

MongoDB – Querying Movies Database

“Querying Movies Database” using MongoDB is an assignment I did for my Data Collection and Curation course in the Big Data Analytics program at Georgian College. The aim of this project was to showcase my understanding of MongoDB and querying data using a non-relational database. I was provided a txt file with a movie database to import into MongoDB and was expected to perform a set of tasks assigned by my professor to show my competency using MongoDB.

Table of Contents

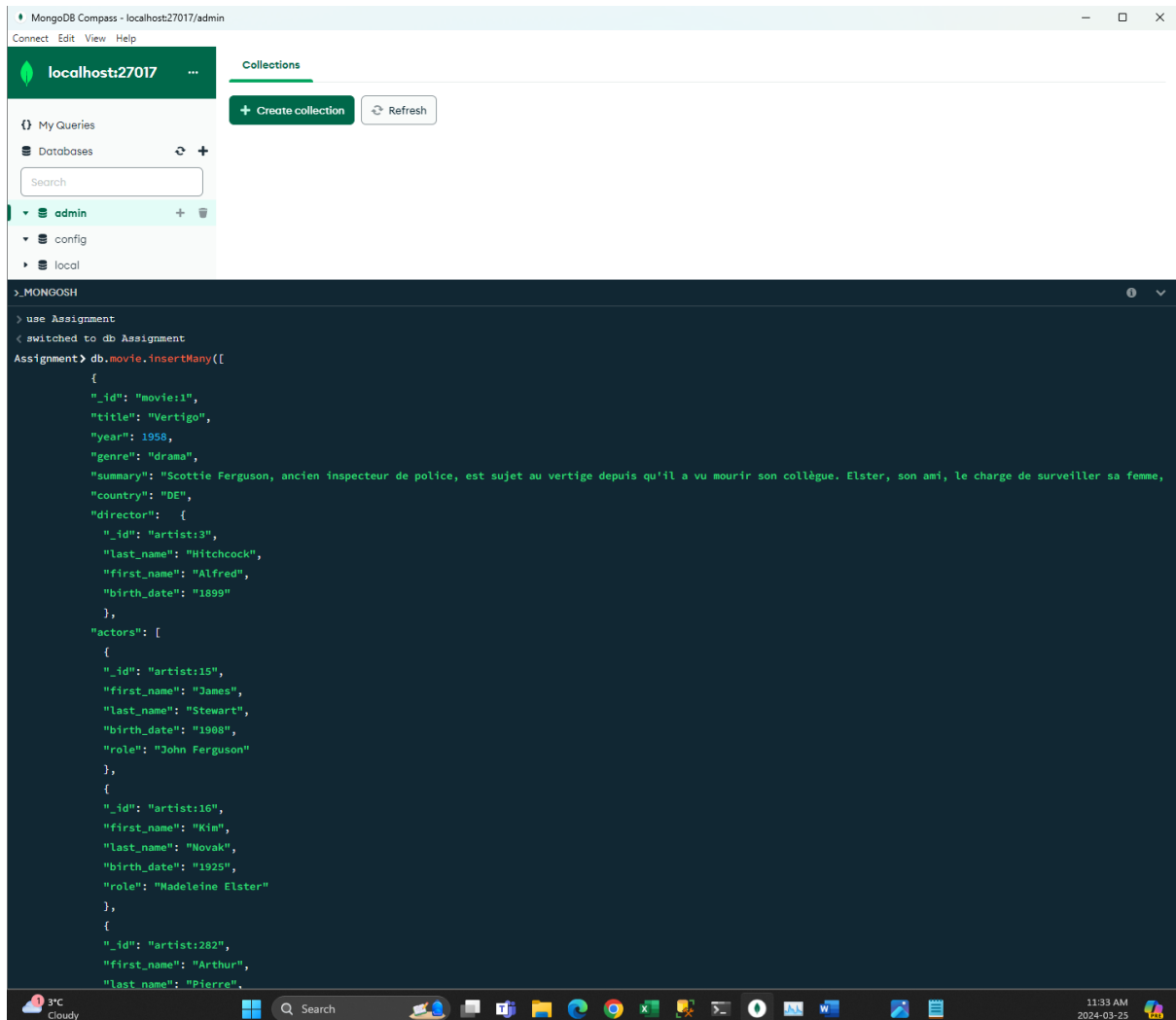
Perform the following Mongo DB Queries	3
A. Write the following script to fetch the document on movie collection.	5
1. Fetch all the document from movie collection.	5
2. Fetch all the document with movies released in the 90s	7
3. Fetch all the document with movies released before the year 2000 or after 2010.	8
4. Fetch all the document with movie "crime" or "drama" genre.	9
B. Write the script to update document on movie collection.	10
1. Update the movie with title "The Titanic" from "Titanic" where _id is "movie:3".	10
2. Update the movie with country "FR" from "USA" where _id is "movie:3".	11
3. Update the movie with director's last_name is "Doe" from "Scott" where _id is "movie:9".	12
4. Update the movies with year 1990 where year is 1994.	13
C. Write the script to Search a string in the document on movie collection.	14
1. Fetch all the movies genre as "drama".	14
2. Fetch all the movies title with "Titanic"	15
3. Fetch all the movies director's first_name as "John".	16
4. Fetch all the movies title start and end with "t"	17
D. Write the script to create the indexes on movie collection.	18
1. Create a single index with multiple attributes (title by ascending order and genre by descending order) on movie.	18
2. Create a wildcard Index on director first_name order by ascending order on movie collection.	20
3. Create a unique index on country attribute by ascending order on movie collection.	21
4. Create a multikey Index on actors first_name order by ascending order on movie collection.	22
E. Write the script to delete document on movie collection.	23
1. Delete the movie with title "Spider-Man".	23
2. Delete the movies which is released in 1990.	24
3. Delete the movie which is from country FR and year 1975.	25
4. Delete the movies with genre is drama.	26

Perform the following Mongo DB Queries

Create an Assignment database.

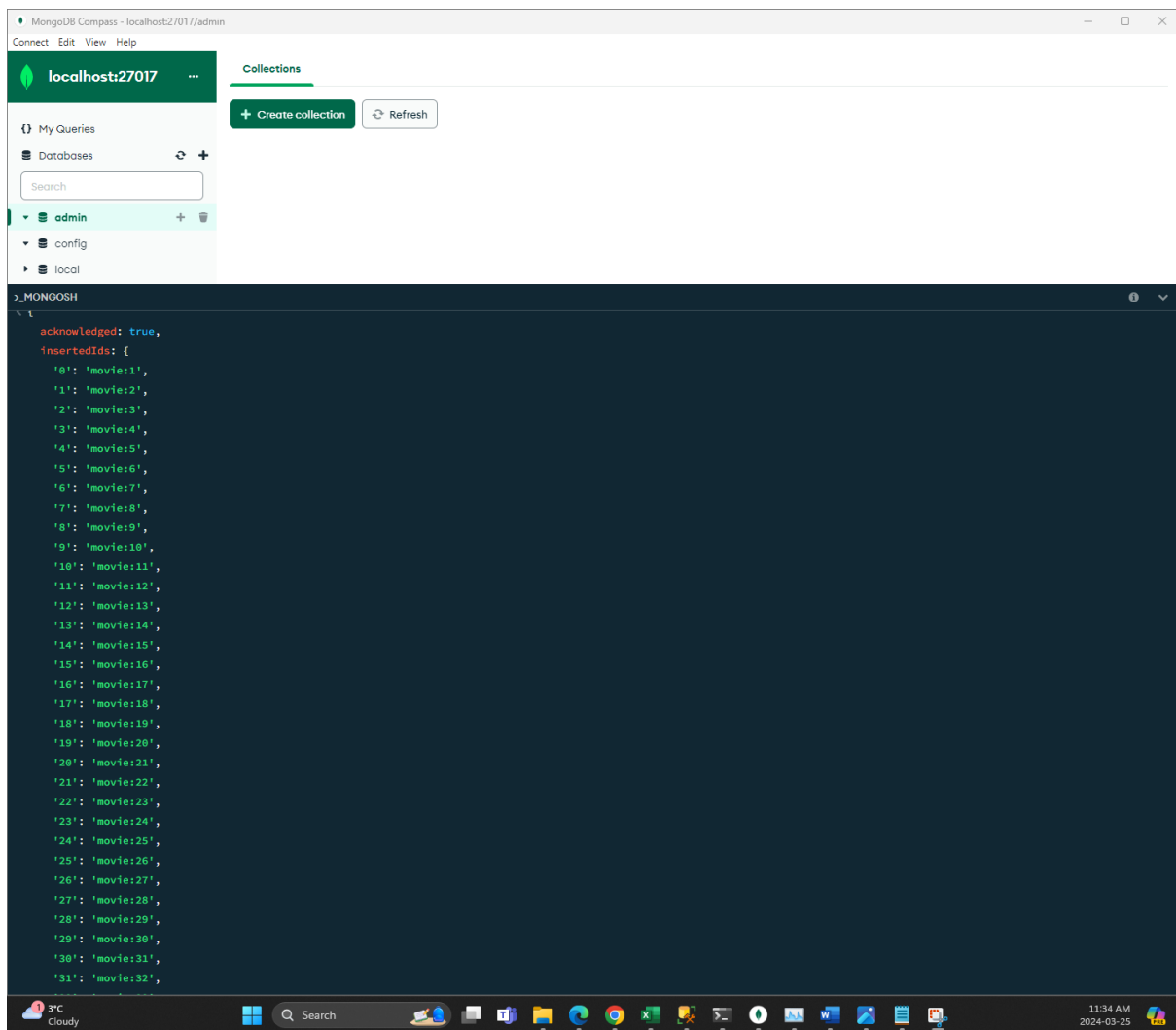
Copy all the script from movie_Script.txt file and execute on the Mongo DB server under Assignment database.

