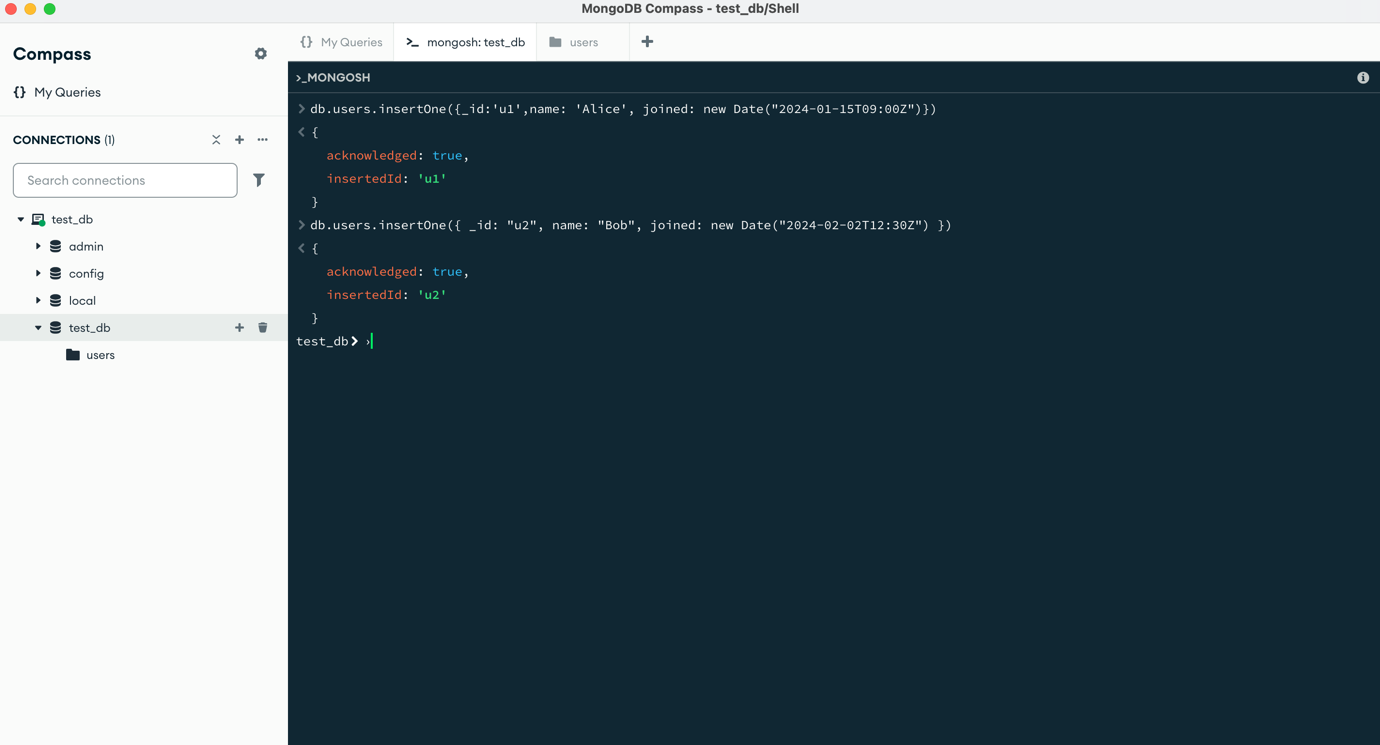
# Creating Collections

1. db.createCollection("users")
2. db.createCollection("follows")
3. db.createCollection("posts")

