Big Data Platforms Level : M Module Code:MMI227050 Coursework 1 (Re-sit)

**Creating Document and Graph database**

**Mahek kathiriya**

**s2231752**

2023

**PART A: CREATING A DOCUMENT DATABASE**

**Scenario – Online Auction**

This section of the report outlines the design and implementation of a MongoDB document database for an online auction application. The data model is designed to accommodate key use cases and optimize data storage and retrieval within the application.

**A1. Creating the Data Model**

The data model consists of three collections: **auctions**, **users**, and **bids**. Each collection captures specific information related to the online auction scenario. The JSON examples below illustrate the structure of documents within each collection.

**Collection: auctions**

{

"\_id": "<auctionId>",

"itemName": "Laptop",

"category": "Electronics",

"description": "Brand new laptop",

"closingDate": "2023-08-31T18:00:00",

"startingBid": 800,

"status": "open",

"createdBy": "<userId>"

}

**Collection: users**

{

"\_id": "<userId>",

"name": "John Doe",

"email": "john@example.com",

"location": "New York"

}

**Collection: bids**

{

"\_id": "<bidId>",

"auctionId": "<auctionId>",

"bidder": "<userId>",

"dateTime": "2023-08-15T14:30:00",

"amount": 850

}

**Justification**

* **Auctions Collection:** Auction details, including item name, category, description, closing date, starting bid, status, and creator's reference, are embedded for efficient retrieval. Embedding data reduces the need for frequent joins and enhances query performance.
* **Users Collection:** User profiles are embedded to simplify user-related queries and reduce database lookups.
* **Bids Collection:** References to auction and user IDs are utilized to establish relationships between auctions, bidders, and bids, maintaining data integrity.

**A2. Creating the Database**

The MongoDB database **Online\_auction** is created using the provided data model. The following method calls are used to create collections and insert test data:

**Creating the Auctions Collection**

db.auctions.insertOne({

"\_id": "auc001",

"itemName": "Laptop",

"category": "Electronics",

"description": "Brand new laptop",

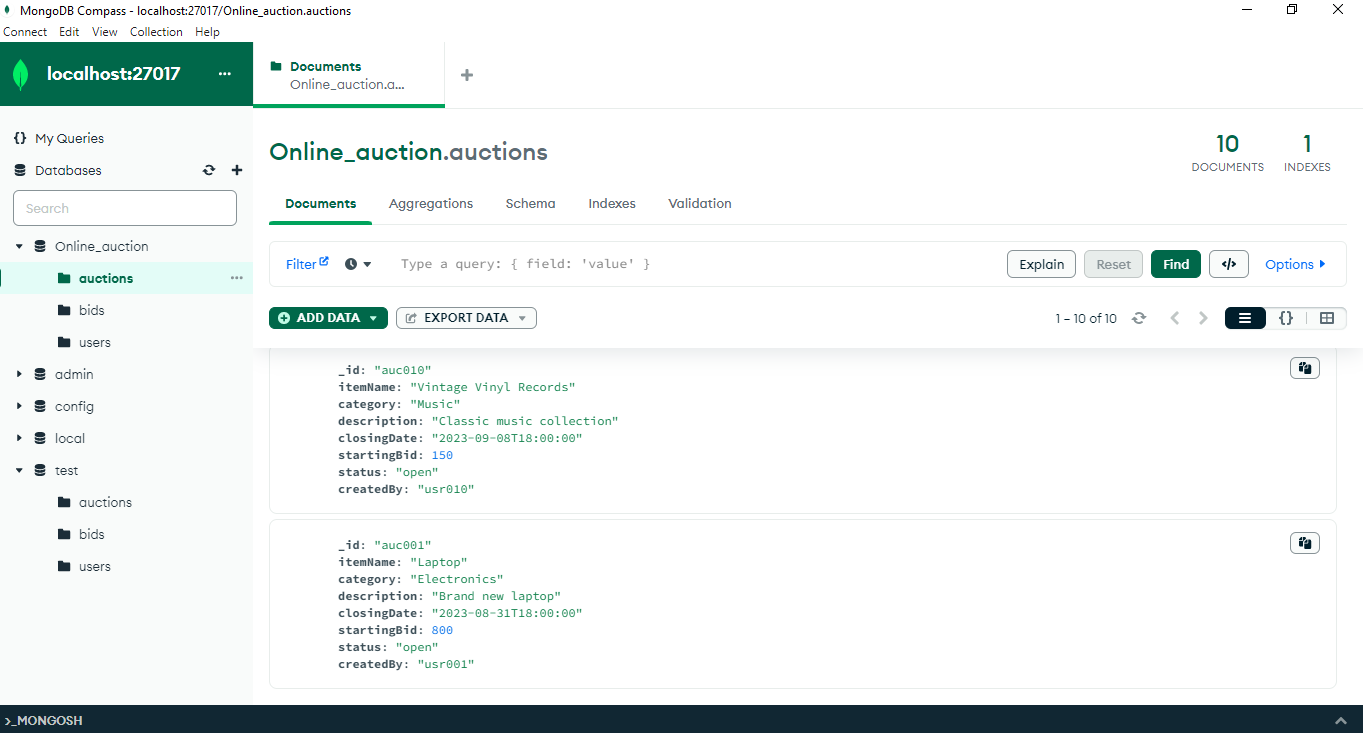
"closingDate": "2023-08-31T18:00:00",

"startingBid": 800,

"status": "open",

"createdBy": "usr001"

});



**Creating the Users Collection**

db.users.insertOne({

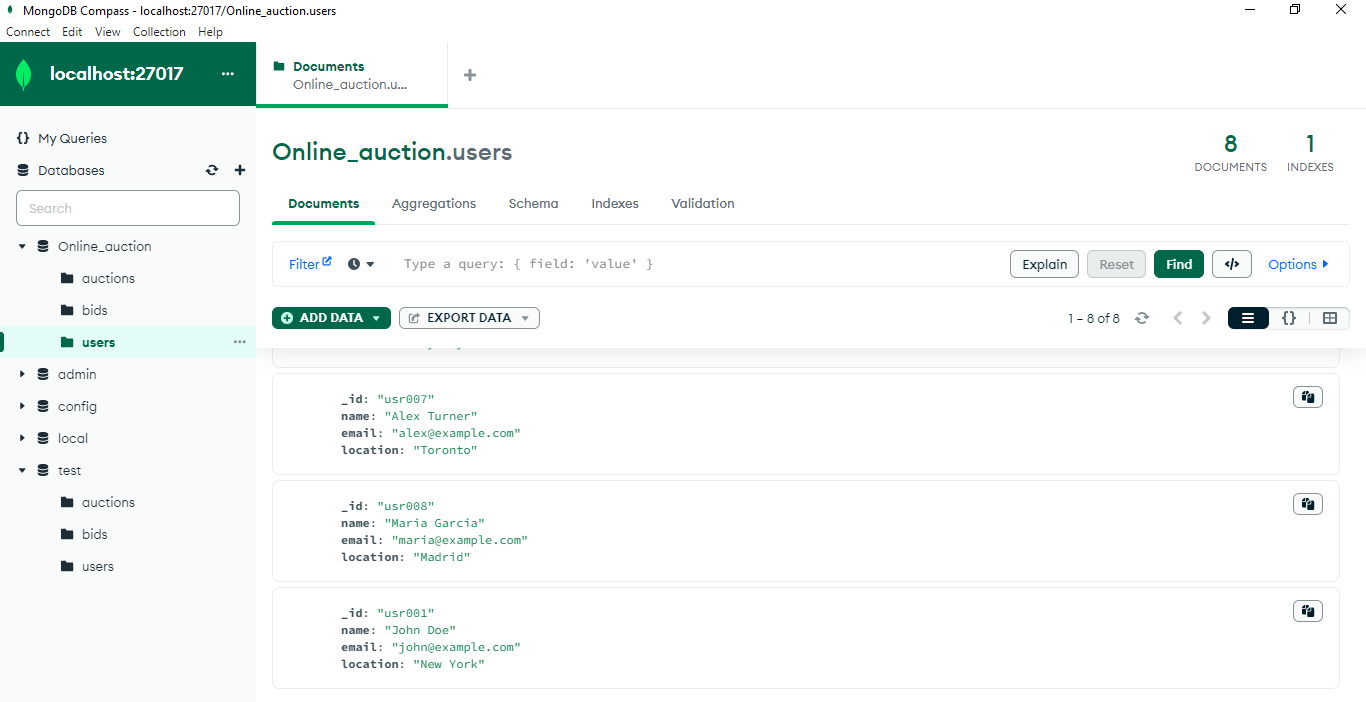
"\_id": "usr001",

"name": "John Doe",

"email": "john@example.com",

"location": "New York"

