Experiment:7

MongoDB Listing & Review Queries

Listing in MongoDB: A Comprehensive Overview

Listing in MongoDB generally refers to the process of retrieving a set of items from a database or a collection. While there's no specific command named "list" in MongoDB, we can achieve similar functionalities using different methods:

1. Listing Databases

- **Purpose:** To view all databases present in the MongoDB instance.
- **Command:** show dbs
- **Output:** A list of database names with basic statistics.

2. Listing Collections

- **Purpose:** To view all collections within a specific database.
- **Command:** show collections (within the desired database)
- **Output:** A list of collection names.

```
db.listingsAndReviews.find({
  "host.host_picture_url": { $exists: true, $ne: null }
}, {
  "listing_url": 1,
  "name": 1,
  "address": 1,
  "host.host_picture_url": 1
})
```

- db.listingsAndReviews.find({}): Targets the listingsAndReviews collection for querying.
- "$exists: true, $ne: null": Ensures the host.host_picture_url field exists and is not null, filtering listings with a valid picture URL.
- "$project: { ... }": Specifies the fields to include in the output:
- "listing_url": Listing URL
- "name": Listing name
- "address": Listing address
- "host.host_picture_url": Host picture URL (nested within the host object

This MongoDB aggregation pipeline:

1. Matches documents where the `host_picture_url` exists and is not null within the `host` subdocument.

2. Projects the required fields (`listing_url`, `name`, `address`, and `host_picture_url`).

3.Assuming that your Listings And Reviews collection is named `listingsAndReviews` and the host's picture URL is stored within a `host` subdocument, this query should retrieve the desired results. Adjust the field names and collection name if they differ in your actual schema.

```
db.reviews.issssssnsert([{
"listing_url": "https://www.example.com/listing/12345",
"name": "Cozy Apartment in Downtown",
"address": "New York",
"host_picture_url": "https://www.example.com/pictures/johndoe.jpg"
},
{
"listing_url": "https://www.example.com/listing/12345",
"name": "Cozy Apartment in Downtown",
"address": "New York"
}])
```

Using E-commerce collection write a query to display reviews summary.

```
db.ecommerce.insert([
{
"product_id": "123",
"rating": 5,
"review": "Great product!"
},
{
```

```
"product_id": "123",
"rating": 4,
"review": "Good quality"
},
{
"product_id": "456",
"rating": 3,
"review": "Average product"
},
{

"product_id": "456",
"rating": 2,
"review": "Poor quality"
},
{
"product_id": "789",
"rating": 5,
"review": "Excellent product!"
}
])

>db.ecommerce.aggregate([
// Unwind the reviews array to de-normalize the data
{ $unwind: "$reviews" },
// Group by product and calculate summary statistics
{
$group: {
_id: "$product_id",
average_rating: { $avg: "$reviews.rating" },
total_reviews: { $sum: 1 },
```

```
// You can add more summary statistics as needed
}
},
// Optionally, you can project the output to shape it as needed
{
$project: {
_id: 0, // Exclude the group _id
product_id: "$_id",
average_rating: 1,
total_reviews: 1,
}
}
])
```

The final output will be a collection of documents, each representing a product with the following fields:

- product_id: The unique identifier of the product.
- average_rating: The average rating of the product based on all reviews.
- total_reviews: The total number of reviews for the product.

This aggregation pipeline effectively calculates and summarizes product information based on their reviews.