

Southern New Hampshire University

CS-340: Assignment 2-1: Module 2 Journal

Prepared on: September 14, 2025

Prepared for: Prof. Jeff H. Sanford

Prepared by: Alexander Ahmann

Contents

1	Introduction	2
1.1	Brief description of the Figures	2
2	Problem Set	3
2.1	Task 1: Importing the <code>city</code> database and its respective <code>inspections</code> collection	3
2.2	Task 2: Confirming the status of the import	3
2.3	Task 3: Inserting an arbitrary document	4
2.4	Task 4: Working with MongoDB search queries	6
2.5	Task 5: Updating an arbitrary document	7
2.6	Task 6: Updating many documents, and removing an arbitrary document	9
3	Summary	10
A	Appendix: Figures depicting the results of tasks worked out	11
A.1	Figure 1: Importing the <code>inspections</code> collection	11
A.2	Figures 2, 3 & 4: Examples of <code>find*()</code> queries against the <code>inspections</code> collection	12
A.3	Figure 5: Insertion of a new document	15
A.4	Figures 6, 7, 8, & 9: Sub-Tasks involved in data modelling . .	16
A.5	Figure 10: Updating a single document	20
A.6	Figures 11, 12 & 13: Sub-Tasks relating to mass updating and removal of a specific document	21

1 Introduction

This journal serves as documentation for the tasks that I was assigned in module 2's assignment regarding further data *extraction, loading, and transformation* with the MongoDB non-relational database system (CS-340, n.d.). This assignment had students work with hypothetical “city inspections” data — which is presumably some kind of database for keeping track of unethical or illegal conduct of various for-profit institutions. Concepts in non-relational databases, along with their implementation in the MongoDB service, covered include: *insertion* of new documents, further development of *find* queries, *updating* existing documents at an individual and mass scale, and *removal* of specific documents.

1.1 Brief description of the Figures

To demonstrate that the tasks were carried out, I have embedded a number of figures depicting screenshots that show the results of executing the given tasks on an online **mongosh** shell. In the next section, I elaborate more on what was specifically done, but the following are a brief description of the figures:

- Figure 1 depicts a solution where I make use of the **mongoimport** utility to take a JSON dataset and put them into the *MongoDB* instance.
- Figures 2, 3, and 4 depicts the results to basic data modelling with the **find*()** of queries.
- Figure 5 depicts the use of **insertOne** queries to insert a new document with specific fields and their respective values.
- Figures 6, 7, 8, and 9 depicts more data modelling with the **find*()** class of queries, along with the **distinct** query.
- Figure 10 depicts a solution where I use the **updateOne()** query to update a document that meets a specific filter conditional.
- Figures 11, 12, and 13 depict the results of using **updateMany** to update multiple **inspections** documents, and the use of the **deleteOne** query to remove a single document that meets a specific filter conditional.

2 Problem Set

2.1 Task 1: Importing the city database and its respective inspections collection

“Access the dataset for this assignment, `city_inspections.json`, in the `datasets` directory in your Codio environment. Using the `mongoimport` tool and documents found in the `city_inspections.json` file, load the database `city` into the `inspections` collection. Complete this task by typing the following commands in the Linux terminal to perform the import in the right directory.” —CS-340 (n.d., §1 of prompt)

The given solution is straightforward: I logged into the *Codio* database, and issued the following command to import the database into the *MongoDB* service:

```
mongoimport --db city --collection inspections
--drop ./city_inspections.json
```

The given command will import the `./city_inspections.json` dataset into the `city` database, and specifically store each entry as a document in the `inspections` collection. Figure 1 depicts a screenshot that demonstrates the execution of this task.

2.2 Task 2: Confirming the status of the import

“Verify your load by “using” the “city” database, and issuing the following queries in the mongo shell:

```
1. db.inspections.find({
    "id" : "10021-2015-ENFO"
})

2. db.inspections.find(
    {"result":"Out of Business"},
    {"business_name":1
}).limit(10)
```

Provide screenshots of the results as evidence.” —CS-340 (n.d., §2 of prompt)

Like with the first task, this one was pretty straightforward. I simply entered in the given commands into the `mongosh` interface. Figures 2, 3, and 4 depicts multiple screenshots that demonstrates the execution and results of this task. Specifically, figure 2 depicts the results of the first executed command, and figures 3 and 4 depicts the results of the second executed command.

2.3 Task 3: Inserting an arbitrary document

“Using the appropriate commands in the mongo shell, insert a document to the database named `city` within the collection named `inspections`. Use the following key-value pairs [as depicted in Table 1] as data for your document. Be sure to insert the address as a subdocument and use the JavaScript function `Date()` for ‘Today’s date.’ Verify your database creation and insertion using the `findOne()` function in the mongo shell. Provide a screenshot as evidence.” —CS-340 (n.d., §3 of the prompt)

Key	Value
id	“20032-2020-ACME”
certificate_number	99988888
business_name	“ACME Explosives”
date	Today’s date
result	“Business Padlocked”
sector	“Explosive Retail Dealer-999”
address	number -> 1721 street -> Boom Road city -> BRONX zip -> 10463

Table 1: Hypothetical data to be inserted as a document into `inspections`.

The data to be inserted is shown in table 1. The exact MongoDB database query that I used to insert the data as depicted in this table is:¹

¹See Giamas (2022, §3.2: *CRUD using the shell* for constructing queries regarding inserting new documents). Also consult the following resource: MongoDB Docs. (n.d.). `db.collection.insertOne()`. Last retrieved on Sept. 14, 2025: <https://www.mongodb.com/docs/manual/reference/method/db.collection.insertOne/>

```

db.inspections.insertOne({
    _id:"20032-2020-ACME",
    certificate_number:9998888,
    business_name:"ACME Explosives",
    date:new Date(),
    result:"Business Padlocked",
    sector:"Explosive Retail Dealer-999",
    address:{
        number:1721,
        street:"Boom Road",
        city:"BRONX",
        zip:10463
    }
})

```

Some “interesting” fields that constitute the document to be inserted are `_id`, `date`, and `address`. The `_id` is a unique string that identifies the document in the collection, the date is something of a “dynamic wildcard” that is dependent on the state of the database system—specifically the current date and time, and the address is another dictionary within the current document, which itself is expressed in terms of a dictionary.

After executing the insert query statement, the command line returned the following bit of information to confirm that a transaction was completed:

```
{ acknowledged: true, insertedId: '20032-2020-ACME' }
```

I then confirmed that the transaction completed successfully with the following query statement:²

²See Giamas (2022, §2-4) as a reference to data modelling and constructing `find*()` non-relational queries. Also see the following resources:

- Factor, P. (Feb. 17, 2020). *MongoDB find Method: Introduction & Examples*. Studio 3T Knowledge Base. Last retrieved on Sept. 14, 2025 from: <https://studio3t.com/knowledge-base/articles/mongodb-find-method/>
- MongoDB Docs (n.d.). *db.collection.find()*. Last retrieved on Sept. 14, 2025 from: <https://www.mongodb.com/docs/manual/reference/method/db.collection.find/>
- MongoDB Docs (n.d.). *db.collection.findOne()*. Last retrieved on Sept. 14, 2025 from: <https://www.mongodb.com/docs/manual/reference/method/db.collection.findOne/>

```
db.inspections.findOne({
  _id:"20032-2020-ACME"
})
```

Figure 5 depicts a screenshot demonstrating the execution of this task, and its respective results.

2.4 Task 4: Working with MongoDB search queries

This task is composed of three smaller tasks that centre on the theme of data modelling and the `find*()` class of MongoDB queries. The first one concerns itself with counting distinct entities: “[w]hat is the distinct list of inspection ‘sector’ in the current inspections collection? How many are in the list? Do not count by hand” (CS-340, n.d., §4.1). To answer this, I began with the following MongoDB query:³

```
db.inspections.distinct("sector")
```

This was just to get an idea of what distinct sectors look like. I then got the exact count of the distinct sectors by appending a `.length` to the previous query:

```
db.inspections.distinct("sector").length
```

This resulted in 87 distinct kinds of sectors. Figure 6 depicts a screenshot which demonstrates the execution of the procedure.

The next subtask concerns itself with data types: “What is the difference in the date data type for the business named ‘AUSTIN 2012’ versus your business document insertion of ‘Acme Explosives?’” (CS-340, n.d., §4.2). To answer this question, I used the `typeof` operator in MongoDB queries for the AUSTIN 2012 and ACME Explosives databases:

³See Giamas (2022, §7) for working with `distinct` values. Also, see the following resources:

- Geeks4Geeks (c.a. Jul. 23, 2025) *MongoDB - Distinct() Method*. Last Retrieved on Sept. 14, 2025 from: <https://www.geeksforgeeks.org/mongodb/mongodb-distinct-method/>
- MongoDB Docs (n.d.). *db.collection.distinct()*. Last Retrieved on Sept. 14, 2025 from: <https://www.mongodb.com/docs/manual/reference/method/db.collection.distinct/>

```

typeof db.inspections.findOne({
  business_name: "AUSTIN 2012"
}).date // for the date type of "AUSTIN 2012"

typeof db.inspections.findOne({
  business_name: "ACME Explosives"
}).date // for the date type of "ACME Explosives"

```

Regarding the **AUSTIN 2012** business, it returned a **string** data type for the data field, and regarding the **ACME Explosives** business, it returned a **object** data type for the date field. From this, I can infer that MongoDB type databases are more flexible in how they process fields.⁴ Figure 7 depicts a screenshot that demonstrates the results after executing the task.