});



**Creating the Bids Collection**

db.bids.insertOne({

"\_id": "bid001",

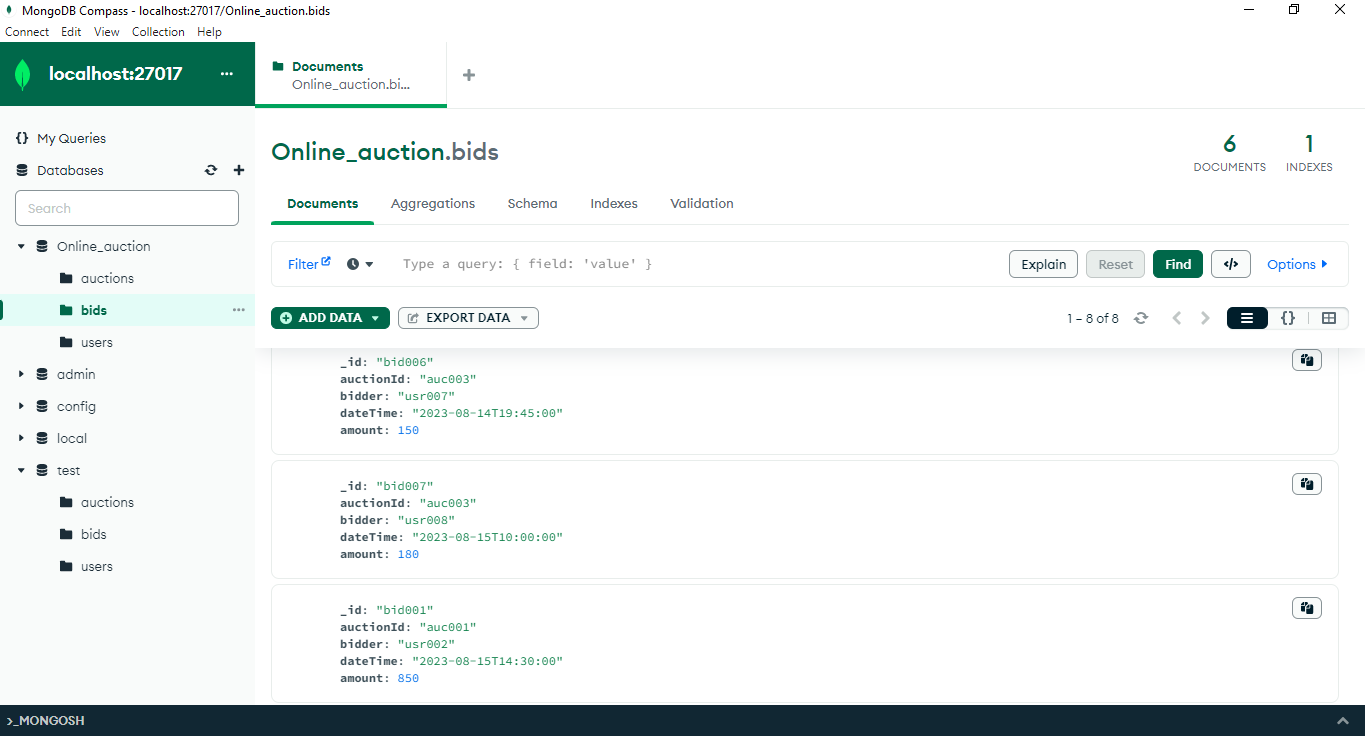
"auctionId": "auc001",

"bidder": "usr002",

"dateTime": "2023-08-15T14:30:00",

"amount": 850

});

