LAB ASSIGNMENT 6

SUBMITTED TO: Dr. Gopikrishnan

VIT-AP

ANDHRA PRADESH

NAME: HRITAM BASAK

REG NO.: 23BCE7618

SLOT: L14+L15

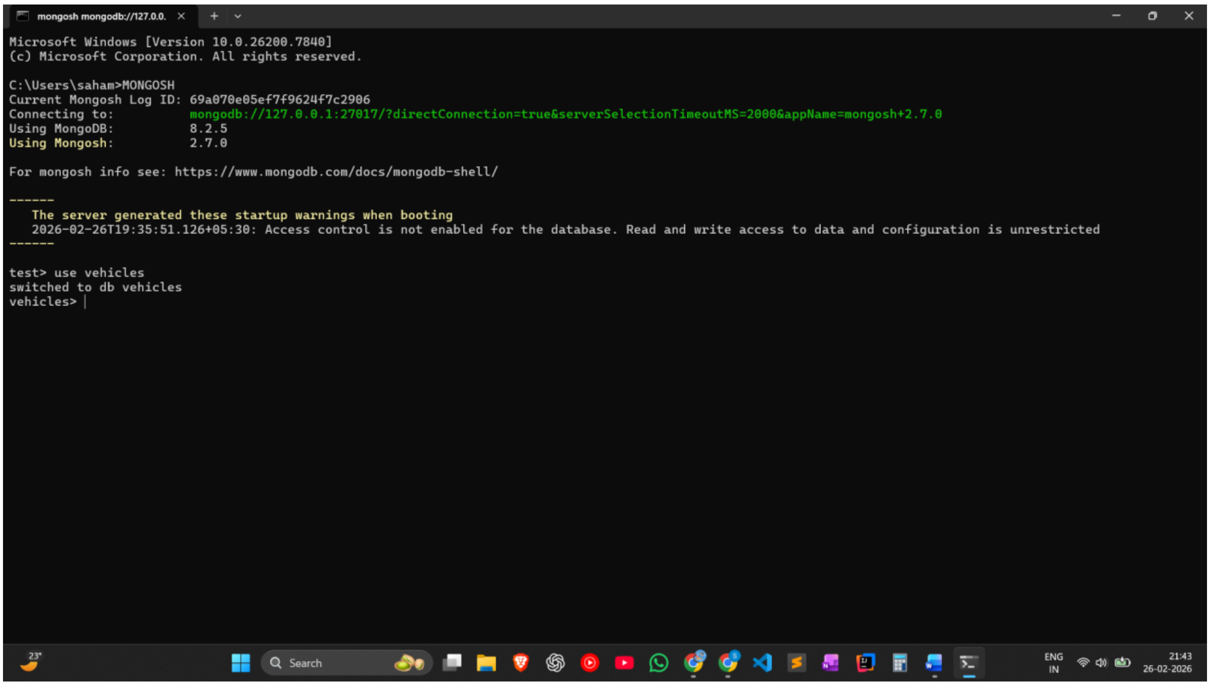
VENUE: CB 502B

COURSE CODE: CSE4004

COURSE NAME: WEB TECHNOLOGIES

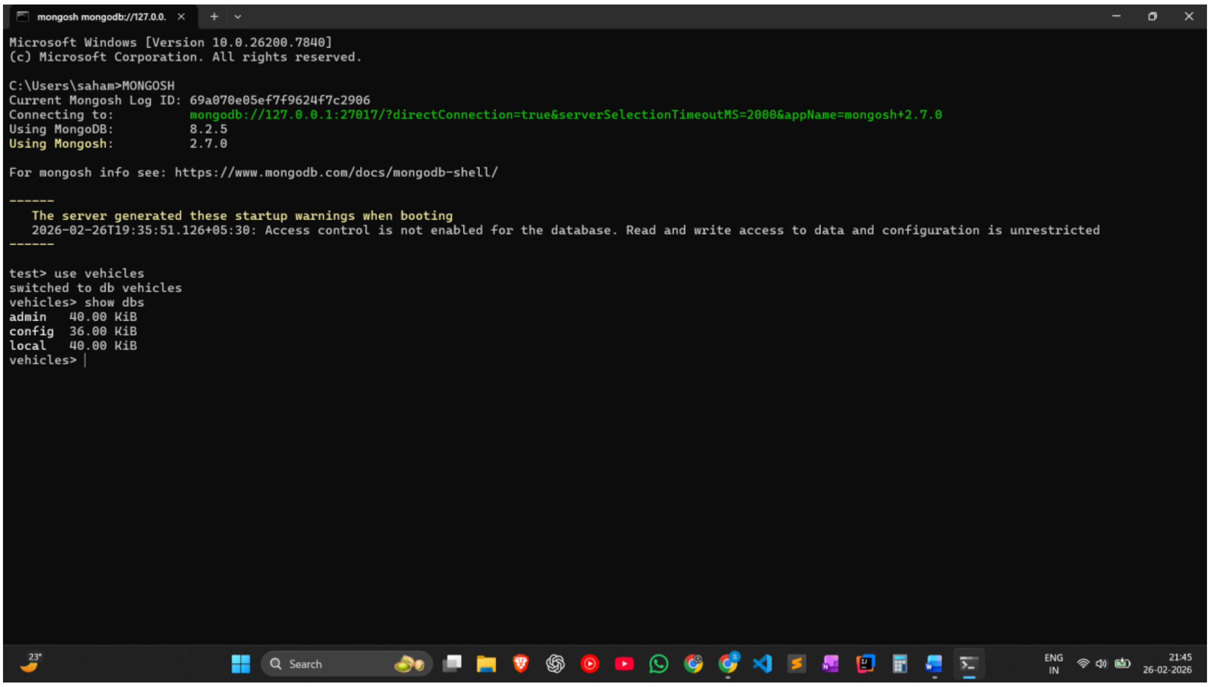
DATABASE – 1: Vehicles

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



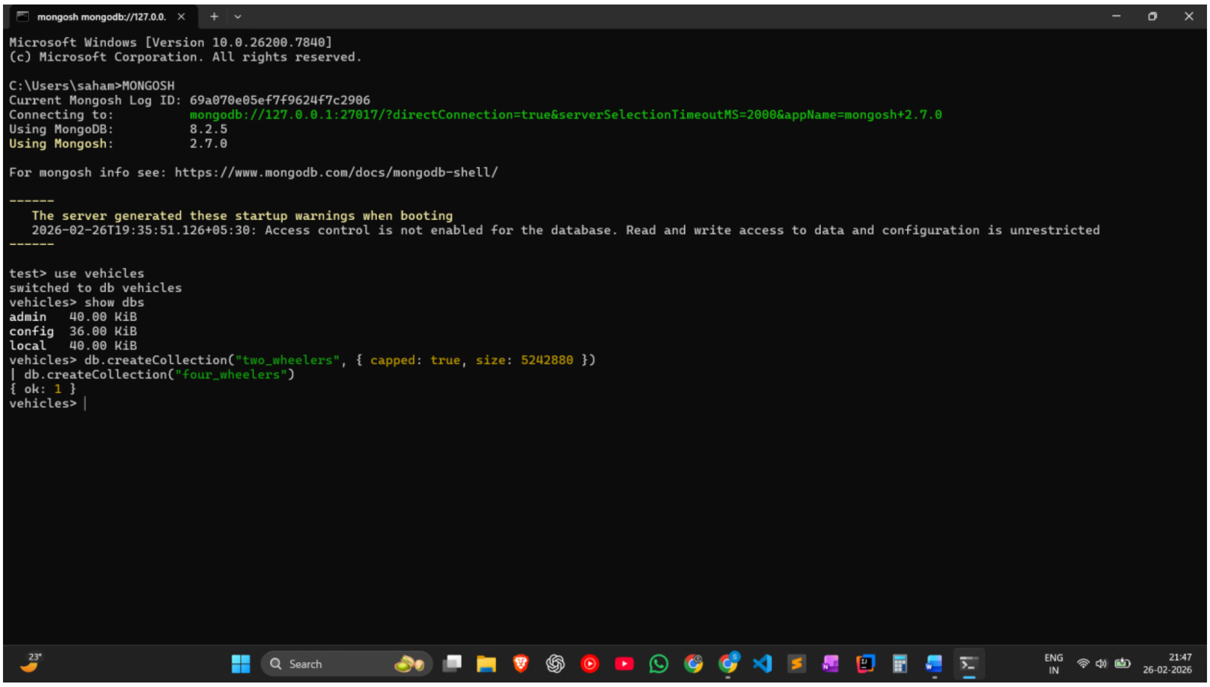
```
Microsoft Windows [Version 10.0.26200.7840]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saham>MONGOSH
Current Mongosh Log ID: 69a070e05ef7f9624f7c2906
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.7.0
Using MongoDB:          8.2.5
Using Mongosh:          2.7.0

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

------
   The server generated these startup warnings when booting
   2026-02-26T19:35:51.126+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use vehicles
switched to db vehicles
vehicles> |
```