The final subtask concerns itself with **distinct** queries that return data conditional to some criteria: “[h]ow many businesses have a ‘Violation Issued?’” To answer this question, I used the following **distinct** query to return the count documents conditional on **business_name** having a value of **Violation Issued**:

```

db.inspections.distinct(
  "business_name",
  { result:"Violation Issued"}
).length

```

The result was 10602 distinct records, see Figure 8 for a sample of the records return, and Figure 9 for the count of total distinct records that meet the criteria.

2.5 Task 5: Updating an arbitrary document

“Using the appropriate command in the mongo shell, update the document with the ID 20032-2020-ACME in the collection **inspections** in the database **city** with the information [in Table 2.5]. Verify your database update using the appropriate **find()** function in the mongo shell.” —CS-340 (n.d., §5)

⁴In previous experience working with the JSON file format, without the use of MongoDB, I noticed that, when iterating through a JSON flat file, not all entries have a certain field. For example, when using a Python for-loop to iterate through a **.jsonl** file and searching for a hypothetical field “field1”, some entries did not have a “field1”, causing earlier versions of my scripts to crash. I overcame this problem with an exception handler.

Key	Value
business_name	“New ACME Flowers”
result	“Business Re-opened”
comments	“Flowers after the explosion”

Table 2: Hypothetical key-value pairs by which to update the “20032-2020-ACME” document in the `inspections` table.

To answer this question, I had to use the `updateOne` query which is conditional to the `_id` of the company that I am interested in making changes to.⁵ The exact query that I used to update the record with the `{ "_id" : "20032-2020-ACME" }` key-value pair is:

```
db.inspections.updateOne( { "_id":"20032-2020-ACME" }, {
  $set: {
    "business_name":"New ACME Flowers",
    "result":"Business Re-opened",
    "comments":"Flowers after the explosion"
  }
})
```

I then used the following query to confirm that the update procedure was successful:

```
db.findOne( {"_id": "20032-2020-ACME"} )
```

Figure 10 depicts a screenshot which demonstrates this task’s completion and its results.

⁵See Giamas (2022, §3.2) for working with `update*`() `delete*`() classes of queries. Also see the following resources:

- Geeks4Geeks (Jul. 23, 2025). *MongoDB - updateOne() Method*. Last Retrieved on Sept. 14, 2025 from: <https://www.geeksforgeeks.org/mongodb/mongodb-updateone-method-db-collection-updateone/>
- MongoDB Docs (n.d.). *db.collection.updateOne()*. Last retrieved on Sept. 14, 2025 from: <https://www.mongodb.com/docs/manual/reference/method/db.collection.updateOne/>
- MongoDB Docs (n.d.). *Delete Documents*. Last retrieved on Sept. 14, 2025 from: <https://www.mongodb.com/docs/drivers/node/current/crud/delete/>

2.6 Task 6: Updating many documents, and removing an arbitrary document

This task is divided up into two subtasks: one involving updating many documents in the `inspections` collection, and the other is removing a specific document that meets a filter criteria. The first subtask states to: “[u]pdate all the documents that contain the key-value pair { "city": "ROSEDALE" } in the address subdocument by changing the zip code in the address subdocument to 76114” (CS-340, n.d., §6.1). To accomplish this task, I used an `updateMany` query.⁶ Specifically, the exact query that I used to update all ZIP codes in documents that contain the { "address.city" : "ROSEDALE" } entry is:

```
db.inspections.updateMany({
  "address.city": "ROSEDALE"
},
{
  $set: {
    "address.zip": 76114
  }
})
```

To verify the successfulness of this transaction, I then use the following query:

```
db.inspections.findOne( {"address.zip": 76114} )
```

Figures 11 and 12 depicts screenshots showing the successfulness of the `updateMany` transaction, and the results of the `findOne` query.

The final subtask, and indeed the final job of this assignment, is to remove an arbitrary document from the `inspections` collection: “[r]emove the first document with the key-value pair "result" : "Violation Issued.” (CS-340, n.d., §6.2). To accomplish this task, I used a `deleteOne()` query, and then verified its successfulness with a `find()` queries before and after the execution. Specifically, I used the following `deleteOne` query:

```
db.inspections.deleteOne({"result": "Violation Issued"})
```

Figure 13 depicts screenshots that demonstrates the results after executing these queries.

⁶See the following resource when referencing `updateMany` queries: MongoDB Docs (n.d.). *db.collection.updateMany()*. Last retrieved on Sept. 14, 2025 from: <https://www.mongodb.com/docs/manual/reference/method/db.collection.updateMany/>

3 Summary

This assignment acts to demonstrate this student's competence in the extracting, loading, and transformation of data with the `mongoimport` utility, and specific use cases for *Create, Read, Update, and Remove* features of the MongoDB database management system. Regarding CRUD features:

- I used the `find*()` class of queries, and the `distinct()` query, to return documents that meet some filter conditional, and to remove redundancies.
- I used the `insertOne()` query to insert new documents into a MongoDB collection.
- I used the `update*()` class of queries to update values for specific keys across a specific document, or even many document, based on some filter conditional.
- I used the `deleteOne()` query to remove a single document that meets a filter conditional.

The bulk of this assignment was more focused on CRUD features of MongoDB, as opposed to ETL methods and utilities. Nonetheless, the importance of knowing how to use the `mongoimport` utility is emphasised in the first task.

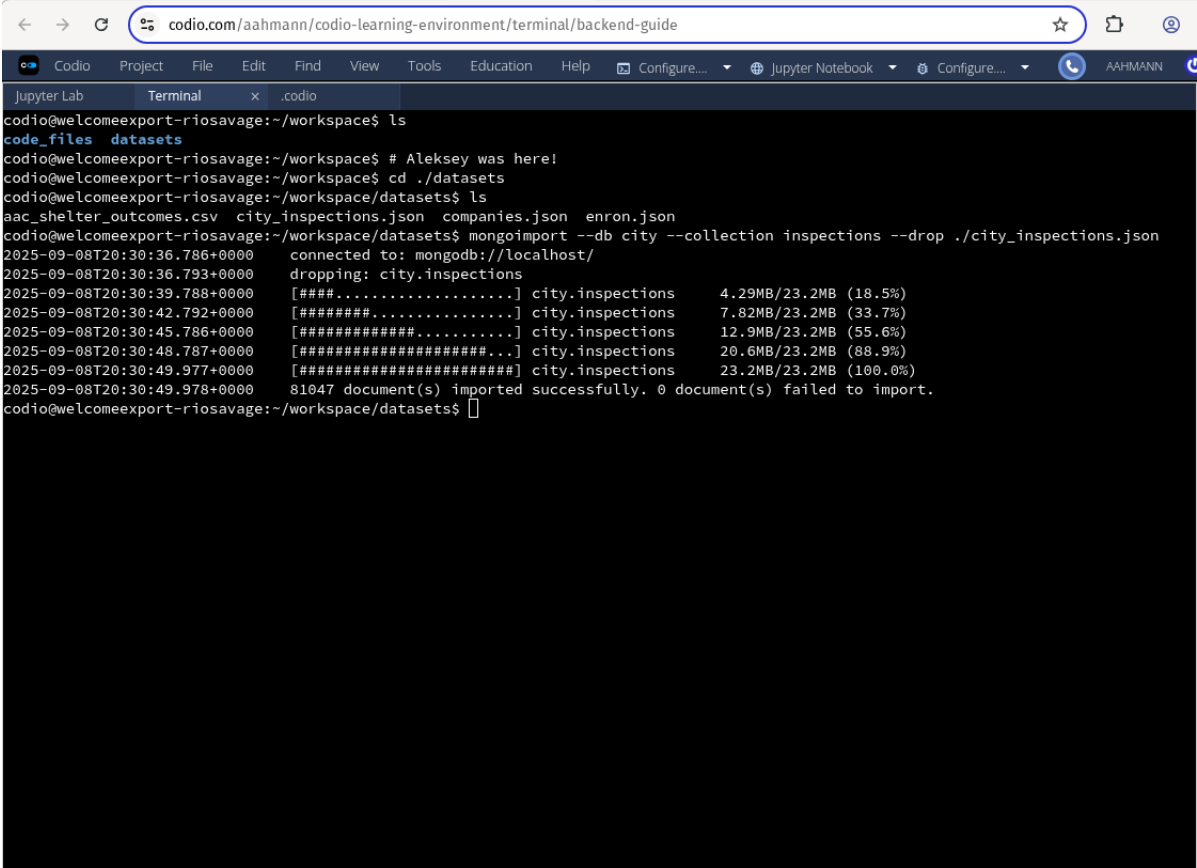
References

CS-340 (n.d.). *CS-340 Module Two Assignment Guidelines and Rubric*.

