

## Contents

<b>Midterm project - Mongo DB.....</b>	<b>2</b>
<b>A Write the following script to fetch the document on movie collection .....</b>	<b>2</b>
<b>B Write the Script to update document on movie collection.</b>	<b>22</b>
<b>C. Write the script to Search a string in the document on movie collection.</b>	<b>33</b>
<b>D Write the script to create the indexes on movie collection.</b>	<b>43</b>
<b>E. Write the script to delete document on movie collection..</b>	<b>54</b>

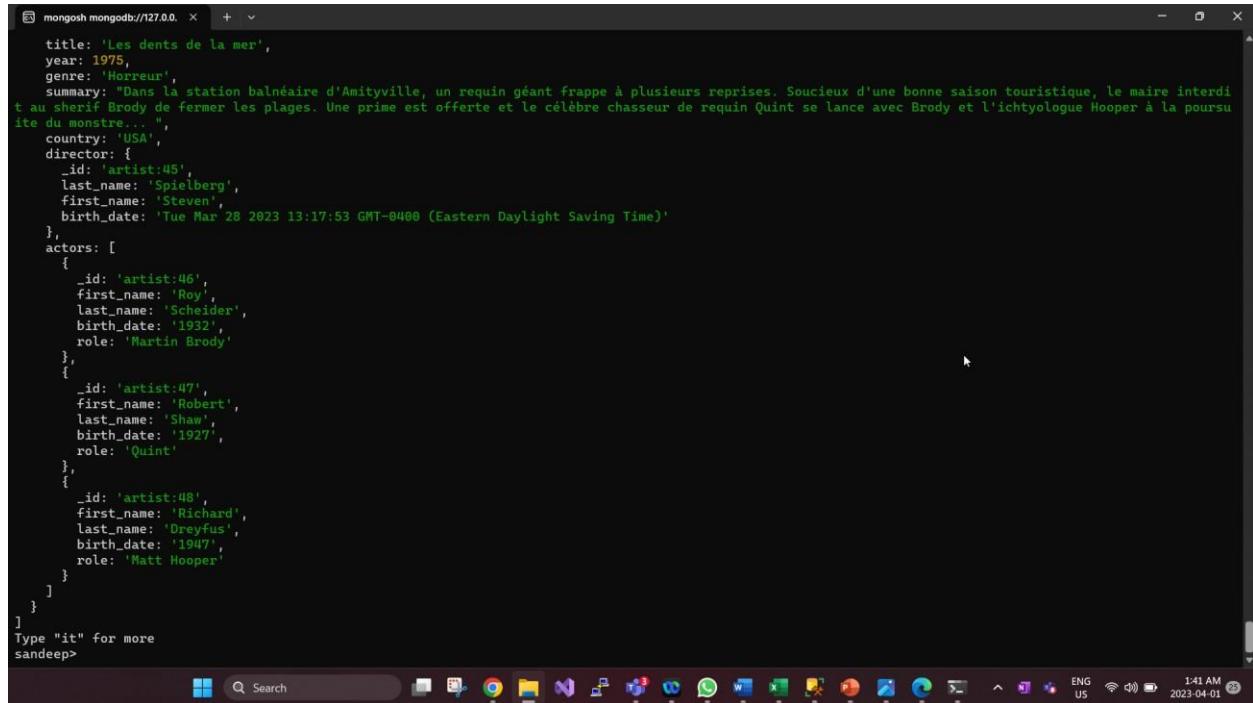
## Midterm project - Mongo DB

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

### 1. Fetch all the document from movie collection.

To fetch all documents from the "movie" collection in MongoDB, I used the `find()` method without any parameters. Here's an example command

**`db.movie.find()`**



```
mongosh mongodb/127.0.0. + v
{
  title: 'Les dents de la mer',
  year: 1975,
  genre: 'Horreur',
  summary: "Dans la station balnéaire d'Amityville, un requin géant frappe à plusieurs reprises. Soucieux d'une bonne saison touristique, le maire interdit au shérif Brody de fermer les plages. Une prime est offerte et le célèbre chasseur de requin Quint se lance avec Brody et l'ichtyologue Hooper à la poursuite du monstre... ",
  country: 'USA',
  director: {
    _id: 'artist:45',
    last_name: 'Spielberg',
    first_name: 'Steven',
    birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
  },
  actors: [
    {
      _id: 'artist:46',
      first_name: 'Roy',
      last_name: 'Scheider',
      birth_date: '1932',
      role: 'Martin Brody'
    },
    {
      _id: 'artist:47',
      first_name: 'Robert',
      last_name: 'Shaw',
      birth_date: '1927',
      role: 'Quint'
    },
    {
      _id: 'artist:48',
      first_name: 'Richard',
      last_name: 'Dreyfus',
      birth_date: '1947',
      role: 'Matt Hooper'
    }
  ]
}
Type "it" for more
sanddeep>
```

I used count command to check the number of movie collection

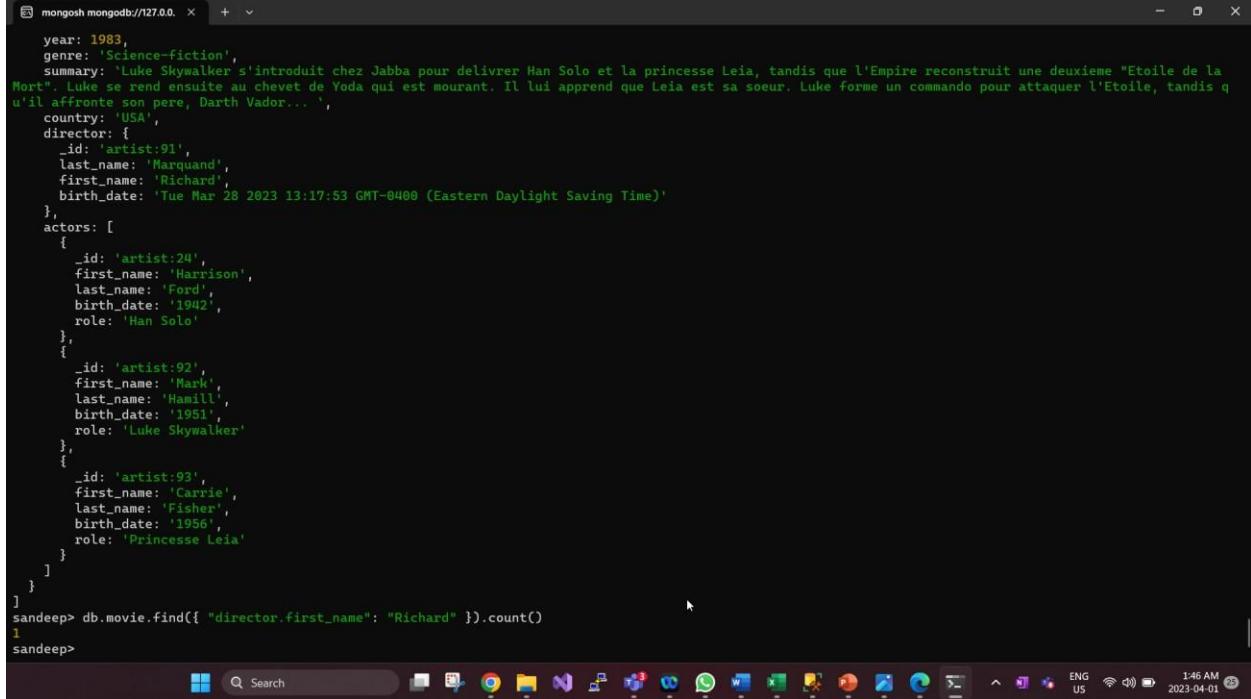
```
mongosh mongodb://127.0.0.1:27017
genre: 'Horreur',
summary: "Dans la station balnéaire d'Amityville, un requin géant frappe à plusieurs reprises. Soucieux d'une bonne saison touristique, le maire interdit au shérif Brody de fermer les plages. Une prime est offerte et le célèbre chasseur de requin Quint se lance avec Brody et l'ichtyologue Hooper à la poursuite du monstre...",
country: 'USA',
director: {
  _id: 'artist:45',
  last_name: 'Spielberg',
  first_name: 'Steven',
  birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
},
actors: [
  {
    _id: 'artist:46',
    first_name: 'Roy',
    last_name: 'Scheider',
    birth_date: '1932',
    role: 'Martin Brody'
  },
  {
    _id: 'artist:47',
    first_name: 'Robert',
    last_name: 'Shaw',
    birth_date: '1927',
    role: 'Quint'
  },
  {
    _id: 'artist:48',
    first_name: 'Richard',
    last_name: 'Dreyfus',
    birth_date: '1947',
    role: 'Matt Hooper'
  }
]
}
Type "it" for more
sandeep> db.movie.find().count()
88
sandeep>
```

## 2 Fetch all the document with director firstname “Richard”.

I have first name of director was Richard so I had to fetch Richard which is first name of director while using `db.movie.find({ "director.first_name": "Richard" })`

```
mongosh mongodb://127.0.0.1:27017
sandeep> db.movie.find({ "director.first_name": "Richard" })
[ {
  _id: 'movie:34',
  title: 'Le retour du Jedi',
  year: 1983,
  genre: 'Science-fiction',
  summary: 'Luke Skywalker s\'introduit chez Jabba pour livrer Han Solo et la princesse Leia, tandis que l\'Empire reconstruit une deuxième "Étoile de la Mort". Luke se rend ensuite au chevet de Yoda qui est mourant. Il lui apprend que Leia est sa sœur. Luke forme un commando pour attaquer l\'Étoile, tandis qu\'il affronte son père, Darth Vader...',
  country: 'USA',
  director: {
    _id: 'artist:91',
    last_name: 'Marquand',
    first_name: 'Richard',
    birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
  },
  actors: [
    {
      _id: 'artist:24',
      first_name: 'Harrison',
      last_name: 'Ford',
      birth_date: '1942',
      role: 'Han Solo'
    },
    {
      _id: 'artist:92',
      first_name: 'Mark',
      last_name: 'Hamill',
      birth_date: '1951',
      role: 'Luke Skywalker'
    },
    {
      _id: 'artist:93',
      first_name: 'Carrie',
      last_name: 'Fisher',
      birth_date: '1956',
      role: 'Princesse Leia'
    }
  ]
}
```

I wanted to know that how many titles where I can see directors name Richard then I used count to know how many there are where I got Richard as shown in Figure



```
mongosh mongodb://127.0.0.1:27017
year: 1983,
genre: 'Science-fiction',
summary: 'Luke Skywalker s'introduit chez Jabba pour livrer Han Solo et la princesse Leia, tandis que l'Empire reconstruit une deuxième "Etoile de la Mort". Luke se rend ensuite au chevet de Yoda qui est mourant. Il lui apprend que Leia est sa soeur. Luke forme un commando pour attaquer l'Etoile, tandis qu'il affronte son père, Darth Vader... ',
country: 'USA',
director: {
  _id: 'artist:91',
  last_name: 'Marguand',
  first_name: 'Richard',
  birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
},
actors: [
  {
    _id: 'artist:24',
    first_name: 'Harrison',
    last_name: 'Ford',
    birth_date: '1942',
    role: 'Han Solo'
  },
  {
    _id: 'artist:92',
    first_name: 'Mark',
    last_name: 'Hamill',
    birth_date: '1951',
    role: 'Luke Skywalker'
  },
  {
    _id: 'artist:93',
    first_name: 'Carrie',
    last_name: 'Fisher',
    birth_date: '1956',
    role: 'Princesse Leia'
  }
]
sandeep> db.movie.find({ "director.first_name": "Richard" }).count()
1
sandeep>
```

### 3 Fetch all the document with director “Eastwood Clint”

To fetch all the documents from a MongoDB collection called "movie" that have a director field matching "Eastwood Clint", Here I used find command to fetch the data like db.movie.find({"director.first\_name" )

```
[mongosh mongoDB://127.0.0.1:27017] + x
}
]
sandeep> db.movie.find({"director.first_name": "Clint", "director.last_name": "Eastwood"})
[
  {
    _id: 'movie:8',
    title: 'Impitoyable',
    year: 1992,
    genre: 'Western',
    summary: "Légendaire hors-la-loi, William Munny s'est reconvertis depuis onze ans en paisible fermier. Il reprend néanmoins les armes pour traquer deux tueurs en compagnie de son vieil ami Ned Logan. Mais ce dernier est capturé, puis exécuté. L'honneur et l'amitié imposent dès lors à Munny de redevenir une dernière fois le héros qu'il fut jadis...",
    country: 'USA',
    director: {
      _id: 'artist:20',
      last_name: 'Eastwood',
      first_name: 'Clint',
      birth_date: '1930'
    },
    actors: [
      {
        _id: 'artist:20',
        first_name: 'Clint',
        last_name: 'Eastwood',
        birth_date: '1930',
        role: 'William Munny'
      },
      {
        _id: 'artist:21',
        first_name: 'Gene',
        last_name: 'Hackman',
        birth_date: '1930',
        role: 'Little Bill Daggett'
      },
      {
        _id: 'artist:22',
        first_name: 'Morgan',
        last_name: 'Freeman',
        birth_date: '1937',
        role: 'Ned Logan'
      }
    ]
  }
]
```

#### 4 Fetch all the document with actor “Bruce Willis”.

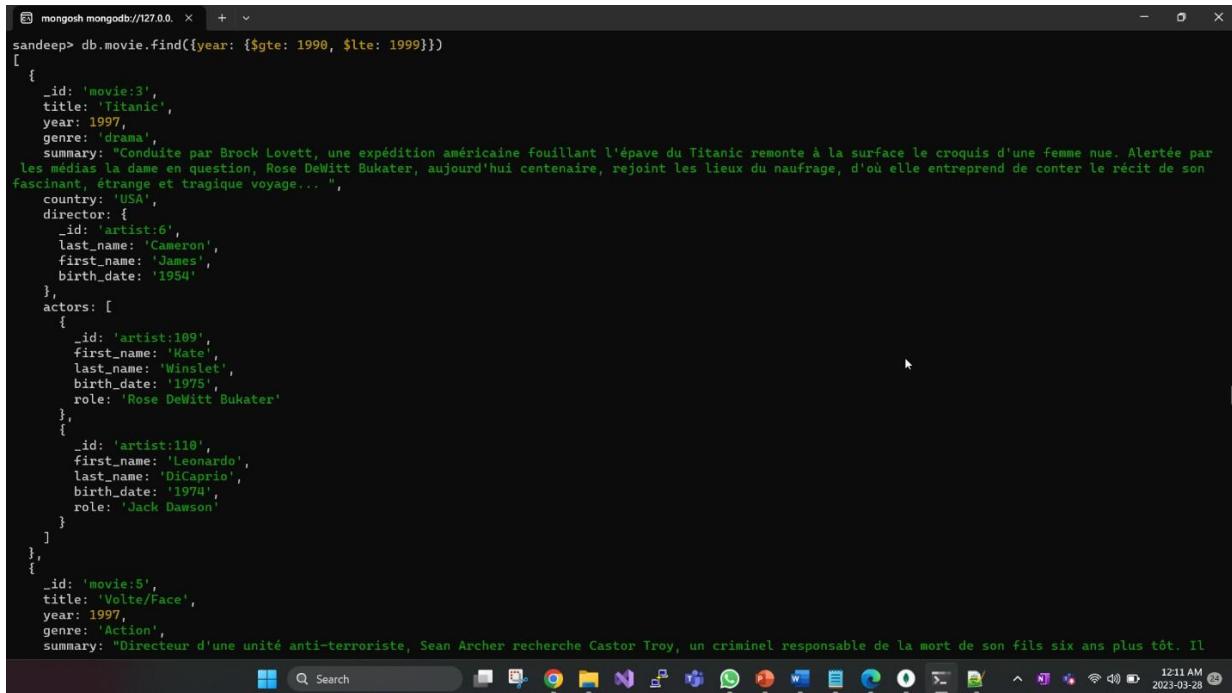
To fetch all the documents from a MongoDB collection called "movie" that have an actor field containing "Bruce Willis", I used

```
db.movie.find({"actors.first_name": "Bruce", "actors.last_name": "Willis"})
```

```
[mongosh mongoDB://127.0.0.1:27017] + x
sandeep> db.movie.find({"actors.first_name": "Bruce", "actors.last_name": "Willis"})
[
  {
    _id: 'movie:11',
    title: 'Piège de cristal',
    year: 1988,
    genre: 'Action',
    summary: "John McClane, policier new-yorkais, vient passer Noël à Los Angeles auprès de sa femme. Dans le bâtiment où elle travaille, il se retrouve témoin de la prise en otage de tout le personnel par 12 terroristes. Objectif de ces derniers, vider les coffres de la société. Cache mais isolé, il entreprend de prévenir l'extermination...",
    country: 'USA',
    director: {
      _id: 'artist:26',
      last_name: 'McTiernan',
      first_name: 'John',
      birth_date: '1951'
    },
    actors: [
      {
        _id: 'artist:27',
        first_name: 'Bruce',
        last_name: 'Willis',
        birth_date: '1955',
        role: 'McClane'
      }
    ],
    {
      _id: 'movie:12',
      title: '58 minutes pour vivre',
      year: 1990,
      genre: 'Action',
      summary: "Venu attendre sa femme à l'aéroport, le policier John McClane remarque la présence de terroristes qui ont pris le contrôle des pistes, empêchant tous les avions d'atterrir et menaçant de laisser les appareils en vol tourner jusqu'à épuisement de leur kérosène. John n'a devant lui que 58 minutes pour éviter la catastrophe...",
      country: 'USA',
      director: {
        _id: 'artist:28',
        last_name: 'Harlin',
        first_name: 'Renny',
      }
    }
]
```

## 5 Fetch all the document with director birthdate is older than 1900.

To fetch all the documents from a MongoDB collection called "movie" that have a director with a birthdate older than 1900, I used **db.movie.find({year: {\$gte: 1990, \$lte: 1999}})**



```
| mongosh mongoDB/127.0.0.1 + v
| sandeep> db.movie.find({year: {$gte: 1990, $lte: 1999}})
[
  {
    _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 question, Rose DeWitt Bukater, aujourd'hui centenaire, rejoint les lieux du naufrage, d'où elle entreprend de raconter le récit de son fascinant, étrange et tragique voyage...",
    country: 'USA',
    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'
      }
    ]
  },
  {
    _id: 'movie:5',
    title: 'Vole/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"
  }
]
```

## 6 Fetch all the document with movies released in the 90s.

I had to fetch all those movie which have been released in 90s so I used find command to find it **db.movie.find({ year : {\$gte: 1990, \$lt: 2000}}**

```
mongosh mongodb://127.0.0.1:27017
sandeep> db.movie.find({ releaseDate: { $gte: 1990,$lt: 2000}})
sandeep> db.movie.find({Date: { $gte: 1990,$lt: 2000}})
sandeep> 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 question, Rose DeWitt Bukater, aujourd'hui centenaire, rejoint les lieux du naufrage, d'où elle entreprend de conter le récit de son fascinant, étrange et tragique voyage... ",
    country: 'USA',
    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'
      }
    ],
    _id: 'movie:5',
  }
]
```

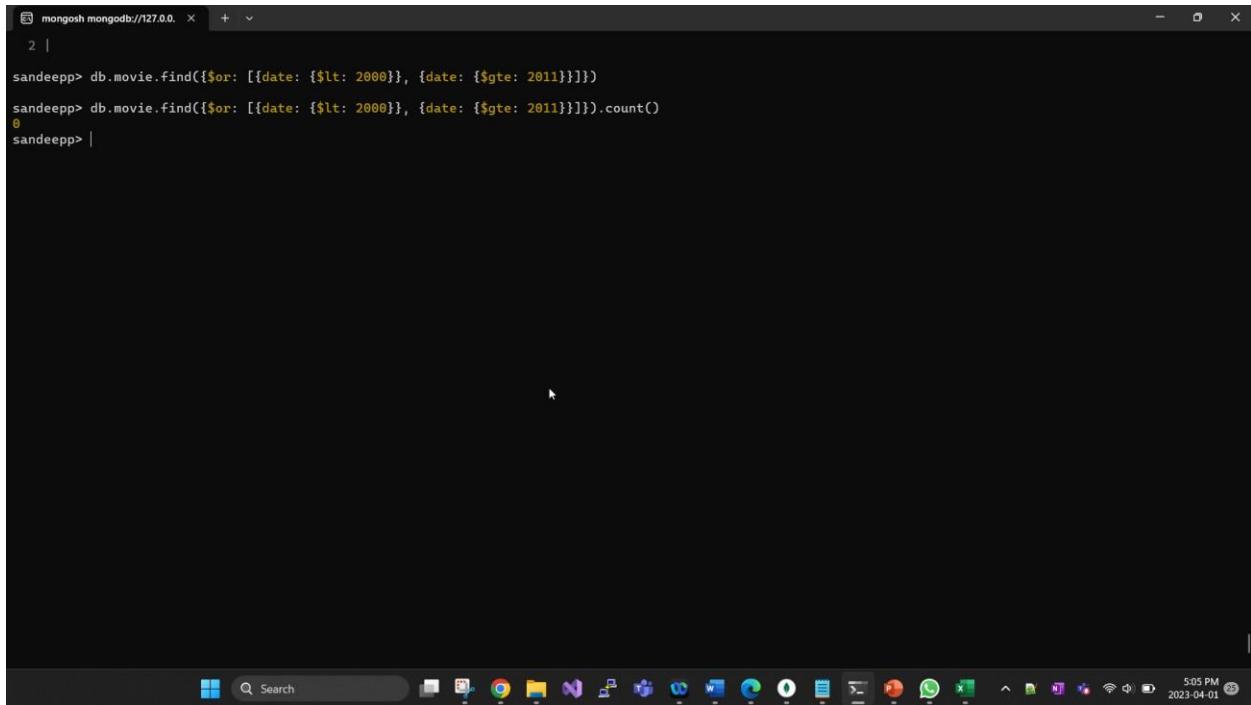
I used count command to know that how many movies which released in 90s so at the end of previous command I used. count() where I got that there are 24 movies as shown in figure.

```
mongosh mongodb://127.0.0.1:27017
title: 'Une journée en enfer',
year: 1995,
genre: 'Action',
summary: 'John McClane est cette fois-ci aux prises avec un maître chanteur, facétieux et dangereux, qui dépose des bombes dans New York.',
country: 'USA',
director: {
  _id: 'artist:168',
  last_name: 'Grier',
  first_name: 'Pam',
  birth_date: '1949'
},
actors: [
  {
    _id: 'artist:27',
    first_name: 'Bruce',
    last_name: 'Willis',
    birth_date: '1955',
    role: 'McClane'
  },
  {
    _id: 'artist:38',
    first_name: 'Samuel L.',
    last_name: 'Jackson',
    birth_date: '1948',
    role: 'Zeus Carver'
  },
  {
    _id: 'artist:169',
    first_name: 'Bridget',
    last_name: 'Fonda',
    birth_date: '1964',
    role: 'Simon Gruber'
  }
]
Type "it" for more
sandeep> db.movie.find({year: { $gte: 1990,$lt: 2000}}).count()
24
sandeep> |
```

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

I used find command with set **db.movie.find({\$or: [{date: {\$lt: 2000}}, {date: {\$gte: 2011}}]})**

Along with it I used count to check that how many movies are there.



