// 1. Find all movies with full information from the 'movies' collection that released in the year 1893.

db.movies.find({year: 1893})

// 2. Find all movies with full information from the 'movies' collection that have a runtime greater than 120 minutes.

db.movies.find({runtime:{$gt:120}})

// 3. Find all movies with full information from the 'movies' collection that have "Short" genre.

db.movies.find({genres:{$in:['Short']}})

// 4. Retrieve all movies from the 'movies' collection that were directed by "William K.L. Dickson" and include complete information for each movie.

db.movies.find({directors:{$in:['William K.L. Dickson']}})

// 5. Retrieve all movies from the 'movies' collection that were released in the USA and include complete information for each movie.

db.movies.find({countries: {$in: ['USA']}})

// 6. Retrieve all movies from the 'movies' collection that have complete information and are rated as "UNRATED".

db.movies.find({rated: 'UNRATED'})

// 7. Retrieve all movies from the 'movies' collection that have complete information and have received more than 1000 votes on IMDb.

db.movies.find({"imdb.votes": {$gt:1000 }})

// 8. Retrieve all movies from the 'movies' collection that have complete information and have an IMDb rating higher than 7.

db.movies.find({"imdb.rating": {$gt:7 }})

// 9. Retrieve all movies from the 'movies' collection that have complete information and have a viewer rating higher than 4 on Tomatoes.

db.movies.find({"tomatoes.viewer.rating": {$gt:4 }})

// 10. Retrieve all movies from the 'movies' collection that have received an award.

db.movies.find({awards: {$exists: true}})

// 11. Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB that have at least one nomination.

db.movies.find({

"awards.nominations":{$exists: true, $gte: 1},

title:{$exists: true, $ne: null},

languages:{$exists: true, $ne: null},

released:{$exists: true, $ne: null},

directors:{$exists: true, $ne: null},

writers:{$exists: true, $ne: null},

year:{$exists: true, $ne: null},

genres:{$exists: true, $ne: null},

runtime:{$exists: true, $ne: null},

cast:{$exists: true, $ne: null},

countries:{$exists: true, $ne: null}

})

// 12. Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB with cast including "Charles Kayser".

db.movies.find({

cast: {$in: ["Charles Kayser"]}

},

{

title: 1,

languages: 1,

released: 1,

directors: 1,

writers: 1,

countries: 1

})

db.movies.find({cast: {$in: ['Charles Kayser']}})

// 13. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that released on May 9, 1893.

db.movies.find({

released: ISODate('1893-05-09T00:00:00.000Z')

},

{

title: 1,

languages: 1,

released: 1,

directors: 1,

writers: 1,

countries: 1

})

// 14. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that have a word "scene" in the title.

db.movies.find({

title: {$regex: /scene/i}

},

{

title: 1,

languages: 1,

released: 1,

directors: 1,

writers: 1,

countries: 1

})

// 15. Find all movies with title, languages, released, directors, viewer, writers, countries from the 'movies' collection in MongoDB

//that have a viewer rating of at least 3 and less than 4 on Tomatoes.

db.movies.find({

"tomatoes.viewer.rating": { $lte: 3 }

},

{

title: 1,

languages: 1,

released: 1,

directors: 1,

writers: 1,

countries: 1

})

// 16. Retrieve all movies with title, languages, released, year, directors, writers, countries from the 'movies' collection in MongoDB that released before the year 1900.

db.movies.find({

released: { $lt: ISODate('1900-01-01T00:00:00.000Z') }

},

{

title: 1,

languages: 1,

released: 1,

directors: 1,

writers: 1,

countries: 1

})

// 17. Find all movies with title, languages, fullplot, released, directors, writers, countries from the 'movies' collection in MongoDB that have a fullplot containing the word "fire".

db.movies.find({

fullplot: {$regex: /fire/i}

},

{

title: 1,

languages: 1,

released: 1,

directors: 1,

writers: 1,

countries: 1,

fullplot:1

})

// 18. Return all movies with title, languages, plot, released, directors, writers, and countries from the 'movies' collection in MongoDB where the word "beer" mentioned in the plot.

db.movies.find({

fullplot: {$regex: /beer/i}

},

{

title: 1,

languages: 1,

released: 1,

directors: 1,

writers: 1,

countries: 1,

fullplot:1

})

