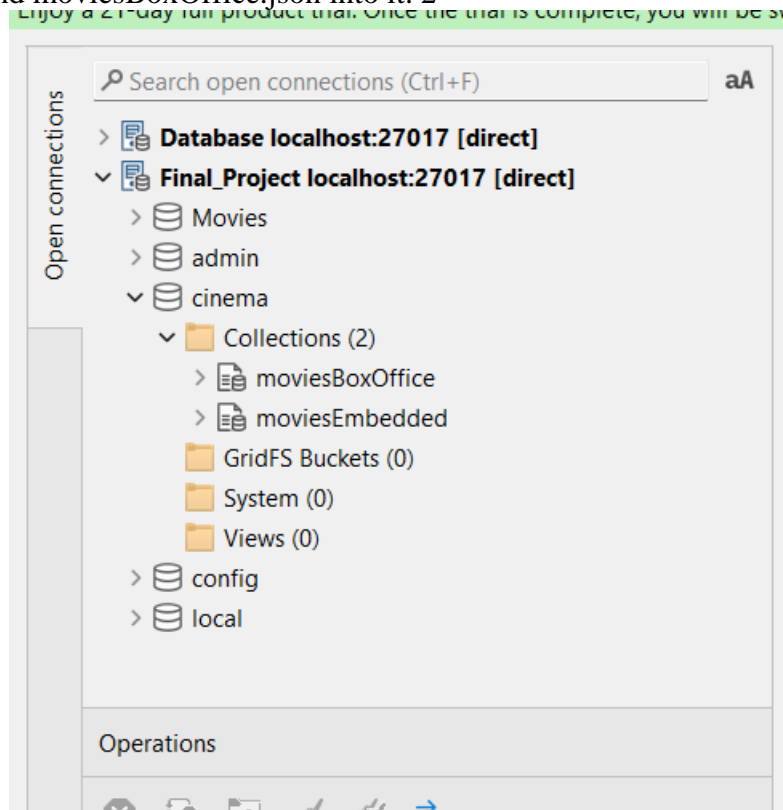


PERSONAL PROJECT-2

Part 1: MongoDB Database

Create a MongoDB database named *cinema* and create the two collections *moviesEmbedded.json* and *moviesBoxOffice.json* into it. 2