Program 2: *Write a MongoDB query to display all the databases.*



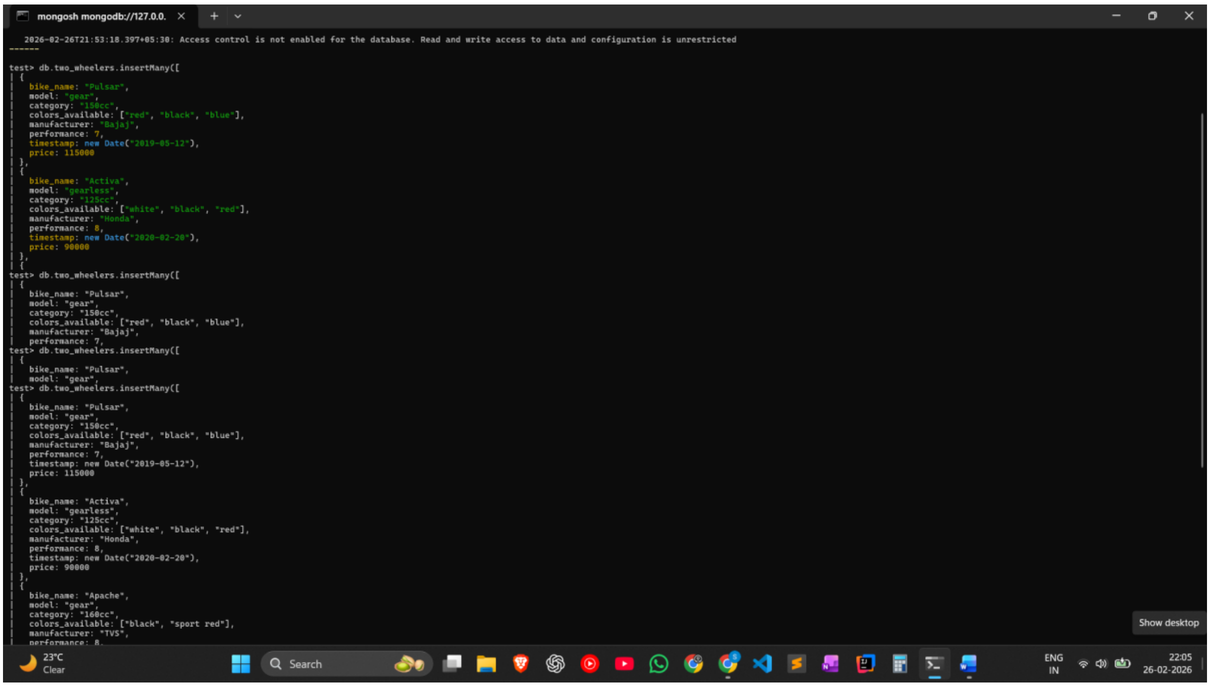
```
Microsoft Windows [Version 10.0.26200.7840]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saham>MONGOSH
Current Mongosh Log ID: 69a070e05ef7f9624f7c2906
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.7.0
Using MongoDB:          8.2.5
Using Mongosh:          2.7.0

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

------
   The server generated these startup warnings when booting
   2026-02-26T19:35:51.126+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use vehicles
switched to db vehicles
vehicles> show dbs
admin    40.00 KiB
config   36.00 KiB
local    40.00 KiB
vehicles> |
```

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



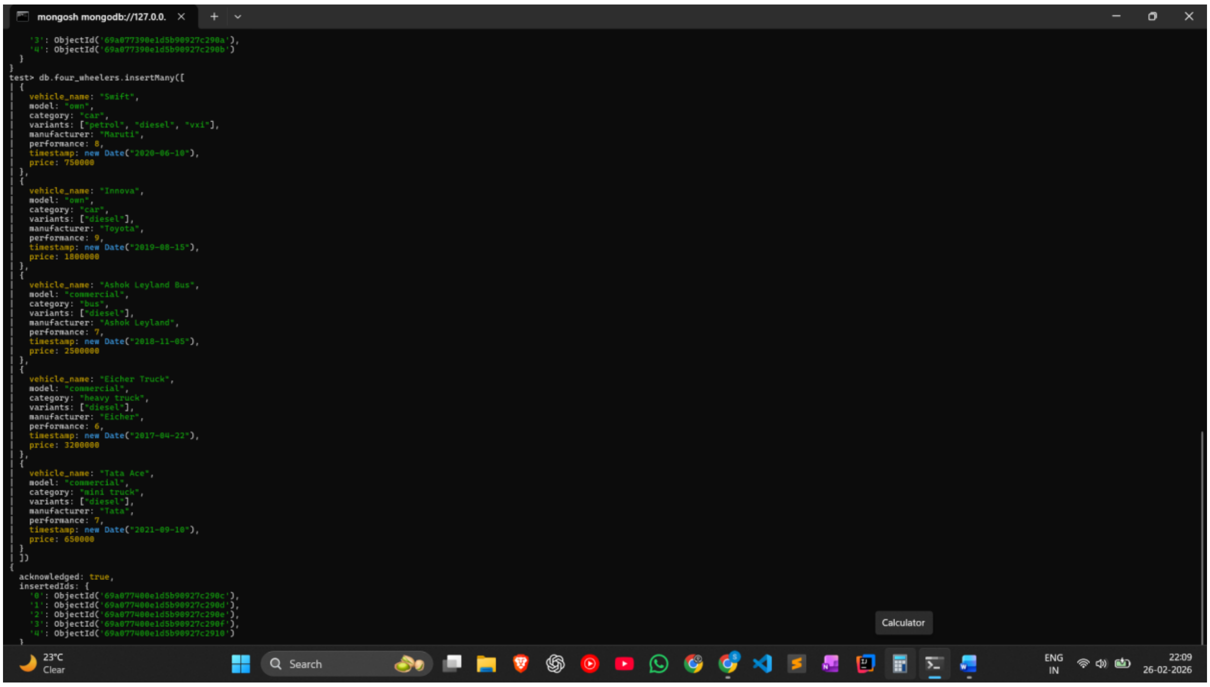
```
Microsoft Windows [Version 10.0.26200.7840]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saham>MONGOSH
Current Mongosh Log ID: 69a070e05ef7f9624f7c2906
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.7.0
Using MongoDB:          8.2.5
Using Mongosh:          2.7.0

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

------
   The server generated these startup warnings when booting
   2026-02-26T19:35:51.126+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use vehicles
switched to db vehicles
vehicles> show dbs
admin   40.00 KiB
config  36.00 KiB
local   40.00 KiB
vehicles> db.createCollection("two_wheelers", { capped: true, size: 5242880 })
| db.createCollection("four_wheelers")
{ ok: 1 }
vehicles> |
```

Program 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.