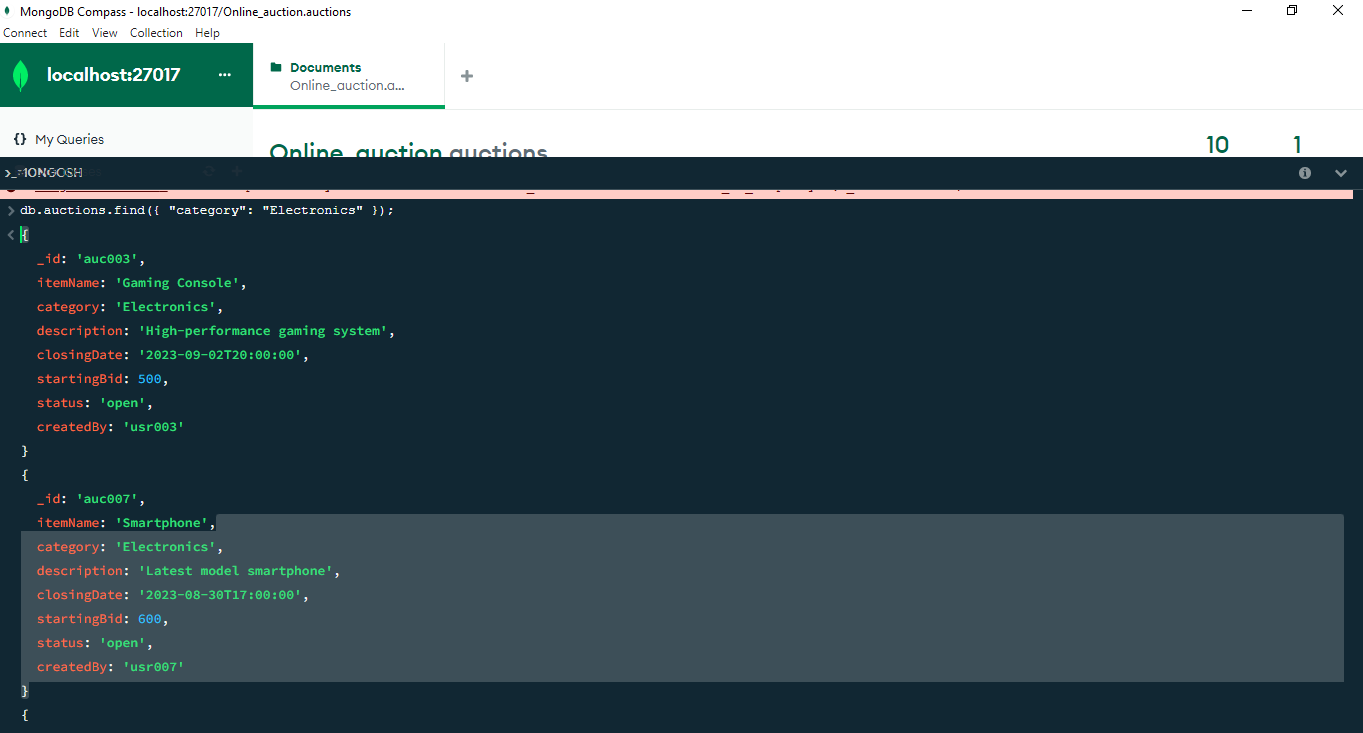


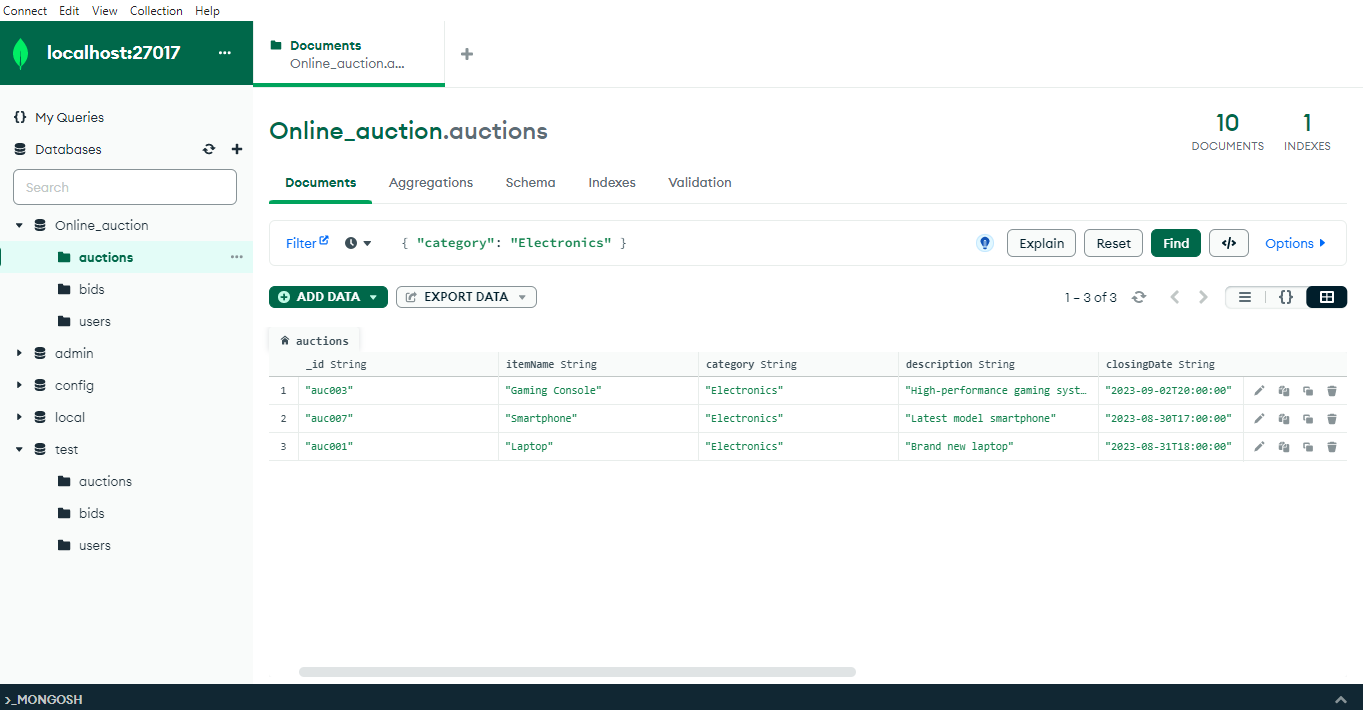
**A3. Queries**

For each key use case, the following MongoDB queries are developed and executed:

**1. Find all auctions for items in a specified category**

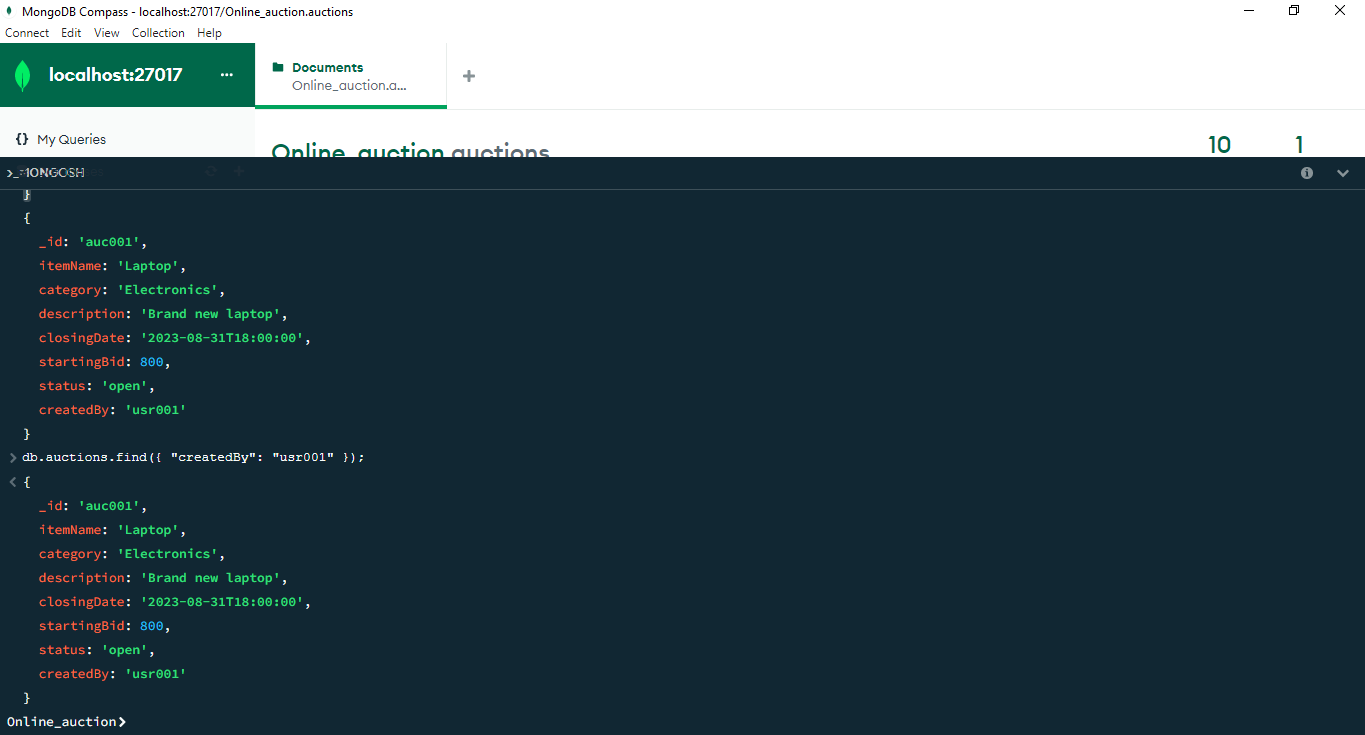
db.auctions.find({ "category": "Electronics" });

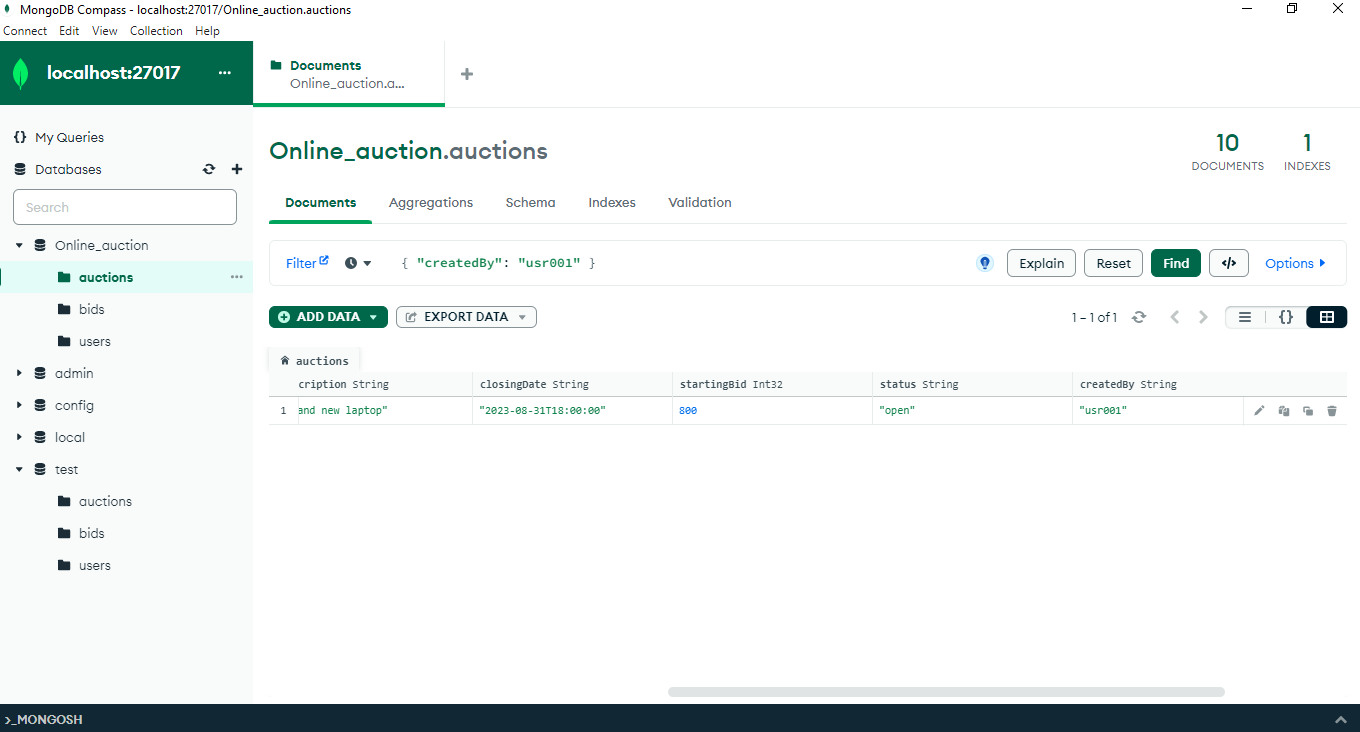




**2. Find all auctions created by a specified user**

db.auctions.find({ "createdBy": "usr001" });