Giamas, A. (2022). *Mastering MongoDB 6.x: Expert Techniques to Run High-volume and Fault-tolerant Database Solutions Using MongoDB 6.x*. Birmingham, UK: Packt Publishing, 2022. Last retrieved on Sept. 12, 2025 from: <https://research.ebsco.com/linkprocessor/plink?id=a5bcc20e-3306-36b5-ad4f-0d0bd1f1567e>

A Appendix: Figures depicting the results of tasks worked out

A.1 Figure 1: Importing the inspections collection

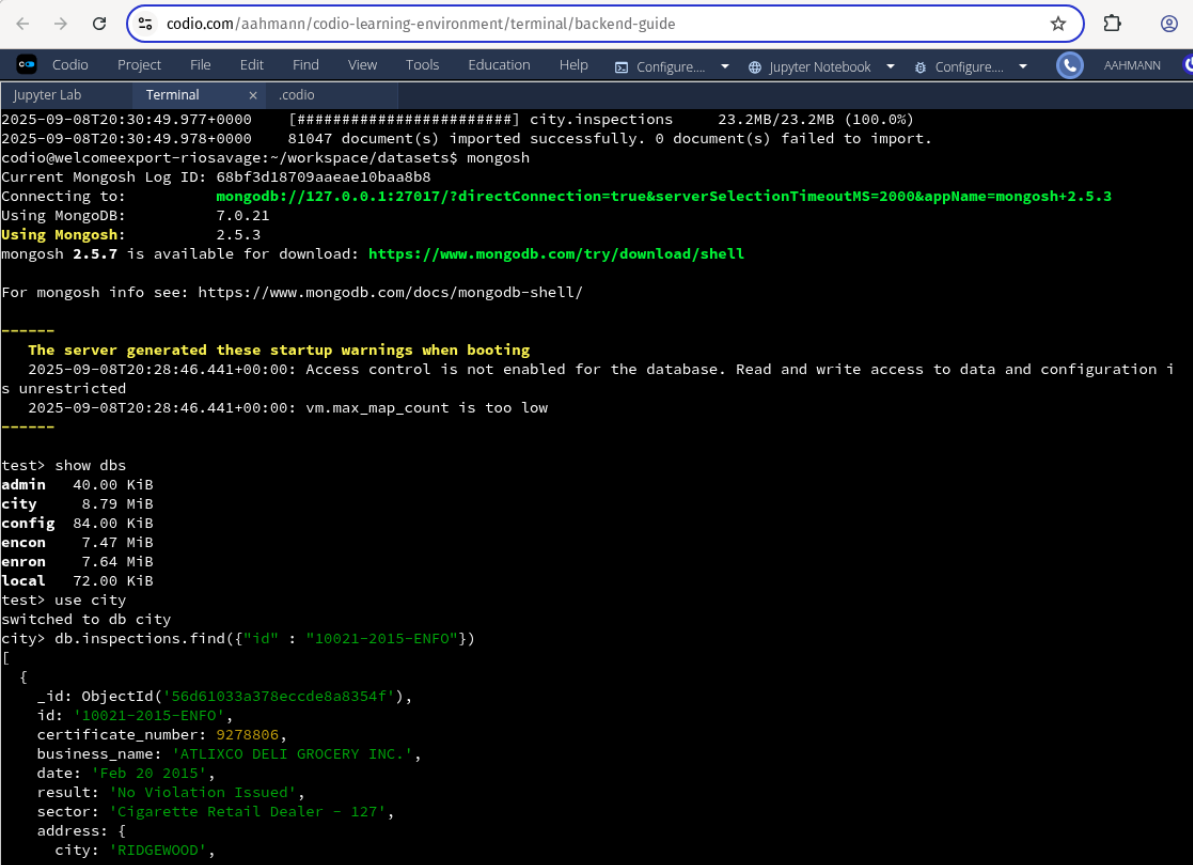


```
codio.com/aahmann/codio-learning-environment/terminal/backend-guide
Codio Project File Edit Find View Tools Education Help Configure... Jupyter Notebook Configure... AAHMANN

Jupyter Lab Terminal x .codio
code_files datasets
codio@welcomeexport-riosavage:~/workspace$ ls
codio@welcomeexport-riosavage:~/workspace$ # Aleksey was here!
codio@welcomeexport-riosavage:~/workspace$ cd ./datasets
codio@welcomeexport-riosavage:~/workspace/datasets$ ls
aac_shelter_outcomes.csv city_inspections.json companies.json enron.json
codio@welcomeexport-riosavage:~/workspace/datasets$ mongoimport --db city --collection inspections --drop ./city_inspections.json
2025-09-08T20:30:36.786+0000 connected to: mongodb://localhost/
2025-09-08T20:30:36.793+0000 dropping: city.inspections
2025-09-08T20:30:39.788+0000 [####.....] city.inspections 4.29MB/23.2MB (18.5%)
2025-09-08T20:30:42.792+0000 [#####.....] city.inspections 7.82MB/23.2MB (33.7%)
2025-09-08T20:30:45.786+0000 [#####.....] city.inspections 12.9MB/23.2MB (55.6%)
2025-09-08T20:30:48.787+0000 [#####.....] city.inspections 20.6MB/23.2MB (88.9%)
2025-09-08T20:30:49.977+0000 [#####.....] city.inspections 23.2MB/23.2MB (100.0%)
2025-09-08T20:30:49.978+0000 81047 document(s) imported successfully. 0 document(s) failed to import.
codio@welcomeexport-riosavage:~/workspace/datasets$
```

Figure 1: Importing the inspections collection.

A.2 Figures 2, 3 & 4: Examples of find*() queries against the inspections collection



```
codio.com/aahmann/codio-learning-environment/terminal/backend-guide

Jupyter Lab Terminal x .codio

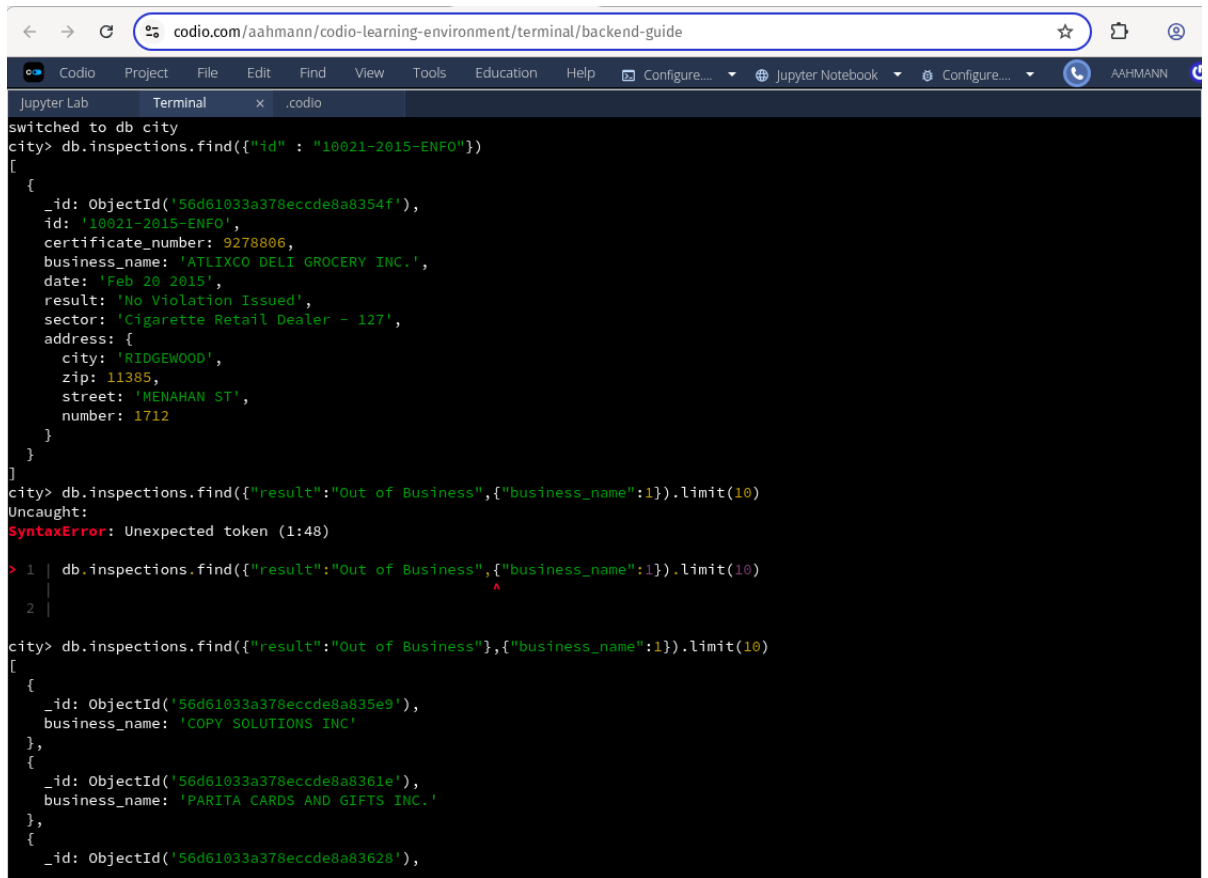
2025-09-08T20:30:49.977+0000 [#####] city.inspections 23.2MB/23.2MB (100.0%)
2025-09-08T20:30:49.978+0000 81047 document(s) imported successfully. 0 document(s) failed to import.
codio@welcomeexport-riosavage:~/workspace/datasets$ mongosh
Current Mongosh Log ID: 68bf3d18709aaeae10baa8b8
Connecting to: mongod://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.3
Using MongoDB: 7.0.21
Using Mongosh: 2.5.3
mongosh 2.5.7 is available for download: https://www.mongodb.com/try/download/shell

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

-----
The server generated these startup warnings when booting
  2025-09-08T20:28:46.441+00:00: Access control is not enabled for the database. Read and write access to data and configuration i
s unrestricted
  2025-09-08T20:28:46.441+00:00: vm.max_map_count is too low
-----

test> show dbs
admin    40.00 KiB
city     8.79 MiB
config   84.00 KiB
encon    7.47 MiB
enron    7.64 MiB
local    72.00 KiB
test> use city
switched to db city
city> db.inspections.find({"id": "10021-2015-ENFO"})
[
  {
    _id: ObjectId('56d61033a378eccde8a8354f'),
    id: '10021-2015-ENFO',
    certificate_number: 9278806,
    business_name: 'ATLIXCO DELI GROCERY INC.',
    date: 'Feb 20 2015',
    result: 'No Violation Issued',
    sector: 'Cigarette Retail Dealer - 127',
    address: {
      city: 'RIDGEWOOD',
```

