Index With Static Mappings and Facets

In this lab, you will use the `mongosh` and the Aggregation Framework, specifically the `$searchMeta` stage to use your newly created Atlas Search index with facets.

Lab Instructions

You will be connected to your Atlas cluster and to the `sample_supplies` database. Use the `sales` collection in this lab.

- Copy and paste the following command to use `mongosh` to connect to your Atlas cluster. For simplicity, we have included your connection string in the command by using the bash variable `MY_ATLAS_CONNECTION_STRING` and appended the correct database for this lab.

```
mongosh -u myAtlasDBUser -p myse
```

- Next: Create an aggregation pipeline

Figure 10.11: Using an Atlas Search Index with Static Mapping and Facets using \$searchMeta

Search X Hide

Lessons

- Lab: Group Search Results by Using Facets
- Quiz: Group Search Results by Using Facets**

CONCLUSION

- Learn

Progress 100%

100% Complete

Show Details

Resources & Forums

Assignments

Notes

Get Support

FAQ'S SIGN OUT

Lesson 5: Group Search Results by Using Facets / Practice

Previous Next

QUIZ RESULTS

You got 2 out of 2 correct: 100%  
You passed!  
50% required for passing grade

Great work! In order to group search results into buckets and view their metadata, you must use the facet operator with the \$searchMeta aggregation stage.

Retake Quiz Review All Question Results Continue to Next Section

Previous Next

Figure 10.12: Quiz 5 for Topic 10

# Proof of Completion

Congratulations to

Daniel Obon

For successfully completing

MongoDB Atlas Search

On 01-11-2024



Sahir Azam  
CPO  
MongoDB, Inc



MDBu062atyv4

Figure 10.13: Topic 10 Proof of Completion

## Topic 11: MongoDB Data Modelling Intro

The screenshot shows the MongoDB Learn interface. On the left, a sidebar displays navigation links like 'Lessons', 'Lab: Introduction to the IDE', 'Quiz: Data Modeling' (which is highlighted in green), 'Lesson 2: TYPES OF DATA RELATIONSHIPS', 'Assignments', 'Notes', and 'Get Support'. The main content area is titled 'Lesson 1: Introduction to Data Modeling / Practice'. It shows a 'QUIZ RESULTS' section with the message: 'You got 3 out of 3 correct: 100%', 'You passed!', '50% required for passing grade', and 'Incredible job learning!'. Below this are three buttons: 'Retake Quiz', 'Review All Question Results', and 'Continue to Next Section'.

Figure 11.1: Quiz 1 for Topic 11

The screenshot shows the MongoDB Learn interface. The sidebar is identical to Figure 11.1. The main content area is titled 'Lesson 2: Types of Data Relationships / Practice'. It shows a 'QUIZ RESULTS' section with the message: 'You got 1 out of 2 correct: 50%', 'You passed!', '50% required for passing grade', and 'Incredible job learning!'. Below this are three buttons: 'Retake Quiz', 'Review All Question Results', and 'Continue to Next Section'.

Figure 11.2: Quiz 2 for Topic 11

The screenshot shows the MongoDB Data Modeling Intro interface. On the left, a sidebar displays 'Lessons' and 'Practice' sections, with 'Quiz: Modeling Data Relationships' currently selected. The main content area is titled 'Lesson 3: Modeling Data Relationships / Practice'. It shows a 'QUIZ RESULTS' section with the message: 'You got 2 out of 2 correct: 100%', 'You passed!', '50% required for passing grade', and 'Incredible job learning!'. Below this are three buttons: 'Retake Quiz', 'Review All Question Results', and a large green 'Continue to Next Section' button. Navigation arrows for 'Previous' and 'Next' are also present.

Figure 11.3: Quiz 3 for Topic 11

The screenshot shows the MongoDB Data Modeling Intro interface. The sidebar shows 'Lessons' and 'Practice' sections, with 'Quiz: Embedding Data in Documents' selected. The main content area is titled 'Lesson 4: Embedding Data in Documents / Practice'. It shows a 'QUIZ RESULTS' section with the message: 'You got 2 out of 2 correct: 100%', 'You passed!', '50% required for passing grade', and 'Incredible job learning!'. Below this are three buttons: 'Retake Quiz', 'Review All Question Results', and a large green 'Continue to Next Section' button. Navigation arrows for 'Previous' and 'Next' are also present.

