.

**Part A. Solution to the first part of Assignment**

**A1. Data Model as per scenario**

* **Names of Collections and Field Values along with Data Types and Schema**

1. **Input\_User Table:**

Following table provides datatypes and field names for the attributed of input\_user collection.

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| Id | int | Unique identifier for each user |
| User\_id | varchar(50) | UUID as a unique identifier for user |
| Login\_name | varchar(50) | Username used for login |
| Full\_name | varchar(50) | Complete Name of user |
| email | varchar(50) | Email address used for login |
| password | varchar(50) | Hashed password for login |

This schema has three main table input\_users, input\_book and input\_comments. All the description of variable has been provided in the tables.

**Example Document**

{ "\_id": ObjectId("623456789012345678901234"),

“user\_id”: “1”

"login\_name": "kamran",

"full\_name": "Kamran Destgir",

"email": "kamrandestgir@gcu.edu.uk",

"password": "kamran1",

"created\_at": ISODate("2023-03-14T11:00:22Z")}

1. **Input\_Books Table**

Following table provides datatypes and field names for the attributed of input\_books collection.

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| Id | Int | Unique identifier for each book |
| B\_id | Varchar(50) | UUID as unique identifier for book |
| B\_title | varchar(100) | Title of the book |
| B\_Class | Varchar(50) | Category of book |
| Main\_Author | varchar(100) | Author of the book |
| Year | DateTime | Date of publication |
| Total\_count | Int | Books available quantity |
| Price | Int | Books price |
| Summary | Text | Description of the book |
| Sample\_images | varchar(500) | URLs for the sample images of the book |

**Example Document**

{ b\_id: "1",

b\_title: "my book 1",

b\_class: "Arts",

main\_author: "Kamran Destgir",

year: 2001,

total\_count: 2,

price: 100,

summary: "first book of Kamran Destgir on arts and humanities.",

sample\_images: ["https://www.kamrandestgir.com/sample.png", "https://www.kamrandestigr.com/sample.png"]

}

1. **Input\_Comments Table**

Following table provides datatypes and field names for the attributed of input\_comments collection.

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| Id | int | Unique identifier for each comment |
| Com\_id | Varchar(50) | UUID Unique identifier for each comment |
| B\_id | Varchar(50) | ID of the book being commented on |
| user\_id | Varchar(50) | ID of the user making the comment |
| Comment\_text | Varchar(500) | Text content of the comment |
| Date | DateTime | Time the comment was created |

**Example Document**

{

com\_id: "1",

b\_id: "1",

user\_id: "1",

date: new Date("2022-01-01"),

comment\_text: "First comment for book one by kamran"

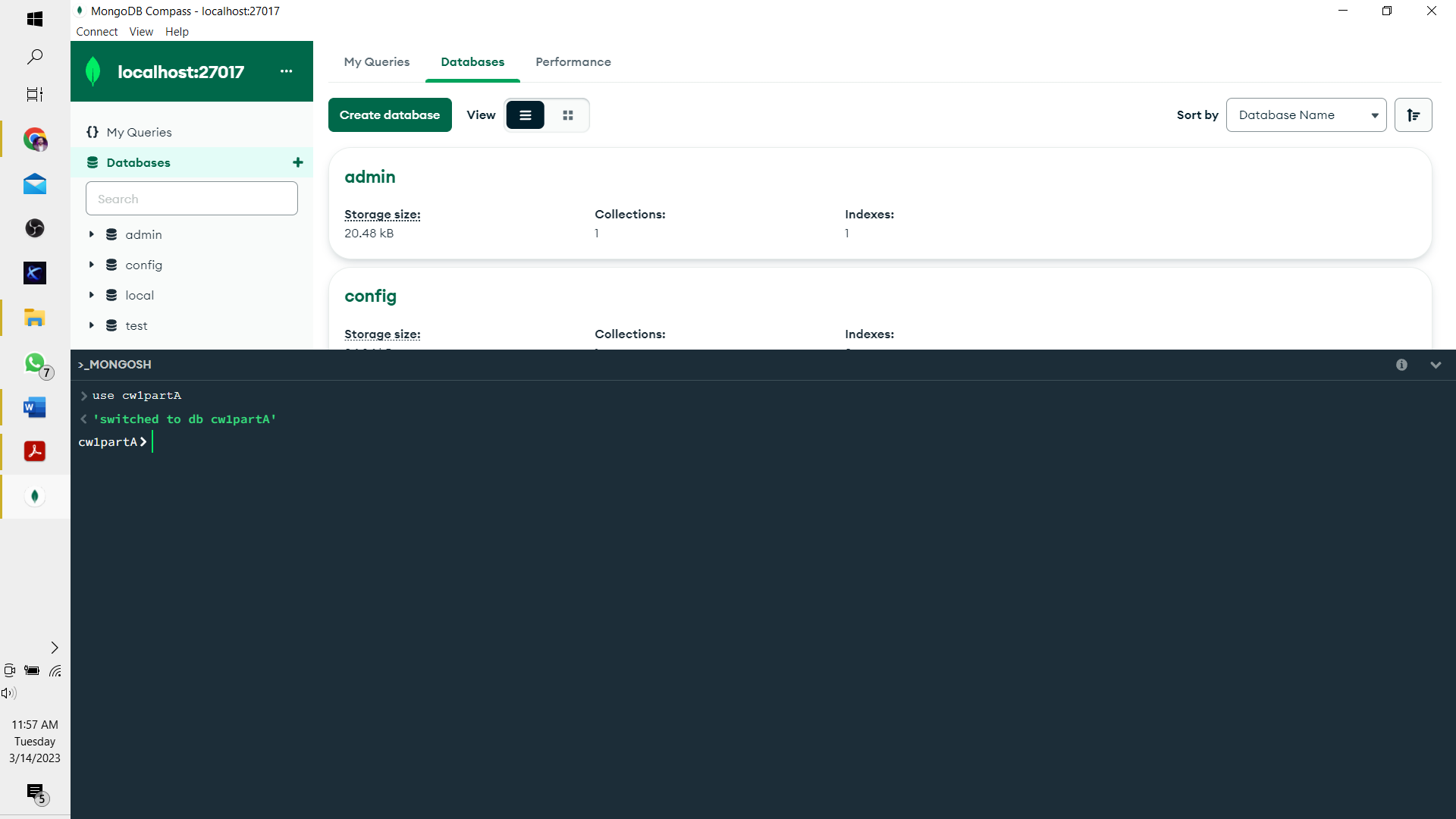
}

**A1.2 Justification**

The opted approach for the bookstore application is based on its requirements. The "input\_books" collection is selected because it can organize and manage book information effectively, allowing storage of all relevant information in a single document. This collection includes book details like b\_title, book\_class, main authors, year, total count, book summary, and two page images, as well as user comments or responce on the book. The chosen approach enables easy updates to book information and comments. The "input\_user" collection stores information about store users who can add any books, update book count, and leave comments. It's a good choice as it separates and manages user information from book information, allowing easy updates and user authentication for actions. In this scenario, the use of references or embedding isn't necessary as book and user information is simple and can be represented effectively in single collections.