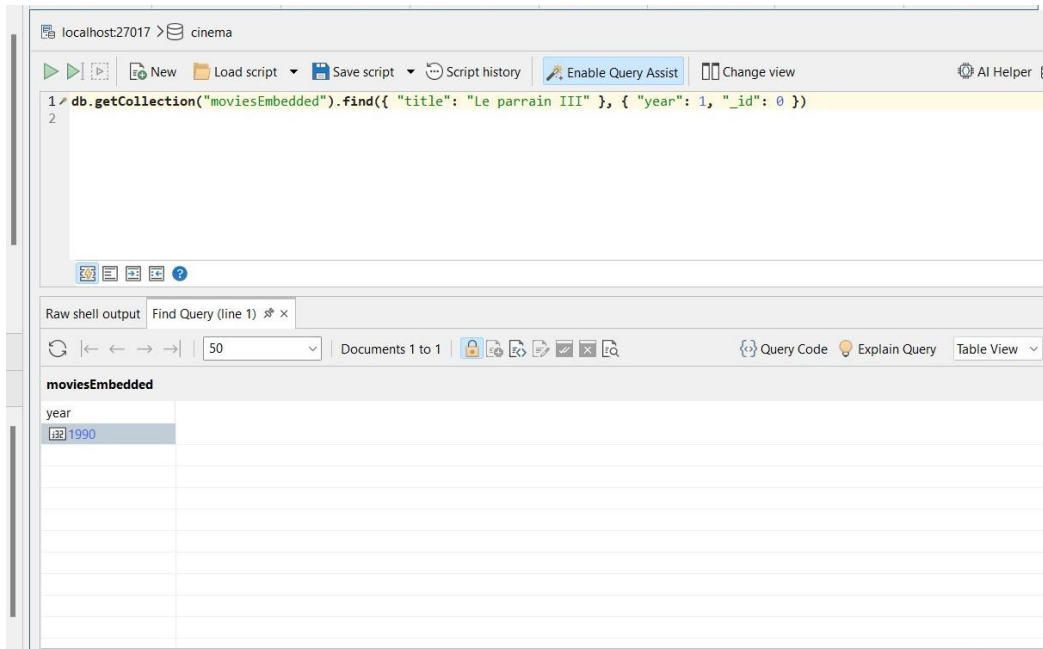
Part 2: MongoDB Queries

Use the collections you created to answer the following queries:

a. What was the release year of the movie “Le parrain III”.

Query:

```
db.getCollection("moviesEmbedded").find(
  { "title": "Le parrain III" }, { "year": 1, "_id": 0 })
```

Snapshots:

b. The titles of the movies released between 1980 and 1990.

Query:

```
db.getCollection("moviesEmbedded").find(
{"year": { $gte: 1980, $lte: 1990 } }, { "title": 1, "_id": 0 })
```

Snapshots:

localhost:27017 > cinema

New

Load script

Save script

Script history

Enable Query Assist

Change view

1 db.getCollection("moviesEmbedded").find({ "year": { \$gte: 1980, \$lte: 1990 } }, { "title": 1, "_id": 0 })

2

Raw shell outputFind Query (line 1) ×

←

→

50

Documents 1 to 2


Query Code


Explain Query

Tabl

moviesEmbedded

title

The Godfather

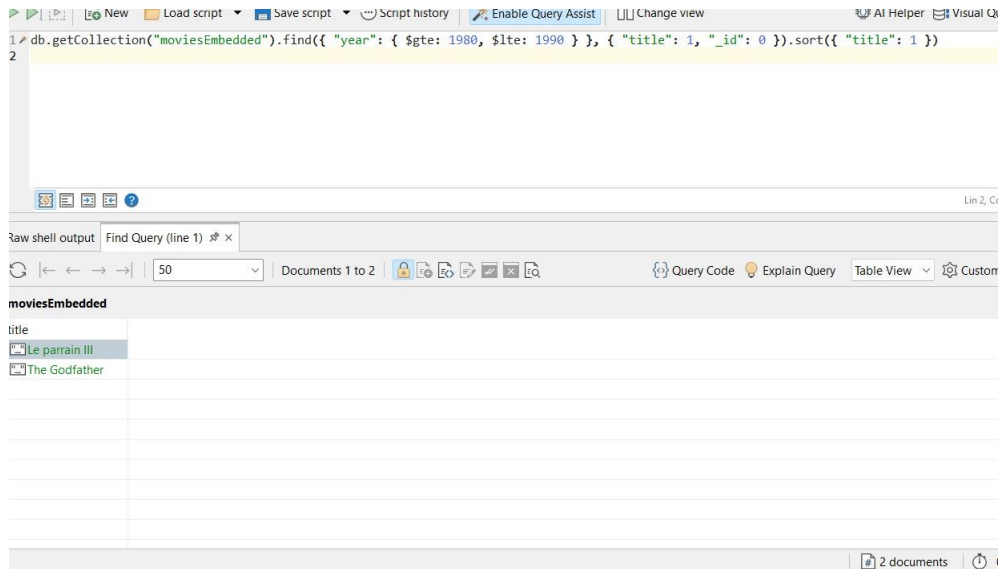
Le parrain III

1 document selected

Cou

c. Same query in b. with the titles sorted in alphabetical order.Query:

```
db.getCollection("moviesEmbedded").find({ "year": { $gte: 1980, $lte: 1990 } }, { "title": 1, "_id": 0
}).sort({ "title": 1 })
```

Snapshot:

Raw shell output: Find Query (line 1) ✕

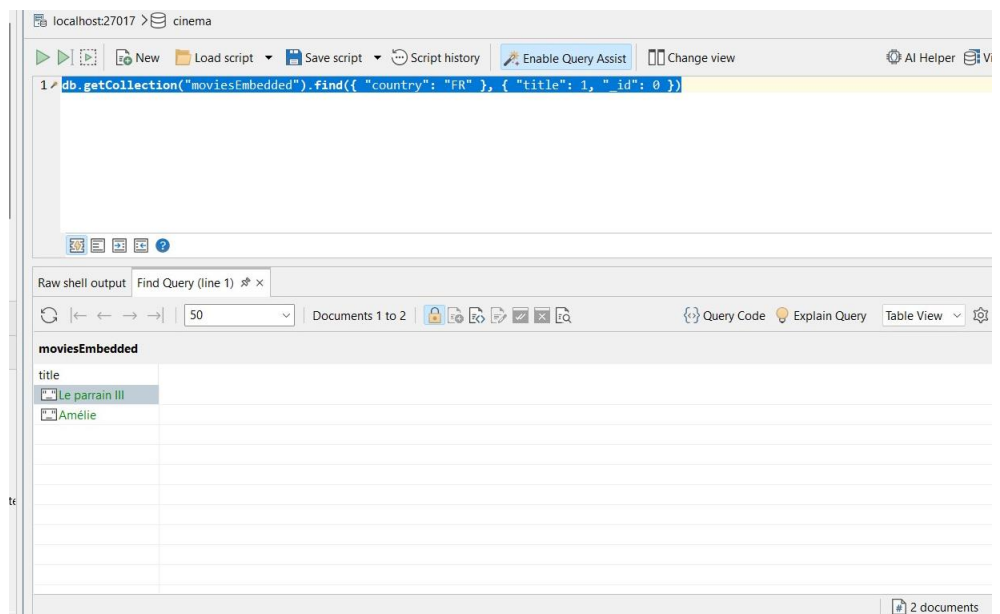
moviesEmbedded

title
Le parrain III
The Godfather

2 documents

d. The titles of all the French movies.**Query:**

```
db.getCollection("moviesEmbedded").find({ "country": "FR" }, { "title": 1, "_id": 0 })
```

Snapshot:

localhost:27017 > cinema

Raw shell output: Find Query (line 1) ✕

moviesEmbedded

title
Le parrain III
Amélie

2 documents

e. The title of the movies with genre “crime” or “drama”.Query:

```
db.getCollection("moviesEmbedded").find({ "genre": { $in: ["Crime", "Drama"] } }, { "title": 1, "_id": 0 })
```

Snapshot:

The screenshot shows a MongoDB query interface. The query is: `1 db.getCollection("moviesEmbedded").find({ "genre": { $in: ["Crime", "Drama"] } }, { "title": 1, "_id": 0 })`. The results are displayed in a table view for the `moviesEmbedded` collection. The table has two columns: `title` and `_id`. The results are:

title	_id
The Godfather	1
Le parrain III	2
Pulp Fiction	3
La La Land	4
The Shawshank ...	5
Forrest Gump	6
Inglourious Bast...	7
The Dark Knight	8

f. The names and birth dates of the directors of French movies.Query:

```
db.getCollection("moviesEmbedded").find({ "country": "FR" }, { "director.first_name": 1, "director.last_name": 1, "director.birth_date": 1, "_id": 0 })
```

Snapshot:

The screenshot shows a MongoDB query interface. The query is: `1 db.getCollection("moviesEmbedded").find({ "country": "FR" }, { "director.first_name": 1, "director.last_name": 1, "director.birth_date": 1, "_id": 0 })`. The results are displayed in a table view for the `moviesEmbedded` collection. The table has two columns: `director` and `_id`. The results are:

director	_id
{ "first_name": "Jean", "last_name": "Pierre", "birth_date": "1929-06-20" }	1
{ "first_name": "Jean", "last_name": "Pierre", "birth_date": "1929-06-20" }	2
{ "first_name": "Jean", "last_name": "Pierre", "birth_date": "1929-06-20" }	3
{ "first_name": "Jean", "last_name": "Pierre", "birth_date": "1929-06-20" }	4
{ "first_name": "Jean", "last_name": "Pierre", "birth_date": "1929-06-20" }	5
{ "first_name": "Jean", "last_name": "Pierre", "birth_date": "1929-06-20" }	6
{ "first_name": "Jean", "last_name": "Pierre", "birth_date": "1929-06-20" }	7
{ "first_name": "Jean", "last_name": "Pierre", "birth_date": "1929-06-20" }	8

g. The title of the movies of which Sofia Coppola is one of the actors.Query:

db.getCollection("moviesEmbedded").find({ "actors.first_name": "Sofia", "actors.last_name": "Coppola" }, { "title": 1, "_id": 0 })

Snapshot:

The screenshot shows a MongoDB query interface. The query is: `1 db.getCollection("moviesEmbedded").find({ "actors.first_name": "Sofia", "actors.last_name": "Coppola" }, { "title": 1, "_id": 0 })`. The results are displayed in a table view for the `moviesEmbedded` collection. The table has two columns: `title` and `genre`. The results are:

title	genre
The Godfather	
Le parrain III	

At the bottom, it indicates "2 documents".

h. The title and the genres of the movies of which the director is George Wanganga.Query:

db.getCollection("moviesEmbedded").find({"director.first_name": "George", "director.last_name": "Wanganga" }, { "title": 1, "genre": 1, "_id": 0 })

Snapshot:

The screenshot shows a MongoDB query interface. The query is: `1 db.getCollection("moviesEmbedded").find({ "director.first_name": "George", "director.last_name": "Wanganga" }, { "title": 1, "genre": 1, "_id": 0 })`. The results are displayed in a table view for the `moviesEmbedded` collection. The table has two columns: `title` and `genre`. The results are:

title	genre
The Godfather	[2 elements]

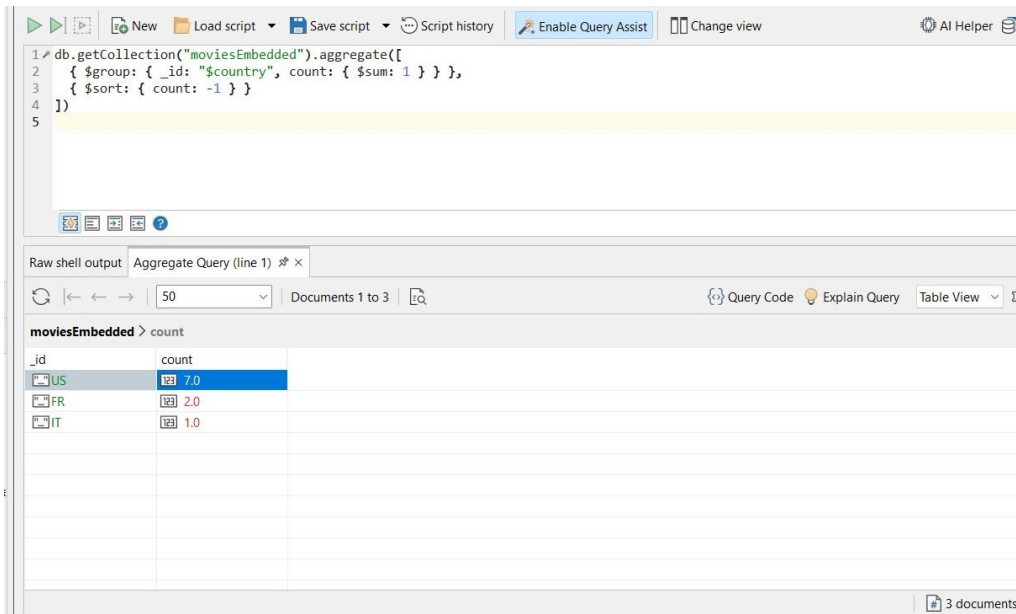
At the bottom, it indicates "1 document".

