Student Name: Shanikwa Haynes

Student ID:

Date: 03/31/2025

D597 MKN1 Task 2: Non-Relational Database Design and Implementation

A.

1.

The healthcare organization selected in Scenario 1 is experiencing difficulty integrating patient medical records with fitness tracker data. This lack of integration prevents medical professionals from accessing real-time insights into patient activity levels, making it challenging to evaluate how lifestyle choices impact individual health outcomes. A database solution is needed to link patient information with tracker usage data in a structured and scalable way. Implementing a unified database would allow the organization to analyze patient wellness trends and personalize care recommendations based on fitness and health indicators.

2.

A NoSQL database solution is appropriate for this business problem because it provides flexible and scalable data handling for the semi-structured nature of medical and fitness data. Unlike relational databases, NoSQL solutions can efficiently store data in varied formats, such as JSON, which aligns with how patient and device data are structured. Furthermore, a NoSQL solution supports horizontal scaling, making it suitable for growing volumes of health records and device metrics over time. The ability to dynamically expand data models without schema redesign allows the healthcare organization to adapt as new fitness tracker types or health measurements are introduced.

3.

The most appropriate NoSQL database type for this problem is a document-based database such as MongoDB. This type of database stores data as JSON-like documents, which matches the structure of the provided patient and fitness tracker files. Each document can represent a unique patient or device model, allowing for flexible and independent storage of individual data records. The document model also enables efficient queries and updates, especially when linking patients to their fitness devices.

4.

The business data in this scenario will be used to connect patient records with their respective fitness tracker models to support health monitoring and decision-making. In the proposed database design, the patients collection will store individual medical data, including conditions, medications, and tracker usage. The fitness_trackers collection will contain specifications and performance data for each model. This data can be queried to identify patterns, track patient fitness engagement, and recommend optimal devices based on patient history.

B.

The proposed NoSQL database design addresses scalability by using techniques that align with the business scenario and anticipated data growth. First, horizontal scaling is achieved through MongoDB's support for sharding, allowing the system to distribute data across multiple servers. Second, indexing is applied to frequently queried fields such as patient_id, tracker, and rating, which improves read performance and response time. Third, the flexible schema design ensures that new fields—such as fitness metrics or additional device attributes—can be added without restructuring the entire database.

C.

To ensure data protection in the proposed database design, several privacy and security measures will be implemented. Role-Based Access Control (RBAC) will restrict access based on user roles, ensuring that only authorized personnel can view or modify sensitive health data. Data encryption will be used both in transit and at rest to protect against unauthorized access or breaches. Audit logging will be enabled to track all database access and changes, supporting accountability and traceability. Additionally, patient identifiers can be anonymized when data is used for research or reporting, ensuring compliance with healthcare data regulations such as HIPAA.

D.

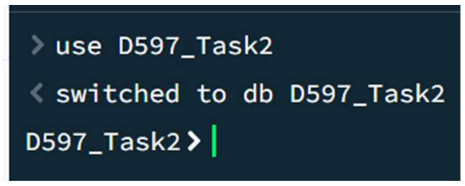
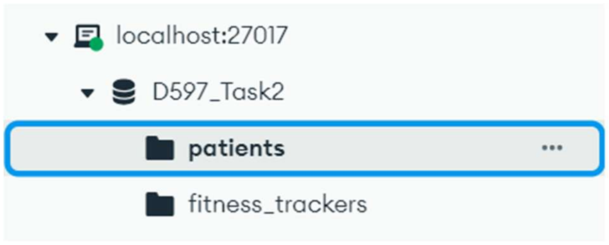
1.

The database instance was created successfully in the WGU Virtual Lab environment using MongoDB. A new database named D597_Task2 was initialized using the MongoDB shell command use D597_Task2. This command both selects and creates the database if it does not already exist. Within this database, two collections were created: patients for storing patient medical records and fitness_trackers for storing device specifications.

In the WGU Virtual Lab environment, create a MongoDB instance and name it `D597_Task2` using the following script:

```
```