Solution:



The screenshot shows the MongoDB Compass interface. The left sidebar displays the database structure with 'admin', 'config', and 'local' databases listed. The main panel shows the 'Collections' tab for the 'Assignment' database. A collection named 'movie' is visible, containing a single document. The document is a JSON object representing a movie, with fields for '_id', 'title', 'year', 'genre', 'summary', 'country', 'director', and 'actors'.

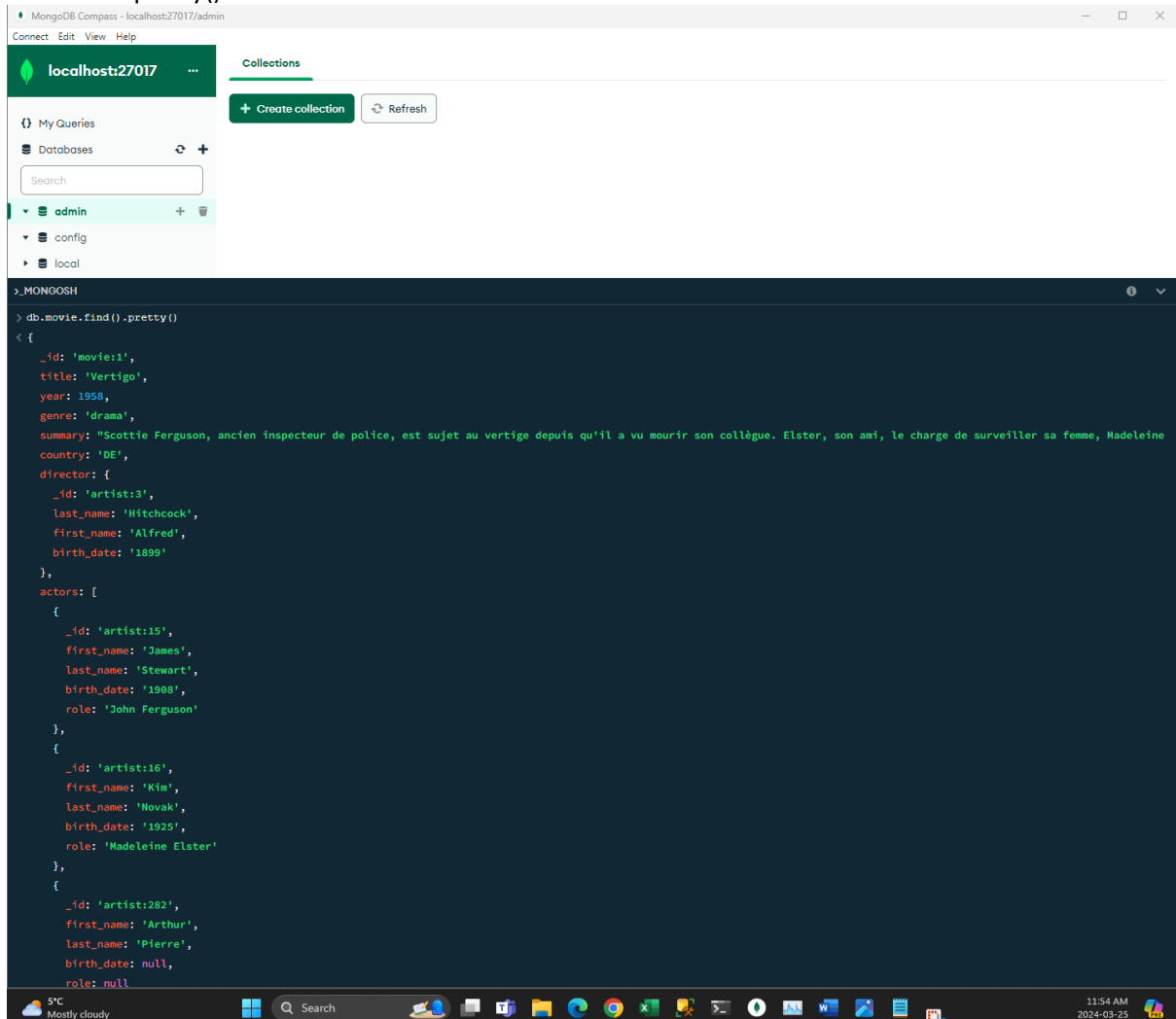
```
>_MONGOSH
> use Assignment
< switched to db Assignment
Assignment> db.movie.insertMany([
  {
    "_id": "movie:1",
    "title": "Vertigo",
    "year": 1958,
    "genre": "drama",
    "summary": "Scottie Ferguson, ancien inspecteur de police, est sujet au vertige depuis qu'il a vu mourir son collègue. Elster, son ami, le charge de surveiller sa femme,
    "country": "DE",
    "director": {
      "_id": "Artist:3",
      "last_name": "Hitchcock",
      "first_name": "Alfred",
      "birth_date": "1899"
    },
    "actors": [
      {
        "_id": "Artist:15",
        "first_name": "James",
        "last_name": "Stewart",
        "birth_date": "1908",
        "role": "John Ferguson"
      },
      {
        "_id": "Artist:16",
        "first_name": "Kim",
        "last_name": "Novak",
        "birth_date": "1925",
        "role": "Madeleine Elster"
      },
      {
        "_id": "Artist:282",
        "first_name": "Arthur",
        "last_name": "Pierre",
        "birth_date": "1925",
        "role": "Madeleine Elster"
      }
    ]
  }
])
```