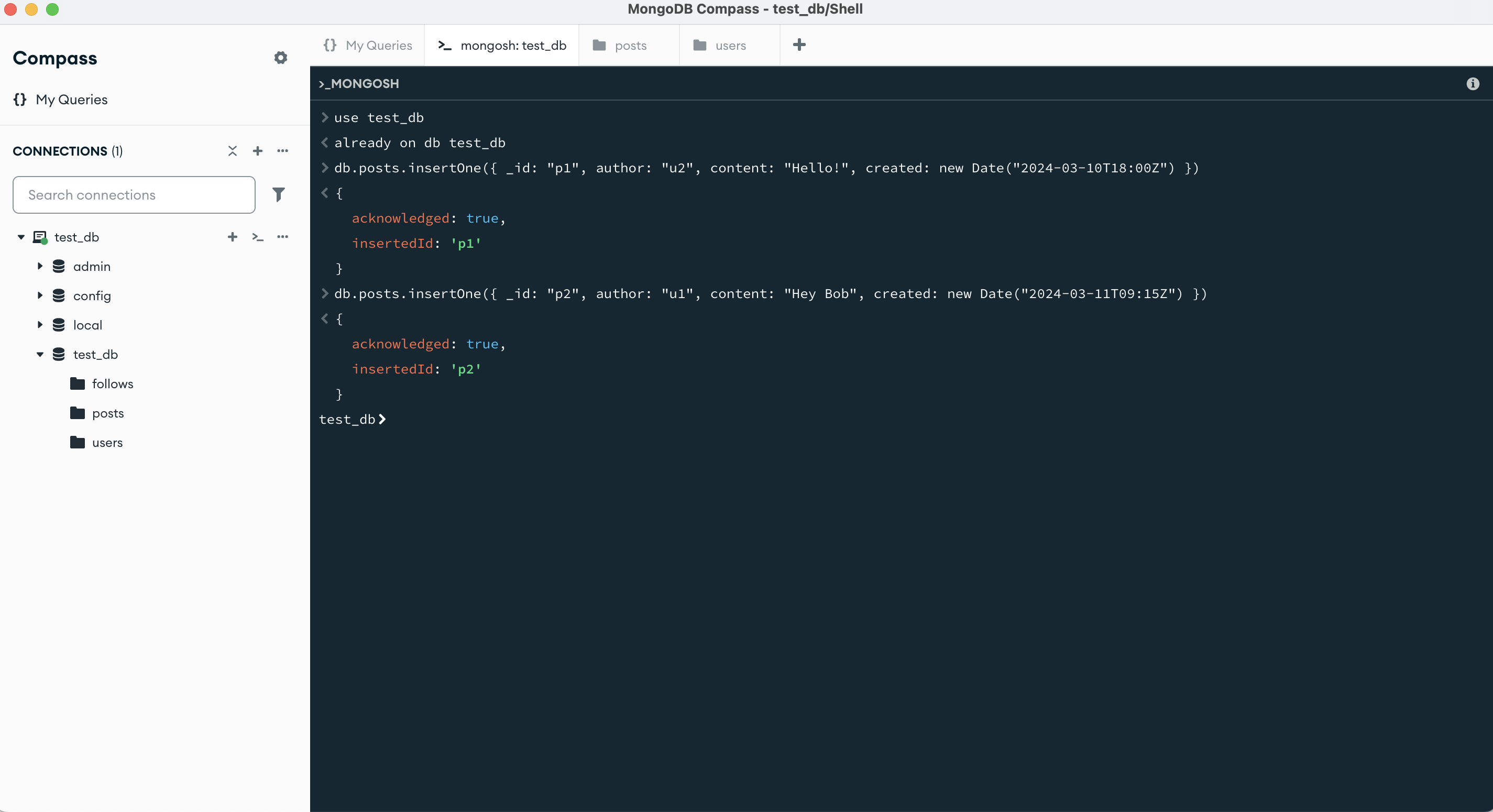
# Adding users

1. db.users.insertOne({ \_id: "u1", name: "Alice", joined: new Date("2024-01-15T09:00Z") })
2. db.users.insertOne({ \_id: "u2", name: "Bob", joined: new Date("2024-02-02T12:30Z") })