The screenshot shows a terminal window titled "mongosh mongodb://127.0.0.1:27017" with the following content:

```
2 | sandeep> db.movie.find({$or: [{date: {$lt: 2000}}, {date: {$gte: 2011}}]})  
sandeep> db.movie.find({$or: [{date: {$lt: 2000}}, {date: {$gte: 2011}}]}).count()  
0  
sandeep> |
```

The terminal window is running on a Windows operating system, as indicated by the taskbar icons at the bottom.

## 8 Fetch all the movie distinct title.

I had to fetch all movie with distinct title. I had to use **db.movie.distinct("title")** where db is my database and movie is a collection to find distinct title I used filter as shown in figure.

```
sandeep> db.movie.distinct("title")
[
  '58 minutes pour vivre',
  'Alien',
  'American Beauty',
  'Bad Lieutenant',
  'Batman begins',
  'Blade Runner',
  'Casino',
  'Casino Royale',
  'De bruit et de fureur',
  'Django unchained',
  'Eyes Wide Shut',
  'Fargo',
  'Fenêtre sur cour',
  'Gladiator',
  'Godzilla',
  'Heat',
  'Impitoyable',
  'Inception',
  'Inglourious Basterds',
  'Interstellar',
  'Jackie Brown',
  "Jeanne d'Arc",
  'Kagemusha',
  'Kill Bill',
  'King of New York',
  'La mort aux trousses',
  'Le bon, la brute et le truand',
  'Le cinquième élément',
  'Le dernier métro',
  'Le gendarme et les extra-terrestres',
  'Le grand bleu',
  'Le last_name de la rose',
  'Le monde perdu',
  'Le parrain',
  'Le parrain II',
  'Le parrain III',
  'Le retour du Jedi',
  'Le silence des agneaux',
]

12:07 PM 2023-04-01
```

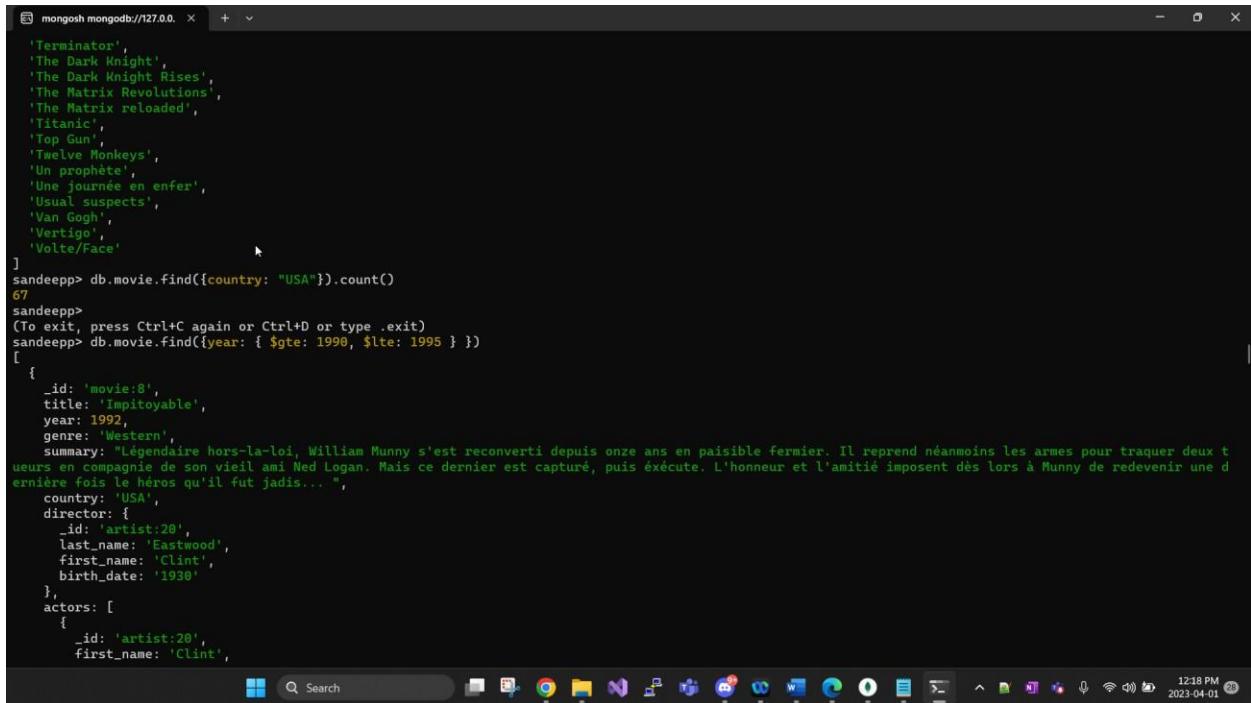
## 9. Count the number of movies released in USA.

In 9<sup>th</sup> question I had to count there how many movies which are released in USA then I used a command to fetch the number of movies released in USA like `db.movie.find({country: "USA"})` as shown in figure.

```
Mission: Impossible',
'Nikita',
'No country for old men',
'Nous trois ou rien',
'Pas de printemps pour Marnie',
'Piège de cristal',
'Pour quelques dollars de plus',
'Psychose',
'Pulp fiction',
'Rain Man',
'Reervoir dogs',
'Rio Grande',
'Sacrifice',
'Seven',
'Shining',
'Sixième sens',
'Skyfall',
'Sleepy Hollow',
'Soleil vert',
'Spider-Man',
'Stalingrad',
'Taxi driver',
'Terminator',
'The Dark Knight',
'The Dark Knight Rises',
'The Matrix Revolutions',
'The Matrix Reloaded',
'Titanic',
'Top Gun',
'Twelve Monkeys',
'Un prophète',
'Une journée en enfer',
'Usual suspects',
'Van Gogh',
'Vertigo',
'Volte/Face'
]
sandeep> db.movie.find({country: "USA"}).count()
67
sandeep>
```

## 10. Count the movie released between year 1990 to 1995.

I had to fetch year which are released in 1990 to 1995 then I used this command `db.movie.find({year: { $gte: 1990, $lte: 1995 } }).count()`. gte stands for greater than equal and less than equal.



```
mongosh mongodb://127.0.0.1:27017
[1] sandeep> db.movie.find().count()
67
sandeep>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
sandeep> db.movie.find({year: { $gte: 1990, $lte: 1995 } })
[
  {
    _id: 'movie:8',
    title: 'Impitoyable',
    year: 1992,
    genre: 'Western',
    summary: "Légendaire hors-la-loi, William Munny s'est reconverti depuis onze ans en paisible fermier. Il reprend néanmoins les armes pour traquer deux tueurs en compagnie de son vieux ami Ned Logan. Mais ce dernier est capturé, puis exécuté. L'honneur et l'amitié imposent dès lors à Munny de redevenir une dernière fois le héros qu'il fut jadis... ",
    country: 'USA',
    director: {
      _id: 'artist:20',
      last_name: 'Eastwood',
      first_name: 'Clint',
      birth_date: '1930'
    },
    actors: [
      {
        _id: 'artist:20',
        first_name: 'Clint',
        last_name: 'Eastwood'
      }
    ]
  }
]
```

`db.movie.find({year: { $gte: 1990, $lte: 1995 } }).count()` I used same command

```
| mongosh mongodb://127.0.0.1:27017 | + - X
| year: 1995,
|   genre: 'crime',
|   summary: 'En 1973, Sam Ace Rothstein est le grand manitou de la ville de toutes les folies, Las Vegas. Il achète et épouse une virtuose de l'arnaque, Ginger Mc Kenna, qui sombre bien vite dans l'alcool et la drogue. Mais un autre ennui guette Sam, son ami d'enfance Nicky Santoro, qui entreprend de mettre la ville en coupe réglée..',
|   country: 'USA',
|   director: {
|     _id: 'artist:241',
|     last_name: 'Scorsese',
|     first_name: 'Martin',
|     birth_date: '1962'
|   },
|   actors: [
|     {
|       _id: 'artist:167',
|       first_name: 'Robert',
|       last_name: 'De Niro',
|       birth_date: '1943',
|       role: "Sam 'Ace' Rothstein"
|     },
|     {
|       _id: 'artist:243',
|       first_name: 'Joe',
|       last_name: 'Pesci',
|       birth_date: '1943',
|       role: 'Nicky Santoro'
|     },
|     {
|       _id: 'artist:245',
|       first_name: 'Sharon',
|       last_name: 'Stone',
|       birth_date: '1958',
|       role: 'Ginger McKenna'
|     }
|   ]
| ] sandeep> db.movie.find({year: { $gte: 1990, $lte: 1995 } }).count()
18
sandeep> |
```

## 11. Fetch all the country where movies released.

I had to fetch all country where movies have been released then we can use distinct command which I have used and shown in figure #####

**db.movie.distinct("country")**

```
| mongosh mongodb://127.0.0.1:27017 - + x
|   role: 'Ginger McKenna'
| }
| ]
| sandeep> db.movie.find({year: { $gte: 1990, $lte: 1995 } }).count()
18
sandeep> db.movie.find({country: {title: 1}})
ReferenceError: country is not defined
sandeep> db.movie.find({country: {title: 1}})
ReferenceError: country is not defined
sandeep>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
sandeep> db.movie.distinct("country")
[ 'DE', 'FR', 'IT', 'JP', 'USA' ]
sandeep>
sandeep> db.movie.distinct("country")
[ 'DE', 'FR', 'IT', 'JP', 'USA' ]
sandeep>
```

## 12. Fetch all the document with movies titled as “Gladiator”.

I used find command to **db.movie.find({ title: "Gladiator" })** I had to fetch movie title Gladiator. There is only one movie which has title Gladiator.

```
| mongosh mongodb://127.0.0.1:27017 | + - x
sandeep>
sandeep> db.movie.distinct("country")
[ 'DE', 'FR', 'IT', 'JP', 'USA' ]
sandeep> db.movie.find({ title: "Gladiator" })
[
  {
    _id: 'movie:9',
    title: 'Gladiator',
    year: 2000,
    genre: 'drama',
    summary: "Le général romain Maximus est le plus fidèle soutien de l'empereur Marc Aurèle, qu'il a conduit de victoire avec une bravoure et un dévouement exemplaires. Jaloux du prestige de Maximus, et plus encore de l'amour que lui voulait l'empereur, le fils de Marc-Aurèle, Commode, s'arrogue brutalement le pouvoir, puis ordonne l'arrestation du général et son exécution. Maximus échappe à ses assassins mais ne peut empêcher le massacre de sa famille. Capturé par un marchand d'esclaves, il devient gladiateur et prépare sa vengeance.",
    country: 'USA',
    director: {
      _id: 'artist:4',
      last_name: 'Scott',
      first_name: 'Ridley',
      birth_date: '1937'
    },
    actors: [
      {
        _id: 'artist:23',
        first_name: 'Russell',
        last_name: 'Crowe',
        birth_date: '1964',
        role: 'Maximus'
      },
      {
        _id: 'artist:107',
        first_name: 'Adam',
        last_name: 'Baldwin',
        birth_date: '1962',
        role: 'Commode'
      },
      {
        _id: 'artist:148',
        first_name: 'Monica',
        last_name: 'Bellucci',
        birth_date: '1964',
        role: 'Julia'
      }
    ]
  }
]
```

### 13. Fetch distinct genre values of movies.

I had to fetch only genre where I used #####db.movie.distinct("genre") where I found all distinct genre.

```
| mongosh mongodb://127.0.0.1:27017 | + - x
|   last_name: 'Crowe',
|   birth_date: '1964',
|   role: 'Maximus'
|,
|   {
|     _id: 'artist:147',
|     first_name: 'Adam',
|     last_name: 'Baldwin',
|     birth_date: '1962',
|     role: 'Commodore'
|,
|   },
|   {
|     _id: 'artist:148',
|     first_name: 'Ryan',
|     last_name: 'ONeal',
|     birth_date: '1941',
|     role: 'Lucilla'
|,
|   },
|   {
|     _id: 'artist:149',
|     first_name: 'Marisa',
|     last_name: 'Berenson',
|     birth_date: '1946',
|     role: 'Marc Aurele'
|
|   }
|
| ]
sandeep> db.movie.find({ title: "Gladiator" }).count()
1
sandeep> db.movie.distinct("genre")
[
  'Action',      'Comédie',
  'Fantastique', 'Guerre',
  'Horreur',     'Science-fiction',
  'Suspense',    'Thriller',
  'Western',     'crime',
  'drama',       'romance'
]
sandeep>
```

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

To fetch all the documents with movies having the “crime” or “drama” genre from the movies collection, we can use the `find()` method in MongoDB with the `$in` operator to match documents where the genre field is either “crime” or “drama”.  
`db.movies.find({ genre: { $in: ["crime", "drama"] } })`

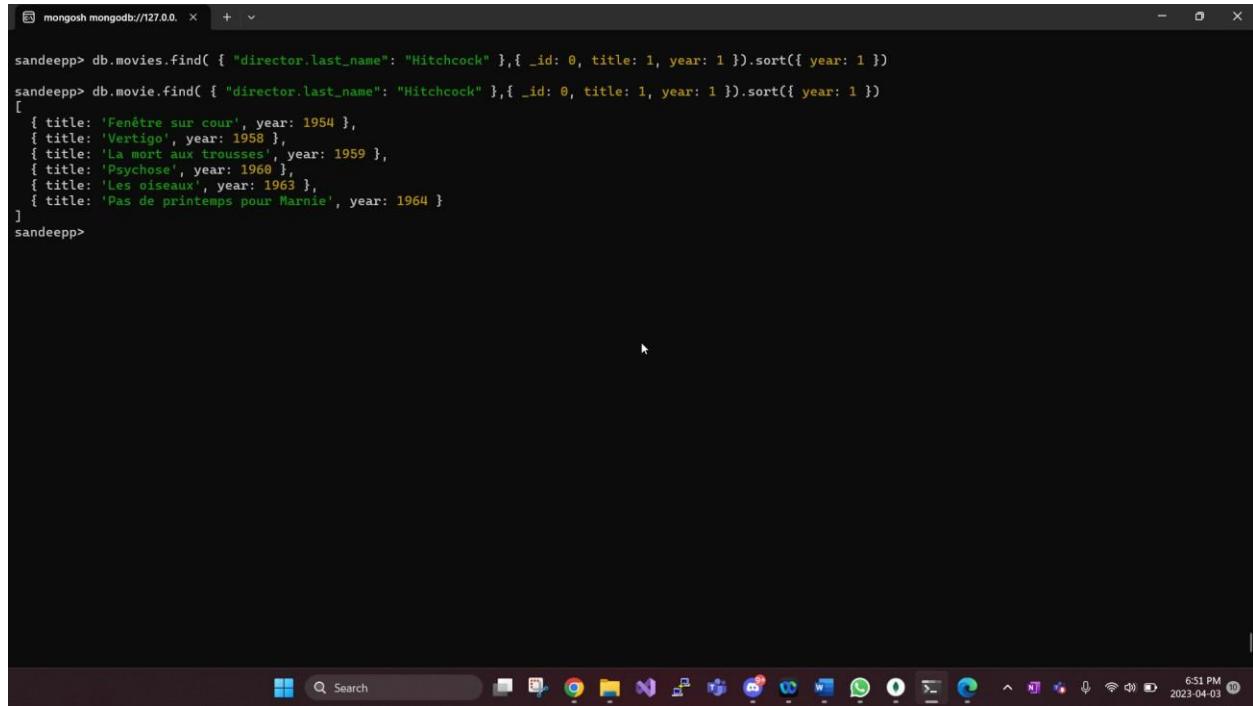
```
| mongosh mongoDB://127.0.0.1:27017 | + - x
| sandeep@DESKTOP-1QH6V9A: ~ |
| 'Suspense',      'Thriller',
| 'Western',       'crime',
| 'drama',         'romance'
|
| sandeep@DESKTOP-1QH6V9A: ~ |
| db.movie.find({ genre: { $in: ["crime", "drama"] } })
| [
|   {
|     _id: 'movie:1',
|     title: '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, ayant des tendances suicidaires. Amoureux de la jeune femme Scottie ne remarque pas le piège qui se trame autour de lui et dont il va être la victime... ",
|     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',
|       }
|     ]
|   }
| ]
|
| sandeep@DESKTOP-1QH6V9A: ~ |
| 12:49 PM 2023-04-01
```

I just used count command to get that how many movies are with these genres.

```
| mongosh mongodb://127.0.0.1:27017 | + - x
|   _id: 'artist:40',
|     first_name: 'Uma',
|     last_name: 'Thurman',
|     birth_date: '1970',
|     role: 'La mariée, alias "Black Mamba"'
|   },
|   {
|     _id: 'artist:213',
|     first_name: 'Lucy',
|     last_name: 'Liu',
|     birth_date: '1968',
|     role: 'O-Ren Ishii'
|   },
|   {
|     _id: 'artist:214',
|     first_name: 'David',
|     last_name: 'Carradine',
|     birth_date: '1936',
|     role: 'Bill'
|   },
|   {
|     _id: 'artist:215',
|     first_name: 'Michael',
|     last_name: 'Madsen',
|     birth_date: '1958',
|     role: 'Budd / Sidewinder'
|   },
|   {
|     _id: 'artist:216',
|     first_name: 'Daryl',
|     last_name: 'Hannah',
|     birth_date: '1960',
|     role: 'Elle Driver'
|   }
| ]
| Type "it" for more
sandeep> db.movie.find({ genre: { $in: ["crime", "drama"] } }).count()
33
12:45 PM 2023-04-01
```

## 15. Fetch all the movies directed by "Hitchcock", display only title and year and sort them by year.

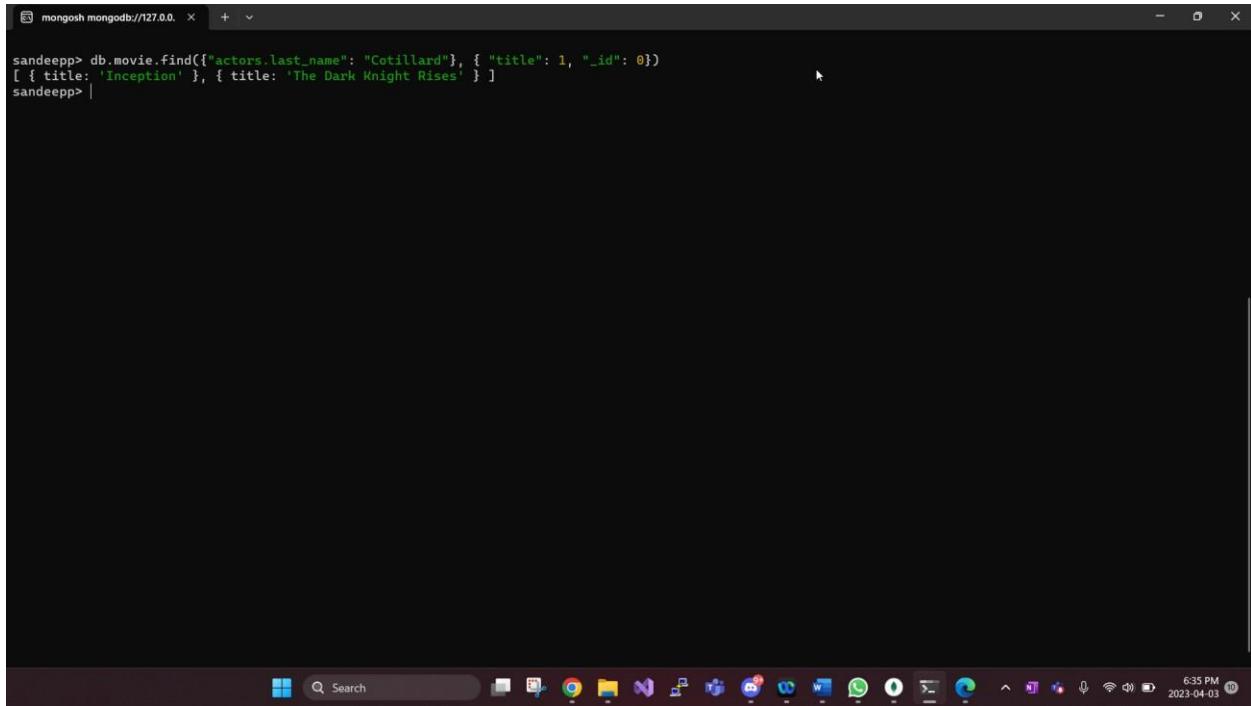
I used the `find()` method to query the movies collection for movies directed by "Hitchcock". I specified the filter `{ "director.last_name": "Hitchcock" }` to match movies directed by "Hitchcock". `db.movies.find( { "director.last_name": "Hitchcock" },{ _id: 0, title: 1, year: 1 }).sort({ year: 1 })`



```
sandeep@DESKTOP-6V9D9CJ: ~ % mongosh mongodb://127.0.0.1:27017
MongoDB shell version v5.0.12
connecting to: mongodb://127.0.0.1:27017/
Implicit connection to admin database
MongoDB server version: 5.0.12
WARNING: This session is not connected to a database.
sandeep> db.movies.find( { "director.last_name": "Hitchcock" }, { _id: 0, title: 1, year: 1 } ).sort({ year: 1 })
sandeep> db.movie.find( { "director.last_name": "Hitchcock" }, { _id: 0, title: 1, year: 1 } ).sort({ year: 1 })
[
  { title: 'Fenêtre sur cour', year: 1954 },
  { title: 'Vertigo', year: 1958 },
  { title: 'La mort aux trousses', year: 1959 },
  { title: 'Psychose', year: 1960 },
  { title: 'Les oiseaux', year: 1963 },
  { title: 'Pas de printemps pour Marnie', year: 1964 }
]
sandeep>
```

