BIG DATA

SECTION D

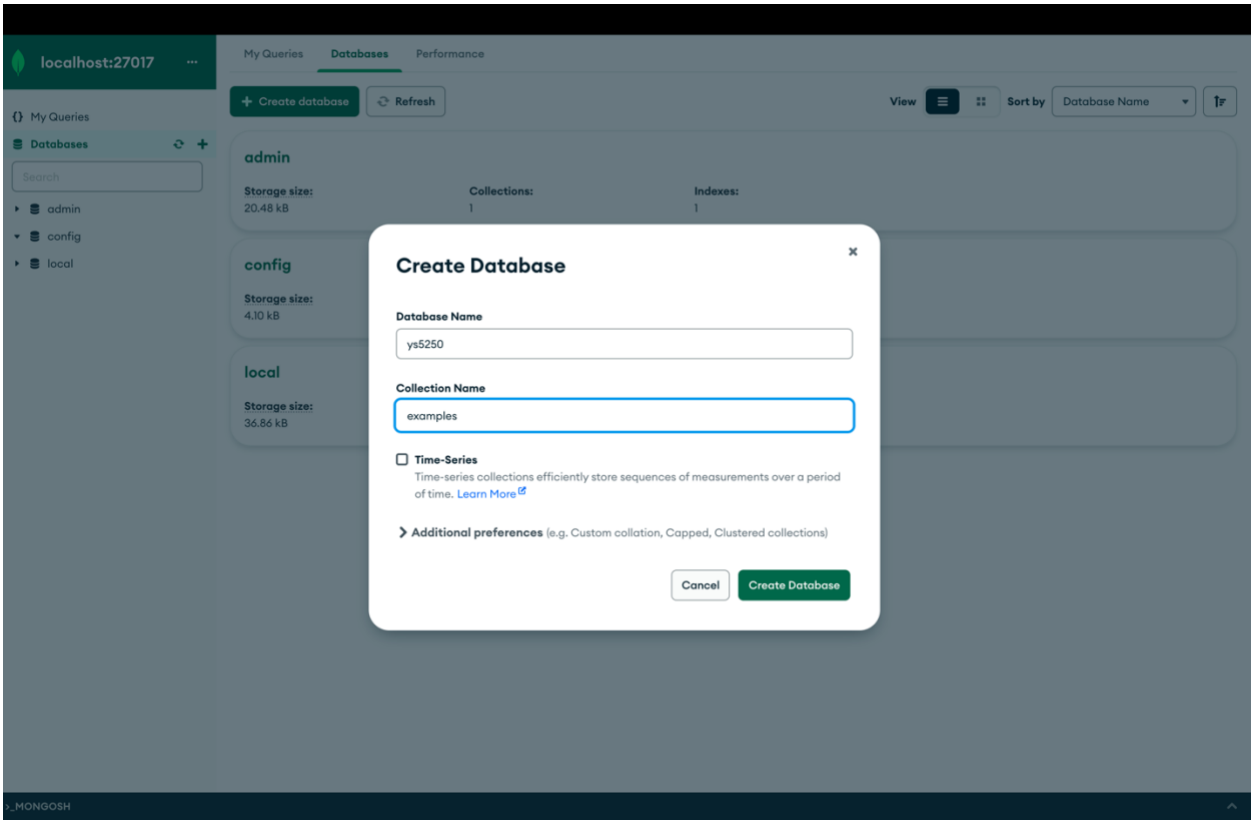
Fall’2023

Yogya Sharma

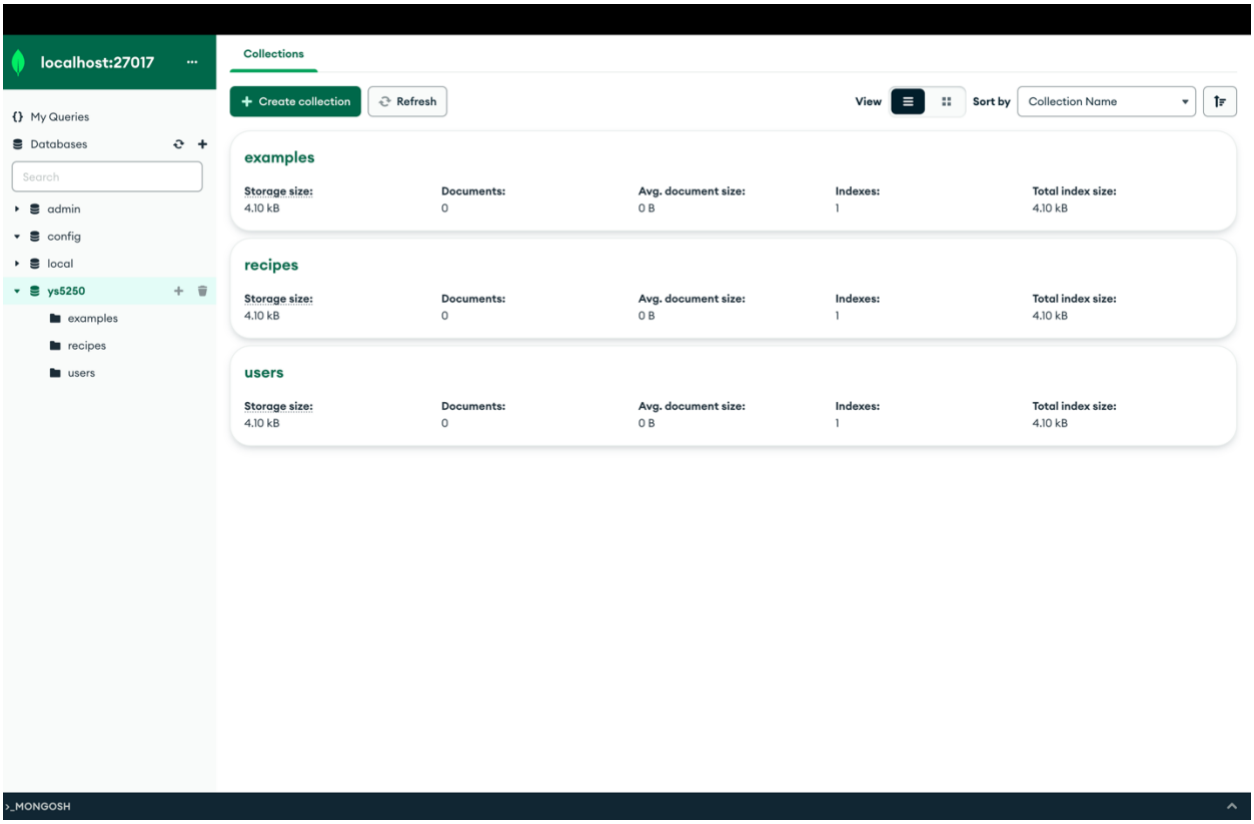
ys5250

Practice Assignment : MongoDB Compass

1. Create a database:



2. Create users and recipes collections:



3. Use db:

```
> use ys5250;
< switched to db ys5250
ys5250>
```

4. Create a new collection:

```
> db.createCollection('myCollection')
< { ok: 1 }
> show collections
< examples
  myCollection
  recipes
  users
ys5250>
```

5. Dropping a collection:

```
> db.myCollection.drop()
< true
> show collections
< examples
  recipes
  users
ys5250>
```

6. Using the insert command to input into a collection:

```
ys5250 > db.recipes.insert({
          "title": "Maggi Noodles",
          "calories_per_serving": 250,
          "cook_time": 10,
          "desc": "Quick and delicious Maggi Noodles",
          "directions": [
            "Boil water in a pot",
            "Add Maggi noodles and cook for 2 minutes",
            "Add the Maggi masala and stir well",
            "Cook for another 2 minutes until noodles are ready"
          ],
          "ingredients": [
            {
              "name": "Maggi Noodles",
              "quantity": {
                "amount": 1,
```

```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("655532a9dd45980d0a46e8da")
    }
  }
ys5250 >
```

```
> db.recipes.find({ "_id": ObjectId("655532a9dd45980d0a46e8da") })
< {
    _id: ObjectId("655532a9dd45980d0a46e8da"),
    title: 'Maggi Noodles',
    calories_per_serving: 250,
    cook_time: 10,
    desc: 'Quick and delicious Maggi Noodles',
    directions: [
      'Boil water in a pot',
      'Add Maggi noodles and cook for 2 minutes',
      'Add the Maggi masala and stir well',
      'Cook for another 2 minutes until noodles are ready'
    ],
    ingredients: [
      {
        name: 'Maggi Noodles',
```