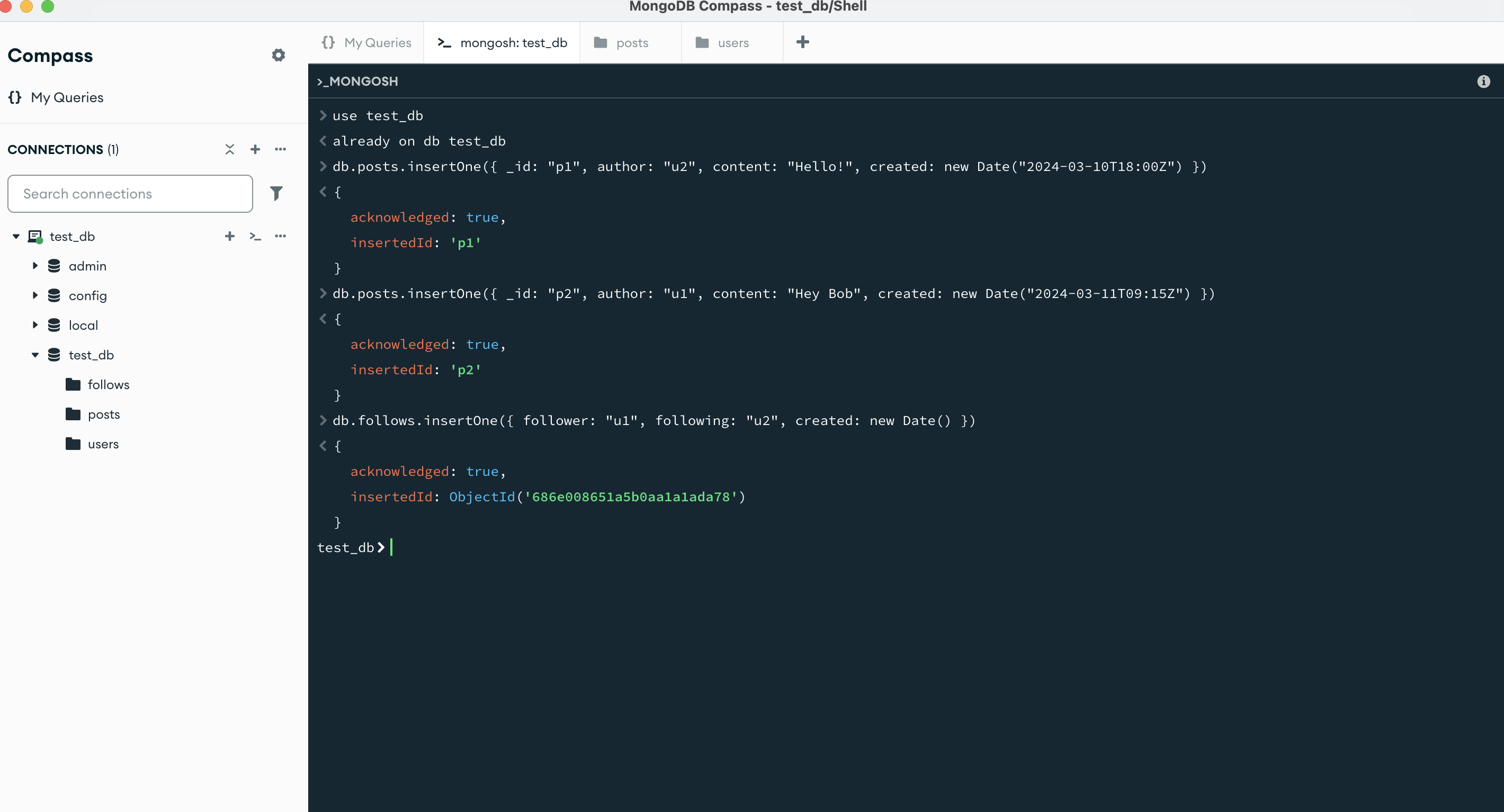
# Adding posts

1. db.posts.insertOne({ \_id: "p1", author: "u2", content: "Hello!", created: new Date("2024-03-10T18:00Z") })
2. db.posts.insertOne({ \_id: "p2", author: "u1", content: "Hey Bob", created: new Date("2024-03-11T09:15Z") })