Figure 2: Results of `db.inspections.find({"id": "10021-2015-ENFO"})`

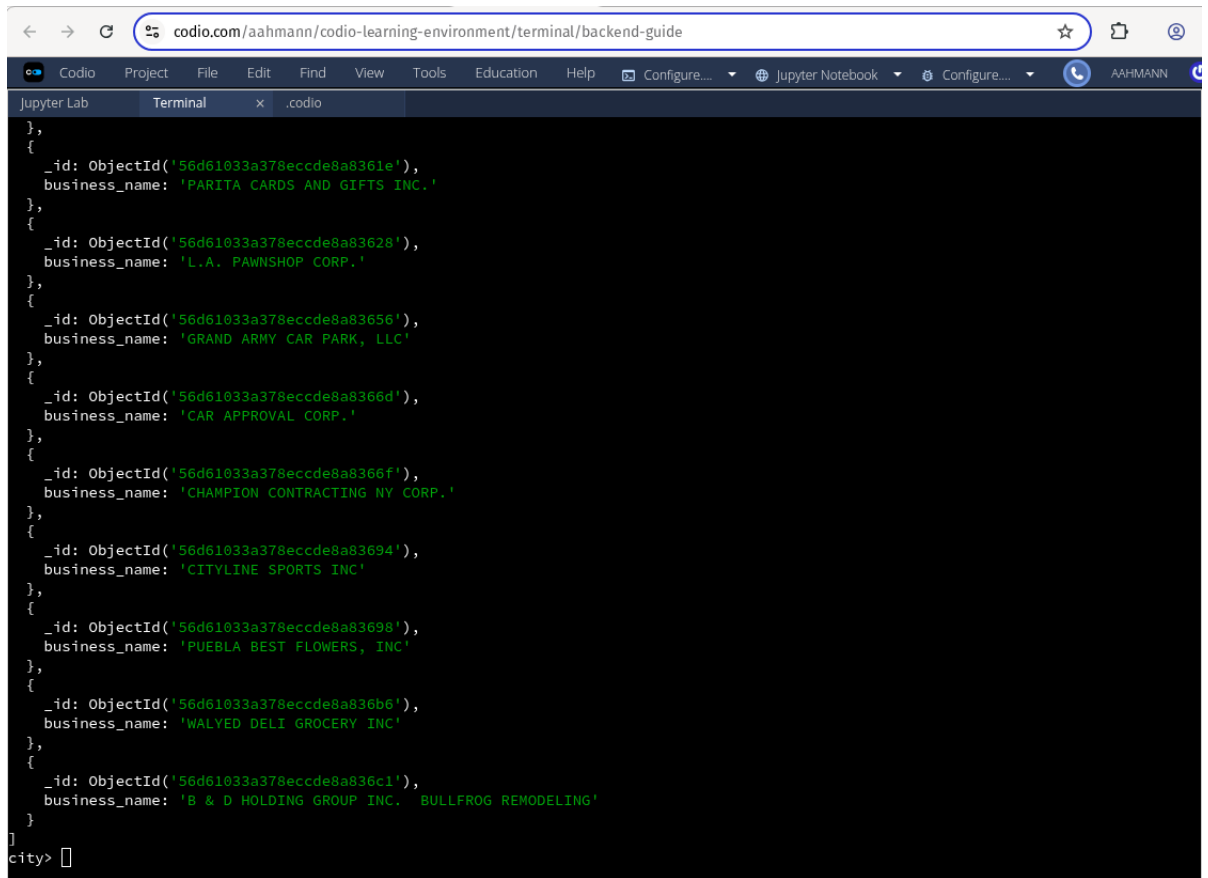


```
switched to db city
city> db.inspections.find({"id" : "10021-2015-ENFO"})
[
  {
    _id: ObjectId('56d61033a378eccde8a8354f'),
    id: '10021-2015-ENFO',
    certificate_number: 9278806,
    business_name: 'ATLIXCO DELI GROCERY INC.',
    date: 'Feb 20 2015',
    result: 'No Violation Issued',
    sector: 'Cigarette Retail Dealer - 127',
    address: {
      city: 'RIDGEWOOD',
      zip: 11385,
      street: 'MENAHAN ST',
      number: 1712
    }
  }
]
city> db.inspections.find({"result":"Out of Business"},{"business_name":1}).limit(10)
Uncaught:
SyntaxError: Unexpected token (1:48)

> 1 | db.inspections.find({"result":"Out of Business"},{"business_name":1}).limit(10)
    |                                                                    ^
    2 |

city> db.inspections.find({"result":"Out of Business"},{"business_name":1}).limit(10)
[
  {
    _id: ObjectId('56d61033a378eccde8a835e9'),
    business_name: 'COPY SOLUTIONS INC'
  },
  {
    _id: ObjectId('56d61033a378eccde8a8361e'),
    business_name: 'PARITA CARDS AND GIFTS INC.'
  },
  {
    _id: ObjectId('56d61033a378eccde8a83628'),
```

Figure 3: Results of `db.inspections.find({"result":"Out of Business"}, {"business_name":1}).limit(10)` (with mistakes, for the sake of full disclosure).

