

Lab 9 Managing Data

CRUD operations:

1. Create Operations

- `db.collection.insertOne()` : Inserts a document into a collection.

`db.collection.insertOne(<document>,<option>)`

- `db.collection.insertMany()` : Inserts multiple documents into a collection.

`db.collection.insertMany([<document1>, <document2>, ...], <option>)`

Reference:

<https://www.mongodb.com/docs/manual/reference/method/db.collection.insertOne/>

https://www.mongodb.com/docs/v6.0/reference/method/db.collection.insertMany/?_ga=2.24326713.2053810714.1665498267-1893344671.1664954494

Example:

```
db.mycol.insertOne(
  {
    title:      'MongoDB Overview',
    description: 'MongoDB is no sql database',
    url:        'http://www.tutorialspoint.com',
    tags:       ['mongodb', 'database', 'NoSQL'],
  }
)

db.products.insertMany( [
  { item: "card", qty: 15 },
  { item: "envelope", qty: 20 },
  { item: "stamps" , qty: 30 }
] );
```

2. Read Operations

- `db.collection.find(query, projection)`: Selects documents in a collection or view and returns a [cursor](#) to the selected documents.

Parameter	Type	Description
query	document	Optional. Specifies selection filter using query operators . To return all documents in a collection, omit this parameter or pass an empty document (<code>{}</code>).
projection	document	Optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter. For details, see Projection .

Returns: A cursor to the documents that match the query criteria. When the `find()` method “returns documents,” the method is actually returning a cursor to the documents.

Comparison Operator:

Syntax: `{ <field> : {<operator> : <value>} }`

`$eq` = Equal to , `$neq` != Not Equal to, `$gt` > Greater Than,

`$gte` >= Greater Than or Equal to, `$lt` < Less Than,

`$lte` <= Less Than or Equal to

Logical Operator:

Syntax: `{ <operator> : [{clause1}, {clause2},...] }`

Syntax: `{ $not : {clause1} }`

- \$and** Match all of the specified query clauses
- \$or** At least one of the query clauses is matched
- \$nor** Fail to match both of given clauses
- \$not** Negates the query requirement

Example:

```
db.routes.find({ "$and": [ { "$or" :[ { "dst_airport": "KZN" },
                                   { "src_airport": "KZN" }
                               ] },
                { "$or" :[ { "airplane": "CR2" },
                           { "airplane": "A81" } ] }
              ] })

db.trips.find({ "tripduration": { "$lte" : 70 },
               "usertype": { "$ne": "Subscriber" }
            })
```

Resource:

<https://docs.mongodb.com/v4.2/reference/method/db.collection.find/>

Cursor Methods: These methods modify the way that the underlying query is executed.

Some Cursor Methods:

cursor.count()	Modifies the cursor to return the number of documents in the result set rather than the documents themselves.
-----------------------	---

cursor.limit()	<p>Constrains the size of a cursor's result set.</p> <p><code>db.collection.find(<query>).limit(<number>)</code></p> <p>You must specify a numeric value for limit().</p> <p>A limit() value of 0 (i.e. .limit(0)) is equivalent to setting no limit.</p>
-----------------------	---

cursor.pretty()	Configures the cursor to display results in an easy-to-read format.
------------------------	---

cursor.sort() Returns results ordered according to a sort specification.
`sort({field : value}, {...})`

Specify in the sort parameter the field or fields to sort by and a value of 1 or -1 to specify an ascending or descending sort respectively.

Example: `sort({ age : -1, posts: 1 })`

More cursor methods: <https://docs.mongodb.com/v4.2/reference/method/js-cursor/>

Example:

```
db.trips.find({ "tripduration": { "$lte" : 70 } }).limit(3);

db.trips.find({ "tripduration": { "$lte" : 70 } }).sort({"tripduration": -1});
```

3. Update Operations

- `db.collection.updateOne ({filter/query}, {update}, {options})`: Updates a single document within the collection based on the filter, finds the first document that matches the filter and applies the specified update modifications.
- `db.collection.updateMany ({filter/query}, {update}, {options})`: updates all matching documents in the collection that match the filter, using the update criteria to apply modifications.

Update Operators:

Fields

[`\$inc`](#): Increments the value of the field by the specified amount.

[`\$rename`](#): Renames a field.

[`\$set`](#): Sets the value of a field in a document.

[`\$unset`](#): Removes the specified field from a document.

Arrays

[`\$pull`](#): Removes all array elements that match a specified query.

[`\$push`](#): Adds an item to an array.

More update operators:

<https://docs.mongodb.com/v4.2/reference/operator/update/#id1>

Example:

```

db.<collection>.updateOne(
  <query>,
  { $set: { status: "D" }, $inc: { quantity: 2 } },
  ...
)

db.restaurant.updateMany(
  { violations: { $gt: 4 } },
  { $set: { "Review" : true } }
);

```

Reference:

<https://www.mongodbtutorial.org/mongodb-crud/mongodb-updateone/>

<https://www.geeksforgeeks.org/mongodb-updateone-method-db-collection-updateone/>

4. Delete Operations

- `db.collection.deleteOne({filter/query}, {options})`: Removes a single document from a collection and deletes the first document that matches the query.
- `db.collection.deleteMany({filter/query}, {options})`: Removes all documents that match the query from a collection.
- `db.<collection>.drop()`: Removes the given collection.

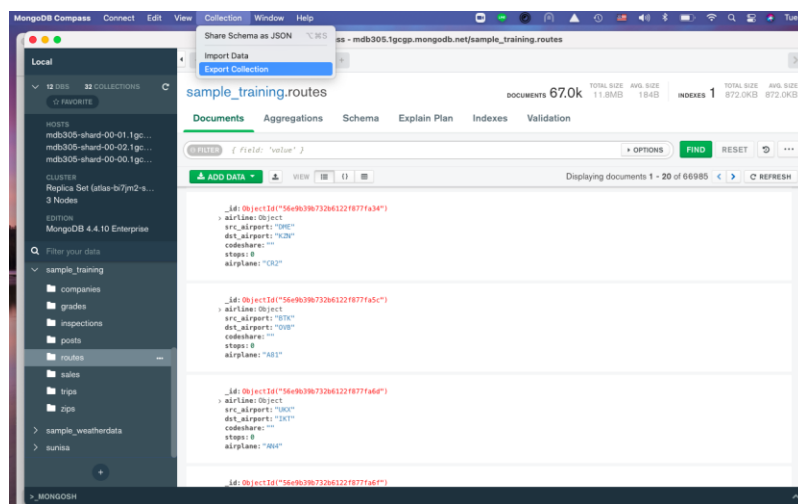
Instruction:

- Use the MongoDB shell (MONGOSH) in the MongoDB Compass for writing commands.
- Use the MongoDB Compass GUI to perform export and import data.