A. Write the following script to fetch the document on movie collection.

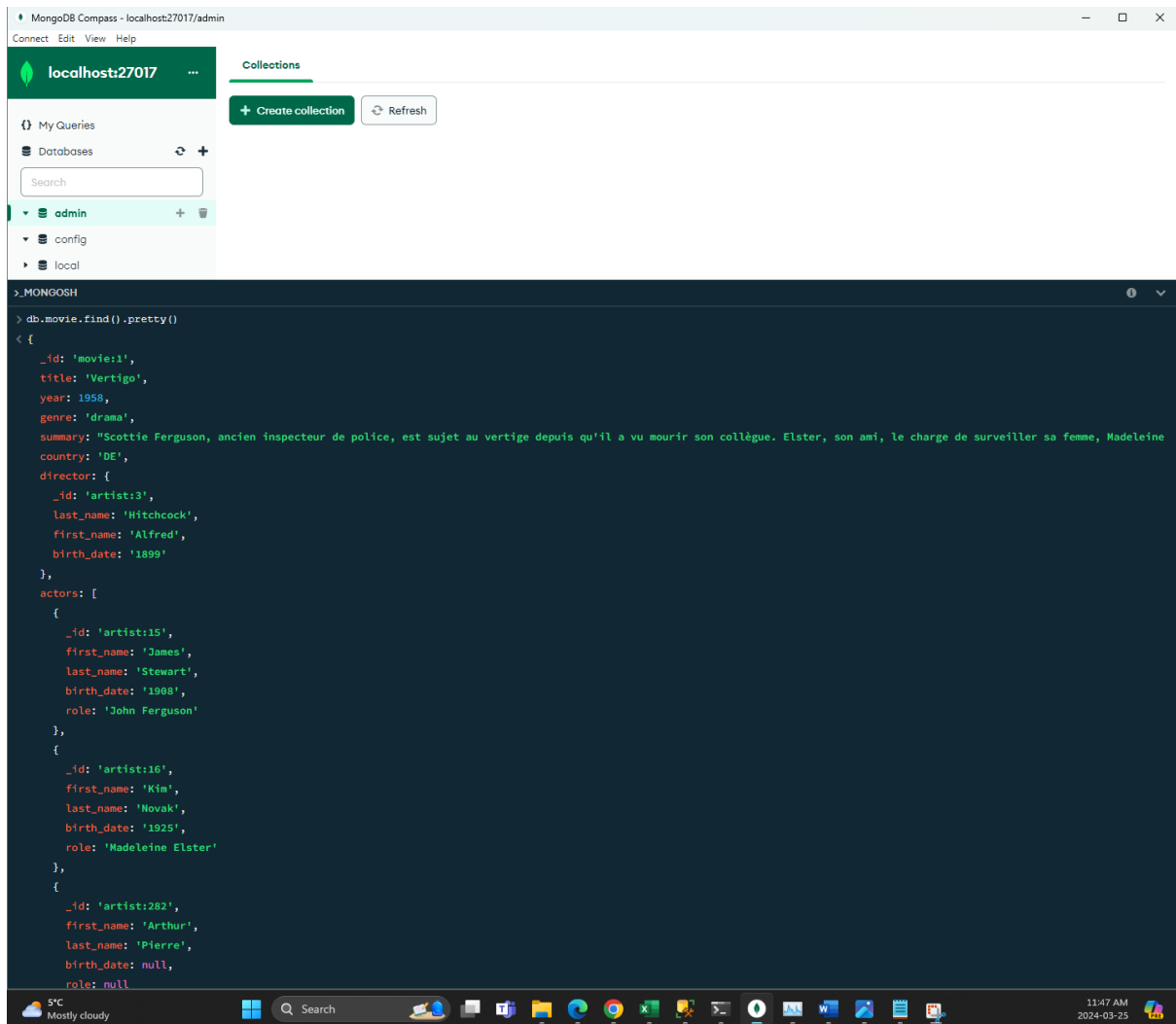
1. Fetch all the document from movie collection.

Solution: We use command `db.movie.find().pretty()` – here movie represents the collection we created to insert the movie data and we use the `find()` method to return all documents in this collection. `pretty()` method is used to make the returned data more readable.



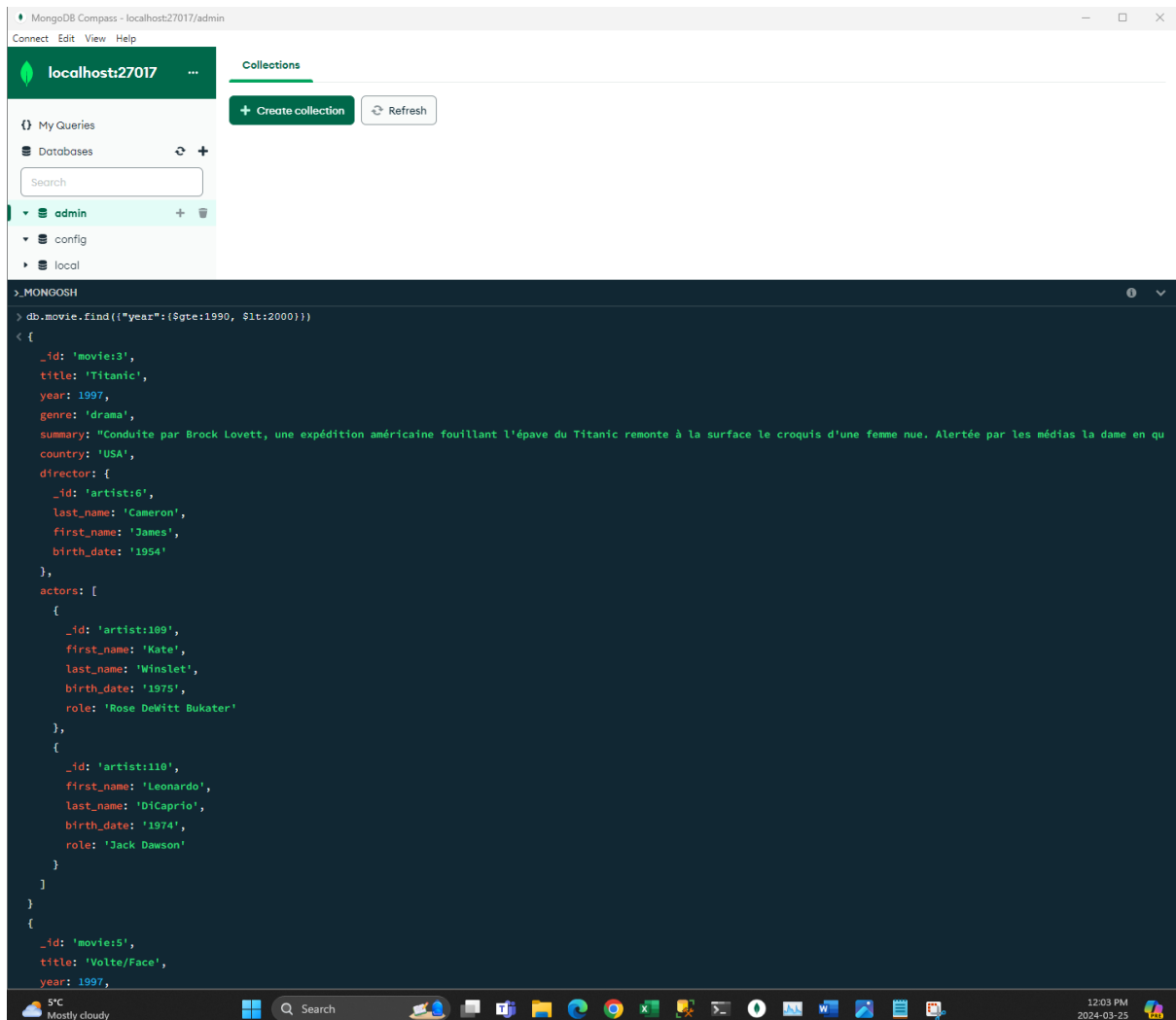
The screenshot displays the MongoDB Compass interface and a terminal window. The Compass window shows the 'admin' database selected, with a search bar and a 'Collections' tab. The terminal window shows the command `db.movie.find().pretty()` being executed, resulting in a JSON document for the movie 'Vertigo'.

```
>_MONGOSH
> db.movie.find().pretty()
< {
  _id: 'movie11',
  title: 'Vertigo',
  year: 1958,
  genre: 'drama',
  summary: "Scottie Ferguson, ancien inspecteur de police, est sujet au vertige depuis qu'il a vu mourir son collègue. Elster, son ami, le charge de surveiller sa femme, Madeleine",
  country: 'DE',
  director: {
    _id: 'artist:3',
    last_name: 'Hitchcock',
    first_name: 'Alfred',
    birth_date: '1899'
  },
  actors: [
    {
      _id: 'artist:15',
      first_name: 'James',
      last_name: 'Stewart',
      birth_date: '1908',
      role: 'John Ferguson'
    },
    {
      _id: 'artist:16',
      first_name: 'Kim',
      last_name: 'Novak',
      birth_date: '1925',
      role: 'Madeleine Elster'
    },
    {
      _id: 'artist:282',
      first_name: 'Arthur',
      last_name: 'Pierre',
      birth_date: null,
      role: null
    }
  ]
}
```



2. Fetch all the document with movies released in the 90s.

Solution: We use command `db.movie.find({"year":{"$gte:1990, $lt:2000}})` – To get the documents that follow our criteria, we pass arguments through the `find()` query method such as `$gte`, which means greater than or equal, and `$lt`, which means less than. We specify the field – “year” and use `$gte` and `$lt` to get the movies released in 1990s.