7. Find command:

```
> db.examples.find({
    "prep_time": 10
  }).pretty()
< {
    _id: ObjectId("5e5e9c470d33e9e8e3891b35"),
    title: 'Tacos',
    calories_per_serving: 210,
    cook_time: 20,
    desc: 'Classic Mexican tacos',
    directions: [
      'Brown beef',
      'Add taco seasoning and water, mix',
      'Bring to boil',
      'Lower heat to simmer 5-10 minutes until desired consistency',
      'Put meat in tacos'
    ],
    ingredients: [
      {
        name: 'ground beef (lean)',
        quantity: {
          amount: 1,
          unit: 'lbs'
        }
      },
      {
        name: 'taco seasoning',
        quantity: {
          amount: 2,
          unit: 'oz'
        }
```

8. Find using In command:

```
> db.examples.find({
    "tags": {
      $in: ["mexican", "quick"]
    }
  }).pretty()
< {
    _id: ObjectId("5e5e9c470d33e9e8e3891b35"),
    title: 'Tacos',
    calories_per_serving: 210,
    cook_time: 20,
    desc: 'Classic Mexican tacos',
    directions: [
      'Brown beef',
      'Add taco seasoning and water, mix',
      'Bring to boil',
      'Lower heat to simmer 5-10 minutes until desired consistency',
      'Put meat in tacos'
    ],
    ingredients: [
      {
        name: 'ground beef (lean)',
        quantity: {
          amount: 1,
          unit: 'lbs'
        }
      },
      {
        name: 'taco seasoning',
        quantity: {
          amount: 2,
```

9. Find command using equal to:

```
> db.examples.find({
    "prep_time": 10
  }).pretty()
< {
    _id: ObjectId("5e5e9c470d33e9e8e3891b35"),
    title: 'Tacos',
    calories_per_serving: 210,
    cook_time: 20,
    desc: 'Classic Mexican tacos',
    directions: [
      'Brown beef',
      'Add taco seasoning and water, mix',
      'Bring to boil',
      'Lower heat to simmer 5-10 minutes until desired consistency',
      'Put meat in tacos'
    ],
    ingredients: [
      {
        name: 'ground beef (lean)',
        quantity: {
          amount: 1,
          unit: 'lbs'
        }
      },
      {
        name: 'taco seasoning',
        quantity: {
          amount: 2,
          unit: 'oz'
        }
```

10. Find using greater than operator:

```
> db.examples.find({
    "rating_avg": {
      $gt: 2
    }
  }).pretty()
< {
    _id: ObjectId("5e5e9c470d33e9e8e3891b35"),
    title: 'Tacos',
    calories_per_serving: 210,
    cook_time: 20,
    desc: 'Classic Mexican tacos',
    directions: [
      'Brown beef',
      'Add taco seasoning and water, mix',
      'Bring to boil',
      'Lower heat to simmer 5-10 minutes until desired consistency',
      'Put meat in tacos'
    ],
    ingredients: [
      {
        name: 'ground beef (lean)',
        quantity: {
          amount: 1,
          unit: 'lbs'
        }
      },
      {
        name: 'taco seasoning',
        quantity: {
          amount: 2,
```