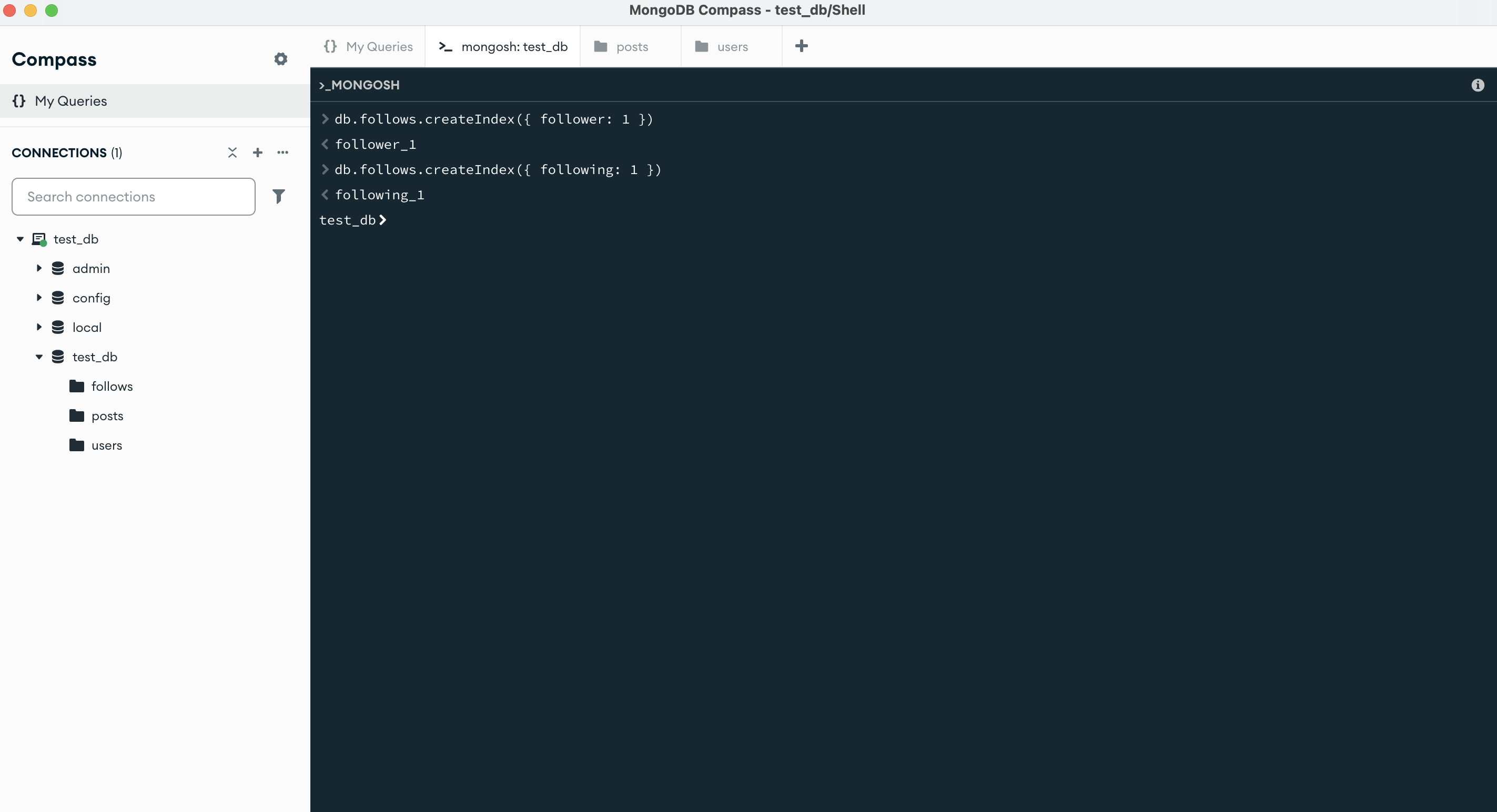
# Adding Follows

1. db.follows.insertOne({ follower: "u1", following: "u2", created: new Date() })



Adding Indexing for Query Performance

1. db.follows.createIndex({ follower: 1 })
2. db.follows.createIndex({ following: 1 })
3. db.posts.createIndex({ author: 1, created: -1 })