// 19. Return all movies with title, languages, fullplot, released, directors, writers, and countries from the 'movies' collection in MongoDB where the word "metal" mentioned in the fullplot.

db.movies.find({

fullplot: {$regex: /metal/i}

},

{

title: 1,

languages: 1,

released: 1,

directors: 1,

writers: 1,

countries: 1,

fullplot:1

})

// 20. Find all movies with title, languages, released, runtime, directors, writers, countries from the 'movies' collection in MongoDB that have a runtime between 60 and 90 minutes.

db.student.find($and:[{u1:{$gt:30}},{u1:{$lt:60}}])

db.movies.find({

$and:[

{runtime: {$gt: 60}},

{runtime: {$lt: 90}}

]

},

{

title: 1,

languages: 1,

released: 1,

directors: 1,

writers: 1,

countries: 1,

fullplot:1

})

// 21. Find all movies with title, languages, released, runtime, directors, writers, countries, imdb from the 'movies' collection in MongoDB for the top 5 movies with the highest IMDb ratings.

db.movies.find({},

{

title: 1,

languages: 1,

released: 1,

directors: 1,

writers: 1,

countries: 1,

fullplot:1

}).sort({"imdb.rating": -1}).limit(5)

// 22. Find all movies from the 'movies' collection in MongoDB with the average runtime of movies released in each country.

db.movies.aggregate([

{

$unwind: "$countries"

},

{

$group: {

\_id: "$countries",

averageRuntime: { $avg: "$runtime" }

}

},

{

$sort: { averageRuntime: -1 }

}

])

// 23. Find from the 'movies' collection in MongoDB with the most common genre among the movies.

db.movies.aggregate([

{

$unwind: "$genres"

},

{

$group: {

\_id: "$genres",

genreCount: { $sum: 1 }

}

},

{

$sort: { genreCount: -1 }

},

{

$limit:1

}

])

// 24. Find the movies released in the year with the highest average IMDb rating from the 'movies' collection in MongoDB.

// -- This Chat-gpt Query, I couldn't figure this out by myslef and still XD

db.movies.aggregate([

{

$project: {

title: 1,

imdbRating: "$imdb.rating",

year: { $year: "$released" } // Extract the year from the 'released' field

}

},

{

$group: {

\_id: "$year", // Group by the year

avgRating: { $avg: "$imdbRating" } // Calculate the average IMDb rating for each year

}

},

{

$sort: { avgRating: -1 } // Sort by average IMDb rating in descending order

},

{

$limit: 1 // Get the year with the highest average IMDb rating

},

{

$lookup: {

from: "movies", // Join back with the 'movies' collection

localField: "\_id", // Match the year (\_id from previous group)

foreignField: { $year: "$released" }, // Match it with the year from the released field

as: "moviesInYear"

}

},

{

$unwind: "$moviesInYear"

},

{

$project: {

title: "$moviesInYear.title",

imdbRating: "$moviesInYear.imdb.rating",

year: "$\_id"

}

}

])

// 25. Find the top 10 directors with the most movies from the 'movies' collection in MongoDB.

db.movies.aggregate([

{

$unwind: "$directors"

},

{

$group: {

\_id: "$directors",

movieCount: {$sum: 1}

}

},

{

$sort: {movieCount:-1}

},

{

$limit:10

}

])

// 26. Write a query in MongoDB to find the average IMDb rating for movies with different ratings (e.g., 'PG', 'R', 'G') from the 'movies' collection.

db.movies.aggregate([

{

$group: {

\_id: "$rated",

avgRating: {$avg: "$imdb.rating"}

}

},

{

$sort: {avgRating:-1}

}

])

// 27. Write a query in MongoDB to find the oldest movie with an award win from the 'movies' collection.

db.movies.find({"awards.wins": { $gt: 0 } }).sort({ released: 1 }).limit(1)

// 28. Write a query in MongoDB to find the movie with the highest IMDb rating// and viewer rating on Tomatoes from the 'movies' collection.

db.movies.find({

"imdb.rating": { $exists: true },

"tomatoes.viewer.rating": { $exists: true }

}).sort({

"imdb.rating": -1,

"tomatoes.viewer.rating": -1

}).limit(1)