**A2. Creating the database**

use('cw1PartA') #this command “use” is used to create a database in mongoDB



db.createCollection("input\_user") #This command will ceate a new collection in database

Graphical user interface, text, application

Description automatically generated

1. **Code for Input\_user:**

#In this code input\_user, I have opted a method to input many user through single prompt. insertMany is used for this purpose. Here total five users with login\_name, full\_name, email and password have been added.

db.input\_user.insertMany([{

user\_id: "1",

login\_name: "karman",

full\_name: "Kamran Destgir",

email: " kamrandestgir@gcu.edu.uk ",

password: " kamran1"

},

{

user\_id: "2",

login\_name: "karman2",

full\_name: "Kamran Destgir2",

email: " kamrandestgir2@gcu.edu.uk ",

password: " kamran2"

},

{

user\_id: "3",

login\_name: "karman3",

full\_name: "Kamran Destgir3",

email: " kamrandestgir3@gcu.edu.uk ",

password: " kamran3"

},{

user\_id: "4",

login\_name: "karman4",

full\_name: "Kamran Destgir4",

email: " kamrandestgir4@gcu.edu.uk ",

password: " kamran4"

},{

user\_id: "5",

login\_name: "karman5",

full\_name: "Kamran Destgir5",

email: " kamrandestgir5@gcu.edu.uk ",

password: " kamran5"

}])

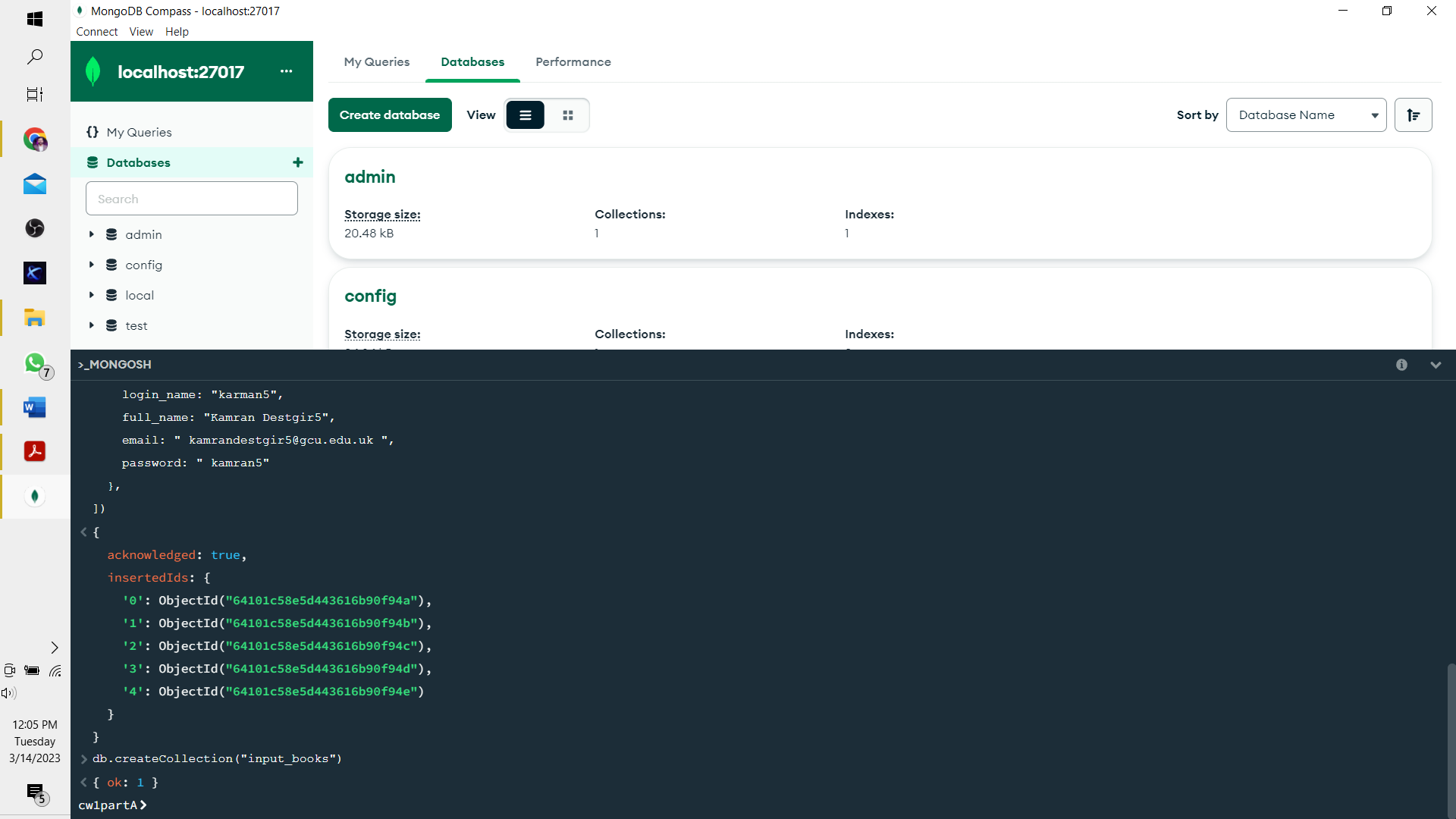
#output for the above code given below

Graphical user interface, text, application

Description automatically generated

1. **Code for input\_books:**

db.createCollection("input\_books") #This command creats a new collection in database



#In this code input\_books, I have opted a method to input many books through single prompt. insertMany is used for this purpose.

db.input\_books.insertMany([

{

b\_id: "1",

b\_title: "my book 1",

b\_class: "Arts",

main\_author: "Kamran Destgir",

year: 2001,

total\_count: 2,

price: 100,

summary: "first book of Kamran Destgir on arts and humanities",

samplePagesImages: ["https://www.kamrandestgir.com/sample.png", "https://www.kamrandestigr.com/sample.png"]

},

{

b\_id: "2",

b\_title: "my book 2",

b\_class: "Arts",

main\_author: "Kamran Destgir",

year: 2002,

total\_count: 2,

price: 100,

summary: "second book of Kamran Destgir on arts and humanities",

samplePagesImages: ["https://www.kamrandestgir.com/sample.png", "https://www.kamrandestigr.com/sample.png"]

},

{

b\_id: "3",

b\_title: "my book 3",

b\_class: "Arts",

main\_author: "Kamran Destgir",

year: 2003,

total\_count: 3,

price: 100,

summary: "thirs book of Kamran Destgir on arts and humanities",

samplePagesImages: ["https://www.kamrandestgir.com/sample.png", "https://www.kamrandestigr.com/sample.png"]

},

{

b\_id: "4",

b\_title: "my book 4",

b\_class: "Science",

main\_author: "Kamran Destgir",

year: 2004,

total\_count: 4,

price: 100,

summary: "fourth book of Kamran Destgir on arts and humanities",

samplePagesImages: ["https://www.kamrandestgir.com/sample.png", "https://www.kamrandestigr.com/sample.png"]

},

{

b\_id: "5",

b\_title: "my book 5",

b\_class: "Science",

main\_author: "Adam",

year: 2005,

total\_count: 5,

price: 500,

summary: "first book of Adam on Science",

samplePagesImages: ["https://www.adam.com/sample.png", "https://www.adam.com/sample.png"]

