NoSQL

REPORT ON MONGODB PROJECT IMPLEMENTATION

Stephen Power (20093364)

NoSQL Databases

MongoDB Project Implementation Report

Stephen Power (20093364)

Contents

Figures

Introduction

This NoSQL project required downloading, installing and testing a Mongo database environment. I created a NoSQL database that modelled a Ticket Management System. This database stores data on tickets, orders, and customers. The database was created using the Document Object Model (DOM). Functionality was created for create, read, update, delete (CRUD) activities. The database is migrated to Atlas, which is MongoDB's cloud-based hosting service. Using Atlas, the database is deployed in a cloud environment. Finally, aggregation pipelines are created for the database.

NoSQL

NoSQL is an acronym for 'not only structured query language'. NoSQL is an approach to database management that can be used by a wide range of data models. NoSQL databases are non-relational, distributed, flexible and scalable. There is a lack of database schema within NoSQL databases which is one of the many differentiating factors between itself and structured query language (SQL).

There are four popular types of NoSQL databases:

1. Document stores (document like-structures)
2. Graph data stores (graphical data nodes)
3. Key-value stores (unique keys and values)
4. Wide-column stores (table-like structures)

Advantages of NoSQL databases:

- **Scalability:** Evolving datasets can be handled efficiently by increasing servers to meet demand with zero downtime.
- **Flexibility:** Data is stored in a free-form fashion instead of hard structured schema's which allows for innovation and fast application development.
- **Availability:** Latency for users is minimised due to the availability of servers, data centres and cloud resources.

MongoDB

MongoDB is a document-orientated database that is designed to store large sets of data and also allows you to work with data efficiently. MongoDB is a NoSQL database as the storage and retrieval of data within the database are not in the form of tables.

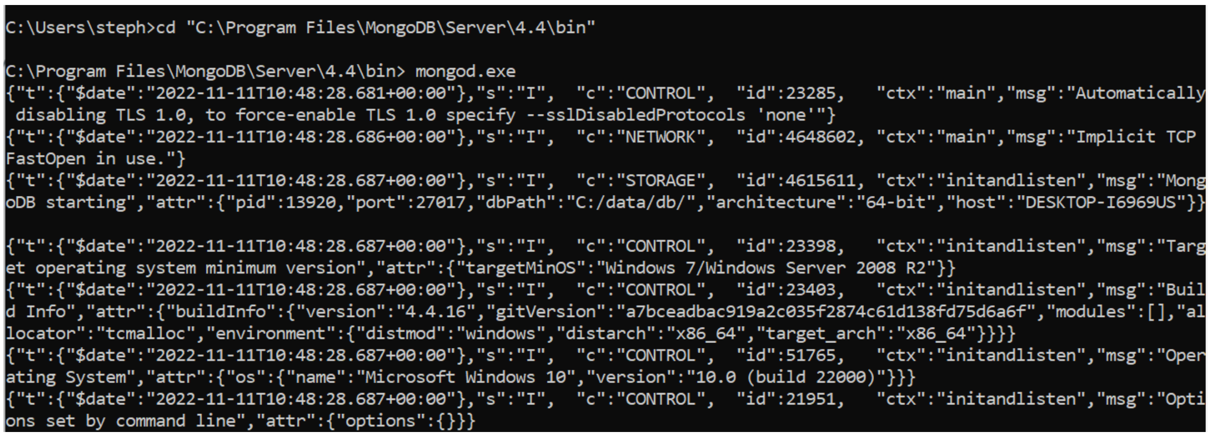
MongoDB gives users a database server where multiple databases can be created and modified. As MongoDB is a NoSQL database, data is stored within collections and documents. Collections are the parent and documents are stored within it. Documents are created using fields. Fields are key-value pairs that store data in the documents. Documents allow the storage of nested data. Nesting data within one document makes working with data a lot easier and results in efficiency.

In a MongoDB server, users are allowed to run multiple databases. Although MongoDB uses high memory for data storage, the availability and high performance it offers makes it a very useful tool for database management projects.

Setting up Mongo

I downloaded the MongoDB Community Server (Version 6.0.2) for the windows package. This package could be downloaded from the mongodb.com website. Once the package was downloaded, the installation took place as the appropriate configurations were selected.

The 'mongod.exe' command is used to start the MongoDB application, which is the database server.



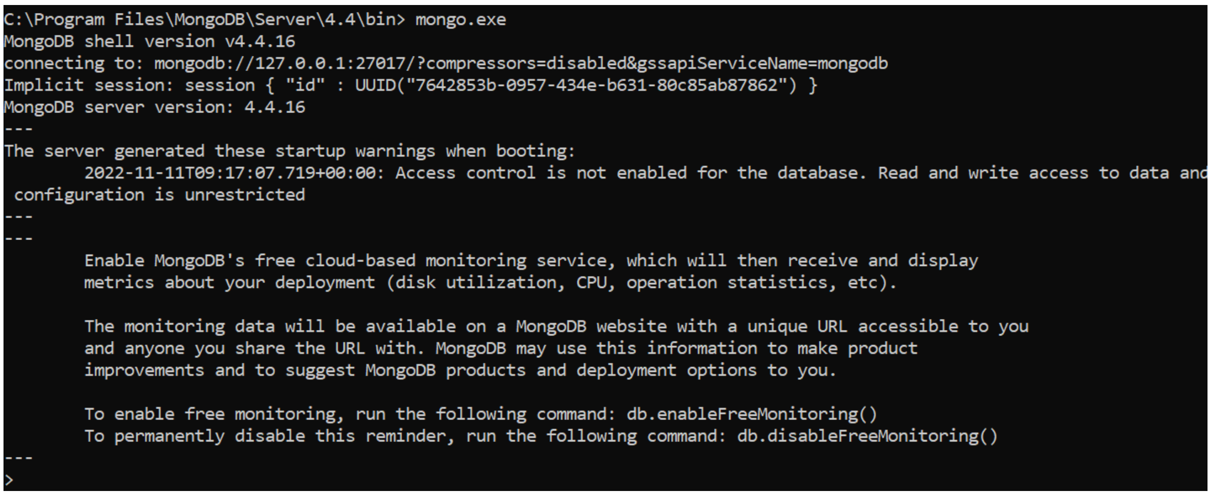
```
C:\Users\steph>cd "C:\Program Files\MongoDB\Server\4.4\bin"

C:\Program Files\MongoDB\Server\4.4\bin> mongod.exe
{"t":{"$date":"2022-11-11T10:48:28.681+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automatically
 disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2022-11-11T10:48:28.686+00:00"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx":"main","msg":"Implicit TCP
FastOpen in use."}
{"t":{"$date":"2022-11-11T10:48:28.687+00:00"},"s":"I",  "c":"STORAGE",  "id":4615611, "ctx":"initandlisten","msg":"Mong
oDB starting","attr":{"pid":13920,"port":27017,"dbPath":"C:/data/db/","architecture":"64-bit","host":"DESKTOP-I6969US"}}

{"t":{"$date":"2022-11-11T10:48:28.687+00:00"},"s":"I",  "c":"CONTROL",  "id":23398,   "ctx":"initandlisten","msg":"Targ
et operating system minimum version","attr":{"targetMinOS":"Windows 7/Windows Server 2008 R2"}}
{"t":{"$date":"2022-11-11T10:48:28.687+00:00"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx":"initandlisten","msg":"Buil
d Info","attr":{"buildInfo":{"version":"4.4.16","gitVersion":"a7bceadbac919a2c035f2874c61d138fd75d6a6f","modules":[],"al
locator":"tcmalloc","environment":{"distmod":"windows","distarch":"x86_64","target_arch":"x86_64"}}}}
{"t":{"$date":"2022-11-11T10:48:28.687+00:00"},"s":"I",  "c":"CONTROL",  "id":51765,   "ctx":"initandlisten","msg":"Oper
ating System","attr":{"os":{"name":"Microsoft Windows 10","version":"10.0 (build 22000)"}}}
{"t":{"$date":"2022-11-11T10:48:28.687+00:00"},"s":"I",  "c":"CONTROL",  "id":21951,   "ctx":"initandlisten","msg":"Opti
ons set by command line","attr":{"options":{}}}
```

Figure 1: mongod.exe

The 'mongo.exe' command is used to start the mongo client process.



```
C:\Program Files\MongoDB\Server\4.4\bin> mongo.exe
MongoDB shell version v4.4.16
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("7642853b-0957-434e-b631-80c85ab87862") }
MongoDB server version: 4.4.16
---
The server generated these startup warnings when booting:
        2022-11-11T09:17:07.719+00:00: Access control is not enabled for the database. Read and write access to data and
 configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```

Figure 2: mongo.exe

Ticket Management System

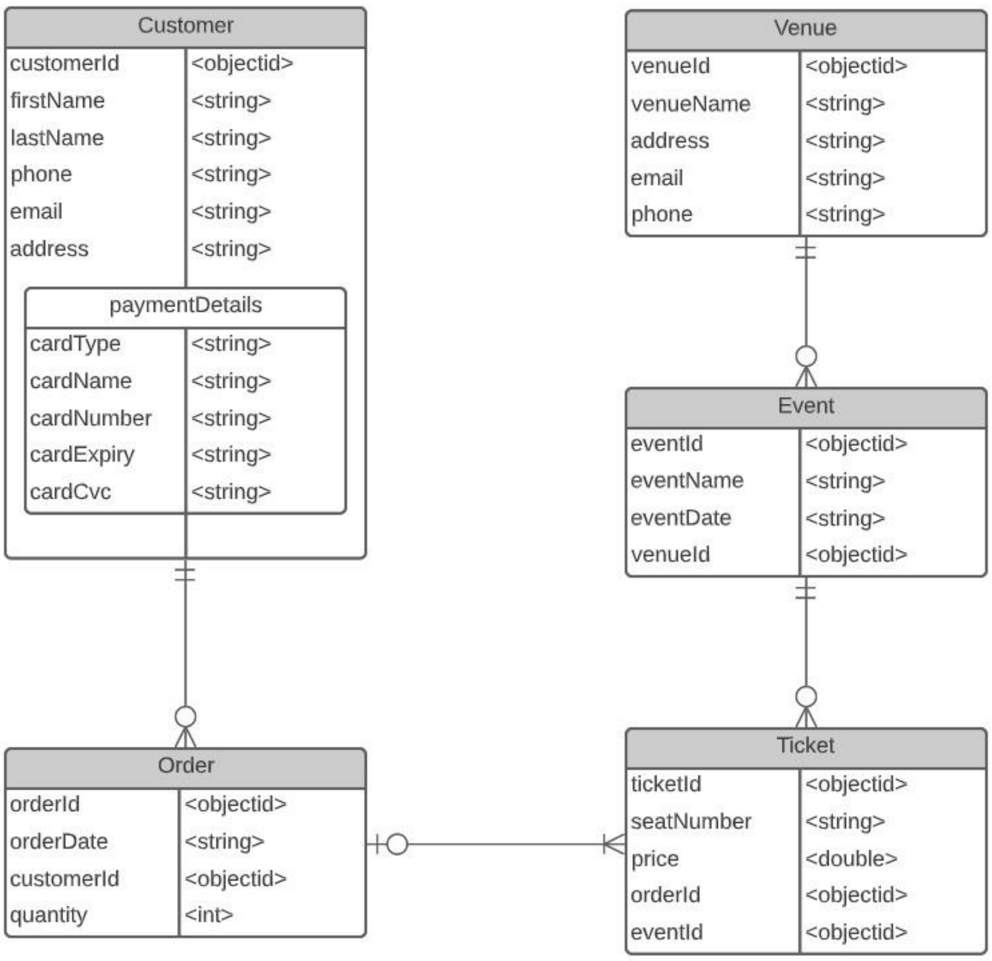


Figure 3: EER Diagram

The Ticket Management System is a database system used to store data on ticket orders for events. The above EER diagram describes the relationship between each of the entities in the database. Three collections are to be created based on this system:

1. **Order Collection** – stores data on the tickets included in the order and the customer that purchased them.
2. **Customer Collection** – stores detailed data on the customer, payment details and previous orders.
3. **Ticket Collection** – stores detailed data on the ticket and the specific event that the ticket is valid for.

Order Collection:

This collection will contain an array of tickets that are included in the order as an order can contain one or more tickets. The order id, date and ticket quantity data is stored. The customer id is also stored as each order must contain one and only one customer.

Customer Collection:

This collection stores an array of orders as a customer can have many orders. The customer id, first name, last name, phone number, email address, and home address data is stored. Customer payment details are stored which contain card number, expiry date and CVC number data.

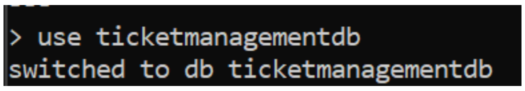
Ticket Collection:

The ticket collection contains the ticket id, seat number and price of the ticket. Each ticket is valid for a specific event so the event id, date and name are stored. Venue data is also stored which consists of venue name, address and email.

CRUD

Creating The Database:

The ‘use’ command creates a database and switches to that newly created database. The ticket management database was initialised by using the command ‘use ticketmanagementdb’. The database will contain 10 tickets, 5 orders, and 2 customers.

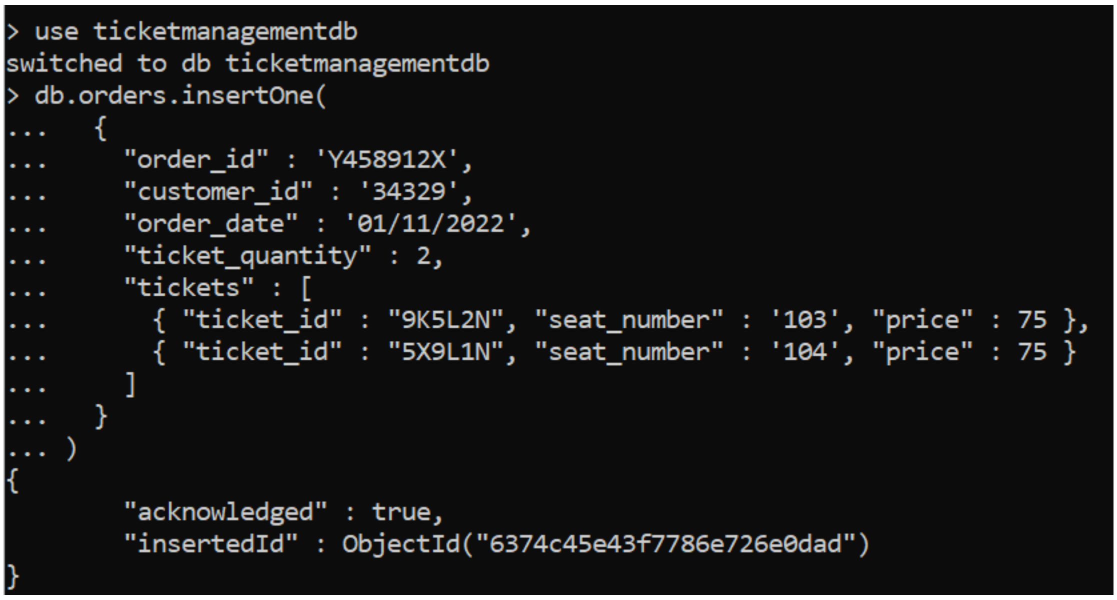