**3. Make a new bid on an auction**

db.bids.insertOne({

"\_id": "bid011",

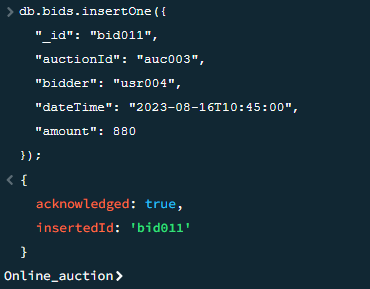
"auctionId": "auc003",

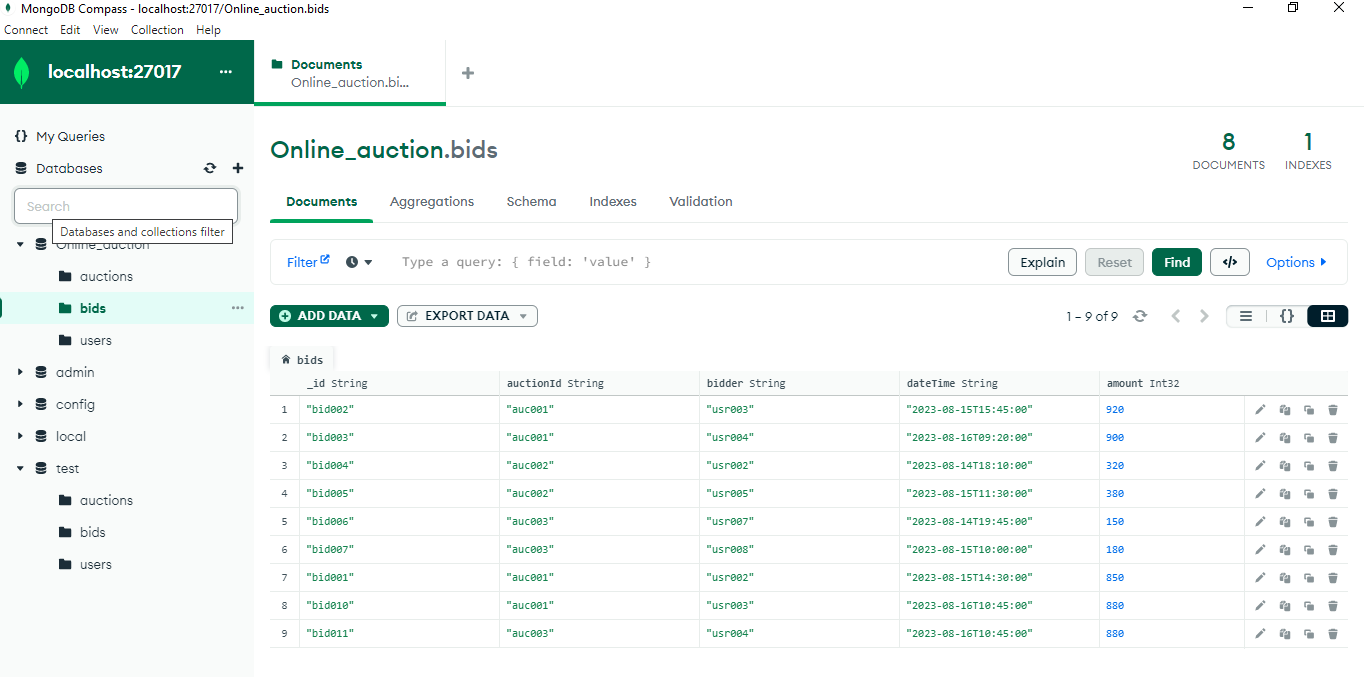
"bidder": "usr004",

"dateTime": "2023-08-16T10:45:00",

"amount": 880

});





**4. Show all the details of an auction including bids and status**

db.auctions.aggregate([

{ "$match": { "\_id": "auc001" } },

{

"$lookup": {

"from": "bids",

"localField": "\_id",

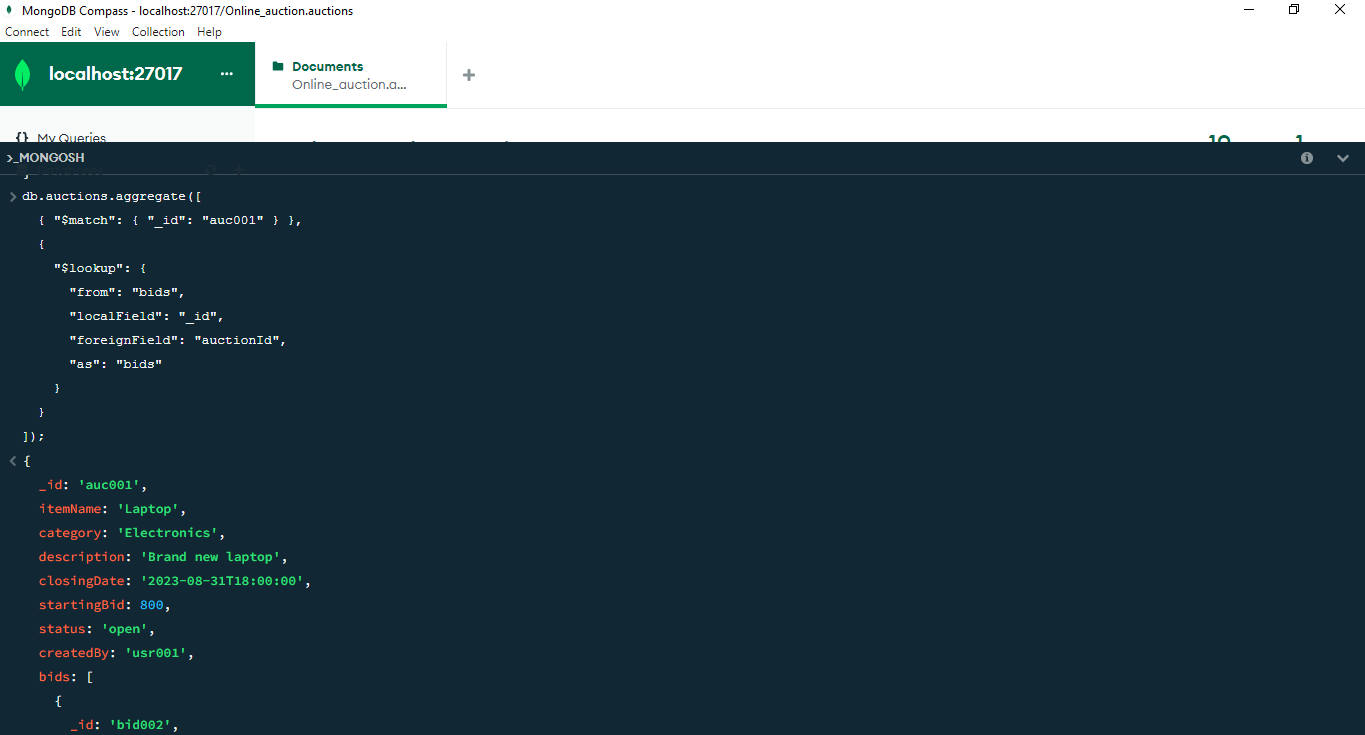
"foreignField": "auctionId",

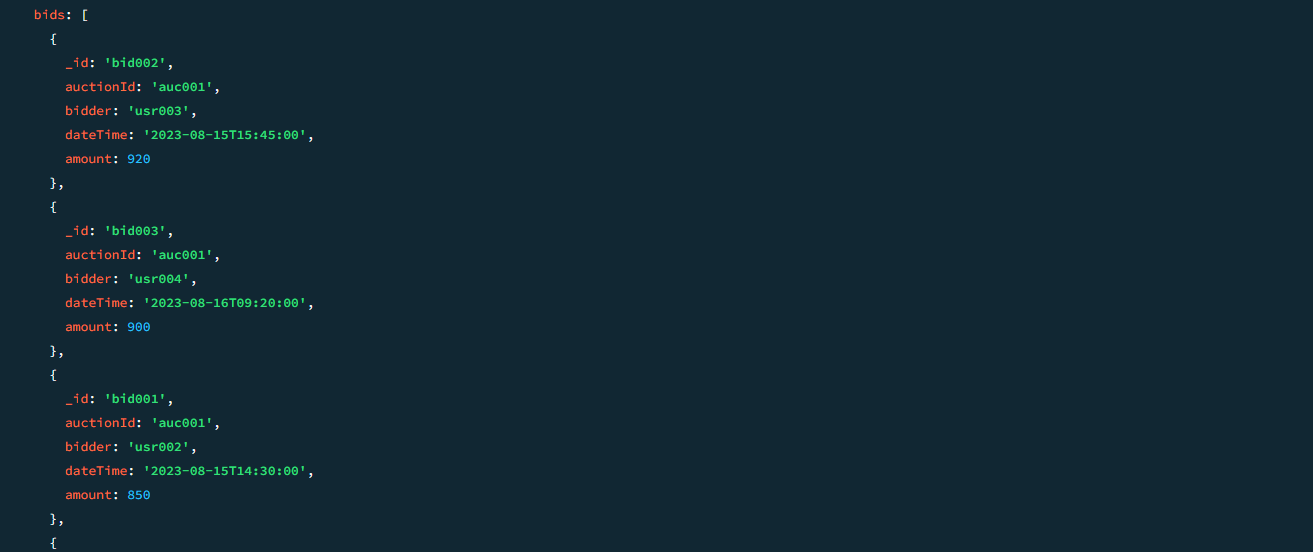
"as": "bids"