},

{

b\_id: "6",

b\_title: "my book 6",

b\_class: "Science",

main\_author: "Adam",

year: 2006,

total\_count: 6,

price: 600,

summary: "second book of adam on science",

samplePagesImages: ["https://www.adam.com/sample.png", "https://www.adam.com/sample.png"]

},

{

b\_id: "7",

b\_title: "my book 7",

b\_class: "Science",

main\_author: "Adam",

year: 2007,

total\_count: 7,

price: 700,

summary: "thirs book of adam on science",

samplePagesImages: ["https://www.adam.com/sample.png", "https://www.adam.com/sample.png"]

},

{

b\_id: "8",

b\_title: "my book 8",

b\_class: "Medical",

main\_author: "Alex",

year: 2008,

total\_count: 8,

price: 800,

summary: "first book of Alex on health",

samplePagesImages: ["https://www.alex.com/sample.png", "https://www.alex.com/sample.png"]

},

{

b\_id: "9",

b\_title: "my book 9",

b\_class: "Health",

main\_author: "Alex",

year: 2009,

total\_count: 9,

price: 900,

summary: "second book of alex on health",

samplePagesImages: ["https://www.alex.com/sample.png", "https://www.alex.com/sample.png"]

},

{

b\_id: "10",

b\_title: "my book 10",

b\_class: "Health",

main\_author: "Alex",

year: 2010,

total\_count: 10,

price: 1000,

summary: "thirs book of alex on science",

samplePagesImages: ["https://www.alex.com/sample.png", "https://www.alex.com/sample.png"]

}])

#output for the above code given below. It shows the object Id for 10 books

Graphical user interface, text, application

Description automatically generated

1. **Code for input\_comments**

db.createCollection("input\_comments") #This command will ceate a new collection in database

Graphical user interface, text, application

Description automatically generated

#In this code of input\_comments, I have opted a method to input many comments through single prompt. insertMany is used for this purpose. Total 10 commets have been added as per the requirements provided in the project description. Total five book have comments where two book are with more than three comments.

db.input\_comments.insertMany([ {

com\_id: "1",

b\_id: "1",

user\_id: "1",

date: new Date("2023-03-03"),

comment\_text: "first comment on book id 1 by user 1"

},{

com\_id: "2",

b\_id: "1",

user\_id: "1",

date: new Date("2022-03-04"),

comment\_text: "second comment on book id 1 by user 1"

},

{

com\_id: "3",

b\_id: "1",

user\_id: "2",

date: new Date("2023-03-05"),

comment\_text: "third comment on book id 1 by user id 2"

},

{

com\_id: "4",

b\_id: "2",

user\_id: "2",

date: new Date("2023-03-06"),

comment\_text: "first comment on book id 2 by user 2"

},

{

com\_id: "5",

b\_id: "2",

user\_id: "2",

date: new Date("2023-03-03"),

comment\_text: "second comment on book id 2 by user id 2"

},

{

com\_id: "6",

b\_id: "2",

user\_id: "3",

date: new Date("2023-03-03"),

comment\_text: "third comment on book id 2 by user id 3"

},{

com\_id: "7",

b\_id: "3",

user\_id: "4",

date: new Date("2023-03-03"),

comment\_text: "this book is amazing"

},

{

com\_id: "8",

b\_id: "4",

user\_id: "4",

date: new Date("2023-03-03"),

comment\_text: "I love this book"

},

{

com\_id: "9",

b\_id: "5",

user\_id: "5",

date: new Date("2023-03-03"),

comment\_text: "there is no specific useful knowledge in this book"

},

{

com\_id: "10",

b\_id: "6",

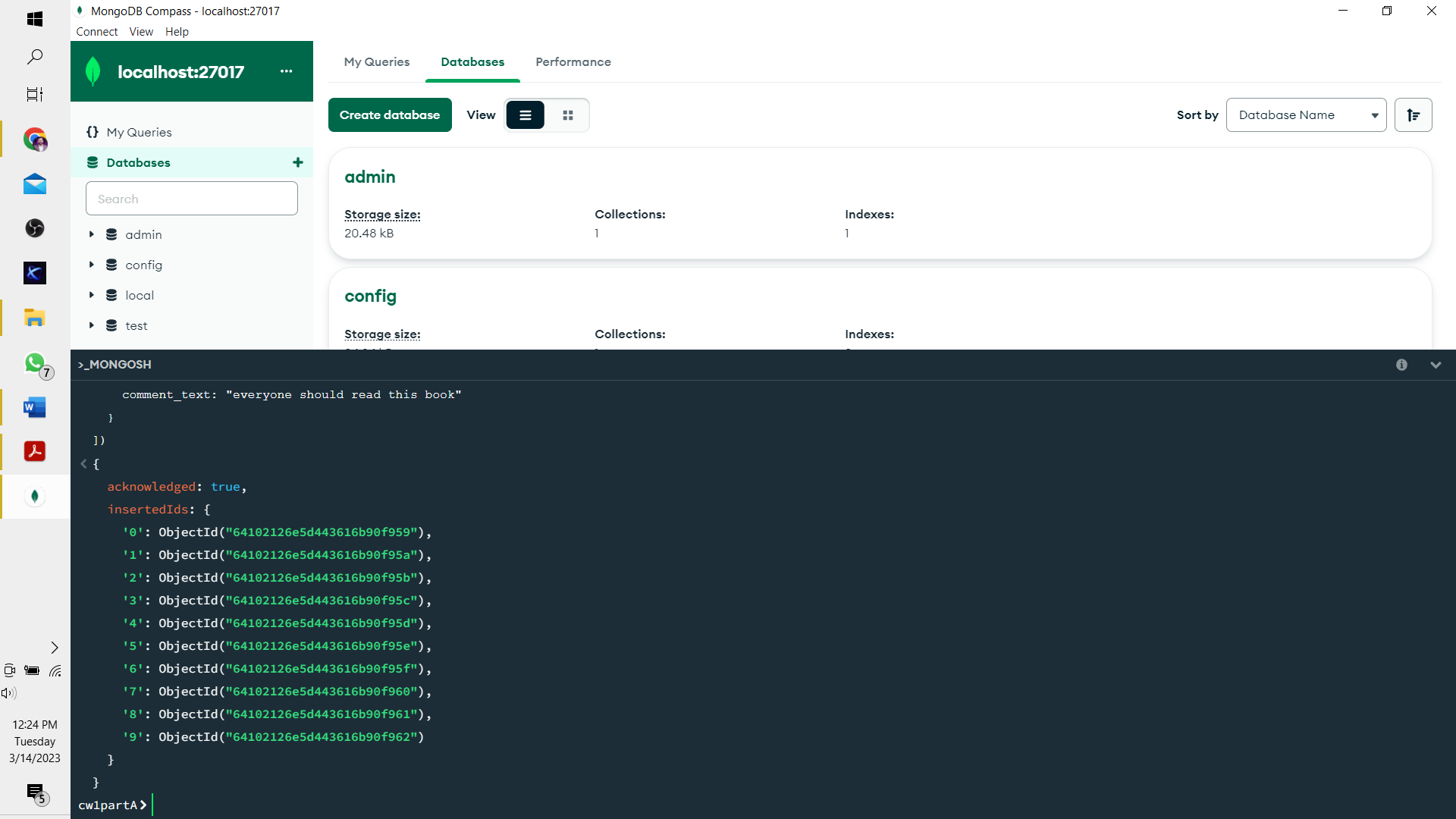
user\_id: "3",

date: new Date("2023-03-03"),

comment\_text: "everyone should read this book"