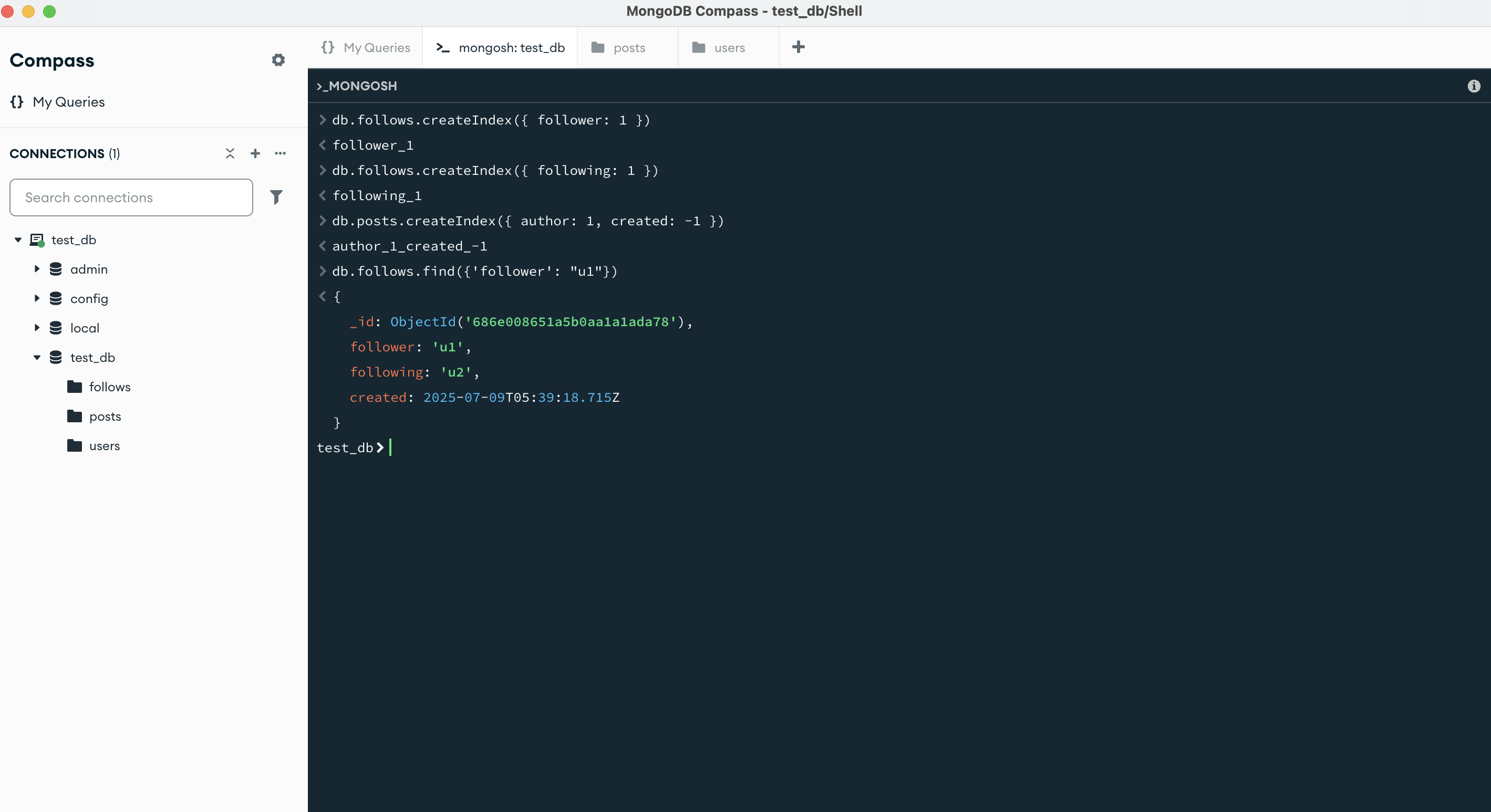
These indexes support efficient lookups of followers/following, and allow reverse-chronological paging of posts