Part 3: MongoDB Aggregations

a. Get the number of movies per country. Show the result in descending order. Query:

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

Snapshots:



The screenshot shows the MongoDB Compass interface. The query editor at the top contains the following aggregate query:

```
1 db.getCollection("moviesEmbedded").aggregate([
2   { $group: { _id: "$country", count: { $sum: 1 } } },
3   { $sort: { count: -1 } }
4 ])
5
```

Below the query editor, the 'Raw shell output' tab is selected, showing the results of the query. The results are displayed in a table with columns '_id' and 'count'.

_id	count
US	7.0
FR	2.0
IT	1.0

The table shows 3 documents. The status bar at the bottom indicates '3 documents'.

b. Get the name of the actor with the role "Mary Corleone" in the movie "Le parrain III".

Query:

```
db.getCollection("moviesEmbedded").aggregate([{$match: { "title": "Le parrain III" } },{$unwind: "$actors" },{$match: { "actors.role": "Mary Corleone" } },{$project: { _id: 0,"actor_name": { $concat: ["$actors.first_name", " ", "$actors.last_name"] } } }])
```








The screenshot shows the MongoDB Compass interface. At the top, there's a toolbar with icons for New, Load script, Save script, Script history, Enable Query Assist, and Change view. The main area displays an aggregate query in a shell:



```
1 db.getCollection("moviesEmbedded").aggregate([
2   { $match: { "title": "Le parrain III" } },
3   { $unwind: "$actors" },
4   { $match: { "actors.role": "Mary Corleone" } },
5   { $project: { _id: 0, "actor_name": { $concat: ["$actors.first_name", " ", "$actors.last_name"] } } }
6 ])
7
```

Below the query, there's a toolbar with icons for Raw shell output, Table View, and a help icon. The 'Raw shell output' tab is active, showing the results of the query. The results are displayed in a table with two columns: 'actor_name' and 'movie_title'. The first row shows 'Sofia Coppola' and 'Le parrain III'.

Query:

Snapshots:













```

1 db.getCollection("moviesEmbedded").aggregate([
2   { $unwind: "$actors" },
3   { $group: { _id: "$title", count: { $sum: 1 } } },
4   { $sort: { count: -1 } }
5 ])
6




```











Lin 6, Col 1

Raw shell output











Aggregate Query (line 1)






Documents 1 to 10
 

moviesEmbedded > count

_id	count
 The Dark Knight	129 2.0
 The Shawshank ...	129 2.0
 Inglourious Bast...	129 2.0
 La La Land	129 2.0
 Pulp Fiction	129 2.0
 Amélie	129 2.0
 Forrest Gump	129 2.0
 Le parrain III	129 1.0
 Inception	129 1.0
 The Godfather	129 1.0

10 documents
 

d. Get the average number of actors per film.**Query:**

```
db.getCollection("moviesEmbedded").aggregate([{$unwind: "$actors"},{$group: { _id: null, totalActors: { $sum: 1 }, totalMovies: { $sum: 1 } }},{$project: { _id: 0, averageActorsPerFilm: { $divide: ["$totalActors", "$totalMovies"] } } }])
```

Snapshots:

Raw shell output: Aggregate Query (line 1)

moviesEmbedded
averageActorsPerFilm
1.0

1 document

e. Find the number of tickets sold for "Le parrain III." Query:

```
db.getCollection("moviesBoxOffice").findOne({ "_id": "2" })
```

Snapshots:

Raw shell output: Find Query (line 1)

moviesBoxOffice	box_office
2	8000000

1 document