// 1. Find all movies directed by Christopher Nolan.

db.movies.find({ director: "Christopher Nolan" });

// 2. Find movies that include the genre "Action" and sort (descending) them by year.

db.movies.find({ genres: "Action" }).sort({ year: -1 });

// 3. Find movies with an IMDb rating greater than 8 and return only the title and IMDb information.

db.movies.find(

{ "imdb.rating": { $gt: 8 } },

{ title: 1, imdb: 1, \_id: 0 }

);

// 4. Find movies that starred both "Tom Hanks" and "Tim Allen".

db.movies.find({ cast: { $all: ["Tom Hanks", "Tim Allen"] } });

// 5. Find movies that starred both and only "Tom Hanks" and "Tim Allen".

db.movies.find({ cast: { $size: 2, $all: ["Tom Hanks", "Tim Allen"] } });

// 6. Find comedy movies that are directed by Steven Spielberg.

db.movies.find({ genres: "Comedy", director: "Steven Spielberg" });

// 1. Add a new field "available\_on" with the value "Sflix" to "The Matrix".

db.movies.updateOne(

{ title: "The Matrix" },

{ $set: { available\_on: "Sflix" } }

);

// 2. Increment the metacritic of "The Matrix" by 1.

db.movies.updateOne(

{ title: "The Matrix" },

{ $inc: { metacritic: 1 } }

);

// 3. Add a new genre "Gen Z" to all movies released in the year 1997.

db.movies.updateMany(

{ year: 1997 },

{ $addToSet: { genres: "Gen Z" } }

);

// 4. Increase IMDb rating by 1 for all movies with a rating less than 5.

db.movies.updateMany(

{ "imdb.rating": { $lt: 5 } },

{ $inc: { "imdb.rating": 1 } }

);

// 1. Delete a comment with a specific ID.

db.comments.deleteOne({ \_id: ObjectId("SPECIFIC\_COMMENT\_ID") });

// 2. Delete all comments made for "The Matrix".

db.comments.deleteMany({ movieTitle: "The Matrix" });

// 3. Delete all movies that do not have any genres.

db.movies.deleteMany({ genres: { $exists: true, $size: 0 } });

// 1. Aggregate movies to count how many were released each year and display from the earliest year to the latest.

db.movies.aggregate([

{ $group: { \_id: "$year", count: { $sum: 1 } } },

{ $sort: { \_id: 1 } }

]);

// 2. Calculate the average IMDb rating for movies grouped by director and display from highest to lowest.

db.movies.aggregate([

{ $group: { \_id: "$director", avgRating: { $avg: "$imdb.rating" } } },

{ $sort: { avgRating: -1 } }

]);