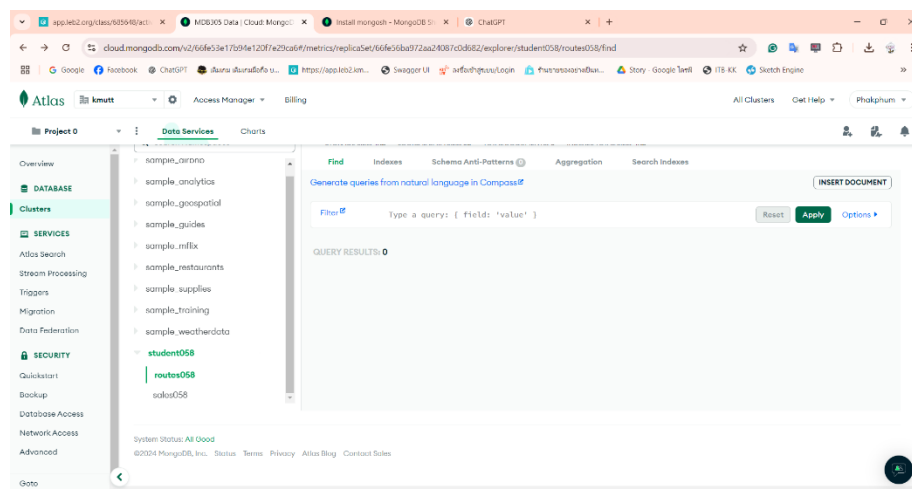
Task 1: Creating a MongoDB database

1.1 Use the MongoDB Compass GUI to export an existing collection named “routes” of the SAMPLE_TRAINING database to a JSON file.

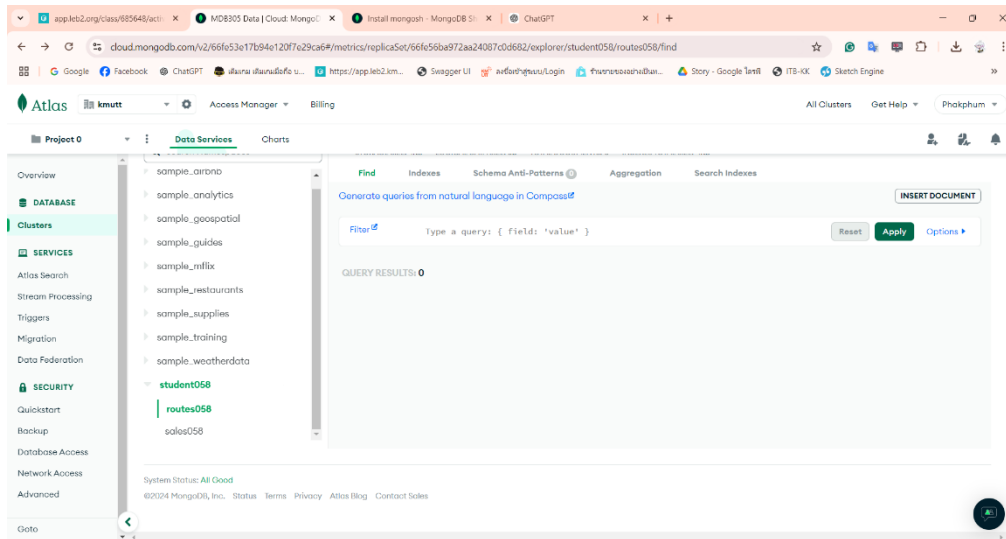
- Click on the collection name that you want to export
- Click on the **Collection** menu at the top and then click the **Export Collection** menu



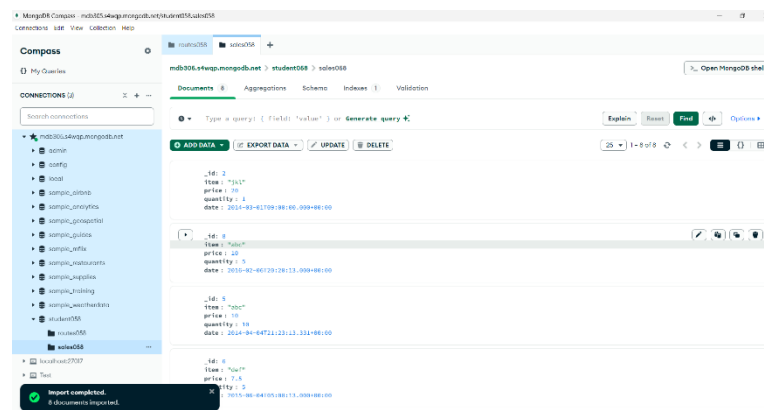
1.2 Write a command to create a new database name “db<studentID>”



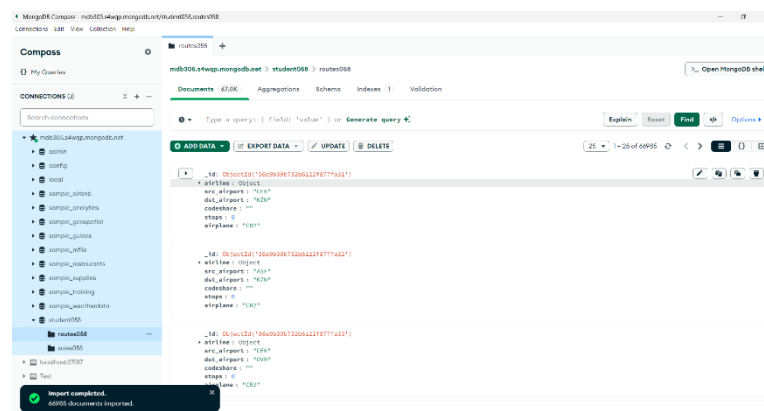
1.3 Write commands to create a collection named “salesXXX” and “routesXXX” (where XXX = the last 3 digits of your student ID).