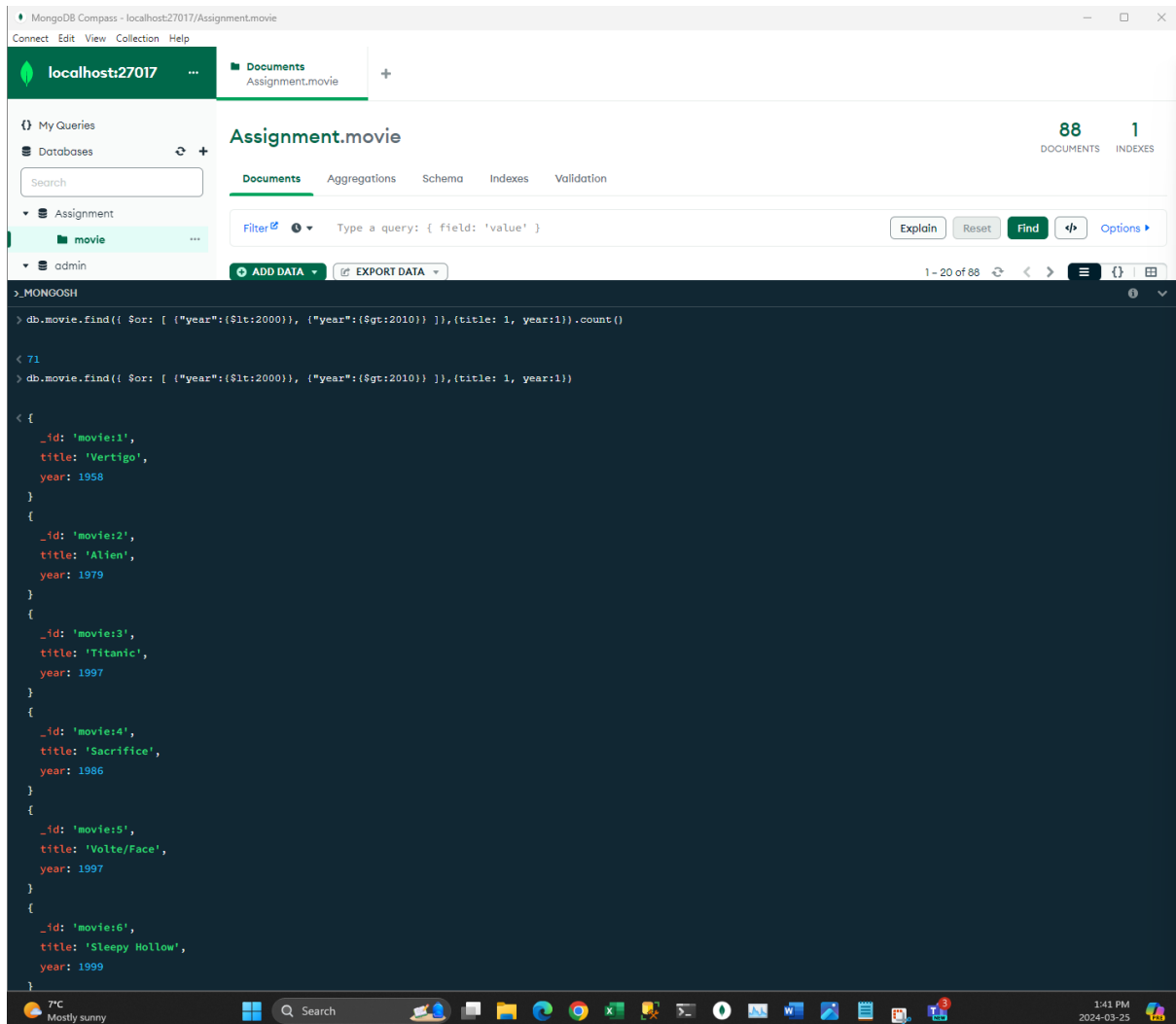
The screenshot shows the MongoDB Compass application running on localhost:27017. The interface includes a sidebar with a database list (admin, config, local) and a main workspace for queries. The 'Collections' tab is active, showing a '+ Create collection' button and a 'Refresh' button. The query editor contains the command `db.movie.find({"year":{"$gte:1990, $lt:2000}})`. The results pane displays two documents from the 'movie' collection:

```
> db.movie.find({"year":{"$gte:1990, $lt:2000}})
< {
  _id: 'movie:3',
  title: 'Titanic',
  year: 1997,
  genre: 'drama',
  summary: "Conduite par Brock Lovett, une expédition américaine fouillant l'épave du Titanic remonte à la surface le croquis d'une femme nue. Alertée par les médias la dame en qu",
  country: 'USA',
  director: {
    _id: 'artist:6',
    last_name: 'Cameron',
    first_name: 'James',
    birth_date: '1954'
  },
  actors: [
    {
      _id: 'artist:189',
      first_name: 'Kate',
      last_name: 'Winslet',
      birth_date: '1975',
      role: 'Rose DeWitt Bukater'
    },
    {
      _id: 'artist:110',
      first_name: 'Leonardo',
      last_name: 'DiCaprio',
      birth_date: '1974',
      role: 'Jack Dawson'
    }
  ]
}
{
  _id: 'movie:5',
  title: 'Volte/Face',
  year: 1997,
```

The Windows taskbar at the bottom shows the system clock as 12:03 PM on 2024-03-25, with a weather widget indicating 5°C and 'Mostly cloudy'.

3. Fetch all the document with movies released before the year 2000 or after 2010.

Solution: We use command `db.movie.find({ $or: [{"year":{$lt:2000}}, {"year":{$gt:2010}}]},{title: 1, year:1})`– It is similar to what we use for movies in the 1990s but we use `$or` to match documents that are either `$lt:2000` or `$gt:2010`. We could also use Projection Query to return only the Title and the year in this one to only get the title and year as shown here, if needed.



4. Fetch all the document with movie “crime” or “drama” genre.

Solution: we use command `db.movie.find({ $or : [{genre:'crime'}, {genre:'drama'}]})` – Its similar to our previous solution where we use \$or but here we switch the ‘year’ with ‘genre’ and specify crime or drama per our requirement.