```
> use ticketmanagementdb
switched to db ticketmanagementdb
```

Figure 4: Initialising Ticket Management Database

Creating The Order Collection:

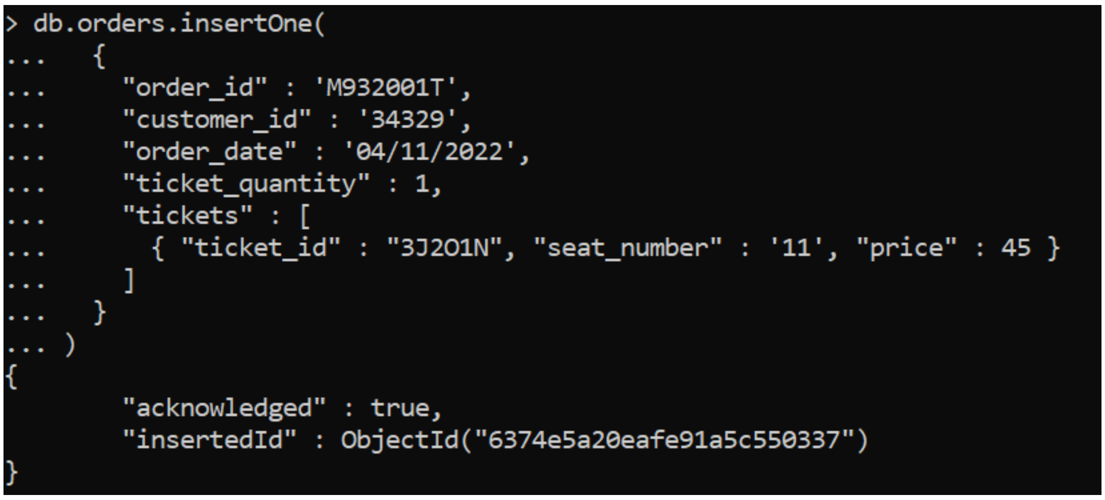
Insert 1 for Orders Collection:



```
> use ticketmanagementdb
switched to db ticketmanagementdb
> db.orders.insertOne(
...    {
...       "order_id" : 'Y458912X',
...       "customer_id" : '34329',
...       "order_date" : '01/11/2022',
...       "ticket_quantity" : 2,
...       "tickets" : [
...         { "ticket_id" : "9K5L2N", "seat_number" : '103', "price" : 75 },
...         { "ticket_id" : "5X9L1N", "seat_number" : '104', "price" : 75 }
...       ]
...    }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374c45e43f7786e726e0dad")
}
```

Figure 5: Orders - Insert 1

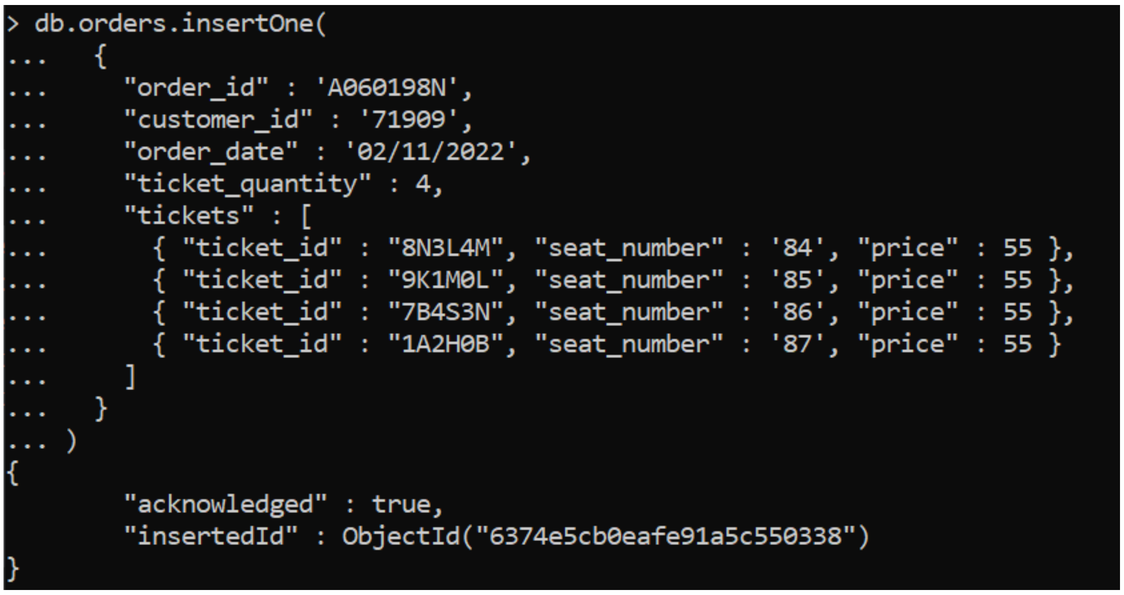
Insert 2 for Orders Collection:



```
> db.orders.insertOne(
...    {
...       "order_id" : 'M932001T',
...       "customer_id" : '34329',
...       "order_date" : '04/11/2022',
...       "ticket_quantity" : 1,
...       "tickets" : [
...         { "ticket_id" : "3J2O1N", "seat_number" : '11', "price" : 45 }
...       ]
...    }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e5a20eafe91a5c550337")
}
```

Figure 6: Orders - Insert 2

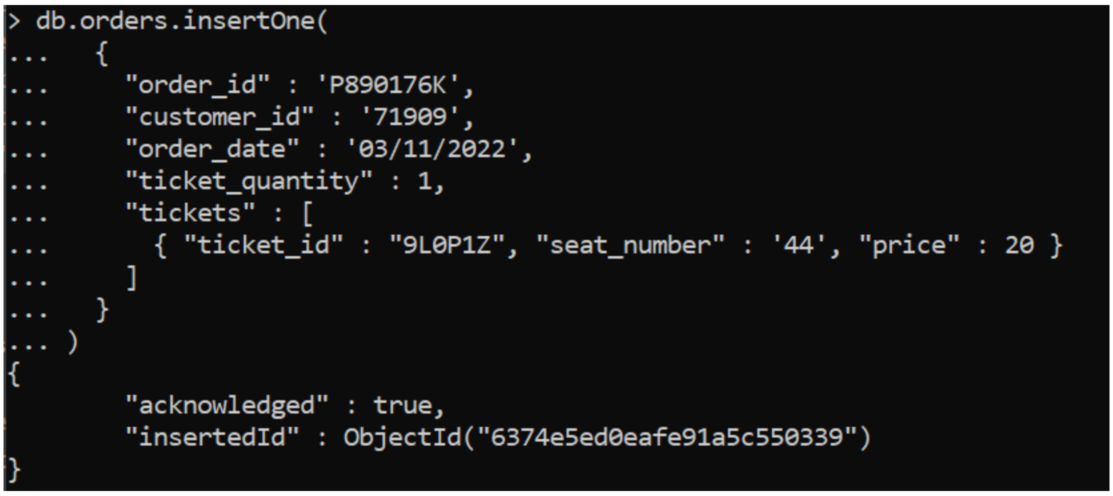
Insert 3 for Orders Collection:



```
> db.orders.insertOne(
...    {
...      "order_id" : 'A060198N',
...      "customer_id" : '71909',
...      "order_date" : '02/11/2022',
...      "ticket_quantity" : 4,
...      "tickets" : [
...        { "ticket_id" : "8N3L4M", "seat_number" : '84', "price" : 55 },
...        { "ticket_id" : "9K1M0L", "seat_number" : '85', "price" : 55 },
...        { "ticket_id" : "7B4S3N", "seat_number" : '86', "price" : 55 },
...        { "ticket_id" : "1A2H0B", "seat_number" : '87', "price" : 55 }
...      ]
...    }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e5cb0eafe91a5c550338")
}
```

Figure 7: Orders - Insert 3

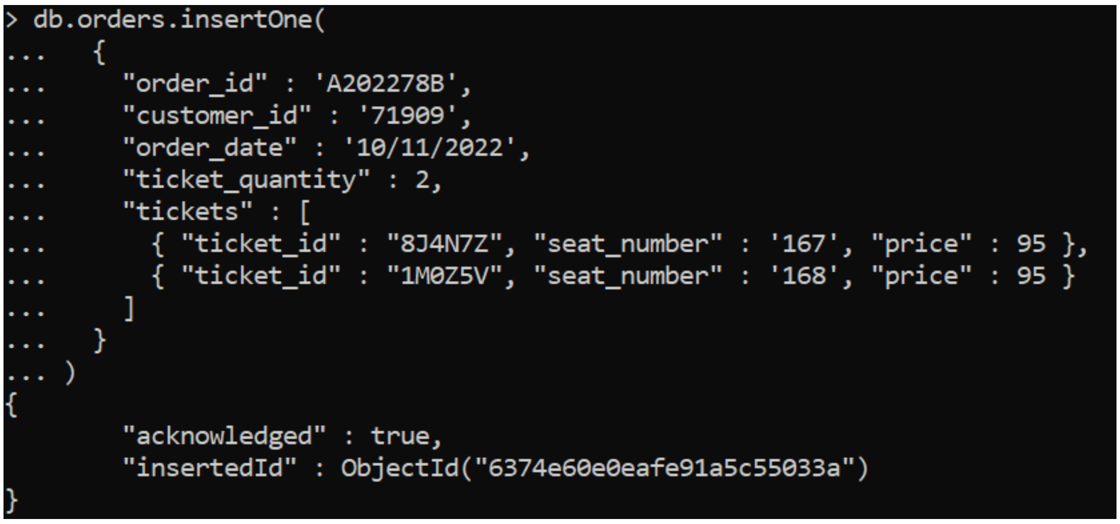
Insert 4 for Orders Collection:



```
> db.orders.insertOne(
...    {
...      "order_id" : 'P890176K',
...      "customer_id" : '71909',
...      "order_date" : '03/11/2022',
...      "ticket_quantity" : 1,
...      "tickets" : [
...        { "ticket_id" : "9L0P1Z", "seat_number" : '44', "price" : 20 }
...      ]
...    }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e5ed0eafe91a5c550339")
}
```

Figure 8: Orders - Insert 4

Insert 5 for Orders Collection:

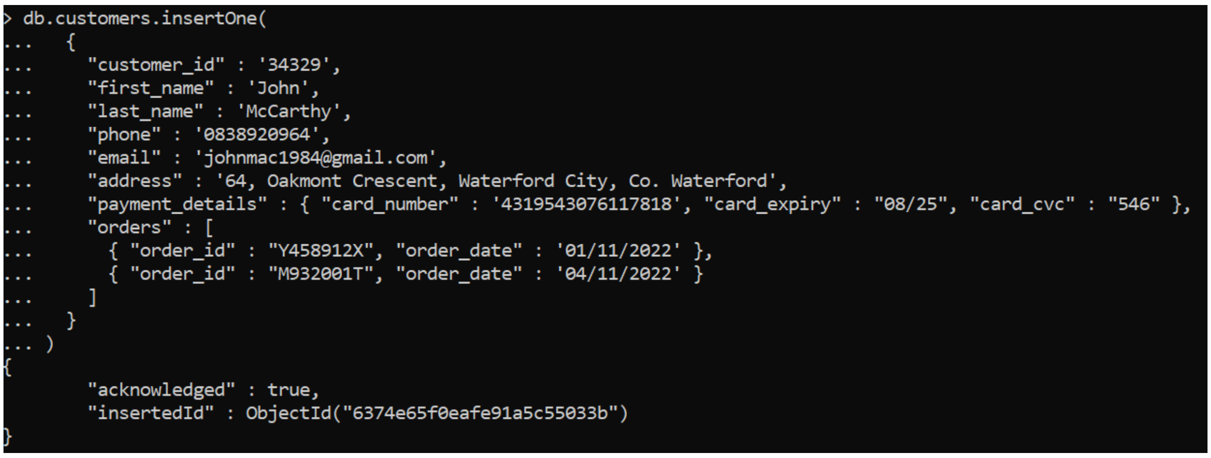


```
> db.orders.insertOne(
...    {
...      "order_id" : 'A202278B',
...      "customer_id" : '71909',
...      "order_date" : '10/11/2022',
...      "ticket_quantity" : 2,
...      "tickets" : [
...        { "ticket_id" : "8J4N7Z", "seat_number" : '167', "price" : 95 },
...        { "ticket_id" : "1M0Z5V", "seat_number" : '168', "price" : 95 }
...      ]
...    }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e60e0eafe91a5c55033a")
}
```

Figure 9: Orders - Insert 5

Creating The Customer Collection:

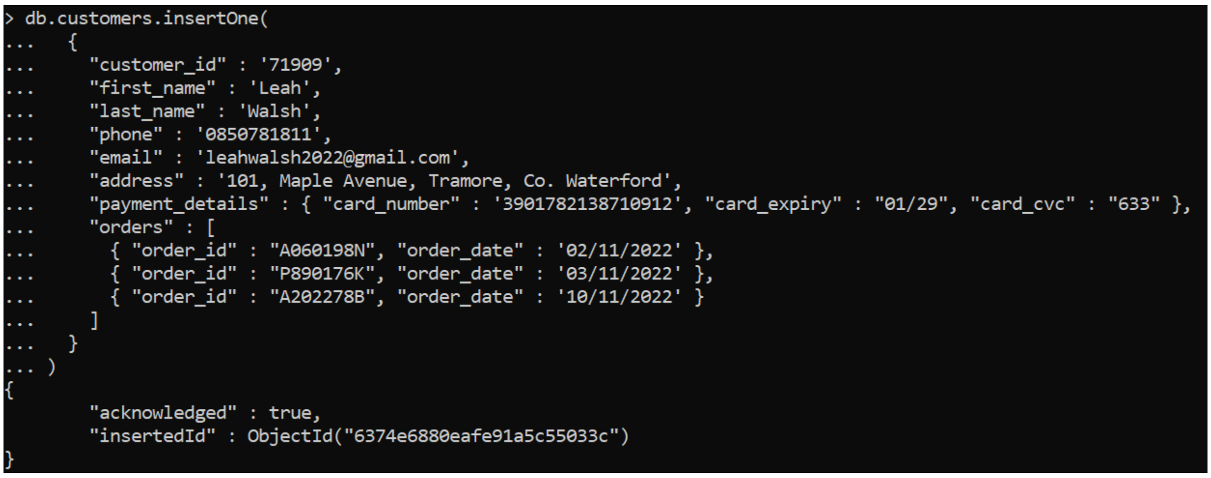
Insert 1 for Customer Collection:



```
> db.customers.insertOne(
...    {
...      "customer_id" : '34329',
...      "first_name" : 'John',
...      "last_name" : 'McCarthy',
...      "phone" : '0838920964',
...      "email" : 'johnmac1984@gmail.com',
...      "address" : '64, Oakmont Crescent, Waterford City, Co. Waterford',
...      "payment_details" : { "card_number" : '4319543076117818', "card_expiry" : "08/25", "card_cvc" : "546" },
...      "orders" : [
...        { "order_id" : "Y458912X", "order_date" : '01/11/2022' },
...        { "order_id" : "M932001T", "order_date" : '04/11/2022' }
...      ]
...    }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e65f0eafe91a5c55033b")
}
```

Figure 10: Customers - Insert 1

Insert 2 for Customer Collection:

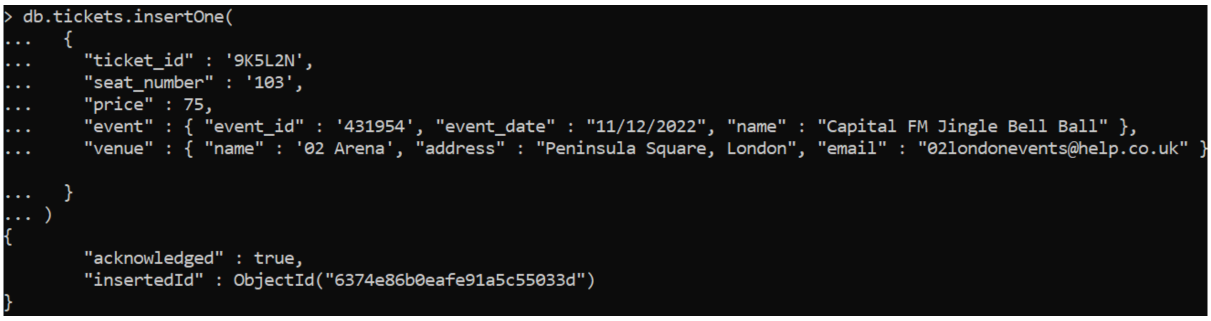


```
> db.customers.insertOne(
...   {
...     "customer_id" : '71909',
...     "first_name" : 'Leah',
...     "last_name" : 'Walsh',
...     "phone" : '0850781811',
...     "email" : 'leahwalsh2022@gmail.com',
...     "address" : '101, Maple Avenue, Tramore, Co. Waterford',
...     "payment_details" : { "card_number" : '3901782138710912', "card_expiry" : "01/29", "card_cvc" : "633" },
...     "orders" : [
...       { "order_id" : "A060198N", "order_date" : '02/11/2022' },
...       { "order_id" : "P890176K", "order_date" : '03/11/2022' },
...       { "order_id" : "A202278B", "order_date" : '10/11/2022' }
...     ]
...   }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e6880eafe91a5c55033c")
}
```

Figure 11: Customers - Insert 2

Creating The Ticket Collection:

Insert 1 for Tickets Collection:

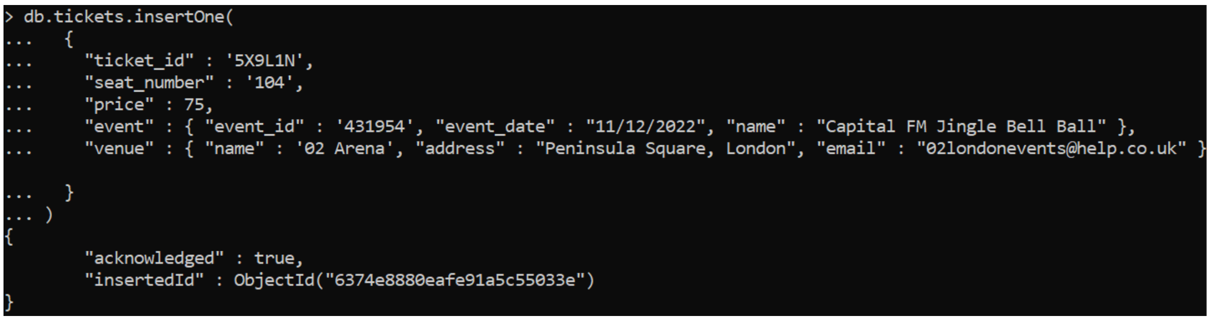


```
> db.tickets.insertOne(
...   {
...     "ticket_id" : '9K5L2N',
...     "seat_number" : '103',
...     "price" : 75,
...     "event" : { "event_id" : '431954', "event_date" : "11/12/2022", "name" : "Capital FM Jingle Bell Ball" },
...     "venue" : { "name" : '02 Arena', "address" : "Peninsula Square, London", "email" : "02londonevents@help.co.uk" }

...   }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e86b0eafe91a5c55033d")
}
```

Figure 12: Tickets - Insert 1

Insert 2 for Tickets Collection:

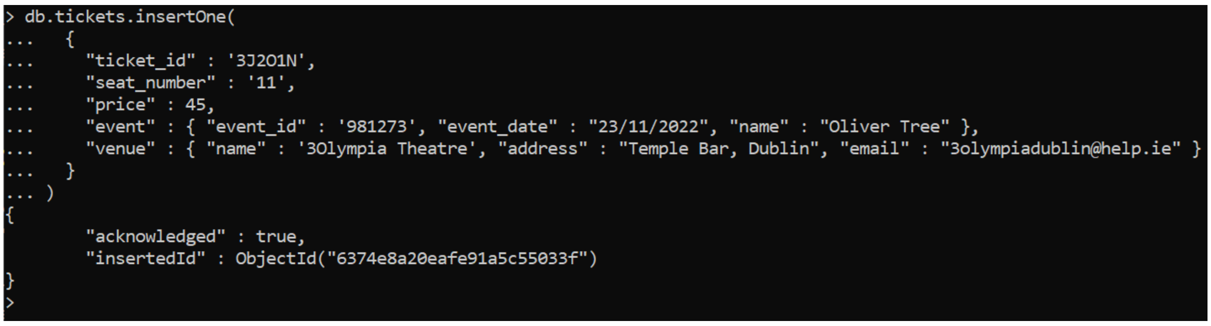


```
> db.tickets.insertOne(
...   {
...     "ticket_id" : '5X9L1N',
...     "seat_number" : '104',
...     "price" : 75,
...     "event" : { "event_id" : '431954', "event_date" : "11/12/2022", "name" : "Capital FM Jingle Bell Ball" },
...     "venue" : { "name" : '02 Arena', "address" : "Peninsula Square, London", "email" : "02londonevents@help.co.uk" }