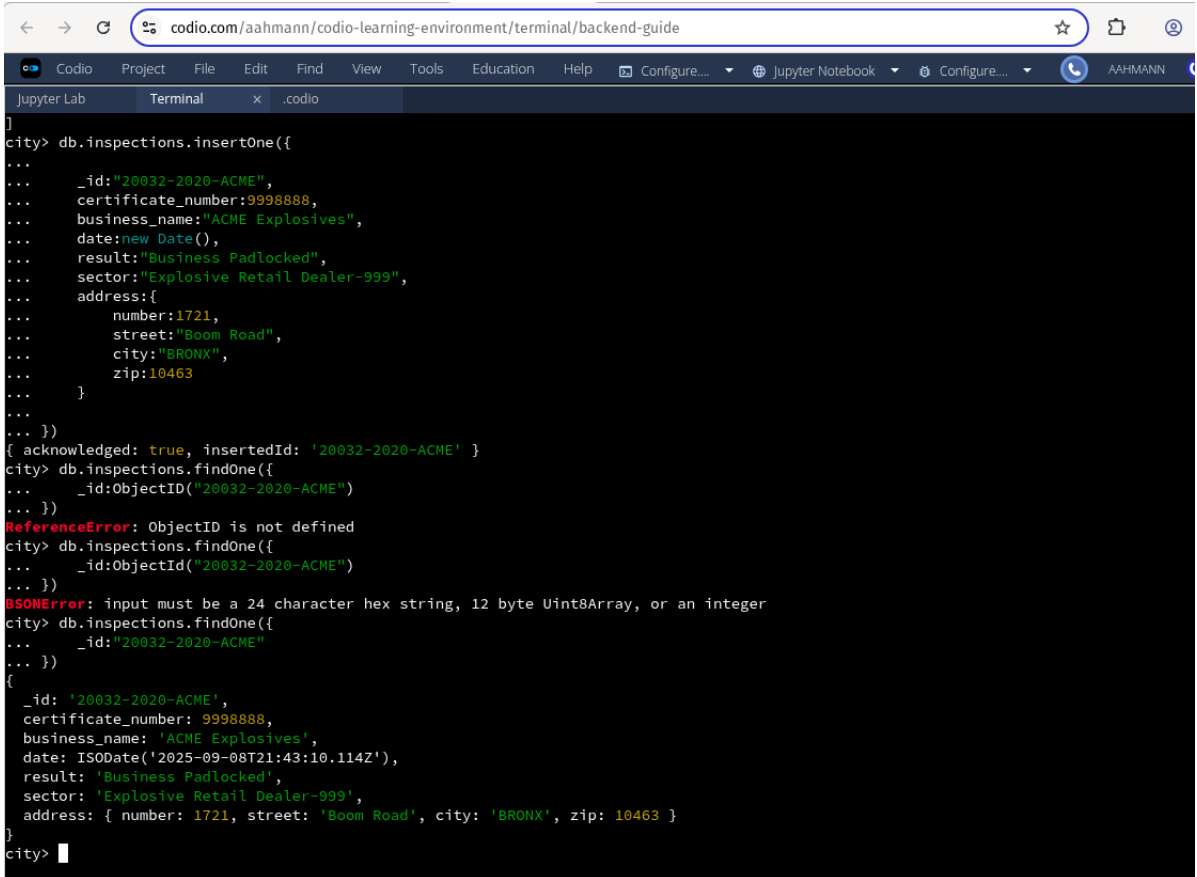


The screenshot shows a web browser window with the address bar displaying `codio.com/aahmann/codio-learning-environment/terminal/backend-guide`. The browser's top bar includes tabs for 'Codio', 'Project', 'File', 'Edit', 'Find', 'View', 'Tools', 'Education', 'Help', 'Configure...', 'Jupyter Notebook', and another 'Configure...' tab. The user's name 'AAHMANN' is visible in the top right corner. The main content area is a terminal window with a dark background. It displays the output of a MongoDB query, showing a list of business records. Each record is a JSON object with an `_id` (ObjectId) and a `business_name`. The records are: 'PARITA CARDS AND GIFTS INC.', 'L.A. PAWNSHOP CORP.', 'GRAND ARMY CAR PARK, LLC', 'CAR APPROVAL CORP.', 'CHAMPION CONTRACTING NY CORP.', 'CITYLINE SPORTS INC', 'PUEBLA BEST FLOWERS, INC', 'WALYED DELI GROCERY INC', and 'B & D HOLDING GROUP INC. BULLFROG REMODELING'. The terminal prompt `city>` is visible at the bottom left.

```
},
{
  _id: ObjectId('56d61033a378eccde8a8361e'),
  business_name: 'PARITA CARDS AND GIFTS INC.'
},
{
  _id: ObjectId('56d61033a378eccde8a83628'),
  business_name: 'L.A. PAWNSHOP CORP.'
},
{
  _id: ObjectId('56d61033a378eccde8a83656'),
  business_name: 'GRAND ARMY CAR PARK, LLC'
},
{
  _id: ObjectId('56d61033a378eccde8a8366d'),
  business_name: 'CAR APPROVAL CORP.'
},
{
  _id: ObjectId('56d61033a378eccde8a8366f'),
  business_name: 'CHAMPION CONTRACTING NY CORP.'
},
{
  _id: ObjectId('56d61033a378eccde8a83694'),
  business_name: 'CITYLINE SPORTS INC'
},
{
  _id: ObjectId('56d61033a378eccde8a83698'),
  business_name: 'PUEBLA BEST FLOWERS, INC'
},
{
  _id: ObjectId('56d61033a378eccde8a836b6'),
  business_name: 'WALYED DELI GROCERY INC'
},
{
  _id: ObjectId('56d61033a378eccde8a836c1'),
  business_name: 'B & D HOLDING GROUP INC. BULLFROG REMODELING'
}
]
city>
```

Figure 4: Results of `db.inspections.find("result":"Out of Business", "business_name":1).limit(10)` (cont.)

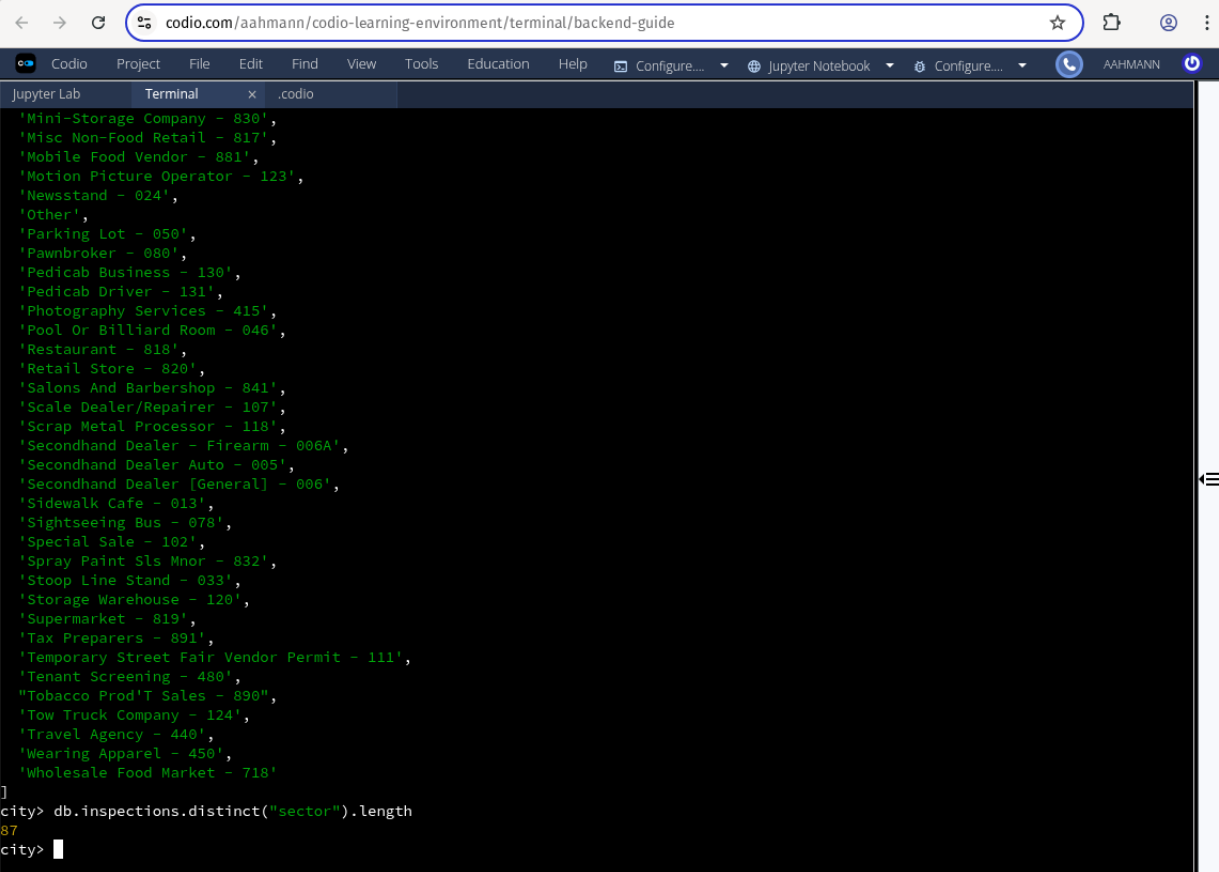
A.3 Figure 5: Insertion of a new document



```
city> db.inspections.insertOne({
...
  _id:"20032-2020-ACME",
  certificate_number:9998888,
  business_name:"ACME Explosives",
  date:new Date(),
  result:"Business Padlocked",
  sector:"Explosive Retail Dealer-999",
  address:{
    number:1721,
    street:"Boom Road",
    city:"BRONX",
    zip:10463
  }
...
})
{ acknowledged: true, insertedId: '20032-2020-ACME' }
city> db.inspections.findOne({
...
  _id:ObjectID("20032-2020-ACME")
...
})
ReferenceError: ObjectId is not defined
city> db.inspections.findOne({
...
  _id:ObjectID("20032-2020-ACME")
...
})
BSoneError: input must be a 24 character hex string, 12 byte Uint8Array, or an integer
city> db.inspections.findOne({
...
  _id:"20032-2020-ACME"
...
})
{
  _id: '20032-2020-ACME',
  certificate_number: 9998888,
  business_name: 'ACME Explosives',
  date: ISODate('2025-09-08T21:43:10.114Z'),
  result: 'Business Padlocked',
  sector: 'Explosive Retail Dealer-999',
  address: { number: 1721, street: 'Boom Road', city: 'BRONX', zip: 10463 }
}
city> 
```

Figure 5: Results of inserting a new document in accordance with Table 1 (with mistakes, for the sake of full disclosure).