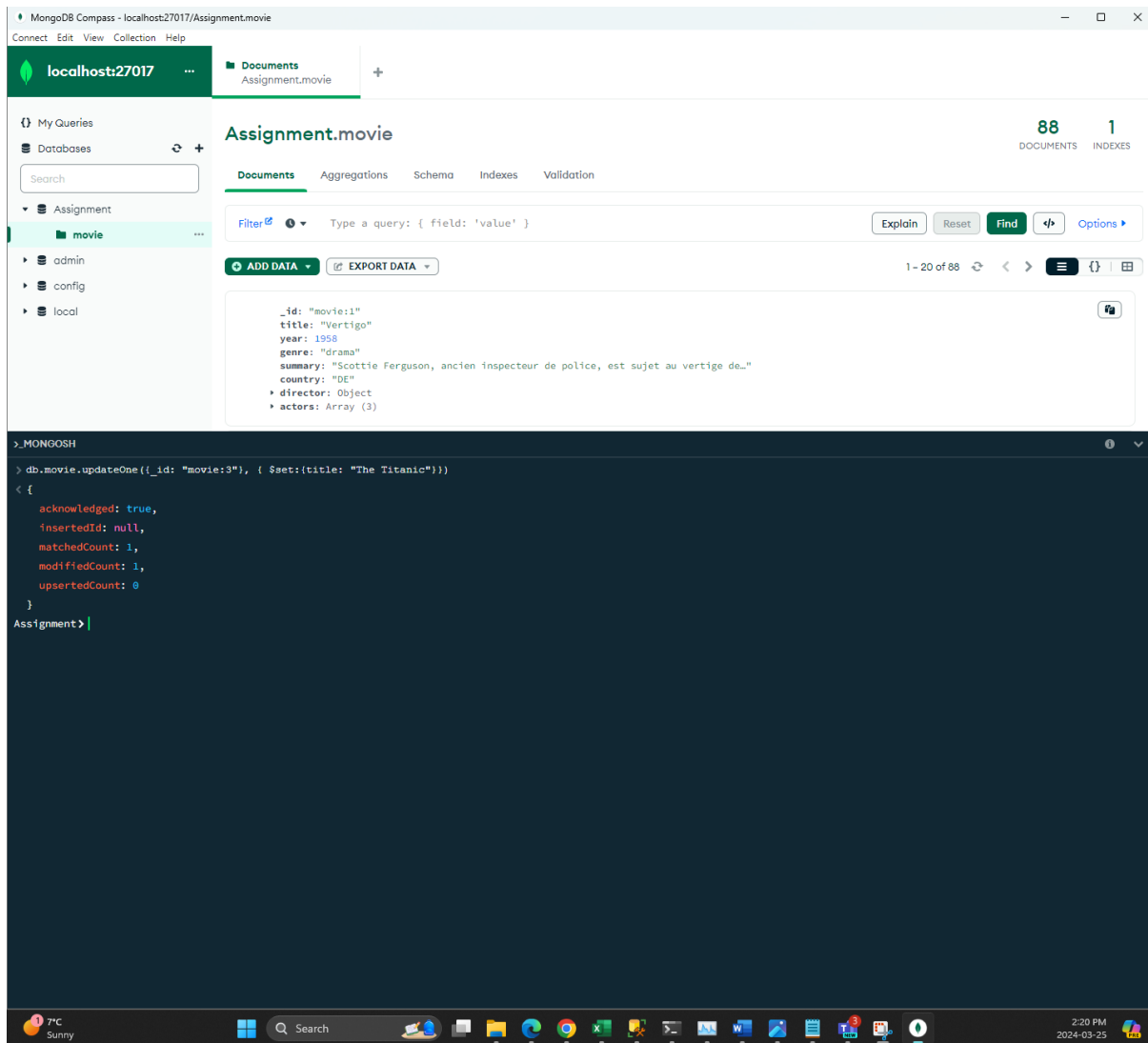
The screenshot shows the MongoDB Compass interface. On the left, the database structure is visible: 'localhost:27017' - Assignment.movie. The 'Documents' tab is selected, showing 88 documents and 1 index. The 'Assignment.movie' document is selected, and the 'Documents' sub-tab is active. A query filter is entered: `{ field: 'value' }`. Below the filter, there are buttons for 'ADD DATA', 'EXPORT DATA', and 'Find'. The main area displays the query result in a dark-themed console window. The query is `db.movie.find({ $or : [{genre:'crime'}, {genre:'drama'}]})`. The result is a JSON document for the movie 'Vertigo' (1958, drama genre), directed by Alfred Hitchcock, featuring James Stewart, Kim Novak, and Arthur Pierre.

```
>_MONGOSH
> db.movie.find({ $or : [{genre:'crime'}, {genre:'drama'}]})
< {
  _id: 'movie:1',
  title: 'Vertigo',
  year: 1958,
  genre: 'drama',
  summary: "Scottie Ferguson, ancien inspecteur de police, est sujet au vertige depuis qu'il a vu mourir son collègue. Elster, son ami, le charge de surveiller sa femme, Madeleine",
  country: 'DE',
  director: {
    _id: 'artist:3',
    last_name: 'Hitchcock',
    first_name: 'Alfred',
    birth_date: '1899'
  },
  actors: [
    {
      _id: 'artist:15',
      first_name: 'James',
      last_name: 'Stewart',
      birth_date: '1908',
      role: 'John Ferguson'
    },
    {
      _id: 'artist:16',
      first_name: 'Kim',
      last_name: 'Novak',
      birth_date: '1925',
      role: 'Madeleine Elster'
    },
    {
      _id: 'artist:282',
      first_name: 'Arthur',
      last_name: 'Pierre',
      birth_date: null,
      role: null
    }
  ]
}
```

B. Write the script to update document on movie collection.

1. Update the movie with title “The Titanic” from “Titanic” where `_id` is "movie:3".

Solution: we use command `db.movie.updateOne({_id: "movie:3"}, { $set:{ title: "The Titanic"}})` – we pass our selection criteria that is `_id: "movie:3"` and the we pass our new value that is the title.



2. Update the movie with country “FR” from "USA" where _id is “movie:3”.

Solution: we use command `db.movie.updateOne({_id: "movie:3"}, { $set:{ country: "FR"}})` – It is similar to our last query but in this one we update the country name.

The screenshot displays the MongoDB Compass interface for the 'Assignment.movie' collection. The left sidebar shows the database structure with 'Assignment' expanded and 'movie' selected. The main panel shows the 'Documents' tab for 'Assignment.movie', displaying 88 documents and 1 index. A sample document is shown with fields: _id: "movie:1", title: "Vertigo", year: 1958, genre: "drama", summary: "Scottie Ferguson, ancien inspecteur de police, est sujet au vertige de...", country: "DE", director: Object, and actors: Array (3). Below the document list, a MongoDB shell terminal is open, showing two update commands and their results. The first command updates the title of the movie with _id "movie:3" to "The Titanic". The second command updates the country of the movie with _id "movie:3" to "FR". Both commands return a success message with fields like acknowledged, insertedId, matchedCount, modifiedCount, and upsertedCount.

```
> db.movie.updateOne({_id: "movie:3"}, { $set:{title: "The Titanic"}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.movie.updateOne({_id: "movie:3"}, { $set:{country: "FR"}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Assignment>
```

3. Update the movie with director's last_name is "Doe" from "Scott" where _id is "movie:9".

Solution: we use command `db.movie.updateOne({_id: "movie:9"}, { $set: { 'director.last_name': "Doe" } })` – It is similar to our last query but in this one we update the director name.

The screenshot displays the MongoDB Compass interface for a local database at localhost:27017. The left sidebar shows the database structure with 'Assignment' expanded to show a 'movie' collection. The main panel shows the 'Assignment.movie' collection with 88 documents and 1 index. A document is selected, showing details like 'summary', 'country', 'director' (with _id, last_name, first_name, birth_date), and 'actors'. Below this, a terminal window shows the execution of the update command: `db.movie.updateOne({_id: "movie:9"}, { $set: { 'director.last_name': "Doe" } })`. The terminal output shows the command was successful, with `acknowledged: true`, `insertedId: null`, `matchedCount: 1`, `modifiedCount: 1`, and `upsertedCount: 0`.

4. Update the movies with year 1990 where year is 1994.

Solution: `db.movie.updateMany({year: "1994"}, { $set:{ year: "1990"}})` – we use the `updateMany()` method and pass our selection criteria `{year: "1994"}` and updates i.e. `{ year: "1990"}`.

The screenshot displays the MongoDB Compass interface and a terminal window. The Compass window shows the 'Assignment.movie' collection with 88 documents and 1 index. The 'movie' collection is selected in the left sidebar. The terminal window shows the following commands and results:

```
> db.movie.updateOne({_id: "movie:9"}, { $set:{ 'director.last_name': "Doe"}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

> db.movie.updateMany({year: "1994"}, { $set:{ year: "1990"}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
Assignment>
```

C. Write the script to Search a string in the document on movie collection.

1. Fetch all the movies genre as “drama”.

Solution: `db.movie.find({genre:'drama'})` – we use the find method and pass the genre criteria through it to get all drama movies.

The screenshot displays the MongoDB Compass interface and a terminal window. The Compass window shows the 'Assignment.movie' collection with 88 documents. The 'Documents' tab is active, showing a preview of a document. The terminal window shows the execution of the query `db.movie.find({genre:'drama'})` and the resulting JSON output for a movie document.