Figure 11.4: Quiz 4 for Topic 11

The screenshot shows the MongoDB Data Modeling Intro interface. On the left, a sidebar displays 'Lessons' for 'Lesson 5: Referencing Data in Documents'. The main content area shows 'Lesson 5: Referencing Data in Documents / Practice'. It displays a 'QUIZ RESULTS' section with the message: 'You got 2 out of 2 correct: 100%', 'You passed!', '50% required for passing grade', and 'Incredible job learning!'. Below this are three buttons: 'Retake Quiz', 'Review All Question Results', and a green 'Continue to Next Section' button.

Figure 11.5: Quiz 5 for Topic 11

The screenshot shows the MongoDB Data Modeling Intro interface. On the left, a sidebar displays 'Lessons' for 'Lesson 6: Scaling a Data Model'. The main content area shows 'Lesson 6: Scaling a Data Model / Practice'. It displays a 'QUIZ RESULTS' section with the message: 'You got 3 out of 3 correct: 100%', 'You passed!', '50% required for passing grade', and 'Incredible job learning!'. Below this are three buttons: 'Retake Quiz', 'Review All Question Results', and a green 'Continue to Next Section' button.

Figure 11.6: Quiz 6 for Topic 11

The screenshot shows the MongoDB Learn interface. On the left, there's a sidebar with a search bar, a 'Lessons' section (Learn and Practice), and a 'Quiz: Data Explorer' card which is highlighted in green. Below that are sections for 'CONCLUSION' (Learn), 'Progress 100%' (100% Complete), 'Resources & Forums' (Assignments, Notes, Get Support), and 'FAQ'S'. At the bottom of the sidebar are 'SIGN OUT' and 'SIGN IN' buttons. The main content area is titled 'Lesson 7: Using Atlas Tools for Schema Help / Practice'. It displays 'QUIZ RESULTS' with the message: 'You got 1 out of 2 correct: 50%', 'You passed!', '50% required for passing grade', and 'Incredible job learning!'. There are three buttons at the bottom: 'Retake Quiz', 'Review All Question Results', and a green 'Continue to Next Section' button. Navigation arrows ('Previous' and 'Next') are located at the top and bottom of the main content area.