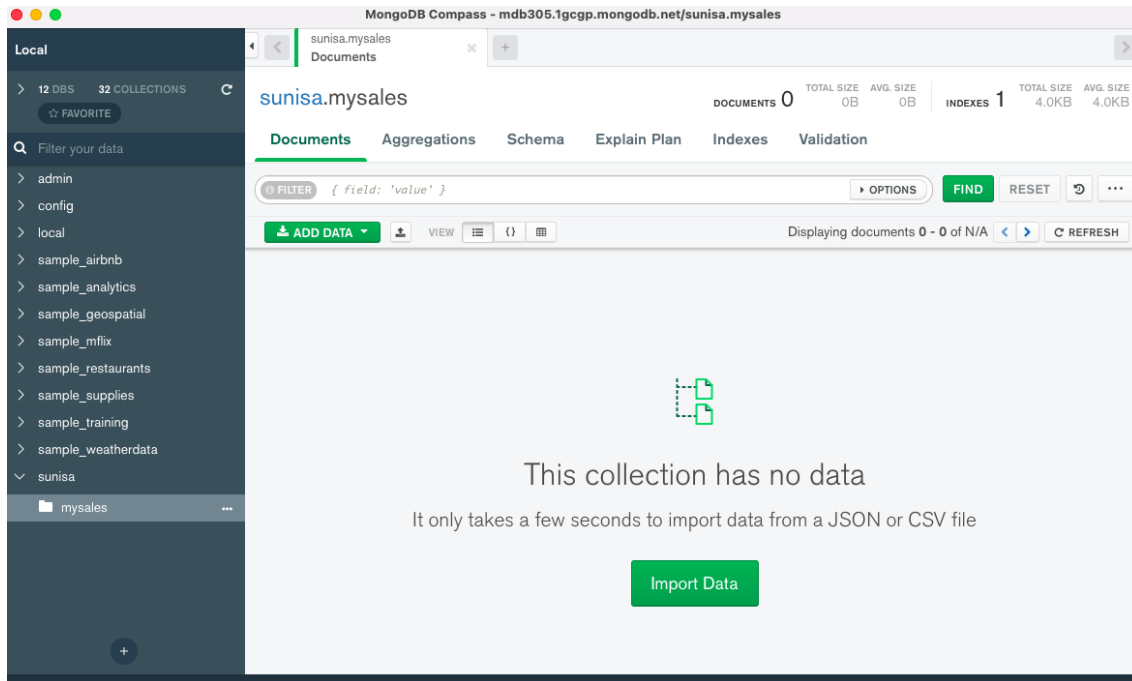
1.4 Download the JSON file named “sales_training.json” from the LEB2 and import all documents using the “sales_training.json” file into the “salesXXX” collection.



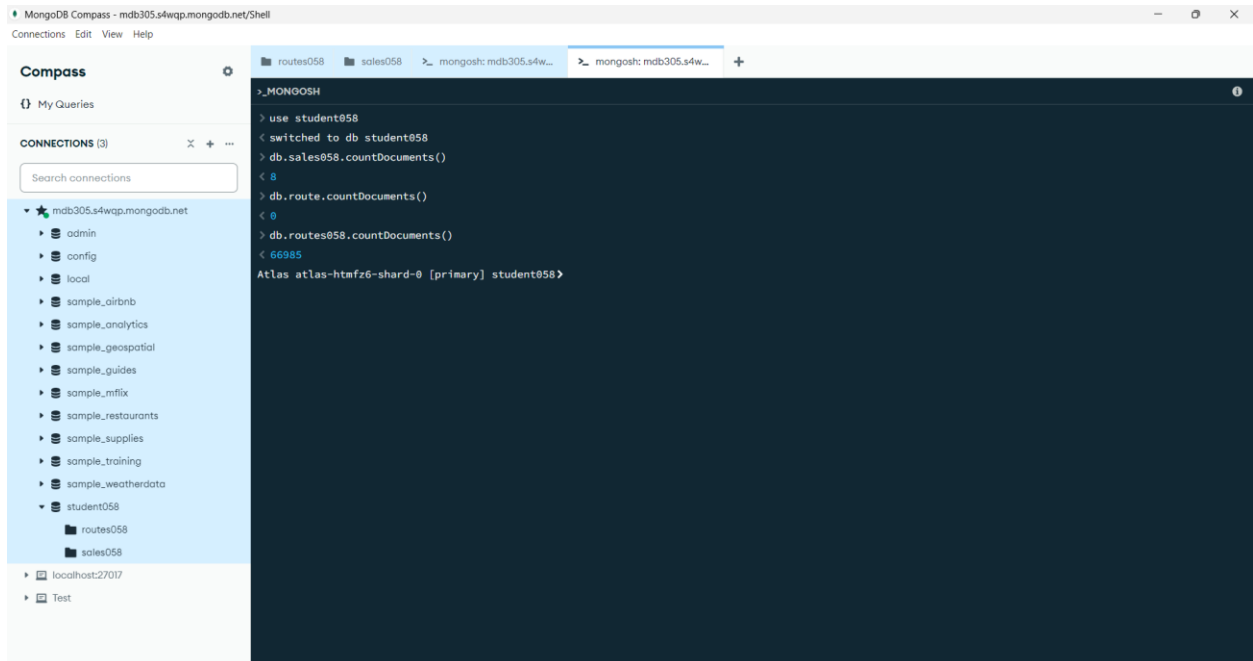
1.5 Import the JSON file in question 1.1 into the new collection named “routesXXX” of your new database in question 1.3.



Hint: Before clicking on the **Import Data** button, you should click on the collection name that you want to store data from the import files first.