# Questions.

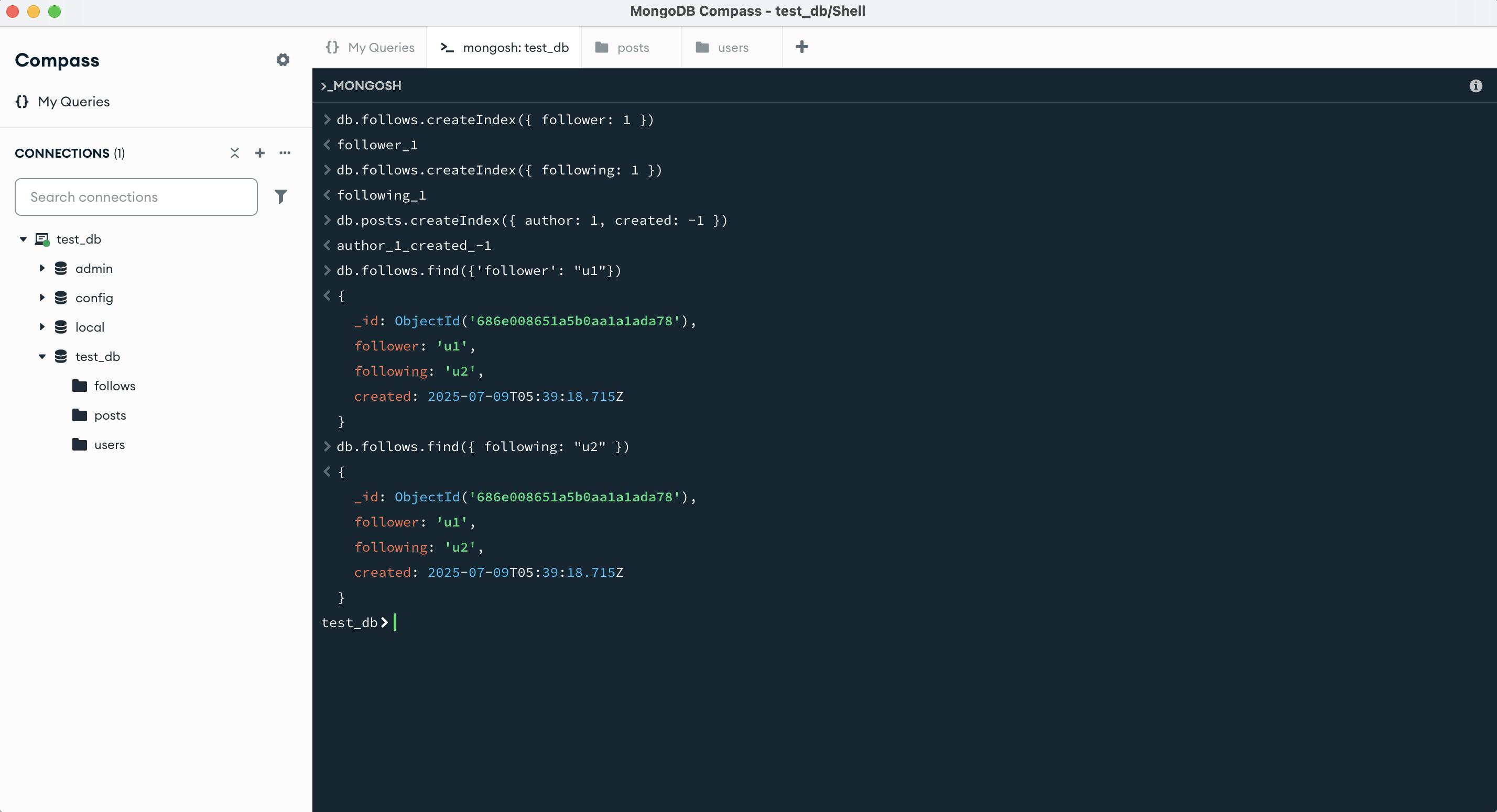
1. Design your collection schemas (briefly, in 5–7 lines) so that:
2. You can list each user’s followers (and who they follow) efficiently.
3. You can page through posts in reverse-chronological order.

## Listing which user follows (u1)

1. db.follows.find({ follower: "u1" })



1. db.follows.find({ following: "u2" })