## 16. Fetch the list of movies where "Cotillard" played.

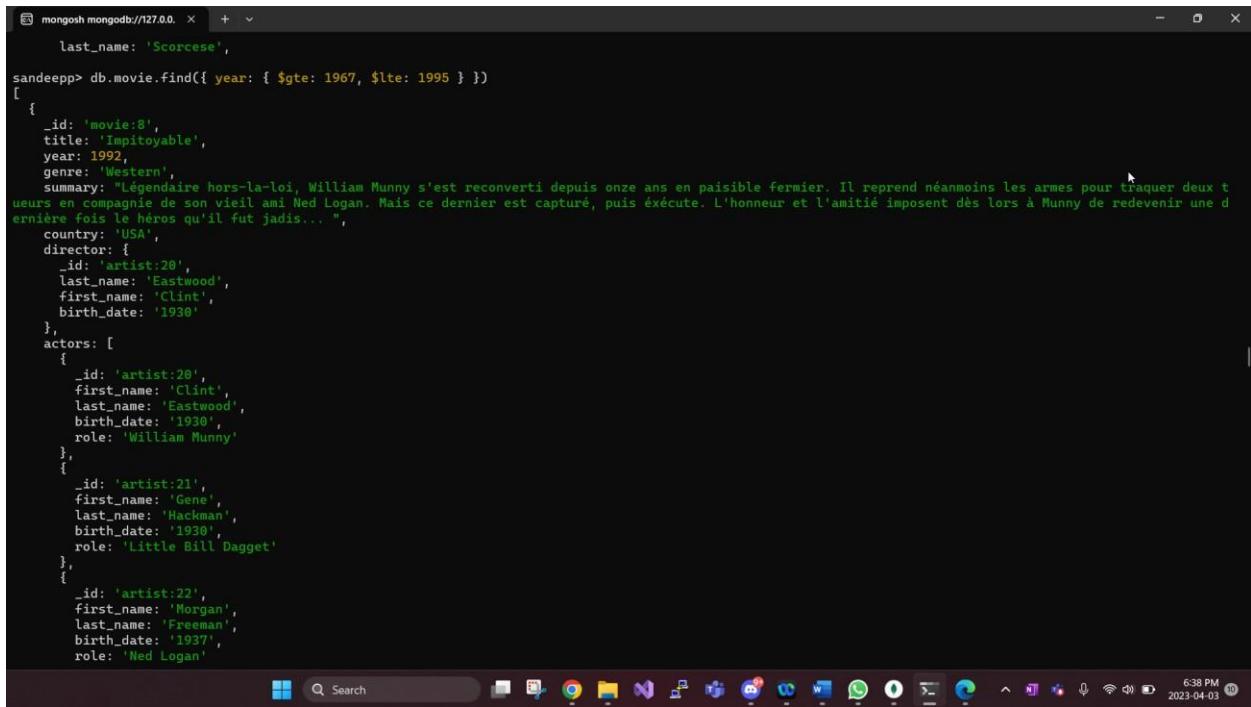
Here I have used find command to fetch the movie where I needed to find and I used **db.movie.find({"actors.last\_name": "Cotillard"}, {"title": 1, "\_id": 0})**



```
sandeep@DESKTOP-1D6C9F5: ~ % mongosh mongodb://127.0.0.1:27017
MongoDB shell version: 4.4.15
connecting to: mongodb://127.0.0.1:27017/test
Implicit connection to admin database
MongoDB server version: 4.4.15
WARNING: This session is not connected to a database.
sandeep@DESKTOP-1D6C9F5: ~ % db.movie.find({ "actors.last_name": "Cotillard" }, { "title": 1, "_id": 0 })
[ { title: 'Inception' }, { title: 'The Dark Knight Rises' } ]
```

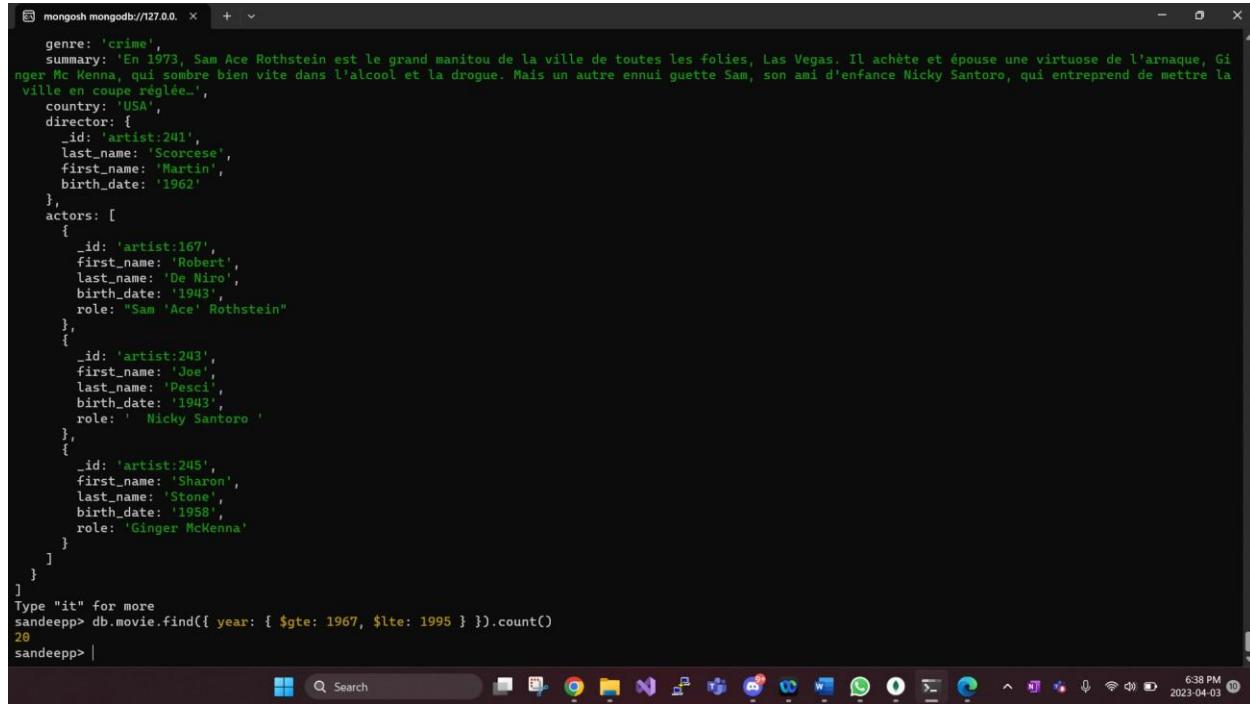
## 17. Fetch the list of movies released between 1967 and 1995.

In this to fetch the year when movie released between 1967 and 1995 and used find command with this function **db.movie.find({year : { \$gte: 1967, \$lte: 1995}}**



```
sandeep@DESKTOP-1D6C9F5: ~ % mongosh mongodb://127.0.0.1:27017
MongoDB shell version: 4.4.15
connecting to: mongodb://127.0.0.1:27017/test
Implicit connection to admin database
MongoDB server version: 4.4.15
WARNING: This session is not connected to a database.
sandeep@DESKTOP-1D6C9F5: ~ % db.movie.find({ year: { $gte: 1967, $lte: 1995 } })
[ {
    _id: 'movie:8',
    title: 'Unforgiven',
    year: 1992,
    genre: 'Western',
    summary: "Légendaire hors-la-loi, William Munny s'est reconvertis depuis onze ans en paisible fermier. Il reprend néanmoins les armes pour traquer deux tueurs en compagnie de son vieil ami Ned Logan. Mais ce dernier est capturé, puis exécuté. L'honneur et l'amitié imposent dès lors à Munny de redevenir une dernière fois le héros qu'il fut jadis...",
    country: 'USA',
    director: {
        _id: 'artist:20',
        last_name: 'Eastwood',
        first_name: 'Clint',
        birth_date: '1930'
    },
    actors: [
        {
            _id: 'artist:20',
            first_name: 'Clint',
            last_name: 'Eastwood',
            birth_date: '1930',
            role: 'William Munny'
        },
        {
            _id: 'artist:21',
            first_name: 'Gene',
            last_name: 'Hackman',
            birth_date: '1930',
            role: 'Little Bill Daggett'
        },
        {
            _id: 'artist:22',
            first_name: 'Morgan',
            last_name: 'Freeman',
            birth_date: '1937',
            role: 'Ned Logan'
        }
    ]
}]
```

Here I used count to check the number .

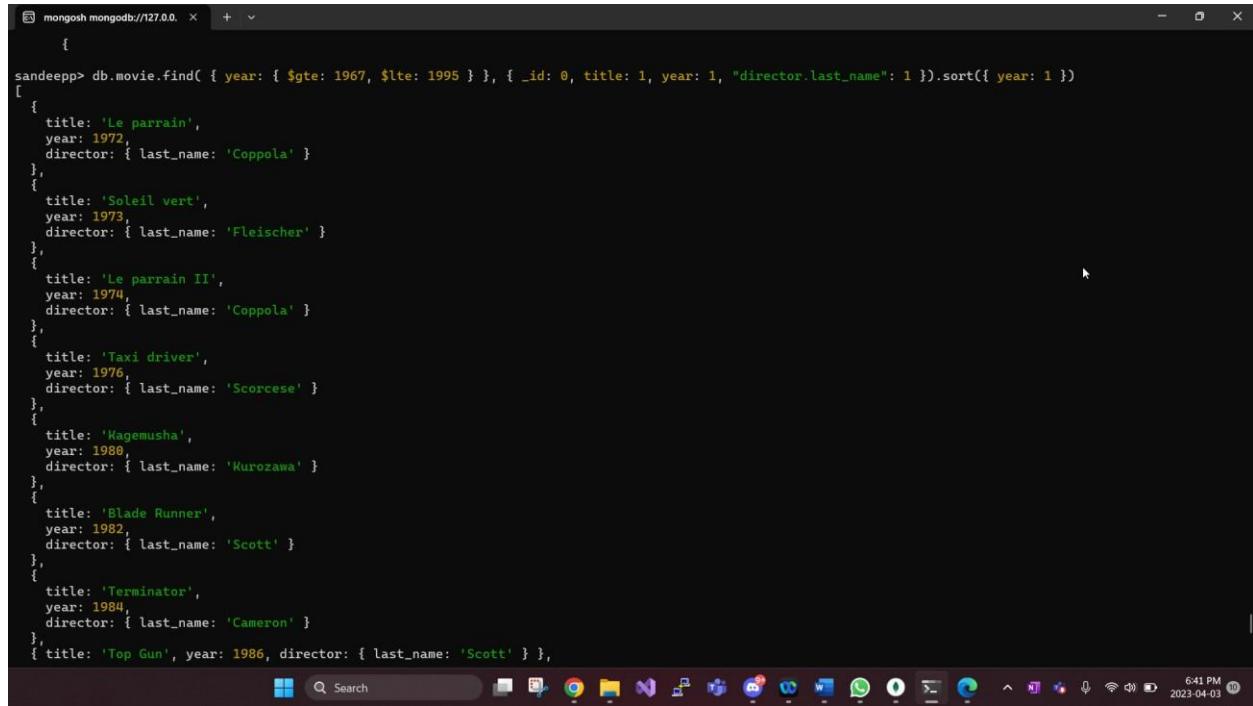


```
mongosh mongodb://127.0.0.1:27017
{
  genre: 'crime',
  summary: 'En 1973, Sam Ace Rothstein est le grand manitou de la ville de toutes les folies, Las Vegas. Il achète et épouse une virtuose de l'arnaque, Ginger Mc Kenna, qui sombre bien vite dans l'alcool et la drogue. Mais un autre ennui guette Sam, son ami d'enfance Nicky Santoro, qui entreprend de mettre la ville en coupe réglée..',
  country: 'USA',
  director: {
    _id: 'artist:241',
    last_name: 'Scorsese',
    first_name: 'Martin',
    birth_date: '1962'
  },
  actors: [
    {
      _id: 'artist:167',
      first_name: 'Robert',
      last_name: 'De Niro',
      birth_date: '1943',
      role: "Sam 'Ace' Rothstein"
    },
    {
      _id: 'artist:243',
      first_name: 'Joe',
      last_name: 'Pesci',
      birth_date: '1943',
      role: "Nicky Santoro"
    },
    {
      _id: 'artist:245',
      first_name: 'Sharon',
      last_name: 'Stone',
      birth_date: '1958',
      role: "Ginger McKenna"
    }
  ]
}
Type "it" for more
sandeep> db.movie.find({ year: { $gte: 1967, $lte: 1995 } }).count()
20
sandeep> |
```

## 18. Fetch the list of movies released between 1967 and 1995, by displaying only title, year, director's last name sorted by year.

I used the find() method to query the movies collection for movies with a year between 1967 and 1995 and used the \$gte and \$lte operators to specify the range of years.

```
db.movie.find( { year: { $gte: 1967, $lte: 1995 } }, { _id: 0, title: 1, year: 1,
  "director.last_name": 1 }).sort({ year: 1 })
```



```
sandeep@DESKTOP-6V9D9C9: ~ % mongosh mongodb://127.0.0.1
MongoDB shell version: 4.4.15
connecting to: mongodb://127.0.0.1:27017/
Implicit connection to admin database
MongoDB shell version: 4.4.15
Implicit connection to admin database
Welcome to the MongoDB shell.
For help enter ? or help()
For more info on this server see /proc/meminfo
For help, type ? or help(command)
db.movie.find( { year: { $gte: 1967, $lte: 1995 } }, { _id: 0, title: 1, year: 1, "director.last_name": 1 }).sort({ year: 1 })
[{"title": "Le parrain", "year": 1972, "director": {"last_name": "Coppola"}}, {"title": "Soleil vert", "year": 1973, "director": {"last_name": "Fleischer"}}, {"title": "Le parrain II", "year": 1974, "director": {"last_name": "Coppola"}}, {"title": "Taxi driver", "year": 1976, "director": {"last_name": "Scorcese"}}, {"title": "Kagemusha", "year": 1980, "director": {"last_name": "Kurozawa"}}, {"title": "Blade Runner", "year": 1982, "director": {"last_name": "Scott"}}, {"title": "Terminator", "year": 1984, "director": {"last_name": "Cameron"}}, {"title": "Top Gun", "year": 1986, "director": {"last_name": "Scott"}}, 641 PM 2023-04-03
```

## 19. Fetch the number of movies by country.

In this we find number of movies by country using find command and used

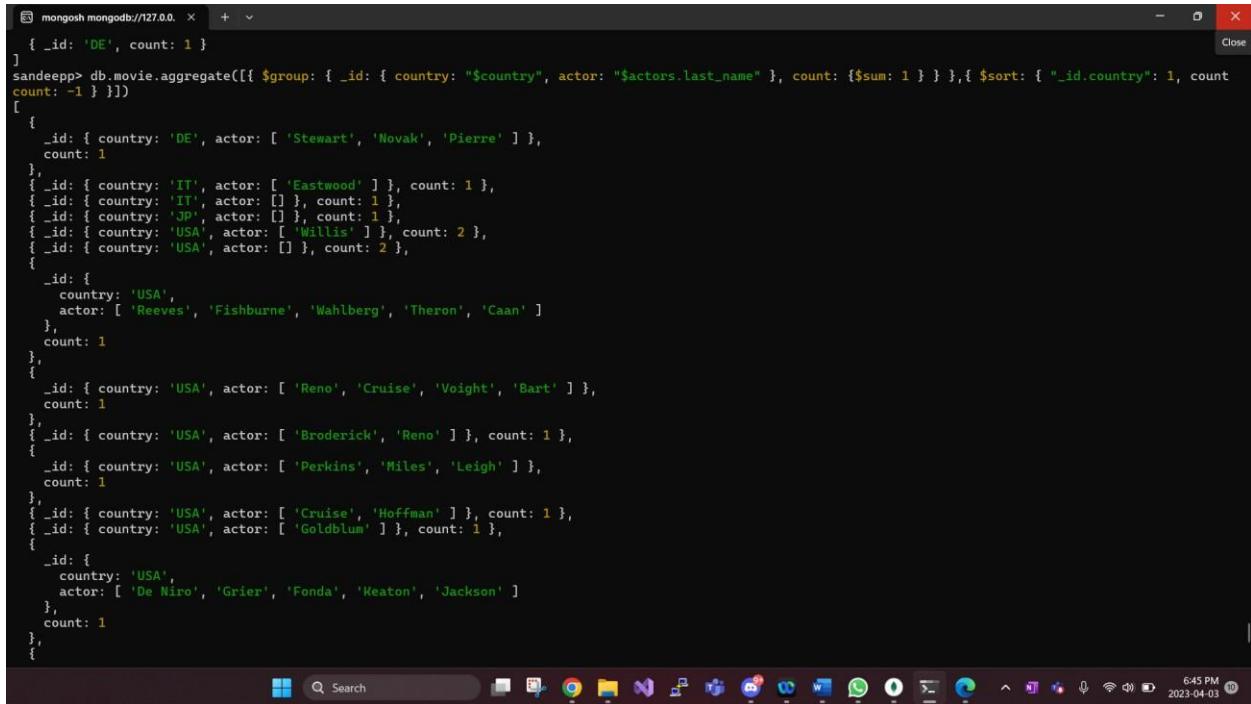
```
db.movie.aggregate([{ $group: { _id: "$country", count: { $sum: 1 } } },{ $sort: { count: -1 } }])
```

```
sandeep@DESKTOP-6V9D9CJ: ~ % mongosh mongodb://127.0.0.1:27017
MongoDB shell version v5.0.15
connecting to: mongodb://127.0.0.1:27017/
Implicit session: ticket: 0
MongoDB shell version v5.0.15
Implicit session: ticket: 0
[sandbox]
> db.movie.aggregate([{$group: {_id: '$country', count: {$sum: 1}}, {$sort: {count: -1}}}])
[{"_id": "USA", "count": 59}, {"_id": "IT", "count": 2}, {"_id": "JP", "count": 1}, {"_id": "DE", "count": 1}]
sandeep@DESKTOP-6V9D9CJ: ~ %
```

## 20. Fetch the number of movies by country and actor

As in previous question we used find command to fetch now we have to fetch number of movies by country and actor

```
db.movie.aggregate([{$group: {_id: {country: "$country", actor: "$actors.last_name"}, count: {$sum: 1}}}, {$sort: {"_id.country": 1, count: -1}}])
```



```
mongosh mongodb://127.0.0.1:27017
{
  _id: 'DE',
  count: 1
}
[sandeep@localhost:27017 movie]$ db.movie.aggregate([
  {
    $group: {
      _id: { country: "$country", actor: "$actors.last_name" },
      count: { $sum: 1 }
    }
  },
  {
    $sort: { "_id.country": 1, "count": -1 }
  }
])
[{"_id": {"country": "DE", "actor": "Novak"}, "count": 1}, {"_id": {"country": "IT", "actor": "Eastwood"}, "count": 1}, {"_id": {"country": "IT", "actor": "Pierre"}, "count": 1}, {"_id": {"country": "JP", "actor": "Novak"}, "count": 1}, {"_id": {"country": "USA", "actor": "Willis"}, "count": 2}, {"_id": {"country": "USA", "actor": "Pierre"}, "count": 2}, {"_id": {"country": "USA", "actor": "Reeves"}, "count": 1}, {"_id": {"country": "USA", "actor": "Fishburne"}, "count": 1}, {"_id": {"country": "USA", "actor": "Wahlberg"}, "count": 1}, {"_id": {"country": "USA", "actor": "Theron"}, "count": 1}, {"_id": {"country": "USA", "actor": "Caan"}, "count": 1}, {"_id": {"country": "USA", "actor": "Reno"}, "count": 1}, {"_id": {"country": "USA", "actor": "Cruise"}, "count": 1}, {"_id": {"country": "USA", "actor": "Voight"}, "count": 1}, {"_id": {"country": "USA", "actor": "Bart"}, "count": 1}, {"_id": {"country": "USA", "actor": "Broderick"}, "count": 1}, {"_id": {"country": "USA", "actor": "Reno"}, "count": 1}, {"_id": {"country": "USA", "actor": "Perkins"}, "count": 1}, {"_id": {"country": "USA", "actor": "Miles"}, "count": 1}, {"_id": {"country": "USA", "actor": "Leigh"}, "count": 1}, {"_id": {"country": "USA", "actor": "Cruise"}, "count": 1}, {"_id": {"country": "USA", "actor": "Hoffman"}, "count": 1}, {"_id": {"country": "USA", "actor": "Goldblum"}, "count": 1}, {"_id": {"country": "USA", "actor": "De Niro"}, "count": 1}, {"_id": {"country": "USA", "actor": "Grier"}, "count": 1}, {"_id": {"country": "USA", "actor": "Fonda"}, "count": 1}, {"_id": {"country": "USA", "actor": "Keaton"}, "count": 1}, {"_id": {"country": "USA", "actor": "Jackson"}, "count": 1}]

6:45 PM 2023-04-03
```

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`”.

I have a movie script which is a collection of Sandeep database and I used update command and filtered it which is `db.movie.updateOne({ _id: "movie:3", title: "Titanic" }, { $set: { title: "The Titanic" } })`. For more explanation I just wanted to update one id that's why I used updateOne where I told the title of movie and used operation to update it as shown in figure.

```
mongosh mongodb://127.0.0.1:27017
{
  birth_date: '1946'
},
actors: [
  {
    _id: 'artist:46',
    first_name: 'Roy',
    last_name: 'Scheider',
    birth_date: '1932',
    role: 'Martin Brody'
  },
  {
    _id: 'artist:47',
    first_name: 'Robert',
    last_name: 'Shaw',
    birth_date: '1927',
    role: 'Quint'
  },
  {
    _id: 'artist:48',
    first_name: 'Richard',
    last_name: 'Dreyfus',
    birth_date: '1947',
    role: 'Matt Hooper'
  }
]
}
Type "it" for more
sandeep> db.movie.updateOne({_id: "movie:3", title: "Titanic"}, {$set: {title: "The Titanic"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
12:00 PM 2023-03-28
```

```
mongosh mongodb://127.0.0.1:27017
2 |
sandeep> db.movie.find({_id: "movie:3"})
[
  {
    _id: 'movie:3',
    title: 'The 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 question, Rose DeWitt Bukater, aujourd'hui centenaire, rejoint les lieux du naufrage, d'où elle entreprend de raconter le récit de son fascinant, étrange et tragique voyage...",
    country: 'USA',
    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'
      }
    ]
}
12:00 PM 2023-03-28
```

## 2. Update the movie with genre “The Comedy” from "Comédie" where `_id` is “`movie:7`”.

I had to update Comédie to The comedy in movie: 7 for that

```

mongosh mongodb://127.0.0.1:27017
sandeep> db.movie.find({_id: "movie:7"})
[
  {
    _id: 'movie:7',
    title: 'American Beauty',
    year: 1999,
    genre: 'Comédie',
    summary: 'Lester Burnham, sa femme Carolyn et leur fille Jane mènent apparemment une vie des plus heureuses dans leur belle banlieue. Mais derrière cette respectable façade se tisse une étrange et grinçante tragî-comédie familiale où désirs inavoués, frustrations et violences refoulées conduiront inexorablement un homme vers la mort. ',
    country: 'USA',
    director: {
      _id: 'artist:17',
      last_name: 'Mendes',
      first_name: 'Sam',
      birth_date: '1965'
    },
    actors: [
      {
        _id: 'artist:18',
        first_name: 'Kevin',
        last_name: 'Spacey',
        birth_date: '1959',
        role: 'Lester Burnham'
      },
      {
        _id: 'artist:19',
        first_name: 'Annette',
        last_name: 'Bening',
        birth_date: '1958',
        role: 'Carolyn Burnham'
      }
    ]
  }
]

```

I used update command `db.movie.updateOne({_id: "movie:7", genre: "Comédie"}, {$set:{genre: "The comedy"}})` as I shown in figure.

```

mongosh mongodb://127.0.0.1:27017
sandeep> db.movie.updateOne({_id: "movie:7", genre: "Comédie"}, {$set: {genre: "The Comedy"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
sandeep> db.movie.updateOne({_id: "movie:7", genre: "Comédie"}, {$set: {genre: "The Comedy"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sandeep> db.movie.updateOne({_id: "movie:7"})
MongoInvalidArgumentError: Update document requires atomic operators
sandeep> db.movie.find({_id: "movie:7"})
[
  {
    _id: 'movie:7',
    title: 'American Beauty',
    year: 1999,
    genre: 'The Comedy',
    summary: 'Lester Burnham, sa femme Carolyn et leur fille Jane mènent apparemment une vie des plus heureuses dans leur belle banlieue. Mais derrière cette respectable façade se tisse une étrange et grinçante tragî-comédie familiale où désirs inavoués, frustrations et violences refoulées conduiront inexorablement un homme vers la mort. '
  }
]

```

**3 Update the movie with year 1990 from 1999 where \_id is "movie:6"**  
In question 3<sup>rd</sup> I have to update the year of the movie id 6.

