

VIT-AP UNIVERSITY, ANDHRA PRADESH

Lab Sheet 6: MongoDB Basic commands

Branch/ Class: B.Tech/M.Tech

Date: 26-02-2026

Faculty Name: Prof. S.Gopikrishnan

School: SCOPE

Student name: Varun Kumar Tenali

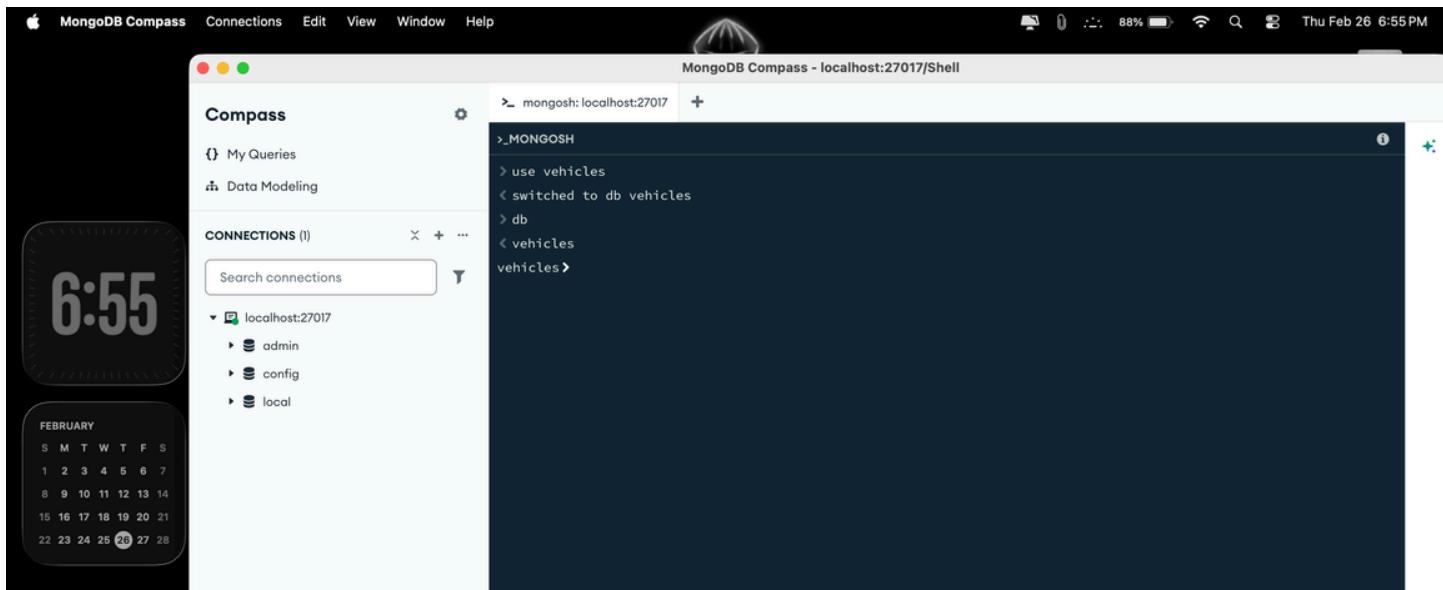
Reg. no.: 23BCE9454



VIT-AP
UNIVERSITY

1. Use MongoDB to implement the following DB operations :

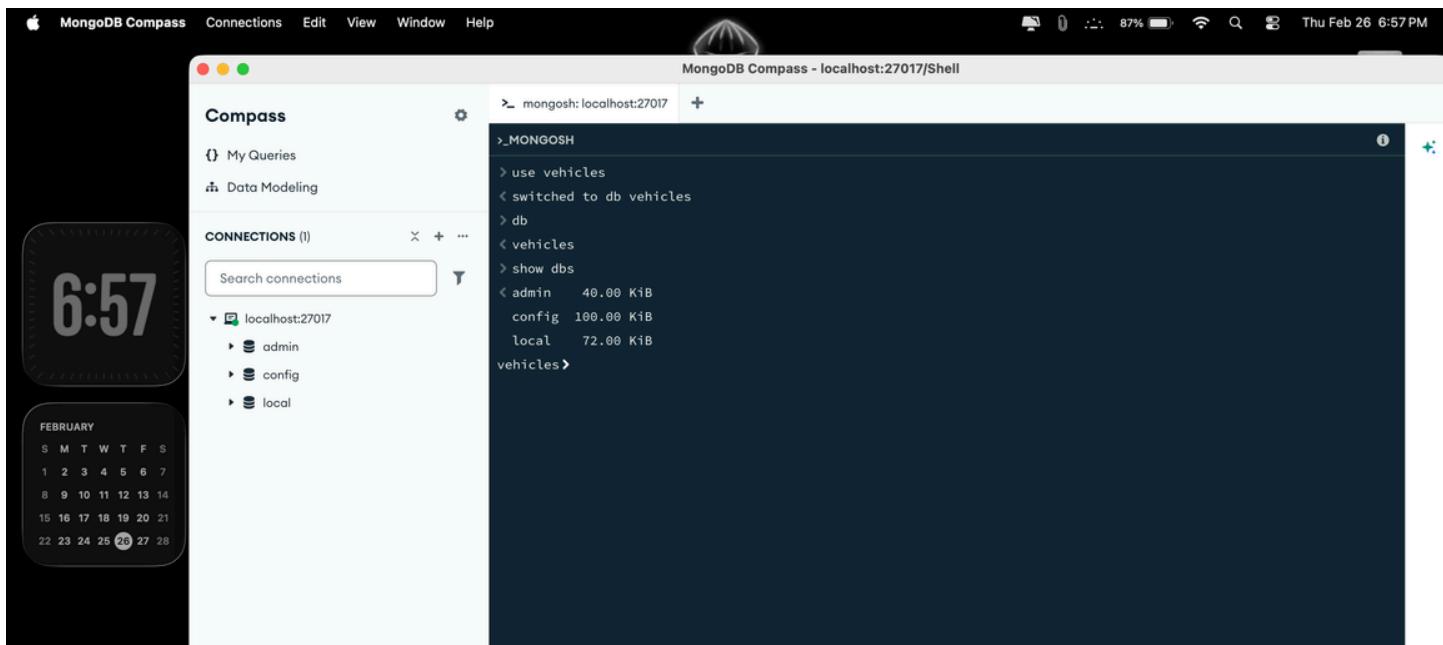
1) Create a database called 'vehicles' and write a MongoDB query to select database as "vehicles".



The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with a calendar for February and a 'Connections' section. The main area is titled 'mongosh: localhost:27017/Shell'. It displays the following MongoDB session:

```
>_MONGOSH
> use vehicles
< switched to db vehicles
> db
< vehicles
vehicles>
```

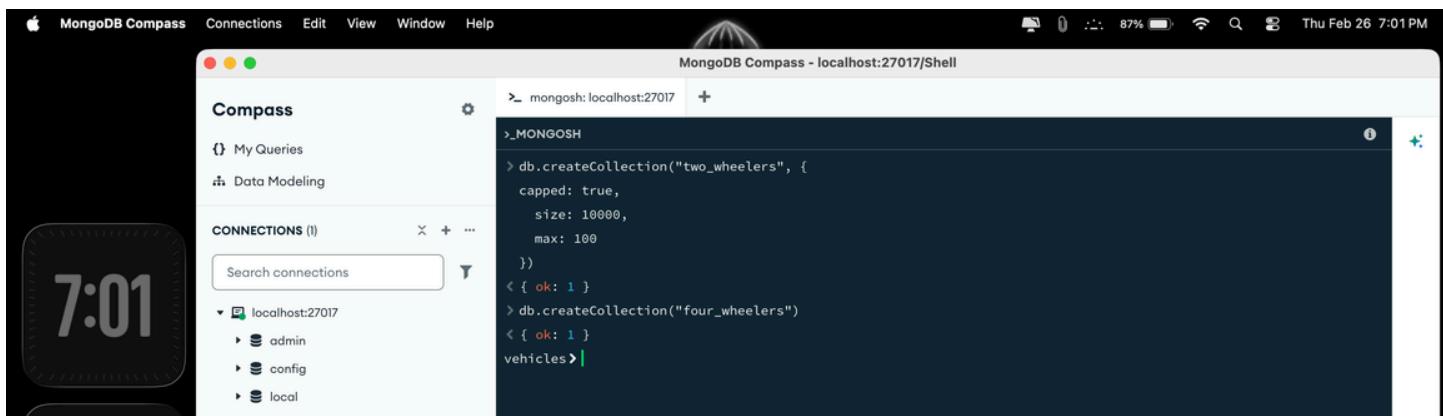
2) Write a MongoDB query to display all the databases.



The screenshot shows the MongoDB Compass interface. The left sidebar includes a calendar for February. The main window shows the same MongoDB session as the previous screenshot, but after the 'use vehicles' command, it includes a 'show dbs' command:

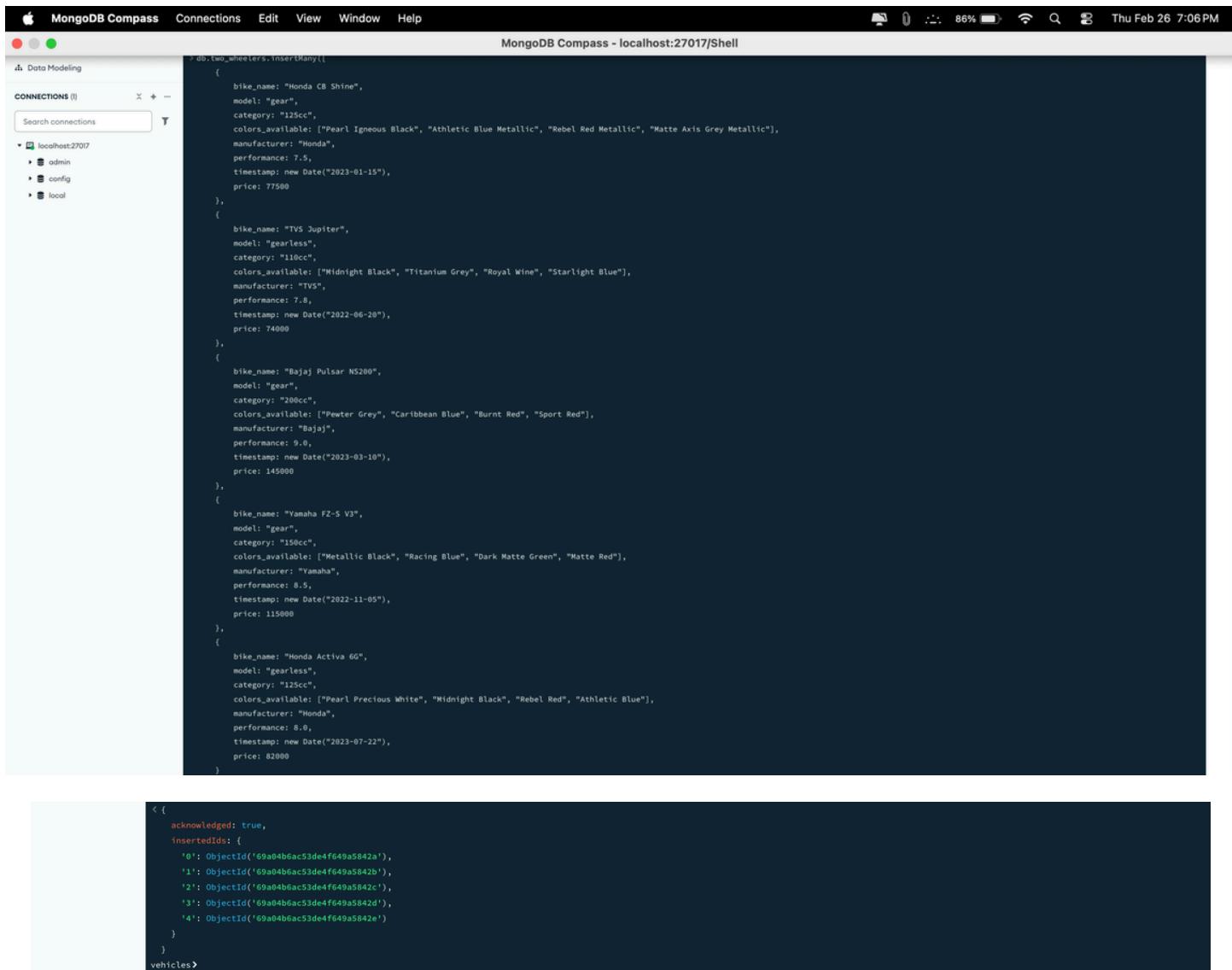
```
>_MONGOSH
> use vehicles
< switched to db vehicles
> db
< vehicles
> show dbs
< admin   40.00 KiB
 config  100.00 KiB
 local    72.00 KiB
 vehicles>
```