MongoDB Compass Interface:

- Database: localhost:27017
- Collection: Assignment.movie
- Documents: 88
- Filter: Type a query: { field: 'value' }
- Buttons: Explain, Reset, Find, Options
- Export Data: EXPORT DATA

Terminal Output:

```
> .MONGOOSH
> db.movie.find({genre:'drama'})
< {
  _id: 'movie:1',
  title: 'Vertigo',
  year: 1958,
  genre: 'drama',
  summary: "Scottie Ferguson, ancien inspecteur de police, est sujet au vertige depuis qu'il a vu mourir son collègue. Elster, son ami, le charge de surveiller sa femme, Madeleine",
  country: 'DE',
  director: {
    _id: 'artist:3',
    last_name: 'Hitchcock',
    first_name: 'Alfred',
    birth_date: '1899'
  },
  actors: [
    {
      _id: 'artist:15',
      first_name: 'James',
      last_name: 'Stewart',
      birth_date: '1908',
      role: 'John Ferguson'
    },
    {
      _id: 'artist:16',
      first_name: 'Kim',
      last_name: 'Novak',
      birth_date: '1925',
      role: 'Madeleine Elster'
    },
    {
      _id: 'artist:282',

```

2. Fetch all the movies title with “Titanic”.

Solution: `db.movie.find({title: /Titanic/ })` – we use this command to search for movie with title ‘Titanic’ as we previously renamed it to ‘The Titanic’ so `{title: ‘Titanic’}` would not work. We can also use `$regex` to find any words that contain the word ‘Titanic’ in the title – `{title: {$regex: ‘Titanic’}}`

The screenshot displays the MongoDB Compass interface and a terminal window. The Compass window shows the 'Assignment.movie' collection with 88 documents and 1 index. The 'Documents' tab is active, showing a document with fields like 'summary', 'country', 'director', and 'actors'. The terminal window shows the command `db.movie.find({title: /Titanic/ })` and its output, which is a JSON document for 'The Titanic' movie, including details like year (1997), genre ('drama'), summary, country ('FR'), director (James Cameron), and actors (Kate Winslet, Leonardo DiCaprio, Jack Dawson).

```
> db.movie.find({title: /Titanic/ })
{
  "_id": "movie:3",
  "title": "The Titanic",
  "year": 1997,
  "genre": "drama",
  "summary": "Conduite par Brock Lovett, une exp\u00e9dition am\u00e9ricaine fouillant l'\u00e9pave du Titanic remonte \u00e0 la surface le croquis d'une femme nue. Alert\u00e9e par les m\u00e9dias la dame en qu",
  "country": "FR",
  "director": {
    "_id": "artist:6",
    "last_name": "Cameron",
    "first_name": "James",
    "birth_date": "1954"
  },
  "actors": [
    {
      "_id": "artist:109",
      "first_name": "Kate",
      "last_name": "Winslet",
      "birth_date": "1975",
      "role": "Rose DeWitt Bukater"
    },
    {
      "_id": "artist:110",
      "first_name": "Leonardo",
      "last_name": "DiCaprio",
      "birth_date": "1974",
      "role": "Jack Dawson"
    }
  ]
}
```

3. Fetch all the movies director's first_name as "John".

Solution: `db.movie.find({'director.first_name': /John/ })` – similar to our previous solution, we only replace our criteria from title to 'director.first_name'.

The screenshot displays the MongoDB Compass interface and a terminal window. The top part shows the MongoDB Compass window with the 'Assignment.movie' collection selected. The 'Documents' tab is active, showing a list of documents. The bottom part shows a terminal window with the MongoDB shell command and its output.

MongoDB Compass Interface:

- Database: localhost27017
- Collection: Assignment.movie
- Documents: 88, Indexes: 1
- Filter: Type a query: { field: 'value' }
- Buttons: Explain, Reset, Find, Options
- Export Data: EXPORT DATA

MongoDB Shell Command and Output:

```
> MONGOSH
> db.movie.find({'director.first_name': /John/ })
< {
  _id: 'movie:5',
  title: 'Volte/Face',
  year: 1997,
  genre: 'Action',
  summary: "Directeur d'une unité anti-terroriste, Sean Archer recherche Castor Troy, un criminel responsable de la mort de son fils six ans plus tôt. Il parvient à l'arrêter mais",
  country: 'USA',
  director: {
    _id: 'artist:10',
    last_name: 'Woo',
    first_name: 'John',
    birth_date: '1946'
  },
  actors: [
    {
      _id: 'artist:11',
      first_name: 'John',
      last_name: 'Travolta',
      birth_date: '1954',
      role: 'Sean Archer/Castor Troy'
    },
    {
      _id: 'artist:12',
      first_name: 'Nicolas',
      last_name: 'Cage',
      birth_date: '1964',
      role: 'Castor Troy/Sean Archer'
    }
  ]
}
```


4. Fetch all the movies title start and end with "t".

Solution: `db.movie.find({title: { $regex: "^t.*t$", $options: "i" }})` – we use `$regex` to perform this task, `^t` means the string starts with t, `.*` allows any characters in the middle and `t$` means it ends with a t. we use `$options` to ignore case.

The screenshot displays the MongoDB Compass interface. The top bar shows the connection to 'localhost:27017' and the 'Assignment.movie' database. The left sidebar lists the databases and collections, with 'movie' selected. The main panel shows the 'Assignment.movie' collection with 88 documents and 1 index. A query filter is applied: `{title: { $regex: '^t.*t$', $options: 'i' }}`. The results are displayed in a table format, showing the first document. The document details are as follows:

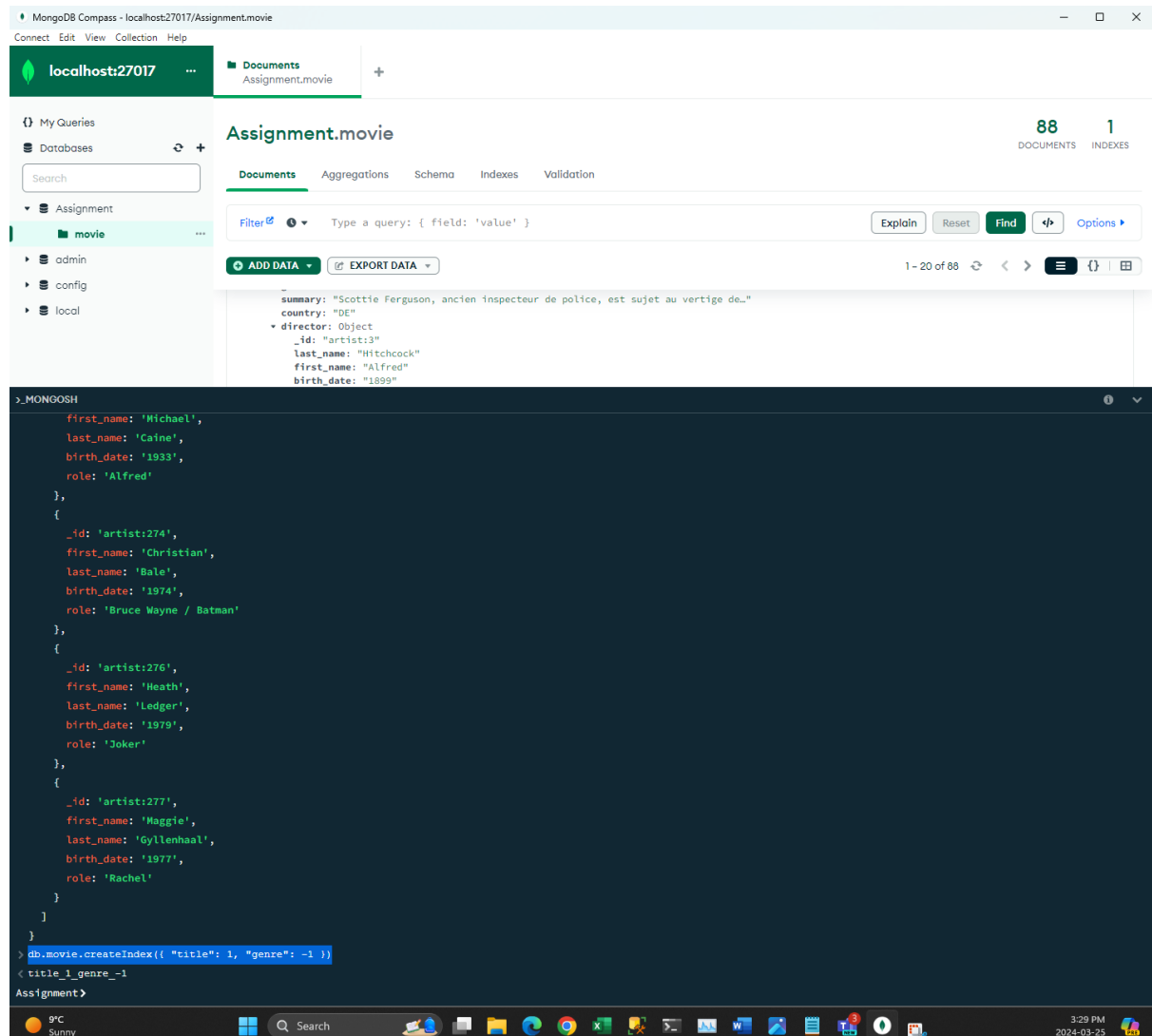
Field	Value
<code>_id</code>	<code>'movie:89'</code>
<code>title</code>	<code>'The Dark Knight'</code>
<code>year</code>	<code>2008</code>
<code>genre</code>	<code>'Science-fiction'</code>
<code>summary</code>	<code>'Dans ce nouveau volet, Batman augmente les mises dans sa guerre contre le crime. Avec l'appui du lieutenant de police Jim Gordon et du procureur de Gotham, Harvey Dent'</code>
<code>country</code>	<code>'USA'</code>
<code>director</code>	<code>{_id: 'artist:266', last_name: 'Nolan', first_name: 'Christopher', birth_date: '1970'}</code>
<code>actors</code>	<code>[{ _id: 'artist:22', first_name: 'Morgan', last_name: 'Freeman', birth_date: '1937', role: 'Lucius Fox' }, { _id: 'artist:116', first_name: 'Gary', last_name: 'Oldman', birth_date: '1958', role: 'Gordon' }, { _id: 'artist:269', ...}]</code>