11. Update:

```
> db.recipes.update({ "_id": ObjectId("5e5e9c470d33e9e8e3891b35") }, { $set: { "cook_time": 30 } })
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

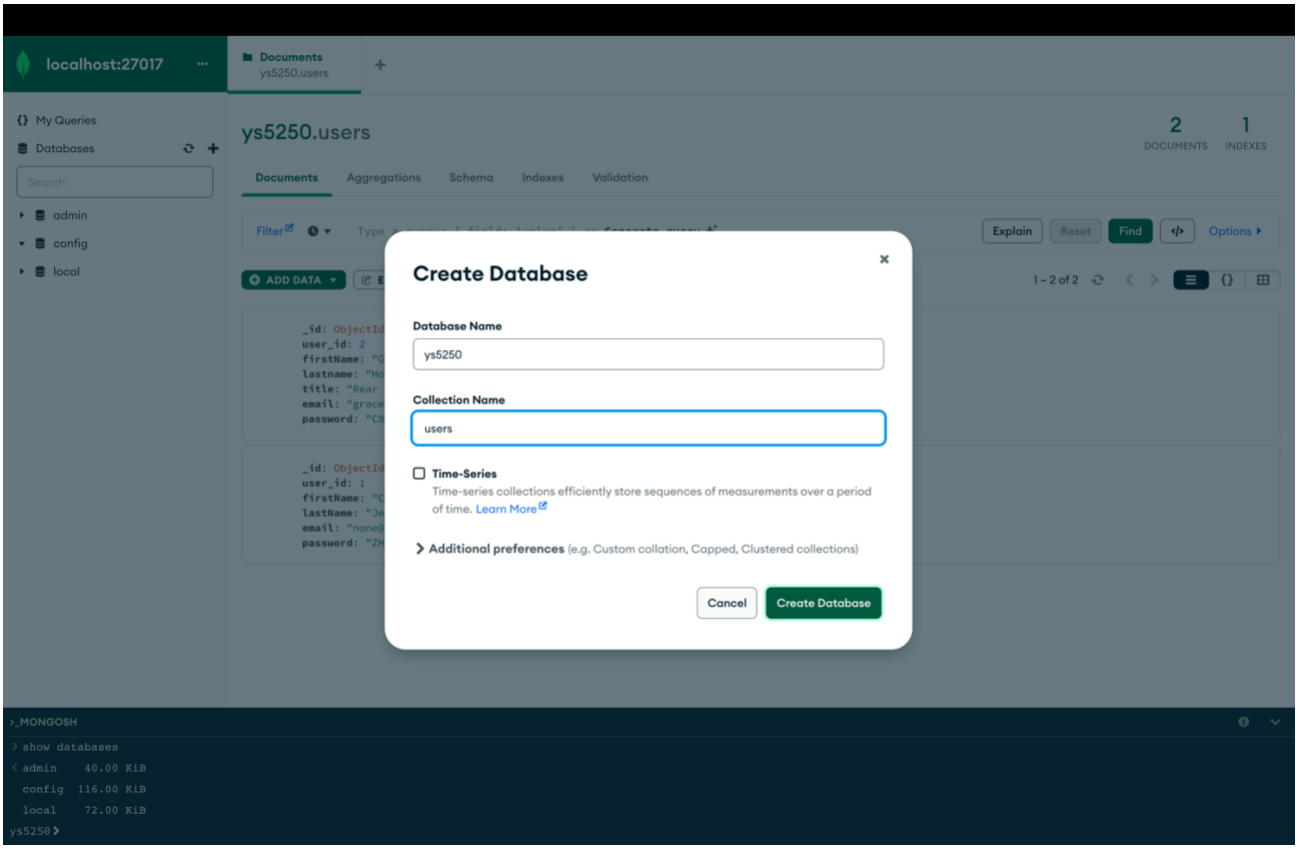
12. Adding element to an array:

```
> db.recipes.update({ "_id": ObjectId("5e5e9c470d33e9e8e3891b35") }, { $push: { "tags": "spicy" } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

