Experiment 6:

Aggregation Pipeline and its operations

Execute Aggregation Pipeline and its operations (pipeline must contain *match*,group, *sort*,project, $skip etc.)

Let's consider a scenario involving a `restaurantDB` database with a `restaurants` collection. Each document in the `restaurants` collection contains details about a restaurant, including its name, cuisine, location, and an array of reviews. Each review includes a rating and a comment. After creating the `restaurantDB` database and insert sample documents into the `restaurants` collection we will create an aggregation pipeline as shown below.

```
use restaurantDB

db.restaurants.insertMany([
  {
    name: "Biryani House",
    cuisine: "Indian",
    location: "Jayanagar",
    reviews: [
      { user: "Aarav", rating: 5, comment: "Amazing biryani!" },
      { user: "Bhavana", rating: 4, comment: "Great place!" }
    ]
  },
  {
    name: "Burger Joint",
    cuisine: "American",
    location: "Koramangala",
    reviews: [
      { user: "Chirag", rating: 3, comment: "Average burger" },
      { user: "Devika", rating: 4, comment: "Good value" }
    ]
  },
  {
    name: "Pasta House",
    cuisine: "Italian",
    location: "Rajajinagar",
    reviews: [
      { user: "Esha", rating: 5, comment: "Delicious pasta!" },
      { user: "Farhan", rating: 4, comment: "Nice ambiance" }
    ]
  },
  {
    name: "Curry Palace",
    cuisine: "Indian",
    location: "Jayanagar",
    reviews: [
      { user: "Gaurav", rating: 4, comment: "Spicy and tasty!" },
      { user: "Harini", rating: 5, comment: "Best curry in town!" }
    ]
  },
  {
    name: "Taco Stand",
    cuisine: "Mexican",
    location: "Jayanagar",
    reviews: [
      { user: "Ishaan", rating: 5, comment: "Fantastic tacos!" },
      { user: "Jaya", rating: 4, comment: "Very authentic" }
    ]
  }
])
```

Now, let's execute an aggregation pipeline that includes the `$match`, `$unwind`, `$group`, `$sort`, `$project`, and `$skip` stages.

Aggregation Pipeline Explanation

1. **`$match`**: Filter restaurants by cuisine (`"Jayanagar"` location).
2. **`$unwind`**: Deconstruct the `reviews` array from each document to output a document for each review.
3. **`$group`**: Group the documents by restaurant name and calculate the average rating and total number of reviews.
4. **`$sort`**: Sort the results by average rating in descending order.
5. **`$project`**: Restructure the output to include only the restaurant name, average rating, and total reviews.
6. **`$skip`**: Skip the first document.

```
db.restaurants.aggregate([
  {
    $match: {
      location: "Jayanagar"
    }
  },
  {
    $unwind: "$reviews"
  },
  {
    $group: {
      _id: "$name",
      averageRating: { $avg: "$reviews.rating" },
      totalReviews: { $sum: 1 }
    }
  },
  {
    $sort: {
      averageRating: -1
    }
  },
  {
    $project: {
      _id: 0,
      restaurant: "$_id",
      averageRating: 1,
      totalReviews: 1
    }
  },
  {
    $skip: 1
  } ]).pretty()
```