Figure 11.7: Quiz 7 for Topic 11

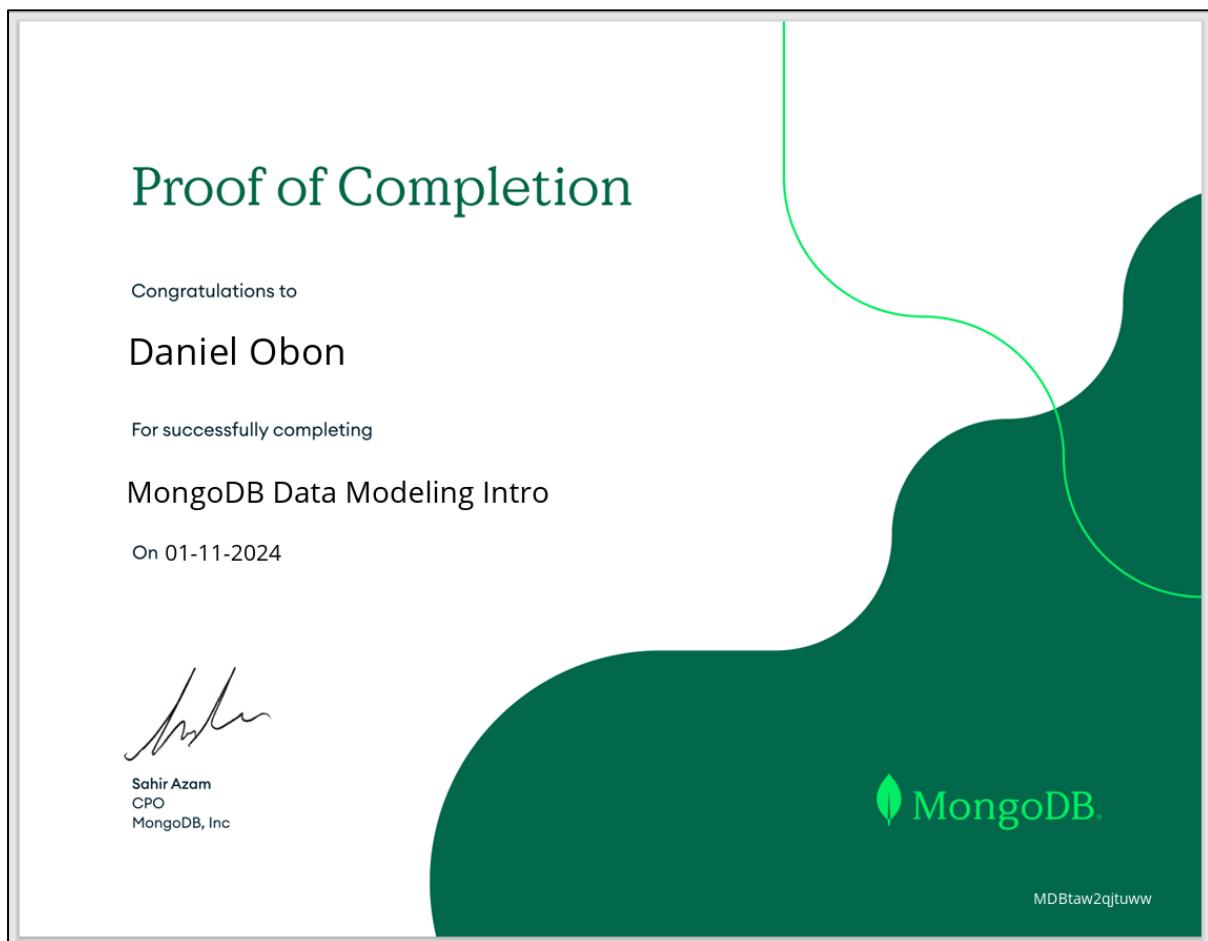


Figure 11.8: Topic 11 Proof of Completion

## Topic 12: MongoDB Transactions

The screenshot shows the MongoDB University interface. On the left, a sidebar displays 'Lessons' with 'Match Pairs: ACID Acronym' and 'Quiz: Introduction to ACID Transactions' highlighted. Below this is 'LESSON 2: ACID TRANSACTIONS IN MONGODB' with 'Learn' and 'Progress 33%' shown. The main content area is titled 'Lesson 1: Introduction to ACID Transactions / Practice'. It shows 'QUIZ RESULTS' with the message: 'You got 2 out of 2 correct: 100%', 'You passed!', '50% required for passing grade', and 'Incredible job learning!'. At the bottom are buttons for 'Retake Quiz', 'Review All Question Results', and 'Continue to Next Section'.

Figure 12.1: Quiz 1 for Topic 12

The screenshot shows the MongoDB University interface. On the left, a sidebar displays 'Lessons' with 'Practice' selected, showing 'Quiz: ACID Transactions in MongoDB' highlighted. Below this is 'LESSON 3: USING TRANSACTIONS IN MONGODB' with 'Learn' and 'Progress 67%' shown. The main content area is titled 'Lesson 2: ACID Transactions in MongoDB / Practice'. It shows 'QUIZ RESULTS' with the message: 'You got 2 out of 2 correct: 100%', 'You passed!', '50% required for passing grade', and 'Incredible job learning!'. At the bottom are buttons for 'Retake Quiz', 'Review All Question Results', and 'Continue to Next Section'.

