
🎬 Capstone Project – Movie Streaming Platform

A company like **Netflix/Hotstar** wants to manage data for **users, movies, subscriptions, and watch history**.

---

1. Database & Collections Setup

```
use movieDB

// Users Collection
db.users.insertMany([
  { _id: 1, name: "Rahul Sharma", email: "rahul@example.com", city: "Bangalore", plan: "Premium" },
  { _id: 2, name: "Priya Singh", email: "priya@example.com", city: "Delhi", plan: "Basic" },
  { _id: 3, name: "Aman Kumar", email: "aman@example.com", city: "Hyderabad", plan: "Standard" }
]);

// Movies Collection
db.movies.insertMany([
  { _id: 101, title: "Inception", genre: "Sci-Fi", year: 2010, rating: 8.8 },
  { _id: 102, title: "3 Idiots", genre: "Comedy", year: 2009, rating: 8.4 },
  { _id: 103, title: "Bahubali", genre: "Action", year: 2015, rating: 8.1 },
  { _id: 104, title: "The Dark Knight", genre: "Action", year: 2008, rating: 9.0 },
  { _id: 105, title: "Dangal", genre: "Drama", year: 2016, rating: 8.5 }
]);

// Subscriptions Collection
db.subscriptions.insertMany([
  { user_id: 1, start_date: ISODate("2025-01-01"), end_date: ISODate("2025-12-31"), amount: 999 },
  { user_id: 2, start_date: ISODate("2025-02-01"), end_date: ISODate("2025-07-31"), amount: 499 },
  { user_id: 3, start_date: ISODate("2025-01-15"), end_date: ISODate("2025-10-15"), amount: 799 }
]);

// Watch History Collection
db.watchHistory.insertMany([
  { user_id: 1, movie_id: 101, watch_date: ISODate("2025-02-10") },
  { user_id: 1, movie_id: 102, watch_date: ISODate("2025-02-12") },
  { user_id: 2, movie_id: 103, watch_date: ISODate("2025-02-11") },
  { user_id: 3, movie_id: 104, watch_date: ISODate("2025-02-13") },
  { user_id: 3, movie_id: 105, watch_date: ISODate("2025-02-14") }
]);
```

---

2. Capstone Exercises

A. CRUD Operations

1. Insert a new user in Mumbai with a `"Standard"` plan.
2. Update `"Bahubali"` rating to 8.3.
3. Delete the movie `"3 Idiots"`.
4. Find all users with `"Premium"` plan.

B. Indexing

5. Create a unique index on `users.email`.
6. Create a compound index on `movies.genre` and `rating`.
7. Verify indexes using `getIndexes()`.
8. Write a query that benefits from this compound index.
9. Write a query that forces a **COLLSCAN**.

C. Aggregation Framework

10. Count how many movies exist in each genre.
11. Find the top 2 highest-rated movies.
12. Calculate the average subscription amount per plan type.
13. Show the total watch count per movie.
14. Find the city with the maximum number of Premium users.
15. Find the most popular genre by total watch count.

D. `$lookup` (Joins)

16. Show all watch history with **user name and movie title**.
17. List all movies watched by `"Rahul Sharma"`.
18. Show each user with their subscription details.
19. Find all users who watched movies released after 2010.
20. For each movie, list all the users who watched it.

E. Advanced Analytics

21. Find users who watched more than 2 movies.
22. Find total revenue collected from subscriptions.
23. Find users whose subscription will expire in the next 30 days.
24. Show the most-watched movie overall.
25. Show the least-watched genre.

 This **Movie Streaming scenario** is highly relatable:

- Real-world entities (**users, movies, subscriptions, watch history**)
- Covers **CRUD, indexing, aggregation, joins, and business reporting**
- Mimics analytics queries streaming companies actually run