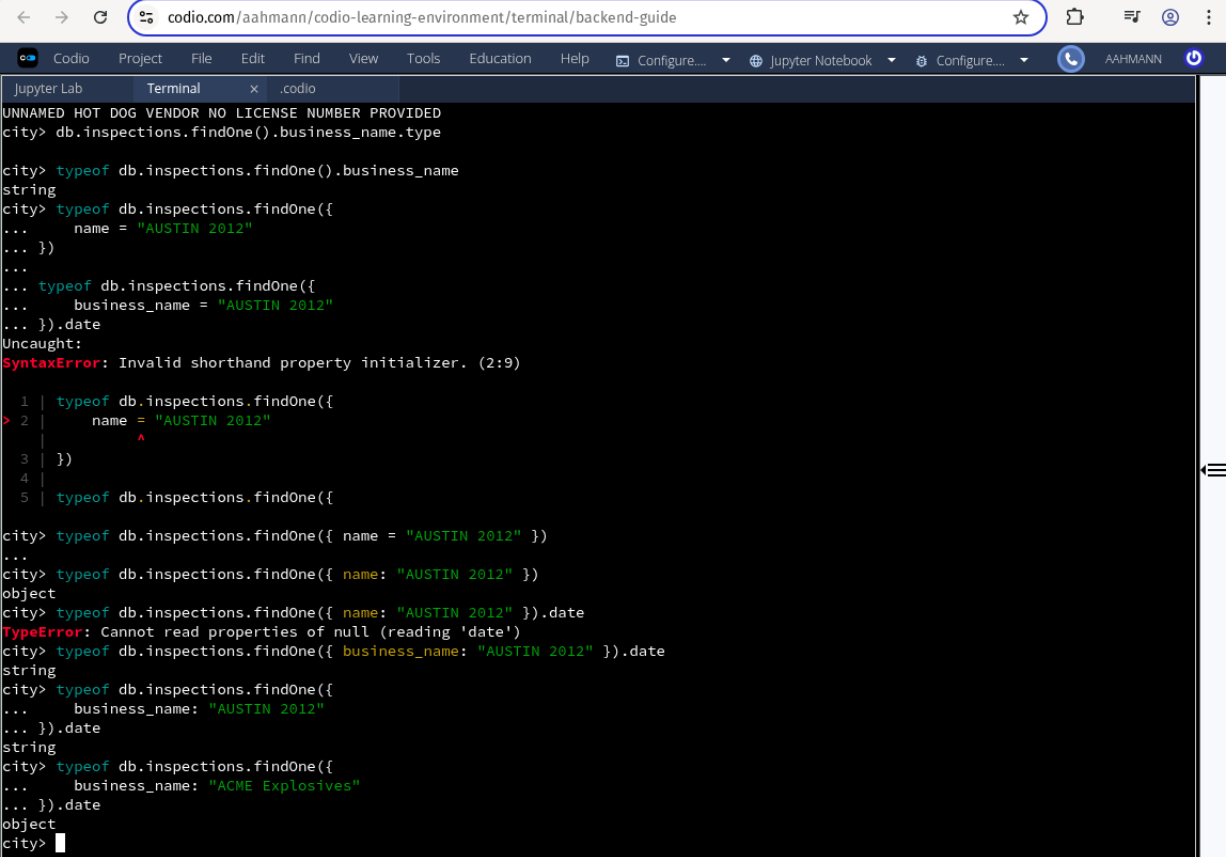
A.4 Figures 6, 7, 8, & 9: Sub-Tasks involved in data modelling



The screenshot shows a web browser window with the address bar displaying `codio.com/aahmann/codio-learning-environment/terminal/backend-guide`. The browser's top menu bar includes options like Codio, Project, File, Edit, Find, View, Tools, Education, Help, and Configuration. Below the menu, there are tabs for 'Jupyter Lab', 'Terminal', and '.codio'. The 'Terminal' tab is active, showing a list of sectors and their counts, followed by a MongoDB command and its output.

```
'Mini-Storage Company - 830',
'Misc Non-Food Retail - 817',
'Mobile Food Vendor - 881',
'Motion Picture Operator - 123',
'Newsstand - 024',
'Other',
'Parking Lot - 050',
'Pawnbroker - 080',
'Pedicab Business - 130',
'Pedicab Driver - 131',
'Photography Services - 415',
'Pool Or Billiard Room - 046',
'Restaurant - 818',
'Retail Store - 820',
'Salons And Barbershop - 841',
'Scale Dealer/Repairer - 107',
'Scrap Metal Processor - 118',
'Secondhand Dealer - Firearm - 006A',
'Secondhand Dealer Auto - 005',
'Secondhand Dealer [General] - 006',
'Sidewalk Cafe - 013',
'Sightseeing Bus - 078',
'Special Sale - 102',
'Spray Paint Sls Mnor - 832',
'Stoop Line Stand - 033',
'Storage Warehouse - 120',
'Supermarket - 819',
'Tax Preparers - 891',
'Temporary Street Fair Vendor Permit - 111',
'Tenant Screening - 480',
'Tobacco Prod'T Sales - 890',
'Tow Truck Company - 124',
'Travel Agency - 440',
'Wearing Apparel - 450',
'Wholesale Food Market - 718'
]
city> db.inspections.distinct("sector").length
87
city> 
```

Figure 6: Distinct count of documents with the `sector` field.



```
UNNAMED HOT DOG VENDOR NO LICENSE NUMBER PROVIDED
city> db.inspections.findOne().business_name.type
string
city> typeof db.inspections.findOne({
...   name = "AUSTIN 2012"
... })
...
... typeof db.inspections.findOne({
...   business_name = "AUSTIN 2012"
... }).date
Uncaught:
SyntaxError: Invalid shorthand property initializer. (2:9)

1 |   typeof db.inspections.findOne({
> 2 |     name = "AUSTIN 2012"
  |           ^
3 |   })
4 |
5 |   typeof db.inspections.findOne({

city> typeof db.inspections.findOne({ name = "AUSTIN 2012" })
...
city> typeof db.inspections.findOne({ name: "AUSTIN 2012" })
object
city> typeof db.inspections.findOne({ name: "AUSTIN 2012" }).date
TypeError: Cannot read properties of null (reading 'date')
city> typeof db.inspections.findOne({ business_name: "AUSTIN 2012" }).date
string
city> typeof db.inspections.findOne({
...   business_name: "AUSTIN 2012"
... }).date
string
city> typeof db.inspections.findOne({
...   business_name: "ACME Explosives"
... }).date
object
city>
```

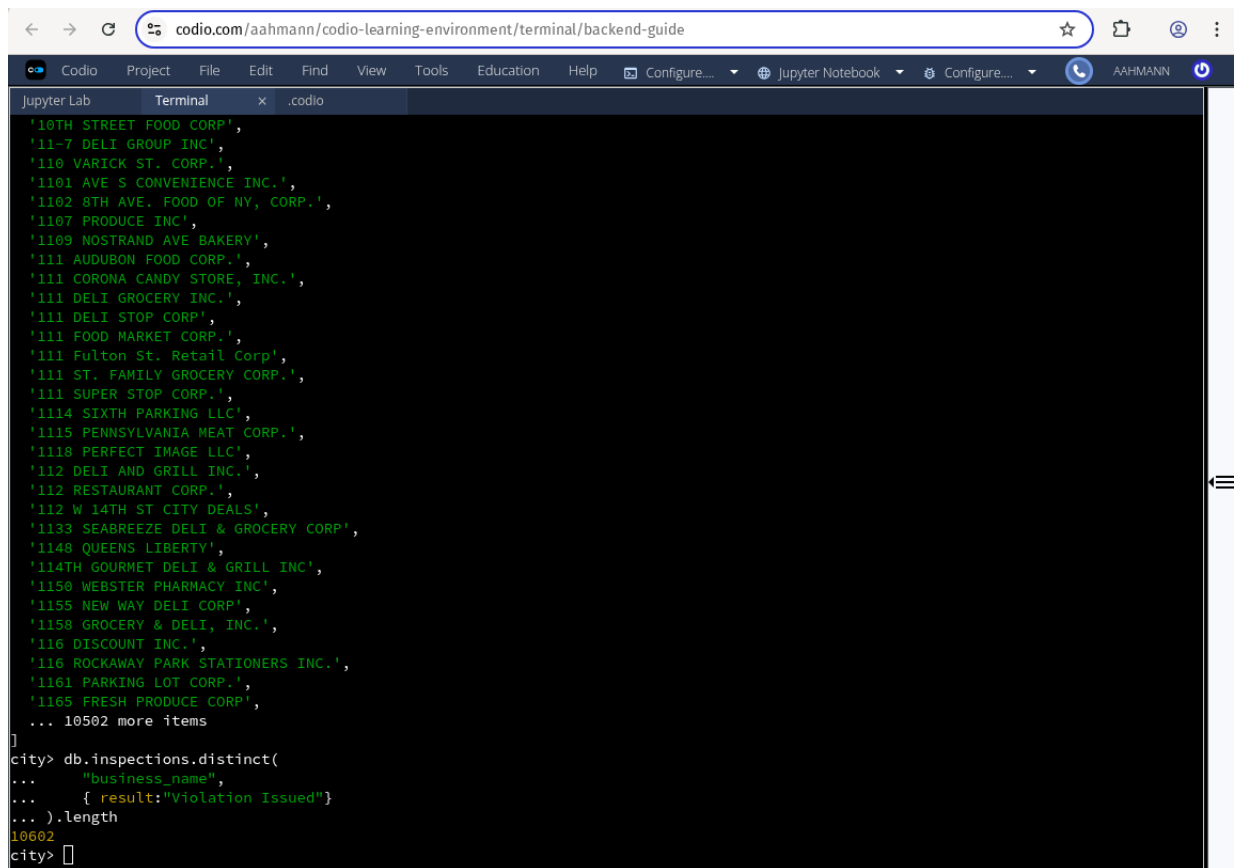
Figure 7: Comparison of the `typeof` `_id` fields for “AUSTIN 2012” and “ACME Explosive” (with mistakes, for the sake of full disclosure).

```
codio.com/aahmann/codio-learning-environment/terminal/backend-guide

Jupyter Lab Terminal x .codio

{
  _id: ObjectId('56d61033a378eccde8a83657'),
  id: '10101-2015-ENFO',
  certificate_number: 5377698,
  business_name: 'MD HOSSAIN MIA',
  date: 'Mar 19 2015',
  result: 'Violation Issued',
  sector: 'Mobile Food Vendor - 881',
  address: { city: 'BROOKLYN', zip: 11218, street: 'CATON AVE', number: 110 }
},
{
  _id: ObjectId('56d61033a378eccde8a83659'),
  id: '10242-2015-ENFO',
  certificate_number: 5377366,
  business_name: 'FORBES, DUDLEY',
  date: 'Apr 1 2015',
  result: 'Violation Issued',
  sector: 'Garage - 049',
  address: { city: 'BROOKLYN', zip: 11208, street: 'EUCLID AVE', number: 447 }
},
{
  _id: ObjectId('56d61033a378eccde8a83655'),
  id: '10060-2015-ENFO',
  certificate_number: 5377483,
  business_name: 'WALGREEN EASTERN CO INC',
  date: 'Apr 3 2015',
  result: 'Violation Issued',
  sector: 'Drug Store Retail - 810',
  address: {
    city: 'BROOKLYN',
    zip: 11208,
    street: 'LIBERTY AVE',
    number: 1242
  }
}
]
Type "it" for more
city> db.inspections.find({ result: "Violation Issued" }).count()
13823
city>
```