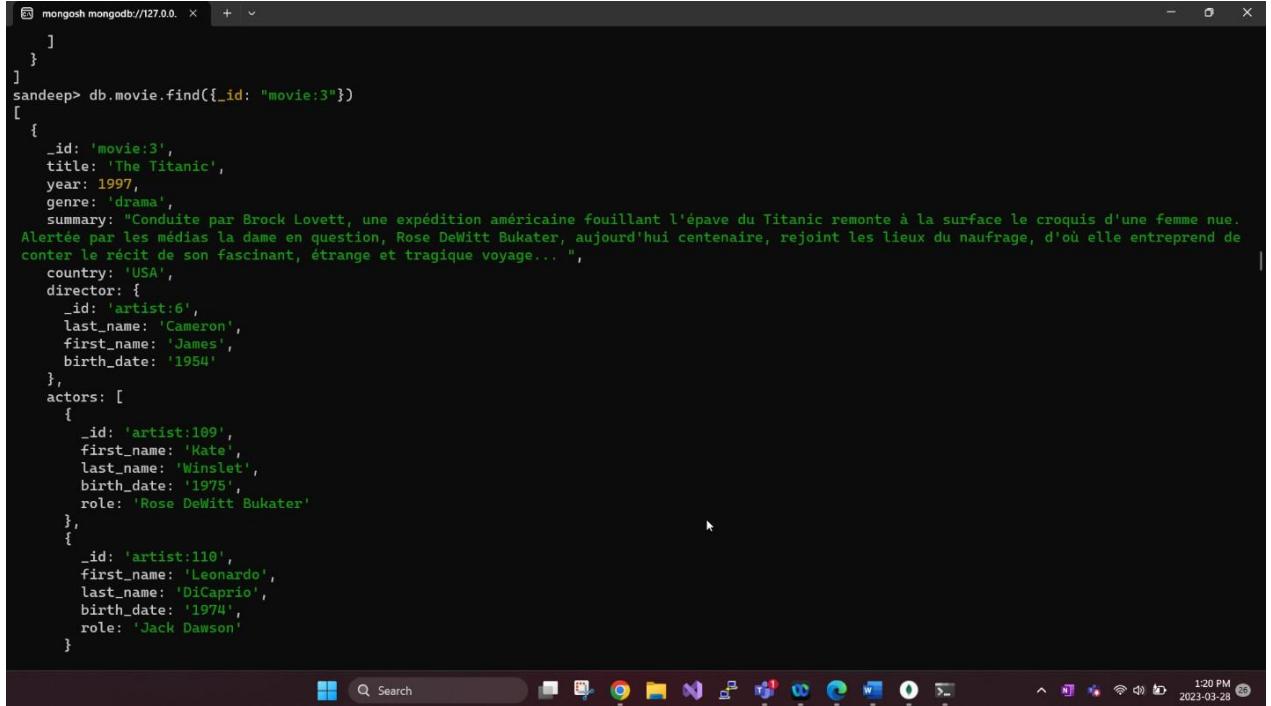
```
[mongosh mongodb://127.0.0.1:27017] sandeep> db.movie.find({_id: "movie:6"})
[
  {
    _id: 'movie:6',
    title: 'Sleepy Hollow',
    year: 1999,
    genre: 'Fantastique',
    summary: 'Nouvelle Angleterre, 1799. A Sleepy Hollow, plusieurs cadavres sont retrouvés décapités. La rumeur attribue ces meurtres à un cavalier lui-même sans tête. Mais le fin limier new-yorkais Ichabod Crane ne croit pas en ses aberrations. Tombé sous le charme de la vénérable Katrina, il mène son enquête au cœur des sortilèges de Sleepy Hollow..',
    country: 'USA',
    director: {
      _id: 'artist:13',
      last_name: 'Burton',
      first_name: 'Tim',
      birth_date: '1958'
    },
    actors: [
      {
        _id: 'artist:14',
        first_name: 'Johnny',
        last_name: 'Depp',
        birth_date: '1964',
        role: 'Constable Ichabod Crane'
      },
      {
        _id: 'artist:96',
        first_name: 'Christina',
        last_name: 'Ricci',
        birth_date: '1980',
        role: 'Katrina Anne Van Tassel'
      }
    ]
  }
]
1:20 PM 2023-03-28
```

I used command **db.movie.updateOne({\_id: “movie:6”, year: 1999} {\$set:{year: 1990}})** as shown in below figure. Db is a database and movie is a collection.

```
[mongosh mongodb://127.0.0.1:27017] sandeep> db.movie.updateOne({_id: "movie:6", year: 1999}, {$set: {year: 1990}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sandeep> db.movie.find({_id: "movie:6"})
[
  {
    _id: 'movie:6',
    title: 'Sleepy Hollow',
    year: 1990,
    genre: 'Fantastique',
    summary: 'Nouvelle Angleterre, 1799. A Sleepy Hollow, plusieurs cadavres sont retrouvés décapités. La rumeur attribue ces meurtres à un cavalier lui-même sans tête. Mais le fin limier new-yorkais Ichabod Crane ne croit pas en ses aberrations. Tombé sous le charme de la vénérable Katrina, il mène son enquête au cœur des sortilèges de Sleepy Hollow..',
    country: 'USA',
    director: {
      _id: 'artist:13',
      last_name: 'Burton',
      first_name: 'Tim',
      birth_date: '1958'
    },
    actors: [
      {
        _id: 'artist:14',
        first_name: 'Johnny',
        last_name: 'Depp',
        birth_date: '1964',
        role: 'Constable Ichabod Crane'
      },
      {
        _id: 'artist:96',
        first_name: 'Christina',
        last_name: 'Ricci',
        birth_date: '1980',
        role: 'Katrina Anne Van Tassel'
      }
    ]
  }
]
1:20 PM 2023-03-28
```

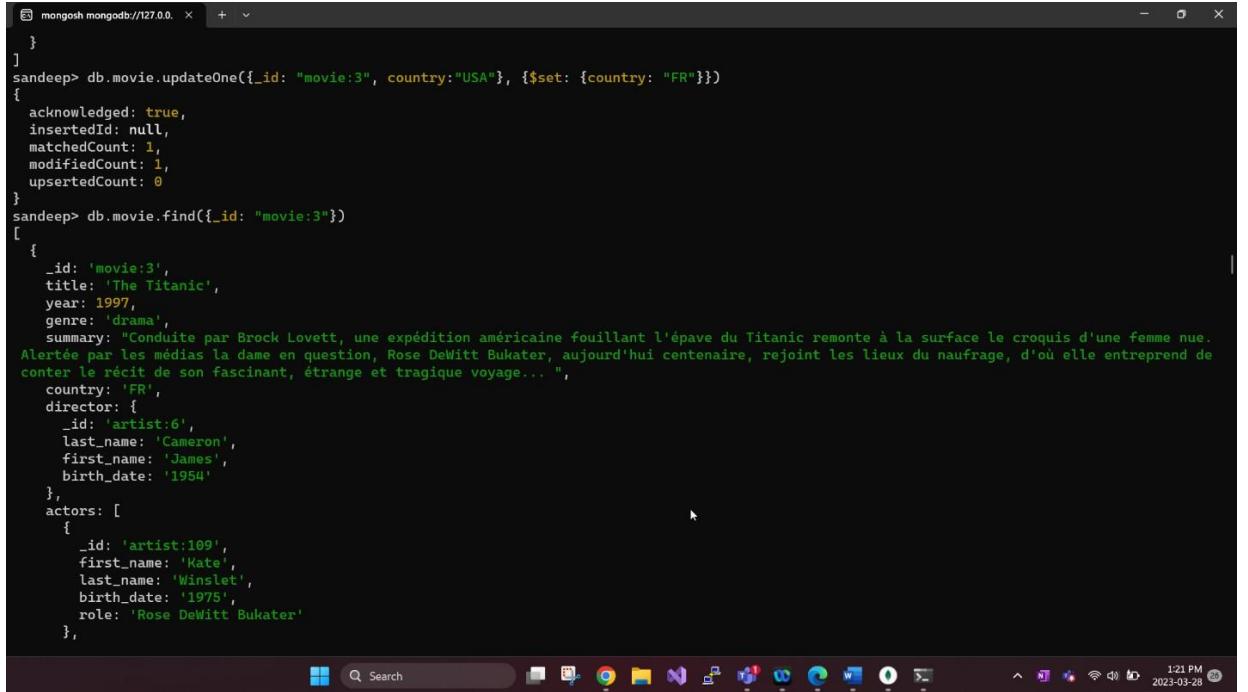
#### 4. Update the movie with country "FR" from "USA" where \_id is "movie:3".

In this we must update only one id of movie in which we have mentioned USA and update with FR as shown in figure.



```
[mongosh mongoDB://127.0.0.1:27017]
]
}
]
sandeep> db.movie.find({_id: "movie:3"})
[
{
  "_id": "movie:3",
  "title": "The 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 question, Rose DeWitt Bukater, aujourd'hui centenaire, rejoint les lieux du naufrage, d'où elle entreprend de raconter le récit de son fascinant, étrange et tragique voyage...",
  "country": "USA",
  "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"
    }
  ]
}
```

I used the command `db.movie.updateOne({_id: "movie:3", country: "USA"} {$set: {country: "FR"}})` as I shown in the below figure.



```
[mongosh mongoDB://127.0.0.1:27017]
]
}
]
sandeep> db.movie.updateOne({_id: "movie:3", country:"USA"}, {$set: {country: "FR"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sandeep> db.movie.find({_id: "movie:3"})
[
{
  "_id": "movie:3",
  "title": "The 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 question, Rose DeWitt Bukater, aujourd'hui centenaire, rejoint les lieux du naufrage, d'où elle entreprend de raconter le récit de son fascinant, étrange et tragique voyage...",
  "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"
    }
  ]
}
```

## 5 Update the movie with director's last\_name is "Doe" from "Scott" where \_id is "movie:9".

I have a id which has named \_id: "movie:9" there was only one updation which I have to do. I had directors's last name Scott which I have updated with Doe.

```
[{"_id": "movie:9", "title": "Gladiator", "year": 2000, "genre": "drama", "summary": "Le g\u00e9n\u00e9ral romain Maximus est le plus fid\u00eble soutien de l'empereur Marc Aur\u00e8le, qu'il a cond\u00e9n\u00e9 \u00e0 la mort pour son r\u00e9sistance au coup d'Etat de l'empereur Commodus. Maximus, bravo\u00e9 et un d\u00e9vouement exemplaires. Jaloux du prestige de Maximus, et plus encore de l'amour que lui voue l'empereur, le fils de Marc-Aur\u00e8le, Commodus, s'arrogue brutalme\u00e7t le pouvoir, puis ordonne l'arrestation du g\u00e9n\u00e9ral et son ex\u00e9cution. Maximus \u00e9chappe \u00e0 sa mort \u00e0 la derni\u00e8re minute, emp\u00e8che le massacre de sa famille. Captur\u00e9 par un marchand d'esclaves, il devient un gladiateur et pr\u00e9pare sa vengeance.", "country": "USA", "director": {"_id": "artist:4", "last_name": "Scott", "first_name": "Ridley", "birth_date": "1937"}, "actors": [{"_id": "artist:23", "first_name": "Russell", "last_name": "Crowe", "birth_date": "1964", "role": "Maximus"}, {"_id": "artist:147", "first_name": "Adam", "last_name": "Baldwin"}]}
```

In the above figure I showed the actual result but In upcoming figure I updated with **db.movie.updateOne({\_id: "movie:9", "director.last\_name": "Scott" } {\$set: {"director.last\_name": "Doe"}}**

```

        }
    ]
}
sandeep> db.movie.updateOne({_id: "movie:9", "director.last_name": "Scott"}, {$set: {"director.last_name": "Doe"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sandeep> db.movie.find({_id: "movie:9"})
[
  {
    _id: 'movie:9',
    title: 'Gladiator',
    year: 2000,
    genre: 'drama',
    summary: "Le général romain Maximus est le plus fidèle soutien de l'empereur Marc Aurèle, qu'il a cond
uit de victoire en victoire avec une bravoure et un dévouement exemplaires. Jaloux du prestige de Maximus,
et plus encore de l'amour que lui voue l'empereur, le fils de Marc-Aurèle, Commode, s'arrog
brutalement le pouvoir, puis ordonne l'arrestation du général et son exécution. Maximus échappe à s
es assassins mais ne peut empêcher le massacre de sa famille. Capturé par un marchand d'esclaves, il devie
nt gladiateur et prépare sa vengeance.",
    country: 'USA',
    director: {
      _id: 'artist:4',
      last_name: 'Doe',
      first_name: 'Ridley',
      birth_date: '1937'
    },
    actors: [
      {
        _id: 'artist:23',
    
```

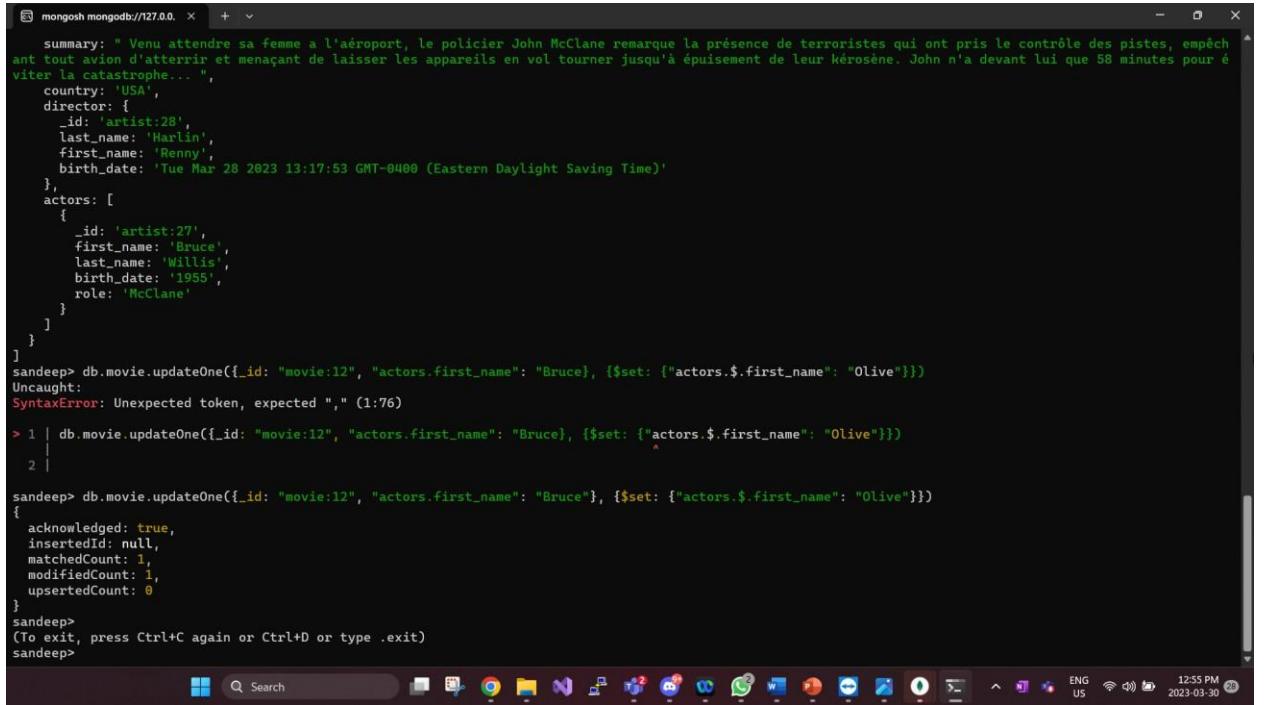
## 6 Update the movie with director's first\_name “Olive” from “Bruce” where \_id is "movie:12".

There was no name Bruce of director in movie: 12 what as I know there was a name bruce of actor which I changed from Bruce to Doe.

```

insertedId: null,
matchedCount: 1,
modifiedCount: 1,
upsertedCount: 0
}
sandeep> db.movie.find({_id: "movie:12"})
[
  {
    _id: 'movie:12',
    title: '58 minutes pour vivre',
    year: 1998,
    genre: 'Drama',
    summary: "Venu attendre sa femme à l'aéroport, le policier John McClane remarque la présence de terroristes qui ont pris le contrôle des pistes, empêchant tout avion d'atterrir et menaçant de laisser les appareils en vol tourner jusqu'à épuisement de leur kérosène. John n'a devant lui que 58 minutes pour éviter la catastrophe...",
    country: 'USA',
    director: {
      _id: 'artist:28',
      last_name: 'Harlin',
      first_name: 'Renny',
      birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
    },
    actors: [
      {
        _id: 'artist:27',
        first_name: 'Bruce',
        last_name: 'Willis',
        birth_date: '1955',
        role: 'McClane'
      }
    ]
}
sandeep> db.movie.updateOne({_id: "movie:12", "actors.first_name": "Bruce"}, {$set: ["actors.$first_name": "Olive"]})
Uncaught:
SyntaxError: Unexpected token, expected "," (1:76)
> 1 | db.movie.updateOne({_id: "movie:12", "actors.first_name": "Bruce"}, {$set: ["actors.$first_name": "Olive"]})
| 
2 | 
```

I changed the actors name from Bruce to Olive. I used command  
**db.movie.updateOne({\_id: “movie:12”, “actors.first\_name”: “Bruce”},  
{\$set: {“actors.\$first\_name”: “Olive”}})**



```
mongosh mongodb://127.0.0.1:27017
summary: " Venu attendre sa femme à l'aéroport, le policier John McClane remarque la présence de terroristes qui ont pris le contrôle des pistes, empêchant tout avion d'atterrir et menaçant de laisser les appareils en vol tourner jusqu'à épuisement de leur kérosène. John n'a devant lui que 58 minutes pour éviter la catastrophe... "
country: 'USA',
director: {
  _id: 'artist:28',
  last_name: 'Harlin',
  first_name: 'Renny',
  birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
},
actors: [
  {
    _id: 'artist:27',
    first_name: 'Bruce',
    last_name: 'Willis',
    birth_date: '1955',
    role: 'McClane'
  }
]
sandee> db.movie.updateOne({_id: "movie:12", "actors.first_name": "Bruce"}, {$set: {"actors.$first_name": "Olive"}})
Uncaught:
SyntaxError: Unexpected token, expected "," (1:76)
> 1 | db.movie.updateOne({_id: "movie:12", "actors.first_name": "Bruce"}, {$set: {"actors.$first_name": "Olive"}})
2 |
sandee> db.movie.updateOne({_id: "movie:12", "actors.first_name": "Bruce"}, {$set: {"actors.$first_name": "Olive"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sandee>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
sandee>
```

## 7 Update the movies with year 1990 where year is 1994.

I have had a movie script where I got 1994 and I had to change to 1990 as I shown in figure where I had only 2 id where I had 1994.

```

mongosh mongodb://127.0.0.1
}
]
sandeep> db.movie.find({year:1995}, {title:1,year:1} )
[
  { _id: 'movie:14', title: 'Seven', year: 1995 },
  { _id: 'movie:15', title: 'Twelve Monkeys', year: 1995 },
  { _id: 'movie:52', title: 'Usual suspects', year: 1995 },
  { _id: 'movie:58', title: 'Une journée en enfer', year: 1995 },
  { _id: 'movie:69', title: 'Heat', year: 1995 },
  { _id: 'movie:72', title: 'Casino', year: 1995 }
]
sandeep> db.movie.find({year:1995}, {title:1,year:0} )
MongoServerError: Cannot do exclusion on field year in inclusion projection
sandeep> db.movie.find({year:1995}, {title:1,year:1} )
[
  { _id: 'movie:14', title: 'Seven', year: 1995 },
  { _id: 'movie:15', title: 'Twelve Monkeys', year: 1995 },
  { _id: 'movie:52', title: 'Usual suspects', year: 1995 },
  { _id: 'movie:58', title: 'Une journée en enfer', year: 1995 },
  { _id: 'movie:69', title: 'Heat', year: 1995 },
  { _id: 'movie:72', title: 'Casino', year: 1995 }
]
sandeep> db.movie.find({year:1994}, {title:1,year:1} )
[
  { _id: 'movie:17', title: 'Pulp fiction', year: 1994 },
  { _id: 'movie:44', title: 'Léon', year: 1994 }
]
sandeep> db.movie.find({}, {title:1})
[
  { _id: 'movie:1', title: 'Vertigo' },
  { _id: 'movie:2', title: 'Alien' },
  { _id: 'movie:3', title: 'The Titanic' },
  { _id: 'movie:4', title: 'Sacifice' },
  { _id: 'movie:5', title: 'Volte/Face' },
  { _id: 'movie:6', title: 'Sleepy Hollow' },

```

I had used update command with many because I had more than one and the command was **db.movie.updateMany({year: 1994}, {\$set:{year: 1990}})**

```

mongosh mongodb://127.0.0.1
{
  { _id: 'movie:20', title: 'Les dents de la mer' }
}
Type "it" for more
sandeep> db.movie.updateMany({year: 1994}, {$set:{year: 1990}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
sandeep> db.movie.find({year:1994})

sandeep> db.movie.find({country:"FR"})
[
  {
    _id: 'movie:3',
    title: 'The 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 question, Rose DeWitt Bukater, aujourd'hui centenaire, rejoint les lieux du naufrage, d'où elle entreprend de raconter le récit de son fascinant, étrange et tragique voyage...",
    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',
      }
    ]
  }
]
```