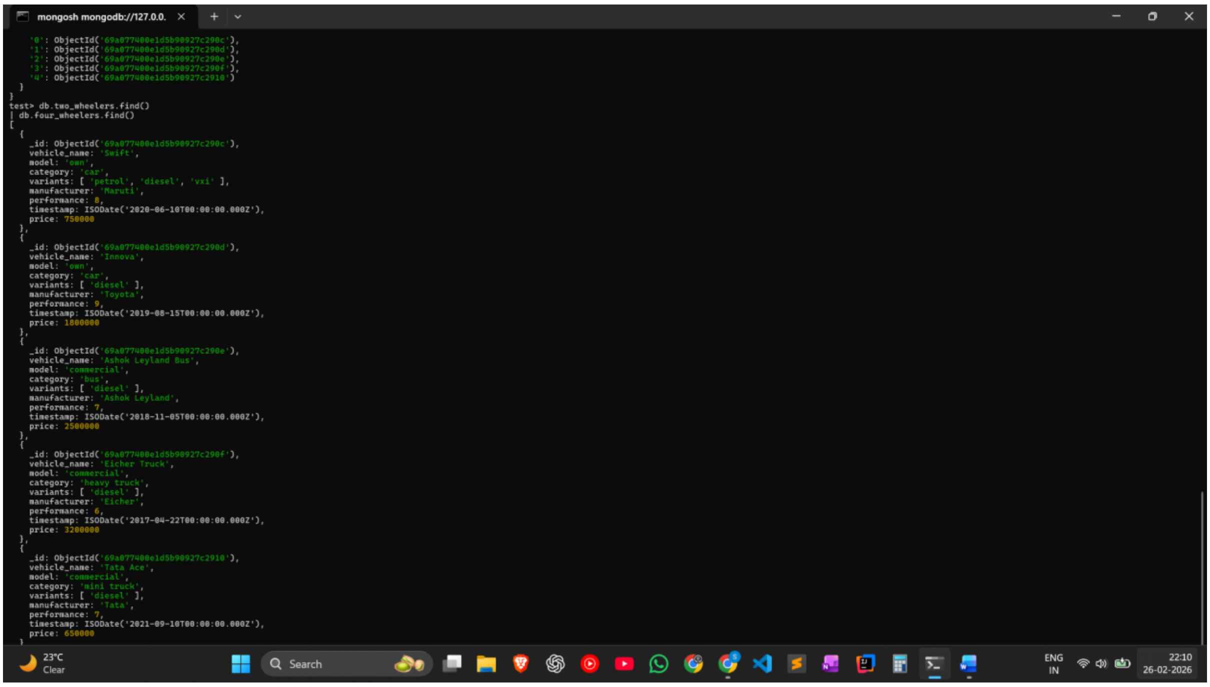
```
   2026-02-26T21:53:18.397+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> db.two_wheelers.insertMany([
| {
|   bike_name: "Pulsar",
|   model: "gear",
|   category: "150cc",
|   colors_available: ["red", "black", "blue"],
|   manufacturer: "Bajaj",
|   performance: 7,
|   timestamp: new Date("2019-05-12"),
|   price: 115000
| },
| {
|   bike_name: "Activa",
|   model: "gearless",
|   category: "125cc",
|   colors_available: ["white", "black", "red"],
|   manufacturer: "Honda",
|   performance: 8,
|   timestamp: new Date("2020-02-20"),
|   price: 90000
| },
| {
test> db.two_wheelers.insertMany([
| {
|   bike_name: "Pulsar",
|   model: "gear",
|   category: "150cc",
|   colors_available: ["red", "black", "blue"],
|   manufacturer: "Bajaj",
|   performance: 7,
test> db.two_wheelers.insertMany([
| {
|   bike_name: "Pulsar",
|   model: "gear",
test> db.two_wheelers.insertMany([
| {
|   bike_name: "Pulsar",
|   model: "gear",
|   category: "150cc",
|   colors_available: ["red", "black", "blue"],
|   manufacturer: "Bajaj",
|   performance: 7,
|   timestamp: new Date("2019-05-12"),
|   price: 115000
| },
| {
|   bike_name: "Activa",
|   model: "gearless",
|   category: "125cc",
|   colors_available: ["white", "black", "red"],
|   manufacturer: "Honda",
|   performance: 8,
|   timestamp: new Date("2020-02-20"),
|   price: 90000
| },
| {
|   bike_name: "Apache",
|   model: "gear",
|   category: "160cc",
|   colors_available: ["black", "sport red"],
|   manufacturer: "TVS",
|   performance: 8,
```

Program 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.



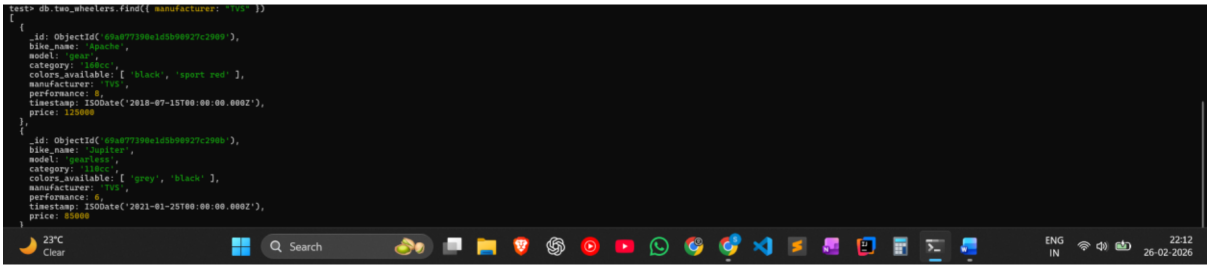
```
    '3': ObjectId('69a077390e1d5b90927c290a'),
    '4': ObjectId('69a077390e1d5b90927c290b')
  }
}
test> db.four_wheelers.insertMany([
|  {
|    vehicle_name: "Swift",
|    model: "own",
|    category: "car",
|    variants: ["petrol", "diesel", "vxi"],
|    manufacturer: "Maruti",
|    performance: 8,
|    timestamp: new Date("2020-06-10"),
|    price: 750000
|  },
|  {
|    vehicle_name: "Innova",
|    model: "own",
|    category: "car",
|    variants: ["diesel"],
|    manufacturer: "Toyota",
|    performance: 9,
|    timestamp: new Date("2019-08-15"),
|    price: 1800000
|  },
|  {
|    vehicle_name: "Ashok Leyland Bus",
|    model: "commercial",
|    category: "bus",
|    variants: ["diesel"],
|    manufacturer: "Ashok Leyland",
|    performance: 7,
|    timestamp: new Date("2018-11-05"),
|    price: 2500000
|  },
|  {
|    vehicle_name: "Eicher Truck",
|    model: "commercial",
|    category: "heavy truck",
|    variants: ["diesel"],
|    manufacturer: "Eicher",
|    performance: 6,
|    timestamp: new Date("2017-04-22"),
|    price: 3200000
|  },
|  {
|    vehicle_name: "Tata Ace",
|    model: "commercial",
|    category: "mini truck",
|    variants: ["diesel"],
|    manufacturer: "Tata",
|    performance: 7,
|    timestamp: new Date("2021-09-10"),
|    price: 650000
|  }
|  ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('69a077400e1d5b90927c290c'),
    '1': ObjectId('69a077400e1d5b90927c290d'),
    '2': ObjectId('69a077400e1d5b90927c290e'),
    '3': ObjectId('69a077400e1d5b90927c290f'),
    '4': ObjectId('69a077400e1d5b90927c2910')
  }
```