3) Create a collection called 'two_wheelers'. (use capping) and Create a collection called 'four_wheelers'.



```
mongosh: localhost:27017/Shell
>_MONGOSH
> db.createCollection("two_wheelers", {
  capped: true,
  size: 10000,
  max: 100
})
< { ok: 1 }
> db.createCollection("four_wheelers")
< { ok: 1 }
vehicles>
```

4) Add 5 two-wheeler details to the collection named 'two_wheelers'. Each document consists of following fields as bike_name, model (gear or gearless), category (100cc, 125cc, 150cc, 200cc), colors_available (red, black, blue, sport red etc) as array, manufacturer, performance (out of 10), timestamp (date and year release) and price.



```
mongosh: localhost:27017/Shell
> db.two_wheelers.insertMany([
  {
    bike_name: "Honda CB Shine",
    model: "gear",
    category: "125cc",
    colors_available: ["Pearl Igneous Black", "Athletic Blue Metallic", "Rebel Red Metallic", "Matte Axis Grey Metallic"],
    manufacturer: "Honda",
    performance: 7.5,
    timestamp: new Date("2023-01-15"),
    price: 77500
  },
  {
    bike_name: "TVS Jupiter",
    model: "gearless",
    category: "110cc",
    colors_available: ["Midnight Black", "Titanium Grey", "Royal Wine", "Starlight Blue"],
    manufacturer: "TVS",
    performance: 7.8,
    timestamp: new Date("2022-06-20"),
    price: 74000
  },
  {
    bike_name: "Bajaj Pulsar NS200",
    model: "gear",
    category: "200cc",
    colors_available: ["Pewter Grey", "Caribbean Blue", "Burnt Red", "Sport Red"],
    manufacturer: "Bajaj",
    performance: 9.0,
    timestamp: new Date("2023-03-10"),
    price: 145000
  },
  {
    bike_name: "Yamaha FZ-S V3",
    model: "gear",
    category: "150cc",
    colors_available: ["Metallic Black", "Racing Blue", "Dark Matte Green", "Matte Red"],
    manufacturer: "Yamaha",
    performance: 8.5,
    timestamp: new Date("2022-11-05"),
    price: 115000
  },
  {
    bike_name: "Honda Activa 6G",
    model: "gearless",
    category: "125cc",
    colors_available: ["Pearl Precious White", "Midnight Black", "Rebel Red", "Athletic Blue"],
    manufacturer: "Honda",
    performance: 8.0,
    timestamp: new Date("2023-07-22"),
    price: 82000
  }
], {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('69a04b6ac53de4f649a5842a'),
    '1': ObjectId('69a04b6ac53de4f649a5842b'),
    '2': ObjectId('69a04b6ac53de4f649a5842c'),
    '3': ObjectId('69a04b6ac53de4f649a5842d'),
    '4': ObjectId('69a04b6ac53de4f649a5842e')
  }
})
vehicles>
```

5) Add 5 four-wheeler details to the collection named 'four_wHEELERS'. Each document consists of following fields as vehicle_name, model (commercial or own), category (car, lorry, bus, mini truck, heavy truck, containers), variants (vxi, zxi, petrol, diesel etc) as array, manufacturer, performance (out of 10), timestamp (date and year release) and price.

The screenshot shows the MongoDB Compass interface with the following details:

- Top Bar:** MongoDB Compass, Connections, Edit, View, Window, Help, Date: Thu Feb 26 7:08 PM, Battery: 86%.
- Left Sidebar:** Data Modeling, CONNECTIONS (1), Search connections, localhost:27017 (selected), admin, config, local.
- Middle Panel:** Shell tab, command: db.four_wHEELERS.insertMany([...]). The code block contains five documents with the following fields:
 - vehicle_name: "Rolls-Royce Silver Shadow", model: "own", category: "car", variants: ["Standard Saloon", "Long Wheelbase", "Two-Door Saloon", "Drophead Coupe"], manufacturer: "Rolls-Royce", performance: 9.5, timestamp: new Date("1965-10-01"), price: 35000000
 - vehicle_name: "Bentley Continental", model: "own", category: "car", variants: ["GT V8", "GT W12", "GT Speed", "GTC Convertible V8", "GTC Convertible W12"], manufacturer: "Bentley", performance: 9.7, timestamp: new Date("1952-01-01"), price: 42000000
 - vehicle_name: "Aston Martin DB5", model: "own", category: "car", variants: ["Standard Coupe", "Vantage Coupe", "Convertible", "Shooting Brake"], manufacturer: "Aston Martin", performance: 9.4, timestamp: new Date("1963-07-01"), price: 55000000
 - vehicle_name: "Porsche 911 Air-Cooled", model: "own", category: "car", variants: ["911 Carrera", "911 Targa", "911 Turbo", "911 RS", "911 Cabriolet"], manufacturer: "Porsche", performance: 9.6, timestamp: new Date("1963-09-12"), price: 38000000
 - vehicle_name: "Mercedes-Benz 300SL Gullwing", model: "own", category: "car", variants: ["Coupe Gullwing", "Roadster", "Racing Specification"], manufacturer: "Mercedes-Benz", performance: 9.8, timestamp: new Date("1954-02-06"), price: 95000000
- Bottom Panel:** Results pane showing the response from the insertMany operation, including acknowledged: true, insertedIds: { '0': ObjectId('69a04c89c53de4f649a5842f'), '1': ObjectId('69a04c89c53de4f649a58430'), '2': ObjectId('69a04c89c53de4f649a58431'), '3': ObjectId('69a04c89c53de4f649a58432'), '4': ObjectId('69a04c89c53de4f649a58433') }.

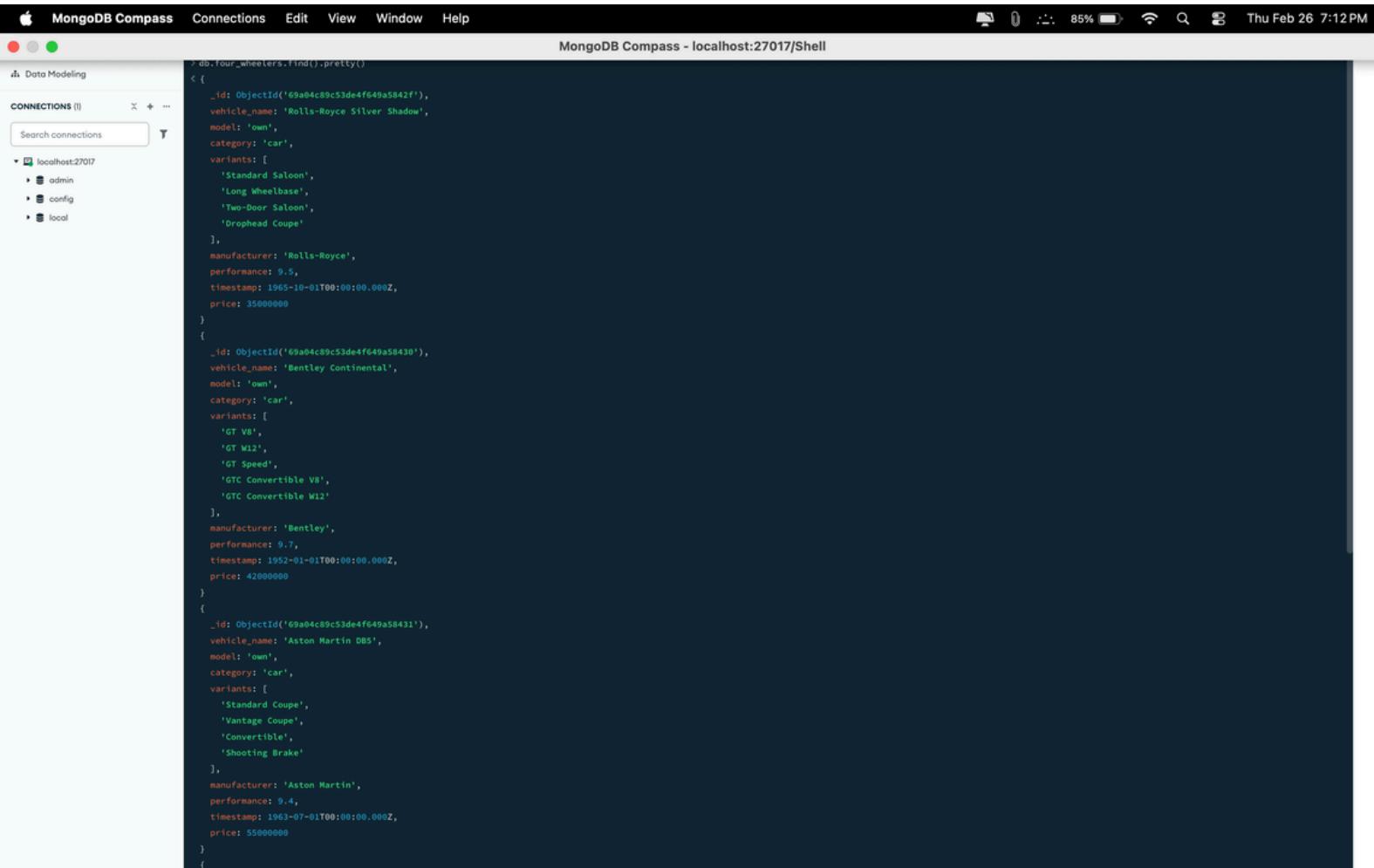
6) Write a MongoDB query to display all documents available in two_wheelers and four_wheelers.