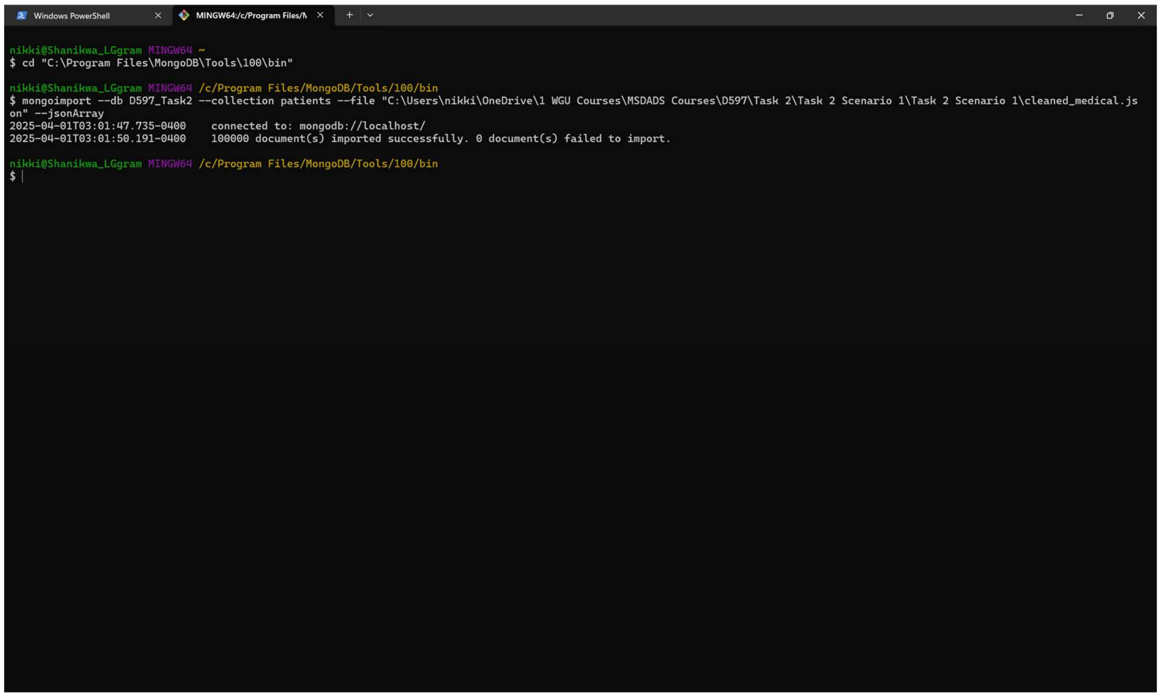
use D597_Task2;

```



```
> use D597_Task2
< switched to db D597_Task2
D597_Task2 >
```

The database instance:



localhost:27017
D597_Task2
patients
fitness_trackers

**2.**

The dataset for the selected scenario was successfully inserted into the MongoDB collections using the insertMany() command. The cleaned JSON files containing patient medical data and fitness tracker details were imported into their respective collections: patients and fitness_trackers. Each record was reviewed to ensure proper data formatting, and fields such as prices and ratings were converted to numerical values where necessary.

Load the patient medical & fitness_tracker datasets into MongoDB:

```

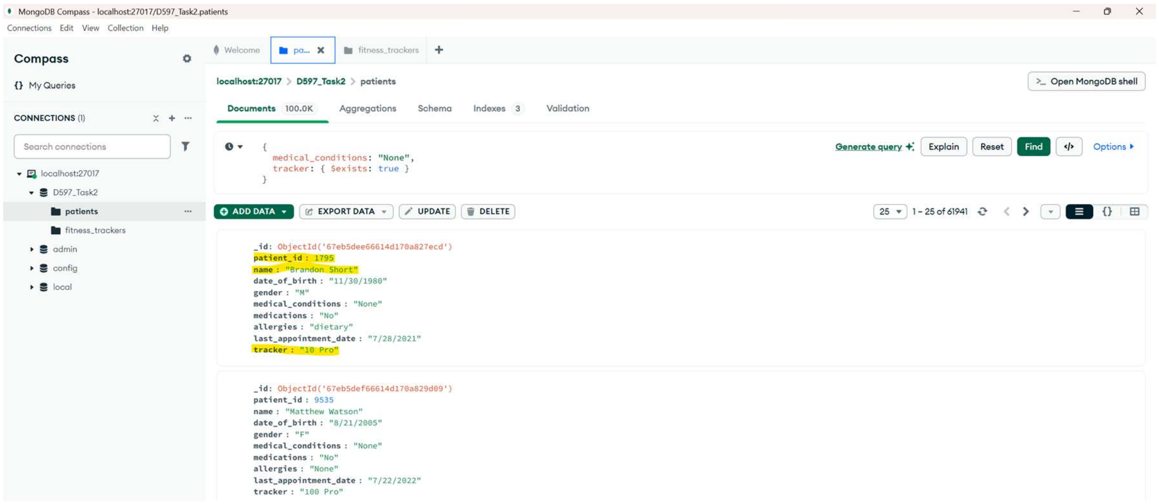
# Import cleaned medical data

mongoimport --db D597_Task2 --collection patients --file "C:\Users\nikki\OneDrive\1 WGU Courses\MSDADS Courses\D597\Task 2\Task 2 Scenario 1\Task 2 Scenario 1\cleaned_medical.json" --jsonArray

Screenshot showing successful import of cleaned_medical.json using mongoimport.



```
nikki@Shanikwa_LGgram MINGW64 ~
$ cd "C:\Program Files\MongoDB\Tools\100\bin"

nikki@Shanikwa_LGgram MINGW64 /c/Program Files/MongoDB/Tools/100/bin
$ mongoimport --db D597_Task2 --collection patients --file "C:\Users\nikki\OneDrive\1 WGU Courses\MSDADS Courses\D597\Task 2\Task 2 Scenario 1\Task 2 Scenario 1\cleaned_medical.js
on" --jsonArray
2025-04-01T03:01:47.735-0400    connected to: mongodb://localhost/
2025-04-01T03:01:50.191-0400    100000 document(s) imported successfully. 0 document(s) failed to import.

nikki@Shanikwa_LGgram MINGW64 /c/Program Files/MongoDB/Tools/100/bin
$ |
```



```
{
  medical_conditions: "None",
  tracker: { $exists: true }
}
```