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



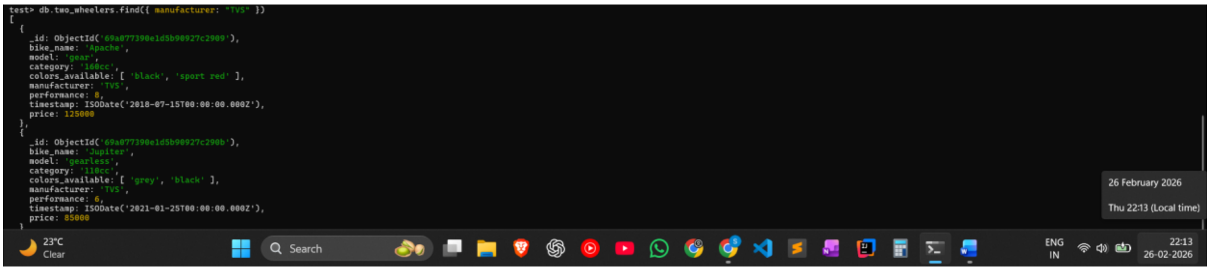
```
    '0': ObjectId('69a077400e1d5b90927c290c'),
    '1': ObjectId('69a077400e1d5b90927c290d'),
    '2': ObjectId('69a077400e1d5b90927c290e'),
    '3': ObjectId('69a077400e1d5b90927c290f'),
    '4': ObjectId('69a077400e1d5b90927c2910')
  }
}
test> db.two_wheelers.find()
| db.four_wheelers.find()
[
  {
    _id: ObjectId('69a077400e1d5b90927c290c'),
    vehicle_name: 'Swift',
    model: 'own',
    category: 'car',
    variants: [ 'petrol', 'diesel', 'vxi' ],
    manufacturer: 'Maruti',
    performance: 8,
    timestamp: ISODate('2020-06-10T00:00:00.000Z'),
    price: 750000
  },
  {
    _id: ObjectId('69a077400e1d5b90927c290d'),
    vehicle_name: 'Innova',
    model: 'own',
    category: 'car',
    variants: [ 'diesel' ],
    manufacturer: 'Toyota',
    performance: 9,
    timestamp: ISODate('2019-08-15T00:00:00.000Z'),
    price: 1800000
  },
  {
    _id: ObjectId('69a077400e1d5b90927c290e'),
    vehicle_name: 'Ashok Leyland Bus',
    model: 'commercial',
    category: 'bus',
    variants: [ 'diesel' ],
    manufacturer: 'Ashok Leyland',
    performance: 7,
    timestamp: ISODate('2018-11-05T00:00:00.000Z'),
    price: 2500000
  },
  {
    _id: ObjectId('69a077400e1d5b90927c290f'),
    vehicle_name: 'Eicher Truck',
    model: 'commercial',
    category: 'heavy truck',
    variants: [ 'diesel' ],
    manufacturer: 'Eicher',
    performance: 6,
    timestamp: ISODate('2017-04-22T00:00:00.000Z'),
    price: 3200000
  },
  {
    _id: ObjectId('69a077400e1d5b90927c2910'),
    vehicle_name: 'Tata Ace',
    model: 'commercial',
    category: 'mini truck',
    variants: [ 'diesel' ],
    manufacturer: 'Tata',
    performance: 7,
    timestamp: ISODate('2021-09-10T00:00:00.000Z'),
    price: 650000
  }
```

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



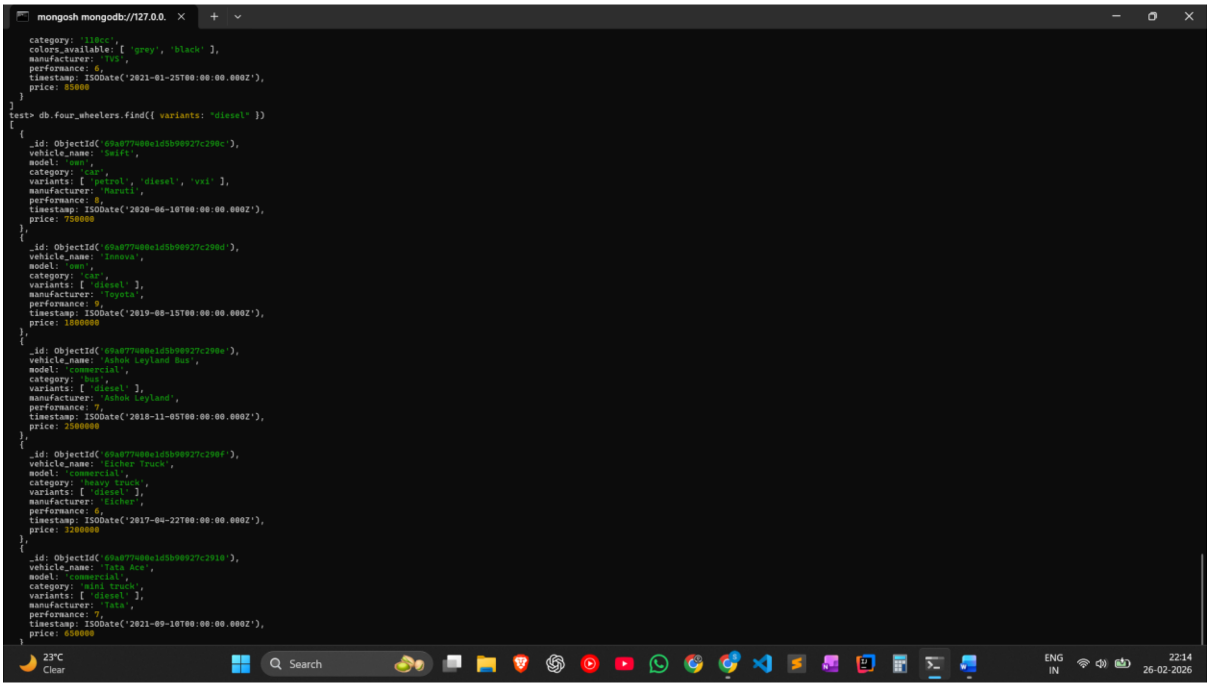
```
test> db.two_wheelers.find({ manufacturer: "TVS" })
[
  {
    _id: ObjectId('69a077390e1d5b90927c2909'),
    bike_name: 'Apache',
    model: 'gear',
    category: '160cc',
    colors_available: [ 'black', 'sport red' ],
    manufacturer: 'TVS',
    performance: 8,
    timestamp: ISODate('2018-07-15T00:00:00.000Z'),
    price: 125000
  },
  {
    _id: ObjectId('69a077390e1d5b90927c290b'),
    bike_name: 'Jupiter',
    model: 'gearless',
    category: '110cc',
    colors_available: [ 'grey', 'black' ],
    manufacturer: 'TVS',
    performance: 6,
    timestamp: ISODate('2021-01-25T00:00:00.000Z'),
    price: 85000
  }
```

Program 8: Write a MongoDB query to display two_wheelers from a particular company



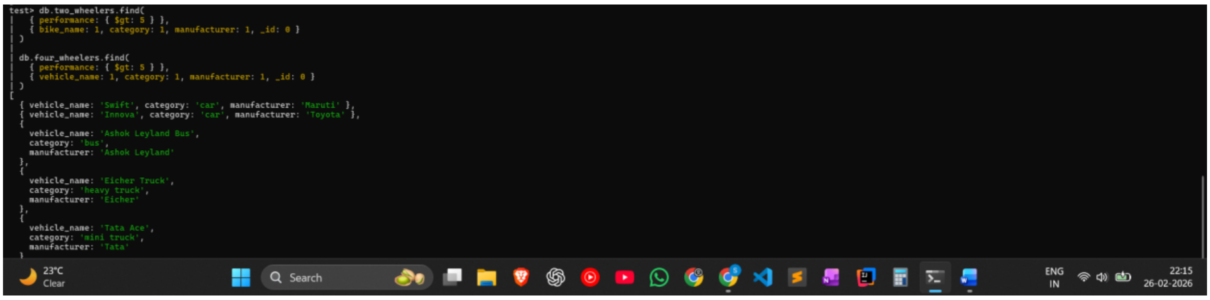
```
test> db.two_wheelers.find({ manufacturer: "TVS" })
[
  {
    _id: ObjectId('69a077390e1d5b90927c2909'),
    bike_name: 'Apache',
    model: 'gear',
    category: '160cc',
    colors_available: [ 'black', 'sport red' ],
    manufacturer: 'TVS',
    performance: 8,
    timestamp: ISODate('2018-07-15T00:00:00.000Z'),
    price: 125000
  },
  {
    _id: ObjectId('69a077390e1d5b90927c290b'),
    bike_name: 'Jupiter',
    model: 'gearless',
    category: '110cc',
    colors_available: [ 'grey', 'black' ],
    manufacturer: 'TVS',
    performance: 6,
    timestamp: ISODate('2021-01-25T00:00:00.000Z'),
    price: 85000
  }
```