→ Documents from two.wheelers :

The screenshot shows the Compass MongoDB interface. On the left, there's a sidebar with 'My Queries', 'Data Modeling', and a 'CONNECTIONS (1)' section containing a dropdown for 'localhost:27017' with options 'admin', 'config', and 'local'. The main area is titled '>_ mongosh: localhost:27017' and shows the command '> db.two_wHEELERS.find().pretty()' followed by its output. The output displays three documents from the 'two_wHEELERS' collection, each representing a motorcycle (bike) with fields like _id, bike_name, model, category, colors_available, manufacturer, performance, timestamp, and price.

```
>_ mongosh: localhost:27017
> db.two_wHEELERS.find().pretty()
< [
  {
    "_id": ObjectId('69a04b6ac53de4f649a5842a'),
    "bike_name": "Honda CB Shine",
    "model": "gear",
    "category": "125cc",
    "colors_available": [
      "Pearl Igneous Black",
      "Athletic Blue Metallic",
      "Rebel Red Metallic",
      "Matte Axis Grey Metallic"
    ],
    "manufacturer": "Honda",
    "performance": 7.5,
    "timestamp": 2023-01-15T00:00:00.000Z,
    "price": 77500
  },
  {
    "_id": ObjectId('69a04b6ac53de4f649a5842b'),
    "bike_name": "TVS Jupiter",
    "model": "gearless",
    "category": "110cc",
    "colors_available": [
      "Midnight Black",
      "Titanium Grey",
      "Royal Wine",
      "Starlight Blue"
    ],
    "manufacturer": "TVS",
    "performance": 7.8,
    "timestamp": 2022-06-20T00:00:00.000Z,
    "price": 74000
  },
  {
    "_id": ObjectId('69a04b6ac53de4f649a5842c'),
    "bike_name": "Bajaj Pulsar NS200",
    "model": "gear",
    "category": "200cc",
    "colors_available": [
      "Pewter Grey",
      "Caribbean Blue",
      "Burnt Red",
      "Sport Red"
    ],
    "manufacturer": "Bajaj",
    "performance": 9,
    "timestamp": 2022-06-20T00:00:00.000Z,
    "price": 105000
  }
]
```

→ Documents from four.wheelers :



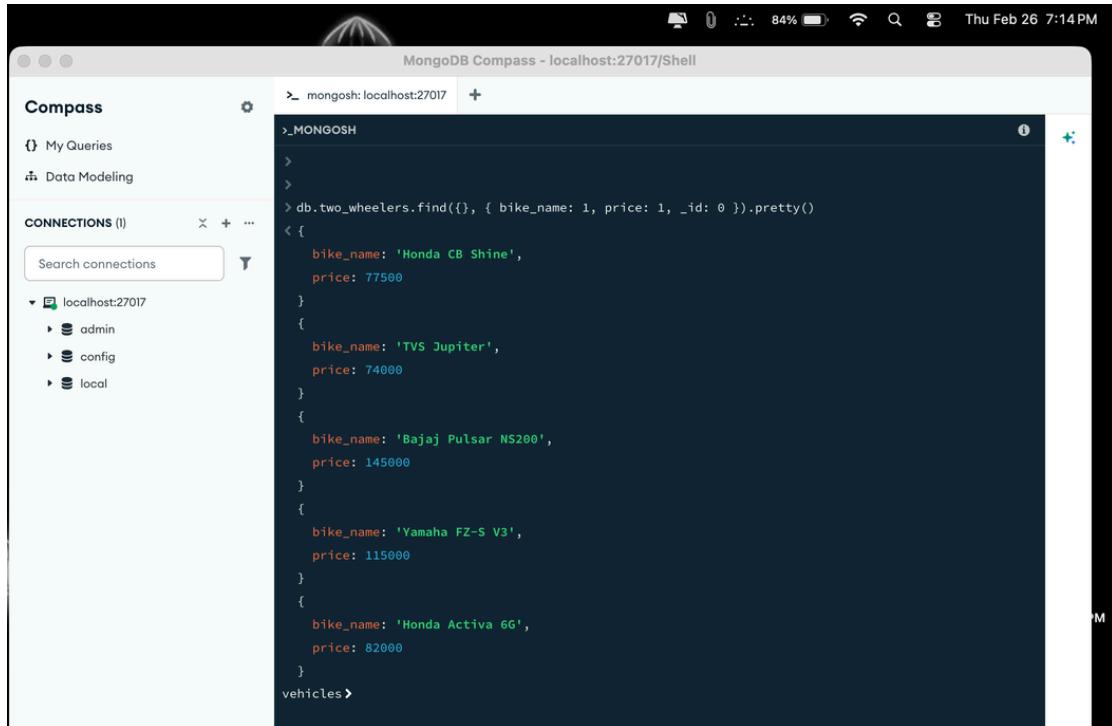
The screenshot shows the MongoDB Compass interface with the following details:

- Header:** MongoDB Compass, Connections, Edit, View, Window, Help, Thu Feb 26 7:12 PM
- Left Sidebar:** Data Modeling, CONNECTIONS (localhost:27017), Search connections, admin, config, local.
- Central Area:** MongoDB Compass - localhost:27017/Shell
- Query Result:** A list of documents from the 'four_wheelers' collection. The first few documents are:

```
db.four_wheelers.find().pretty()
[ {
  "_id": ObjectId('69a04c89c53de4f649a5842f'),
  "vehicle_name": "Rolls-Royce Silver Shadow",
  "model": "own",
  "category": "car",
  "variants": [
    "Standard Saloon",
    "Long Wheelbase",
    "Two-Door Saloon",
    "Drophead Coupe"
  ],
  "manufacturer": "Rolls-Royce",
  "performance": 9.5,
  "timestamp": 1965-10-01T00:00:00.000Z,
  "price": 35000000
},
{
  "_id": ObjectId('69a04c89c53de4f649a58430'),
  "vehicle_name": "Bentley Continental",
  "model": "own",
  "category": "car",
  "variants": [
    "GT V8",
    "GT W12",
    "GT Speed",
    "GTC Convertible V8",
    "GTC Convertible W12"
  ],
  "manufacturer": "Bentley",
  "performance": 9.7,
  "timestamp": 1952-01-01T00:00:00.000Z,
  "price": 42000000
},
{
  "_id": ObjectId('69a04c89c53de4f649a58431'),
  "vehicle_name": "Aston Martin DB5",
  "model": "own",
  "category": "car",
  "variants": [
    "Standard Coupe",
    "Vantage Coupe",
    "Convertible",
    "Shooting Brake"
  ],
  "manufacturer": "Aston Martin",
  "performance": 9.4,
  "timestamp": 1963-07-01T00:00:00.000Z,
  "price": 55000000
} ]
```

7) Write a MongoDB query to display only vehicle name and price in all the collection of the database

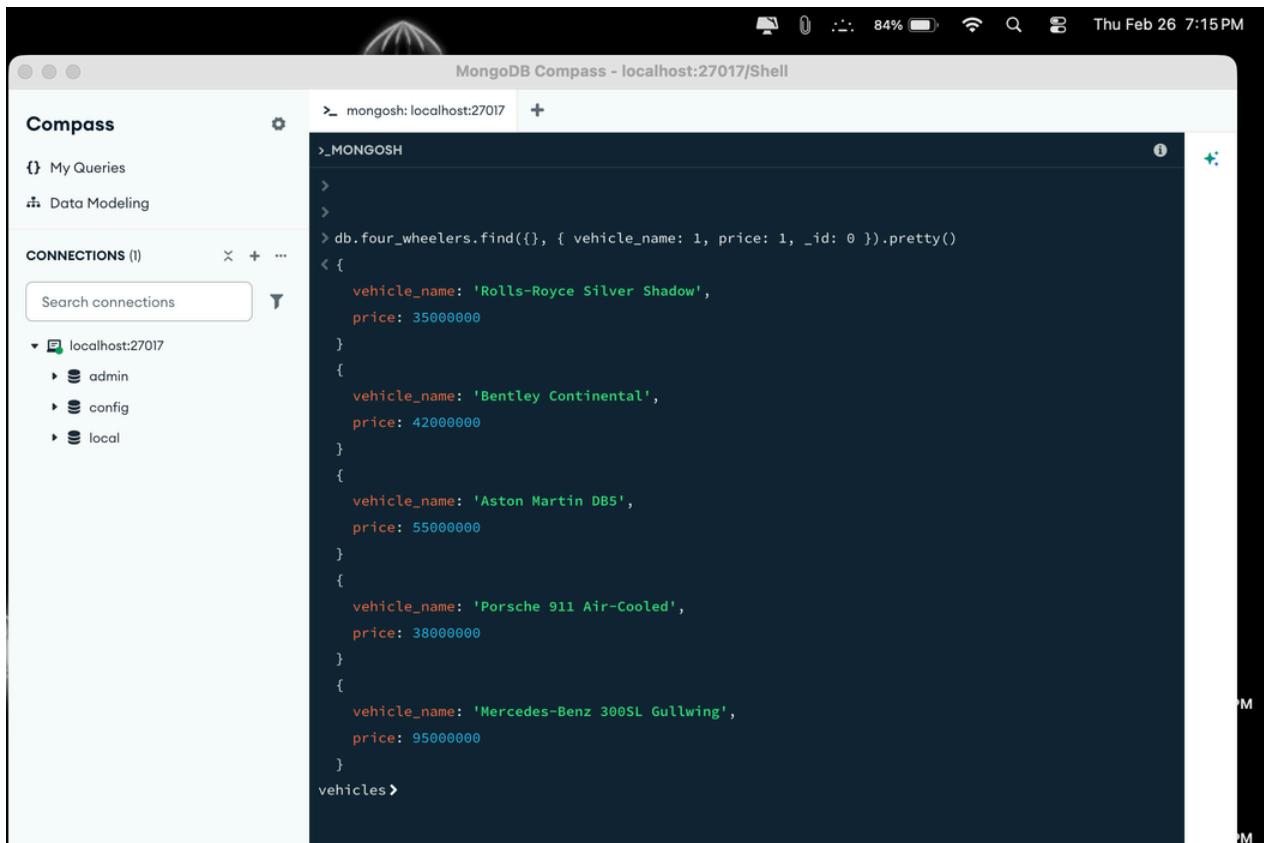
→ From two.wheelers :



The screenshot shows the MongoDB Compass interface. On the left, the 'Compass' sidebar is visible with sections for 'My Queries' and 'Data Modeling'. Below it, the 'CONNECTIONS' section shows a connection to 'localhost:27017' with databases 'admin', 'config', and 'local'. The main area is titled 'mongosh: localhost:27017/Shell' and contains the following command and its output:

```
> db.two_wheelers.find({}, { bike_name: 1, price: 1, _id: 0 }).pretty()
< [
  {
    bike_name: 'Honda CB Shine',
    price: 77500
  },
  {
    bike_name: 'TVS Jupiter',
    price: 74000
  },
  {
    bike_name: 'Bajaj Pulsar NS200',
    price: 145000
  },
  {
    bike_name: 'Yamaha FZ-S V3',
    price: 115000
  },
  {
    bike_name: 'Honda Activa 6G',
    price: 82000
  }
]
vehicles>
```