...   }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e8880eafe91a5c55033e")
}
```

Figure 13: Tickets - Insert 2

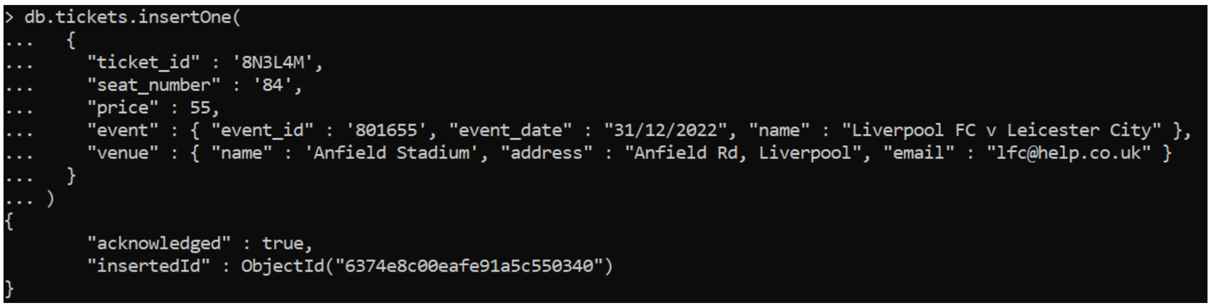
Insert 3 for Tickets Collection:



```
> db.tickets.insertOne(
...   {
...     "ticket_id" : '3J2O1N',
...     "seat_number" : '11',
...     "price" : 45,
...     "event" : { "event_id" : '981273', "event_date" : "23/11/2022", "name" : "Oliver Tree" },
...     "venue" : { "name" : '3Olympia Theatre', "address" : "Temple Bar, Dublin", "email" : "3olympiadublin@help.ie" }
...   }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e8a20eafe91a5c55033f")
}
>
```

Figure 14: Tickets - Insert 3

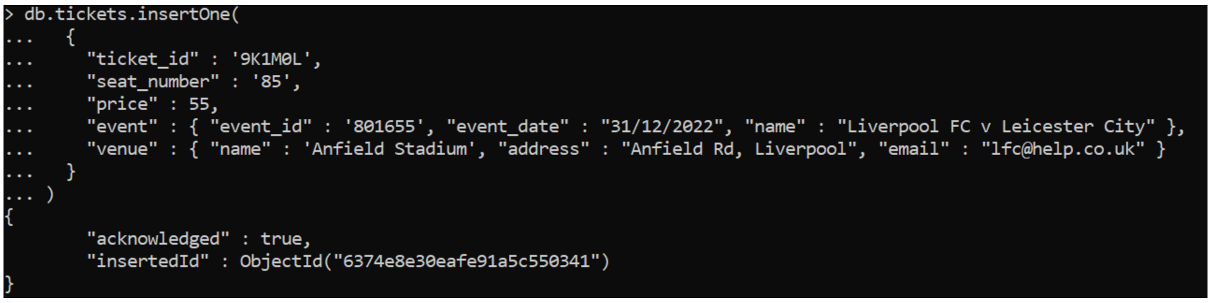
Insert 4 for Tickets Collection:



```
> db.tickets.insertOne(
...   {
...     "ticket_id" : '8N3L4M',
...     "seat_number" : '84',
...     "price" : 55,
...     "event" : { "event_id" : '801655', "event_date" : "31/12/2022", "name" : "Liverpool FC v Leicester City" },
...     "venue" : { "name" : 'Anfield Stadium', "address" : "Anfield Rd, Liverpool", "email" : "lfc@help.co.uk" }
...   }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e8c00eafe91a5c550340")
}
```

Figure 15: Tickets - Insert 4

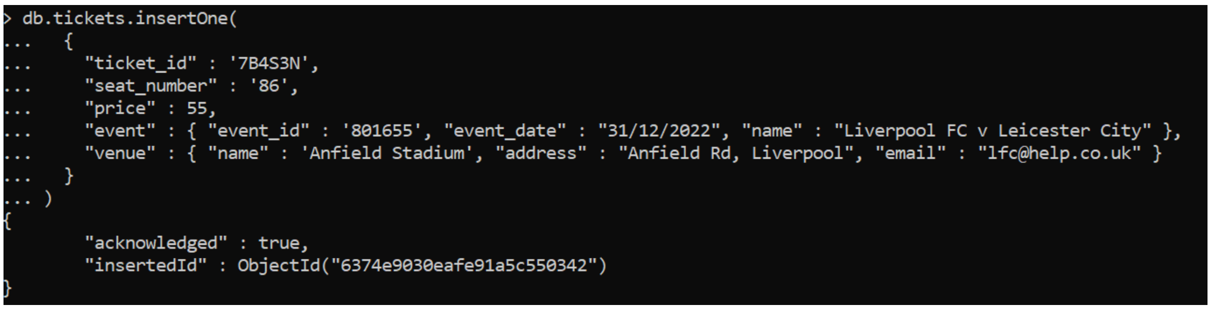
Insert 5 for Tickets Collection:



```
> db.tickets.insertOne(
...   {
...     "ticket_id" : '9K1M0L',
...     "seat_number" : '85',
...     "price" : 55,
...     "event" : { "event_id" : '801655', "event_date" : "31/12/2022", "name" : "Liverpool FC v Leicester City" },
...     "venue" : { "name" : 'Anfield Stadium', "address" : "Anfield Rd, Liverpool", "email" : "lfc@help.co.uk" }
...   }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e8e30eafe91a5c550341")
}
```

Figure 16: Tickets - Insert 5

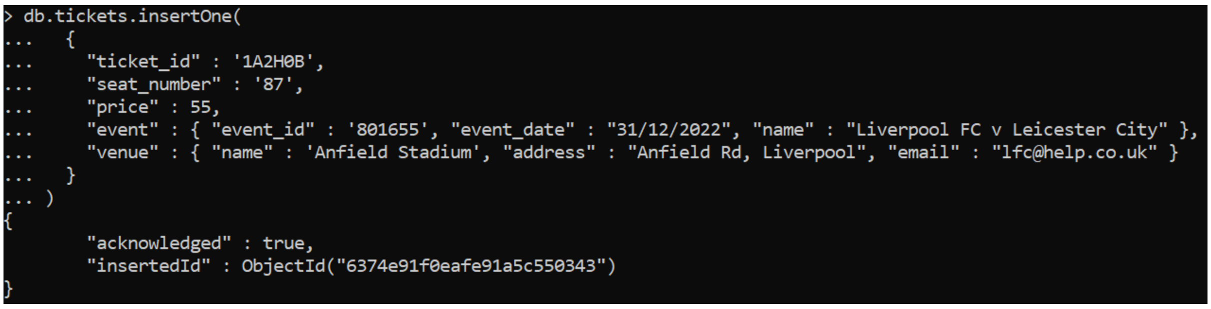
Insert 6 for Tickets Collection:



```
> db.tickets.insertOne(
...    {
...      "ticket_id" : '7B4S3N',
...      "seat_number" : '86',
...      "price" : 55,
...      "event" : { "event_id" : '801655', "event_date" : "31/12/2022", "name" : "Liverpool FC v Leicester City" },
...      "venue" : { "name" : 'Anfield Stadium', "address" : "Anfield Rd, Liverpool", "email" : "lfc@help.co.uk" }
...    }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e9030eafe91a5c550342")
}
```

Figure 17: Tickets - Insert 6

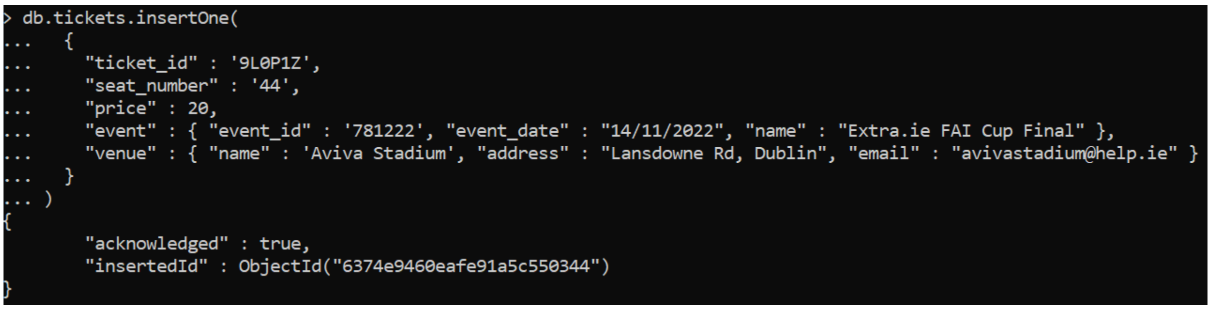
Insert 7 for Tickets Collection:



```
> db.tickets.insertOne(
...    {
...      "ticket_id" : '1A2H0B',
...      "seat_number" : '87',
...      "price" : 55,
...      "event" : { "event_id" : '801655', "event_date" : "31/12/2022", "name" : "Liverpool FC v Leicester City" },
...      "venue" : { "name" : 'Anfield Stadium', "address" : "Anfield Rd, Liverpool", "email" : "lfc@help.co.uk" }
...    }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e91f0eafe91a5c550343")
}
```

Figure 18: Tickets - Insert 7

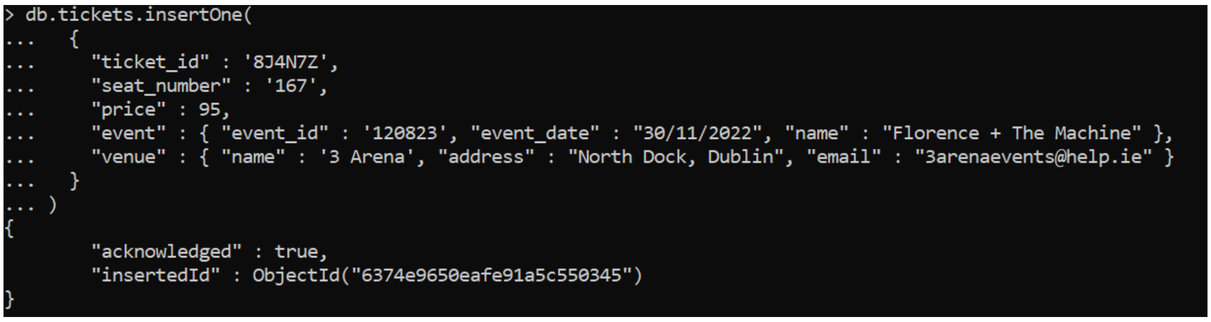
Insert 8 for Tickets Collection:



```
> db.tickets.insertOne(
...    {
...      "ticket_id" : '9L0P1Z',
...      "seat_number" : '44',
...      "price" : 20,
...      "event" : { "event_id" : '781222', "event_date" : "14/11/2022", "name" : "Extra.ie FAI Cup Final" },
...      "venue" : { "name" : 'Aviva Stadium', "address" : "Lansdowne Rd, Dublin", "email" : "avivastadium@help.ie" }
...    }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e9460eafe91a5c550344")
}
```

Figure 19: Tickets - Insert 8

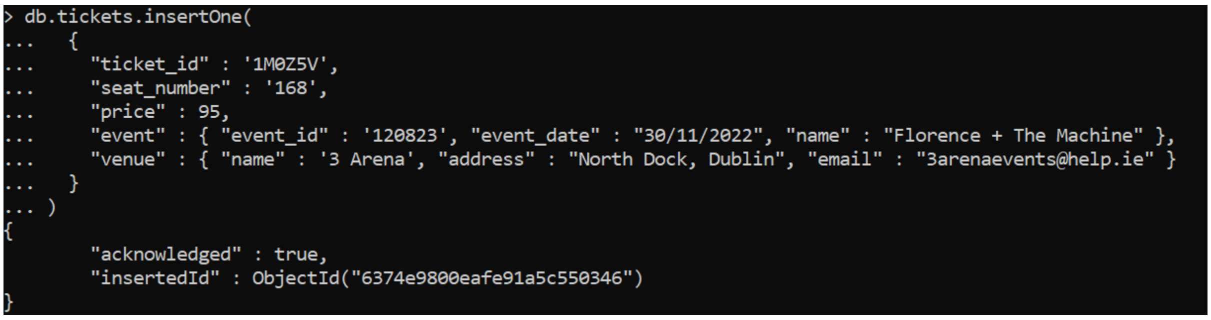
Insert 9 for Tickets Collection:



```
> db.tickets.insertOne(
...   {
...     "ticket_id" : '8J4N7Z',
...     "seat_number" : '167',
...     "price" : 95,
...     "event" : { "event_id" : '120823', "event_date" : "30/11/2022", "name" : "Florence + The Machine" },
...     "venue" : { "name" : '3 Arena', "address" : "North Dock, Dublin", "email" : "3arenaevents@help.ie" }
...   }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e9650eafe91a5c550345")
}
```

Figure 20: Tickets - Insert 9

Insert 10 for Tickets Collection:

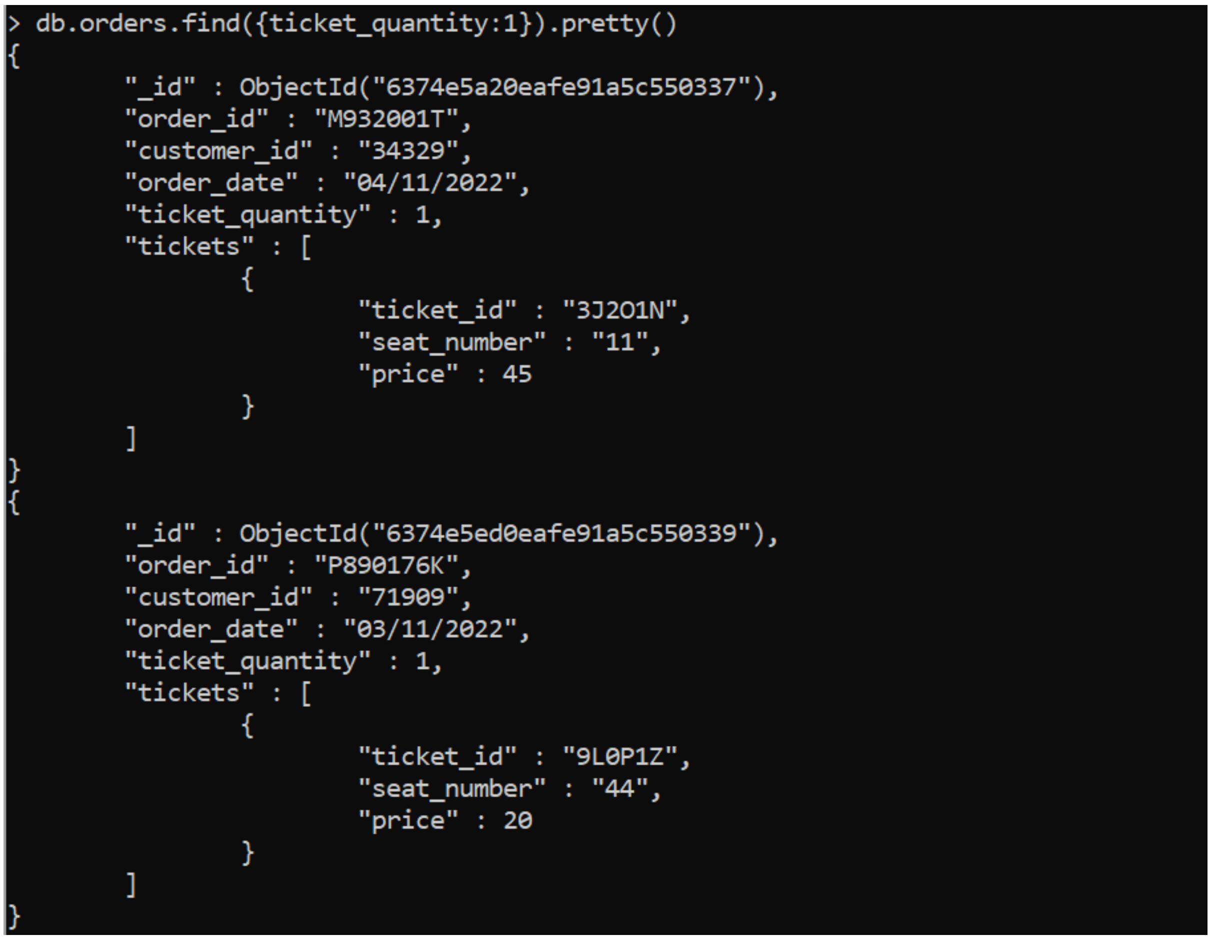


```
> db.tickets.insertOne(
...   {
...     "ticket_id" : '1M0Z5V',
...     "seat_number" : '168',
...     "price" : 95,
...     "event" : { "event_id" : '120823', "event_date" : "30/11/2022", "name" : "Florence + The Machine" },
...     "venue" : { "name" : '3 Arena', "address" : "North Dock, Dublin", "email" : "3arenaevents@help.ie" }
...   }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6374e9800eafe91a5c550346")
}
```

Figure 21: Tickets - Insert 10

Finding Data in the Database:

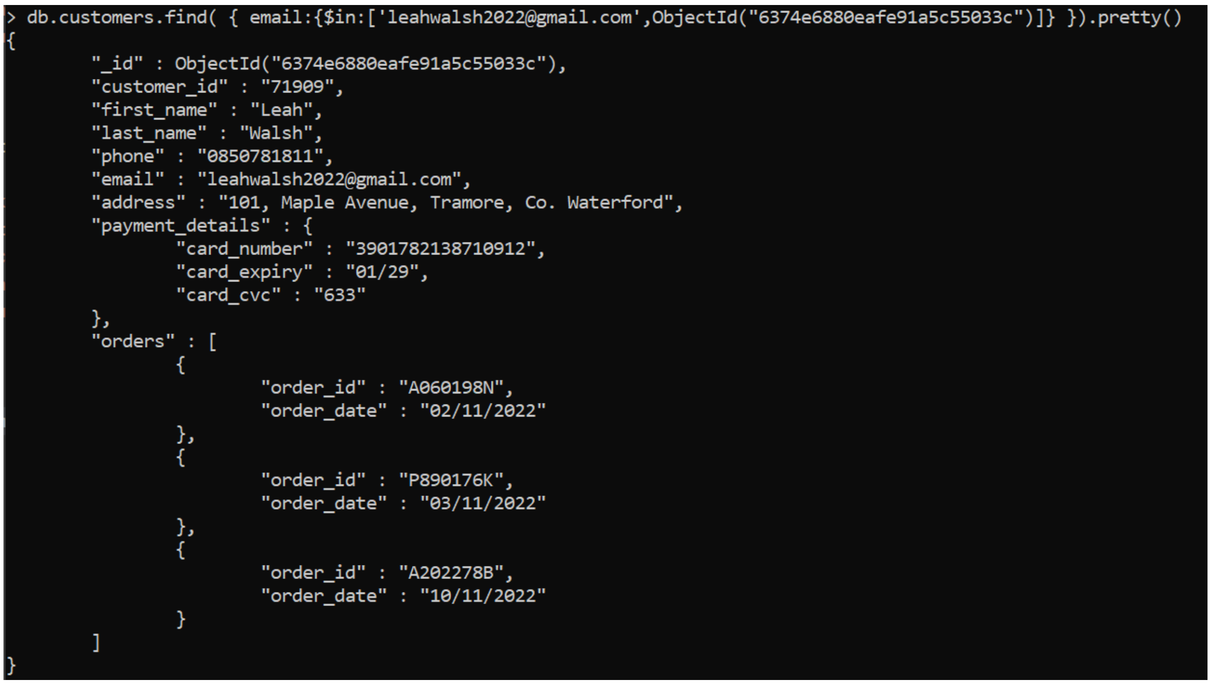
1. Find documents in the order collection where ‘ticket_quantity’ equals 1:



```
> db.orders.find({ticket_quantity:1}).pretty()
{
        "_id" : ObjectId("6374e5a20eafe91a5c550337"),
        "order_id" : "M932001T",
        "customer_id" : "34329",
        "order_date" : "04/11/2022",
        "ticket_quantity" : 1,
        "tickets" : [
                {
                        "ticket_id" : "3J2O1N",
                        "seat_number" : "11",
                        "price" : 45
                }
        ]
}
{
        "_id" : ObjectId("6374e5ed0eafe91a5c550339"),
        "order_id" : "P890176K",
        "customer_id" : "71909",
        "order_date" : "03/11/2022",
        "ticket_quantity" : 1,
        "tickets" : [
                {
                        "ticket_id" : "9L0P1Z",
                        "seat_number" : "44",
                        "price" : 20
                }
        ]
}
```

Figure 22: Orders - Find Ticket Quantity

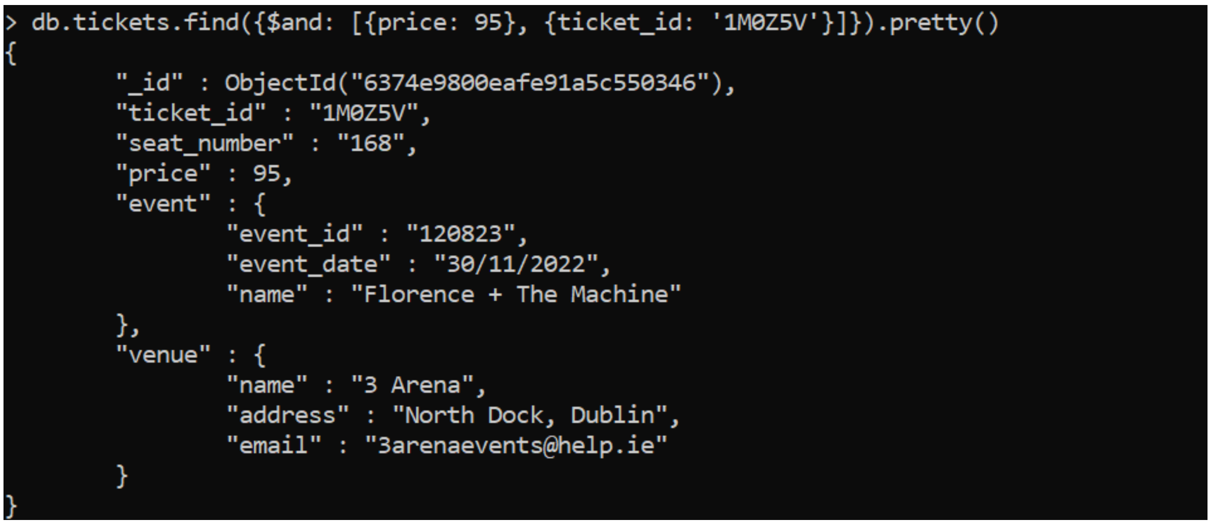
2. Find documents in the customer collection where the customer email equals 'leahwalsh2022@gmail.com' and the object id equals '6374e6880eafe91a5c55033c':