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



```
    category: '110cc',
    colors_available: [ 'grey', 'black' ],
    manufacturer: 'TVS',
    performance: 6,
    timestamp: ISODate('2021-01-25T00:00:00.000Z'),
    price: 85000
  }
]
test> db.four_wheelers.find({ variants: "diesel" })
[
  {
    _id: ObjectId('69a077400e1d5b90927c290c'),
    vehicle_name: 'Swift',
    model: 'own',
    category: 'car',
    variants: [ 'petrol', 'diesel', 'vxi' ],
    manufacturer: 'Maruti',
    performance: 8,
    timestamp: ISODate('2020-06-10T00:00:00.000Z'),
    price: 750000
  },
  {
    _id: ObjectId('69a077400e1d5b90927c290d'),
    vehicle_name: 'Innova',
    model: 'own',
    category: 'car',
    variants: [ 'diesel' ],
    manufacturer: 'Toyota',
    performance: 9,
    timestamp: ISODate('2019-08-15T00:00:00.000Z'),
    price: 1800000
  },
  {
    _id: ObjectId('69a077400e1d5b90927c290e'),
    vehicle_name: 'Ashok Leyland Bus',
    model: 'commercial',
    category: 'bus',
    variants: [ 'diesel' ],
    manufacturer: 'Ashok Leyland',
    performance: 7,
    timestamp: ISODate('2018-11-05T00:00:00.000Z'),
    price: 2500000
  },
  {
    _id: ObjectId('69a077400e1d5b90927c290f'),
    vehicle_name: 'Eicher Truck',
    model: 'commercial',
    category: 'heavy truck',
    variants: [ 'diesel' ],
    manufacturer: 'Eicher',
    performance: 6,
    timestamp: ISODate('2017-04-22T00:00:00.000Z'),
    price: 3200000
  },
  {
    _id: ObjectId('69a077400e1d5b90927c2910'),
    vehicle_name: 'Tata Ace',
    model: 'commercial',
    category: 'mini truck',
    variants: [ 'diesel' ],
    manufacturer: 'Tata',
    performance: 7,
    timestamp: ISODate('2021-09-10T00:00:00.000Z'),
    price: 650000
  }
```

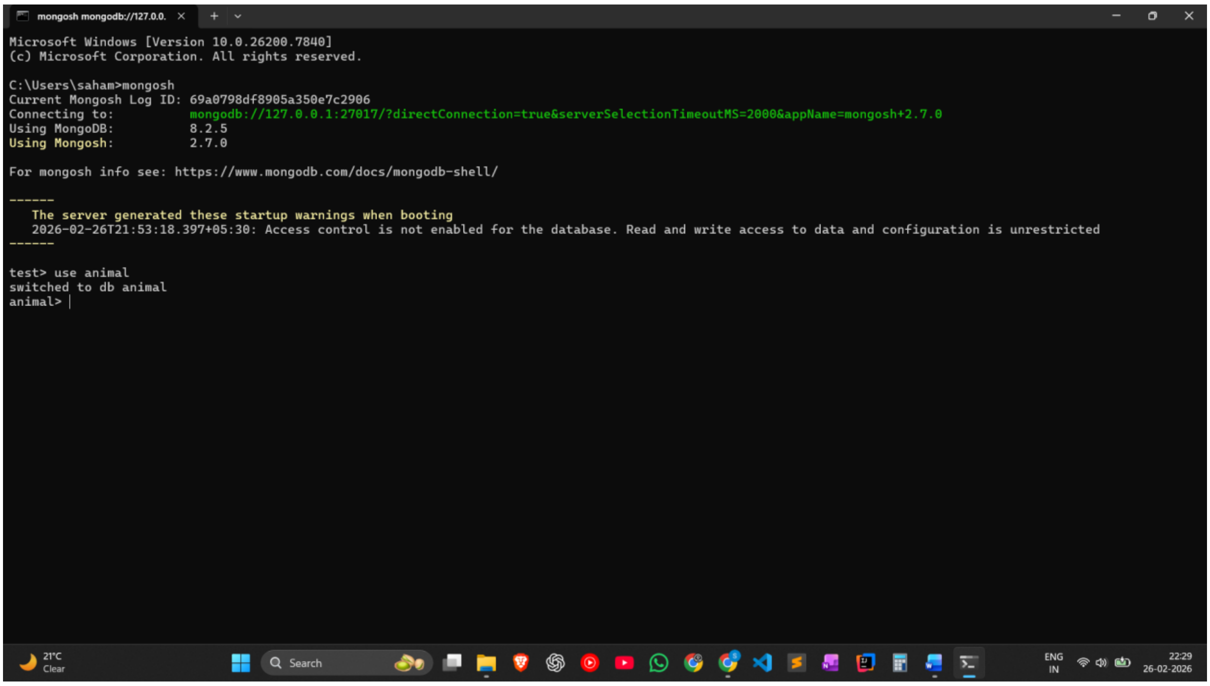
Program 10: Write a MongoDB query to display vehicles name, category and manufacturer details whose rating is more than 5.



```
test> db.two_wheelers.find(
|   { performance: { $gt: 5 } },
|   { bike_name: 1, category: 1, manufacturer: 1, _id: 0 }
| )
|
| db.four_wheelers.find(
|   { performance: { $gt: 5 } },
|   { vehicle_name: 1, category: 1, manufacturer: 1, _id: 0 }
| )
[
  { vehicle_name: 'Swift', category: 'car', manufacturer: 'Maruti' },
  { vehicle_name: 'Innova', category: 'car', manufacturer: 'Toyota' },
  {
    vehicle_name: 'Ashok Leyland Bus',
    category: 'bus',
    manufacturer: 'Ashok Leyland'
  },
  {
    vehicle_name: 'Eicher Truck',
    category: 'heavy truck',
    manufacturer: 'Eicher'
  },
  {
    vehicle_name: 'Tata Ace',
    category: 'mini truck',
    manufacturer: 'Tata'
  }
```

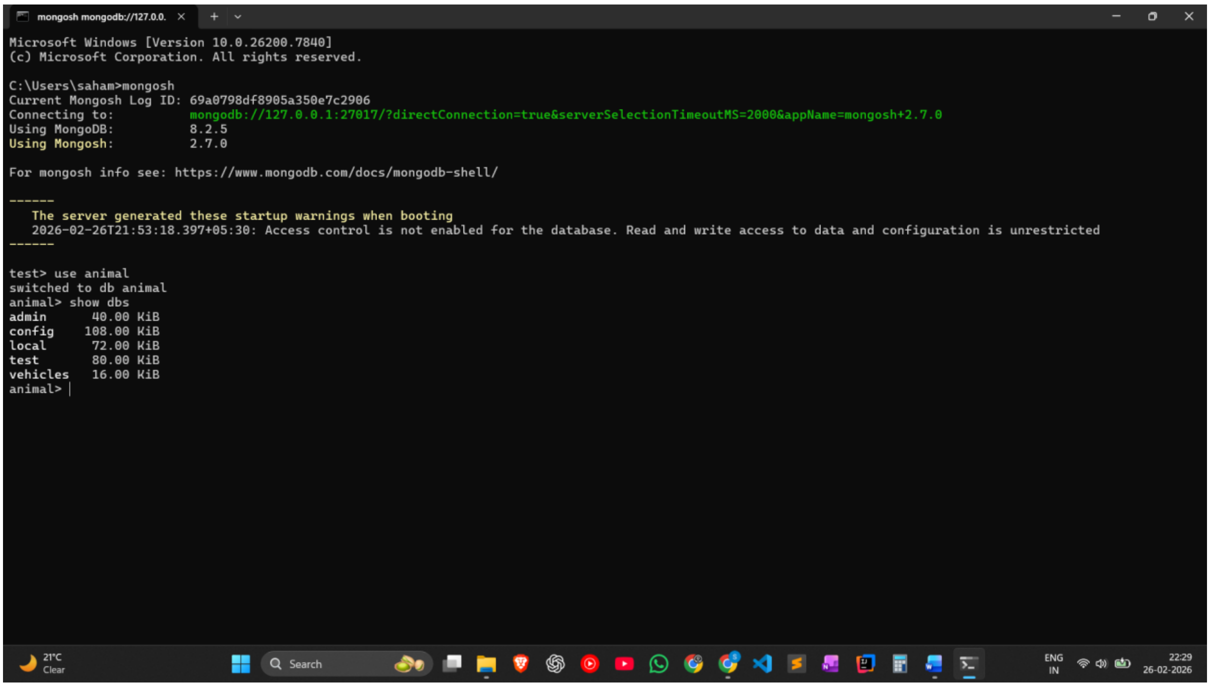
DATABASE – 2: ZOO

Program 1: Create a database called 'animal' and write a MongoDB query to select database as 'animal'.