## 8 Update the movies with country “USA” from “FR”.

I have had a movie script where I got FR country and I had to change to USA as I shown in figure where I had 18 id where I had FR.

```

mongosh mongodb://127.0.0.1:27017
[1] > db.movie.find({title: 'Les dents de la mer'})
[1] >
[{"_id": "movie:20", "title": "Les dents de la mer"}]
Type "it" for more
sandeep> db.movie.updateMany({year: 1994}, {$set: {year: 1990}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
sandeep> db.movie.find({year:1994})
sandeep> db.movie.find({country:"FR"})
[
  {
    "_id": "movie:3",
    "title": "The 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 question, Rose DeWitt Bukater, aujourd'hui centenaire, rejoint les lieux du naufrage, d'où elle entreprend de raconter le récit de son fascinant, étrange et tragique voyage...",
    "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"
      }
    ]
  }
]

```

I had used update command with many because I had more than one and the command was **db.movie.updateMany({country: "FR"}, {\$set: {country: "USA"}})**

```

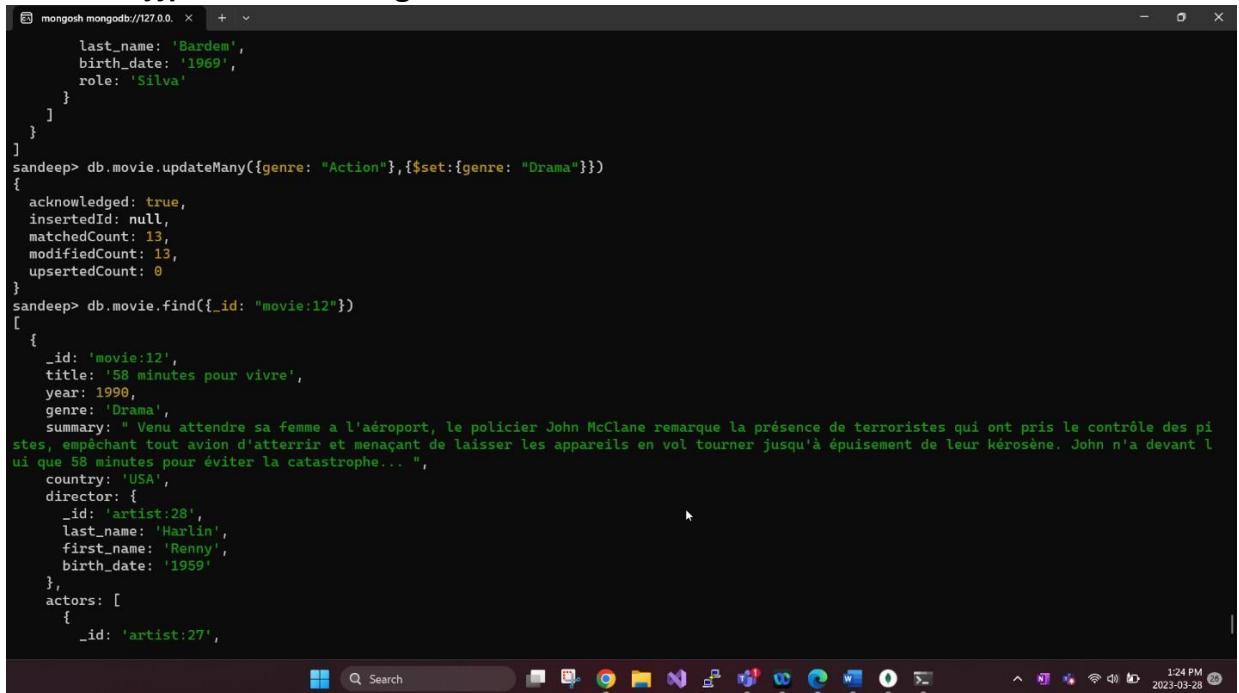
mongosh mongodb://127.0.0.1:27017
[1] > db.movie.find({year: 2015,
  genre: 'drama',
  summary: 'D\'un petit village du sud de l'Iran aux cités parisiennes, Kheiron nous raconte le destin hors du commun de ses parents Hibat et Ferehreh, éternels optimistes, dans une comédie aux airs de conte universel qui évoque l'amour familial, le don de soi et surtout l'idéal d'un vrai-ensemble.'},
  country: 'FR',
  director: {
    "_id": "artist:281",
    "last_name": "Kheiron",
    "first_name": "Tabib",
    "birth_date": null
  },
  actors: []
}
[1] >
[{"_id": "movie:18", "year": 2015, "genre": "drama", "summary": "D'un petit village du sud de l'Iran aux cit\u00e9s parisiennes, Kheiron nous raconte le destin hors du commun de ses parents Hibat et Ferehreh, \u00e9ternels optimistes, dans une com\u00e9die aux airs de conte universel qui \u00e9voque l'amour familial, le don de soi et surtout l'id\u00e9al d'un vrai-ensemble.", "country": "FR", "director": {"_id": "artist:281", "last_name": "Kheiron", "first_name": "Tabib", "birth_date": null}, "actors": []}]
sandeep> db.movie.updateMany({country: "FR"}, {$set: {country: "USA"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 18,
  modifiedCount: 18,
  upsertedCount: 0
}
sandeep> db.movie.find({genre: "Action"})
[
  {
    "_id": "movie:5",
    "title": "Volte/Face",
    "year": 1997,
    "genre": "Action",
    "summary": "Directeur d'une unit\u00e9 anti-terroriste, Sean Archer recherche Castor Troy, un criminel responsable de la mort de son fils six ans plus t\u00e9t. Il parvient \u00e0 l'arr\u00e8ter mais apprend que Troy a cach\u00e9 une bombe au Palais des Congr\u00e8s de Los Angeles. Seul le fr\u00e8re de Troy peut la d\u00e9amorcer et, pour l'approcher, Archer se fait greffer le visage de Troy. ",
    "country": "USA",
    "director": {
      ...
    }
  }
]

```

## 9 Update the movies with genre “Drama” from “Action”.

I have had a movie script where I got Action genre and I had to change to Drama as I shown in figure where I had 13 id where I had Action.

I had used update command with many because I had more than one and the command was **db.movie.updateMany({genre: “Action”}, {\$set: {genre: “Drama”}})** as shown in figure.

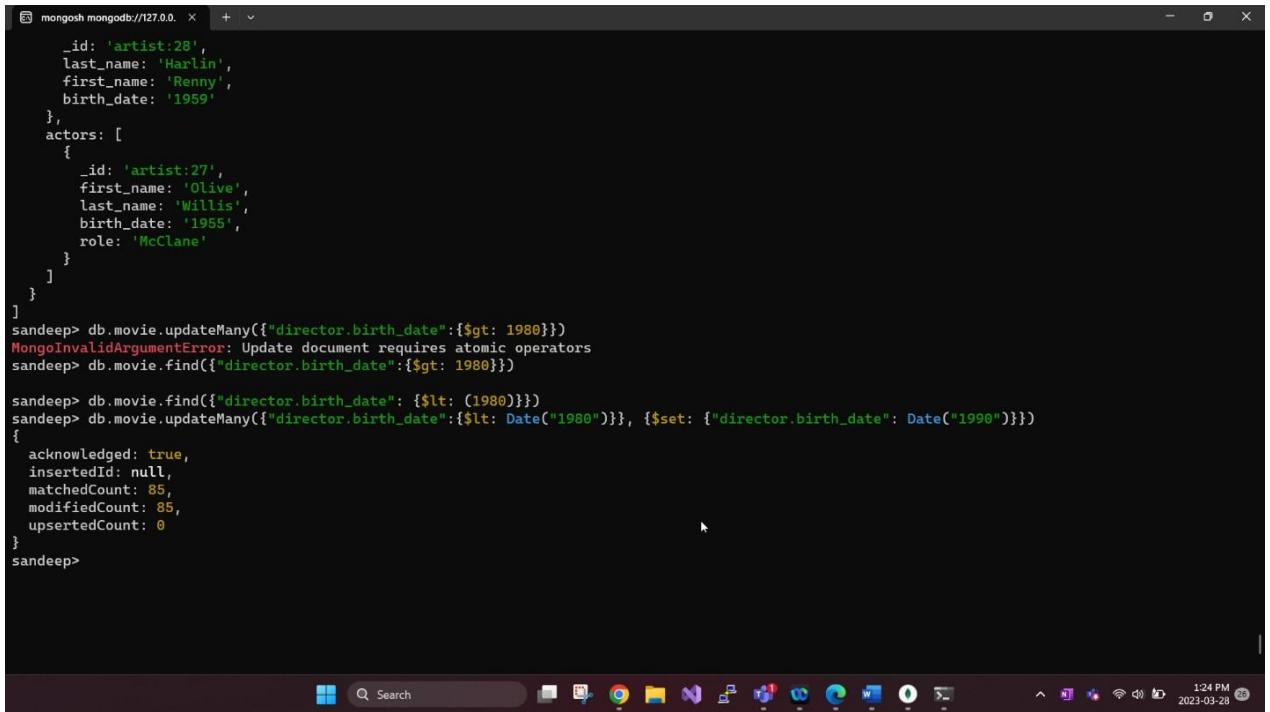


```
mongosh mongodb://127.0.0.1:27017
[sandeep@localhost:27017] 124 PM
> db.movie.updateMany({genre: "Action"}, {$set: {genre: "Drama"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 13,
  modifiedCount: 13,
  upsertedCount: 0
}
[sandeep@localhost:27017] 124 PM
> db.movie.find({_id: "movie:12"})
[
  {
    _id: 'movie:12',
    title: '58 minutes pour vivre',
    year: 1990,
    genre: 'Drama',
    summary: "Venu attendre sa femme à l'aéroport, le policier John McClane remarque la présence de terroristes qui ont pris le contrôle des pistes, empêchant tout avion d'atterrir et menaçant de laisser les appareils en vol tourner jusqu'à épuisement de leur kérosène. John n'a devant lui que 58 minutes pour éviter la catastrophe... ",
    country: 'USA',
    director: {
      _id: 'artist:28',
      last_name: 'Marlin',
      first_name: 'Renny',
      birth_date: '1959'
    },
    actors: [
      {
        _id: 'artist:27',
        last_name: 'Hannibal',
        first_name: 'Lecter',
        birth_date: '1937'
      }
    ]
  }
]
```

## 10 Update the movies with director’s birth\_date to 1990 where birth date older than 1980.

I had directors birth\_date with 1980 which I had to change to 1990 there was 85 ids where I got 1990 as I shown in figure while using

**db.movie.updateMany({"director.birth\_date": {\$lt: Date("190")}}, {\$set: {"directors.birth\_date": Date("1990")}})**



```
_id: 'artist:28',
last_name: 'Harlin',
first_name: 'Renny',
birth_date: '1959'
},
actors: [
{
_id: 'artist:27',
first_name: 'Olive',
last_name: 'Willis',
birth_date: '1955',
role: 'McClane'
}
]
]
}
sandeep> db.movie.updateMany({ "director.birth_date": { $gt: 1980 } })
MongoInvalidArgumentError: Update document requires atomic operators
sandeep> db.movie.find({ "director.birth_date": { $gt: 1980 } })

sandeep> db.movie.find({ "director.birth_date": { $lt: (1980) } })
sandeep> db.movie.updateMany({ "director.birth_date": { $lt: Date("1980") } }, { $set: { "director.birth_date": Date("1990") } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 85,
  modifiedCount: 85,
  upsertedCount: 0
}
sandeep>
```

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

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

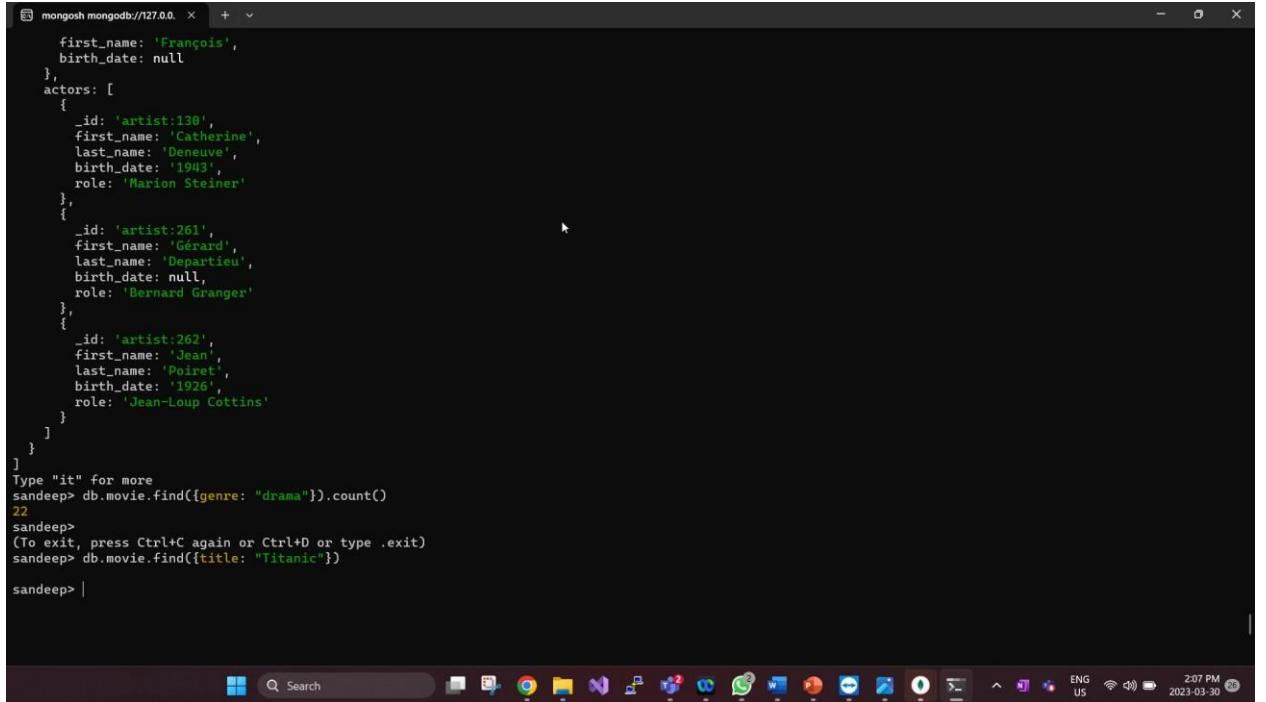
I had to fetch drama genre from all the movies and for that I used command like **db.movie.find({genre: “drama”})**. There are 22 movie where I had genre with drama as shown in two figure.

```
mongosh mongodb://127.0.0.1:27017

sandeep> 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, ayant des tendances suicidaires. Amoureux de la jeune femme Scottie ne remarque pas le piège qui se trame autour de lui et dont il va être la victime...",
  country: 'DE',
  director: {
    _id: 'artist:3',
    last_name: 'Hitchcock',
    first_name: 'Alfred',
    birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
  },
  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 movies title with “Titanic”.**

As in previous part I updated Titanic to The Titanic so it will not show anything there will no movie with Titanic title as shown in figure. The command is **db.movie.find({title: "Titanic"})**



```
mongosh mongodb://127.0.0.1:27017
{
  first_name: 'François',
  birth_date: null
},
actors: [
  {
    _id: 'artist:130',
    first_name: 'Catherine',
    last_name: 'Deneuve',
    birth_date: '1943',
    role: 'Marion Steiner'
  },
  {
    _id: 'artist:261',
    first_name: 'Gérard',
    last_name: 'Départieu',
    birth_date: null,
    role: 'Bernard Granger'
  },
  {
    _id: 'artist:262',
    first_name: 'Jean',
    last_name: 'Poirot',
    birth_date: '1926',
    role: 'Jean-Loup Cottins'
  }
]
}
Type "it" for more
sandeep> db.movie.find({genre: "drama"}).count()
22
sandeep>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
sandeep> db.movie.find({title: "Titanic"})
sandeep> |
```

### 3 Fetch all the movies director's first\_name as “John”.

Well, I had to fetch John which is director name from all movies and it's a first\_name of director as I shown in figure the command is  
**db.movie.find({"director.first\_name": "john"})**

```
sandeep>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
sandeep> db.movie.find({"director.first_name": "John"})
[
  {
    _id: 'movie:5',
    title: 'Volte/Face',
    year: 1997,
    genre: 'Drama',
    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 apprend que Troy a caché une bombe au Palais des Congrès de Los Angeles. Seul le frère de Troy peut la désamorcer et, pour l'approcher, Archer se fait greffer le visage de Troy. ",
    country: 'USA',
    director: {
      _id: 'artist:10',
      last_name: 'Woo',
      first_name: 'John',
      birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
    },
    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'
      }
    ]
  },
  {
    _id: 'movie:11',
    title: 'Piège de cristal',
  }
]
```

As I shown in figure there was 3 movies where I found John which is the directors first\_name.

```
]
},
{
  _id: 'movie:84',
  title: 'Rio Grande',
  year: 1950,
  genre: 'Western',
  summary: "Colonel de la cavalerie américaine, Kirby York accueille dans son régiment son propre fils, le jeune Jeff, recalé à West Point. Séparée de York depuis un épisode tragique de la Guerre de Sécession, Kathleen, la mère de Jeff, intervient auprès de son époux pour qu'il n'accepte pas le jeune homme dans sa garnison. La guerre contre les Indiens fait rage, et la mère craint pour la vie de son fils. Une terrible bataille contre les Apaches s'annonce... ",
  country: 'USA',
  director: {
    _id: 'artist:263',
    last_name: 'Ford',
    first_name: 'John',
    birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
  },
  actors: [
    {
      _id: 'artist:264',
      first_name: 'John',
      last_name: 'Wayne',
      birth_date: '1907',
      role: 'Lt. Col. Kirby York'
    },
    {
      _id: 'artist:265',
      first_name: 'Maureen',
      last_name: 'O'Hara',
      birth_date: '1920',
      role: 'Mrs. Kathleen York'
    }
  ]
}
sandeep> db.movie.find({"director.first_name": "John"}).count()
3
sandeep>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
sandeep>
```

#### 4 Fetch all the movies title start with "G".

To fetch all the documents from a MongoDB collection called "movie" that have movie titles starting with "G", we can use the \$regex operator with a regular expression pattern that matches the desired title format.

```
db.movie.find({title: /^G/})
```

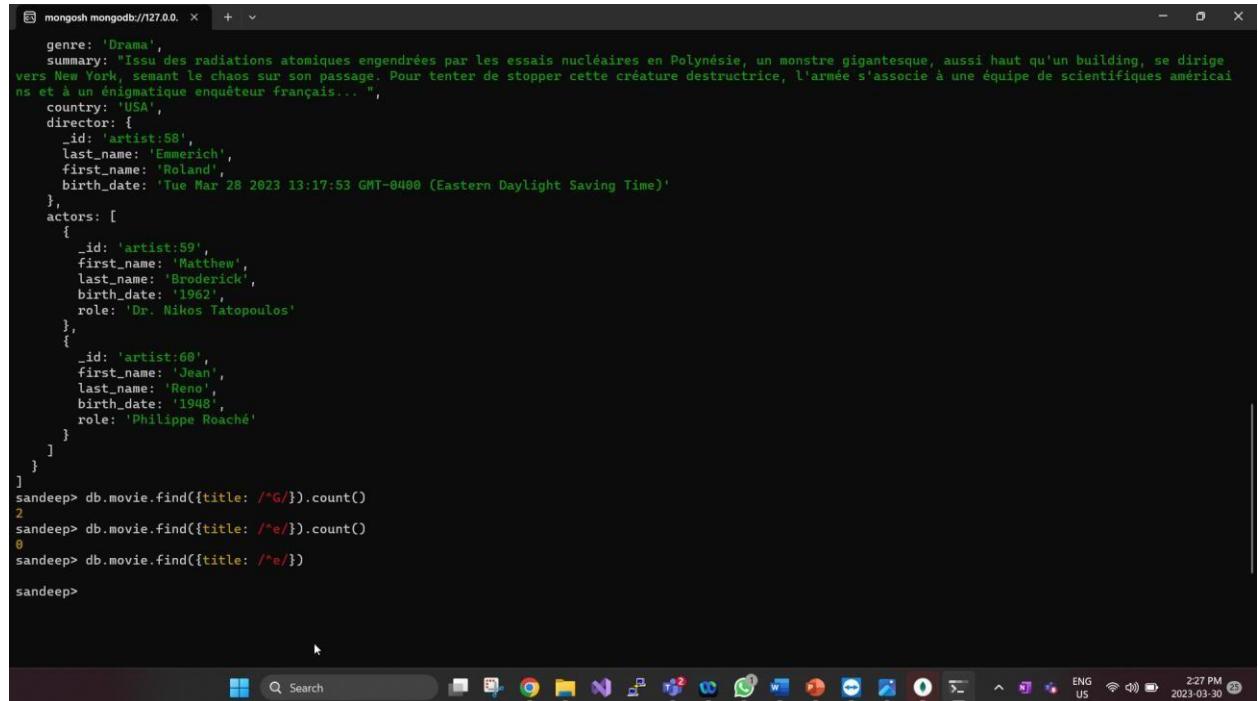
```
mongosh> use sandeep
switched to db sandeep
sandeep> db.movie.find({title: /^G/})
[ {
    _id: 'movie:9',
    title: 'Gladiator',
    year: 2000,
    genre: 'drama',
    summary: "Le général romain Maximus est le plus fidèle soutien de l'empereur Marc Aurèle, qu'il a conduit de victoire avec une bravoure et un dévouement exemplaires. Jaloux du prestige de Maximus, et plus encore de l'amour que lui voulait l'empereur, le fils de Marc-Aurèle, Commode, s'arroge brutalement le pouvoir, puis ordonne l'arrestation du général et son exécution. Maximus échappe à ses assassins mais ne peut empêcher le massacre de sa famille. Capturé par un marchand d'esclaves, il devient gladiateur et prépare sa vengeance.",
    country: 'USA',
    director: {
        _id: 'artist:4',
        last_name: 'Doe',
        first_name: 'Ridley',
        birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
    },
    actors: [
        {
            _id: 'artist:23',
            first_name: 'Russell',
            last_name: 'Crowe',
            birth_date: '1964',
            role: 'Maximus'
        },
        {
            _id: 'artist:147',
            first_name: 'Adam',
            last_name: 'Baldwin',
            birth_date: '1962',
            role: 'Commode'
        },
        {
            _id: 'artist:148',
            first_name: 'Ryan',
        }
    ]
},
{
    _id: 'movie:22',
    title: 'Godzilla',
    year: 1998,
    genre: 'Drama',
    summary: "Issu des radiations atomiques engendrées par les essais nucléaires en Polynésie, un monstre gigantesque, aussi haut qu'un building, se dirige vers New York, semant le chaos sur son passage. Pour tenter de stopper cette créature destructrice, l'armée s'associe à une équipe de scientifiques américains et à un énigmatique enquêteur français...",
    country: 'USA',
    director: {
        _id: 'artist:58',
        last_name: 'Emmerich',
        first_name: 'Roland',
        birth_date: '1962',
        birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
    },
    actors: [
        {
            _id: 'artist:59',
            first_name: 'Matthew',
            last_name: 'Broderick',
            birth_date: '1962',
            role: 'Dr. Nikos Tatopoulos'
        },
        {
            _id: 'artist:60',
            first_name: 'Jean',
            last_name: 'Reno',
            birth_date: '1948',
            role: 'Philippe Roaché'
        }
    ]
}
]
sandeep> db.movie.find({title: /^G/}).count()
2
sandeep> |
```

```
mongosh> use sandeep
switched to db sandeep
sandeep> db.movie.find({title: /^G/}).count()
2
sandeep> |
```

## 5 Fetch all the movies title end with "e".

To fetch all the documents from a MongoDB collection called "movie" that have movie titles ending with "e", we can use the \$regex operator with a regular expression pattern that matches the desired title format.