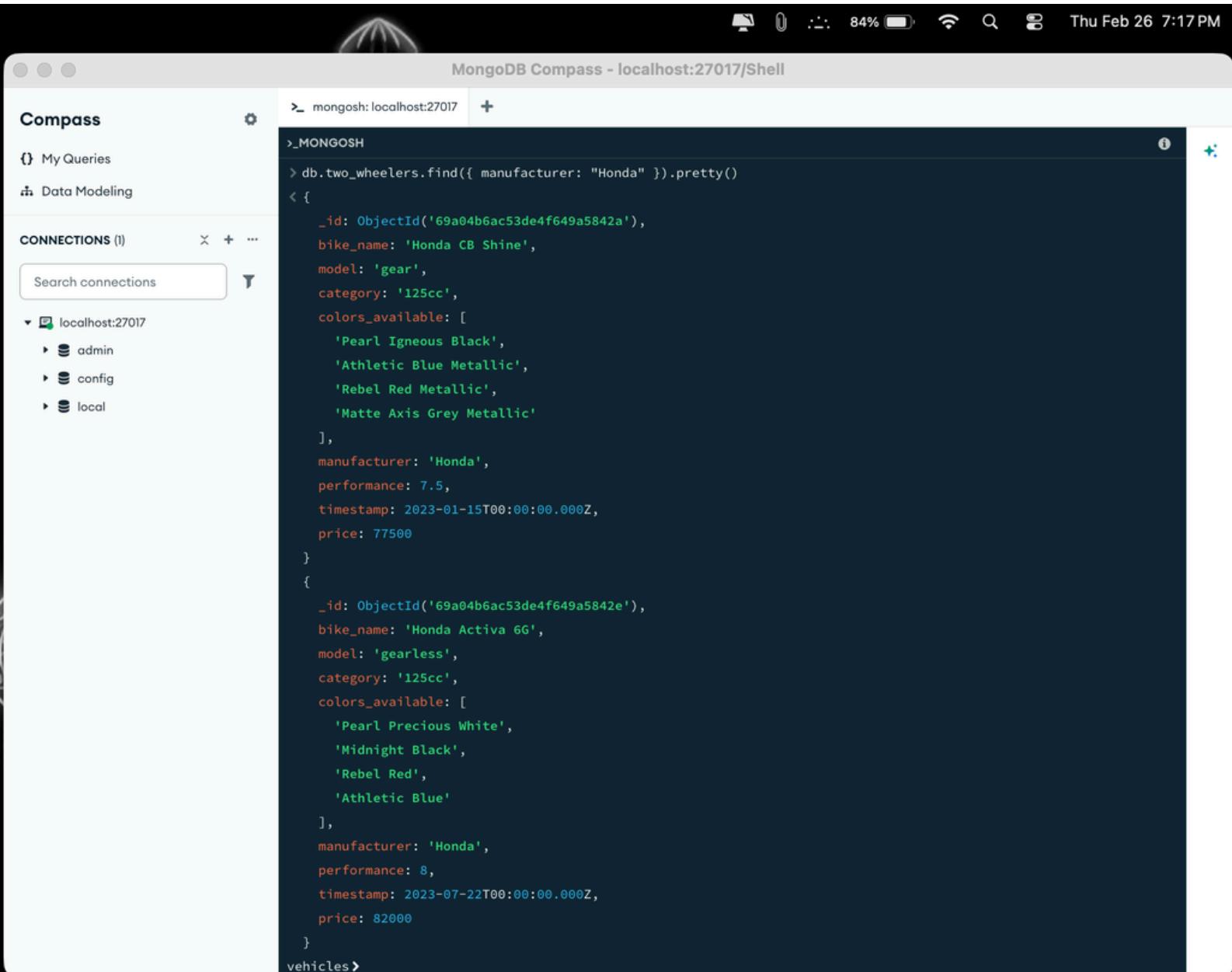
→ From four.wheelers :



The screenshot shows the MongoDB Compass interface. The 'Compass' sidebar and 'CONNECTIONS' section are identical to the previous screenshot. The main area is titled 'mongosh: localhost:27017/Shell' and contains the following command and its output:

```
> db.four_wheelers.find({}, { vehicle_name: 1, price: 1, _id: 0 }).pretty()
< [
  {
    vehicle_name: 'Rolls-Royce Silver Shadow',
    price: 35000000
  },
  {
    vehicle_name: 'Bentley Continental',
    price: 42000000
  },
  {
    vehicle_name: 'Aston Martin DB5',
    price: 55000000
  },
  {
    vehicle_name: 'Porsche 911 Air-Cooled',
    price: 38000000
  },
  {
    vehicle_name: 'Mercedes-Benz 300SL Gullwing',
    price: 95000000
  }
]
vehicles>
```

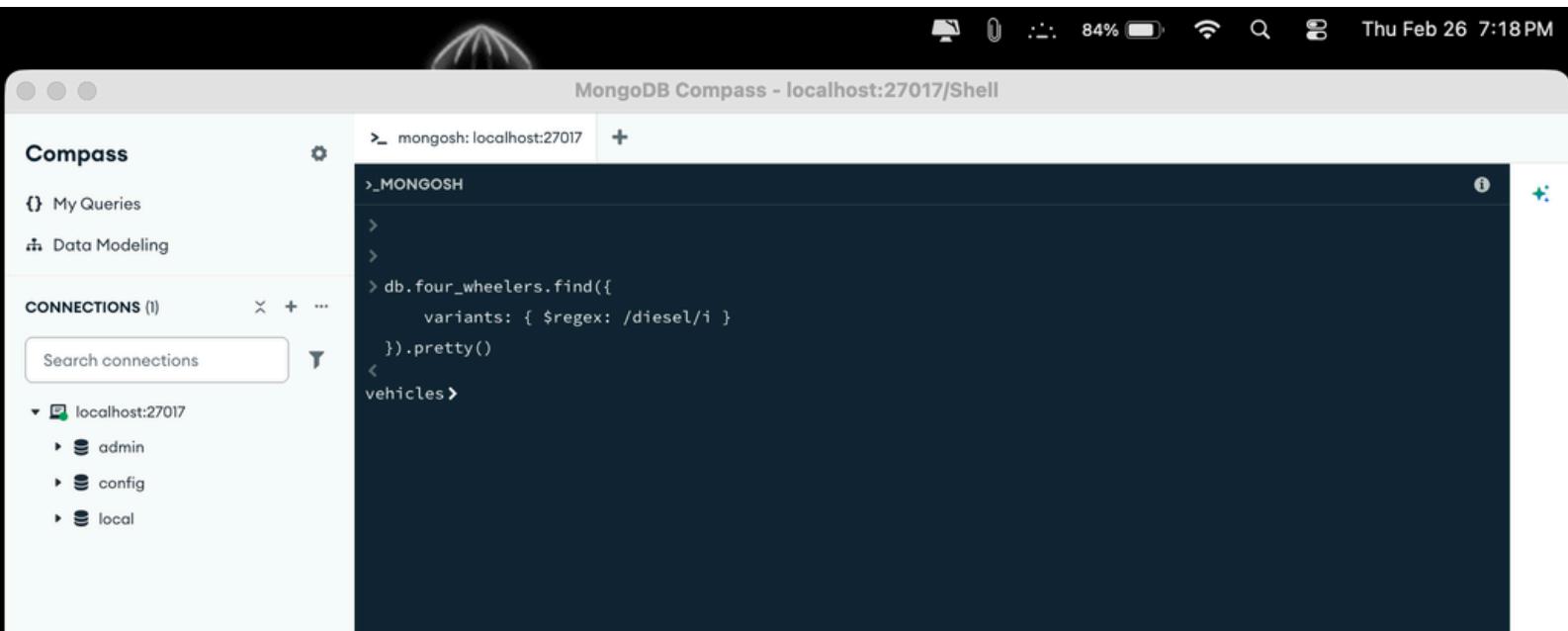
8) Write a MongoDB query to display two_wheelers from a particular company



The screenshot shows the MongoDB Compass interface running on localhost:27017. The left sidebar displays connections, with 'localhost:27017' selected. The main pane shows the results of a query run in the mongo shell:

```
>_ mongosh:localhost:27017
>_ MONGOSH
> db.two_wheelers.find({ manufacturer: "Honda" }).pretty()
< [
  {
    "_id": ObjectId('69a04b6ac53de4f649a5842a'),
    "bike_name": "Honda CB Shine",
    "model": "gear",
    "category": "125cc",
    "colors_available": [
      "Pearl Igneous Black",
      "Athletic Blue Metallic",
      "Rebel Red Metallic",
      "Matte Axis Grey Metallic"
    ],
    "manufacturer": "Honda",
    "performance": 7.5,
    "timestamp": 2023-01-15T00:00:00.000Z,
    "price": 77500
  },
  {
    "_id": ObjectId('69a04b6ac53de4f649a5842e'),
    "bike_name": "Honda Activa 6G",
    "model": "gearless",
    "category": "125cc",
    "colors_available": [
      "Pearl Precious White",
      "Midnight Black",
      "Rebel Red",
      "Athletic Blue"
    ],
    "manufacturer": "Honda",
    "performance": 8,
    "timestamp": 2023-07-22T00:00:00.000Z,
    "price": 82000
  }
]
vehicles>
```

9) Write a MongoDB query to display four_wheelers available in diesel variants



The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'Compass' navigation, 'My Queries', 'Data Modeling', and a 'CONNECTIONS' section listing 'localhost:27017' with databases 'admin', 'config', and 'local'. The main area is titled 'MongoDB Compass - localhost:27017/Shell'. It displays a mongo shell session with the following command and output:

```
>_ mongosh: localhost:27017 +  
>_ MONGOSH  
>  
>  
> db.four_wheelers.find({  
    variants: { $regex: /diesel/i }  
}).pretty()  
<  
vehicles>
```

The status bar at the top right shows system icons and the date/time: 'Thu Feb 26 7:18 PM'.

10) Write a MongoDB query to display vehicles name, category and manufacturer details whose rating is more than 5.

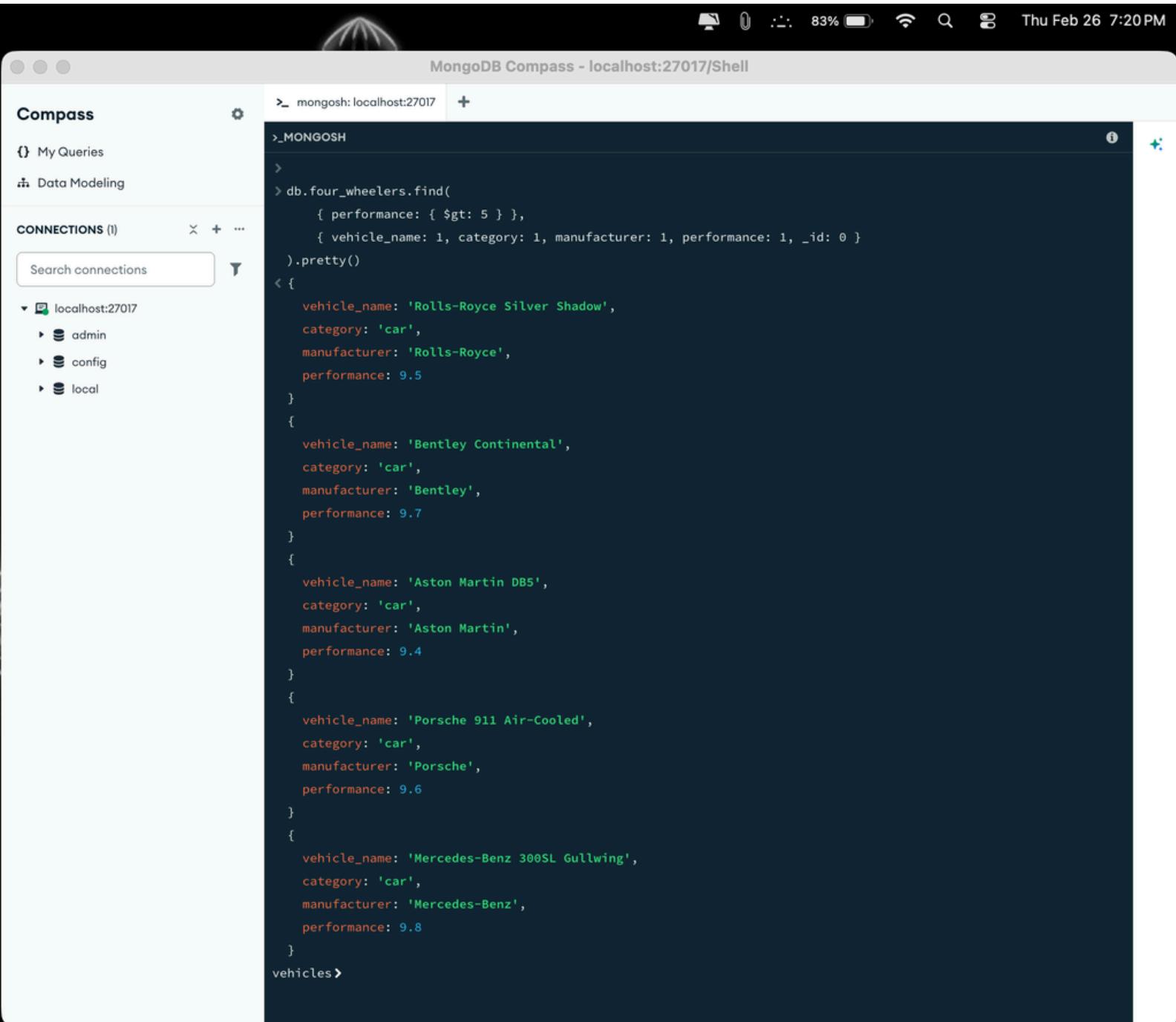
→ From two wheelers :