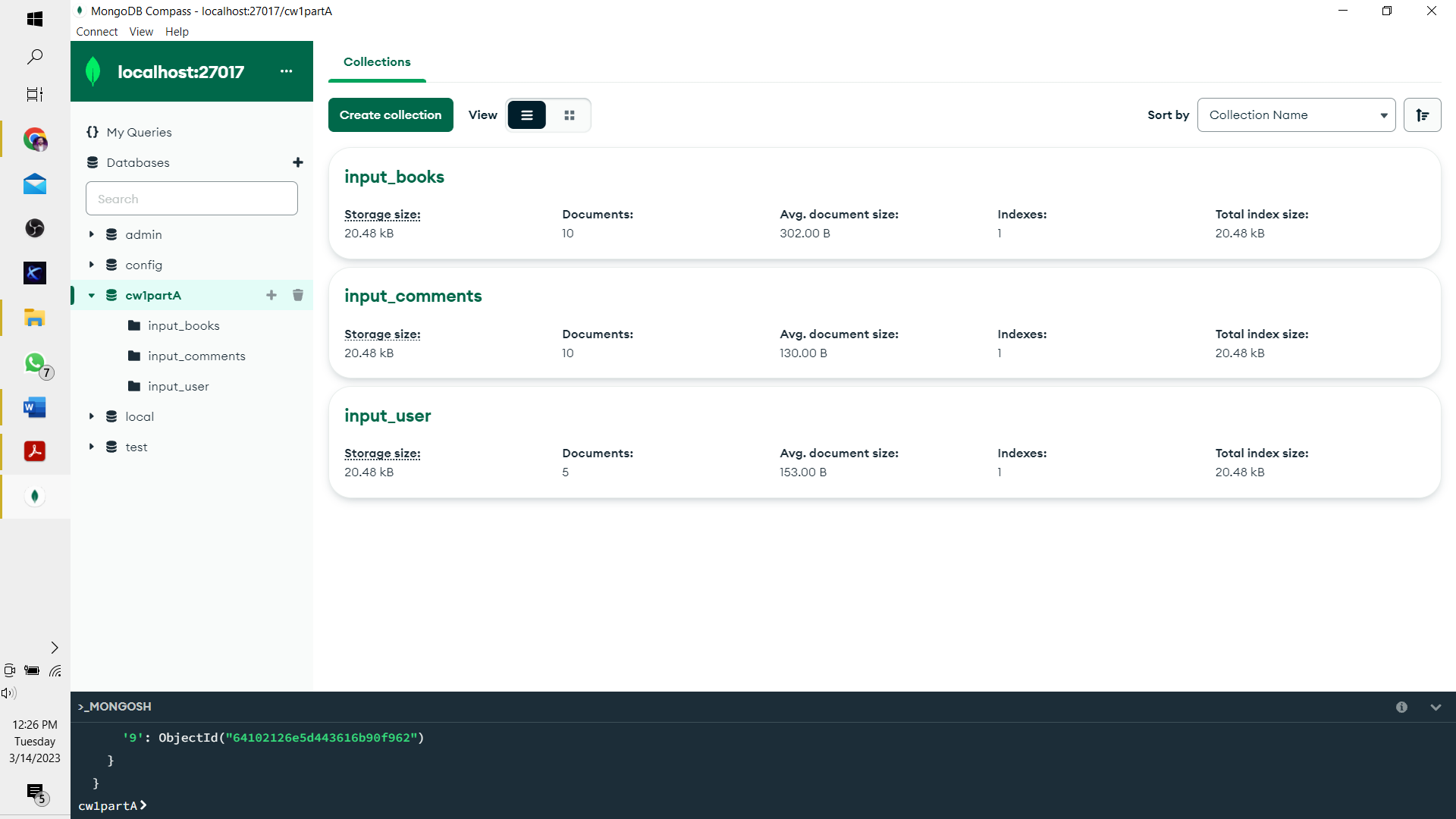
}])

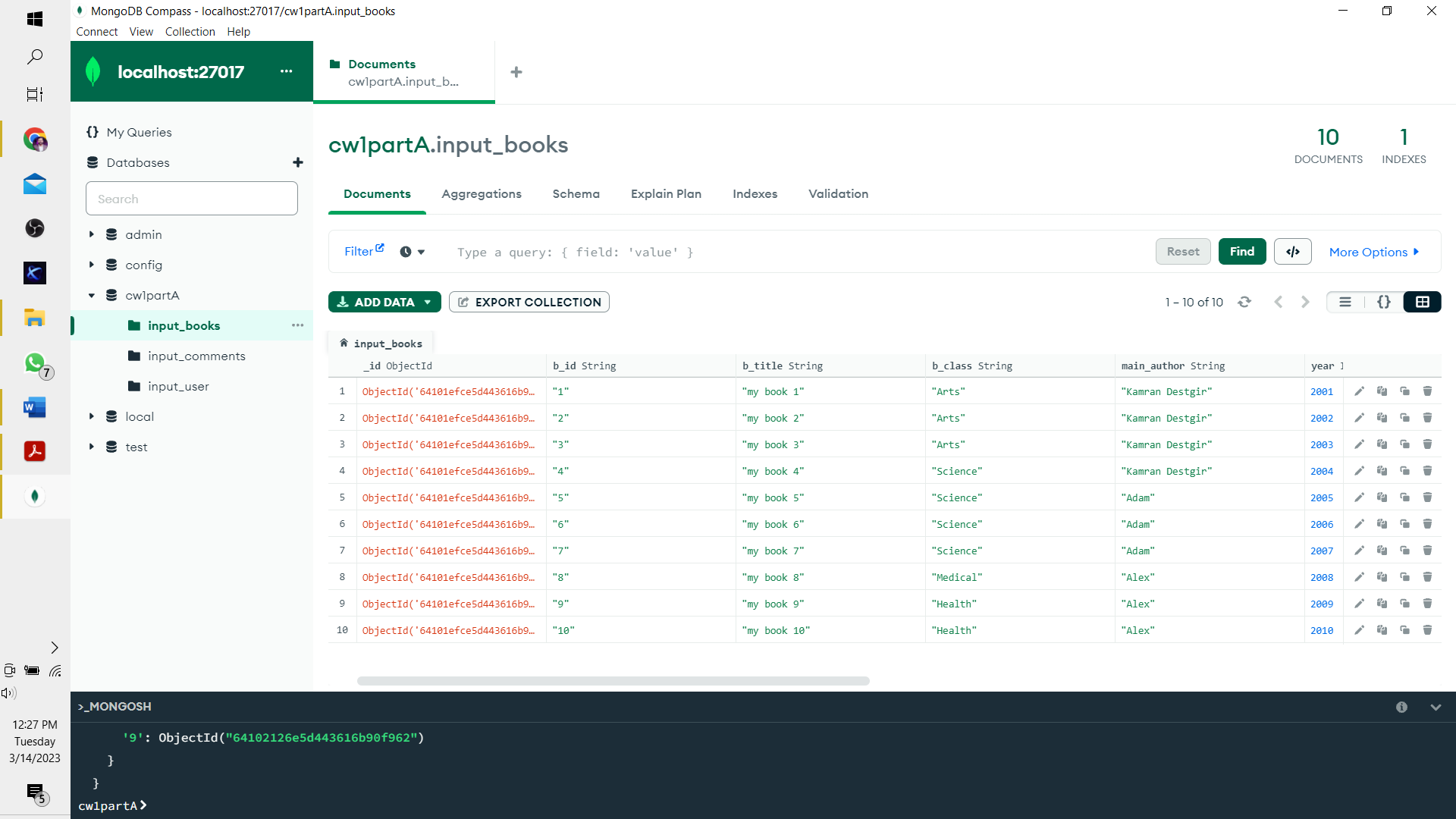
#output for the above code given below. It shows the object Id for 10 comments.