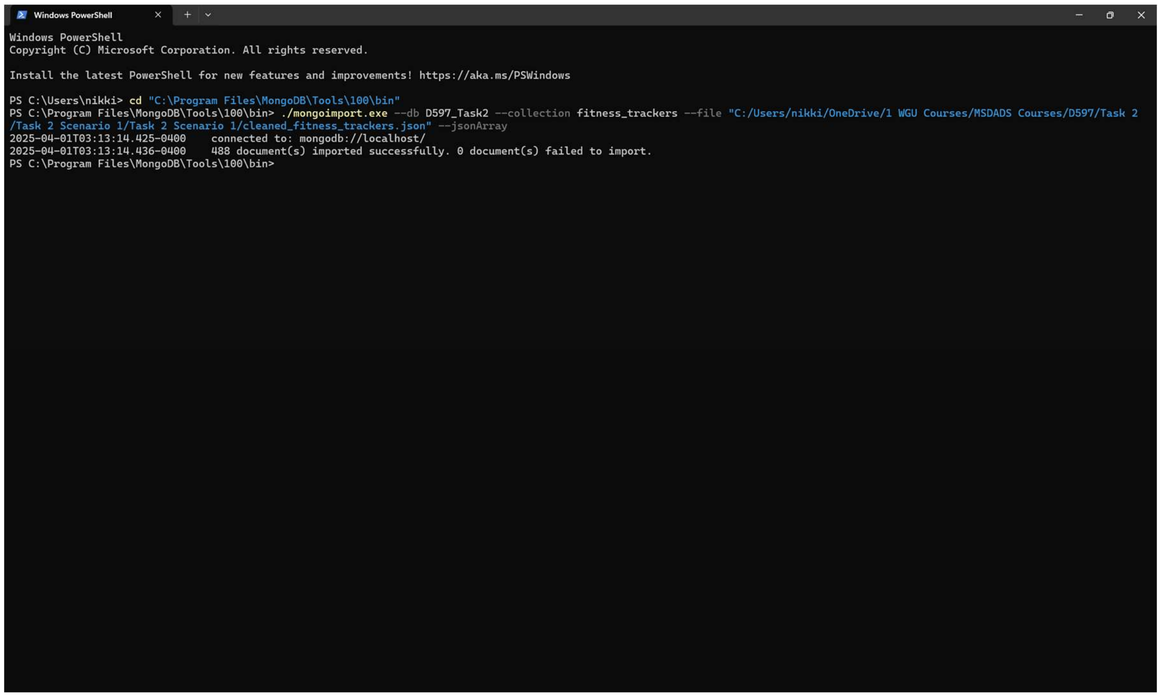
```
_id: ObjectId('67eb5dee66614d170a827ecd')
patient_id : 1795
name : "Brandon Short"
date_of_birth : "11/30/1980"
gender : "M"
medical_conditions : "None"
medications : "No"
allergies : "dietary"
last_appointment_date : "7/28/2021"
tracker : "10 Pro"
```

```
_id: ObjectId('67eb5def66614d170a829d09')
patient_id : 9535
name : "Matthew Watson"
date_of_birth : "8/21/2005"
gender : "F"
medical_conditions : "None"
medications : "No"
allergies : "None"
last_appointment_date : "7/22/2022"
tracker : "100 Pro"
```

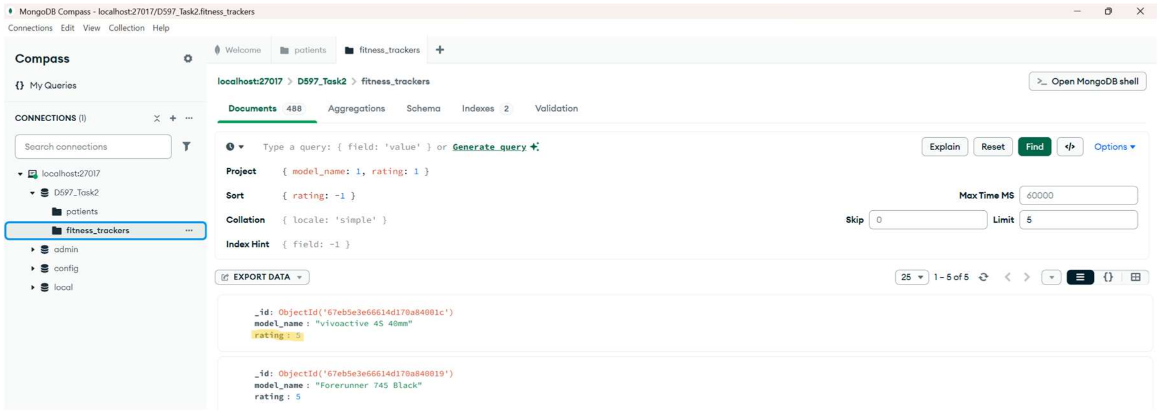
MongoDB Compass view confirming that patient records were successfully inserted into the patients collection.

# Import cleaned fitness tracker data

mongoimport --db D597_Task2 --collection fitness_trackers --file "C:\Users\nikki\OneDrive\1 WGU Courses\MSDADS Courses\D597\Task 2\Task 2 Scenario 1\Task 2 Scenario 1\cleaned_fitness_trackers.json" --jsonArray

```

Screenshot showing successful import of cleaned_fitness_trackers.json using mongoimport.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

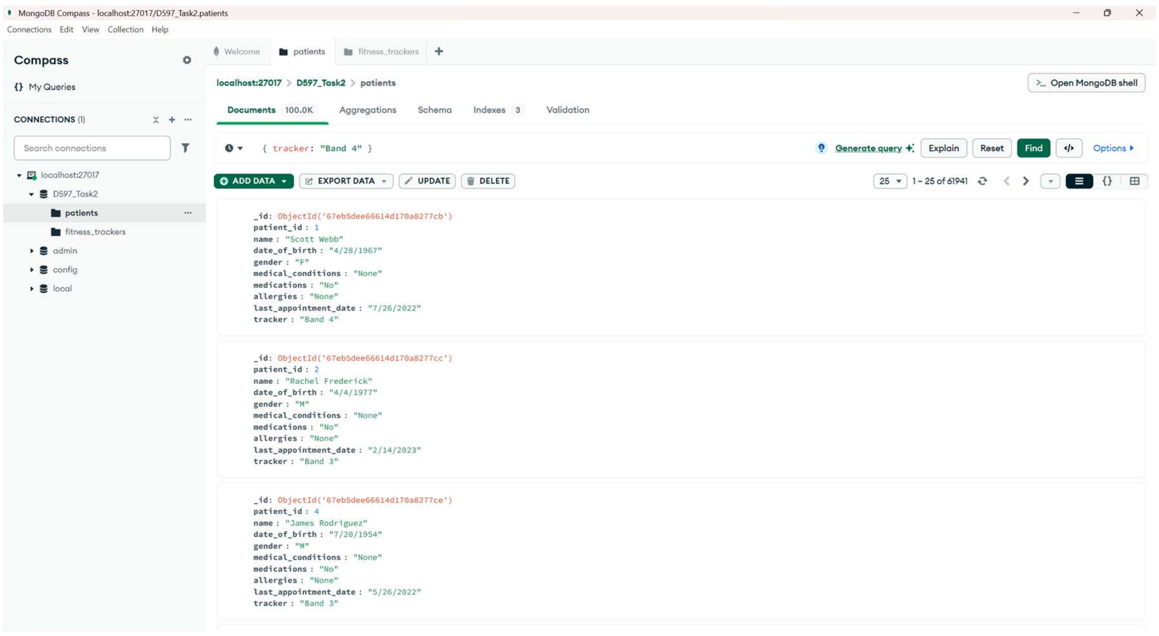
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\nikki> cd "C:\Program Files\MongoDB\Tools\100\bin"
PS C:\Program Files\MongoDB\Tools\100\bin> ./mongoimport.exe --db D597_Task2 --collection fitness_trackers --file "C:/Users/nikki/OneDrive/1 WGU Courses/MSDADS Courses/D597/Task 2
/Task 2 Scenario 1/Task 2 Scenario 1/cleaned_fitness_trackers.json" --jsonArray
2025-04-01T03:13:14.425-0400   connected to: mongodb://localhost/
2025-04-01T03:13:14.436-0400   488 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Program Files\MongoDB\Tools\100\bin>
```