```
db.movie.find({title: /^e/})
```



The screenshot shows a terminal window titled "mongosh mongodb/127.0.0.1" running on a Windows operating system. The window displays the results of a MongoDB query. The output shows a single document with a long title in French, followed by two counts: 2 and 0. The counts correspond to the results of the queries shown above the counts. The terminal window is located on a desktop with a taskbar at the bottom containing various icons for applications like File Explorer, Google Chrome, and Microsoft Word.

```
mongosh mongodb/127.0.0.1
{
  genre: 'Drama',
  summary: "Issu des radiations atomiques engendrées par les essais nucléaires en Polynésie, un monstre gigantesque, aussi haut qu'un building, se dirige vers New York, semant le chaos sur son passage. Pour tenter de stopper cette créature déstructrice, l'armée s'associe à une équipe de scientifiques américains et à un énigmatique enquêteur français... ",
  country: 'USA',
  director: {
    _id: 'artist:58',
    last_name: 'Emmerich',
    first_name: 'Roland',
    birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
  },
  actors: [
    {
      _id: 'artist:59',
      first_name: 'Matthew',
      last_name: 'Broderick',
      birth_date: '1962',
      role: 'Dr. Nikos Tatopoulos'
    },
    {
      _id: 'artist:60',
      first_name: 'Jean',
      last_name: 'Reno',
      birth_date: '1948',
      role: 'Philippe Roaché'
    }
  ]
}
2
0
0
sandeep> db.movie.find({title: /^G/}).count()
2
sandeep> db.movie.find({title: /e/}).count()
0
sandeep> db.movie.find({title: /e/})
sandeep>
```

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

To fetch all the documents from a MongoDB collection called "movie" that have movie titles starting and ending with "t", you can use the \$regex operator with a regular expression pattern that matches the desired title format.

```
db.movie.find({ title: /t.*t$/i }, {title: 1})
```

```
| mongosh mongodb://127.0.0.1:27017 | + - x
|   role: 'Gordon'
| },
| {
|   _id: 'artist:269',
|   first_name: 'Michael',
|   last_name: 'Caine',
|   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'
| }
| ]
sandeep> db.movie.find({ title: /^t.*t$/i }).count()
1
sandeep> db.movie.find({ title: /^t.*t$/i }, {title: 1}).count()
1
sandeep> db.movie.find({ title: /^t.*t$/i }, {title: 1})
[ { _id: 'movie:89', title: 'The Dark Knight' } ]
sandeep> |
```

## 7 find all movies that have a summary that contains the word "Los Angeles".

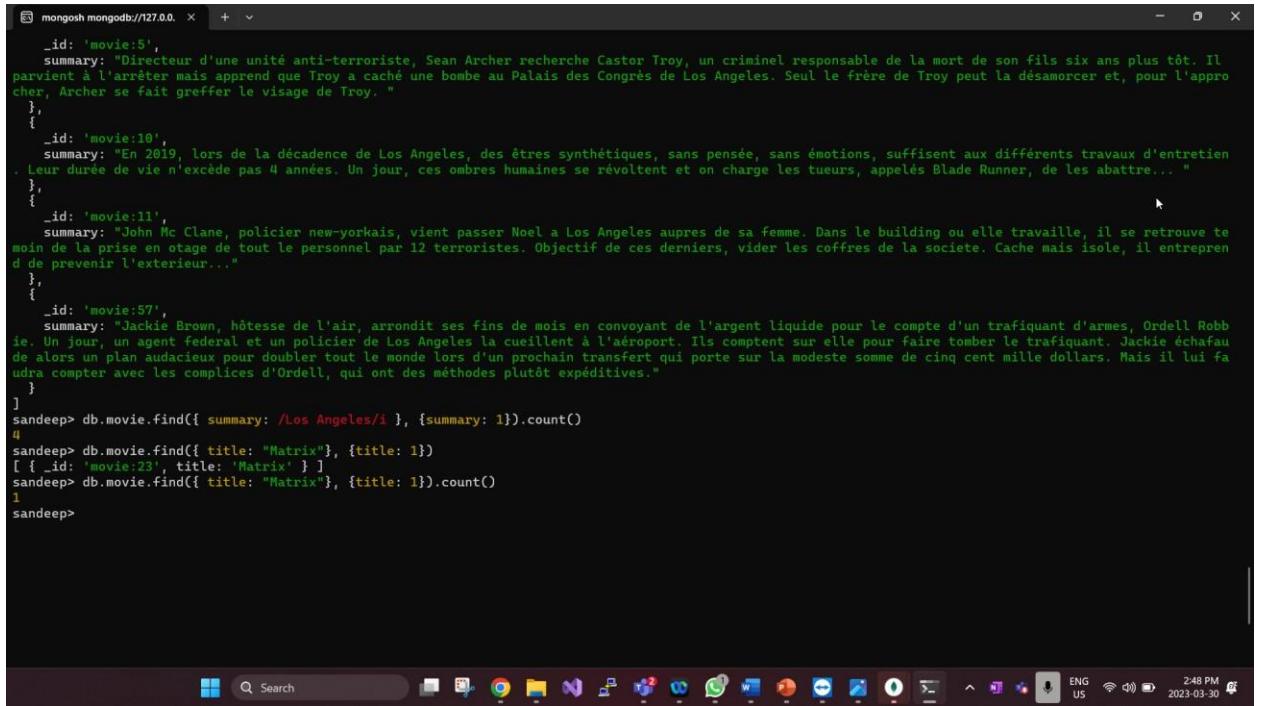
To fetch all the documents from a MongoDB collection called "movie" that have a summary containing the word "Los Angeles", we can use the \$regex operator with a regular expression pattern that matches the desired summary format.

```
db.movie.find({ summary: /Los Angeles/i }, {summary: 1})
```

```
| mongosh mongodb://127.0.0.1:27017 | + - x
{
  _id: 'artist:212',
  first_name: 'Samuel',
  last_name: 'Jackson',
  birth_date: '1948',
  role: 'Ordell Robbie'
}
]
sandeep> db.movie.find({ summary: /Los Angeles/i }, {summary: 1})
[
{
  _id: 'movie:5',
  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 apprend que Troy a caché une bombe au Palais des Congrès de Los Angeles. Seul le frère de Troy peut la désamorcer et, pour l'approcher, Archer se fait greffer le visage de Troy."
},
{
  _id: 'movie:10',
  summary: "En 2019, lors de la décadence de Los Angeles, des êtres synthétiques, sans pensée, sans émotions, suffisent aux différents travaux d'entretien. Leur durée de vie n'excède pas 4 années. Un jour, ces ombres humaines se révoltent et en charge les tueurs, appelés Blade Runner, de les abattre... "
},
{
  _id: 'movie:11',
  summary: "John Mc Clane, policier new-yorkais, vient passer Noel à Los Angeles auprès de sa femme. Dans le building où elle travaille, il se retrouve témoin de la prise en otage de tout le personnel par 12 terroristes. Objectif de ces derniers, vider les coffres de la société. Cache mais isolé, il entreprend de prévenir l'extérieur..."
},
{
  _id: 'movie:57',
  summary: "Jackie Brown, hôtesse de l'air, arroindit ses fins de mois en convoyant de l'argent liquide pour le compte d'un trafiquant d'armes, Ordell Robbie. Un jour, un agent fédéral et un policier de Los Angeles la cueillent à l'aéroport. Ils comptent sur elle pour faire tomber le trafiquant. Jackie échafauda alors un plan audacieux pour doubler tout le monde lors d'un prochain transfert qui porte sur la modeste somme de cinq cent mille dollars. Mais il lui faudra compter avec les complices d'Ordell, qui ont des méthodes plutôt expéditives."
}
]
sandeep> db.movie.find({ summary: /Los Angeles/i }, {summary: 1}).count()
4
sandeep> |
```

## 8 find all movies that have a title that contains the word "Matrix".

To fetch all the documents from a MongoDB collection called "movie" that have a title containing the word "Matrix", you can use the \$regex operator with a regular expression pattern that matches the desired title format. `db.movie.find({ title: "Matrix" }, {title: 1})`



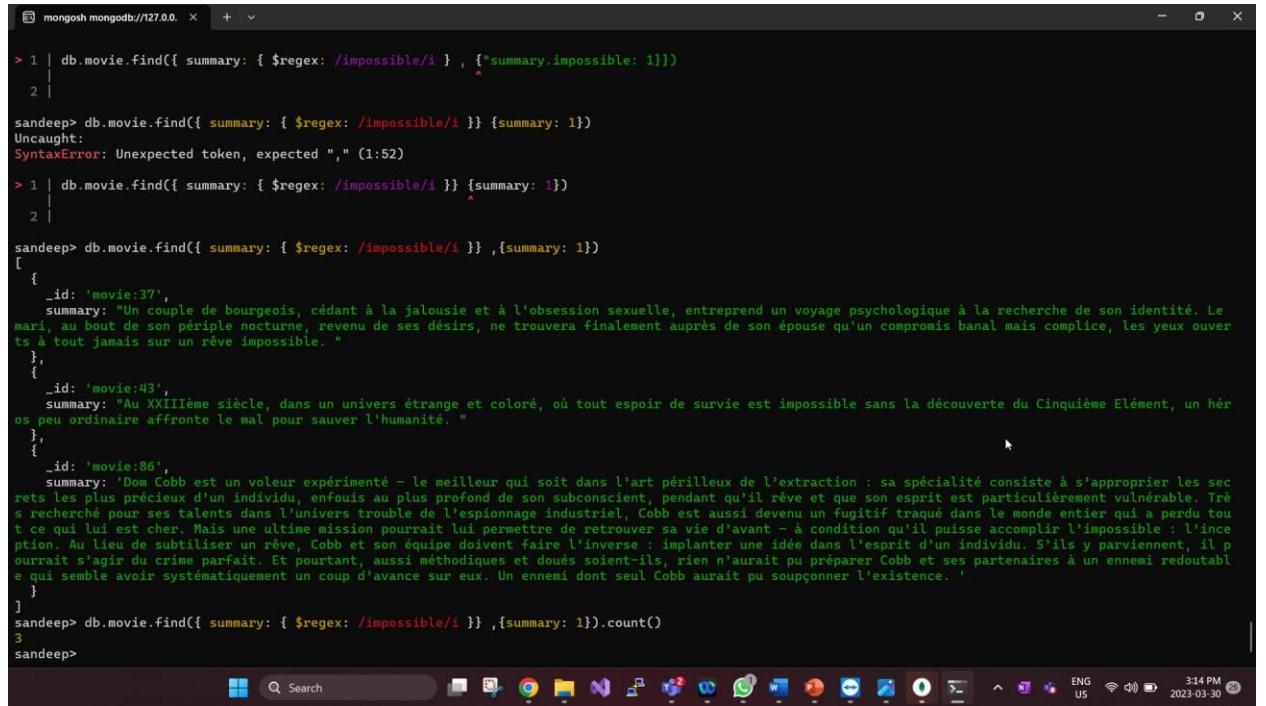
A screenshot of a terminal window titled "mongosh mongodb://127.0.0.1:27017". The window displays several movie documents from a collection named "movie". The documents include fields like \_id, title, and summary. One document has a summary about Sean Archer from the movie "Castor Troy". Another document has a summary about Blade Runner. A third document has a summary about Jackie Brown from "Pulp Fiction". The user then runs a query to find movies where the summary contains the word "/Los Angeles/i". The result shows 4 matches. The user then runs another query to find movies where the title is "Matrix", which also returns 4 results.

```
_id: 'movie:5',
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 apprend que Troy a caché une bombe au Palais des Congrès de Los Angeles. Seul le frère de Troy peut la désamorcer et, pour l'approcher, Archer se fait greffer le visage de Troy. "
},
{
_id: 'movie:10',
summary: "En 2019, lors de la décadence de Los Angeles, des êtres synthétiques, sans pensée, sans émotions, suffisent aux différents travaux d'entretien. Leur durée de vie n'excède pas 4 années. Un jour, ces ombres humaines se révoltent et on charge les tueurs, appelés Blade Runner, de les abattre... "
},
{
_id: 'movie:11',
summary: "John Mc Clane, policier new-yorkais, vient passer Noel à Los Angeles auprès de sa femme. Dans le building où elle travaille, il se retrouve témoin de la prise en otage de tout le personnel par 12 terroristes. Objectif de ces derniers, vider les coffres de la société. Cache mais isolé, il entreprend de prévenir l'extérieur..."
},
{
_id: 'movie:57',
summary: "Jackie Brown, hôtesse de l'air, arrondit ses fins de mois en convoyant de l'argent liquide pour le compte d'un trafiquant d'armes, Ordell Robbie. Un jour, un agent fédéral et un policier de Los Angeles la cueillent à l'aéroport. Ils comptent sur elle pour faire tomber le trafiquant. Jackie échafauda alors un plan audacieux pour doubler tout le monde lors d'un prochain transfert qui porte sur la modeste somme de cinq cent mille dollars. Mais il lui faudra compter avec les complices d'Ordell, qui ont des méthodes plutôt expéditives."
}
]
sandeep> db.movie.find({ summary: /Los Angeles/i }, {summary: 1}).count()
4
sandeep> db.movie.find({ title: "Matrix"}, {title: 1})
[ { _id: 'movie:23', title: 'Matrix' } ]
sandeep> db.movie.find({ title: "Matrix"}, {title: 1}).count()
1
sandeep>
```

## 9 find all movies that have a summary that contains the word "impossible".

To find all movies that have a summary that contains the word "impossible" in MongoDB, you can use the \$regex operator to perform a regular expression search.

```
db.movie.find({ summary: { $regex: /impossible/i }},{summary: 1})
```



```
mongosh mongodb://127.0.0.1:27017

> 1 | db.movie.find({ summary: { $regex: /impossible/i } , {"summary.impossible": 1}})
2 |
3 |

sandeep> db.movie.find({ summary: { $regex: /impossible/i } } {summary: 1})
Uncaught:
SyntaxError: Unexpected token, expected "," (1:52)
> 1 | db.movie.find({ summary: { $regex: /impossible/i } } {"summary": 1})
2 |
3 |

sandeep> db.movie.find({ summary: { $regex: /impossible/i } },{summary: 1})
[
  {
    _id: 'movie:37',
    summary: "Un couple de bourgeois, cédant à la jalouse et à l'obsession sexuelle, entreprend un voyage psychologique à la recherche de son identité. Le mari, au bout de son périple nocturne, revenu de ses désirs, ne trouvera finalement auprès de son épouse qu'un compromis banal mais complice, les yeux ouverts à tout jamais sur un rêve impossible. "
  },
  {
    _id: 'movie:43',
    summary: "Au XXIIIème siècle, dans un univers étrange et coloré, où tout espoir de survie est impossible sans la découverte du Cinquième Élément, un héros peu ordinaire affronte le mal pour sauver l'humanité. "
  },
  {
    _id: 'movie:86',
    summary: "Dom Cobb est un voleur expérimenté - le meilleur qui soit dans l'art périlleux de l'extraction : sa spécialité consiste à s'approprier les secrets les plus précieux d'un individu, enfouis au plus profond de son subconscient, pendant qu'il rêve et que son esprit est particulièrement vulnérable. Très recherché pour ses talents dans l'univers trouble de l'espionnage industriel, Cobb est aussi devenu un fugitif traqué dans le monde entier qui a perdu tout ce qui lui est cher. Mais une ultime mission pourrait lui permettre de retrouver sa vie d'avant - à condition qu'il puisse accomplir l'impossible : l'inception. Au lieu de subtiliser un rêve, Cobb et son équipe doivent faire l'inverse : planter une idée dans l'esprit d'un individu. S'ils y parviennent, il pourraient s'agir du crime parfait. Et pourtant, aussi méthodiques et doués soient-ils, rien n'aurait pu préparer Cobb et ses partenaires à un ennemi redoutable qui semble avoir systématiquement un coup d'avance sur eux. Un ennemi dont seul Cobb aurait pu soupçonner l'existence. "
  }
]
sandeep> db.movie.find({ summary: { $regex: /impossible/i } },{summary: 1}).count()
3
sandeep>
```

## 10 find all movies that have a summary that contains the word "Peter Parker".

To find all movies that have a summary that contains the phrase "Peter Parker" in MongoDB, you can use the \$regex operator to perform a regular expression search.

```
db.movies.find({ summary: { $regex: /Peter Parker/i } }, {summary: 1})
```

```
sandeep> db.movie.find({ summary: { $regex: /impossible/i } } ,{summary: 1})
Unccaught:
SyntaxError: Unexpected token, expected "," (1:52)
> 1 | db.movie.find({ summary: { $regex: /impossible/i } } ,{summary: 1})
2 |
sandeep> db.movie.find({ summary: { $regex: /impossible/i } } ,{summary: 1})
[
  {
    _id: 'movie:37',
    summary: "Un couple de bourgeois, cédant à la jalousie et à l'obsession sexuelle, entreprend un voyage psychologique à la recherche de son identité. Le mari, au bout de son périple nocturne, revenu de ses désirs, ne trouvera finalement auprès de son épouse qu'un compromis banal mais complice, les yeux ouverts à tout jamais sur un rêve impossible."
  },
  {
    _id: 'movie:43',
    summary: "Au XXIIIème siècle, dans un univers étrange et coloré, où tout espoir de survie est impossible sans la découverte du Cinquième Élément, un héros peu ordinaire affronte le mal pour sauver l'humanité."
  },
  {
    _id: 'movie:86',
    summary: 'Dom Cobb est un voleur expérimenté - le meilleur qui soit dans l'art périlleux de l'extraction : sa spécialité consiste à s'approprier les secrets les plus précieux d'un individu, enfouis au plus profond de son subconscient, pendant qu'il rêve et que son esprit est particulièrement vulnérable. Très recherché pour ses talents dans l'univers trouble de l'espionnage industriel, Cobb est aussi devenu un fugitif traqué dans le monde entier qui a perdu tout ce qui lui est cher. Mais une ultime mission pourrait lui permettre de retrouver sa vie d'avant - à condition qu'il puisse accomplir l'impossible : l'inception. Au lieu de subtiliser un rêve, Cobb et son équipe doivent faire l'inverse : implanter une idée dans l'esprit d'un individu. S'ils y parviennent, il pourraient s'agir du crime parfait. Et pourtant, aussi méthodiques et doués soient-ils, rien n'aurait pu préparer Cobb et ses partenaires à un ennemi redoutable qui semble avoir systématiquement un coup d'avance sur eux. Un ennemi dont seul Cobb aurait pu soupçonner l'existence.'
  }
]
sandeep> db.movie.find({ summary: { $regex: /impossible/i } } ,{summary: 1}).count()
3
sandeep> db.movies.find({ summary: { $regex: /Peter Parker/i } } ,{summary: 1})
sandeep> db.movies.find({ summary: { $regex: /Peter Parker/i } }).count()
0
sandeep> |
```

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

### 1. Create an index on title attribute in ascending order on movie collection.

I used this command `db.movie.createIndex({ title: 1 })`. This command creates a new index on the title attribute of the movie collection with an ascending order. The value 1 specifies that the index should be created in ascending order. `db.movie.getIndexes()`

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

I used this command. `db.movie.createIndex({ title: 1, genre: -1 })`. This command creates a new index on the title and genre attributes of the movie collection with title in ascending order and genre in descending order. The value 1 specifies that the title index should be created in ascending order, and -1 specifies that the genre index should be created in descending order.