## Page through posts by u1, newest first (page 1):

db.posts.aggregate([

{ $match: { author: "u1" } },

{

$lookup: {

from: "users", // the collection to join

localField: "author", // field in posts

foreignField: "\_id", // field in users

as: "author\_info" // output array name

}

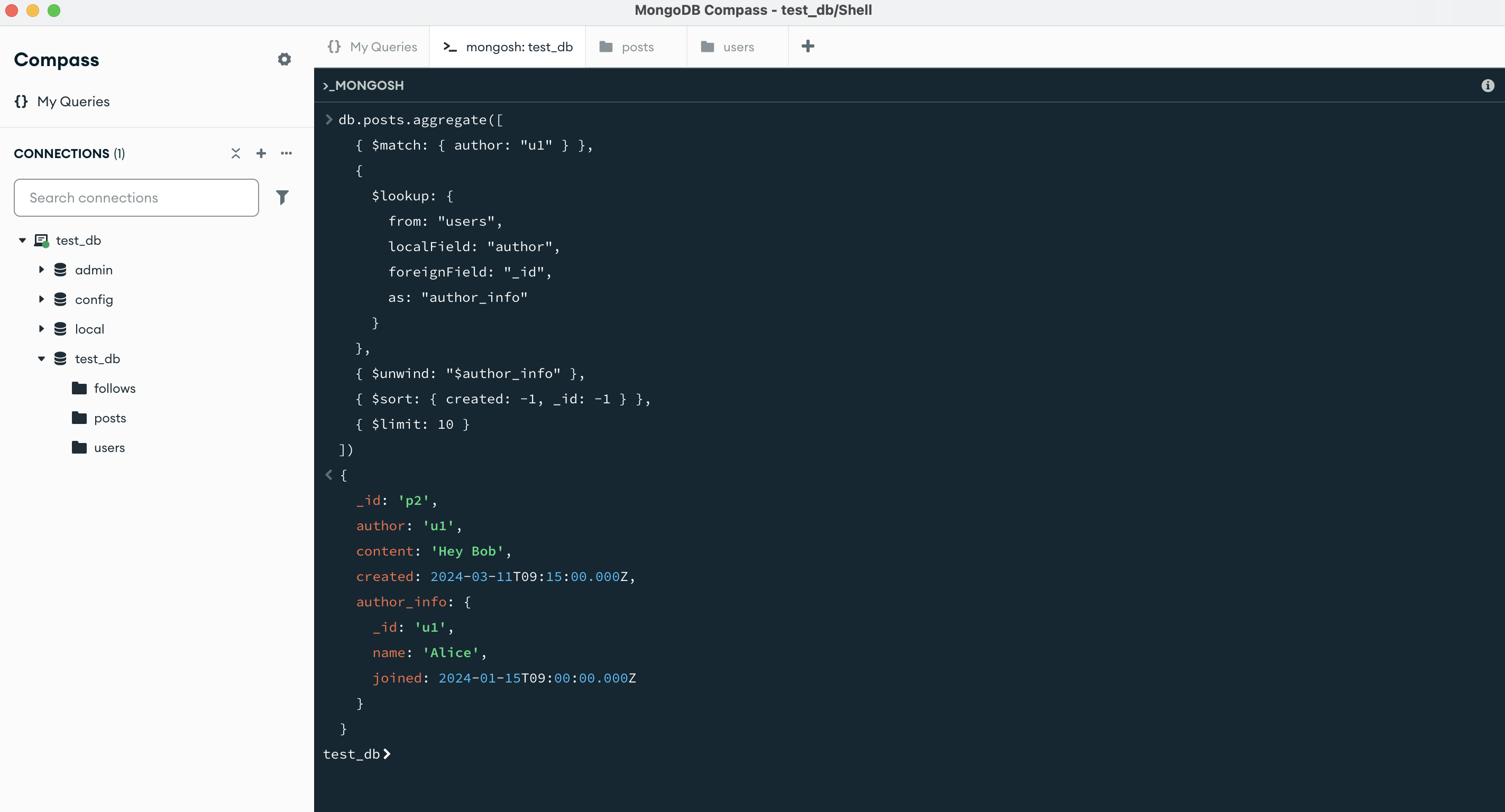
},

{ $unwind: "$author\_info" },

{ $sort: { created: -1, \_id: -1 } },

{ $limit: 10 }

])



1. Write a single aggregation pipeline that, for a given userID, returns:  
   a. The 10 most recent posts from that user’s followings, sorted newest → oldest.

b. Each with the post’s, and the author’s name.

db.posts.aggregate([

// 1. Get the list of users that "u1" follows

{

$lookup: {

from: "follows",

let: { authorId: "$author" },

pipeline: [

{ $match: { $expr: { $and: [

{ $eq: ["$follower", "u1"] },

{ $eq: ["$following", "$$authorId"] }

] } } }

],

as: "isFollowedByU1"

}

},

// 2. Only include posts where isFollowedByU1 array is non-empty

{ $match: { "isFollowedByU1.0": { $exists: true } } },

// 3. Join author details

{

$lookup: {

from: "users",

localField: "author",

foreignField: "\_id",

as: "author\_info"