MongoDB Compass - localhost:27017/D597_Task2.fitness_trackers
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (1)
Search connections
localhost:27017
D597_Task2
patients
fitness_trackers
admin
config
local
Welcome
patients
fitness_trackers
localhost:27017 > D597_Task2 > fitness_trackers
Open MongoDB shell
Documents 488
Aggregations
Schema
Indexes 2
Validation
Explain
Reset
Find
Options
EXPORT DATA
25
1 - 5 of 5

```
Type a query: { field: 'value' } or Generate query
Project     { model_name: 1, rating: 1 }
Sort        { rating: -1 }                         Max Time MS  60000
Collation   { locale: 'simple' }        Skip  0    Limit  5
Index Hint  { field: -1 }

_id: ObjectId('67eb5e3e66614d170a84001c')
model_name : "vivoactive 4S 40mm"
rating : 5

_id: ObjectId('67eb5e3e66614d170a840019')
model_name : "Forerunner 745 Black"
rating : 5
```

MongoDB Compass view confirming that fitness tracker records were successfully inserted into the fitness_trackers collection.

3.

Three functional queries were written and executed in MongoDB Compass to retrieve data that supports the identified business problem.

The first query retrieved all patients using a specific tracker model, allowing for personalized device-based analysis.
```

```javascript
db.patients.find({ "tracker": "Band 4" });
```

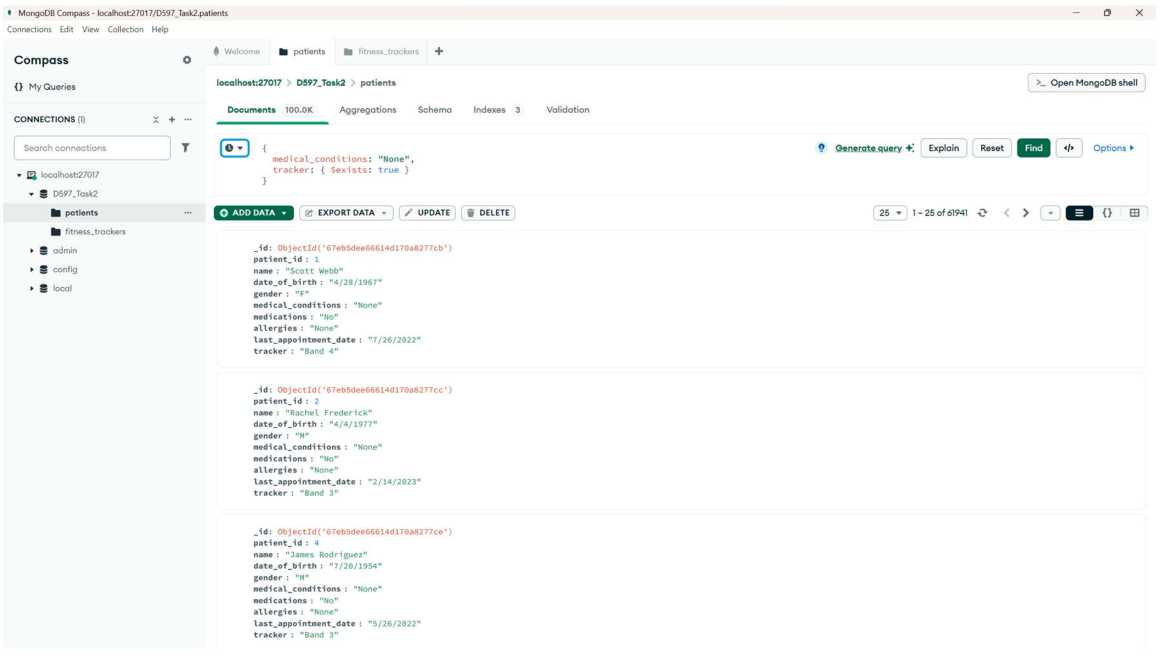


```
{ tracker: "Band 4" }
```

```
_id: ObjectId('67eb5dee66614d170a8277cb')
patient_id : 1
name : "Scott Webb"
date_of_birth : "4/28/1967"
gender : "F"
medical_conditions : "None"
medications : "No"
allergies : "None"
last_appointment_date : "7/26/2022"
tracker : "Band 4"

_id: ObjectId('67eb5dee66614d170a8277cc')
patient_id : 2
name : "Rachel Frederick"
date_of_birth : "4/4/1977"
gender : "M"
medical_conditions : "None"
medications : "No"
allergies : "None"
last_appointment_date : "2/14/2023"
tracker : "Band 3"

_id: ObjectId('67eb5dee66614d170a8277ce')
patient_id : 4
name : "James Rodriguez"
date_of_birth : "7/20/1954"
gender : "M"
medical_conditions : "None"
medications : "No"
allergies : "None"
last_appointment_date : "5/26/2022"
tracker : "Band 3"
```

The second query identified patients who had no medical conditions and were using trackers, supporting preventive care targeting.