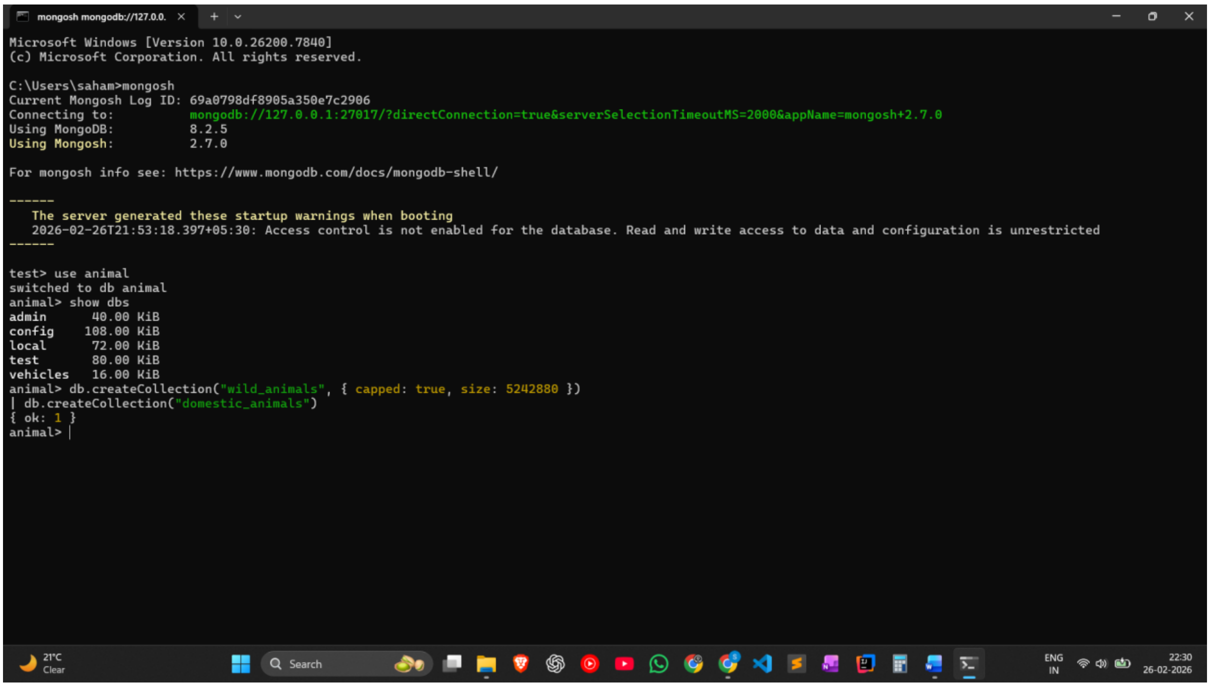
```
Microsoft Windows [Version 10.0.26200.7840]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saham>mongosh
Current Mongosh Log ID: 69a0798df8905a350e7c2906
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.7.0
Using MongoDB:          8.2.5
Using Mongosh:          2.7.0

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

------
   The server generated these startup warnings when booting
   2026-02-26T21:53:18.397+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use animal
switched to db animal
animal> |
```

Program 2: Write a MongoDB query to display all the databases.



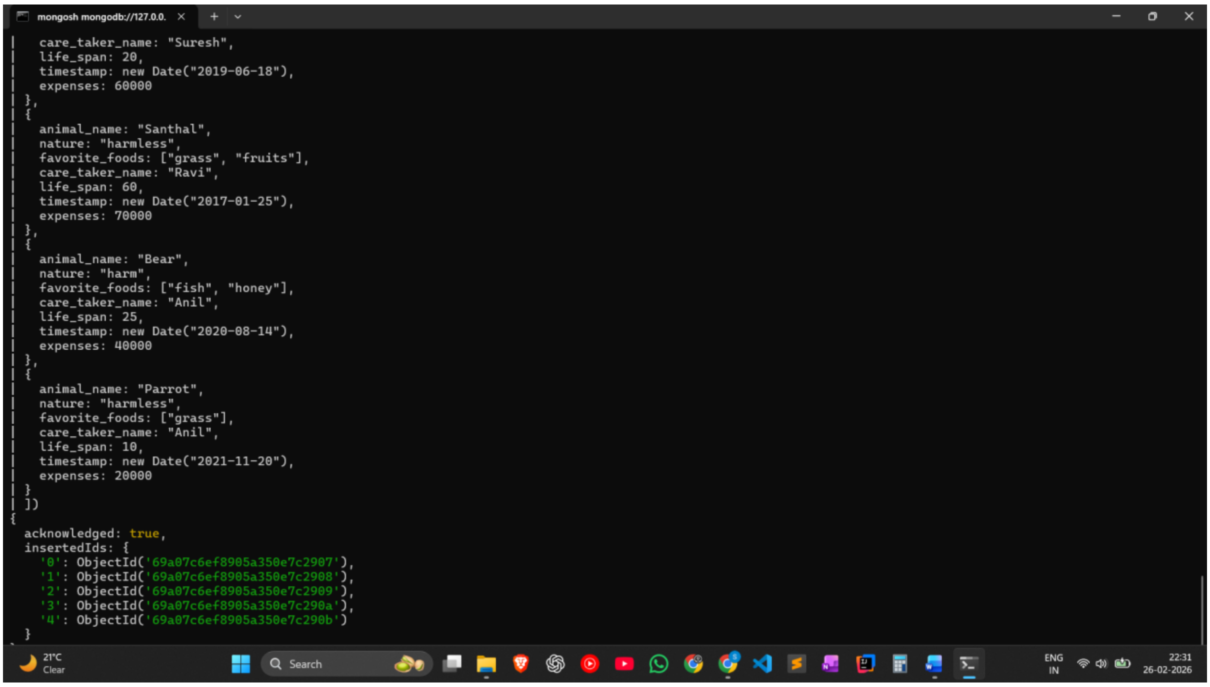
```
Microsoft Windows [Version 10.0.26200.7840]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saham>mongosh
Current Mongosh Log ID: 69a0798df8905a350e7c2906
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.7.0
Using MongoDB:          8.2.5
Using Mongosh:          2.7.0

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

------
   The server generated these startup warnings when booting
   2026-02-26T21:53:18.397+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use animal
switched to db animal
animal> show dbs
admin      40.00 KiB
config    108.00 KiB
local      72.00 KiB
test       80.00 KiB
vehicles   16.00 KiB
animal> |
```