```
> db.customers.find( { email:{$in:['leahwalsh2022@gmail.com',ObjectId("6374e6880eafe91a5c55033c")]} }).pretty()
{
        "_id" : ObjectId("6374e6880eafe91a5c55033c"),
        "customer_id" : "71909",
        "first_name" : "Leah",
        "last_name" : "Walsh",
        "phone" : "0850781811",
        "email" : "leahwalsh2022@gmail.com",
        "address" : "101, Maple Avenue, Tramore, Co. Waterford",
        "payment_details" : {
                "card_number" : "3901782138710912",
                "card_expiry" : "01/29",
                "card_cvc" : "633"
        },
        "orders" : [
                {
                        "order_id" : "A060198N",
                        "order_date" : "02/11/2022"
                },
                {
                        "order_id" : "P890176K",
                        "order_date" : "03/11/2022"
                },
                {
                        "order_id" : "A202278B",
                        "order_date" : "10/11/2022"
                }
        ]
}
```

Figure 23: Customers - ($in)

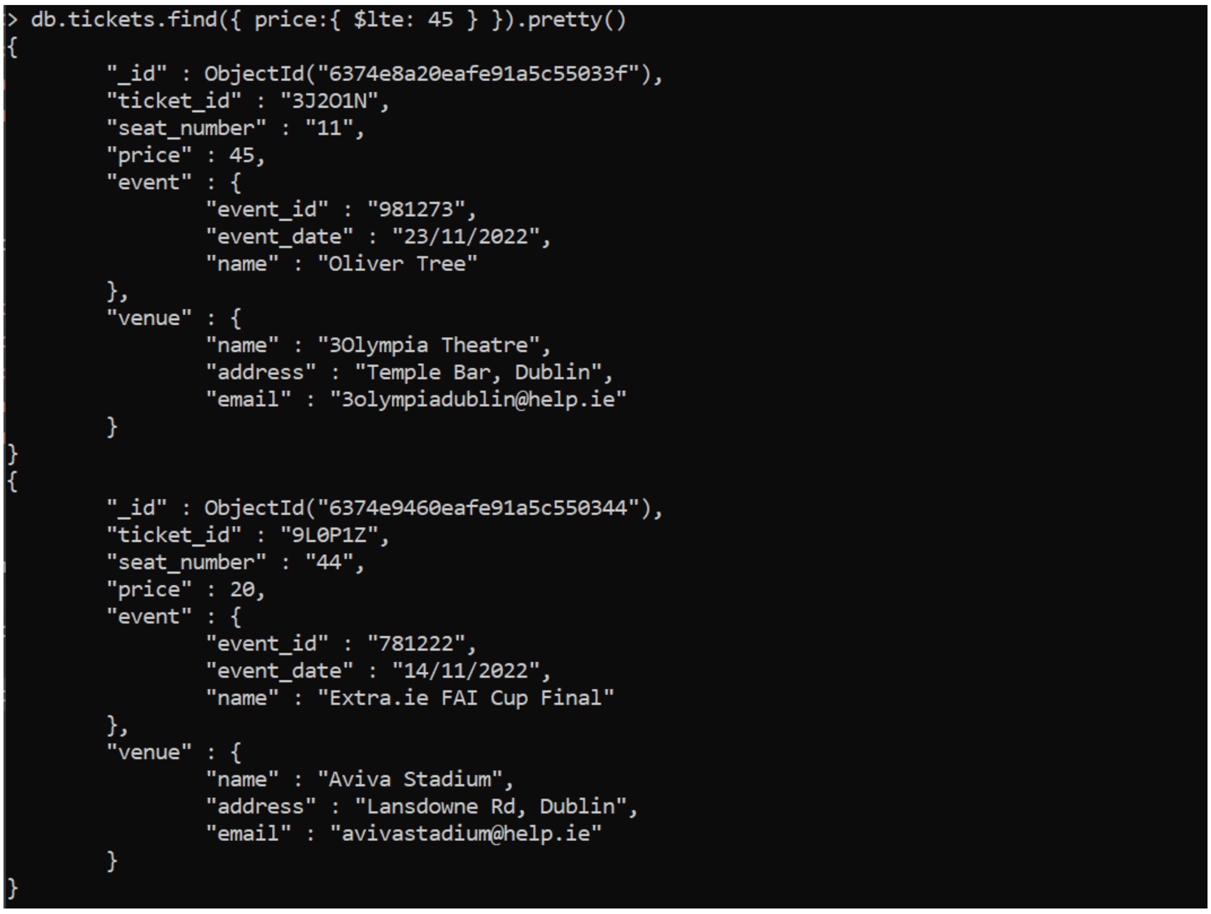
3. Find documents in the ticket collection where the ticket price equals 95 and the ticket id equals '1M0Z5V':



```
> db.tickets.find({$and: [{price: 95}, {ticket_id: '1M0Z5V'}]}).pretty()
{
        "_id" : ObjectId("6374e9800eafe91a5c550346"),
        "ticket_id" : "1M0Z5V",
        "seat_number" : "168",
        "price" : 95,
        "event" : {
                "event_id" : "120823",
                "event_date" : "30/11/2022",
                "name" : "Florence + The Machine"
        },
        "venue" : {
                "name" : "3 Arena",
                "address" : "North Dock, Dublin",
                "email" : "3arenaevents@help.ie"
        }
}
```

Figure 24: Tickets - ($and)

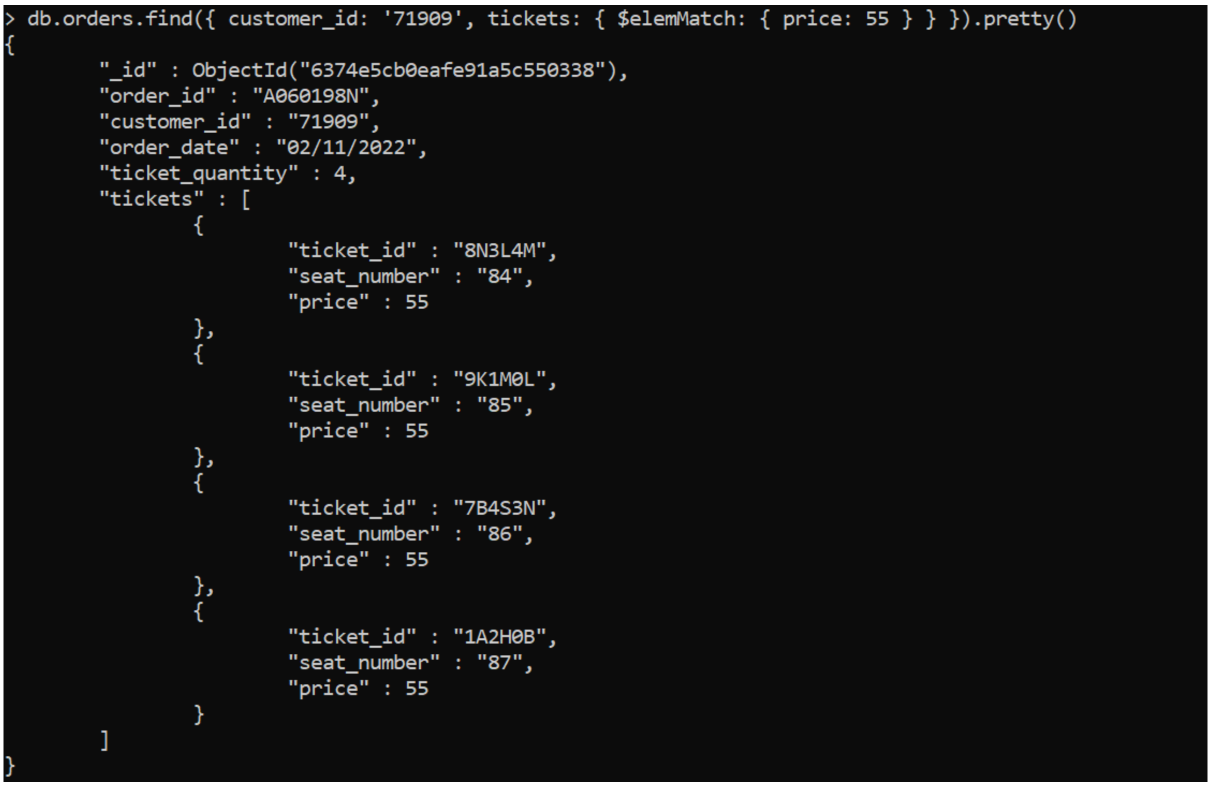
4. Find documents in the ticket collection where the ticket price is equal to or less than 45:



```
> db.tickets.find({ price:{ $lte: 45 } }).pretty()
{
        "_id" : ObjectId("6374e8a20eafe91a5c55033f"),
        "ticket_id" : "3J2O1N",
        "seat_number" : "11",
        "price" : 45,
        "event" : {
                "event_id" : "981273",
                "event_date" : "23/11/2022",
                "name" : "Oliver Tree"
        },
        "venue" : {
                "name" : "3Olympia Theatre",
                "address" : "Temple Bar, Dublin",
                "email" : "3olympiadublin@help.ie"
        }
}
{
        "_id" : ObjectId("6374e9460eafe91a5c550344"),
        "ticket_id" : "9L0P1Z",
        "seat_number" : "44",
        "price" : 20,
        "event" : {
                "event_id" : "781222",
                "event_date" : "14/11/2022",
                "name" : "Extra.ie FAI Cup Final"
        },
        "venue" : {
                "name" : "Aviva Stadium",
                "address" : "Lansdowne Rd, Dublin",
                "email" : "avivastadium@help.ie"
        }
}
```

Figure 25: Tickets - ($lte)

5. Find documents in the order collection where the customer id equals '71909' and the ticket price element equals 55:

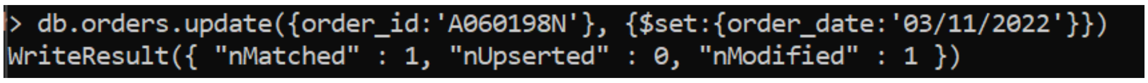


```
> db.orders.find({ customer_id: '71909', tickets: { $elemMatch: { price: 55 } } }).pretty()
{
        "_id" : ObjectId("6374e5cb0eafe91a5c550338"),
        "order_id" : "A060198N",
        "customer_id" : "71909",
        "order_date" : "02/11/2022",
        "ticket_quantity" : 4,
        "tickets" : [
                {
                        "ticket_id" : "8N3L4M",
                        "seat_number" : "84",
                        "price" : 55
                },
                {
                        "ticket_id" : "9K1M0L",
                        "seat_number" : "85",
                        "price" : 55
                },
                {
                        "ticket_id" : "7B4S3N",
                        "seat_number" : "86",
                        "price" : 55
                },
                {
                        "ticket_id" : "1A2H0B",
                        "seat_number" : "87",
                        "price" : 55
                }
        ]
}
```

Figure 26: Orders - ($elemMatch)

Updating the Database:

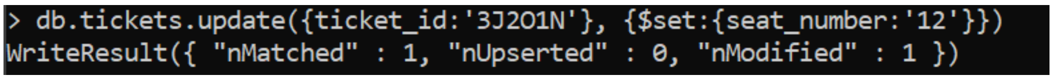
1. Update the order collection to modify the document with order_id: 'A060198N' from order_date '02/11/2022' to '03/11/2022':



```
> db.orders.update({order_id:'A060198N'}, {$set:{order_date:'03/11/2022'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Figure 27: Orders - Update

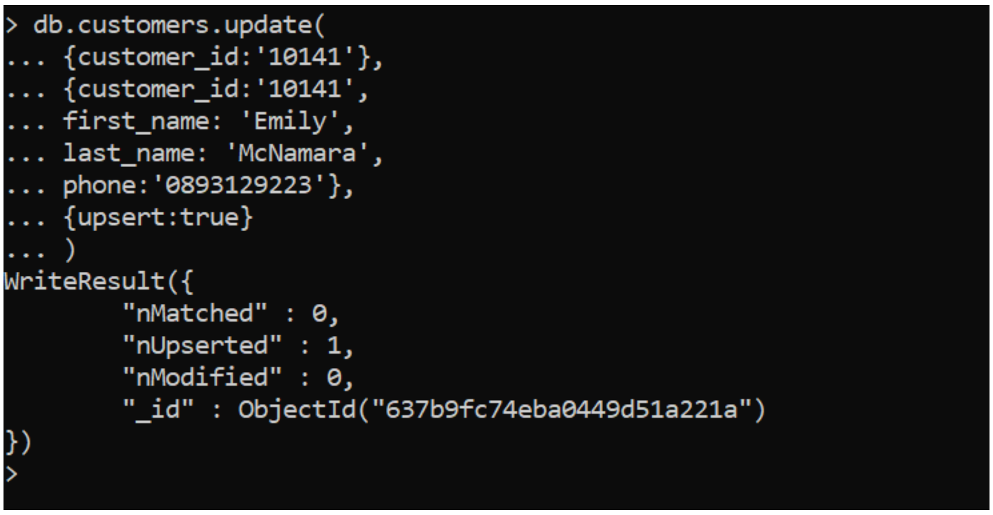
2. Update the ticket collection to modify the document with ticket_id: '3J2O1N' from seat_number '11' to '12':



```
> db.tickets.update({ticket_id:'3J2O1N'}, {$set:{seat_number:'12'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

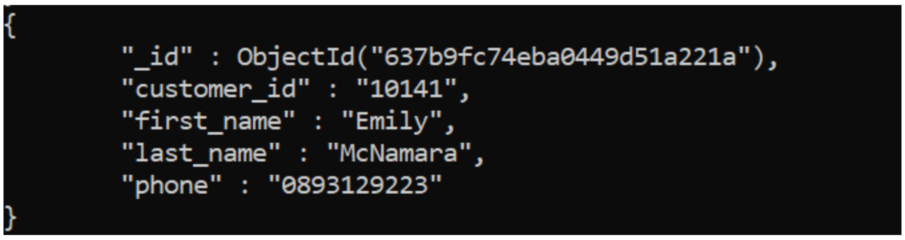
Figure 28: Tickets - Update

3. Update the document with customer_id: in the customer collection and if it doesn't exist then create a new document:



```
> db.customers.update(
... {customer_id:'10141'},
... {customer_id:'10141',
... first_name: 'Emily',
... last_name: 'McNamara',
... phone:'0893129223'},
... {upsert:true}
... )
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : ObjectId("637b9fc74eba0449d51a221a")
})
>
```

Figure 29: Customers – Update



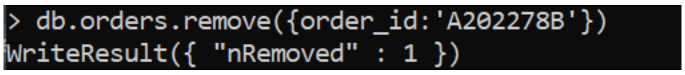
```
{
        "_id" : ObjectId("637b9fc74eba0449d51a221a"),
        "customer_id" : "10141",
        "first_name" : "Emily",
        "last_name" : "McNamara",
        "phone" : "0893129223"
}
```

Figure 30: Customers – Upserted document

Deleting in the Database:

Florence + The Machine had to call off all tour dates due to an injury. All the orders and tickets relevant to Florence + The Machine events needed to be removed from the database.

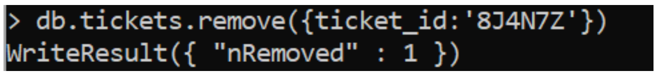
1. Remove the document in the order collection with order_id equal to 'A202278B':



```
> db.orders.remove({order_id:'A202278B'})
WriteResult({ "nRemoved" : 1 })
```

Figure 31: Orders – Delete

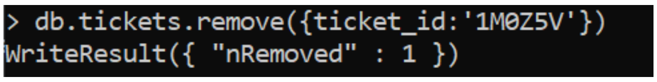
2. Remove the document in the ticket collection with ticket_id equal to ‘8J4N7Z’:



```
> db.tickets.remove({ticket_id:'8J4N7Z'})
WriteResult({ "nRemoved" : 1 })
```

Figure 32: Tickets - Deletion 1

3. Remove the document in the ticket collection with ticket_id equal to ‘1M0Z5V’:

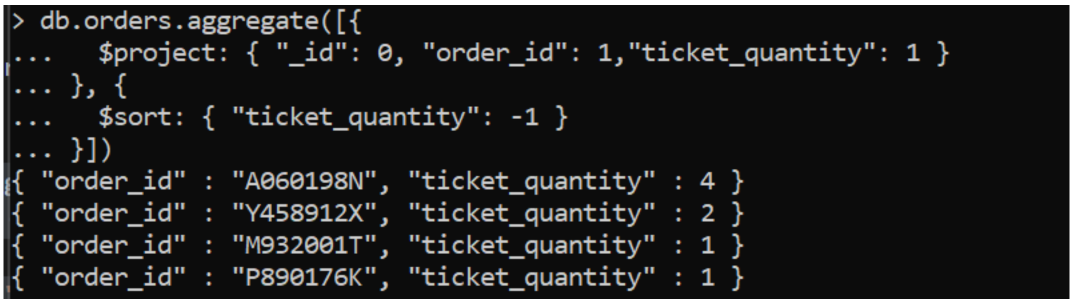


```
> db.tickets.remove({ticket_id:'1M0Z5V'})
WriteResult({ "nRemoved" : 1 })
```

Figure 33: Tickets - Deletion 2

Aggregation

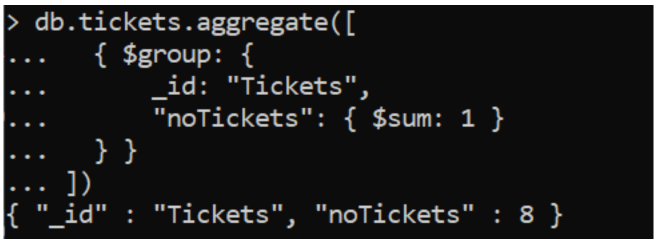
1. Sort documents in the order collection in terms of ‘ticket_quantity’ values in descending order:



```
> db.orders.aggregate([{
...   $project: { "_id": 0, "order_id": 1,"ticket_quantity": 1 }
... }, {
...   $sort: { "ticket_quantity": -1 }
... }])
{ "order_id" : "A060198N", "ticket_quantity" : 4 }
{ "order_id" : "Y458912X", "ticket_quantity" : 2 }
{ "order_id" : "M932001T", "ticket_quantity" : 1 }
{ "order_id" : "P890176K", "ticket_quantity" : 1 }
```

Figure 34: Orders - Aggregation ($sort)

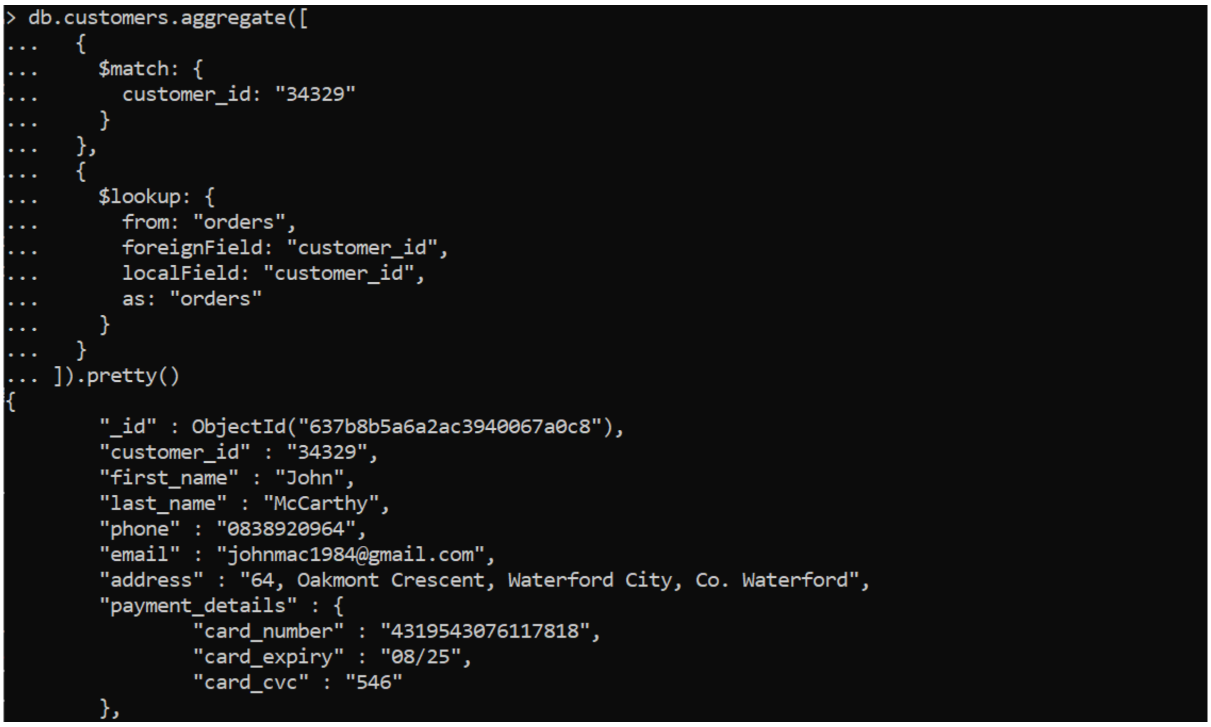
2. Count the number of documents in the ticket collection:



```
> db.tickets.aggregate([
...   { $group: {
...       _id: "Tickets",
...       "noTickets": { $sum: 1 }
...   } }
... ])
{ "_id" : "Tickets", "noTickets" : 8 }
```

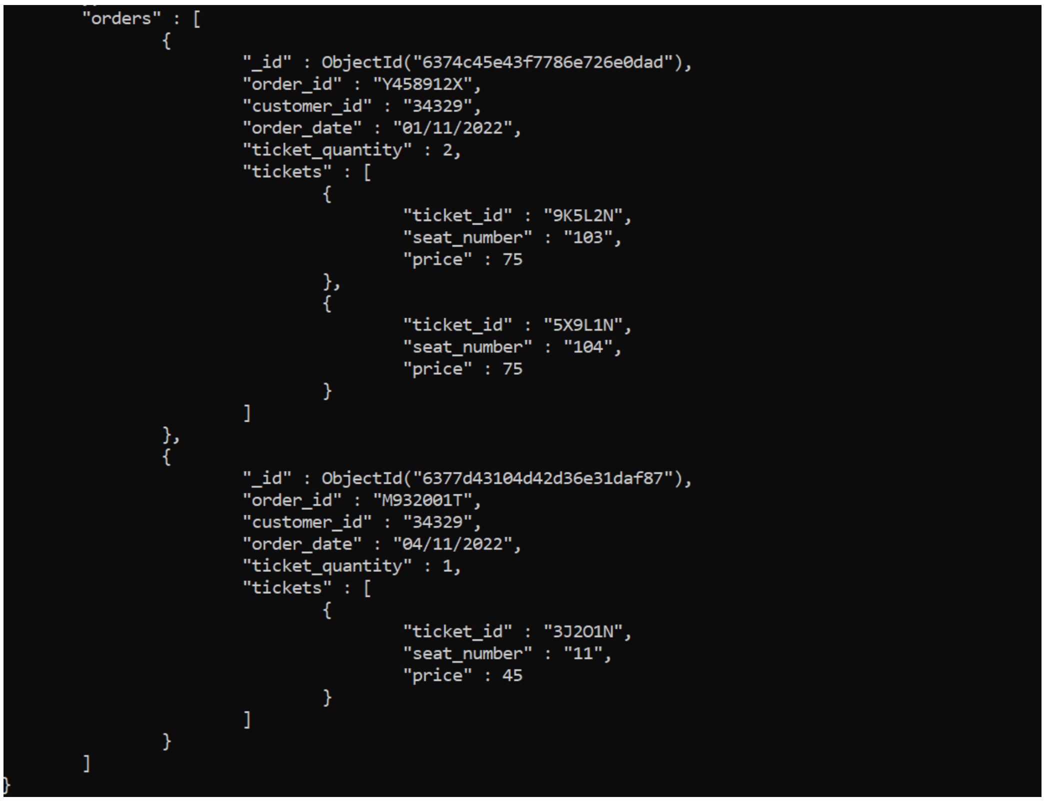
Figure 35: Tickets - Aggregation ($sum)

3. Lookup between the order and customer collections to display documents that match the customer id '34329':