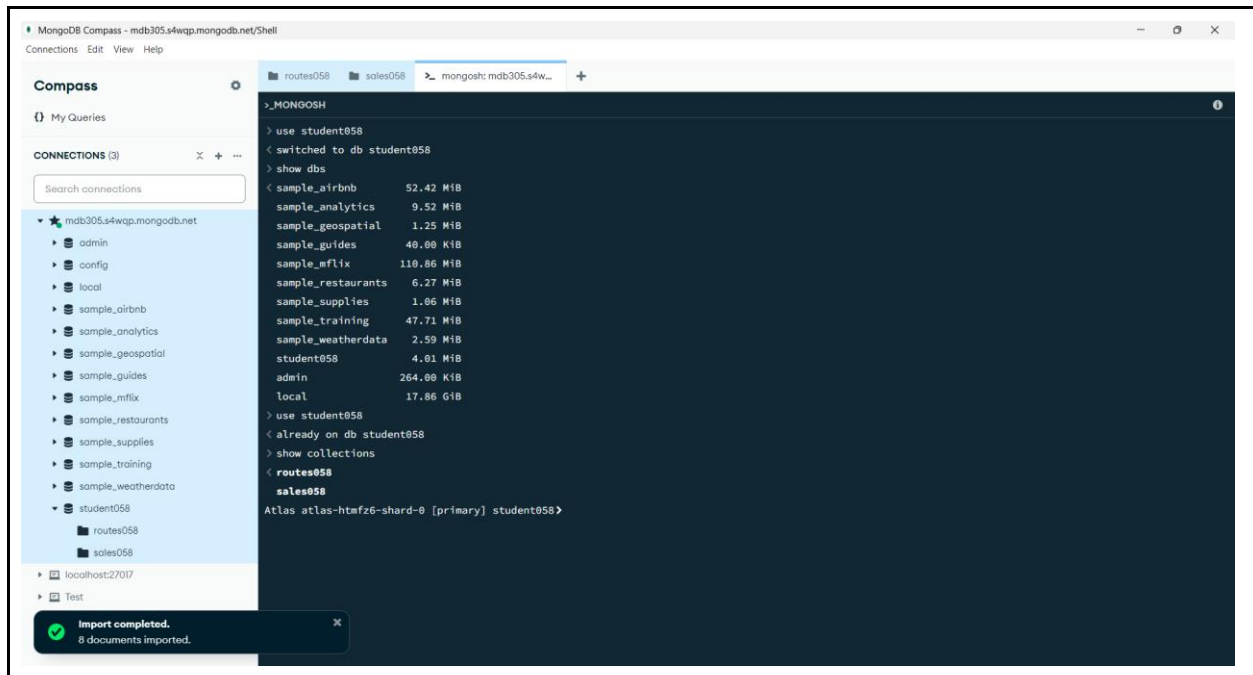
After successfully importing in question 1.4-1.5, please write commands to show the number of documents in each collection.



1.6 Run the following commands:

- > show dbs;
- > use <your new database>;
- > show collections;

Capture a screenshot(s) of all results and paste here:



Task 2: Performing CRUD operations

Use your new database named “db<StudentID>” created in Task 1 and write MongoDB commands for answering the following questions:

2.1 Write commands to add two new documents into the “salesXXX” collection. Please capture the command and the result of each command.

```

MongoDB Compass - mdb305.s4wqp.mongodb.net/Shell
Connections Edit View Help

Compass
My Queries
CONNECTIONS (3)
  Search connections
  mdb305.s4wqp.mongodb.net
    admin
    config
    local
    sample_airbnb
    sample_analytics
    sample_geospatial
    sample_guides
    sample_mflix
    sample_restaurants
    sample_supplies
    sample_training
    sample_weatherdata
  student058
    routes058
    sales058
  localhost:27017
  Test

>_MONGOSH
> use student058
< switched to db student058
> db.sales058.insertMany(
  { '_id':9, 'item':'jkl', 'price':20, 'quantity':2, 'date':ISODate("2014-10-01T09:00:00.000+00:00") },
  { '_id':10, 'item':'def', 'price':7.5, 'quantity':10, 'date':ISODate("date:2014-09-09T09:00:00.000+00:00") })
• MongoInvalidInputError: [COMMON-10001] "date:2014-09-09T09:00:00.000+00:00" is not a valid ISODate
> db.sales058.insertMany(
  { '_id':9, 'item':'jkl', 'price':20, 'quantity':2, 'date':ISODate("2014-10-01T09:00:00.000+00:00") },
  { '_id':10, 'item':'def', 'price':7.5, 'quantity':10, 'date':ISODate("2014-09-09T09:00:00.000+00:00") })
• MongoInvalidArgumentError: Argument "docs" must be an array of documents
> db.sales058.insertMany([
  { '_id':9, 'item':'jkl', 'price':20, 'quantity':2, 'date':ISODate("2014-10-01T09:00:00.000+00:00") },
  { '_id':10, 'item':'def', 'price':7.5, 'quantity':10, 'date':ISODate("2014-09-09T09:00:00.000+00:00") } ])
< {
  acknowledged: true,
  insertedIds: {
    '0': 9,
    '1': 10
  }
}
Atlas atlas-htmfz6-shard-0 [primary] student058>

```

```

MongoDB Compass - mdb305.s4wqp.mongodb.net/Shell
Connections Edit View Help

Compass
My Queries
CONNECTIONS (3)
  Search connections
  mdb305.s4wqp.mongodb.net
    admin
    config
    local
    sample_airbnb
    sample_analytics
    sample_geospatial
    sample_guides
    sample_mflix
    sample_restaurants
    sample_supplies
    sample_training
    sample_weatherdata
  student058
    routes058
    sales058
  localhost:27017
  Test

>_MONGOSH
> db.sales058.find({'_id': { $in:[9,10]}})
< [
  {
    _id: 9,
    item: 'jkl',
    price: 20,
    quantity: 2,
    date: 2014-10-01T09:00:00.000Z
  },
  {
    _id: 10,
    item: 'def',
    price: 7.5,
    quantity: 10,
    date: 2014-09-09T09:00:00.000Z
  }
]
Atlas atlas-htmfz6-shard-0 [primary] student058>

```

Hint: For DATE data type, you may use `ISODate()` to convert data to date format.