}

}

]);





**5. Calculate the average current price for each item category**

db.auctions.aggregate([

{

"$group": {

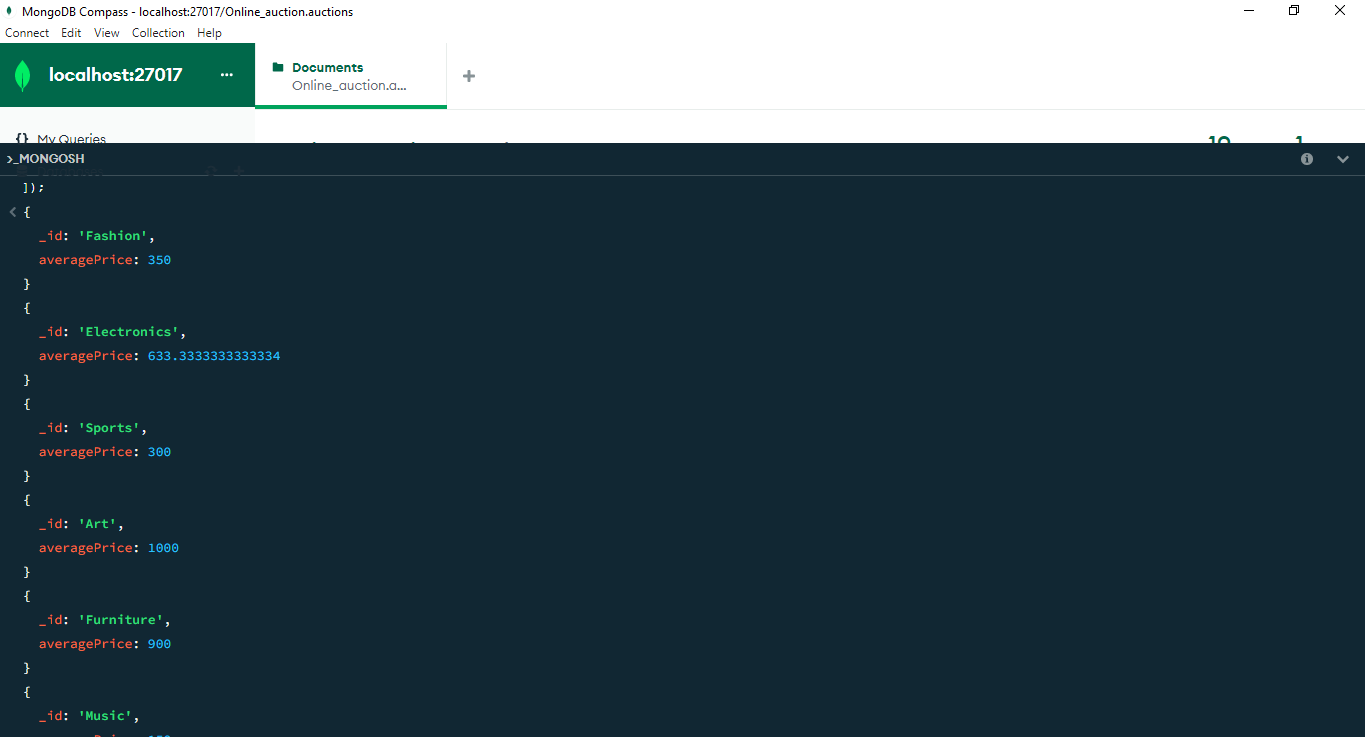
"\_id": "$category",

"averagePrice": { "$avg": "$startingBid" }

}

}

]);



This concludes Part A of the coursework report, focusing on the design and implementation of a MongoDB document database for an online auction application. The data model is optimized to address key use cases and enhance application performance.

**PART B: CREATING A GRAPH DATABASE REPORT**

In this section, we will outline the steps taken to create and populate a Neo4j database based on the provided Formula 1 data model. Additionally, we will provide the Cypher queries devised and executed to retrieve specific information from the database.

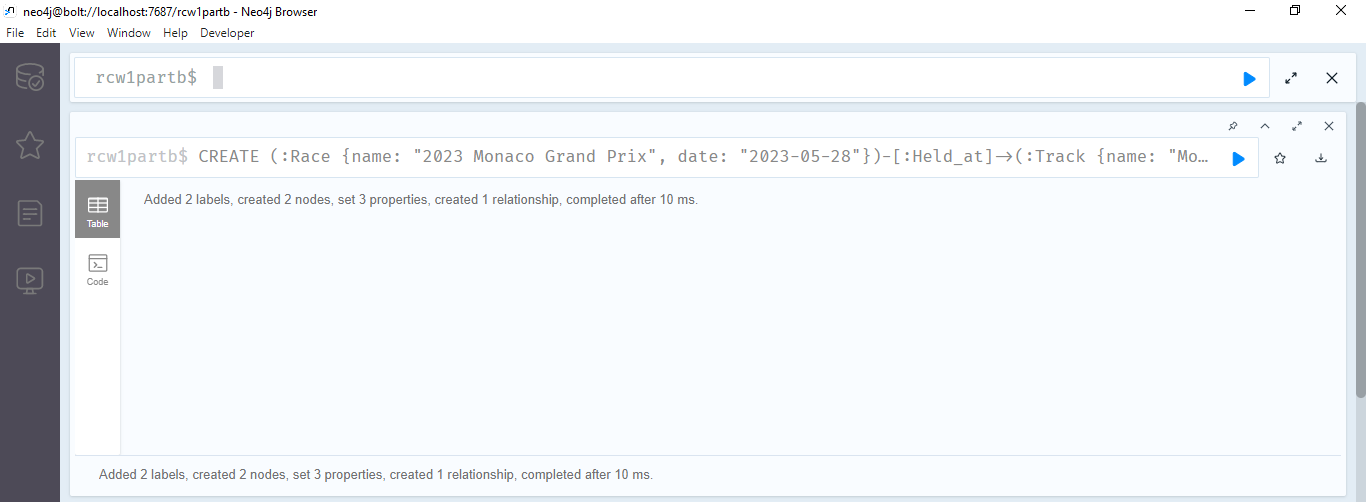
**B1. Creating the Database**

For this task, we utilized the Neo4j Sandbox to create a database named "Rcw1PartB." We followed the given data model to create nodes and edges representing the 2023 Monaco Grand Prix, 10 teams, 20 drivers, and their finishing positions and points.

The following Cypher statements were used to create nodes and edges:

1. Create the Monaco Grand Prix and its Track:

CREATE (:Race {name: "2023 Monaco Grand Prix", date: "2023-05-28"})-[:Held\_at]->(:Track {name: "Monte Carlo"})



1. Create Teams and Drivers:

// Create Teams

CREATE (:Team {name: "Red Bull"})

CREATE (:Team {name: "Aston Martin"})

CREATE (:Team {name: "Alpine"})

CREATE (:Team {name: "Mercedes"})

CREATE (:Team {name: "Ferrari"})

CREATE (:Team {name: "McLaren"})

CREATE (:Team {name: "Alfa Romeo"})

CREATE (:Team {name: "AlphaTauri"})

CREATE (:Team {name: "Williams"})

CREATE (:Team {name: "Haas"})

// Create Drivers and their relationship to Teams

CREATE (:Driver {name: "Max Verstappen", number: 1})-[:Drives\_for]->(:Team {name: "Red Bull"})

CREATE (:Driver {name: "Fernando Alonso", number: 14})-[:Drives\_for]->(:Team {name: "Aston Martin"})

CREATE (:Driver {name: "Esteban Ocon", number: 31})-[:Drives\_for]->(:Team {name: "Alpine"})

CREATE (:Driver {name: "Lewis Hamilton", number: 44})-[:Drives\_for]->(:Team {name: "Mercedes"})