```javascript
db.patients.find({ "medical_conditions": "None", "tracker": { "$exists": true } });
```

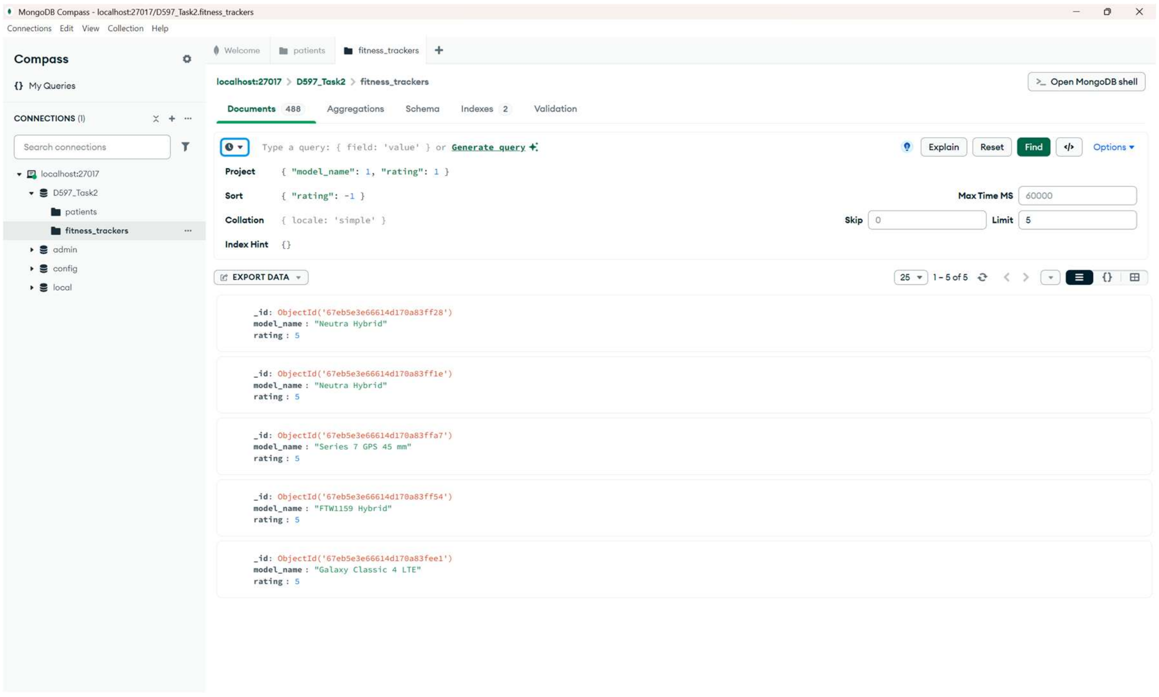


```
{
  medical_conditions: "None",
  tracker: { $exists: true }
}
```

```
_id: ObjectId('67eb5dee66614d170a8277cb')
patient_id : 1
name : "Scott Webb"
date_of_birth : "4/28/1967"
gender : "F"
medical_conditions : "None"
medications : "No"
allergies : "None"
last_appointment_date : "7/26/2022"
tracker : "Band 4"

_id: ObjectId('67eb5dee66614d170a8277cc')
patient_id : 2
name : "Rachel Frederick"
date_of_birth : "4/4/1977"
gender : "M"
medical_conditions : "None"
medications : "No"
allergies : "None"
last_appointment_date : "2/14/2023"
tracker : "Band 3"

_id: ObjectId('67eb5dee66614d170a8277ce')
patient_id : 4
name : "James Rodriguez"
date_of_birth : "7/20/1954"
gender : "M"
medical_conditions : "None"
medications : "No"
allergies : "None"
last_appointment_date : "5/26/2022"
tracker : "Band 3"
```

The third query retrieved the top five fitness trackers sorted by customer rating, aiding in device recommendation strategies.

```javascript
db.fitness_trackers.find().sort({ "rating": -1 }).limit(5);
```



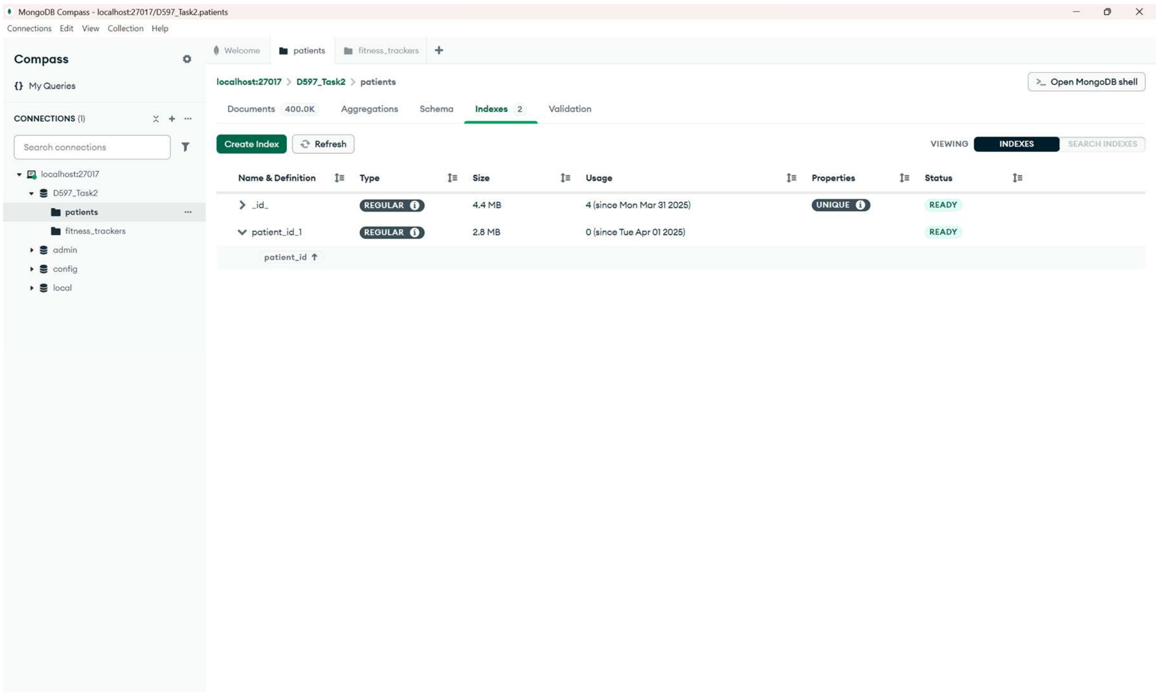
```
Project     { "model_name": 1, "rating": 1 }
Sort        { "rating": -1 }                          Max Time MS  60000
Collation   { locale: 'simple' }           Skip  0    Limit  5
Index Hint  {}
```