The screenshot shows the MongoDB Compass interface running on localhost:27017. The left sidebar displays 'Compass' sections for 'My Queries' and 'Data Modeling', and a 'CONNECTIONS' list with 'localhost:27017' selected, showing databases 'admin', 'config', and 'local'. The main pane shows the MongoDB shell with the following command and results:

```
>_ mongosh: localhost:27017
>_ MONGOSH
>
>
> db.two_wheelers.find(
  { performance: { $gt: 5 } },
  { bike_name: 1, category: 1, manufacturer: 1, performance: 1, _id: 0 }
).pretty()
< [
  {
    bike_name: 'Honda CB Shine',
    category: '125cc',
    manufacturer: 'Honda',
    performance: 7.5
  },
  {
    bike_name: 'TVS Jupiter',
    category: '110cc',
    manufacturer: 'TVS',
    performance: 7.8
  },
  {
    bike_name: 'Bajaj Pulsar NS200',
    category: '200cc',
    manufacturer: 'Bajaj',
    performance: 9
  },
  {
    bike_name: 'Yamaha FZ-S V3',
    category: '150cc',
    manufacturer: 'Yamaha',
    performance: 8.5
  },
  {
    bike_name: 'Honda Activa 6G',
    category: '125cc',
    manufacturer: 'Honda',
    performance: 8
  }
]
vehicles >
```

The results show five entries for two-wheeler vehicles: Honda CB Shine (7.5), TVS Jupiter (7.8), Bajaj Pulsar NS200 (9), Yamaha FZ-S V3 (8.5), and Honda Activa 6G (8). The '_id' field is omitted from the output.

→ From four wheelers :

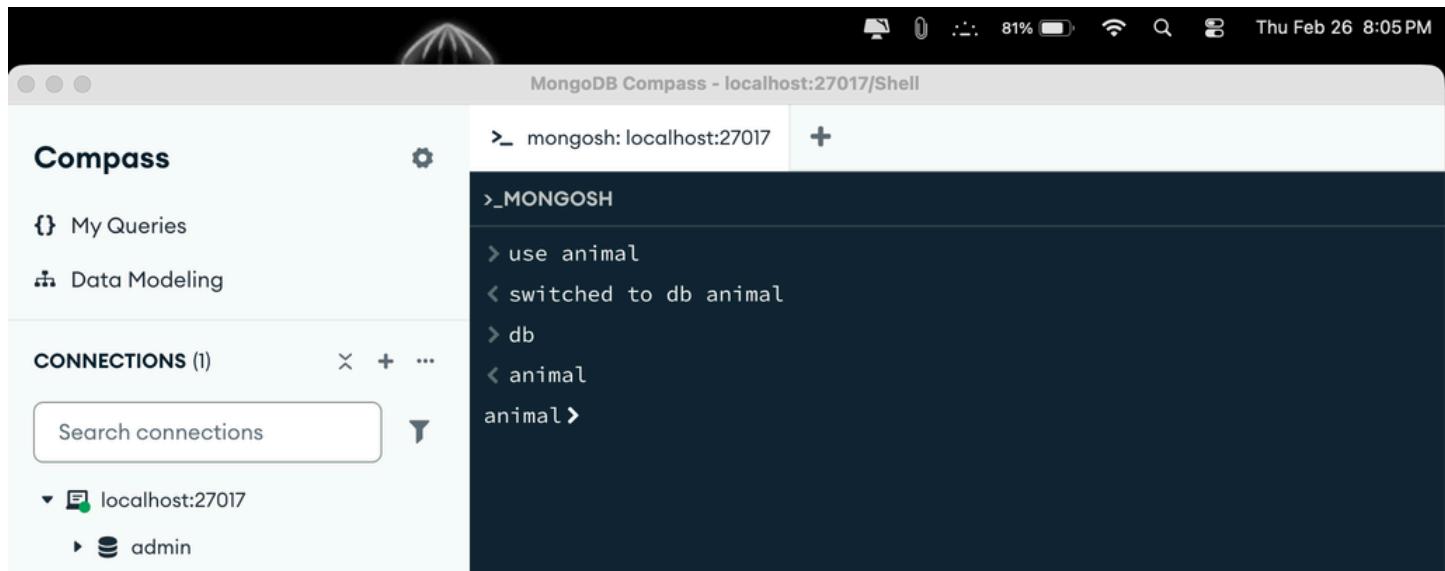


The screenshot shows the MongoDB Compass interface running on localhost:27017. The left sidebar has sections for 'Compass', 'My Queries', 'Data Modeling', and 'CONNECTIONS (1)'. Under 'CONNECTIONS', there is a connection to 'localhost:27017' with databases 'admin', 'config', and 'local'. The main area is titled 'MongoDB Compass - localhost:27017/Shell' and contains the following MONGOSH session:

```
>_ mongosh:localhost:27017 +  
  >  
  > db.four_wheelers.find(  
      { performance: { $gt: 5 } },  
      { vehicle_name: 1, category: 1, manufacturer: 1, performance: 1, _id: 0 }  
    ).pretty()  
  < {  
      vehicle_name: 'Rolls-Royce Silver Shadow',  
      category: 'car',  
      manufacturer: 'Rolls-Royce',  
      performance: 9.5  
    }  
    {  
      vehicle_name: 'Bentley Continental',  
      category: 'car',  
      manufacturer: 'Bentley',  
      performance: 9.7  
    }  
    {  
      vehicle_name: 'Aston Martin DB5',  
      category: 'car',  
      manufacturer: 'Aston Martin',  
      performance: 9.4  
    }  
    {  
      vehicle_name: 'Porsche 911 Air-Cooled',  
      category: 'car',  
      manufacturer: 'Porsche',  
      performance: 9.6  
    }  
    {  
      vehicle_name: 'Mercedes-Benz 300SL Gullwing',  
      category: 'car',  
      manufacturer: 'Mercedes-Benz',  
      performance: 9.8  
    }  
  vehicles >
```

2. Use MongoDB to implement the following DB operations for a Zoo

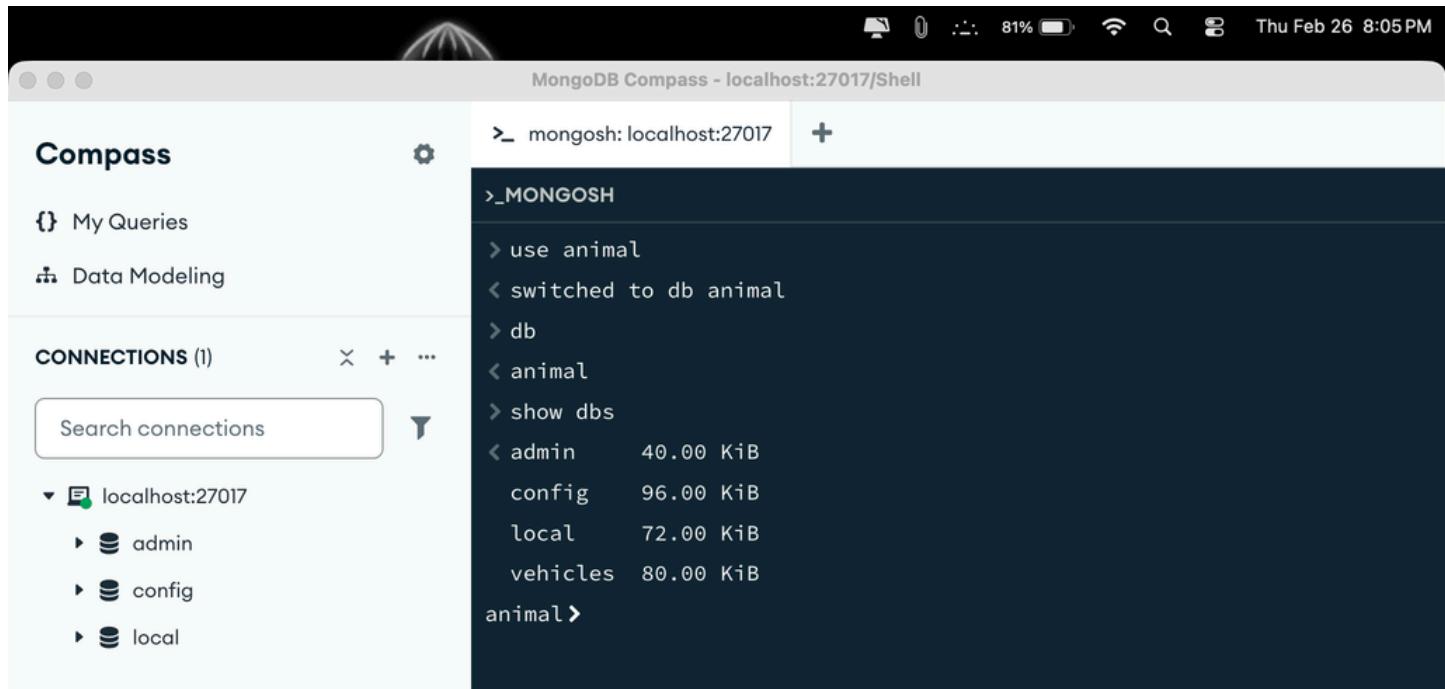
1) Create a database called 'animal' and write a MongoDB query to select database as "animal".



The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'Compass' at the top, followed by 'My Queries' and 'Data Modeling'. Below that is a 'CONNECTIONS (1)' section with a dropdown menu and a search bar labeled 'Search connections'. Under 'CONNECTIONS', it shows a connection to 'localhost:27017' with a user 'admin'. The main area is titled 'MongoDB Compass - localhost:27017/Shell'. It has tabs for 'mongosh: localhost:27017' and '+'. The 'mongosh' tab contains the following MongoDB shell session:

```
>_ MONGOSH
> use animal
< switched to db animal
> db
< animal
animal>
```

2) Write a MongoDB query to display all the databases.