CREATE (:Driver {name: "George Russell", number: 63})-[:Drives\_for]->(:Team {name: "Mercedes"})

CREATE (:Driver {name: "Charles Leclerc", number: 16})-[:Drives\_for]->(:Team {name: "Ferrari"})

CREATE (:Driver {name: "Pierre Gasly", number: 10})-[:Drives\_for]->(:Team {name: "Alpine"})

CREATE (:Driver {name: "Carlos Sainz Jnr", number: 55})-[:Drives\_for]->(:Team {name: "Ferrari"})

CREATE (:Driver {name: "Lando Norris", number: 4})-[:Drives\_for]->(:Team {name: "McLaren"})

CREATE (:Driver {name: "Oscar Piastri", number: 81})-[:Drives\_for]->(:Team {name: "McLaren"})

CREATE (:Driver {name: "Valtteri Bottas", number: 77})-[:Drives\_for]->(:Team {name: "Alfa Romeo"})

CREATE (:Driver {name: "Nyck de Vries", number: 21})-[:Drives\_for]->(:Team {name: "AlphaTauri"})

CREATE (:Driver {name: "Zhou Guanyu", number: 24})-[:Drives\_for]->(:Team {name: "Alfa Romeo"})

CREATE (:Driver {name: "Alexander Albon", number: 23})-[:Drives\_for]->(:Team {name: "Williams"})

CREATE (:Driver {name: "Yuki Tsunoda", number: 22})-[:Drives\_for]->(:Team {name: "AlphaTauri"})

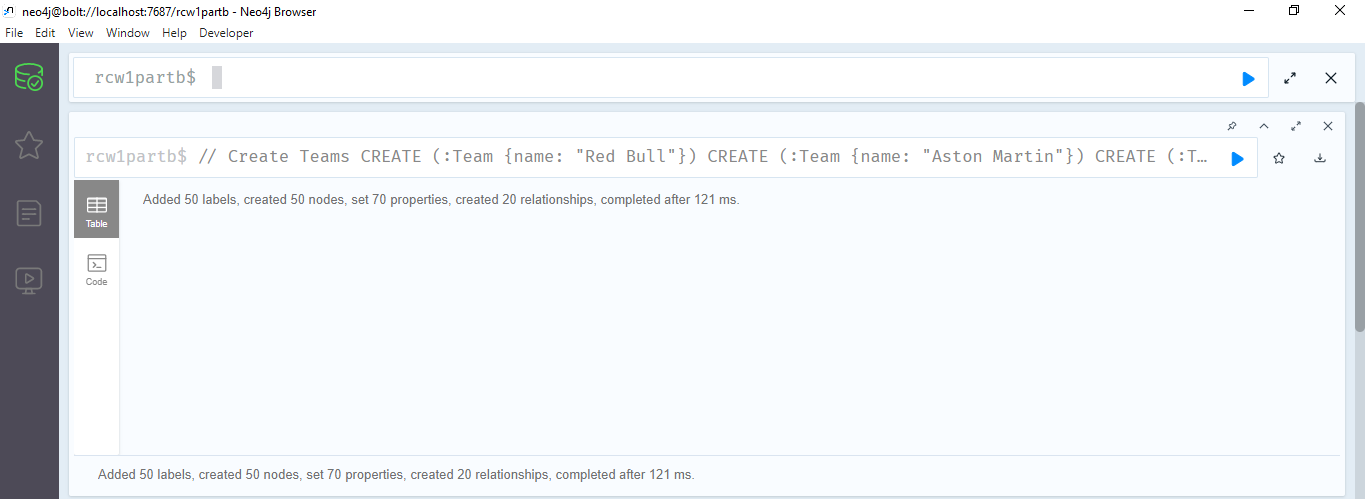
CREATE (:Driver {name: "Sergio Perez", number: 11})-[:Drives\_for]->(:Team {name: "Red Bull"})

CREATE (:Driver {name: "Nico Hulkenberg", number: 27})-[:Drives\_for]->(:Team {name: "Haas"})

CREATE (:Driver {name: "Logan Sargeant", number: 2})-[:Drives\_for]->(:Team {name: "Williams"})

CREATE (:Driver {name: "Kevin Magnussen", number: 20})-[:Drives\_for]->(:Team {name: "Haas"})

CREATE (:Driver {name: "Lance Stroll", number: 18})-[:Drives\_for]->(:Team {name: "Aston Martin"})



1. Populate Finishing Positions and Points:

// Create Drivers and their relationship to Race

// Create Drivers and their relationship to Race

MATCH (race:Race {name: "2023 Monaco Grand Prix"})

// Driver 1

CREATE (:Driver {name: "Max Verstappen", number: 11,position: 1, points: 25})-[:Finished {position: 1, points: 25}]->(race)

// Driver 2

CREATE (:Driver {name: "Fernando Alonso", number: 14,position: 2, points: 18})-[:Finished {position: 2, points: 18}]->(race)

// Driver 3

CREATE (:Driver {name: "Esteban Ocon", number: 31,position: 3, points: 15})-[:Finished {position: 3, points: 15}]->(race)

// Driver 4

CREATE (:Driver {name: "Lewis Hamilton", number: 44,position: 4, points: 13})-[:Finished {position: 4, points: 13}]->(race)

// Driver 5

CREATE (:Driver {name: "George Russell", number: 63,position: 5, points: 10})-[:Finished {position: 5, points: 10}]->(race)

// Driver 6

CREATE (:Driver {name: "Charles Leclerc", number: 16,position: 6, points: 8})-[:Finished {position: 6, points: 8}]->(race)

// Driver 7

CREATE (:Driver {name: "Pierre Gasly", number: 10,position: 7, points: 6})-[:Finished {position: 7, points: 6}]->(race)

// Driver 8

CREATE (:Driver {name: "Carlos Sainz Jnr", number: 55,position: 8, points: 4})-[:Finished {position: 8, points: 4}]->(race)

// Driver 9

CREATE (:Driver {name: "Lando Norris", number: 4,position: 9, points: 2})-[:Finished {position: 9, points: 2}]->(race)

// Driver 10

CREATE (:Driver {name: "Oscar Piastri", number: 81,position: 10, points: 1})-[:Finished {position: 10, points: 1}]->(race)

// Driver 11 to 19

CREATE (:Driver {name: "Valtteri Bottas", number: 77,position: 11, points: 0})-[:Finished {position: 11, points: 0}]->(race)

CREATE (:Driver {name: "Nyck de Vries", number: 21,position: 12, points: 0})-[:Finished {position: 12, points: 0}]->(race)

CREATE (:Driver {name: "Zhou Guanyu", number: 24,position: 13, points: 0})-[:Finished {position: 13, points: 0}]->(race)

CREATE (:Driver {name: "Alexander Albon", number: 23,position: 14, points: 0})-[:Finished {position: 14, points: 0}]->(race)

CREATE (:Driver {name: "Yuki Tsunoda", number: 22,position: 15, points: 0})-[:Finished {position: 15, points: 0}]->(race)

CREATE (:Driver {name: "Sergio Perez", number: 11,position: 16, points: 0})-[:Finished {position: 16, points: 0}]->(race)

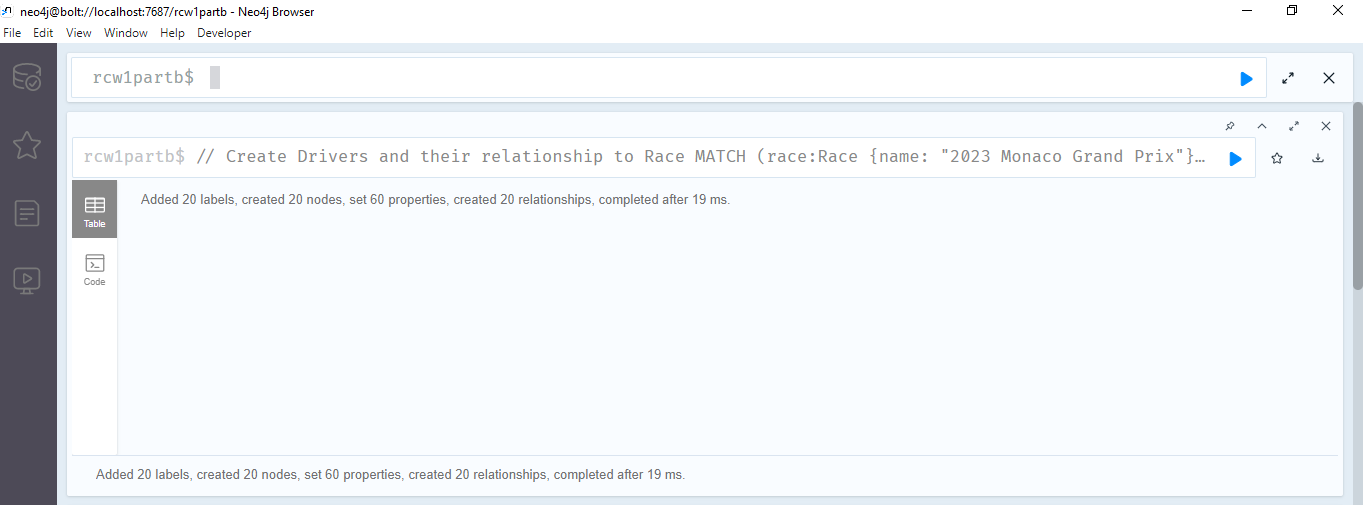
CREATE (:Driver {name: "Nico Hulkenberg", number: 27,position: 17, points: 0})-[:Finished {position: 17, points: 0}]->(race)