**Overview of MongoDB for this report:**

This is the additional section for the report, which provides an overview of MongoDB before running queries. In the figure below, three collections can be observed as defined in above codes.



Figures below is the detailed view of the three collections. It can be observed that collection attributes are as per the above code

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

**A3. Queries and Their Response:**

As per the requirements, queries have been established below, where the resulted output is also attached

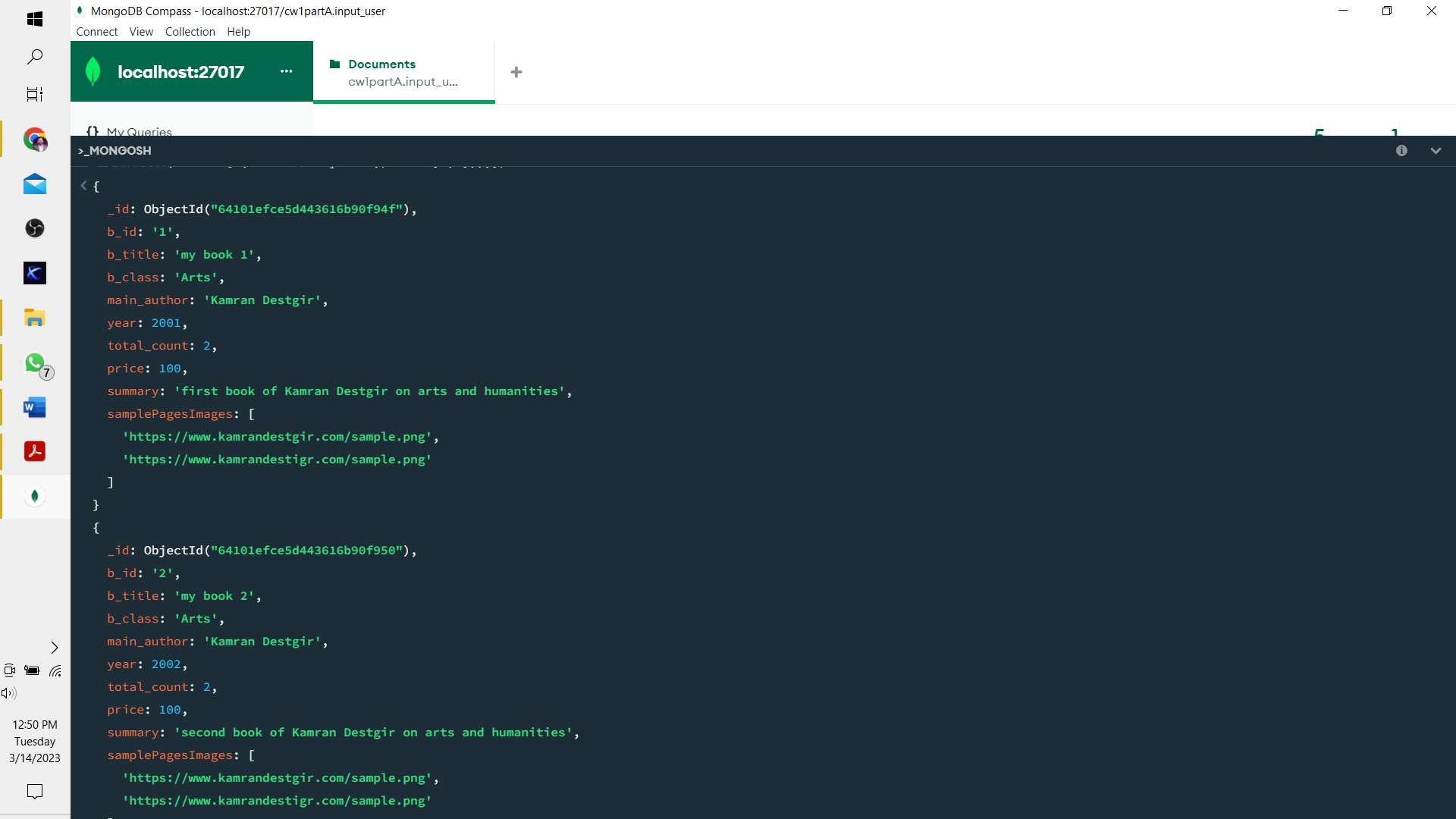
**First Query:**

This query searches for the book class “Arts” from the year 2001 to 2002. The screenshot of output indicates that total 2 books are present in the database with such details.

db.input\_books.aggregate([{ $match: { b\_class: "Arts", $expr: {

$and: [{ $gte: [ { $toInt: "$year" }, 2001 ] },

{ $lte: [ { $toInt: "$year" }, 2002 ] } ]}}}])



**Second Query:**

This query searches for a specific autor present in the database. In the output it can be observed that book 8,9, and 10 have been authord by the Alex.

db.input\_books.find({ main\_author: "Alex" })

Text

Description automatically generated Text

Description automatically generated

**Third Query:**

This query inputs a new comment by the user 1. Who is Kamran Destgir. This comment has been added to input\_comments collection. Which can be observed from provided JSON file as well. Output of this query indicates that the result for this query has been successfully updated.

db.input\_comments.insert({

com\_id: "1", book\_id: "1", user\_id: "1", date: new Date(),

comment\_text: "New Comment added by Kamran Destgir" })

Graphical user interface, text, application

Description automatically generated

**Fourth Query:**

This query look for the certain word in the summary and output of this query indicates that is has identified that from database.