The screenshot shows the same MongoDB Compass interface as the previous one. The 'CONNECTIONS' section now lists four databases: 'admin', 'config', 'local', and 'animal'. The main area still shows the MongoDB shell session in the 'mongosh' tab:

```
>_ MONGOSH
> use animal
< switched to db animal
> db
< animal
> show dbs
< admin      40.00 KiB
 config     96.00 KiB
 local      72.00 KiB
 vehicles   80.00 KiB
animal>
```

3) Create a collection called 'wild_animals'. (use capping) and Create a collection called 'domestic_animals'.

```
>_ mongosh: localhost:27017/Shell
>_ MONGOSH
>
> db.createCollection("wild_animals", {
  capped: true,
  size: 10000,
  max: 100
})
< { ok: 1 }
> db.createCollection("domestic_animals")
< { ok: 1 }
animal>
```

4) Add 5 wild_animal details to the collection named 'wild_animals'. Each document consists of following fields as animal_name, nature (harm or harmless), favorite_foods (meat, rabbits, deer etc) as array, care_taker_name, life span (in years), timestamp (when the animal registered at the Zoo) and expenses.

```
>_ MongoDB Compass  Connections  Edit  View  Window  Help
>_ Data Modeling
>_ CONNECTIONS ()  +  ...
>_ Search connections
>_ localhost:27017
>_ admin
>_ config
>_ local
>_ MONGOSH
> db.wild_animals.insertMany([
  {
    animal_name: "Bengal Tiger",
    nature: "harm",
    favorite_foods: ["deer", "wild boar", "buffalo", "rabbits", "antelope"],
    care_taker_name: "Varun",
    life_span: 15,
    timestamp: new Date("2021-03-10"),
    expenses: 85000
  },
  {
    animal_name: "African Lion",
    nature: "harm",
    favorite_foods: ["zebra", "wildebeest", "buffalo", "gazelle", "warthog"],
    care_taker_name: "Siva",
    life_span: 16,
    timestamp: new Date("2020-07-18"),
    expenses: 95000
  },
  {
    animal_name: "Giant Panda",
    nature: "harmless",
    favorite_foods: ["bamboo shoots", "bamboo leaves", "fruits", "vegetables", "honey"],
    care_taker_name: "Pavan",
    life_span: 20,
    timestamp: new Date("2022-01-25"),
    expenses: 120000
  },
  {
    animal_name: "Nile Crocodile",
    nature: "harm",
    favorite_foods: ["fish", "zebra", "wildebeest", "birds", "turtles"],
    care_taker_name: "Shreya",
    life_span: 45,
    timestamp: new Date("2019-11-05"),
    expenses: 60000
  },
  {
    animal_name: "Red Fox",
    nature: "harmless",
    favorite_foods: ["rabbits", "birds", "insects", "berries", "frogs"],
    care_taker_name: "Dosth",
    life_span: 18,
    timestamp: new Date("2023-05-14"),
    expenses: 35000
  }
])
< [
  {
    acknowledged: true,
    insertedIds: [
      '0': ObjectId('69a05c45c3de4f649a58434'),
      '1': ObjectId('69a05c45c3de4f649a58435'),
      '2': ObjectId('69a05c45c3de4f649a58436'),
      '3': ObjectId('69a05c45c3de4f649a58437'),
      '4': ObjectId('69a05c45c3de4f649a58438')
    ]
  }
]
animal>
```

5) Add 5 domestic-animal details to the collection named 'domestic_animals'. Each document consists of following fields as animal_name, gender (male or female), favorite_foods (meat, rabbits, deer etc) as array, animal_petsname, life span (in years), timestamp (when the animal registered at the Zoo) and expenses.

The screenshot shows the MongoDB Compass interface with a connection to 'localhost:27017'. The 'Data Modeling' tab is selected. In the main pane, a command is being run in the shell:

```
animal> db.domestic_animals.insertMany([{
  animal_name: "Dog",
  gender: "male",
  favorite_foods: ["meat", "bones", "dog biscuits", "milk", "rice"],
  animal_petsname: "Bruno",
  life_span: 13,
  timestamp: new Date("2021-04-12"),
  expenses: 15000
},
{
  animal_name: "Cat",
  gender: "female",
  favorite_foods: ["fish", "milk", "chicken", "cat food", "tuna"],
  animal_petsname: "Kitty",
  life_span: 15,
  timestamp: new Date("2022-02-20"),
  expenses: 12000
},
{
  animal_name: "Cow",
  gender: "female",
  favorite_foods: ["grass", "hay", "wheat", "corn", "vegetables"],
  animal_petsname: "Ganga",
  life_span: 20,
  timestamp: new Date("2020-08-15"),
  expenses: 25000
},
{
  animal_name: "Rabbit",
  gender: "male",
  favorite_foods: ["carrots", "lettuce", "cabbage", "hay", "apples"],
  animal_petsname: "Fluffy",
  life_span: 8,
  timestamp: new Date("2023-01-10"),
  expenses: 8000
},
{
  animal_name: "Parrot",
  gender: "male",
  favorite_foods: ["seeds", "fruits", "nuts", "vegetables", "berries"],
  animal_petsname: "Mittu",
  life_span: 25,
  timestamp: new Date("2022-09-05"),
  expenses: 10000
}])
```

The response shows the inserted documents with their ObjectIds:

```
< {
  acknowledged: true,
  insertedIds: [
    '69a05cf1c53de4f649a58439',
    '69a05cf1c53de4f649a5843a',
    '69a05cf1c53de4f649a5843b',
    '69a05cf1c53de4f649a5843c',
    '69a05cf1c53de4f649a5843d'
  ]
}>
```

6) Write a MongoDB query to display all documents available in wild_animals and domestic_animals

→ Documents from wild animals :

The screenshot shows the MongoDB Compass interface running on localhost:27017. The left sidebar displays connections, with 'localhost:27017' selected. The main pane shows the command line interface (mongosh) with the following query and its results:

```
>_MONGOSH
>
> db.wild_animals.find().pretty()
< {
  "_id": ObjectId('69a05c45c53de4f649a58434'),
  "animal_name": "Bengal Tiger",
  "nature": "harm",
  "favorite_foods": [
    "deer",
    "wild boar",
    "buffalo",
    "rabbits",
    "antelope"
  ],
  "care_taker_name": "Varun",
  "life_span": 15,
  "timestamp": 2021-03-10T00:00:00.000Z,
  "expenses": 85000
}
{
  "_id": ObjectId('69a05c45c53de4f649a58435'),
  "animal_name": "African Lion",
  "nature": "harm",
  "favorite_foods": [
    "zebra",
    "wildebeest",
    "buffalo",
    "gazelle",
    "warthog"
  ],
  "care_taker_name": "Siva",
  "life_span": 16,
  "timestamp": 2020-07-18T00:00:00.000Z,
  "expenses": 95000
}
```

The results show two documents from the 'wild_animals' collection. The first document is for a 'Bengal Tiger' with an ID of '69a05c45c53de4f649a58434'. The second document is for an 'African Lion' with an ID of '69a05c45c53de4f649a58435'. Both documents include fields for animal name, nature, favorite foods, care taker name, life span, timestamp, and expenses.

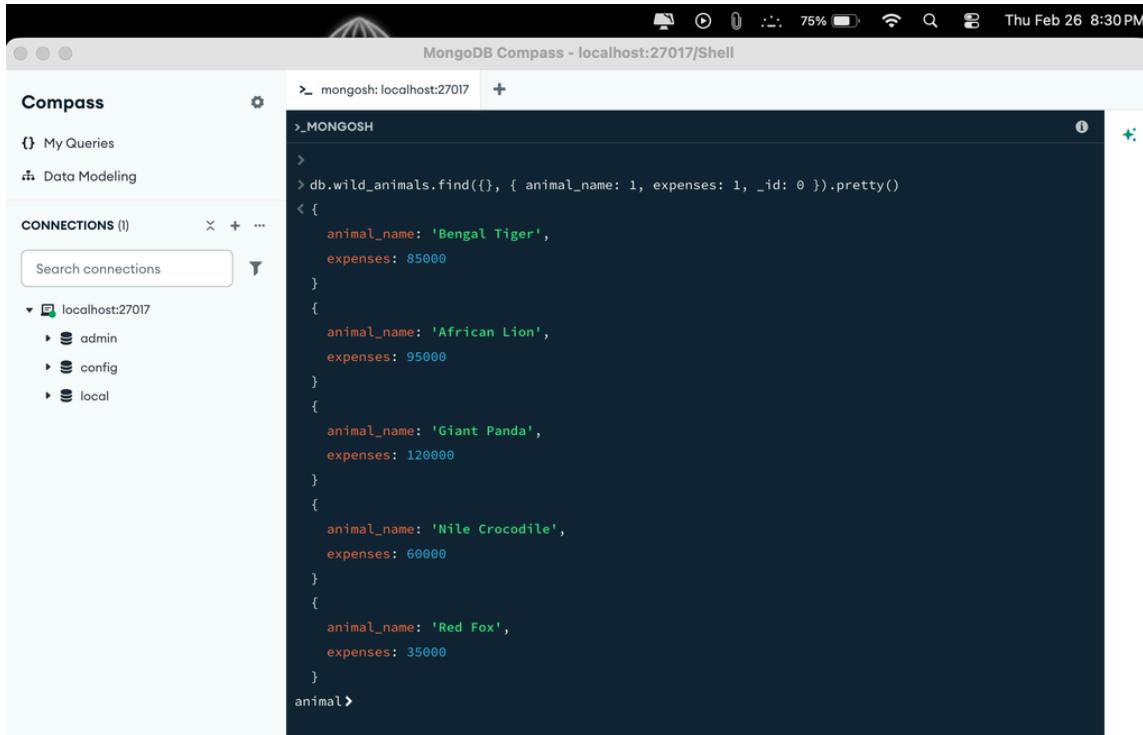
→ Documents from domestic_animals :

The screenshot shows the MongoDB Compass interface connected to localhost:27017. The left sidebar displays connections, and the main area shows the results of a query on the 'domestic_animals' collection. The results are displayed in a JSON-like format, showing three documents (Dog, Cat, Cow) with their respective details.

```
>_ MONGOSH
> db.domestic_animals.find().pretty()
< [
  {
    "_id": ObjectId('69a05cf1c53de4f649a58439'),
    "animal_name": "Dog",
    "gender": "male",
    "favorite_foods": [
      "meat",
      "bones",
      "dog biscuits",
      "milk",
      "rice"
    ],
    "animal_petname": "Bruno",
    "life_span": 13,
    "timestamp": 2021-04-12T00:00:00.000Z,
    "expenses": 15000
  },
  {
    "_id": ObjectId('69a05cf1c53de4f649a5843a'),
    "animal_name": "Cat",
    "gender": "female",
    "favorite_foods": [
      "fish",
      "milk",
      "chicken",
      "cat food",
      "tuna"
    ],
    "animal_petname": "Kitty",
    "life_span": 15,
    "timestamp": 2022-02-20T00:00:00.000Z,
    "expenses": 12000
  },
  {
    "_id": ObjectId('69a05cf1c53de4f649a5843b'),
    "animal_name": "Cow",
    "gender": "female",
    "favorite_foods": [
      "grass",
      "hay",
      "wheat",
      "corn",
      "vegetables"
    ],
    "animal_petname": "Ganga",
    "life_span": 20,
    "timestamp": 2020-08-15T00:00:00.000Z,
    "expenses": 18000
  }
]
```

7) Write a MongoDB query to display only animal name and expenses in all the collection of the database