CREATE (:Driver {name: "Logan Sargeant", number: 2,position: 18, points: 0})-[:Finished {position: 18, points: 0}]->(race)

CREATE (:Driver {name: "Kevin Magnussen", number: 20,position: 19, points: 0})-[:Finished {position: 19, points: 0}]->(race)

// Driver 20 (non-classified drivers)

CREATE (:Driver {name: "Lance Stroll", number: 18,position: 99, points: 0})-[:Unfinished {position: 99, points: 0}]->(race)



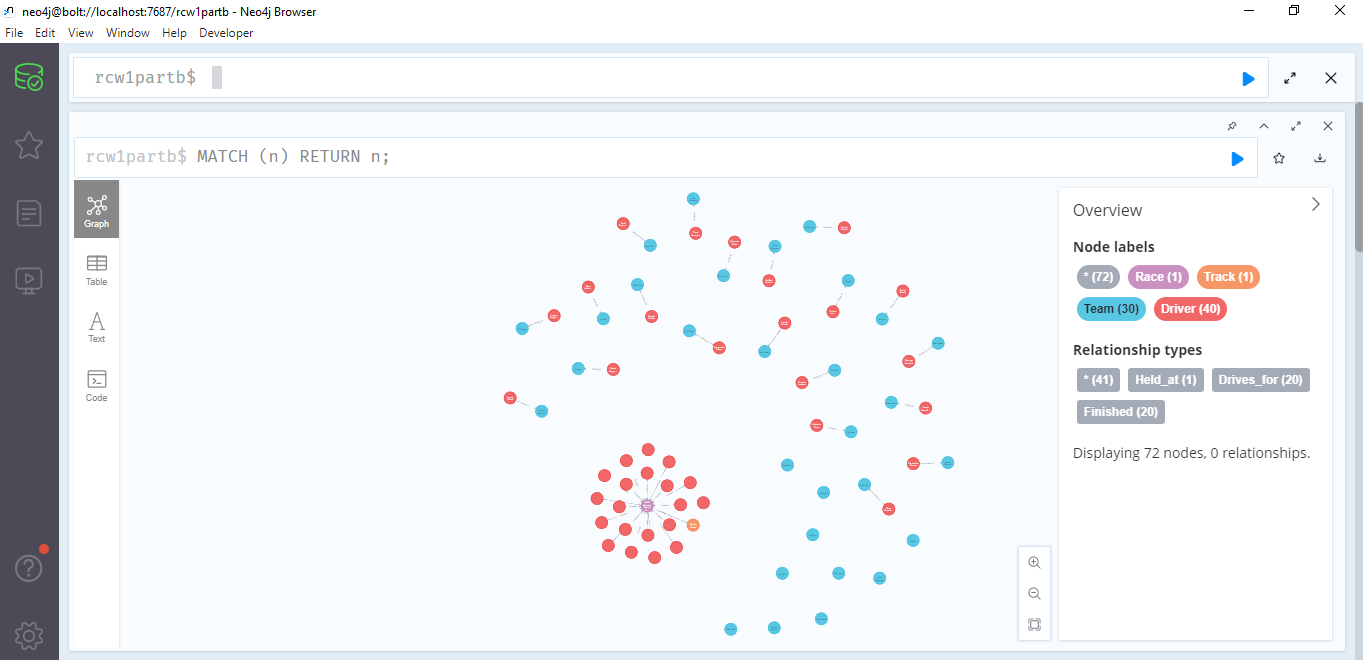
**B2. Queries**

We developed and executed Cypher queries to retrieve specific data from the database, as follows:

**Query 1: Retrieve all nodes and edges in the database**

MATCH (n)