Figure 8: Sample of businesses with “Violations Found”.



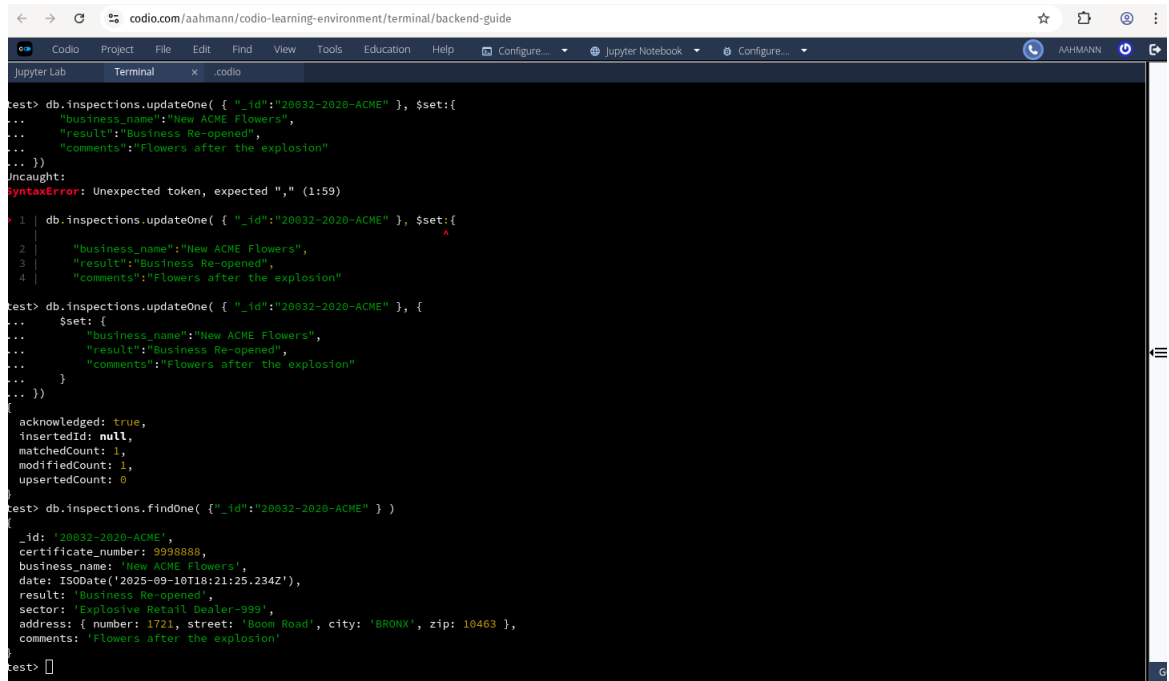
```
codio.com/aahmann/codio-learning-environment/terminal/backend-guide

Jupyter Lab  Terminal  x  .codio

'10TH STREET FOOD CORP',
'11-7 DELI GROUP INC',
'110 VARICK ST. CORP.',
'1101 AVE S CONVENIENCE INC.',
'1102 8TH AVE. FOOD OF NY, CORP.',
'1107 PRODUCE INC',
'1109 NOSTRAND AVE BAKERY',
'111 AUDUBON FOOD CORP.',
'111 CORONA CANDY STORE, INC.',
'111 DELI GROCERY INC.',
'111 DELI STOP CORP',
'111 FOOD MARKET CORP.',
'111 Fulton St. Retail Corp',
'111 ST. FAMILY GROCERY CORP.',
'111 SUPER STOP CORP.',
'1114 SIXTH PARKING LLC',
'1115 PENNSYLVANIA MEAT CORP.',
'1118 PERFECT IMAGE LLC',
'112 DELI AND GRILL INC.',
'112 RESTAURANT CORP.',
'112 W 14TH ST CITY DEALS',
'1133 SEABREEZE DELI & GROCERY CORP',
'1148 QUEENS LIBERTY',
'114TH GOURMET DELI & GRILL INC',
'1150 WEBSTER PHARMACY INC',
'1155 NEW WAY DELI CORP',
'1158 GROCERY & DELI, INC.',
'116 DISCOUNT INC.',
'116 ROCKAWAY PARK STATIONERS INC.',
'1161 PARKING LOT CORP.',
'1165 FRESH PRODUCE CORP',
... 10502 more items
]
city> db.inspections.distinct(
...   "business_name",
...   { result:"Violation Issued"}
... ).length
10602
city> 
```

Figure 9: Count of businesses with “Violations Found” (with a few more samples).

A.5 Figure 10: Updating a single document

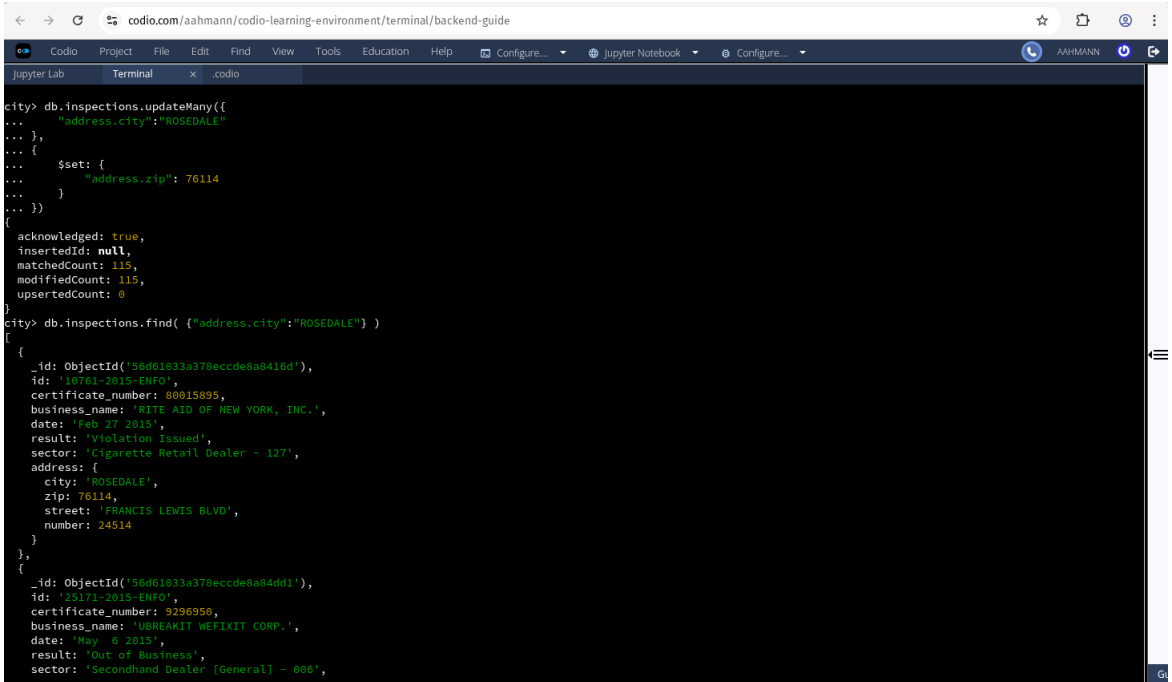


```
test> db.inspections.updateOne( { "_id":"20032-2020-ACME" }, $set:{
...   "business_name":"New ACME Flowers",
...   "result":"Business Re-opened",
...   "comments":"Flowers after the explosion"
... })
Uncaught:
SyntaxError: Unexpected token, expected ",", (1:59)

1 | db.inspections.updateOne( { "_id":"20032-2020-ACME" }, $set:{
  |
  |   "business_name":"New ACME Flowers",
  |   "result":"Business Re-opened",
  |   "comments":"Flowers after the explosion"
  |
test> db.inspections.updateOne( { "_id":"20032-2020-ACME" }, {
...   $set: {
...     "business_name":"New ACME Flowers",
...     "result":"Business Re-opened",
...     "comments":"Flowers after the explosion"
...   }
... })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.inspections.findOne( { "_id":"20032-2020-ACME" } )
{
  _id: '20032-2020-ACME',
  certificate_number: 9998888,
  business_name: 'New ACME Flowers',
  date: ISODate('2025-09-10T18:21:25.234Z'),
  result: 'Business Re-opened',
  sector: 'Explosive Retail Dealer-999',
  address: { number: 1721, street: 'Boom Road', city: 'BRONX', zip: 10463 },
  comments: 'Flowers after the explosion'
}
```

Figure 10: Updating a single document in the `inspections` collection (with mistakes, for the sake of full disclosure).