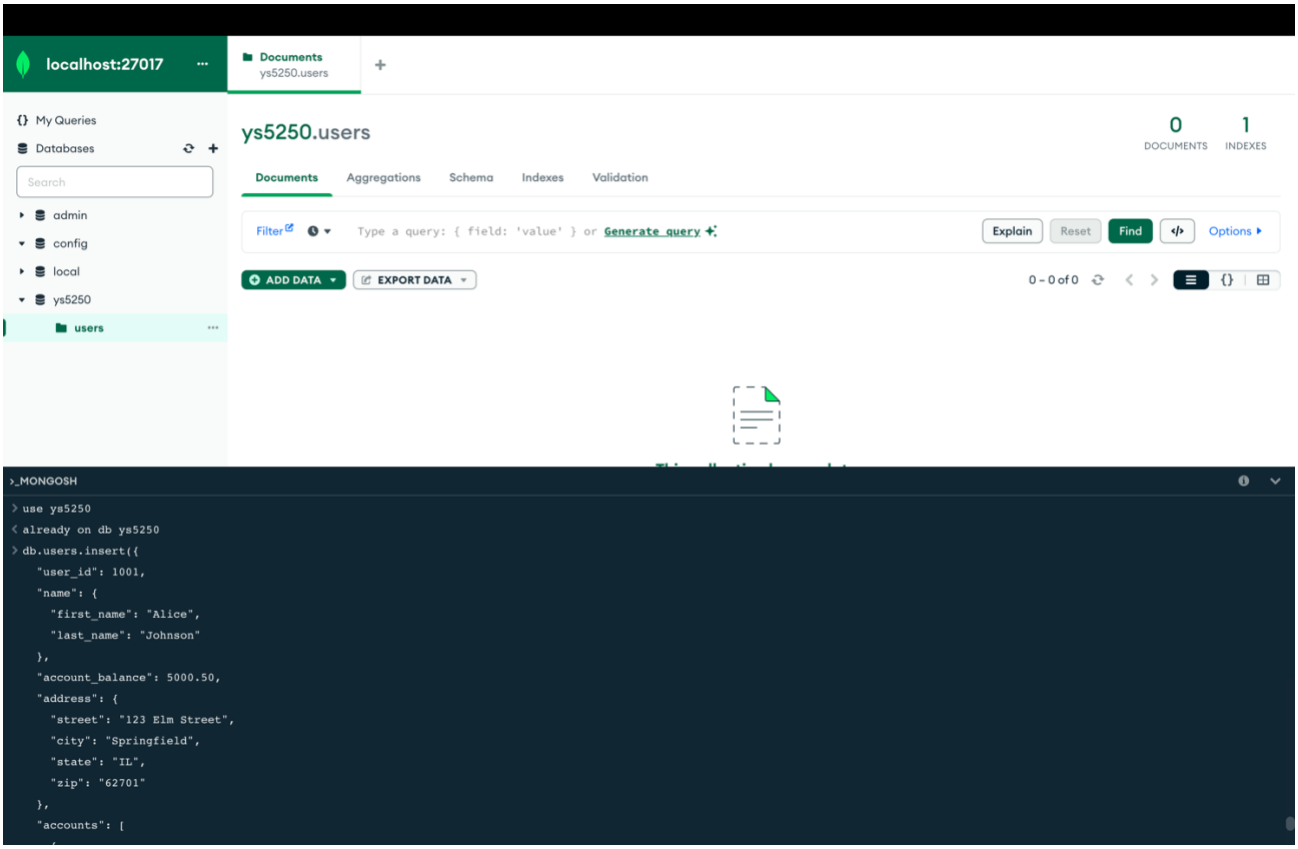
13. Drop the database

```
> db.dropDatabase()
< { ok: 1, dropped: 'ys5250' }
> show databases
< admin    40.00 KiB
  config  116.00 KiB
  local    72.00 KiB
ys5250 >
```