```
_id: ObjectId('67eb5e3e66614d170a83ff28')
model_name : "Neutra Hybrid"
rating : 5

_id: ObjectId('67eb5e3e66614d170a83ff1e')
model_name : "Neutra Hybrid"
rating : 5

_id: ObjectId('67eb5e3e66614d170a83ffa7')
model_name : "Series 7 GPS 45 mm"
rating : 5

_id: ObjectId('67eb5e3e66614d170a83ff54')
model_name : "FTW1159 Hybrid"
rating : 5

_id: ObjectId('67eb5e3e66614d170a83fee1')
model_name : "Galaxy Classic 4 LTE"
rating : 5
```

4.

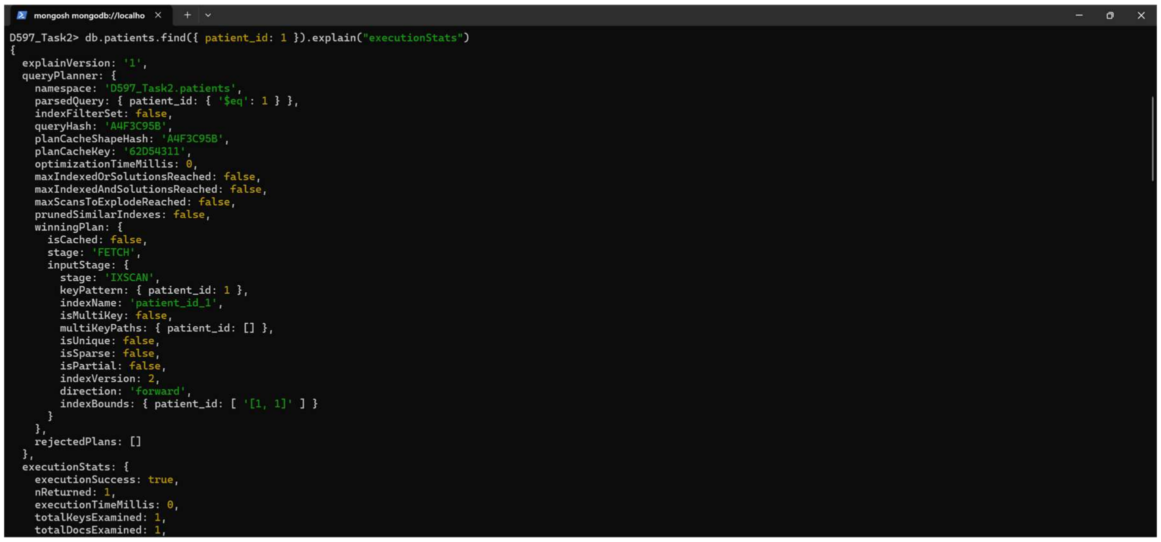
To improve the performance of the queries written in Part D3, indexing was applied to key fields in the MongoDB collections. Specifically, indexes were created on the patient_id, tracker, and rating fields using the createIndex() command. These indexes allowed the queries to execute faster by reducing the number of documents scanned. The effectiveness of the optimization was demonstrated using MongoDB Compass's Explain Plan feature, which showed the use of indexed scans instead of full collection scans.

- **Indexing:** Improve query performance by indexing frequently searched fields:



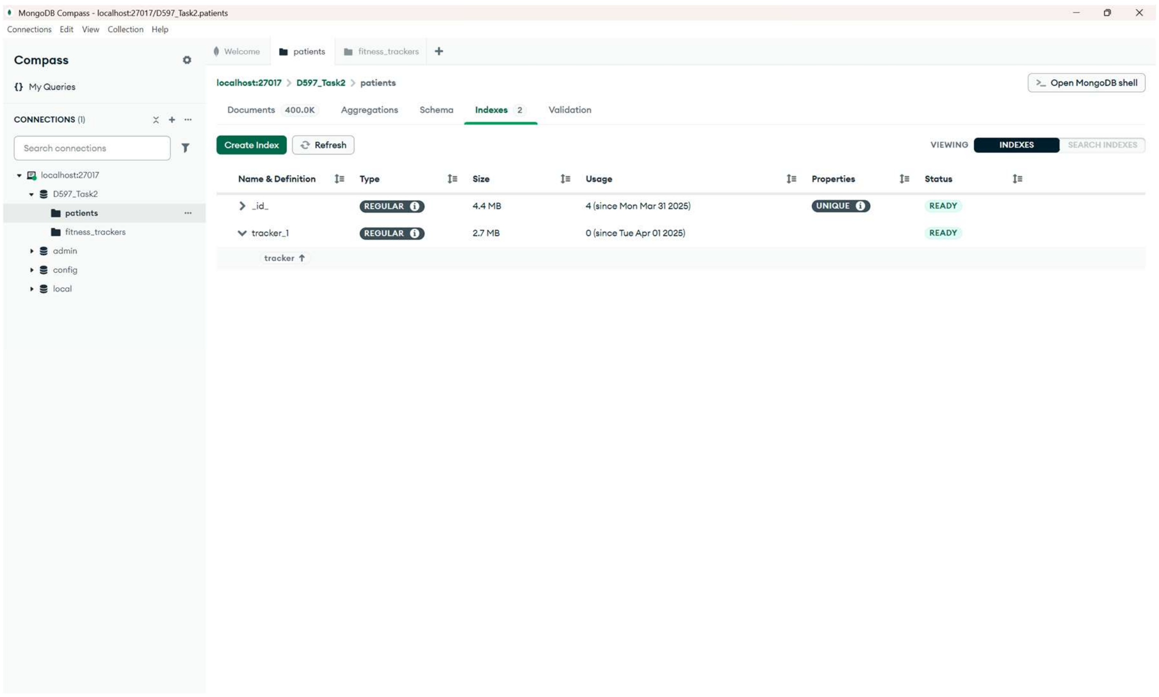
Name & Definition	Type	Size	Usage	Properties	Status
_id_	REGULAR	4.4 MB	4 (since Mon Mar 31 2025)	UNIQUE	READY
patient_id_1	REGULAR	2.8 MB	0 (since Tue Apr 01 2025)		READY
patient_id ↑					