Example => date: `ISODate("2014-10-01T09:00:00.000+00:00")`

```
{_id:9, item:"jkl", price:20, quantity:2, date:2014-10-01T09:00:00.000+00:00}
{_id:10, item:"def", price:7.5, quantity:10, date:2014-09-09T09:00:00.000+00:00}
```

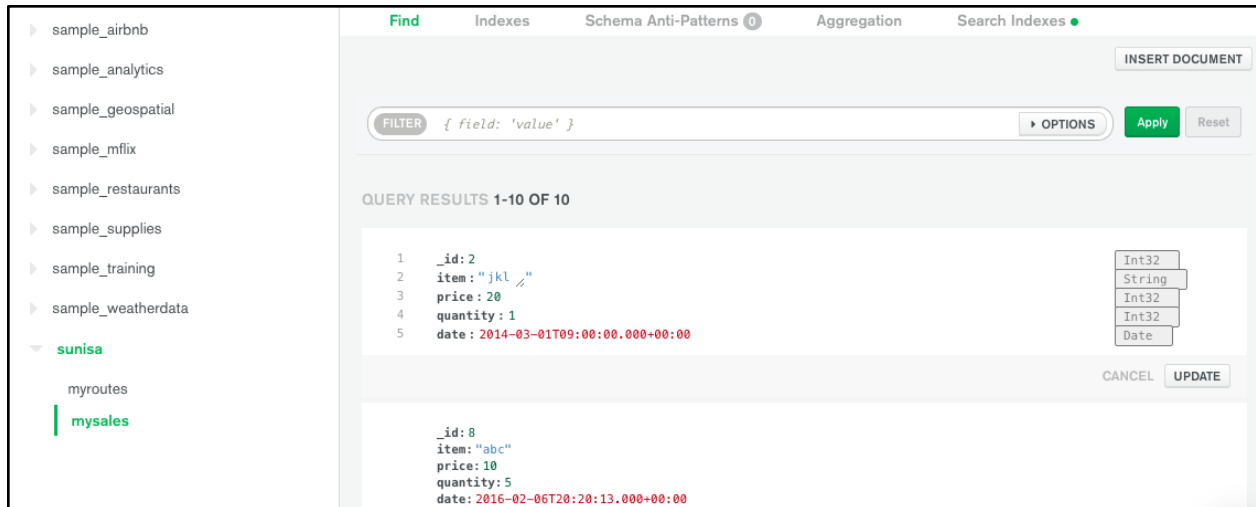
Field Name	Data Type
_id	Integer
item	String
price	Integer
quantity	Integer
date	Date

After the insertion, please refresh data in both the MongoDB Atlas and MongoDB Compass

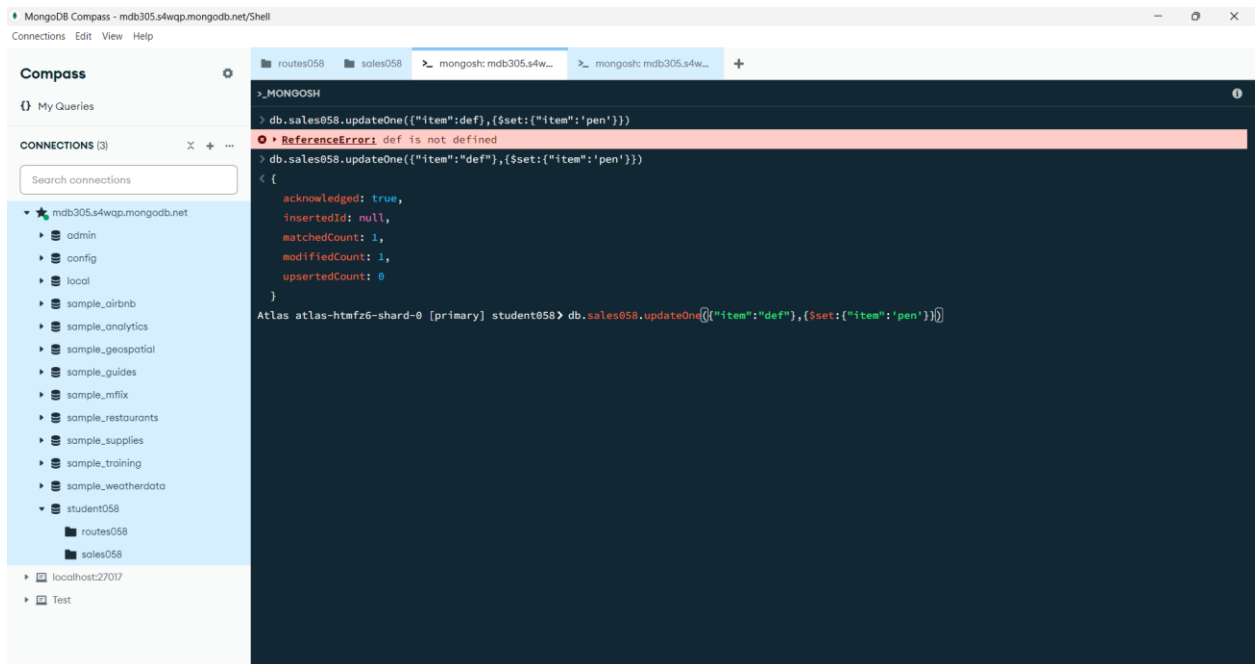
Note: if you want to check the data types of each field, you can check by clicking on the Edit icon in MongoDB Atlas Console as follows:

The screenshot shows the MongoDB Atlas console interface. On the left, a sidebar lists various sample datasets, with 'mysales' selected under the 'sunisa' category. The main panel displays the 'Find' tab with a filter bar containing the text '{ field: 'value' }'. Below the filter bar, the 'QUERY RESULTS 1-10 OF 10' section shows two document snippets. The first snippet is for '_id: 2' and contains the fields: 'item: "jkl"', 'price: 20', 'quantity: 1', and 'date: 2014-03-01T09:00:00.000+00:00'. The second snippet is for '_id: 8' and contains the fields: 'item: "abc"', 'price: 10', 'quantity: 5', and 'date: 2016-02-06T20:20:13.000+00:00'. To the right of the document snippets, there are icons for editing, copying, and deleting, along with an 'Edit Document' button.

- You will see data types of each field.



2.2 Use updateOne() collection method to change the item of all sales documents that have the item named "def" to a new item named "pen". You should query all documents of the item "def" first before the modification. Please show the result of each command.



2.3 Use `updateMany()` collection method to change the item named “jkl” to a new item named “book”. You should query all documents of the item “jkl” first before the modification. Please show the results of each command.

The screenshot shows the MongoDB Compass interface. On the left, the 'Connections' panel lists several databases, including 'student058' which contains collections 'routes058' and 'sales058'. The main panel shows a shell session with the following commands and results:

```

> db.sales058.updateOne({"item": "def"}, {$set: {"item": "pen"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

> db.sales058.updateMany({"item": "jkl"}, {$set: {"item": "book"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}

```

A red error message is visible above the second command: `ReferenceError: def is not defined`. The status bar at the bottom indicates the connection is 'Atlas atlas-htmfz6-shard-0 [primary] student058'.