db.input\_books.aggregate([

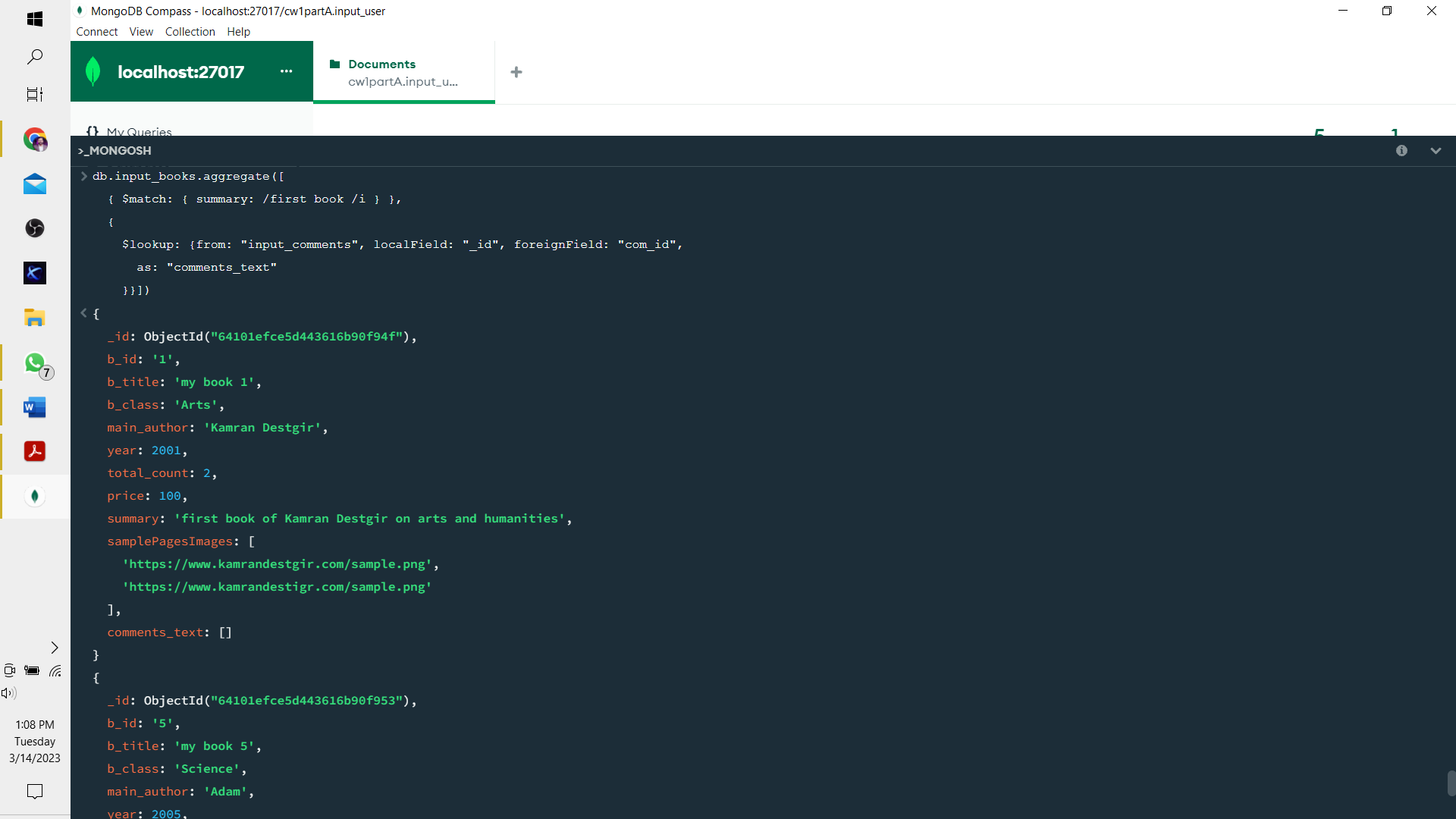
{ $match: { summary: /first book /i } },

{

$lookup: {from: "input\_comments", localField: "\_id", foreignField: "com\_id",

as: "comments\_text"

}}])



Graphical user interface, text

Description automatically generated

**Part B. Creating a graph database**

This table provides complete cypher step by step which I have used on Neo4J.

|  |  |
| --- | --- |
| **Cypher** | **Output** |
| CREATE DATABASE cw1PartB | (1 system update, no records) |
| CREATE (:Airport {name: "London"})  CREATE (:Airport {name: "Scotland"})  CREATE (:Airport {name: "Lahore"})  CREATE (:Airport {name: "Karachi"})  CREATE (:Airport {name: "MarryLand"}) | Added 5 labels, created 5 nodes, set 5 properties, completed after 26 ms. |
| CREATE (:Flight\_Detail {airline\_name: "Emirates", date: "2023-03-3", distance: 150, duration: 520})  CREATE (:Flight\_Detail { airline\_name: "Qatar", date: "2023-03-4", distance: 1600, duration: 12})  CREATE (:Flight\_Detail { airline\_name: "PIA", date: "2023-03-5", distance: 1700, duration: 360})  CREATE (:Flight\_Detail { airline\_name: "SialAir", date: "2023-03-6", distance: 1800, duration: 365})  CREATE (:Flight\_Detail { airline\_name: "Serene", date: "2023-03-7", distance: 1900, duration: 370})  CREATE (:Flight\_Detail { airline\_name: "AirBlue", date: "2023-03-11", distance: 2000, duration: 375})  CREATE (:Flight\_Detail { airline\_name: "Queen", date: "2023-04-12", distance: 2100, duration: 308})  CREATE (:Flight\_Detail { airline\_name: "Airbuss", date: "2023-05-13", distance: 2200, duration: 370})  CREATE (:Flight\_Detail { airline\_name: "Boing", date: "2023-06-14", distance: 2300, duration: 300})  CREATE (:Flight\_Detail { airline\_name: "SaudiAir", date: "2023-07-15", distance: 2400, duration: 290}) | Added 10 labels, created 10 nodes, set 40 properties, completed after 150 ms. |
| CREATE (:Tickets\_node {passenger\_class: "First\_Class", price: 750})  CREATE (:Tickets\_node { passenger\_class: "Economy Class", price: 450}) | Added 2 labels, created 2 nodes, set 4 properties, completed after 65 ms. |
| **Flight Relation with Airline Tickets** | |
| MATCH (q: Flight\_Detail {date: "2023-03-3"})  MATCH (w:Tickets\_node { passenger\_class: "First\_Class"})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 19 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-3"})  MATCH (w:Tickets\_node { passenger\_class: " Economy Class "})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 14 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-4"})  MATCH (w:Tickets\_node { passenger\_class: "First\_Class"})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 22 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-4"})  MATCH (w:Tickets\_node { passenger\_class: "Economy Class"})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 19 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-5"})  MATCH (w:Tickets\_node { passenger\_class: "First\_Class"})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 24 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-5"})  MATCH (w:Tickets\_node { passenger\_class: "Economy Class"})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 19 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-6"})  MATCH (w:Tickets\_node { passenger\_class: "First\_Class"})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 19 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-6"})  MATCH (w:Tickets\_node { passenger\_class: " Economy Class "})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 15 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-7"})  MATCH (w:Tickets\_node { passenger\_class: "First\_Class"})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 14 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-7"})  MATCH (w:Tickets\_node { passenger\_class: " Economy Class "})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 15 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-11"})  MATCH (w:Tickets\_node { passenger\_class: "First\_Class"})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 15 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-11"})  MATCH (w:Tickets\_node { passenger\_class: " Economy Class "})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 14 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-12"})  MATCH (w:Tickets\_node { passenger\_class: "First\_Class"})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 14 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-12"})  MATCH (w:Tickets\_node { passenger\_class: " Economy Class "})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 14 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-13"})  MATCH (w:Tickets\_node { passenger\_class: "First\_Class"})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 14 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-13"})  MATCH (w:Tickets\_node { passenger\_class: " Economy Class "})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 15 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-14"})  MATCH (w:Tickets\_node { passenger\_class: "First\_Class"})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 29 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-14"})  MATCH (w:Tickets\_node { passenger\_class: " Economy Class "})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 28 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-15"})  MATCH (w:Tickets\_node { passenger\_class: "First\_Class"})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 29 ms. |
| MATCH (q: Flight\_Detail {date: "2023-03-15"})  MATCH (w:Tickets\_node { passenger\_class: " Economy Class "})  CREATE (q)-[:HAS]->(w) | Created 2 relationships, completed after 22 ms. |
| **Flight Relation with Airline Tickets [Origin]** | |
| MATCH (q:Flight\_Detail {date: "2023-03-3"})  MATCH (e:Airport {name: "London"})  CREATE (q)-[:ORIGIN]->(e) | Created 6 relationship, completed after 19 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-4"})  MATCH (e:Airport {name: "Scotland"})  CREATE (q)-[:ORIGIN]->(e) | Created 1 relationship, completed after 19 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-5"})  MATCH (e:Airport {name: "Lahore"})  CREATE (q)-[:ORIGIN]->(e) | Created 1 relationship, completed after 17 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-6"})  MATCH (e:Airport {name: "Karachi"})  CREATE (q)-[:ORIGIN]->(e) | Created 1 relationship, completed after 18 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-7"})  MATCH (e:Airport {name: "MarryLand"})  CREATE (q)-[:ORIGIN]->(e) | Created 1 relationship, completed after 16 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-11"})  MATCH (e:Airport {name: "London"})  CREATE (q)-[:ORIGIN]->(e) | Created 2 relationships, completed after 13 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-12"})  MATCH (e:Airport {name: "Scotland"})  CREATE (q)-[:ORIGIN]->(e) | Created 2 relationships, completed after 13 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-13"})  MATCH (e:Airport {name: "Lahore"})  CREATE (q)-[:ORIGIN]->(e) | Created 1 relationship, completed after 16 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-14"})  MATCH (e:Airport {name: "Karachi"})  CREATE (q)-[:ORIGIN]->(e) | Created 1 relationship, completed after 16 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-15"})  MATCH (e:Airport {name: "MarryLand"})  CREATE (q)-[:ORIGIN]->(e) | Created 1 relationship, completed after 16 ms. |
| **Flight Relation with Airline Tickets [Destination]** | |
| MATCH (q:Flight\_Detail {date: "2023-03-3"})  MATCH (e:Airport {name: "London"})  CREATE (q)-[:DESTINATION]->(e) | Created 6 relationship, completed after 15 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-4"})  MATCH (e:Airport {name: "Scotland"})  CREATE (q)-[:DESTINATION]->(e) | Created 1 relationship, completed after 18 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-5"})  MATCH (e:Airport {name: "Lahore"})  CREATE (q)-[:DESTINATION]->(e) | Created 1 relationship, completed after 17 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-6"})  MATCH (e:Airport {name: "Karachi"})  CREATE (q)-[:DESTINATION]->(e) | Created 1 relationship, completed after 11 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-7"})  MATCH (e:Airport {name: "MarryLand"})  CREATE (q)-[:DESTINATION]->(e) | Created 1 relationship, completed after 15 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-11"})  MATCH (e:Airport {name: "London"})  CREATE (q)-[:DESTINATION]->(e) | Created 2 relationships, completed after 14 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-12"})  MATCH (e:Airport {name: "Scotland"})  CREATE (q)-[:DESTINATION]->(e) | Created 2 relationships, completed after 19 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-13"})  MATCH (e:Airport {name: "Lahore"})  CREATE (q)-[:DESTINATION]->(e) | Created 1 relationship, completed after 15 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-14"})  MATCH (e:Airport {name: "Karachi"})  CREATE (q)-[:DESTINATION]->(e) | Created 1 relationship, completed after 12 ms. |
| MATCH (q:Flight\_Detail {date: "2023-03-15"})  MATCH (e:Airport {name: "MarryLand"})  CREATE (q)-[:DESTINATION]->(e) | Created 1 relationship, completed after 12 ms. |
|  |  |