```javascript

// Index to speed up queries by patient ID

db.patients.createIndex({ "patient_id": 1 });


db.patients.find({ "patient_id": 1 }).explain("executionStats");
```



```
D597_Task2> db.patients.find({ patient_id: 1 }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'D597_Task2.patients',
    parsedQuery: { patient_id: { '$eq': 1 } },
    indexFilterSet: false,
    queryHash: 'A4F3C95B',
    planCacheShapeHash: 'A4F3C95B',
    planCacheKey: '62D54311',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { patient_id: 1 },
        indexName: 'patient_id_1',
        isMultiKey: false,
        multiKeyPaths: { patient_id: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { patient_id: [ '[1, 1]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 0,
    totalKeysExamined: 1,
    totalDocsExamined: 1,
```

To improve performance for retrieving individual patient records, an index was created on the patient_id field in the patients collection. This index was verified using the explain("executionStats") function. The query planner output showed "stage": "IXSCAN" and "indexName": "patient_id_1", confirming that the index was used to locate matching documents efficiently. Furthermore, "keysExamined": 1 and "docsExamined": 1" indicated that the database engine only searched one key and returned one result, proving the index significantly optimized the query.

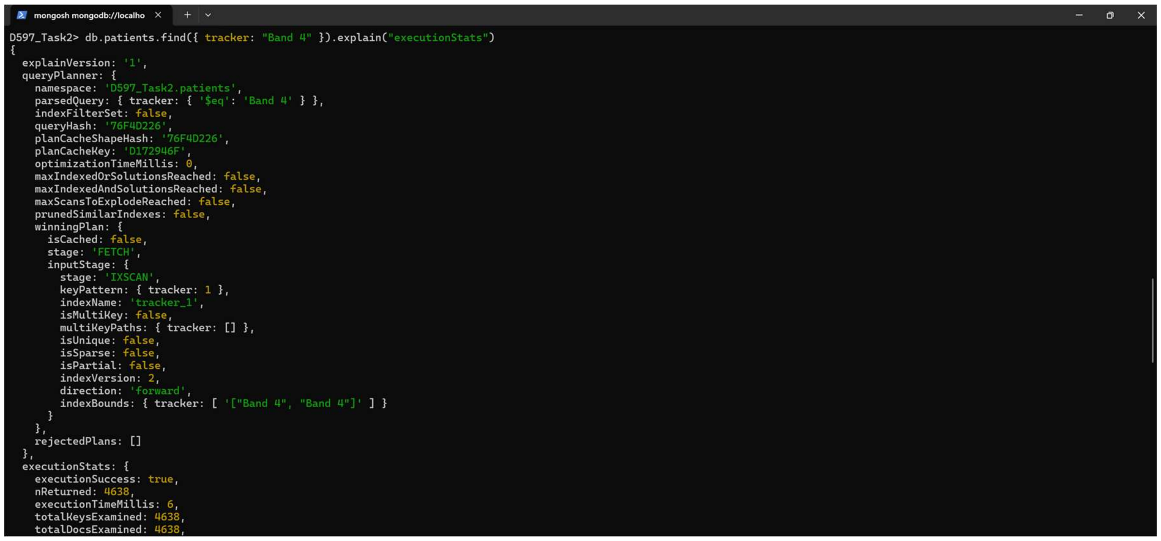


Name & Definition	Type	Size	Usage	Properties	Status
_id_	REGULAR	4.4 MB	4 (since Mon Mar 31 2025)	UNIQUE	READY
tracker_1	REGULAR	2.7 MB	0 (since Tue Apr 01 2025)		READY
tracker ↑					

// Index to improve performance when querying by tracker model

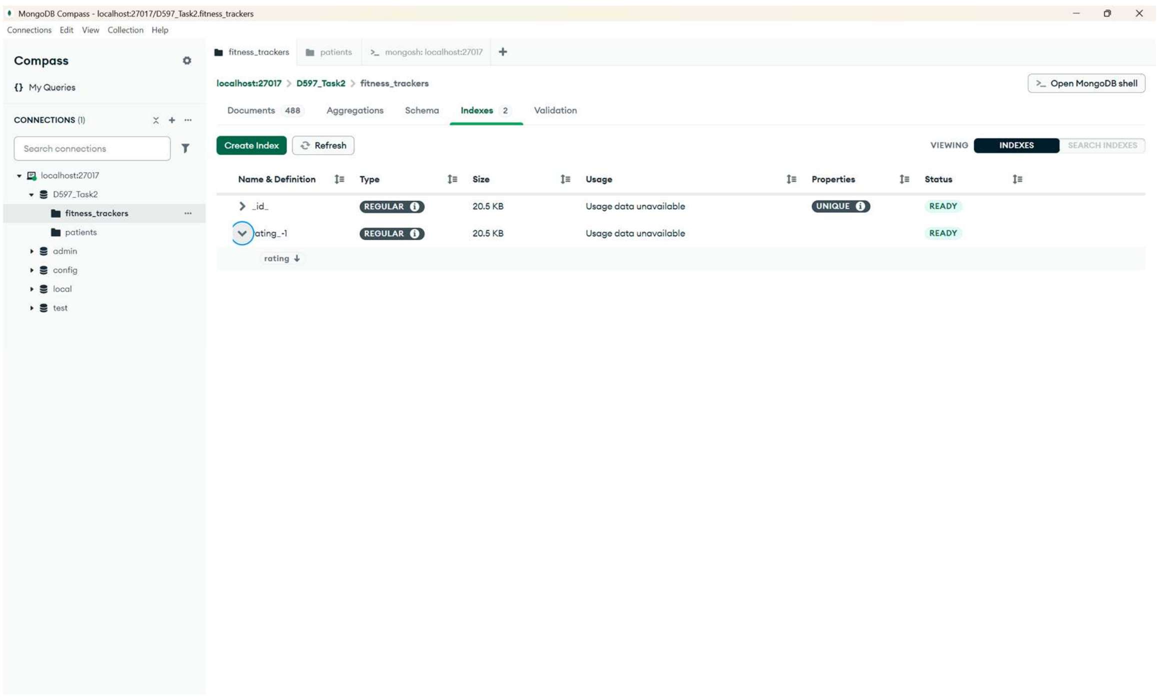
```
db.patients.createIndex({ "tracker": 1 });
```