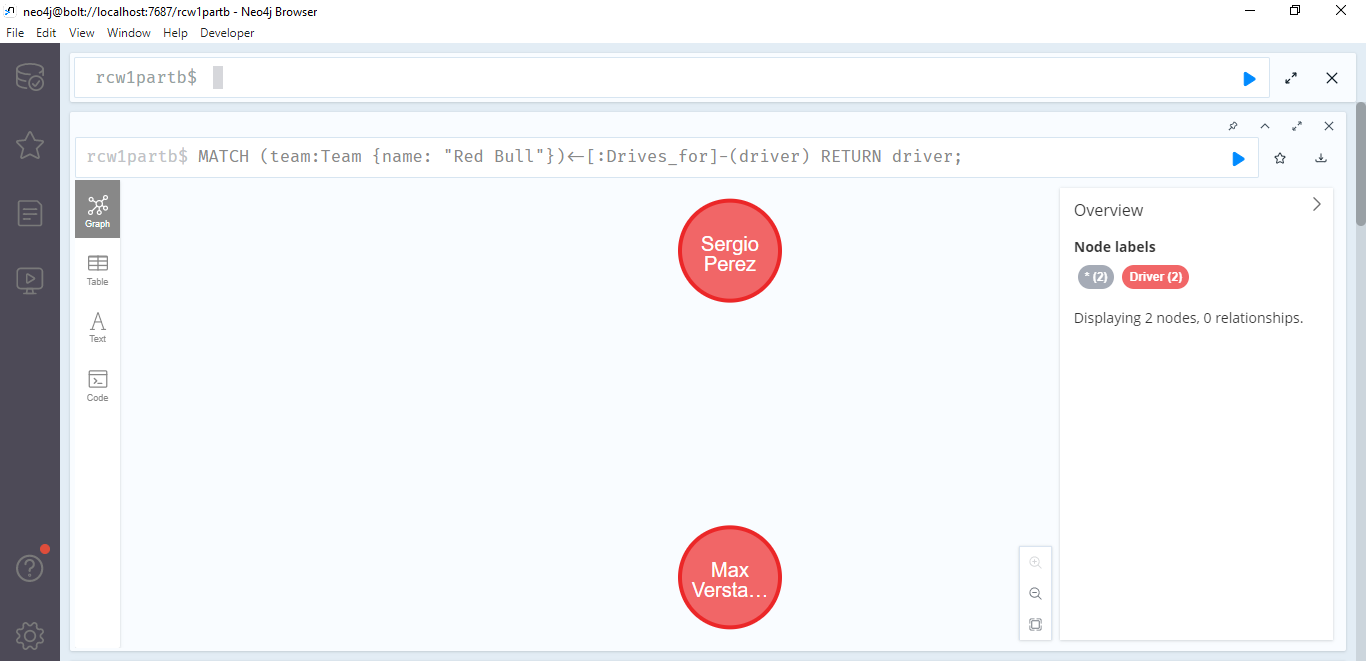
RETURN n

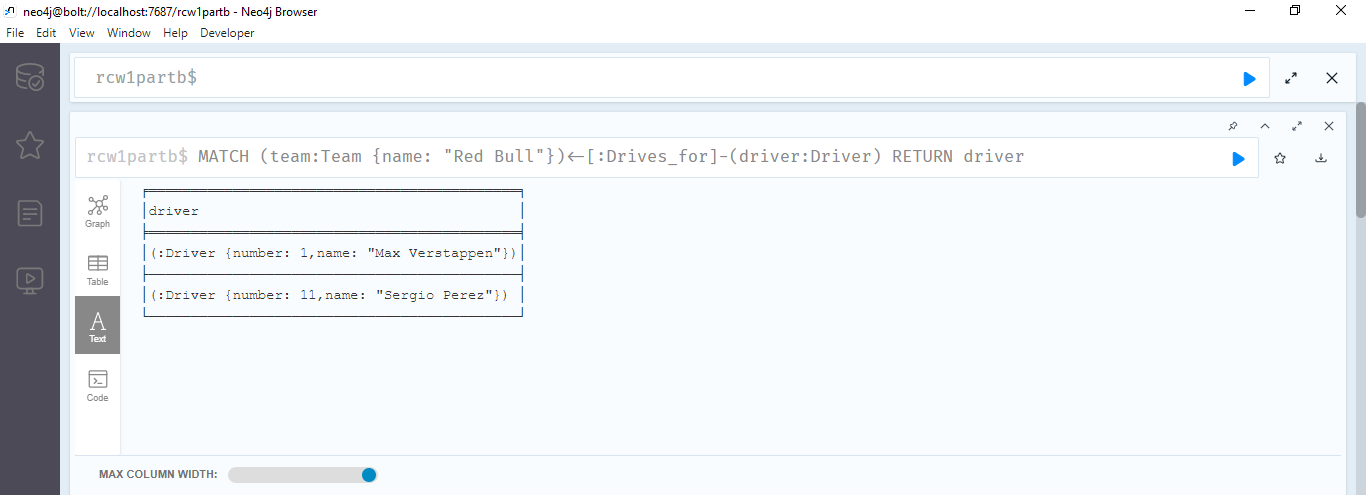


**Query 2: Retrieve drivers who drive for a specified team (e.g., "Red Bull")**

MATCH (team:Team {name: "Red Bull"})<-[:Drives\_for]-(driver:Driver)

RETURN driver





**Query 3: Retrieve the top ten finishers (drivers) in order**

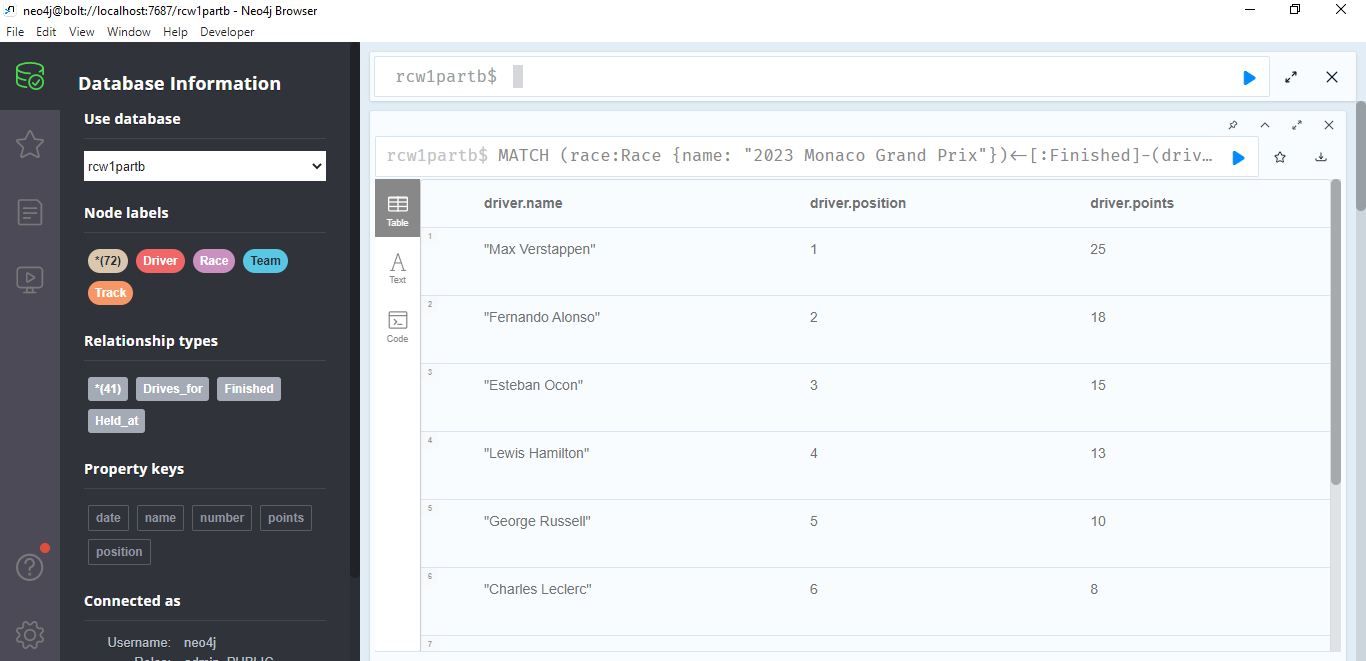
MATCH (race:Race {name: "2023 Monaco Grand Prix"})<-[finished:Finished]-(driver:Driver)

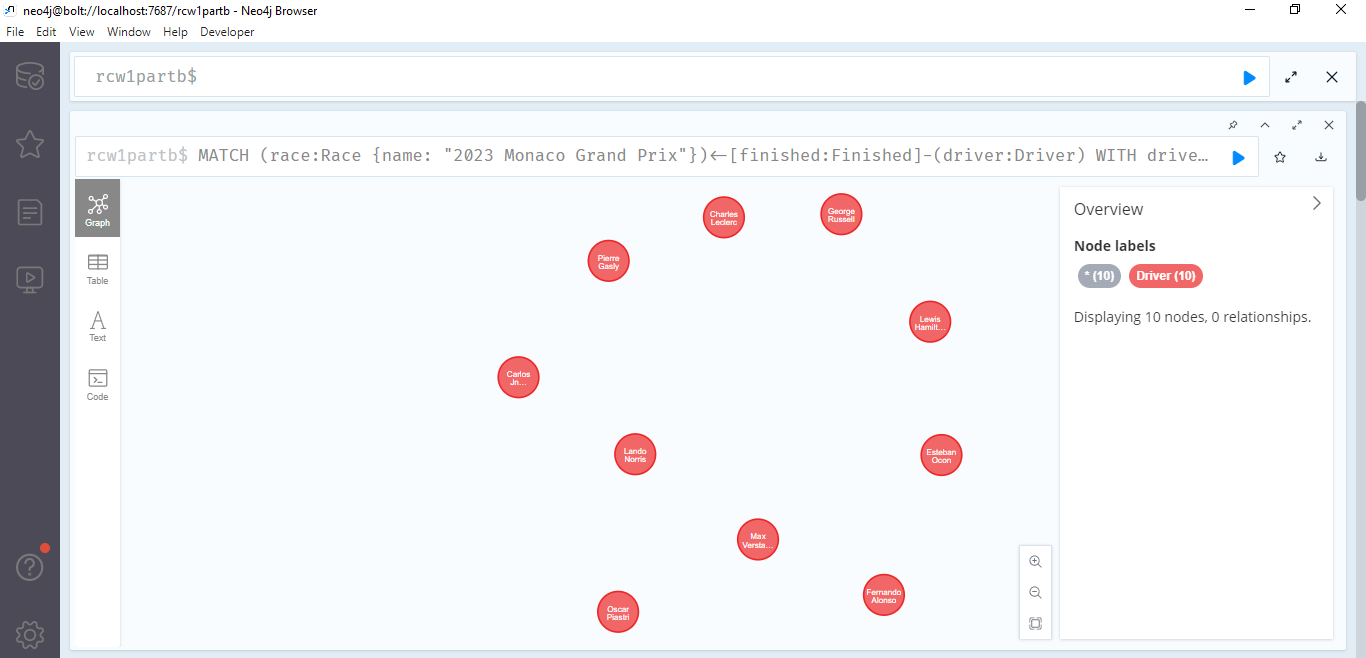
WITH driver, finished

ORDER BY finished.position

LIMIT 10

RETURN driver





**Query 4: Retrieve the team-mate of a specified driver (e.g., "Max Verstappen")**

MATCH (driver:Driver {name: "Max Verstappen"})-[:Drives\_for]->(team:Team)<-[:Drives\_for]-(teammate:Driver)

WHERE teammate <> driver

RETURN teammate

**Query 5: Retrieve the number of points scored by a specified team (e.g., "Red Bull")**

MATCH (team:Team {name: "Red Bull"})<-[:Drives\_for]-(driver:Driver)-[:Finished]->(race:Race {name: "2023 Monaco Grand Prix"})

RETURN SUM(CASE WHEN driver.position <= 10 THEN driver.points ELSE 0 END) AS teamPoints

**Conclusion:** In this report, we have outlined the process of creating and populating a Neo4j database based on the provided Formula 1 data model. We have also presented the Cypher queries devised and executed to retrieve specific information from the database. The database and queries accurately represent the Formula 1 data model, providing insights into races, teams, drivers, and race results.