A.6 Figures 11, 12 & 13: Sub-Tasks relating to mass updating and removal of a specific document

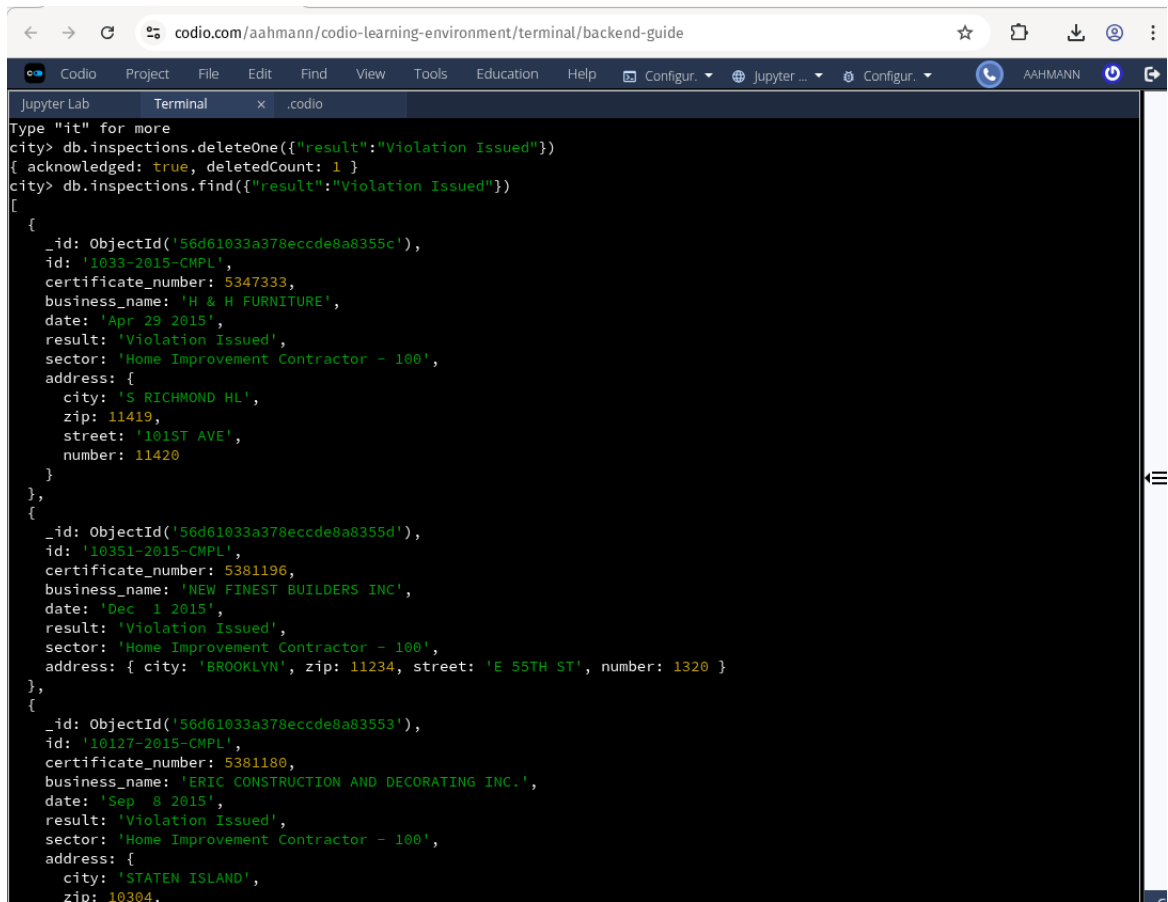


```
city> db.inspections.updateMany({
...   "address.city": "ROSEDALE"
... },
... {
...   $set: {
...     "address.zip": 76114
...   }
... })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 115,
  modifiedCount: 115,
  upsertedCount: 0
}
city> db.inspections.find( {"address.city": "ROSEDALE"} )
[
  {
    _id: ObjectId('56d61033a378eccde8a8416d'),
    id: '10761-2015-ENFO',
    certificate_number: 80015895,
    business_name: 'RIIE AID OF NEW YORK, INC.',
    dates: 'Feb 27 2015',
    result: 'Violation Issued',
    sector: 'Cigarette Retail Dealer - 127',
    address: {
      city: 'ROSEDALE',
      zip: 76114,
      street: 'FRANCIS LEWIS BLVD',
      number: 24514
    }
  },
  {
    _id: ObjectId('56d61033a378eccde8a84dd1'),
    id: '25171-2015-ENFO',
    certificate_number: 9296950,
    business_name: 'UBREAKIT WEFIXIT CORP.',
    date: 'May 6 2015',
    result: 'Out of Business',
    sector: 'Secondhand Dealer [General] - 086',
```

Figure 11: Updating documents in the `inspections` collection that meet the criteria `{ "address.city": "ROSEDALE" }` with a ZIP code of 76114.

```
city.inspections
city> db.inspections.find( {"result":"Violation Issued"})
[
  {
    _id: ObjectId('56d61033a378eccde8a83550'),
    id: '10057-2015-ENFO',
    certificate_number: 6007104,
    business_name: 'LD BUSINESS SOLUTIONS',
    date: 'Feb 25 2015',
    result: 'Violation Issued',
    sector: 'Tax Preparers - 891',
    address: {
      city: 'NEW YORK',
      zip: 10030,
      street: 'FREDERICK DOUGLASS BLVD',
      number: 2655
    }
  },
  {
    _id: ObjectId('56d61033a378eccde8a8355c'),
    id: '1033-2015-CMPL',
    certificate_number: 5347333,
    business_name: 'H & H FURNITURE',
    date: 'Apr 29 2015',
    result: 'Violation Issued',
    sector: 'Home Improvement Contractor - 100',
    address: {
      city: 'S RICHMOND HL',
      zip: 11419,
      street: '101ST AVE',
      number: 11420
    }
  },
  {
    _id: ObjectId('56d61033a378eccde8a8355d'),
    id: '10351-2015-CMPL',
    certificate_number: 5381196,
    business_name: 'NEW FINEST BUILDERS INC',
    date: 'Dec 1 2015',
    result: 'Violation Issued',
    sector: 'Home Improvement Contractor - 100',
    address: {
      city: 'NEW YORK',
      zip: 10030,
      street: 'FREDERICK DOUGLASS BLVD',
      number: 2655
    }
  }
]
```

Figure 12: Confirming the updateMany transaction.



```
codio.com/aahmann/codio-learning-environment/terminal/backend-guide
Type "it" for more
city> db.inspections.deleteOne({"result":"Violation Issued"})
{ acknowledged: true, deletedCount: 1 }
city> db.inspections.find({"result":"Violation Issued"})
[
  {
    _id: ObjectId('56d61033a378eccde8a8355c'),
    id: '1033-2015-CMPL',
    certificate_number: 5347333,
    business_name: 'H & H FURNITURE',
    date: 'Apr 29 2015',
    result: 'Violation Issued',
    sector: 'Home Improvement Contractor - 100',
    address: {
      city: 'S RICHMOND HL',
      zip: 11419,
      street: '101ST AVE',
      number: 11420
    }
  },
  {
    _id: ObjectId('56d61033a378eccde8a8355d'),
    id: '10351-2015-CMPL',
    certificate_number: 5381196,
    business_name: 'NEW FINEST BUILDERS INC',
    date: 'Dec 1 2015',
    result: 'Violation Issued',
    sector: 'Home Improvement Contractor - 100',
    address: { city: 'BROOKLYN', zip: 11234, street: 'E 55TH ST', number: 1320 }
  },
  {
    _id: ObjectId('56d61033a378eccde8a83553'),
    id: '10127-2015-CMPL',
    certificate_number: 5381180,
    business_name: 'ERIC CONSTRUCTION AND DECORATING INC.',
    date: 'Sep 8 2015',
    result: 'Violation Issued',
    sector: 'Home Improvement Contractor - 100',
    address: {
      city: 'STATEN ISLAND',
      zip: 10304,
```

Figure 13: Results of removing a single, specific document from the inspections collection.