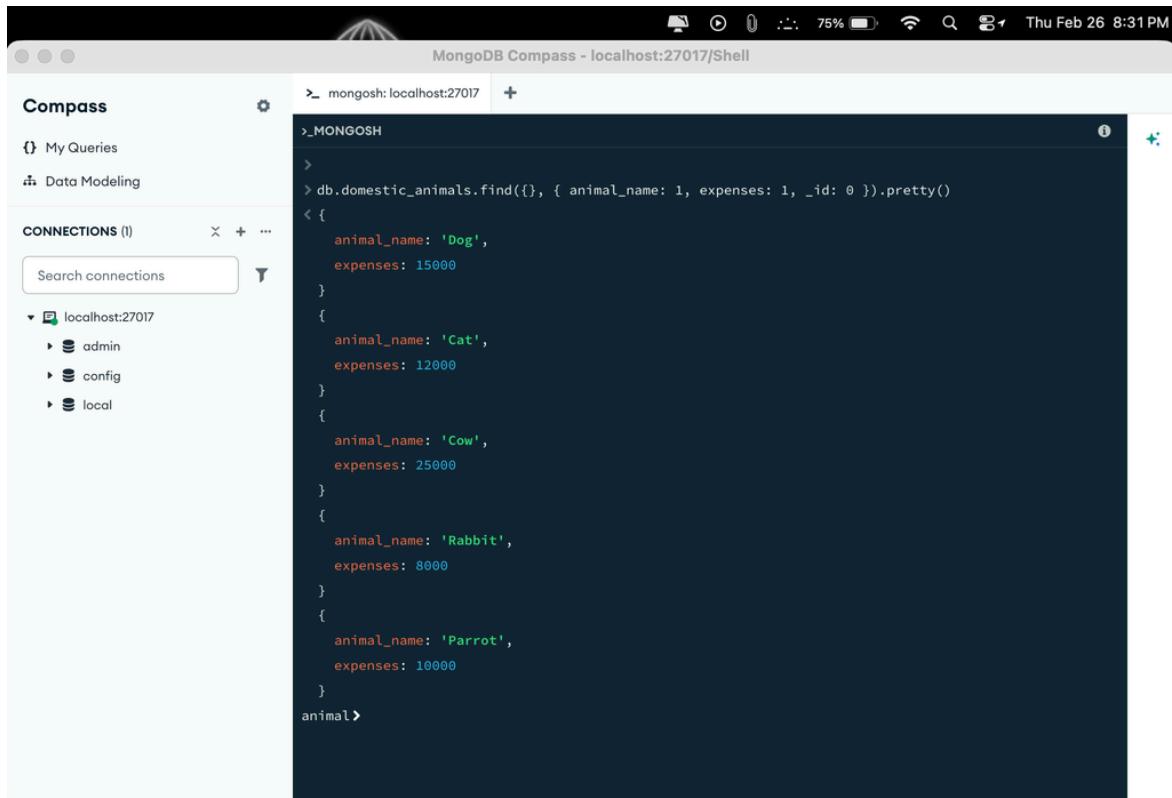
→ From wild animals:



The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays 'Compass' with sections for 'My Queries' and 'Data Modeling'. Under 'CONNECTIONS (1)', it lists 'localhost:27017' with sub-options for 'admin', 'config', and 'local'. The main panel is titled 'mongosh: localhost:27017/Shell' and contains the following command and its output:

```
>_MONGOSH
>
> db.wild_animals.find({}, { animal_name: 1, expenses: 1, _id: 0 }).pretty()
< [
  {
    animal_name: 'Bengal Tiger',
    expenses: 85000
  },
  {
    animal_name: 'African Lion',
    expenses: 95000
  },
  {
    animal_name: 'Giant Panda',
    expenses: 120000
  },
  {
    animal_name: 'Nile Crocodile',
    expenses: 60000
  },
  {
    animal_name: 'Red Fox',
    expenses: 35000
  }
]
animal>
```

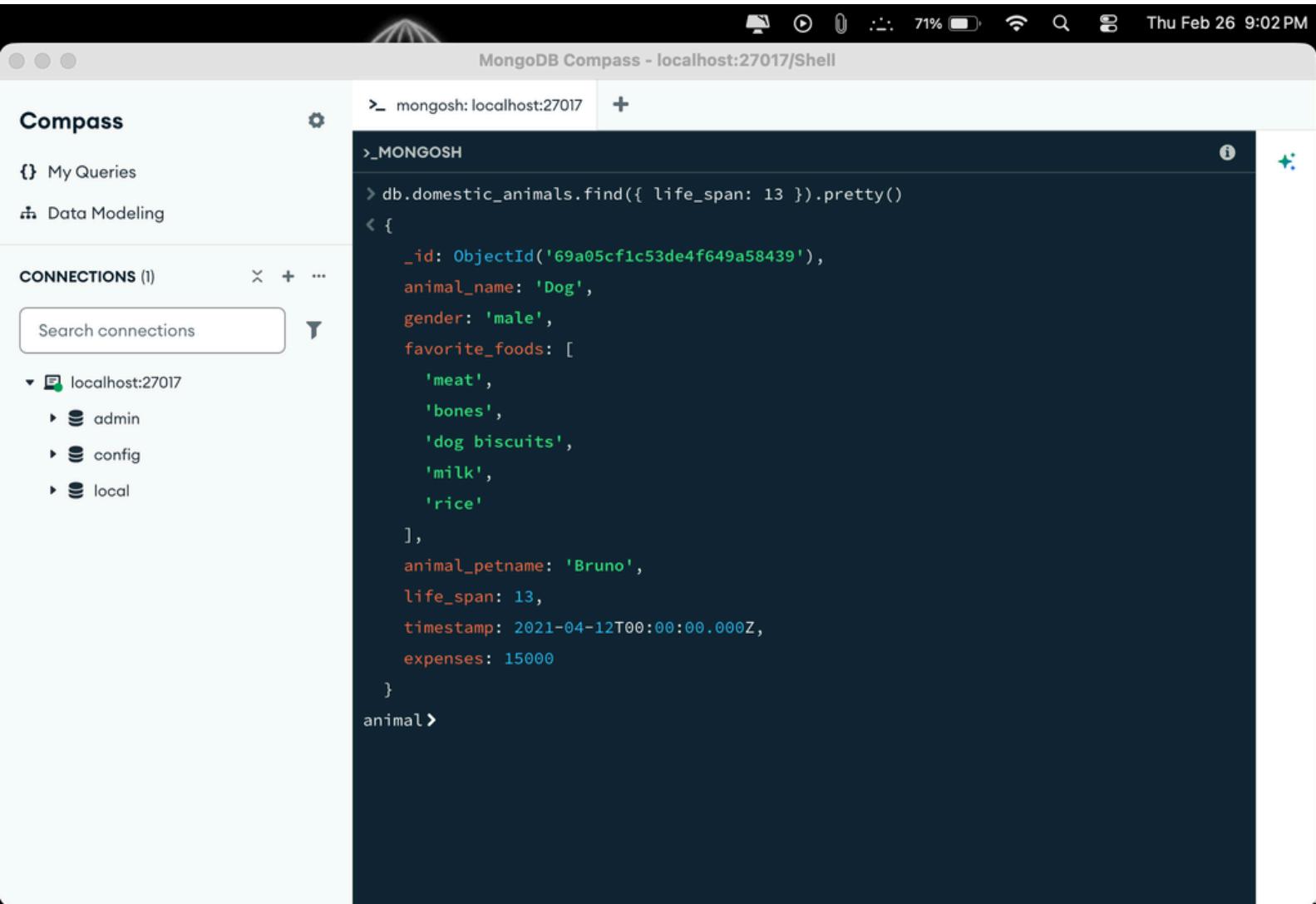
→ From domestic animals:



The screenshot shows the MongoDB Compass interface. The sidebar and connection list are identical to the previous screenshot. The main panel is titled 'mongosh: localhost:27017/Shell' and contains the following command and its output:

```
>_MONGOSH
>
> db.domestic_animals.find({}, { animal_name: 1, expenses: 1, _id: 0 }).pretty()
< [
  {
    animal_name: 'Dog',
    expenses: 15000
  },
  {
    animal_name: 'Cat',
    expenses: 12000
  },
  {
    animal_name: 'Cow',
    expenses: 25000
  },
  {
    animal_name: 'Rabbit',
    expenses: 8000
  },
  {
    animal_name: 'Parrot',
    expenses: 10000
  }
]
animal>
```

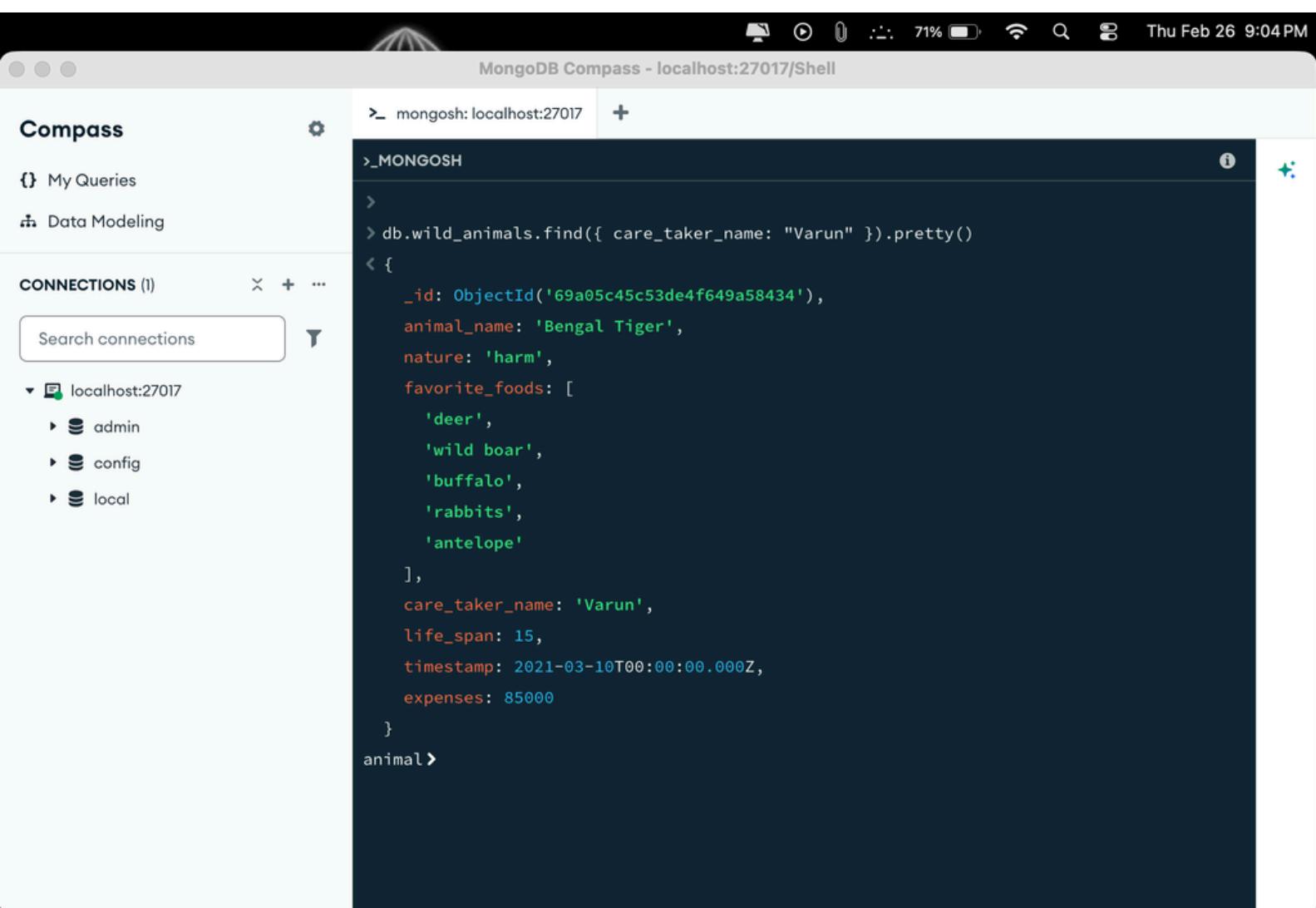
8) Write a MongoDB query to display domestic_animals whose life is a particular year