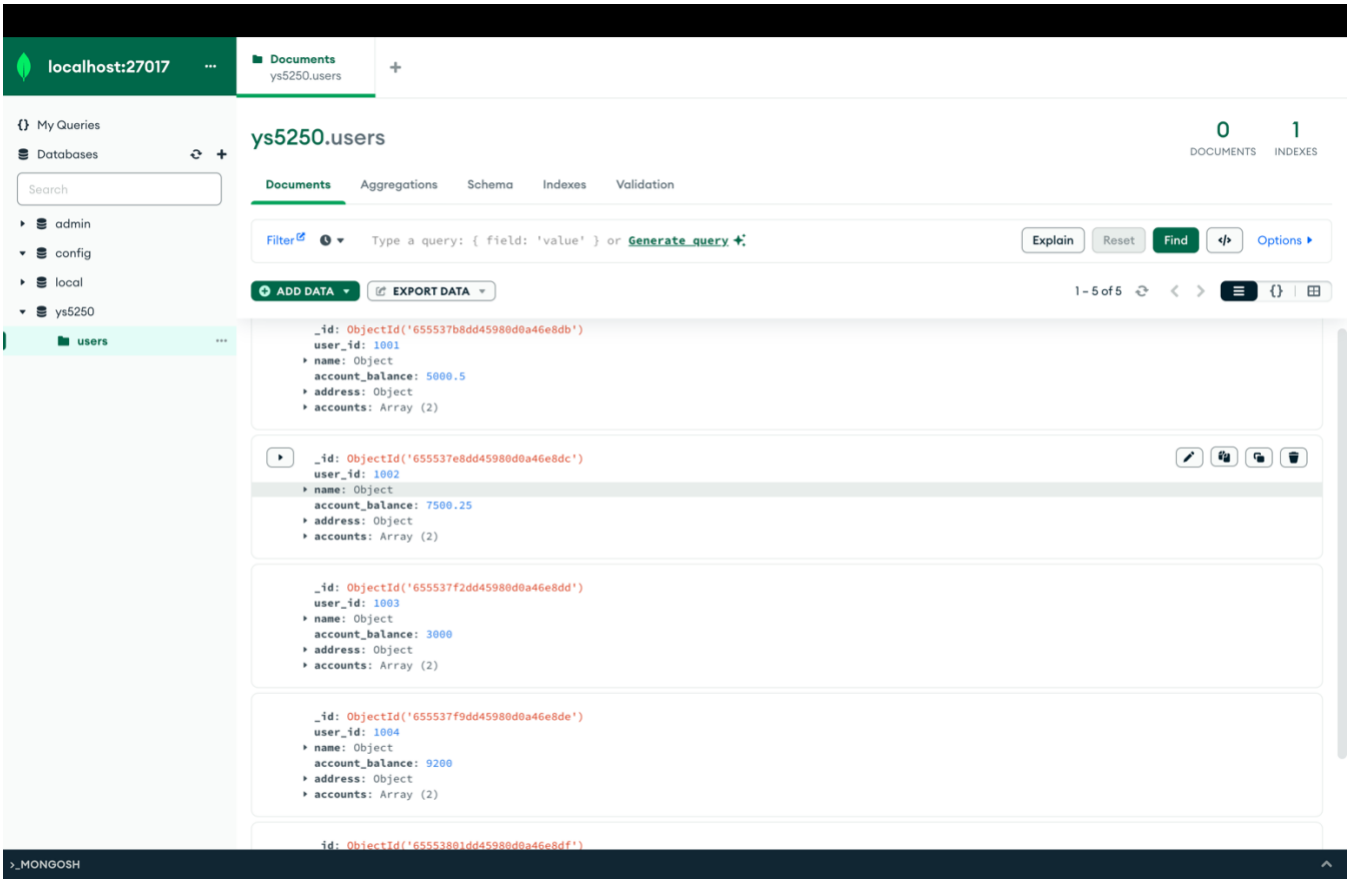
1. Create a database and collection users:



2. Insert users:



Similarly for each user, so here is the updated collection:



3. Find Users by Last Name:

```
> db.users.find(
    { "name.last_name": "Smith" },
    { "name.first_name": 1, "account_balance": 1, "_id": 0 }
  ).pretty()
< {
    name: {
      first_name: 'Bob'
    },
    account_balance: 7500.25
  }
ys5250 >
```

4. Find Users with a Specific Account Type:

```
>_MONGOSH
> db.users.find(
    { "accounts.account_type": "Savings" },
    { "name": 1, "accounts.account_number": 1, "_id": 0 }
  ).pretty()
< {
    name: {
      first_name: 'Alice',
      last_name: 'Johnson'
    },
    accounts: [
      {
        account_number: 'Savings-123'
      },
      {
        account_number: 'Checking-456'
      }
    ]
  }
  {
    name: {
      first_name: 'Bob',
      last_name: 'Smith'
    },
    accounts: [
      {
        account_number: 'Savings-789'
      },
      {
        account_number: 'Checking-987'
      }
    ]
  }
  {
```

5. Find Users with Account Balances Greater Than $5000:

```
>_MONGOSH
  }
> db.users.find(
    { "account_balance": { $gt: 5000 } },
    { "name": 1, "account_balance": 1, "_id": 0 }
  ).pretty()
< {
    name: {
      first_name: 'Alice',
      last_name: 'Johnson'
    },
    account_balance: 5000.5
  }
  {
    name: {
      first_name: 'Bob',
      last_name: 'Smith'
    },
    account_balance: 7500.25
  }
  {
    name: {
      first_name: 'David',
      last_name: 'Lee'
    },
    account_balance: 9200
  }
  {
    name: {
      first_name: 'Ella',
      last_name: 'Wong'
    },
    account_balance: 6500.75
  }
```