```
db.patients.find({ "tracker": "Band 4" }).explain("executionStats");
```



```
D597_Task2> db.patients.find({ tracker: "Band 4" }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'D597_Task2.patients',
    parsedQuery: { tracker: { '$eq': 'Band 4' } },
    indexFilterSet: false,
    queryHash: '76F4D226',
    planCacheShapeHash: '76F4D226',
    planCacheKey: 'D172946F',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { tracker: 1 },
        indexName: 'tracker_1',
        isMultiKey: false,
        multiKeyPaths: { tracker: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { tracker: [ '["Band 4", "Band 4"]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4638,
    executionTimeMillis: 6,
    totalKeysExamined: 4638,
    totalDocsExamined: 4638,
```

To enable efficient filtering of patients based on the fitness tracker used, an index was created on the tracker field in the patients collection. When tested with a query for patients using "Band 4", the explain("executionStats") output showed "stage": "IXSCAN" and "indexName": "tracker_1".



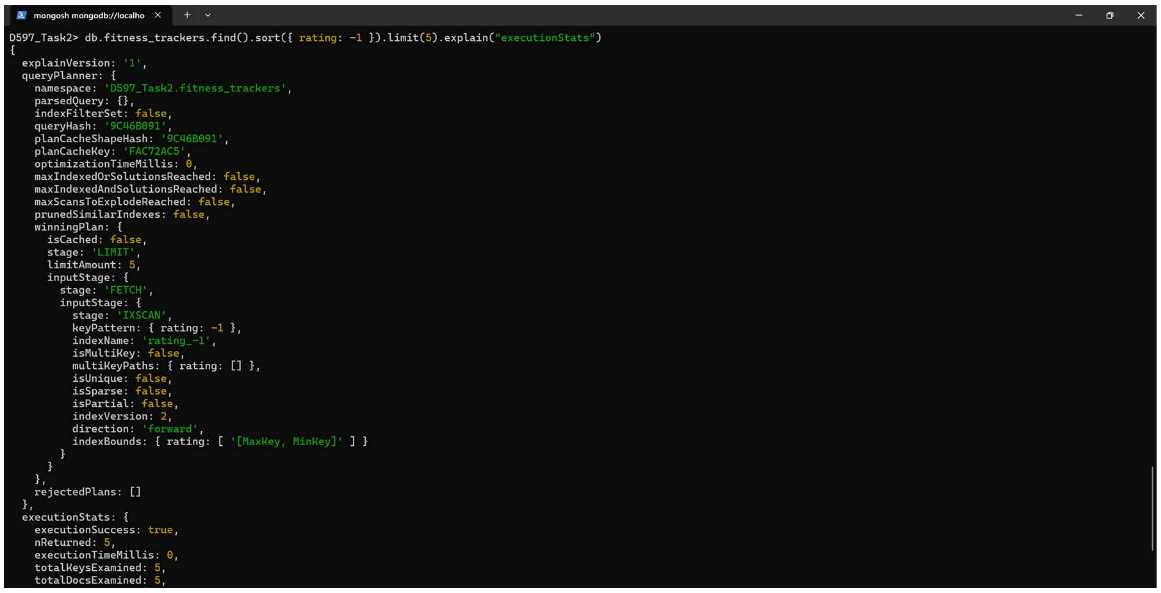
Name & Definition	Type	Size	Usage	Properties	Status
_id_	REGULAR	20.5 KB	Usage data unavailable	UNIQUE	READY
rating_-1	REGULAR	20.5 KB	Usage data unavailable		READY
rating ↓					

// Index to optimize sorting by rating in the fitness tracker collection

db.fitness_trackers.createIndex({ "rating": -1 });

```
```

db.fitness_trackers.find().sort({ rating: -1 }).limit(5).explain("executionStats");



```
D597_Task2> db.fitness_trackers.find().sort({ rating: -1 }).limit(5).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'D597_Task2.fitness_trackers',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: '9C46B091',
    planCacheShapeHash: '9C46B091',
    planCacheKey: 'FAC72AC5',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'LIMIT',
      limitAmount: 5,
      inputStage: {
        stage: 'FETCH',
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: { rating: -1 },
          indexName: 'rating_-1',
          isMultiKey: false,
          multiKeyPaths: { rating: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: { rating: [ '[MaxKey, MinKey]' ] }
        }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 5,
    executionTimeMillis: 0,
    totalKeysExamined: 5,
    totalDocsExamined: 5,
```

To support sorting by fitness tracker ratings in descending order, an index was created on the rating field using the -1 direction. When executing a query to retrieve the top five rated trackers, the explain("executionStats") output confirmed the use of this index with "stage": "IXSCAN" and "indexName": "rating_-1". Although the scan direction was "forward", the descending index still applied correctly due to the data's structure in the index. Only five keys and documents were examined, showing the query was optimized for performance.

E. Panopto Recording Link: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=a80a586a-753a-48ab-bc5b-b2b7013f9c77

F.

The content and design of this submission were based entirely on the WGU course materials provided and publicly available MongoDB documentation.