**B2. Queries**

1. **First Query:**

MATCH (q)

OPTIONAL MATCH (q)-[w]->(e)

RETURN q, w, e

**Output:**

Graphical user interface

Description automatically generated

**Second Query:**

MATCH (e:Airport {name: " London "})<-[:ORIGIN]-(q:Flight)

RETURN q.airline, q.date, q.distance, q.duration

**Output:**

╒════════════════╤════════════╤════════════╤════════════╕

│"q.airline\_name"│"q.date" │"q.distance"│"q.duration"│

╞════════════════╪════════════╪════════════╪════════════╡

│"AirBlue" │"2023-03-11"│2000 │375 │

├────────────────┼────────────┼────────────┼────────────┤

│"AirBlue" │"2023-03-11"│2000 │375 │

├────────────────┼────────────┼────────────┼────────────┤

│"Emirates" │"2023-03-3" │150 │520 │

├────────────────┼────────────┼────────────┼────────────┤

│"Emirates" │"2023-03-3" │150 │520 │

├────────────────┼────────────┼────────────┼────────────┤

│"Serene" │"2023-03-3" │1900 │370 │

├────────────────┼────────────┼────────────┼────────────┤

│"Qatar" │"2023-03-3" │1600 │12 │

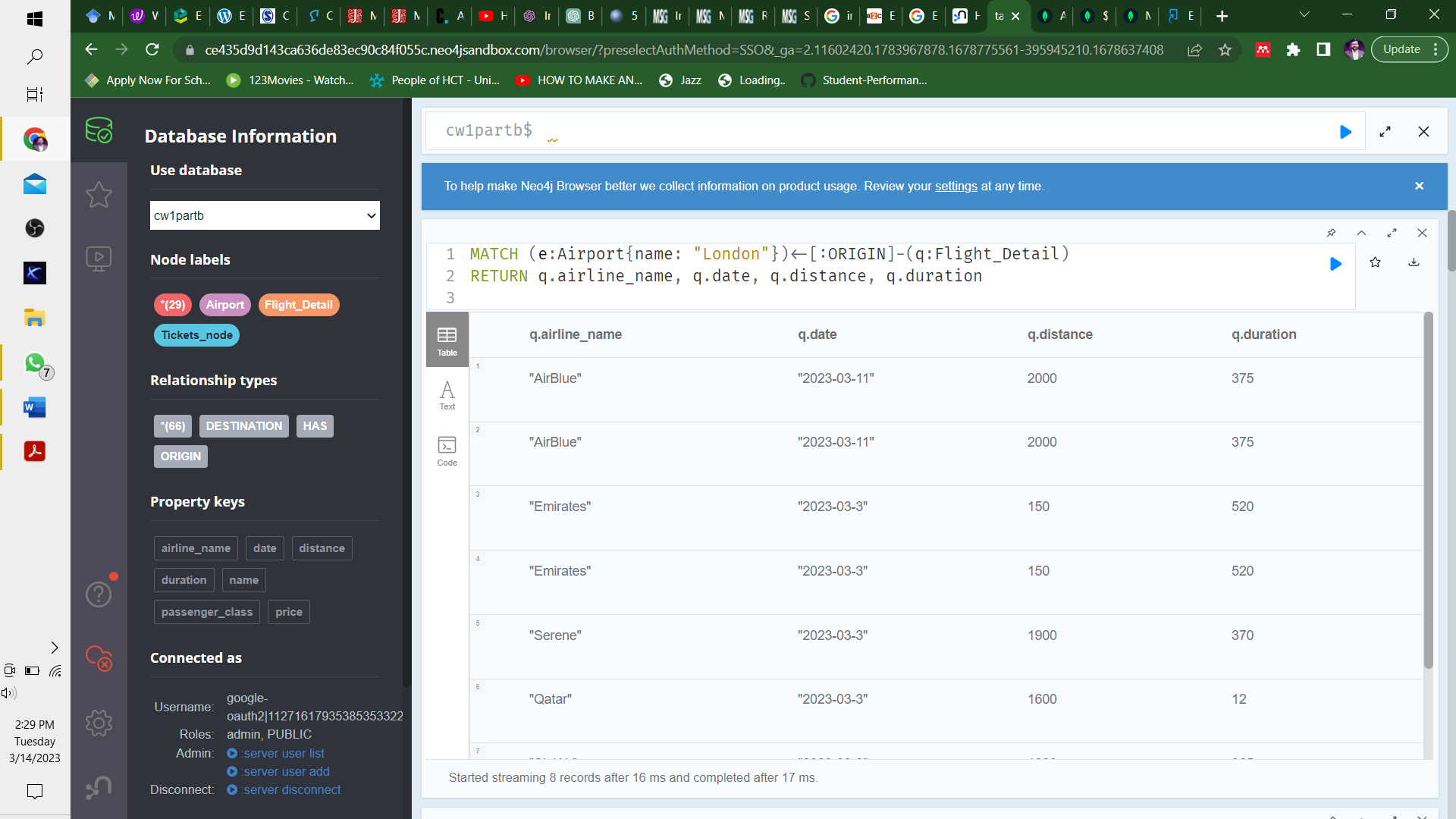
├────────────────┼────────────┼────────────┼────────────┤