```
> db.customers.aggregate([
...   {
...     $match: {
...       customer_id: "34329"
...     }
...   },
...   {
...     $lookup: {
...       from: "orders",
...       foreignField: "customer_id",
...       localField: "customer_id",
...       as: "orders"
...     }
...   }
... ]).pretty()
{
        "_id" : ObjectId("637b8b5a6a2ac3940067a0c8"),
        "customer_id" : "34329",
        "first_name" : "John",
        "last_name" : "McCarthy",
        "phone" : "0838920964",
        "email" : "johnmac1984@gmail.com",
        "address" : "64, Oakmont Crescent, Waterford City, Co. Waterford",
        "payment_details" : {
                "card_number" : "4319543076117818",
                "card_expiry" : "08/25",
                "card_cvc" : "546"
        },
```

Figure 36: Customer and Order Aggregation Pipelining 1



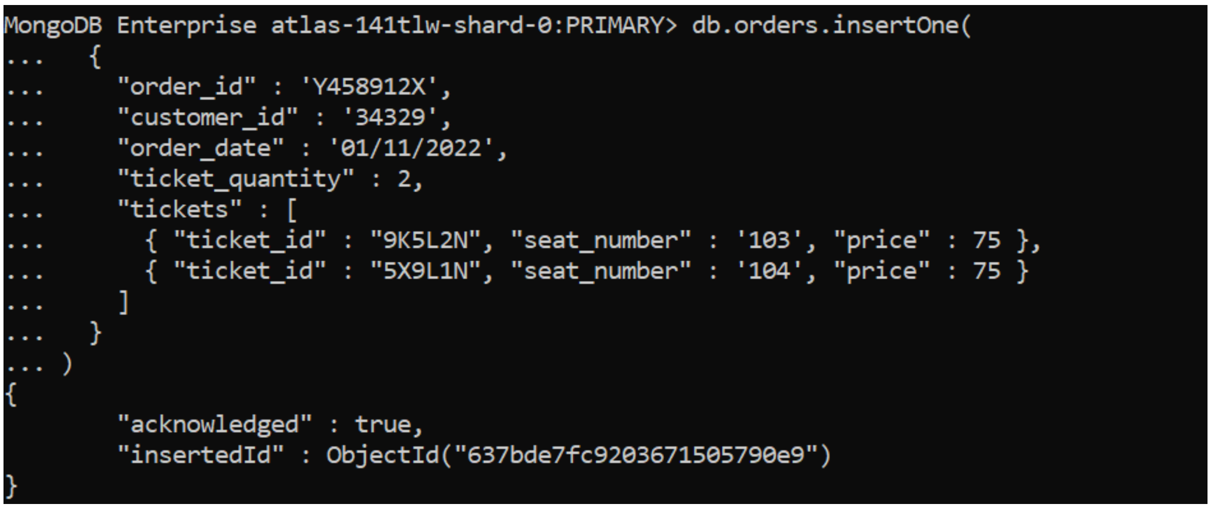
```
        "orders" : [
                {
                        "_id" : ObjectId("6374c45e43f7786e726e0dad"),
                        "order_id" : "Y458912X",
                        "customer_id" : "34329",
                        "order_date" : "01/11/2022",
                        "ticket_quantity" : 2,
                        "tickets" : [
                                {
                                        "ticket_id" : "9K5L2N",
                                        "seat_number" : "103",
                                        "price" : 75
                                },
                                {
                                        "ticket_id" : "5X9L1N",
                                        "seat_number" : "104",
                                        "price" : 75
                                }
                        ]
                },
                {
                        "_id" : ObjectId("6377d43104d42d36e31daf87"),
                        "order_id" : "M932001T",
                        "customer_id" : "34329",
                        "order_date" : "04/11/2022",
                        "ticket_quantity" : 1,
                        "tickets" : [
                                {
                                        "ticket_id" : "3J2O1N",
                                        "seat_number" : "11",
                                        "price" : 45
                                }
                        ]
                }
        ]
}
```

Figure 37: Customer and Order Aggregation Pipelining 2

Cloud Environment

Atlas was used to host the database in a cloud environment. A MongoDB user was set up with a suitable password. A shared cluster was also set up as it was free. The cluster could be connected to using the recently created MongoDB user account. To input data from the command line into this cloud-hosted database, the current directory on the command line had to be set to the local Mongo bin folder. A command from the Atlas web page was copied and executed in the command prompt that prompted the MongoDB user password. Once this was set up, the connection was successful using the mongo shell and data could be inputted from the command line.

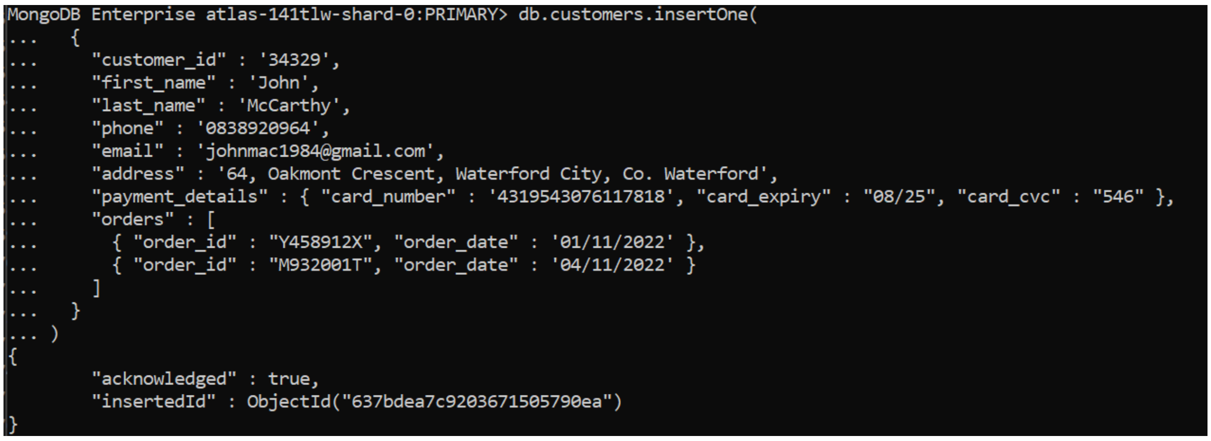
Insert 1 using MongoDB Atlas:



```
MongoDB Enterprise atlas-141tlw-shard-0:PRIMARY> db.orders.insertOne(
...     {
...       "order_id" : 'Y458912X',
...       "customer_id" : '34329',
...       "order_date" : '01/11/2022',
...       "ticket_quantity" : 2,
...       "tickets" : [
...         { "ticket_id" : "9K5L2N", "seat_number" : '103', "price" : 75 },
...         { "ticket_id" : "5X9L1N", "seat_number" : '104', "price" : 75 }
...       ]
...     }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("637bde7fc9203671505790e9")
}
```

Figure 38: MongoDB Atlas - Orders Insert

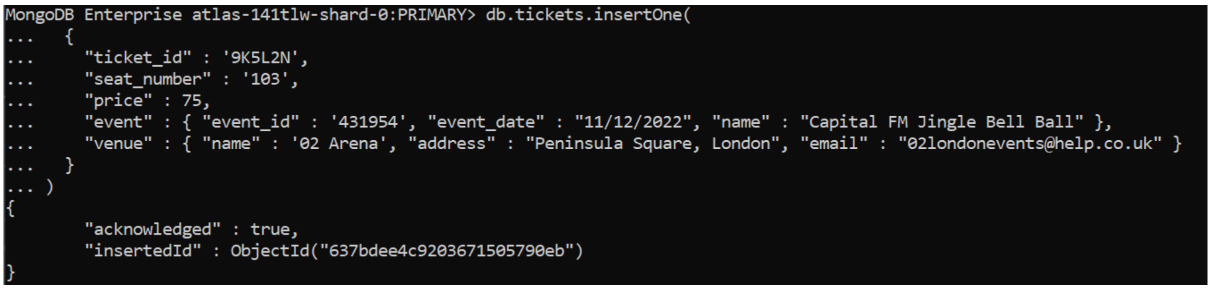
Insert 2 using MongoDB Atlas:



```
MongoDB Enterprise atlas-141tlw-shard-0:PRIMARY> db.customers.insertOne(
...     {
...       "customer_id" : '34329',
...       "first_name" : 'John',
...       "last_name" : 'McCarthy',
...       "phone" : '0838920964',
...       "email" : 'johnmac1984@gmail.com',
...       "address" : '64, Oakmont Crescent, Waterford City, Co. Waterford',
...       "payment_details" : { "card_number" : '4319543076117818', "card_expiry" : "08/25", "card_cvc" : "546" },
...       "orders" : [
...         { "order_id" : "Y458912X", "order_date" : '01/11/2022' },
...         { "order_id" : "M932001T", "order_date" : '04/11/2022' }
...       ]
...     }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("637bdea7c9203671505790ea")
}
```

Figure 39: MongoDB Atlas - Customers Insert

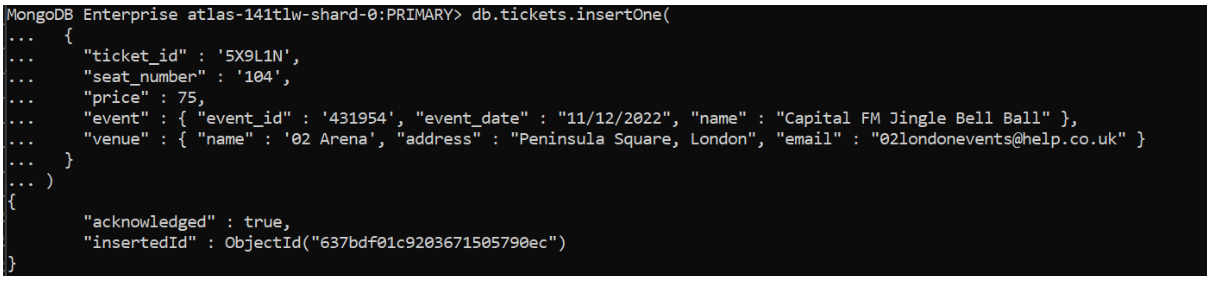
Insert 3 using MongoDB Atlas:



```
MongoDB Enterprise atlas-141tlw-shard-0:PRIMARY> db.tickets.insertOne(
...     {
...         "ticket_id" : '9K5L2N',
...         "seat_number" : '103',
...         "price" : 75,
...         "event" : { "event_id" : '431954', "event_date" : "11/12/2022", "name" : "Capital FM Jingle Bell Ball" },
...         "venue" : { "name" : '02 Arena', "address" : "Peninsula Square, London", "email" : "02londonevents@help.co.uk" }
...     }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("637bdee4c9203671505790eb")
}
```

Figure 40: MongoDB Atlas - Tickets Insert 1

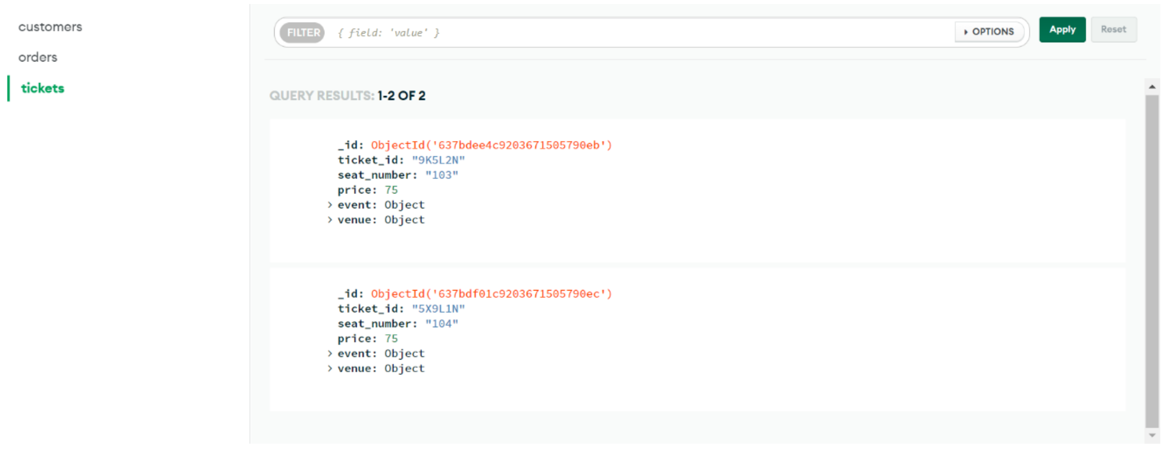
Insert 4 using MongoDB Atlas:



```
MongoDB Enterprise atlas-141tlw-shard-0:PRIMARY> db.tickets.insertOne(
...     {
...         "ticket_id" : '5X9L1N',
...         "seat_number" : '104',
...         "price" : 75,
...         "event" : { "event_id" : '431954', "event_date" : "11/12/2022", "name" : "Capital FM Jingle Bell Ball" },
...         "venue" : { "name" : '02 Arena', "address" : "Peninsula Square, London", "email" : "02londonevents@help.co.uk" }
...     }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("637bdf01c9203671505790ec")
}
```

Figure 41: MongoDB Atlas - Tickets Insert 2

Data stored in MongoDB Atlas:



QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('637bdee4c9203671505790eb')
ticket_id: "9K5L2N"
seat_number: "103"
price: 75
> event: Object
> venue: Object

_id: ObjectId('637bdf01c9203671505790ec')
ticket_id: "5X9L1N"
seat_number: "104"
price: 75
> event: Object
> venue: Object
```

Figure 42: Mongo Atlas - Data Stored

Conclusion

Through the experience of setting up a NoSQL database, it was enjoyable setting up a flexible database that didn't need adhering to strict rules and structures. Once data was added to the database, a large range of useful queries could be used effectively on the database.

As Mongo was used for the database, the data was stored in document-orientated format. This document format of storing data was useful as the data did not need to be stored in table like structures. MongoDB Atlas was a very useful tool for storing data in a cloud environment. Through appropriate set up, insertion commands could be executed on the command line and the inserted data would be stored in a cluster. Data can be accessed very easily through the MongoDB Atlas graphical user interface.

References

1. SearchDataManagement. (n.d.). *What is NoSQL and How do NoSQL Databases Work?* [online] Available at: https://www.techtarget.com/searchdatamanagement/definition/NoSQL-Not-Only-SQL.

2. GeeksforGeeks. (2015). *MongoDB: An introduction*. [online] Available at: https://www.geeksforgeeks.org/mongodb-an-introduction/.