The bottom panel shows the MongoDB shell with the query `db.movie.find({title: { $regex: "^t.*t$", $options: "i" }})` and the resulting JSON output.

D. Write the script to create the indexes on movie collection.

1. Create a single index with multiple attributes (title by ascending order and genre by descending order) on movie.

Solution: `db.movie.createIndex({ "title": 1, "genre": -1 })` – we use `createIndex()` method and specify the order of attributes for the index. 'title': 1 implies an ascending order and 'genre': -1 implies a descending order.



To add year as a single index by descending order we use `db.movie.createIndex({ "year": -1 })`

The screenshot displays the MongoDB Compass interface and the MongoDB Shell. The top section shows the 'Assignment.movie' collection with 88 documents and 1 index. The left sidebar lists the databases and collections, including 'movie'. The main panel shows the 'Documents' tab with a list of movie documents. The bottom section shows the MongoDB Shell with the following commands and output:

```
> MONGODB
{
  birth_date: '1933',
  role: 'Alfred'
},
{
  _id: 'artist:274',
  first_name: 'Christian',
  last_name: 'Bale',
  birth_date: '1974',
  role: 'Bruce Wayne / Batman'
},
{
  _id: 'artist:276',
  first_name: 'Heath',
  last_name: 'Ledger',
  birth_date: '1979',
  role: 'Joker'
},
{
  _id: 'artist:277',
  first_name: 'Maggie',
  last_name: 'Gyllenhaal',
  birth_date: '1977',
  role: 'Rachel'
}
]
}
}
> db.movie.createIndex({ "title": 1, "genre": -1 })
< title_1_genre_-1
> db.movie.createIndex({ "year": -1 })
< year_-1
Assignment>
```

The bottom status bar shows the system time as 3:33 PM on 2024-03-25.

2. Create a wildcard Index on director first_name order by ascending order on movie collection.

Solution: `db.movie.createIndex({ "director.first_name.$**": 1 })` – we use the `director.first_name` and `$**` to create a wildcard index.

The screenshot displays the MongoDB Compass interface for the 'Assignment.movie' collection. The left sidebar shows the database structure with 'movie' selected. The main panel shows the 'Documents' tab with a list of movie documents. Below the Compass interface, a terminal window shows the following commands and output:

```
>_MONGOOSH
},
{
  _id: 'artist:274',
  first_name: 'Christian',
  last_name: 'Bale',
  birth_date: '1974',
  role: 'Bruce Wayne / Batman'
},
{
  _id: 'artist:276',
  first_name: 'Heath',
  last_name: 'Ledger',
  birth_date: '1979',
  role: 'Joker'
},
{
  _id: 'artist:277',
  first_name: 'Maggie',
  last_name: 'Gyllenhaal',
  birth_date: '1977',
  role: 'Rachel'
}
]
}
}
> db.movie.createIndex({ "title": 1, "genre": -1 })
< title_1_genre_-1
> db.movie.createIndex({ "year": -1 })
< year_-1
> db.movie.createIndex({ "director.first_name.$**": 1 })
< director.first_name.$**_1
Assignment>
```

The terminal output shows the successful creation of the wildcard index on the `director.first_name` field.

3. Create a unique index on country attribute by ascending order on movie collection.

Solution: `db.movie.createIndex({ "country": 1 }, { unique: true })` – we use the `createIndex` method and specify “country”, its order and `unique: true` to create a unique index. Since the fields in ‘country’ are common between many movie documents, it does not generate a unique index due to duplicates.

The screenshot displays the MongoDB Compass interface and the MONGODB shell. In the Compass window, the 'Assignment.movie' collection is selected, showing 88 documents and 1 index. The 'Documents' tab is active, displaying a document with a summary and a director object. The MONGODB shell window shows the following commands and their output:

```
> db.movie.createIndex({ "title": 1, "genre": -1 })
< title_1_genre_-1
> db.movie.createIndex({ "year": -1 })
< year_-1
> db.movie.createIndex({ "director.first_name.$**": 1 })
< director.first_name.$**_1
> db.movie.createIndex({ "country": 1 }, { unique: true })
Error: Error: on Assignment.movie ( 6fb33449-bc46-4586-ba8b-f1e8db6408c2 ) :: caused by :: E11000 duplicate key error collection: Assignment.movie index: country_1 dup key: { country: "FR" }
```

The error message indicates that the unique index creation failed due to a duplicate key for the 'country' field, specifically for the value 'FR'.

4. Create a multikey Index on actors first_name order by ascending order on movie collection.

Solution: `db.movie.createIndex({ "actors.first_name": 1 })` – we use `createIndex()` method but since we are indexing a subdocument or element of an array, it creates a multikey index.

The image shows a MongoDB Compass interface for the 'Assignment.movie' collection. The left sidebar shows the database structure with 'movie' selected. The main panel displays three document entries. Below the documents, a terminal window shows the following commands and output:

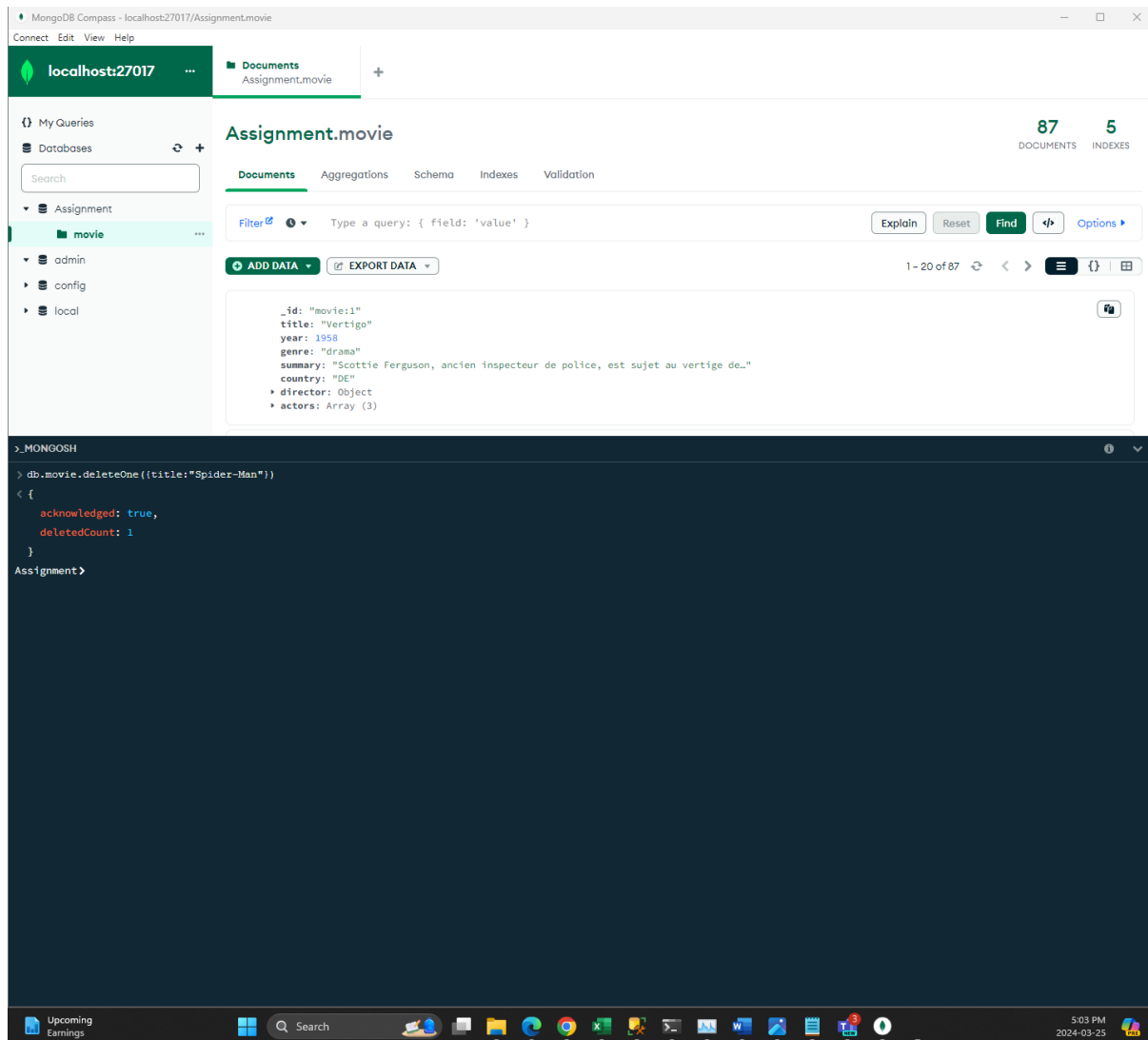
```
> .MONGOSH
> db.movie.createIndex({ "title": 1, "genre": -1 })
< title_1_genre_-1
> db.movie.createIndex({ "year": -1 })
< year_-1
> db.movie.createIndex({ "director.first_name.$**": 1 })
< director.first_name.$**_1
> db.movie.createIndex({ "country": 1 }, { unique: true })
Error: E11000 duplicate key error collection: Assignment.movie index: country_1 dup key: { country: "FR" }
> db.movie.createIndex({ "actors.first_name": 1 })
< actors.first_name_1
Assignment>
```

The error message indicates a duplicate key error for the 'country' index. The terminal also shows the weather and system time at the bottom.

E. Write the script to delete document on movie collection.

1. Delete the movie with title "Spider-Man".

Solution: `db.movie.deleteOne({title:"Spider-Man"})` – we use `deleteOne()` method and specify the title to delete movie Spider-Man.



2. Delete the movies which is released in 1990.

Solution: `db.movie.deleteMany({year:1990})` – We use `DeleteMany()` method and specify the year to delete all movies in 1990.

The screenshot displays the MongoDB Compass interface for the 'Assignment.movie' database. The left sidebar shows the database structure with 'movie' selected. The main panel shows a document for 'Vertigo' (year: 1958). Below the document, the Mongosh terminal shows the command `db.movie.deleteMany({year:1990})` and its output: `{ acknowledged: true, deletedCount: 7 }`. The bottom of the image shows a Windows taskbar with the date 2024-03-23 and time 5:07 PM.

MongoDB Compass - localhost:27017/Assignment.movie

Connect Edit View Help

localhost:27017

Documents
Assignment.movie

Assignment.movie

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' }

EXPLAIN RESET FIND Options

ADD DATA EXPORT DATA

1 - 20 of 87

`{
 _id: "movie:1"
 title: "Vertigo"
 year: 1958
 genre: "drama"
 summary: "Scottie Ferguson, ancien inspecteur de police, est sujet au vertige de..."
 country: "DE"
 director: Object
 actors: Array (3)
}`

> MONGOSH

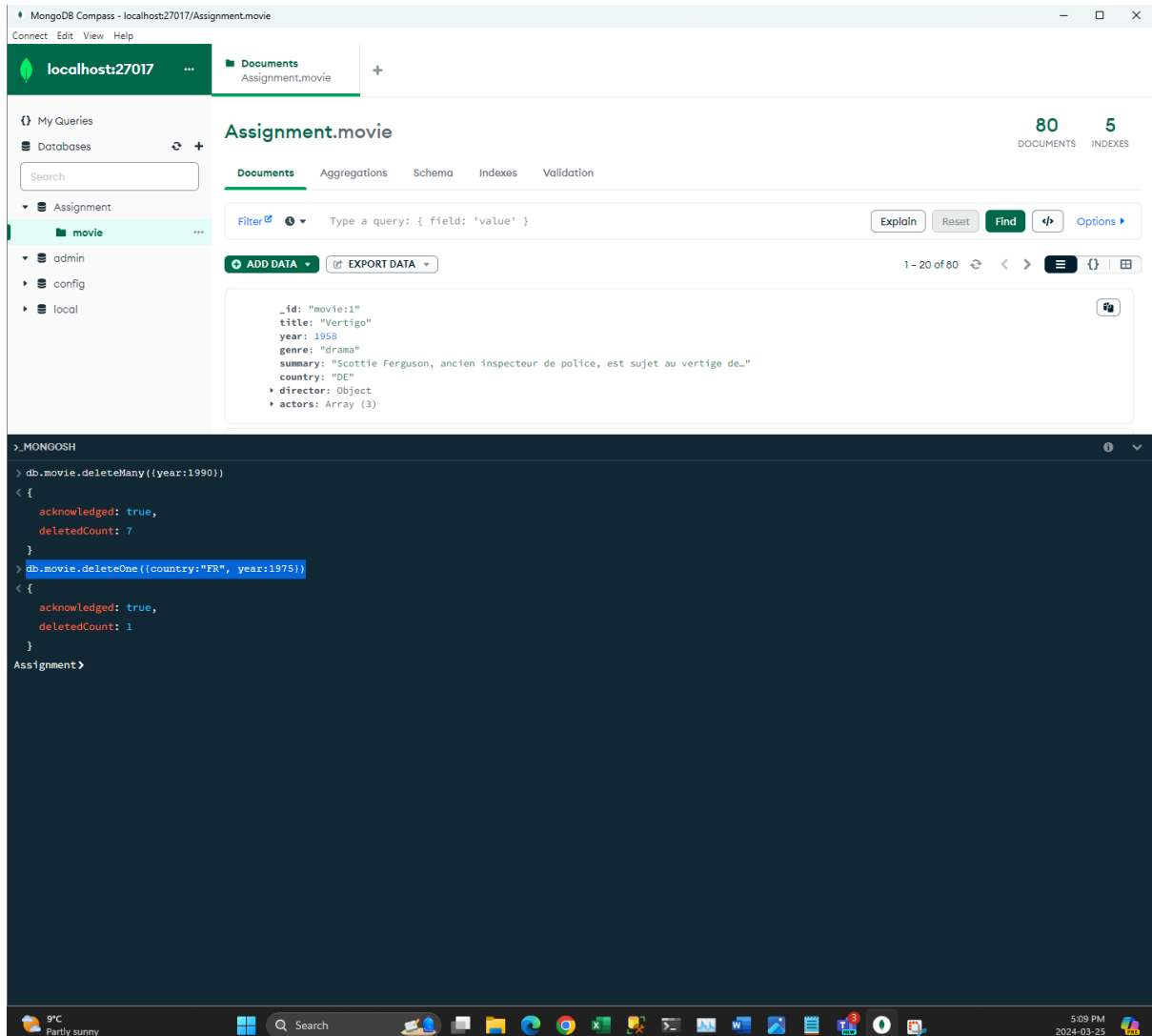
```
> db.movie.deleteMany({year:1990})  
< {  
  acknowledged: true,  
  deletedCount: 7  
}
```

Assignment>

9°C Partly sunny 5:07 PM 2024-03-23

3. Delete the movie which is from country FR and year 1975.

Solution: `db.movie.deleteOne({country:"FR", year:1975})` – we use `deleteOne()` method and specify the country and year of the movie document we want to delete.



4. Delete the movies with genre is drama.

Solution: `db.movie.deleteMany({genre:"drama"})` – we use the `deleteMany()` method and specify the genre of movies to delete.