│"SialAir" │"2023-03-3" │1800 │365 │

├────────────────┼────────────┼────────────┼────────────┤

│"PIA" │"2023-03-3" │1700 │360 │

└────────────────┴────────────┴────────────┴────────────┘



**Third Query**

MATCH (origin:Airport {name: "Lahoor"})<-[:ORIGIN]-(q:Flight\_Detail)-[:DESTINATION]->(destination:Airport)

RETURN destination.name

**Output:**

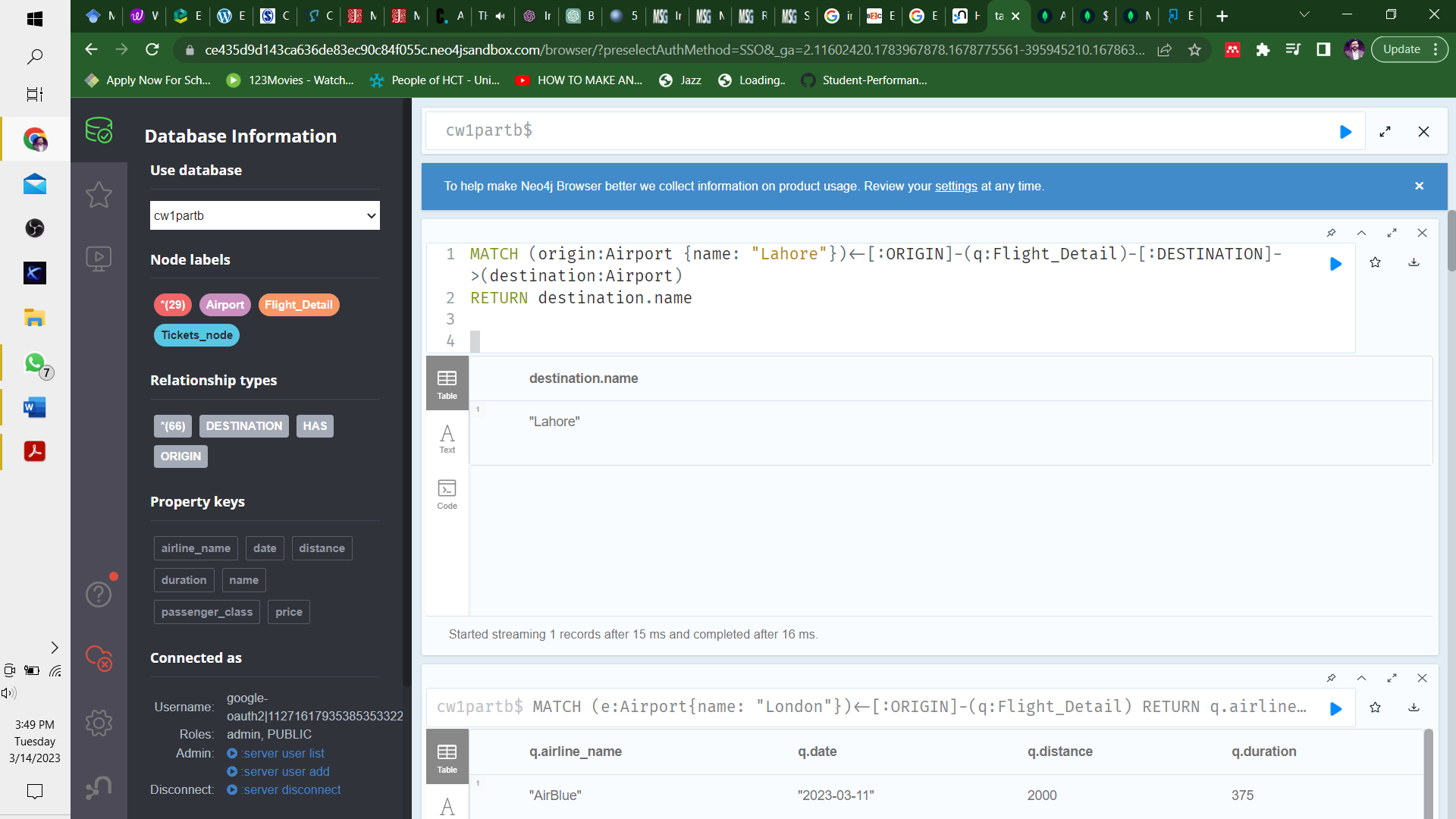
╒══════════════════╕

│"destination.name"│

╞══════════════════╡

│"Lahore" │

└──────────────────┘



**Fourth Query:**

MATCH (q:Flight\_Detail{airline\_name: "Qatar"})

CREATE (q)-[:HAS]->(:Tickets\_node {passenger\_class: "executive class", price: 1100})

**Output:**

Added 2 labels, created 2 nodes, set 4 properties, created 2 relationships, completed after 14 ms.

**Fifth Query:**

MATCH (destination:Airport {name: "Lahore"})<-[to:DESTINATION]-(q:Flight\_Detail)-[has:HAS]->(w:Tickets\_node)

RETURN w. passenger\_class, w.price

**Output:**

╒════════════════════╤═════════╕

│"w. passenger\_class"│"w.price"│

╞════════════════════╪═════════╡

│"Economy Class" │450 │

├────────────────────┼─────────┤

│"Economy Class" │450 │

├────────────────────┼─────────┤

│"Economy Class" │450 │

├────────────────────┼─────────┤

│"Economy Class" │450 │

├────────────────────┼─────────┤

│"First\_Class" │750 │

├────────────────────┼─────────┤

│"First\_Class" │750 │

└────────────────────┴─────────┘

Graphical user interface, text, application, email

Description automatically generated

**References:**

1. <https://www.mongodb.com/docs/manual/aggregation/>
2. <https://www.mongodb.com/docs/v4.0/reference/limits/>