6. Find Users in a Specific State:

```
> db.users.find(
    { "address.state": "CA" },
    { "name": 1, "address": 1, "account_balance": 1, "_id": 0 }
  ).pretty()
< {
    name: {
      first_name: 'Charlie',
      last_name: 'Brown'
    },
    account_balance: 3000,
    address: {
      street: '789 Maple Lane',
      city: 'Los Angeles',
      state: 'CA',
      zip: '90001'
    }
  }
  {
    name: {
      first_name: 'Ella',
      last_name: 'Wong'
    },
    account_balance: 6500.75,
    address: {
      street: '789 Cedar Road',
      city: 'San Francisco',
      state: 'CA',
      zip: '94101'
    }
  }
ys5250 >
```

7. Find Users with More Than One Bank Account:

```
>_MONGOSH
> db.users.aggregate([
    {
      $project: {
        "name": 1,
        "num_of_accounts": { $size: "$accounts" },
        "_id": 0
      }
    },
    {
      $match: { "num_of_accounts": { $gt: 1 } }
    }
  ]).pretty()
< {
    name: {
      first_name: 'Alice',
      last_name: 'Johnson'
    },
    num_of_accounts: 2
  }
  {
    name: {
      first_name: 'Bob',
      last_name: 'Smith'
    },
    num_of_accounts: 2
  }
  {
    name: {
      first_name: 'Charlie',
      last_name: 'Brown'
    },
    num_of_accounts: 2
  }
```

8. Find Users with a Specific Account Number:

```
>_MONGOSH
> db.users.find(
    { "accounts.account_number": "Savings-789" }
  ).pretty()
< {
    _id: ObjectId("655537e8dd45980d0a46e8dc"),
    user_id: 1002,
    name: {
      first_name: 'Bob',
      last_name: 'Smith'
    },
    account_balance: 7500.25,
    address: {
      street: '456 Oak Avenue',
      city: 'New York',
      state: 'NY',
      zip: '10001'
    },
    accounts: [
      {
        account_number: 'Savings-789',
        account_type: 'Savings',
        balance: 5500.5
      },
      {
        account_number: 'Checking-987',
        account_type: 'Checking',
        balance: 2000.75
      }
    ]
  }
  {
    _id: ObjectId("65553801dd45980d0a46e8df"),
    user_id: 1005,
```

9. Find Users with a Savings Account Balance Greater Than $6000:

```
>_MONGOSH
> db.users.find(
    {
      "accounts": {
        $elemMatch: {
          "account_type": "Savings",
          "balance": { $gt: 6000 }
        }
      }
    },
    {
      "name": 1,
      "accounts.account_number": 1,
      "accounts.balance": 1,
      "_id": 0
    }
  ).pretty()
< {
    name: {
      first_name: 'David',
      last_name: 'Lee'
    },
    accounts: [
      {
        account_number: 'Savings-246',
        balance: 7200.5
      },
      {
        account_number: 'Checking-864',
        balance: 2000.5
      }
    ]
  }
ys5250>
```

10. Find Users with Multiple Checking Accounts:

```
>_MONGOSH
> db.users.find(
    {
      "accounts": {
        $all: [
          { $elemMatch: { "account_type": "Checking" } },
          { $elemMatch: { "account_type": "Checking" } }
        ]
      }
    },
    {
      "name": 1,
      "accounts.account_type": 1,
      "accounts.account_number": 1,
      "_id": 0
    }
  ).pretty()
< {
    name: {
      first_name: 'Alice',
      last_name: 'Johnson'
    },
    accounts: [
      {
        account_number: 'Savings-123',
        account_type: 'Savings'
      },
      {
        account_number: 'Checking-456',
        account_type: 'Checking'
      }
    ]
  }
  {
```

11. Find Users with a Specific Tag:

```
> db.users.aggregate([
    {
      $match: {
        "accounts.tag": { $exists: true }
      }
    },
    {
      $project: {
        "name": 1,
        "matching_accounts": {
          $filter: {
            input: "$accounts",
            as: "account",
            cond: { $eq: ["$$account.tag", "some_tag"] }
          }
        },
        "_id": 0
      }
    },
    {
      $match: {
        "matching_accounts": { $gt: [] }
      }
    }
  ]).pretty()
<
ys5250 >
```