2.4 Update the quantity field of all sales documents that _id is less than 5 to increment value by 2.

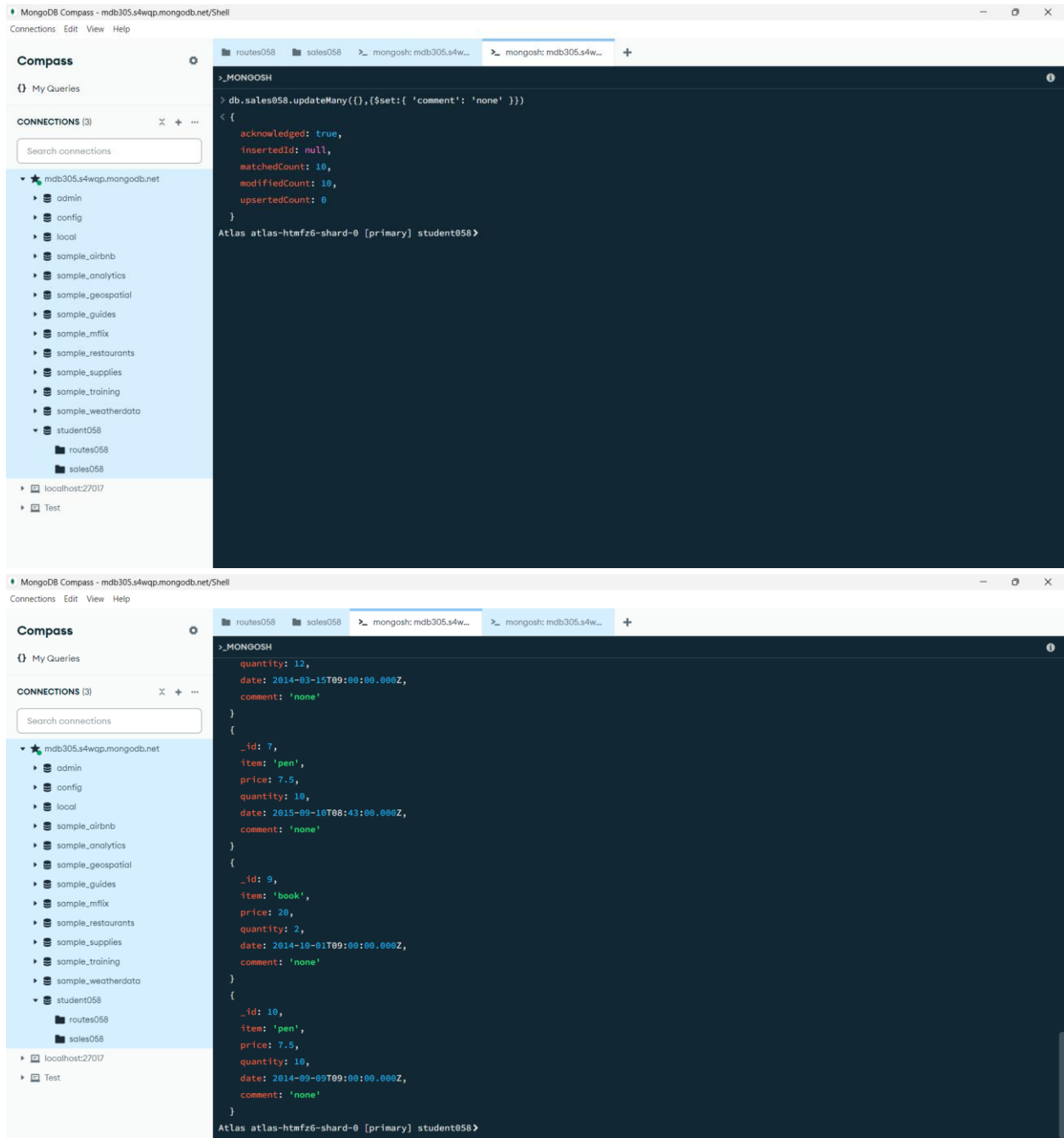
The first screenshot shows the MongoDB Compass interface with the 'sales058' collection selected. The command window displays the following query:

```
>_MONGOOSH
> db.sales058.updateMany(
  { "_id": { $lt: 5 } },
  { $inc: { 'quantity': 2 } }
)
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
}
```

The second screenshot shows the same interface after the query has been executed. The command window displays the resulting documents:

```
< {
  {
    _id: 1,
    item: 'abc',
    price: 10,
    quantity: 4,
    date: 2014-03-01T08:00:00.000Z
  },
  {
    _id: 2,
    item: 'book',
    price: 20,
    quantity: 3,
    date: 2014-03-01T09:00:00.000Z
  },
  {
    _id: 3,
    item: 'xyz',
    price: 5,
    quantity: 12,
    date: 2014-03-15T09:00:00.000Z
  },
  {
    _id: 4,
    item: 'xyz',
    price: 5,
    quantity: 22,
    date: 2014-04-04T11:21:39.736Z
  }
}
```

2.5 Add a new field named “comment” with value “none” into all documents of the “salesXXX” collection. After the modification, please display only the comment field of all documents. Please show the results of each command.



The first screenshot shows the MongoDB Compass interface with the MongoDB shell open. The command executed is:

```
> db.sales058.updateMany({},{$set:{ 'comment': 'none' }})
```

The result of the command is:

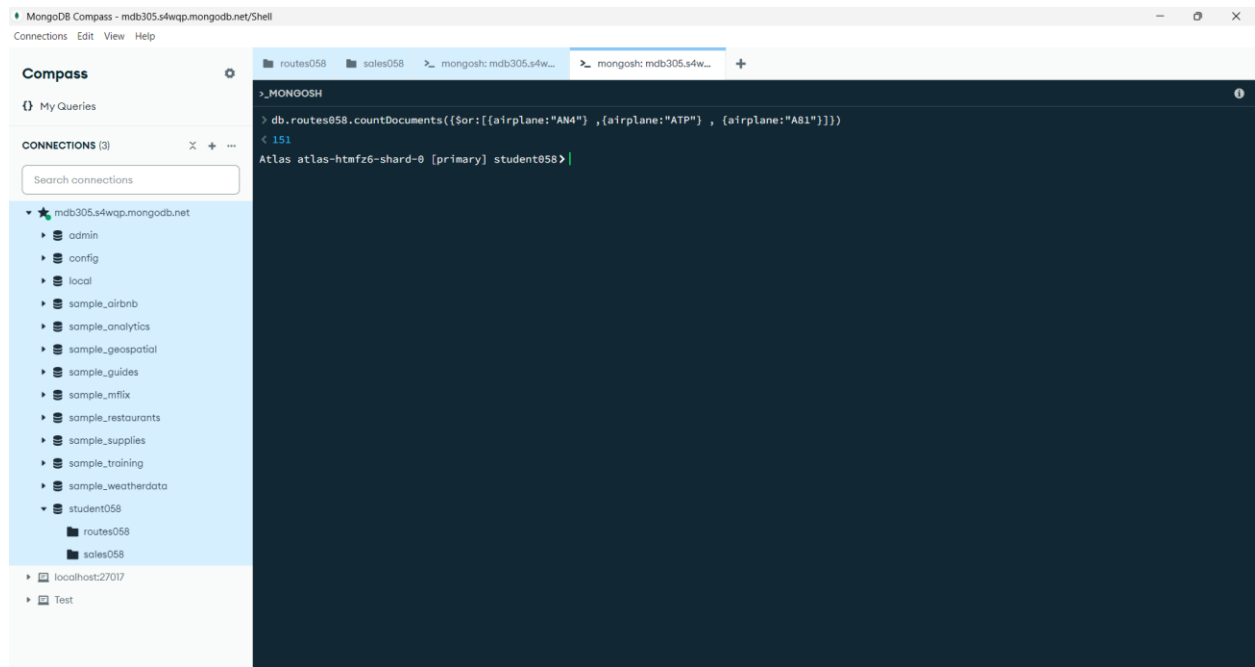
```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 10,
  modifiedCount: 10,
  upsertedCount: 0
}
```

The second screenshot shows the same MongoDB Compass interface, but the shell now displays the result of the command, showing the updated documents with the 'comment' field:

```
{
  quantity: 12,
  date: 2014-03-15T09:00:00.000Z,
  comment: 'none'
},
{
  _id: 7,
  item: 'pen',
  price: 7.5,
  quantity: 10,
  date: 2015-09-10T08:43:00.000Z,
  comment: 'none'
},
{
  _id: 9,
  item: 'book',
  price: 20,
  quantity: 2,
  date: 2014-10-01T09:00:00.000Z,
  comment: 'none'
},
{
  _id: 10,
  item: 'pen',
  price: 7.5,
  quantity: 10,
  date: 2014-09-09T09:00:00.000Z,
  comment: 'none'
}
```

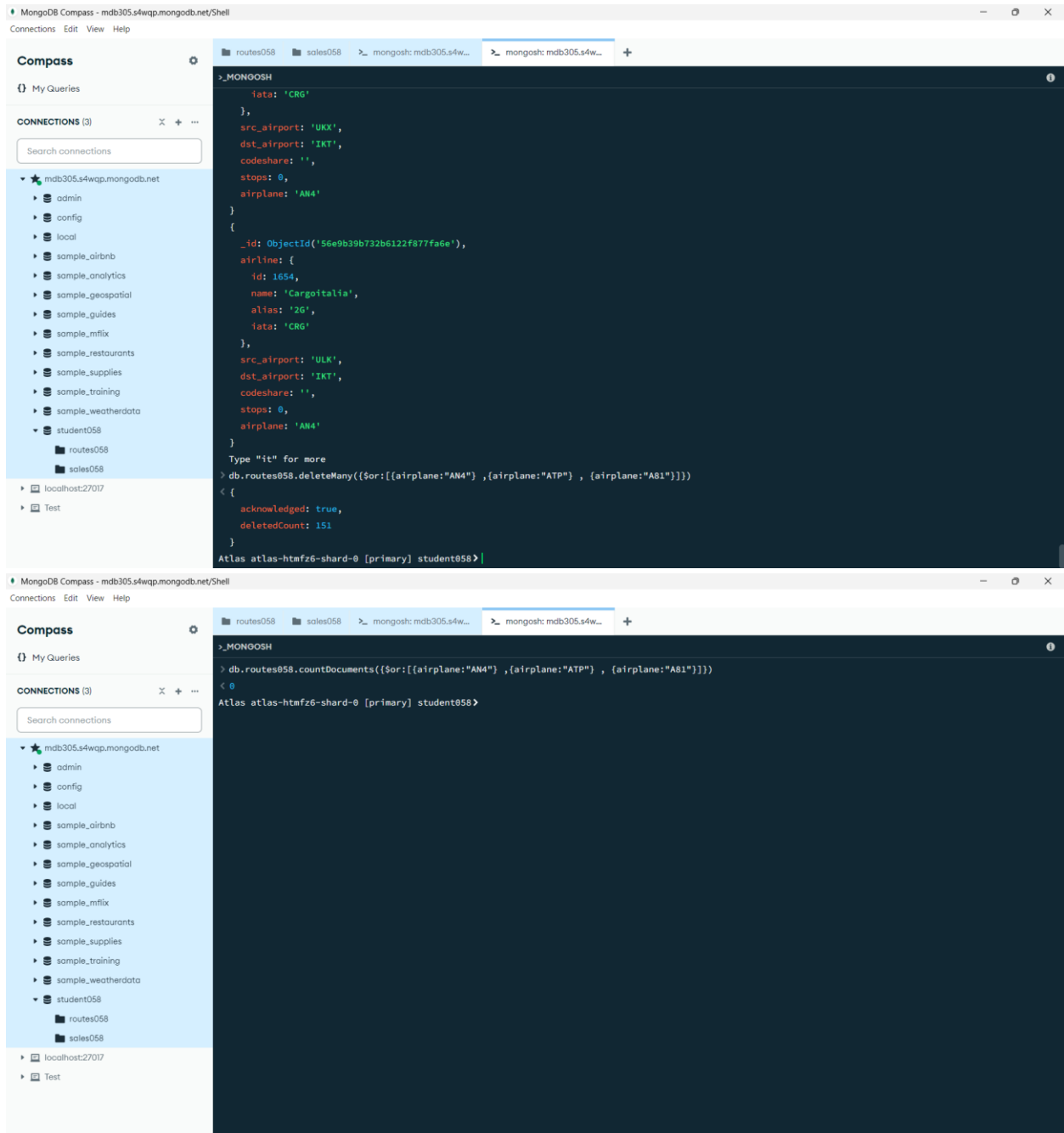
Use the **db<StudentID>.routesXXX** namespace

2.6 How many documents show that their airplane name is “AN4” or “ATP” or “A81”. Please show the results of each command.