```
mongosh mongodb://127.0.0.1:27017
title_1
sandeep> db.movie.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1 }, name: 'title_1' }
]
sandeep> db.movie.createIndex({ title: 1, genre: -1 })
title_1_genre_-1
sandeep> db.movie.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { title: 1, genre: -1 }, name: 'title_1_genre_-1' }
]
sandeep> |
```

### 3 Create a single index with multiple attributes (title by descending order and country by ascending order) on movie.

Well I used this command **db.movie.createIndex({ title: -1, country: 1 })**. This command creates a new index on the title and country attributes of the movie collection with title in descending order and country in ascending order. The value -1 specifies that the title index should be created in descending order, and 1 specifies that the country index should be created in ascending order.  
db.movie.getIndexes()

```
| mongosh mongodb://127.0.0.1:27017 | + - x
title_1
sandeep> db.movie.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1 }, name: 'title_1' }
]
sandeep> db.movie.createIndex({ title: 1, genre: -1 })
title_1_genre_-1
sandeep> db.movie.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { title: 1, genre: -1 }, name: 'title_1_genre_-1' }
]
sandeep> db.movie.createIndex({ title: -1, country: 1 })
title_-1_country_1
sandeep> db.movie.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { title: 1, genre: -1 }, name: 'title_1_genre_-1' },
  { v: 2, key: { title: -1, country: 1 }, name: 'title_-1_country_1' }
]
sandeep> |
```

#### 4. Create multiple indexes based on the multiple attributes including composite index.

**title: single index by ascending order**

**title and genre: composite index by descending order**

**year: single index by descending order**

To create a single index on the title field in ascending order:

**db.movie.createIndex({ title: 1 })**

```
sandeep@ localhost:~ % mongosh mongodb://127.0.0.1:27017
MongoDB shell version: 4.4.6
connecting to: mongodb://127.0.0.1:27017/test
+ 
sandeep> db.movie.createIndex({ title: 1 })
title_1
sandeep> db.movie.getIndexes()
[ 
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { title: 1, genre: -1 }, name: 'title_1_genre_-1' },
  { v: 2, key: { title: -1, country: 1 }, name: 'title_-1_country_1' }
]
sandeep>
```

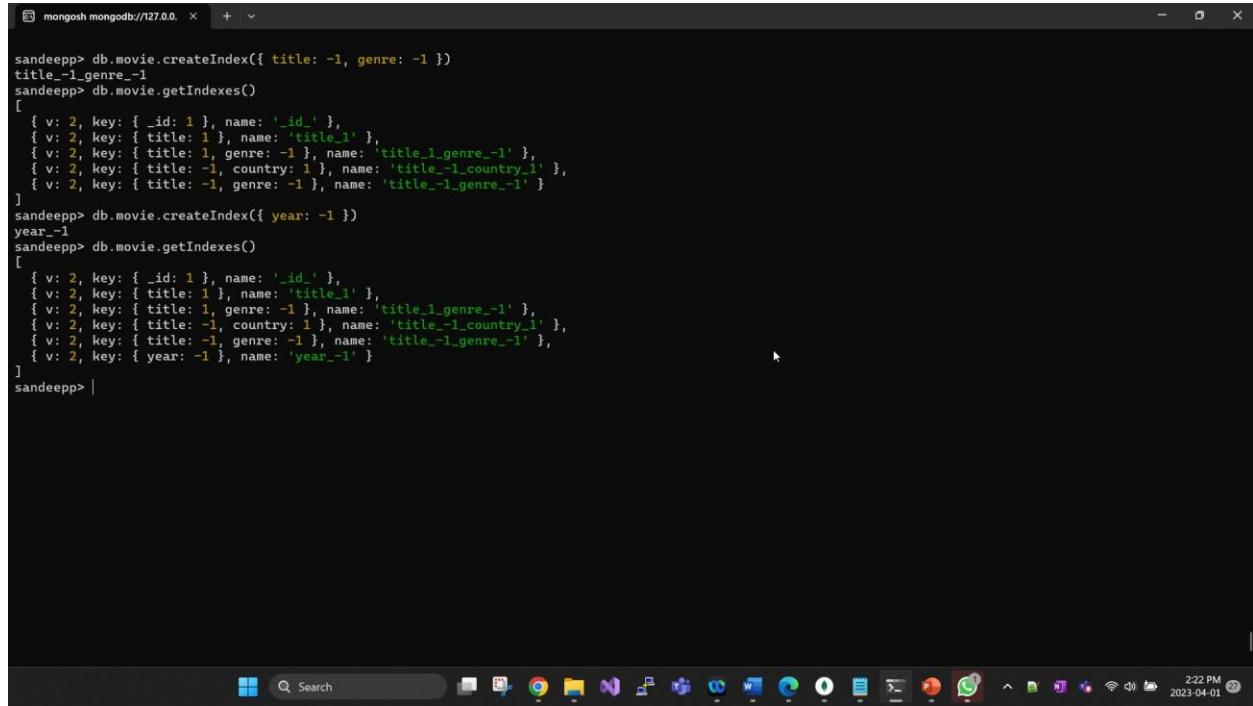
To create a composite index on the title and genre fields in descending order:

**db.movie.createIndex({ title: -1, genre: -1 })**

```
sandeep@ localhost:~ % mongosh mongodb://127.0.0.1:27017
MongoDB shell version: 4.4.6
connecting to: mongodb://127.0.0.1:27017/test
+ 
sandeep> db.movie.createIndex({ title: -1, genre: -1 })
title_-1_genre_-1
sandeep> db.movie.getIndexes()
[ 
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { title: 1, genre: -1 }, name: 'title_1_genre_-1' },
  { v: 2, key: { title: -1, country: 1 }, name: 'title_-1_country_1' },
  { v: 2, key: { title: -1, genre: -1 }, name: 'title_-1_genre_-1' }
]
sandeep>
```

To create a single index on the year field in descending order:

```
db.movie.createIndex({ year: -1 })
```



The screenshot shows a Windows desktop with a MongoDB shell window open. The title bar says "mongosh mongodb://127.0.0.1:27017". The command line shows:

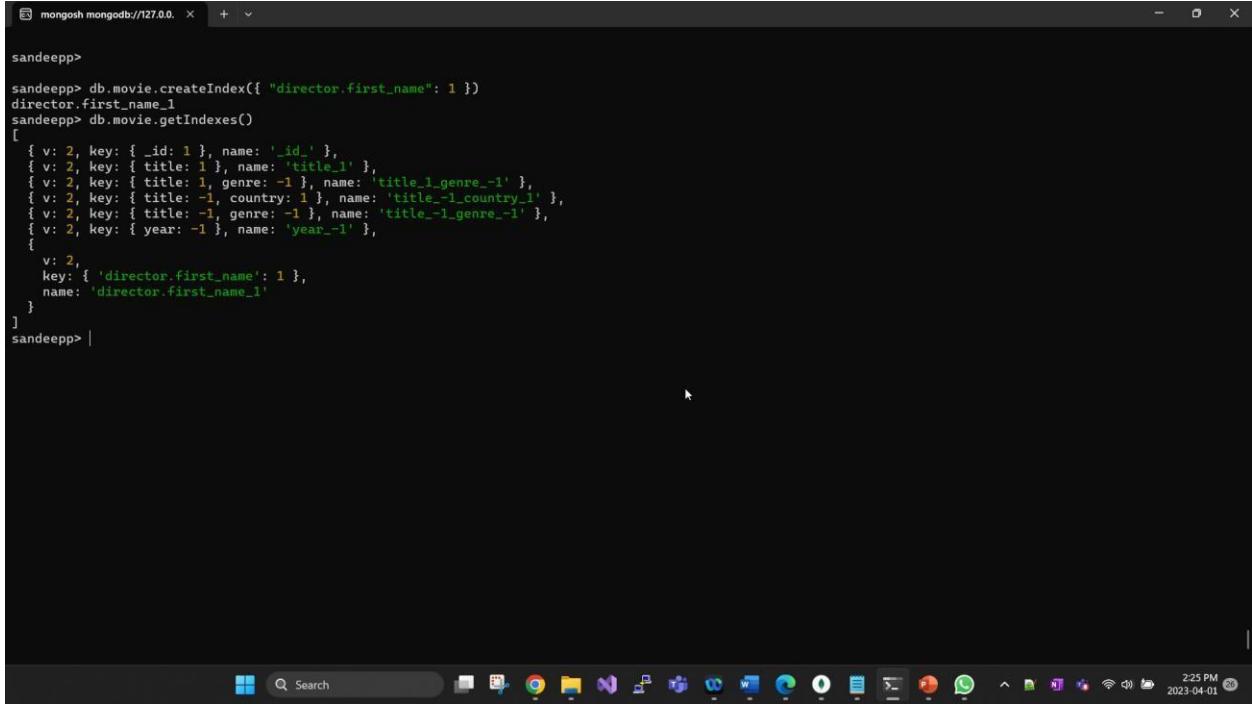
```
sandeep@DESKTOP-6D9C9A: ~ % mongosh
MongoDB shell version: 4.4.1
connecting to: mongodb://127.0.0.1:27017/test
Implicit session: ticket: 0
MongoDB shell version: 4.4.1
Implicit session: ticket: 0
> db.movie.createIndex({ title: -1, genre: -1 })
title_-1_genre_-1
> db.movie.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { title: 1, genre: -1 }, name: 'title_1_genre_-1' },
  { v: 2, key: { title: -1, country: 1 }, name: 'title_-1_country_1' },
  { v: 2, key: { title: -1, genre: -1 }, name: 'title_-1_genre_-1' } ]
> db.movie.createIndex({ year: -1 })
year_-1
> db.movie.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { title: 1, genre: -1 }, name: 'title_1_genre_-1' },
  { v: 2, key: { title: -1, country: 1 }, name: 'title_-1_country_1' },
  { v: 2, key: { title: -1, genre: -1 }, name: 'title_-1_genre_-1' },
  { v: 2, key: { year: -1 }, name: 'year_-1' } ]
>
```

The taskbar at the bottom shows various application icons.

## 5 Create a wildcard Index on director first\_name order by ascending order on movie collection.

This command creates a new wildcard index on the director.first\_name field of the movie collection, which indexes all of the words in the director.first\_name field in ascending order. This index can be used to search for any words or phrases in the director.first\_name field using the \$text operator.

```
db.movie.createIndex({ "director.first_name": 1 })
```

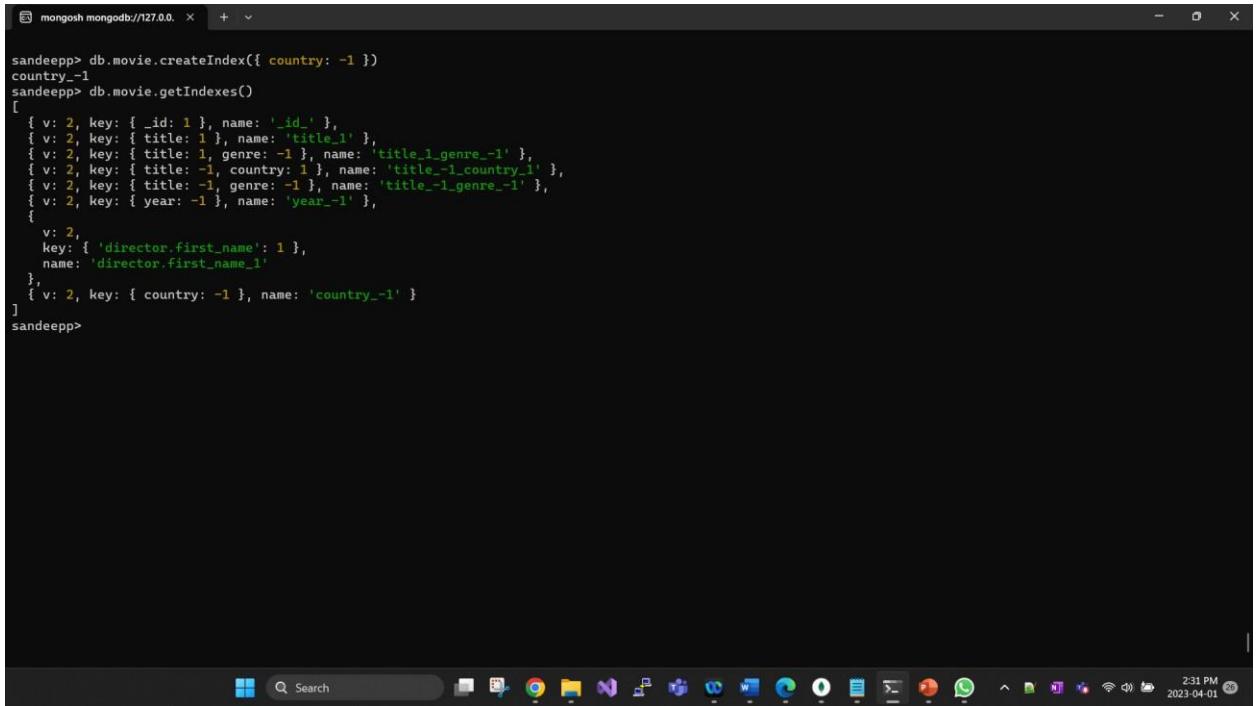


The screenshot shows a Windows desktop environment with a MongoDB shell window open. The shell window title is "mongosh mongodb://127.0.0.1:27017". The command "db.movie.createIndex({ director.first\_name: 1 })" is run, followed by "db.movie.getIndexes()", which lists several indexes including ones on \_id, title, title genre, title country, and year.

```
sandeep>
sandeep> db.movie.createIndex({ "director.first_name": 1 })
sandeep> db.movie.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { title: 1, genre: -1 }, name: 'title_1_genre_-1' },
  { v: 2, key: { title: -1, country: 1 }, name: 'title_-1_country_1' },
  { v: 2, key: { title: -1, genre: -1 }, name: 'title_-1_genre_-1' },
  { v: 2, key: { year: -1 }, name: 'year_-1' },
  {
    v: 2,
    key: { 'director.first_name': 1 },
    name: 'director.first_name_1'
  }
]
sandeep> |
```

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

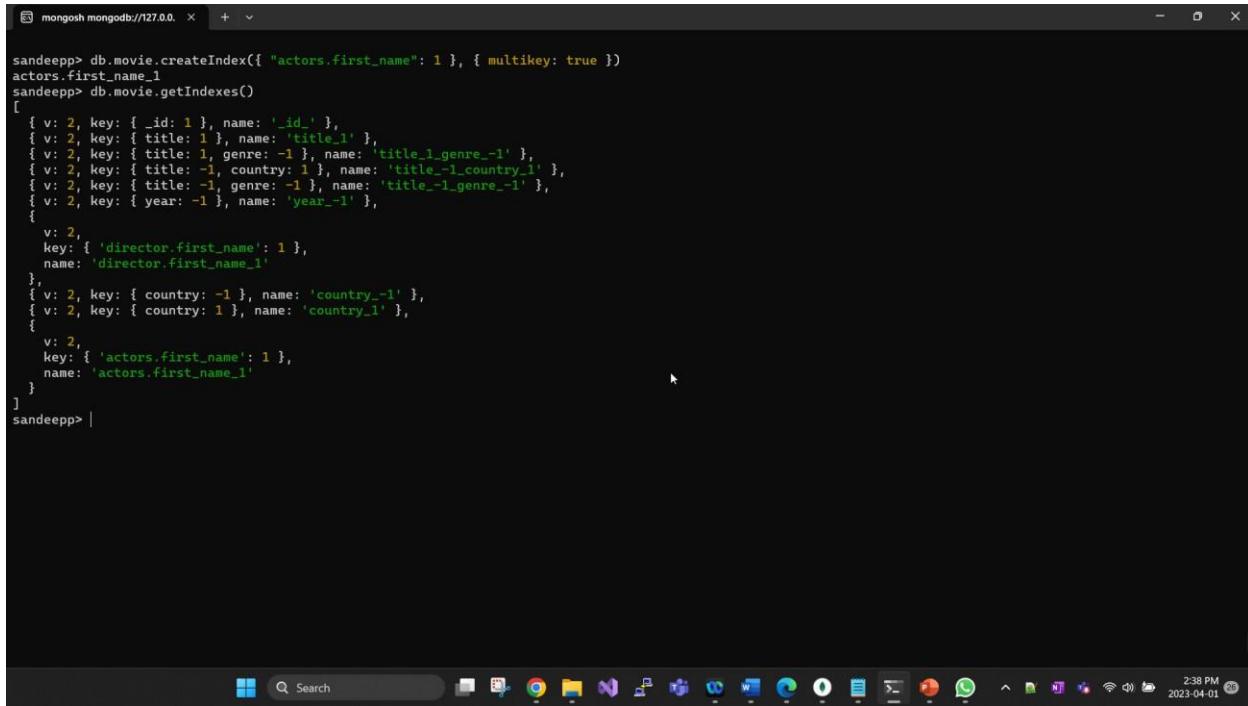
**db.movie.createIndex({ country: 1 }).**This command creates a new unique index on the country attribute of the movie collection, which indexes the country attribute in ascending order and enforces uniqueness on the values of the country attribute. This ensures that there can be no two documents in the movie collection with the same country value.



```
sandeeppp> db.movie.createIndex({ country: -1 })
country_-1
sandeeppp> db.movie.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { title: -1 }, name: 'title_-1_genre_-1' },
  { v: 2, key: { title: -1, country: 1 }, name: 'title_-1_country_1' },
  { v: 2, key: { title: -1, genre: -1 }, name: 'title_-1_genre_-1' },
  { v: 2, key: { year: -1 }, name: 'year_-1' },
  {
    v: 2,
    key: { 'director.first_name': 1 },
    name: 'director.first_name_1'
  },
  { v: 2, key: { country: -1 }, name: 'country_-1' }
]
sandeeppp>
```

## 7 Create a multikey Index on actors first\_name order by ascending order on movie collection.

**db.movie.createIndex({ "actors.first\_name": 1 }, { multikey: true })**. This command creates a new multikey index on the actors.first\_name field of the movie collection, which indexes all of the values of the actors.first\_name array in ascending order. This index can be used to search for any values in the actors.first\_name array using the \$in operator.



```
sandeep@DESKTOP-6D9C9A: ~ % mongosh mongodb://127.0.0.1
MongoDB shell version: 4.4.15
connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=5000
Implicit session: session#1 started.
MongoDB server version: 4.4.15
WARNING: This session is not yet connected to a database.
> db.movie.createIndex({ "actors.first_name": 1 }, { multikey: true })
actors.first_name_1
> db.movie.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { title: 1, genre: -1 }, name: 'title_1_genre_-1' },
  { v: 2, key: { title: -1, country: 1 }, name: 'title_-1_country_1' },
  { v: 2, key: { title: -1, genre: -1 }, name: 'title_-1_genre_-1' },
  { v: 2, key: { year: -1 }, name: 'year_-1' },
  {
    v: 2,
    key: { 'director.first_name': 1 },
    name: 'director.first_name_1'
  },
  { v: 2, key: { country: -1 }, name: 'country_-1' },
  { v: 2, key: { country: 1 }, name: 'country_1' },
  {
    v: 2,
    key: { 'actors.first_name': 1 },
    name: 'actors.first_name_1'
  }
]
>
```

## 8 Get all the Indexes from movie collection.

I used **db.movie.getIndexes()**. This command returns an array of documents, with each document describing an index on the movie collection. The documents include information such as the index name, the indexed fields, and any additional options set for the index.

### **db.movie.showIndexes()**

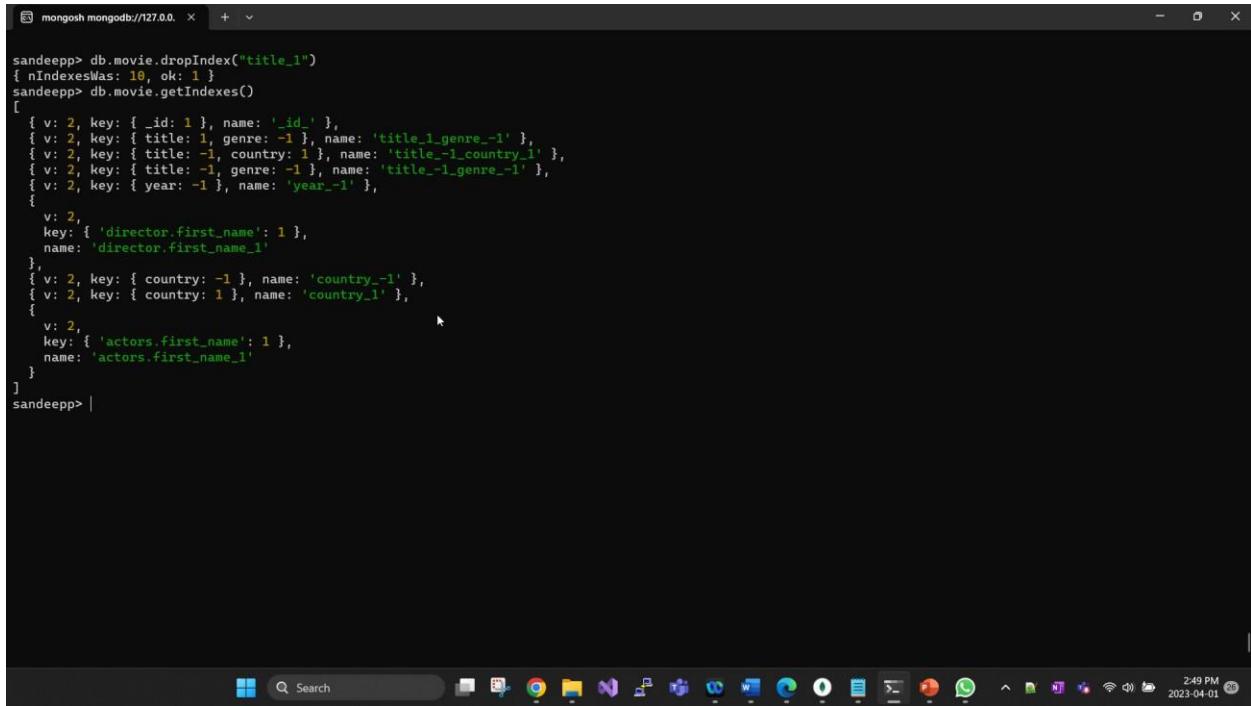
This command returns the same array of documents as `getIndexes()`, but with a more concise output format.

```
sandeep@DESKTOP-6V9D9C9: ~ % mongosh mongodb://127.0.0.1:27017
MongoDB shell version: 4.4.15
connecting to: mongodb://127.0.0.1:27017/
Implicit connection to admin database
MongoDB shell version: 4.4.15
Implicit connection to admin database
Welcome to the MongoDB shell.
For help enter ? or help()
For more info on this server see /proc/meminfo
For help, type ? or help(command)
db.movie.getIndexes()
[{"v": 2, "key": {"_id": 1}, "name": "_id_1"}, {"v": 2, "key": {"title": 1}, "name": "title_1"}, {"v": 2, "key": {"title": 1, "genre": -1}, "name": "title_1_genre_-1"}, {"v": 2, "key": {"title": -1, "country": 1}, "name": "title_-1_country_1"}, {"v": 2, "key": {"title": -1, "genre": -1}, "name": "title_-1_genre_-1"}, {"v": 2, "key": {"year": -1}, "name": "year_-1"}, {"v: 2, key: { 'director.first_name': 1 }, name: 'director.first_name_1' }, {"v: 2, key: { country: -1 }, name: 'country_-1' }, {"v: 2, key: { country: 1 }, name: 'country_1' }, {"v: 2, key: { 'actors.first_name': 1 }, name: 'actors.first_name_1' } ]
sandeep@DESKTOP-6V9D9C9: ~ %
```

## 9 Remove “title” index from movie collection.

```
movie.dropIndex("title_1")
```

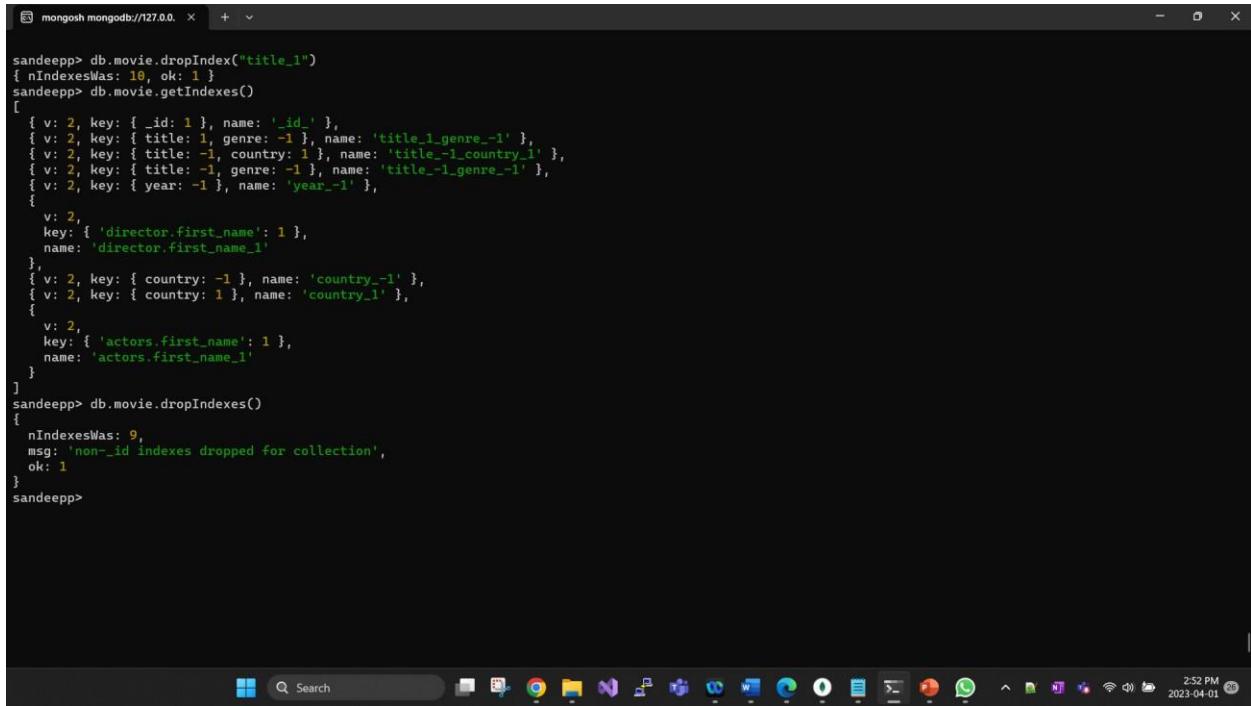
This command removes the index with the name "title\_1" from the movie collection. The name of the index is a combination of the field name ("title") and the sort order (1 for ascending order). If the index was created with a different name, you will need to replace "title\_1" with the correct name of the index.