Figure 12.2: Quiz 2 for Topic 12

The screenshot shows a browser window with the URL <https://instructt.com/>. The page title is "mongosh". The main content area displays a MongoDB shell session for the "bank" database. The session shows the creation of a new account document and its update. A progress bar at the top right indicates "Progress" and "Creating a Multi-Document Transaction". To the right of the session, there is a sidebar with instructions and a "Well done!" message.

```

Atlas atlas-d3opcw-shard-0 [primary] bank>
Atlas atlas-d3opcw-shard-0 [primary] bank> account.insertOne({
...   account_id: "MDB454252264",
...   account_holder: "Florence Taylor",
...   account_type: "savings",
...   balance: 100.0,
...   transfers_complete: [],
...   last_updated: new Date()
... })
{
  acknowledged: true,
  insertedId: ObjectId("659fde8f29af66b63373ba80")
}
Atlas atlas-d3opcw-shard-0 [primary] bank> account.updateOne( { account_id: "MDB963134500" }, { $inc: { balance: -100.00 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-d3opcw-shard-0 [primary] bank>
Atlas atlas-d3opcw-shard-0 [primary] bank> session.commitTransaction()
{
  ok: 1,
  "clusterTime": {
    clusterTime: Timestamp({ t: 1704976053, i: 2 }),
    signature: {
      hash: Binary(Buffer.from("21f483771e2e0d776d11b2123dcfdff2f9c26df?", "hex"), 0),
      keyid: Long("7266845783591092292")
    }
  },
  operationTime: Timestamp({ t: 1704976053, i: 1 })
}
Atlas atlas-d3opcw-shard-0 [primary] bank> []
[ mongodbs ][ 0*#mongosh: mongodb+srv://<credentials>@instructt-test.3xfvk.mongodb.net/bank ] [ 11/01 12:27 ]

```

Figure 12.3: Creating a Multi-Document Transaction

The screenshot shows a browser window with the URL <https://instructt.com/>. The page title is "mongosh". The main content area displays a MongoDB shell session for the "bank" database. The session shows the creation of a new account document and its update. A progress bar at the top right indicates "Progress" and "Abort a Multi-Document Transaction". To the right of the session, there is a sidebar with instructions and a "Next" button.

```

For mongosh info see: https://docs.mongodb.com/mongodb-shell/
Atlas atlas-d3opcw-shard-0 [primary] bank> var session = db.getMongo().startSession()
Atlas atlas-d3opcw-shard-0 [primary] bank>
Atlas atlas-d3opcw-shard-0 [primary] bank> session.startTransaction()
Atlas atlas-d3opcw-shard-0 [primary] bank>
Atlas atlas-d3opcw-shard-0 [primary] bank> var account = session.getDatabase('bank').getCollection('accounts')
Atlas atlas-d3opcw-shard-0 [primary] bank>
Atlas atlas-d3opcw-shard-0 [primary] bank> account.updateOne( { account_id: "MDB740836066" }, { $inc: { balance: 100 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-d3opcw-shard-0 [primary] bank>
Atlas atlas-d3opcw-shard-0 [primary] bank> account.updateOne( { account_id: "MDB963134500" }, { $inc: { balance: -5 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-d3opcw-shard-0 [primary] bank> session.abortTransaction()
Atlas atlas-d3opcw-shard-0 [primary] bank>
Atlas atlas-d3opcw-shard-0 [primary] bank> []
[ mongodbs ][ 0*#mongosh: mongodb+srv://<credentials>@instructt-test.3xfvk.mongodb.net/bank ] [ 11/01 12:28 ]

```

Figure 12.4: Abort a Multi-Document Transaction

The screenshot shows a web browser window for MongoDB University. The URL is <https://learn.mongodb.com/learn/course/mongodb-transactions/lesson-3-using-transactions-in-mongodb/practice?client=customer&page=2>. The main content area displays the results of a quiz titled "Lesson 3: Using Transactions in MongoDB / Practice". The quiz results section shows: "You got 1 out of 2 correct: 50%", "You passed!", and "50% required for passing grade". A message below says, "Congratulations! You learned how to commit transactions". At the bottom of this section are three buttons: "Retake Quiz", "Review All Question Results", and a green "Continue to Next Section" button. To the left of the main content is a sidebar with sections like "Lessons", "CONCLUSION", "Progress 100%", "Resources & Forums", "Assignments", and "Notes". The "Quiz: Using Transactions in MongoDB" section is highlighted in green.

Figure 12.5: Quiz 3 for Topic 12



Figure 12.6: Topic 12 Proof of Completion