2.7 Delete all documents that their airplane name is "AN4" or "ATP" or "A81".

After the deletion, please verify that the deleted data is gone.



2.8 Display the airline name and airplane fields of all documents in the “routesXXX” collection that the airplane is “CR2”. Display only 5 documents. Please show the results of each command.

The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS' panel lists several connections, with 'mongo305.s4egp.mongodb.net' selected. The main panel displays a MongoDB shell session with the following commands and results:

```

> MONGOOSH
> db.routes858.find({airplane:"CR2"},{'_id':0,"airplane":1, "airline.name":1}).limit(5)
<
[
  {
    "airline": {
      "name": "Aerocendor"
    },
    "airplane": "CR2"
  },
  {
    "airline": {
      "name": "Aerocendor"
    },
    "airplane": "CR2"
  },
  {
    "airline": {
      "name": "Aerocendor"
    },
    "airplane": "CR2"
  },
  {
    "airline": {
      "name": "Aerocendor"
    },
    "airplane": "CR2"
  },
  {
    "airline": {
      "name": "Aerocendor"
    },
    "airplane": "CR2"
  }
]
Atlas atlas-hbfz6-shard-0 [primary] student858>

```

2.9 How many documents were found based on the query conditions in question 2.8?

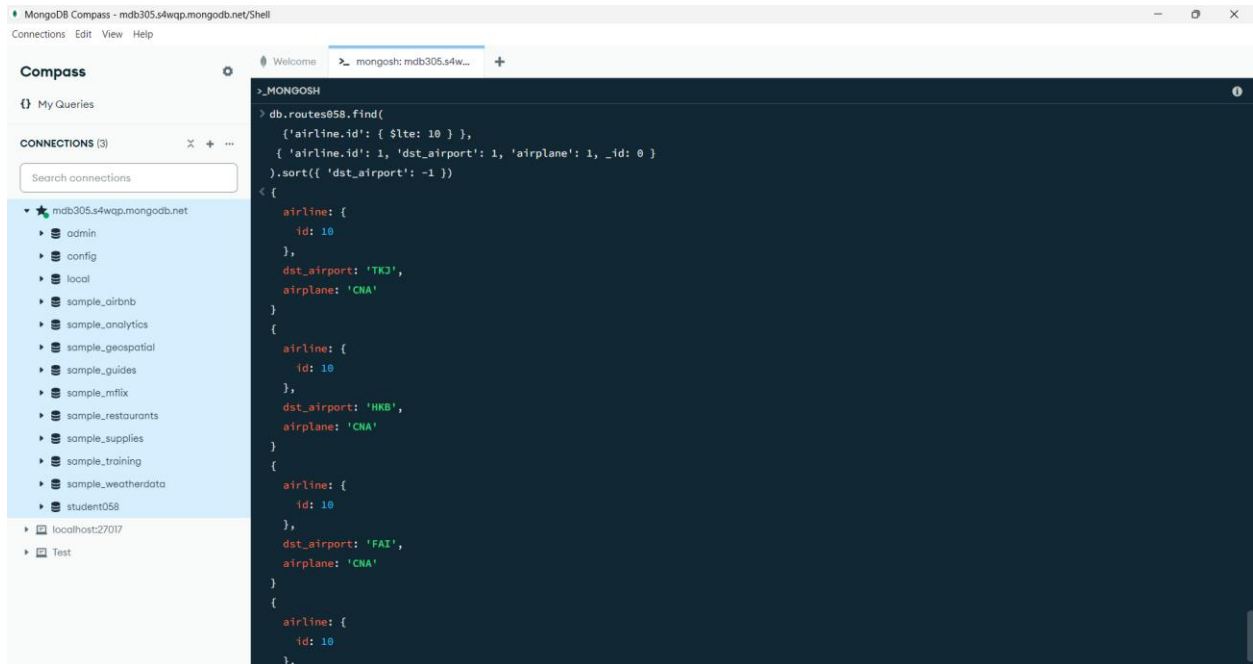
The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS' panel lists several connections, with 'mongo305.s4egp.mongodb.net' selected. The main panel displays a MongoDB shell session with the following commands and results:

```

> MONGOOSH
> db.routes858.countDocuments({airplane:"CR2"},{"airline":1, "airline.name":1})
< 296
Atlas atlas-hbfz6-shard-0 [primary] student858>

```

2.10 Display the airline id, dst_airport and airplane fields of all documents in the “routesXXX” collection that the airline id is less than or equal to 10. Sorting the result in descending order by the dst_airport field. Please show the results of each command.



Task 3: Design a data model from relational tables into the “Denormalized Data Model” in MongoDB.**Evaluate the structure and data of the BRANCH and STAFF tables:**

```
CREATE TABLE branch
```

```
(branchNo char(5) PRIMARY KEY,
```

```
street varchar(35),
```

```
city varchar(10),
```

```
postcode varchar(10)
```

```
);
```

```
INSERT INTO branch VALUES('B005','22 Deer Rd','London','SW1 4EH');
```

```
INSERT INTO branch VALUES('B003','163 Main St', 'Glasgow','G11 9QX');
```

```
CREATE TABLE staff
```

```
(staffNo char(5) PRIMARY KEY,
```

```
fName varchar(10),
```

```
lName varchar(10),
```

```
position varchar(10),
```

```
gender char(1),
```

```
DOB date,
```

```
salary int,
```

```
branchNo char(5)
```

```
);
```

```
INSERT INTO staff VALUES('SL21','John','White','Manager','M','1965-10-01',30000,'B005');
```

```
INSERT INTO staff VALUES('SG37','Ann','Beech','Assistant','F','1980-11-10',12000,'B003');
```

```
INSERT INTO staff VALUES('SG14','David','Ford','Supervisor','M','1978-03-24',18000,'B003');
```

```
INSERT INTO staff VALUES('SG5','Susan','Brand','Manager','F','1960-06-03',24000,'B003');
```

3.1 Convert two tables into an embedded data and write the data into the JSON format and save a file named "staffXXX.json". XXX is the last 3 digits of your student id.

*** Please submit a JSON file into LEB2 system. ***

Requirements: The primary data is the staff information and please omit the _id field in JSON file.

Hint: Do not write a JSON schema.

3.2 Create a new collection named "staffXXX" and try to import data from the "staffXXX.json" file created in Question 3.1 into the new collection and verify the result.