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



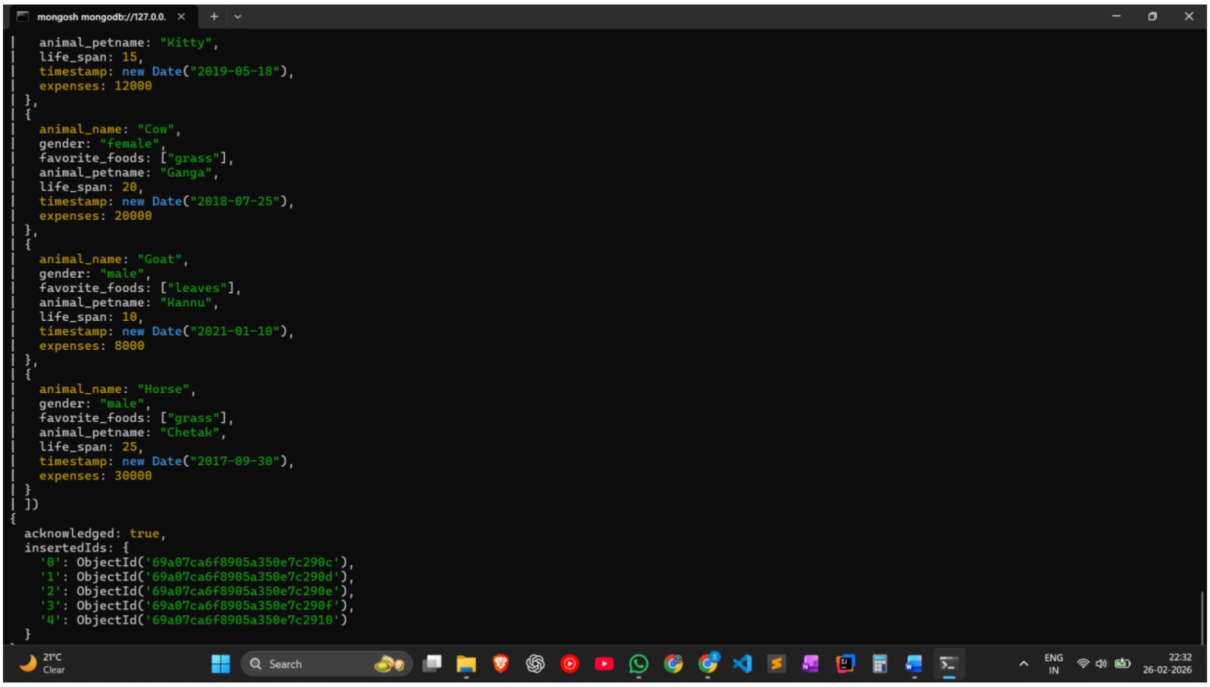
```
Microsoft Windows [Version 10.0.26200.7840]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saham>mongosh
Current Mongosh Log ID: 69a0798df8905a350e7c2906
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.7.0
Using MongoDB:          8.2.5
Using Mongosh:          2.7.0

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

------
   The server generated these startup warnings when booting
   2026-02-26T21:53:18.397+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use animal
switched to db animal
animal> show dbs
admin      40.00 KiB
config    108.00 KiB
local      72.00 KiB
test       80.00 KiB
vehicles   16.00 KiB
animal> db.createCollection("wild_animals", { capped: true, size: 5242880 })
| db.createCollection("domestic_animals")
{ ok: 1 }
animal> |
```

Program 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.



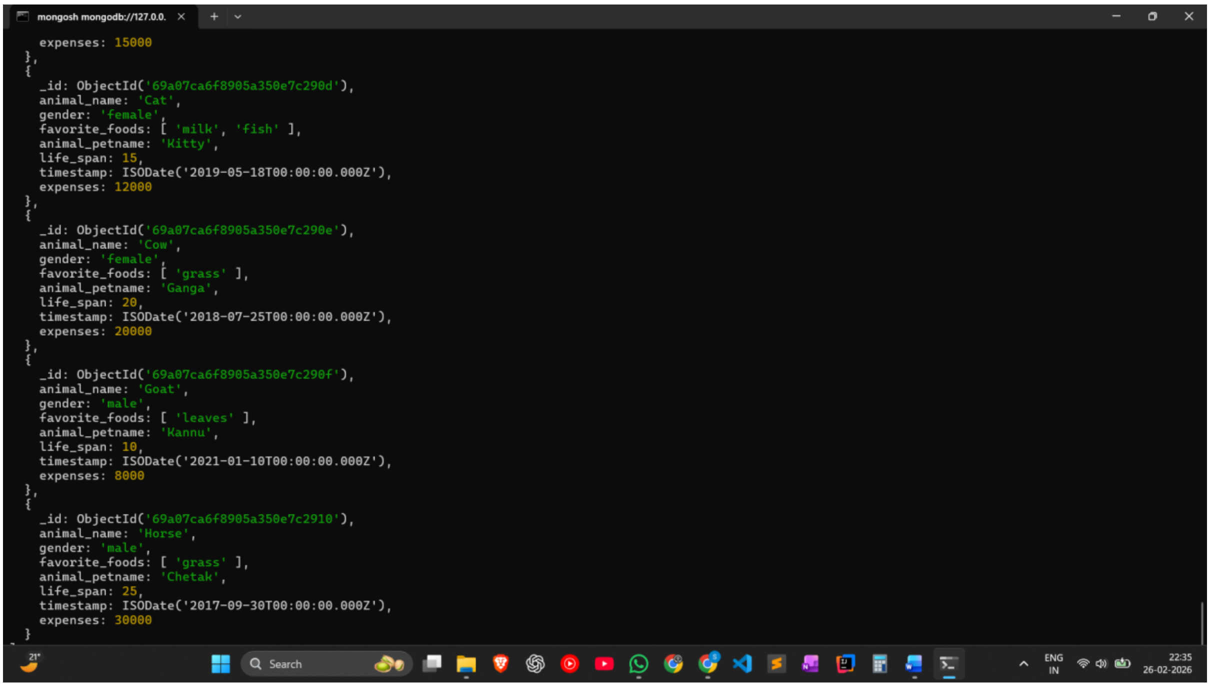
```
|   care_taker_name: "Suresh",
|   life_span: 20,
|   timestamp: new Date("2019-06-18"),
|   expenses: 60000
| },
| {
|   animal_name: "Santhal",
|   nature: "harmless",
|   favorite_foods: ["grass", "fruits"],
|   care_taker_name: "Ravi",
|   life_span: 60,
|   timestamp: new Date("2017-01-25"),
|   expenses: 70000
| },
| {
|   animal_name: "Bear",
|   nature: "harm",
|   favorite_foods: ["fish", "honey"],
|   care_taker_name: "Anil",
|   life_span: 25,
|   timestamp: new Date("2020-08-14"),
|   expenses: 40000
| },
| {
|   animal_name: "Parrot",
|   nature: "harmless",
|   favorite_foods: ["grass"],
|   care_taker_name: "Anil",
|   life_span: 10,
|   timestamp: new Date("2021-11-20"),
|   expenses: 20000
| }
| ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('69a07c6ef8905a350e7c2907'),
    '1': ObjectId('69a07c6ef8905a350e7c2908'),
    '2': ObjectId('69a07c6ef8905a350e7c2909'),
    '3': ObjectId('69a07c6ef8905a350e7c290a'),
    '4': ObjectId('69a07c6ef8905a350e7c290b')
  }
```

Program 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_petname, life span (in years), timestamp (when the animal registered at the Zoo) and expenses.



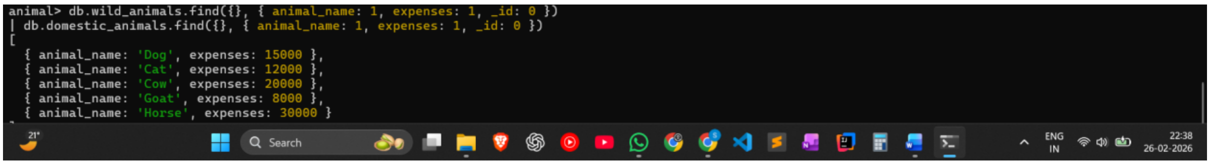
```
|   animal_petname: "Kitty",
|   life_span: 15,
|   timestamp: new Date("2019-05-18"),
|   expenses: 12000
| },
| {
|   animal_name: "Cow",
|   gender: "female",
|   favorite_foods: ["grass"],
|   animal_petname: "Ganga",
|   life_span: 20,
|   timestamp: new Date("2018-07-25"),
|   expenses: 20000
| },
| {
|   animal_name: "Goat",
|   gender: "male",
|   favorite_foods: ["leaves"],
|   animal_petname: "Kannu",
|   life_span: 10,
|   timestamp: new Date("2021-01-10"),
|   expenses: 8000
| },
| {
|   animal_name: "Horse",
|   gender: "male",
|   favorite_foods: ["grass"],
|   animal_petname: "Chetak",
|   life_span: 25,
|   timestamp: new Date("2017-09-30"),
|   expenses: 30000
| }
| ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('69a07ca6f8905a350e7c290c'),
    '1': ObjectId('69a07ca6f8905a350e7c290d'),
    '2': ObjectId('69a07ca6f8905a350e7c290e'),
    '3': ObjectId('69a07ca6f8905a350e7c290f'),
    '4': ObjectId('69a07ca6f8905a350e7c2910')
  }
```

Program 6: Write a MongoDB query to display all documents available in wild_animals and domastic_animals.