```
sandeep> db.movie.dropIndex("title_1")
{ nIndexesWas: 10, ok: 1 }
sandeep> db.movie.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: 1, genre: -1 }, name: 'title_1_genre_-1' },
  { v: 2, key: { title: -1, country: 1 }, name: 'title_-1_country_1' },
  { v: 2, key: { title: -1, genre: -1 }, name: 'title_-1_genre_-1' },
  { v: 2, key: { year: -1 }, name: 'year_-1' },
  { v: 2,
    key: { 'director.first_name': 1 },
    name: 'director.first_name_1'
  },
  { v: 2, key: { country: -1 }, name: 'country_-1' },
  { v: 2, key: { country: 1 }, name: 'country_1' },
  { v: 2,
    key: { 'actors.first_name': 1 },
    name: 'actors.first_name_1'
  }
]
sandeep> |
```

## 10. Drop all the indexes from movie collection.

**db.movie.dropIndexes()**. This command removes all indexes from the movie collection, including the default index on the `_id` field. After executing this command, you will need to recreate any indexes that are necessary for your application to function properly.



```
sandeep> db.movie.dropIndex("title_1")
{ nIndexesWas: 10, ok: 1 }
sandeep> db.movie.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { v: 2, key: { title: -1 }, name: 'title_-1_genre_-1' },
  { v: 2, key: { title: -1, country: 1 }, name: 'title_-1_country_1' },
  { v: 2, key: { title: -1, genre: -1 }, name: 'title_-1_genre_-1' },
  { v: 2, key: { year: -1 }, name: 'year_-1' },
  {
    v: 2,
    key: { 'director.first_name': 1 },
    name: 'director.first_name_1'
  },
  { v: 2, key: { country: -1 }, name: 'country_-1' },
  { v: 2, key: { country: 1 }, name: 'country_1' },
  {
    v: 2,
    key: { 'actors.first_name': 1 },
    name: 'actors.first_name_1'
  }
]
sandeep> db.movie.dropIndexes()
{
  nIndexesWas: 9,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
sandeep>
```

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

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

**db.movie.deleteMany({ title: "Spider-Man" })**

This command removes all documents that match the { title: "Spider-Man" } filter from the movie collection. Make sure to use the correct filter to ensure that you are deleting the correct documents.

## 2 Delete the movies which is released in 1990.

```
db.movie.deleteMany({ release_year: 1990 })
```

This command removes all documents that match the { release\_year: 1990 } filter from the movie collection. Make sure to use the correct filter to ensure that you are deleting the correct documents.

**3. Delete the movies with director first\_name is “Richard”.**

This command removes all documents that match the { "director.first\_name": "Richard" } filter from the movie collection. Make sure to use the correct filter to ensure that you are deleting the correct documents. **db.movie.deleteMany({ "director.first\_name": "Richard" })**

#### **4 Delete a top movie released in USA.**

```
db.movie.deleteOne({ country: "USA" }, { sort: { rating: -1 }, limit: 1 })
```

This command finds all documents in the movie collection that have a country value of "USA", sorts them in descending order by rating, and deletes the top document (i.e., the one with the highest rating). If there are multiple movies with the same highest rating, the command will delete only the first one it encounters.

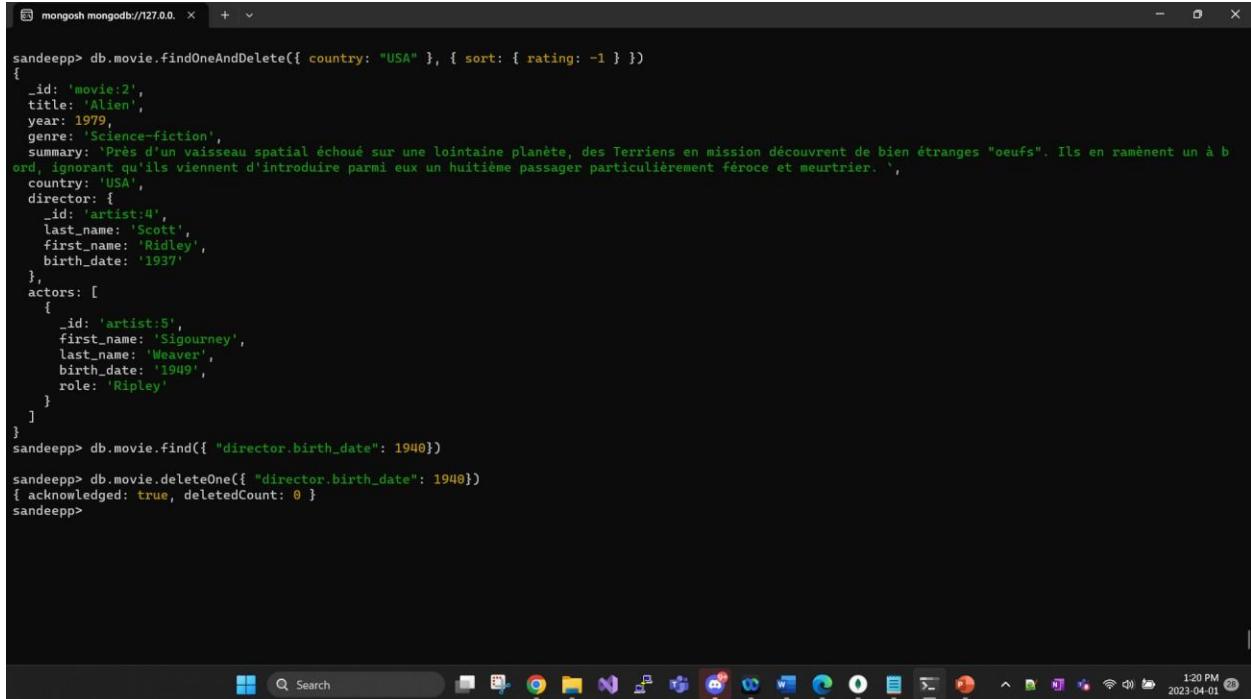
```
mongosh mongodb://127.0.0.1:27017
{
  acknowledged: true,
  deletedCount: 1
}
sandeep> db.movie.find({ "director.first_name": "Richard" }).count()
0
sandeep> db.movie.deleteOne({ country: "USA" }, { sort: { rating: -1 }, limit: 1 })
{
  acknowledged: true,
  deletedCount: 1
}
sandeep> db.movie.find({ country: "USA" }, { sort: { rating: -1 }, limit: 1 }).count()
71
sandeep> db.movie.find({ country: "USA" })
[
  {
    _id: 'movie:3',
    title: 'The 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 question, Rose DeWitt Bukater, aujourd'hui centenaire, rejoint les lieux du naufrage, d'où elle entreprend de raconter le récit de son fascinant, étrange et tragique voyage...",
    country: 'USA',
    director: {
      _id: 'artist:6',
      last_name: 'Cameron',
      first_name: 'James',
      birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
    },
    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'
      }
    ]
  }
]
10:33 PM
2023-04-01
```

```
]
},
{
  _id: 'movie:29',
  title: 'Rain Man',
  year: 1988,
  genre: 'drama',
  summary: "À la mort de son père, Charlie se voit déposséder de son héritage par un frère dont il ignorait l'existence, Raymond. Celui-ci est autiste et vit dans un hôpital psychiatrique. Charlie enlève Raymond afin de prouver qu'il est capable de s'en occuper et de toucher l'héritage.",
  country: 'USA',
  director: {
    _id: 'artist:79',
    last_name: 'Levinson',
    first_name: 'Barry',
    birth_date: 'Tue Mar 28 2023 13:17:53 GMT-0400 (Eastern Daylight Saving Time)'
  },
  actors: [
    {
      _id: 'artist:65',
      first_name: 'Tom',
      last_name: 'Cruise',
      birth_date: '1962',
      role: 'Charlie Babbitt'
    },
    {
      _id: 'artist:80',
      first_name: 'Dustin',
      last_name: 'Hoffman',
      birth_date: '1937',
      role: 'Raymond Babbitt'
    }
  ]
}
Type "it" for more
sandeep> db.movie.deleteMany({ country: "USA" })
{
  acknowledged: true,
  deletedCount: 70
}
sandeep> db.movie.find({ country: "USA" }).count()
0
sandeep>
```

## 5 Delete a movie with director birth\_date is 1940.

There was no birth\_date of any director so I got 0. I used deleteOne command.

```
db.movie.deleteOne({ "director.birth_date": 1940 })
```

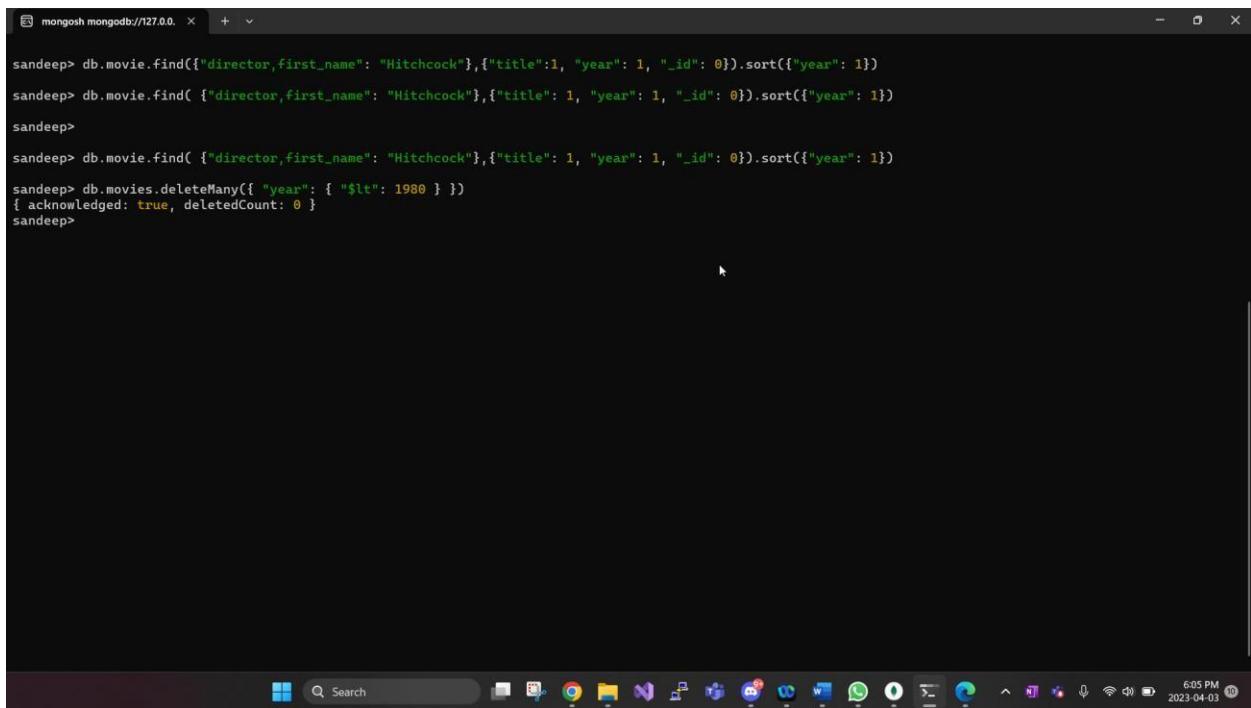


```
sandeep@ localhost:~ % mongosh mongodb://127.0.0.1:27017
sandeep> db.movie.findOneAndDelete({ country: "USA" }, { sort: { rating: -1 } })
{
  _id: 'movie:2',
  title: 'Alien',
  year: 1979,
  genre: 'Science-fiction',
  summary: 'Prés d'un vaisseau spatial échoué sur une lointaine planète, des Terriens en mission découvrent de bien étranges "œufs". Ils en ramènent un à bord, ignorant qu'ils viennent d'introduire parmi eux un huitième passager particulièrement féroce et meurtrier. ',
  country: 'USA',
  director: {
    _id: 'artist:4',
    last_name: 'Scott',
    first_name: 'Ridley',
    birth_date: '1937'
  },
  actors: [
    {
      _id: 'artist:5',
      first_name: 'Sigourney',
      last_name: 'Weaver',
      birth_date: '1949',
      role: 'Ripley'
    }
  ]
}
sandeep> db.movie.find({ "director.birth_date": 1940 })
[{"_id": "movie:2", "title": "Alien", "year": 1979, "genre": "Science-fiction", "summary": "Pr\u00e8s d'un vaisseau spatial \u00e9chou\u00e9 sur une lointaine plan\u00e8te, des Terriens en mission d\u00e9couvrent de bien \u00e9tranges \"oeufs\". Ils en ram\u00e8nent un \u00e0 bord, ignorant qu'ils viennent d'introduire parmi eux un huiti\u00e8me passager particuli\u00e8rement f\u00e9roce et meurtrier.", "country": "USA", "director": {"_id": "artist:4", "last_name": "Scott", "first_name": "Ridley", "birth_date": "1937"}, "actors": [{"_id": "artist:5", "first_name": "Sigourney", "last_name": "Weaver", "birth_date": "1949", "role": "Ripley"}]}
sandeep> db.movie.deleteOne({ "director.birth_date": 1940 })
{ acknowledged: true, deletedCount: 0 }
sandeep>
```

## 6 Delete the movies which is older than 1980.

Next, we use the deleteMany() method to delete the movies which are older than 1980. The query object { "year": { "\$lt": 1980 } } specifies the condition for selecting the documents to delete.

```
db.movie.deleteMany({ "year": { "$lt": 1980 } })
```

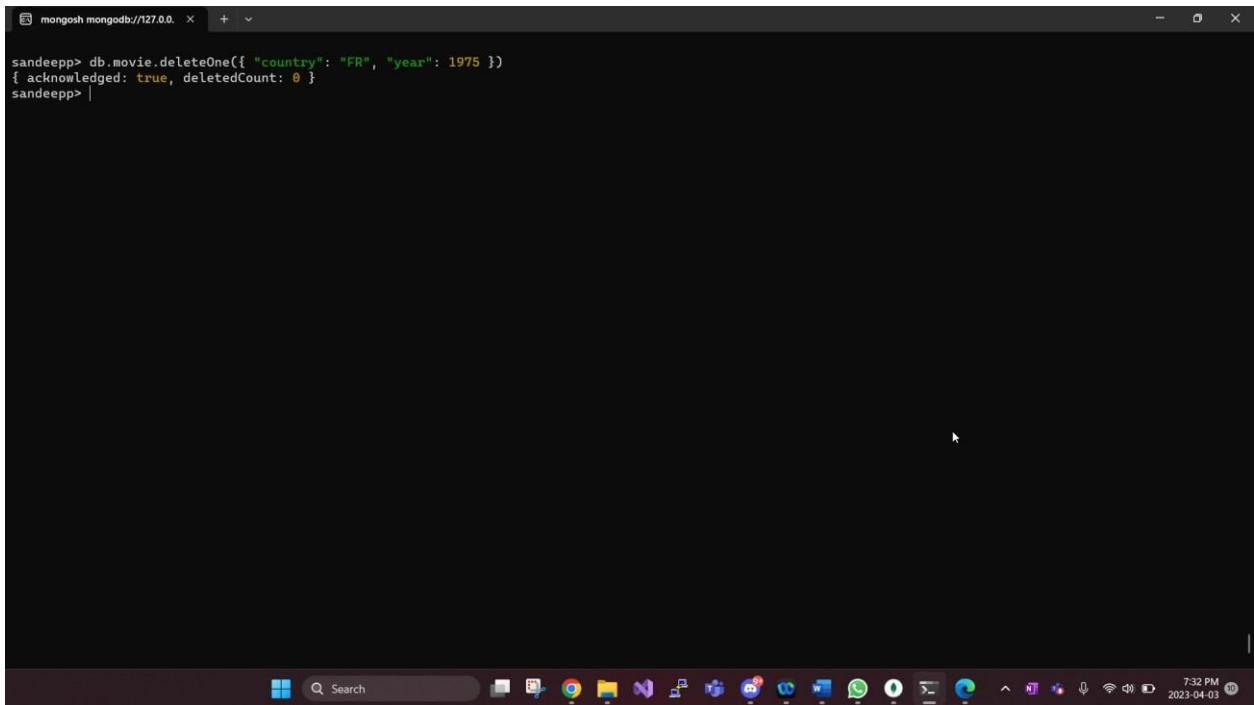


```
sandeep> db.movie.find({ "director": "Hitchcock", "year": 1, "_id": 0 }).sort({ "year": 1 })
sandeep> db.movie.find({ "director": "Hitchcock", "year": 1, "_id": 0 }).sort({ "year": 1 })
sandeep>
sandeep> db.movie.find({ "director": "Hitchcock", "year": 1, "_id": 0 }).sort({ "year": 1 })
sandeep> db.movies.deleteMany({ "year": { "$lt": 1980 } })
{ acknowledged: true, deletedCount: 0 }
sandeep>
```

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

we use the deleteOne() method to delete the movie which is from country "FR" and year 1975. The query object { "country": "FR", "year": 1975 } specifies the condition for selecting the document to delete.

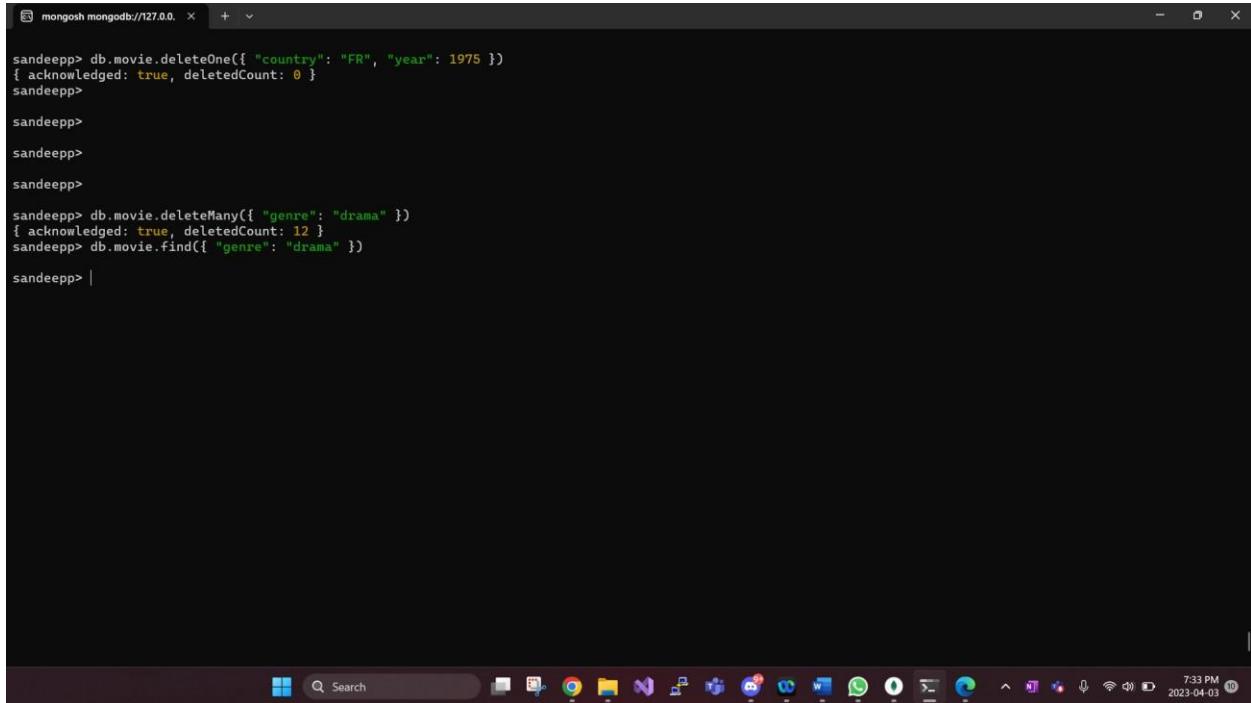
```
db.movie.deleteOne({ "country": "FR", "year": 1975 })
```



```
sandepp> db.movie.deleteOne({ "country": "FR", "year": 1975 })
{ acknowledged: true, deletedCount: 0 }
sandepp> |
```

## 8 Delete the movies with genre is drama.

we use the deleteMany() method to delete the movies with the genre "drama". The query object { "genre": "drama" } specifies the condition for selecting the documents to delete. **db.movie.deleteMany({“genre”: “drama”})**



```
mongosh mongodb://127.0.0.1:27017
sandeep> db.movie.deleteOne({ "country": "FR", "year": 1975 })
{ acknowledged: true, deletedCount: 0 }
sandeep>
sandeep>
sandeep>
sandeep> db.movie.deleteMany({ "genre": "drama" })
{ acknowledged: true, deletedCount: 12 }
sandeep> db.movie.find({ "genre": "drama" })
sandeep> |
```

## 9 Delete the movies with year is 1990 and genre is Action.

we use the deleteMany() method to delete the movies with year 1990 and genre "Action". The query object { "year": 1990, "genre": "Action" } specifies the condition for selecting the documents to delete. in this I have already deleted so it is showing 0 the command is **db.movie.deleteMany({“year”: 1990, “genre”: “action”})**

```
sandeep> db.movie.deleteMany({ "year": 1990, "genre": "action" })
{ acknowledged: true, deletedCount: 0 }
sandeep>
```

## 10 Delete the movies with last\_name “Burton”.

we use the `deleteMany()` method to delete the movies with last name "Burton". The query object `{ "last_name": "Burton" }` specifies the condition for selecting the documents to delete. `db.movie.deleteMany({"director.last_name": "Burton"})`

```
sandeep: mongosh mongodb://127.0.0.1:27017
+ 
sandeep> db.movie.deleteMany({ "director.last_name": "Burton" })
{ acknowledged: true, deletedCount: 0 }
sandeep>
```