}

},

{ $unwind: "$author\_info" },

// 4. Sort and limit to 10 most recent

{ $sort: { created: -1, \_id: -1 } },

{ $limit: 10 },

// 5. Project the required fields

{

$project: {

\_id: 1,

content: 1,

created: 1,

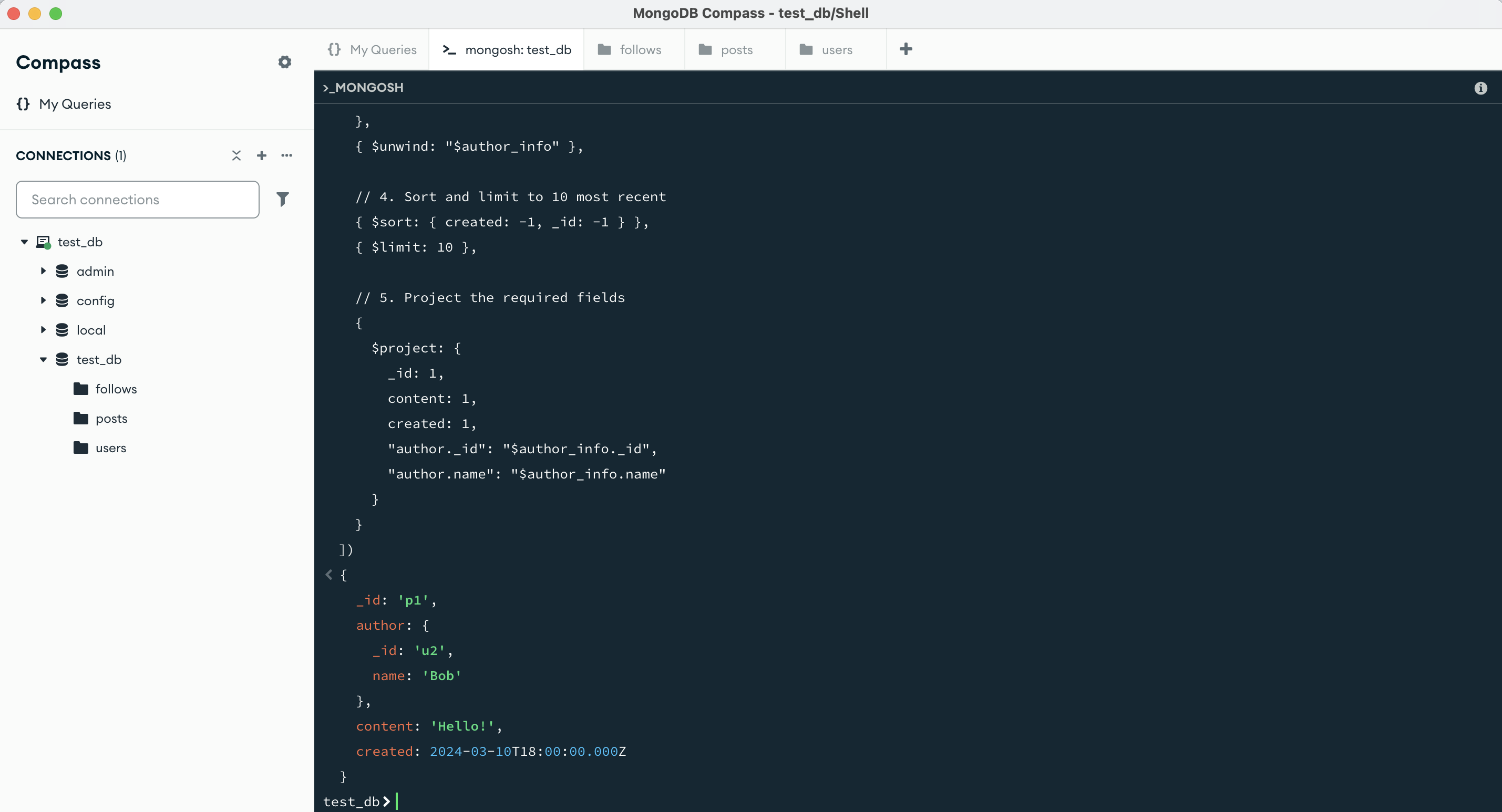
"author.\_id": "$author\_info.\_id",

"author.name": "$author\_info.name"

}

}

])



3. Explain in 2–3 sentences how you’d index these collections for maximum read performance.

1. On the “Posts” collection, create a **compound index** on ‘{ author: 1, created: -1 }’ to efficiently fetch recent posts by a user's followings in reverse-chronological order.
2. On the “follows” collection, index ‘{ follower: 1 }’ to quickly retrieve the list of users someone follows.
3. Also index ‘{ following: 1 }’ if you frequently need to list followers of a specific user.