12. Find Users by User ID:

```
>_MONGOSH
> db.users.find({}).sort({ "user_id": 1 }).pretty()
< {
    _id: ObjectId("655537b8dd45980d0a46e8db"),
    user_id: 1001,
    name: {
      first_name: 'Alice',
      last_name: 'Johnson'
    },
    account_balance: 5000.5,
    address: {
      street: '123 Elm Street',
      city: 'Springfield',
      state: 'IL',
      zip: '62701'
    },
    accounts: [
      {
        account_number: 'Savings-123',
        account_type: 'Savings',
        balance: 3500.75
      },
      {
        account_number: 'Checking-456',
        account_type: 'Checking',
        balance: 1500.75
      }
    ]
  }
  {
    _id: ObjectId("655537e8dd45980d0a46e8dc"),
    user_id: 1002,
    name: {
      first_name: 'Bob',
```

13. Find users with a specific user id:

```
> db.users.find({"user_id":1005}).pretty()
< {
    _id: ObjectId("65553801dd45980d0a46e8df"),
    user_id: 1005,
    name: {
      first_name: 'Ella',
      last_name: 'Wong'
    },
    account_balance: 6500.75,
    address: {
      street: '789 Cedar Road',
      city: 'San Francisco',
      state: 'CA',
      zip: '94101'
    },
    accounts: [
      {
```

14. Find Users and Skip the First Result:

```
> db.users.find({}).skip(1).pretty()
< {
    _id: ObjectId("655537e8dd45980d0a46e8dc"),
    user_id: 1002,
    name: {
      first_name: 'Bob',
      last_name: 'Smith'
    },
    account_balance: 7500.25,
    address: {
      street: '456 Oak Avenue',
      city: 'New York',
      state: 'NY',
      zip: '10001'
    },
    accounts: [
      {
```

15. Find Users and Limit the Results:

```
> db.users.find({}).limit(2).pretty()
< {
    _id: ObjectId("655537b8dd45980d0a46e8db"),
    user_id: 1001,
    name: {
      first_name: 'Alice',
      last_name: 'Johnson'
    },
    account_balance: 5000.5,
    address: {
      street: '123 Elm Street',
      city: 'Springfield',
      state: 'IL',
      zip: '62701'
    },
    accounts: [
      {
        account_number: 'Savings-123',
        account_type: 'Savings',
        balance: 3500.75
      },
      {
        account_number: 'Checking-456',
        account_type: 'Checking',
        balance: 1500.75
      }
    ]
  }
  {
    _id: ObjectId("655537e8dd45980d0a46e8dc"),
    user_id: 1002,
    name: {
```

16. Create an Index on User ID and drop it:

```
> db.users.createIndex({ "user_id": 1 })
< user_id_1
> db.users.dropIndex({ "user_id": 1 })
< { nIndexesWas: 2, ok: 1 }
```

Learning:

In this assignment, the focus was on mastering essential database management and query operations. Beginning with data importation, the process involved creating databases and collections, importing documents, and performing real-time queries in MongoDB Compass. The tutorial adeptly covered fundamental commands, such as creating, using, and dropping databases, as well as creating and manipulating collections. It introduced various data types supported by MongoDB and provided practical insights into document insertion, update, and retrieval using specific operators. Furthermore, the documentation delved into complex use cases, illustrating the handling of composite and multivalued attributes, showcasing advanced queries, and emphasizing the importance of indexing for optimizing query performance.

The second section highlighted the utilization of the MongoDB terminal, the >MONGOSH, as a powerful tool for running commands. It explored sample commands for displaying existing databases, creating, and dropping databases, creating, and dropping collections, and querying documents with operators such as $in, $eq, and $gt. The section also elucidated data types, offering a comprehensive understanding of MongoDB's versatile data representation.

In the final section, the documentation transitioned to dealing with complex use cases. It provided a practical example of creating database with a "users" collection, emphasizing the integration of composite and multivalued attributes. The tutorial showcased various queries tailored to complex scenarios, such as finding users based on last names, specific account types, or account balances. It concluded with an exploration of advanced queries involving granular array-level operations and the creation and removal of indexes to enhance query efficiency. Overall, this assignment offers a nuanced understanding of MongoDB Compass to navigate sophisticated database management tasks.