The screenshot shows the MongoDB Compass interface running on localhost:27017. The left sidebar displays 'Compass' with sections for 'My Queries' and 'Data Modeling', and a 'CONNECTIONS' list containing 'localhost:27017' with sub-options for 'admin', 'config', and 'local'. The main window title is 'MongoDB Compass - localhost:27017/Shell'. The query results pane shows the output of the command: 'db.domestic_animals.find({ life_span: 13 }).pretty()'. The result is a single document representing a dog named 'Bruno' with a life span of 13 years.

```
>_ MONGOSH
> db.domestic_animals.find({ life_span: 13 }).pretty()
< {
  _id: ObjectId('69a05cf1c53de4f649a58439'),
  animal_name: 'Dog',
  gender: 'male',
  favorite_foods: [
    'meat',
    'bones',
    'dog biscuits',
    'milk',
    'rice'
  ],
  animal_petsname: 'Bruno',
  life_span: 13,
  timestamp: 2021-04-12T00:00:00.000Z,
  expenses: 15000
}
animal>
```

9) Write a MongoDB query to display wild_animals available under a particular care_taker

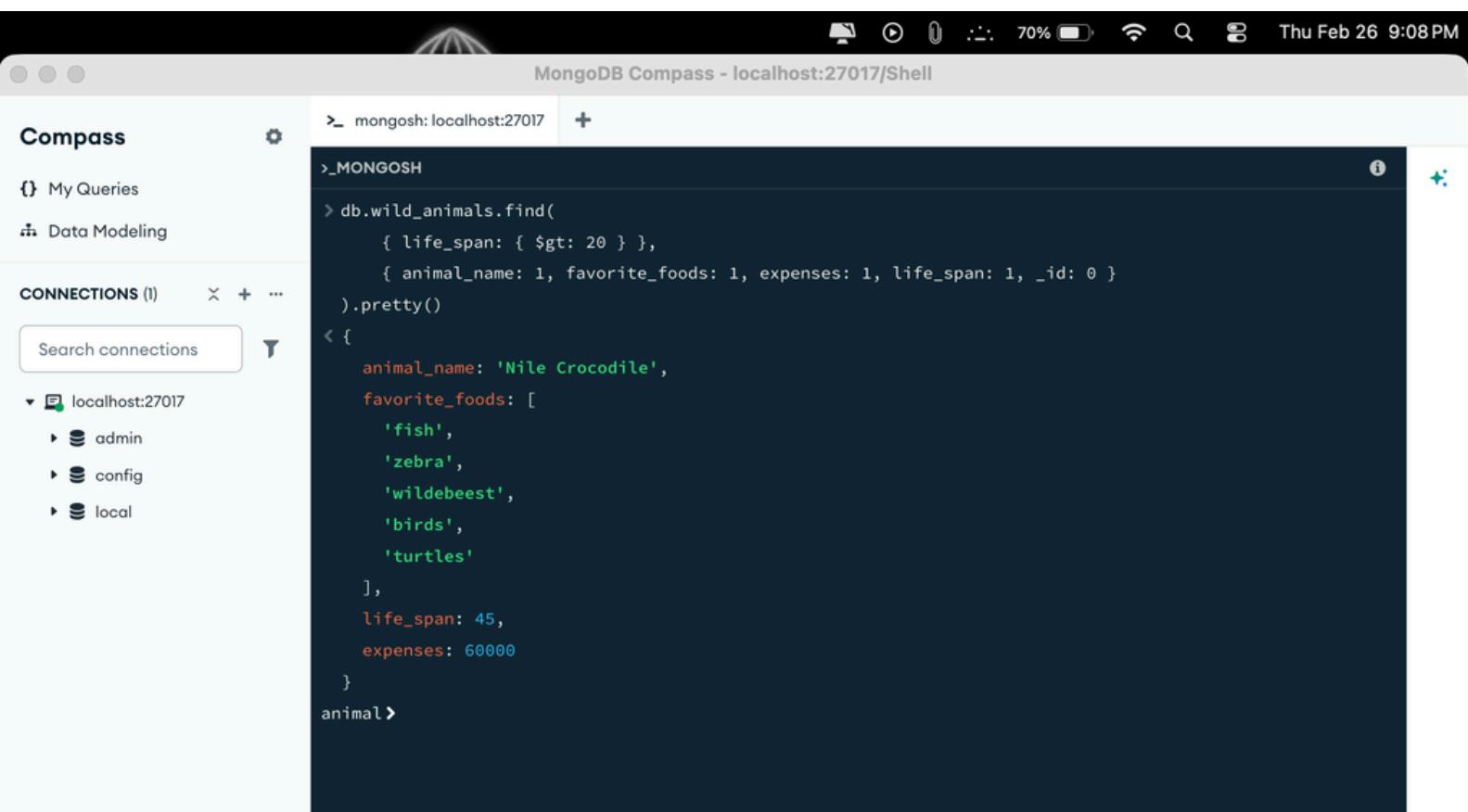


The screenshot shows the MongoDB Compass interface running on localhost:27017. The left sidebar displays connections, with 'localhost:27017' selected. The main pane shows a mongo shell session with the following command and result:

```
>_ mongosh: localhost:27017
>_ MONGOSH
>
> db.wild_animals.find({ care_taker_name: "Varun" }).pretty()
< {
  "_id": ObjectId('69a05c45c53de4f649a58434'),
  "animal_name": 'Bengal Tiger',
  "nature": 'harm',
  "favorite_foods": [
    'deer',
    'wild boar',
    'buffalo',
    'rabbits',
    'antelope'
  ],
  "care_taker_name": 'Varun',
  "life_span": 15,
  "timestamp": 2021-03-10T00:00:00.000Z,
  "expenses": 85000
}
animal>
```

10) Write a MongoDB query to display animal name, favorite_foods and expenses details whose lifespan is more than 5 years.

→ From Wild Animals :

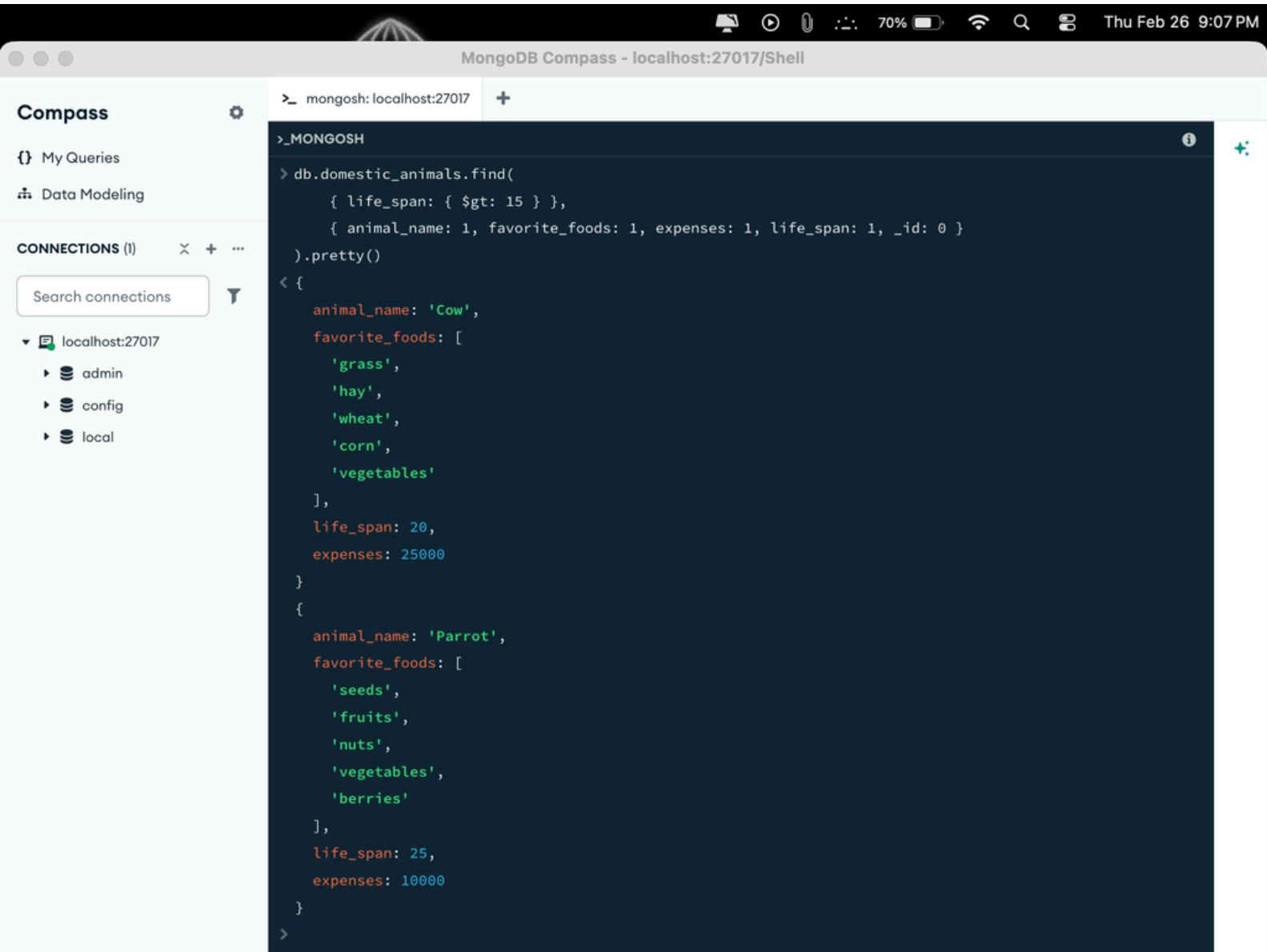


The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'Compass' at the top, followed by 'My Queries' and 'Data Modeling'. Below that is a 'CONNECTIONS' section with a dropdown set to 'localhost:27017' containing 'admin', 'config', and 'local'. The main area is the 'mongosh: localhost:27017' shell window. It displays the following MongoDB query:

```
>_MONGOSH
> db.wild_animals.find(
  { life_span: { $gt: 20 } },
  { animal_name: 1, favorite_foods: 1, expenses: 1, life_span: 1, _id: 0 }
).pretty()
< {
  animal_name: 'Nile Crocodile',
  favorite_foods: [
    'fish',
    'zebra',
    'wildebeest',
    'birds',
    'turtles'
  ],
  life_span: 45,
  expenses: 60000
}
animal>
```

The query filters documents in the 'wild_animals' collection where 'life_span' is greater than 20. It then selects the '_id' field and projects 'animal_name', 'favorite_foods', 'expenses', and 'life_span'. The results are displayed in a pretty-printed JSON format, showing one document for the 'Nile Crocodile' with its favorite foods and expenses.

→ From Domestic Animals:



The screenshot shows the MongoDB Compass interface. On the left, the sidebar has sections for 'Compass', 'My Queries', 'Data Modeling', and 'CONNECTIONS' (localhost:27017). The 'CONNECTIONS' section lists databases: admin, config, and local. In the main area, the title bar says 'MongoDB Compass - localhost:27017/Shell'. The shell window displays the following command and its results:

```
>_ mongosh: localhost:27017 +  
  >_ MONGOSH  
  > db.domestic_animals.find(  
    { life_span: { $gt: 15 } },  
    { animal_name: 1, favorite_foods: 1, expenses: 1, life_span: 1, _id: 0 }  
  ).pretty()  
  < {  
    animal_name: 'Cow',  
    favorite_foods: [  
      'grass',  
      'hay',  
      'wheat',  
      'corn',  
      'vegetables'  
    ],  
    life_span: 20,  
    expenses: 25000  
  }  
  {  
    animal_name: 'Parrot',  
    favorite_foods: [  
      'seeds',  
      'fruits',  
      'nuts',  
      'vegetables',  
      'berries'  
    ],  
    life_span: 25,  
    expenses: 10000  
  }  
  >
```