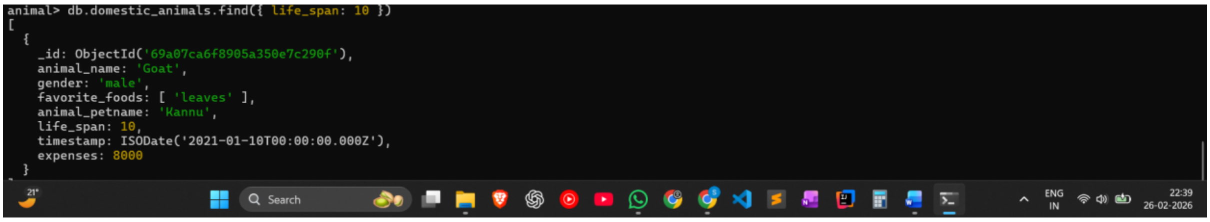
```
    expenses: 15000
  },
  {
    _id: ObjectId('69a07ca6f8905a350e7c290d'),
    animal_name: 'Cat',
    gender: 'female',
    favorite_foods: [ 'milk', 'fish' ],
    animal_petname: 'Kitty',
    life_span: 15,
    timestamp: ISODate('2019-05-18T00:00:00.000Z'),
    expenses: 12000
  },
  {
    _id: ObjectId('69a07ca6f8905a350e7c290e'),
    animal_name: 'Cow',
    gender: 'female',
    favorite_foods: [ 'grass' ],
    animal_petname: 'Ganga',
    life_span: 20,
    timestamp: ISODate('2018-07-25T00:00:00.000Z'),
    expenses: 20000
  },
  {
    _id: ObjectId('69a07ca6f8905a350e7c290f'),
    animal_name: 'Goat',
    gender: 'male',
    favorite_foods: [ 'leaves' ],
    animal_petname: 'Kannu',
    life_span: 10,
    timestamp: ISODate('2021-01-10T00:00:00.000Z'),
    expenses: 8000
  },
  {
    _id: ObjectId('69a07ca6f8905a350e7c2910'),
    animal_name: 'Horse',
    gender: 'male',
    favorite_foods: [ 'grass' ],
    animal_petname: 'Chetak',
    life_span: 25,
    timestamp: ISODate('2017-09-30T00:00:00.000Z'),
    expenses: 30000
  }
```

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



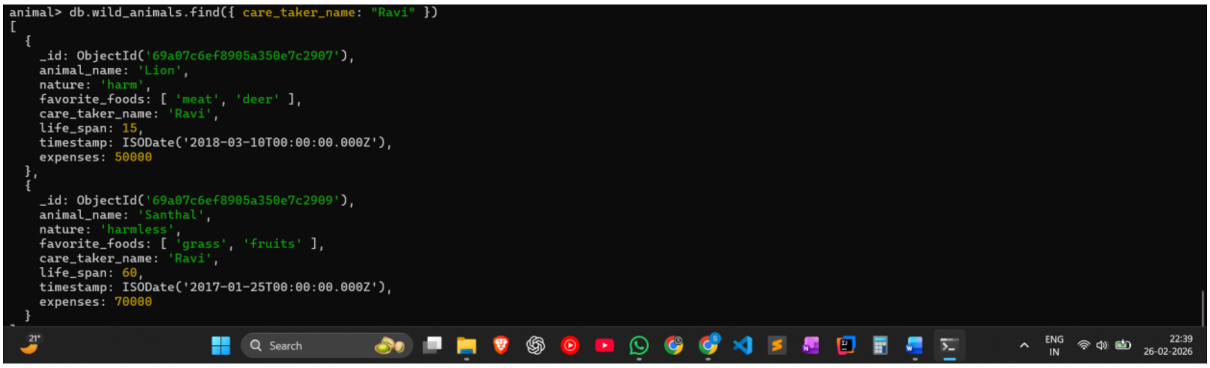
```
animal> db.wild_animals.find({}, { animal_name: 1, expenses: 1, _id: 0 })
| db.domestic_animals.find({}, { animal_name: 1, expenses: 1, _id: 0 })
[
  { animal_name: 'Dog', expenses: 15000 },
  { animal_name: 'Cat', expenses: 12000 },
  { animal_name: 'Cow', expenses: 20000 },
  { animal_name: 'Goat', expenses: 8000 },
  { animal_name: 'Horse', expenses: 30000 }
```

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



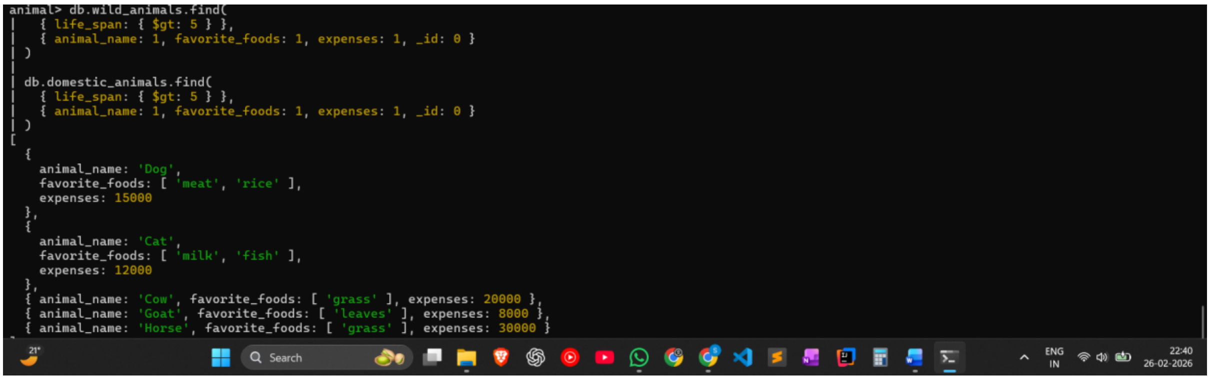
```
animal> db.domestic_animals.find({ life_span: 10 })
[
  {
    _id: ObjectId('69a07ca6f8905a350e7c290f'),
    animal_name: 'Goat',
    gender: 'male',
    favorite_foods: [ 'leaves' ],
    animal_petname: 'Kannu',
    life_span: 10,
    timestamp: ISODate('2021-01-10T00:00:00.000Z'),
    expenses: 8000
  }
```

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



```
animal> db.wild_animals.find({ care_taker_name: "Ravi" })
[
  {
    _id: ObjectId('69a07c6ef8905a350e7c2907'),
    animal_name: 'Lion',
    nature: 'harm',
    favorite_foods: [ 'meat', 'deer' ],
    care_taker_name: 'Ravi',
    life_span: 15,
    timestamp: ISODate('2018-03-10T00:00:00.000Z'),
    expenses: 50000
  },
  {
    _id: ObjectId('69a07c6ef8905a350e7c2909'),
    animal_name: 'Santhal',
    nature: 'harmless',
    favorite_foods: [ 'grass', 'fruits' ],
    care_taker_name: 'Ravi',
    life_span: 60,
    timestamp: ISODate('2017-01-25T00:00:00.000Z'),
    expenses: 70000
  }
```

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



```
animal> db.wild_animals.find(
|   { life_span: { $gt: 5 } },
|   { animal_name: 1, favorite_foods: 1, expenses: 1, _id: 0 }
| )
|
| db.domestic_animals.find(
|   { life_span: { $gt: 5 } },
|   { animal_name: 1, favorite_foods: 1, expenses: 1, _id: 0 }
| )
[
  {
    animal_name: 'Dog',
    favorite_foods: [ 'meat', 'rice' ],
    expenses: 15000
  },
  {
    animal_name: 'Cat',
    favorite_foods: [ 'milk', 'fish' ],
    expenses: 12000
  },
  { animal_name: 'Cow', favorite_foods: [ 'grass' ], expenses: 20000 },
  { animal_name: 'Goat', favorite_foods: [ 'leaves' ], expenses: 8000 },
  { animal_name: 'Horse', favorite_foods: [ 'grass' ], expenses: 30000 }
```