Nicole Gallo
D597 – Database Management
June 10, 2025
MKN1 Task 2: Non-Relational Database Design and Implementation

Table of Contents

Part 1: Design Document 2

A1: Business Problem 2

A2: Proposal for a NoSQL Database 2

A3: Justification for NoSQL Database 2

A4: Business Data Usage 2

B: Logical Data Model for Storage 3

C: Database Collections and File Attributes 4

D: Scalability 4

E: Privacy and Security Measures 4

Part 2: Implementation 5

F. Implement the proposed database design in MongoDB 5

F1 - Write script to create a database named “D597 Task 2” and its respective collections (patients and fitness trackers) 5

F2 - Write script to insert the data records from JSON file into the database (fitness_trackers.json and medical.json) 5

F3 - Write script for **three** queries to solve the identified business problem 9

F4 – Optimization Techniques to improve runtime with Indexes 13

Part 3: Presentation 27

H: Citations 27

Part 1: Design Document

A1: Business Problem

A healthcare clinic provides wearable fitness trackers to patients for remote monitoring of their activity levels and general well-being. The clinic has acquired multiple fitness tracker brands and models but lacks a centralized system to evaluate their effectiveness.

Clinicians want to assess device performance based on patient profiles (e.g., age, condition, medication use) and make informed decisions about which tracker to recommend. They need to analyze battery life, rating, and display types of the trackers used by different patient groups to optimize healthcare outcomes and resource allocation.

A2: Proposal for a NoSQL Database

A non-relational JSON-based structure is ideal for this problem due to the flexible and nested nature of fitness tracker specifications and diverse patient metadata. The data structure will include two main collections:

1. `fit_trackers`: Each document contains detailed specifications of wearable devices, such as brand, display type, rating, and battery life.
2. `patients`: Each document stores patient data including demographics, medical background, and the tracker they use.

These collections can be joined via the **_id** attribute (and possibly incorporating the **tracker name**), allowing the separate datasets to reference one another for further analysis.

A3: Justification for NoSQL Database

A database solution enables clinicians to:

- Quickly retrieve tracker performance stats by patient condition or age group
- Analyze device usage trends over time
- Automate recommendations using query logic

A NoSQL document store is especially justified here due to its support for semi-structured data, high scalability, and ease of importing JSON files directly.

A4: Business Data Usage

The fitness tracker data will be used to extract insights about each device's cost, battery life, and rating. The patient dataset will provide demographic and medical information, allowing for:

- Filtering trackers by user satisfaction (rating)

- Grouping devices by effectiveness in different medical contexts
- Identifying cost-effective devices for bulk purchasing

B: Logical Data Model for Storage

Collection 1: fit_trackers

```
{
  "_id": {
    "$oid": "68486cbdf4687d58595c5cdd"
  },
  "Brand_Name": "Xiaomi",
  "Device_Type": "FitnessBand",
  "Model_Name": "Smart Band 4",
  "Color": "Black",
  "Selling_Price": "2,099",
  "Original_Price": "2,499",
  "Display": "AMOLED Display",
  "Rating": 4.2,
  "Strap_Material": "Thermoplastic
polyurethane",
  " Battery_Life_Days": 14,
  "Reviews": ""
}
```

Collection 2: patients

```
{
  "_id": {
    "$oid": "68486cdaab2670c2b818af7f"
  },
  "patient_id": 9,
  "name": "Kenneth Johnson",
  "date_of_birth": "4/15/2011",
  "gender": "F",
  "medical_conditions": "Mild",
  "medications": "Yes",
  "allergies": "None",
  "last_appointment_date": "3/12/2023",
  "Tracker_Model": "Band 5i"
}
```

C: Database Collections and File Attributes

Collections:

- `fit_trackers`
- `patients`

Key Attributes:

- Fit Tracker attributes: `model_name, brand_name, device_type, display, battery_life_days, rating, selling_price, color, original_price, strap_material, reviews`
- Patient attributes: `patient_id, name, date_of_birth, gender, medical_conditions, tracker_model, medications, allergies, last_appointment_date`

D: Scalability

The use of MongoDB ensures horizontal scalability, high write throughput, and flexible schema evolution as new devices, more attributes, and future patients are added. Indexing on `tracker_model` and `patient_id` supports efficient queries.

Strategies for scalability are as follows:

- Sharding by `tracker_model` to distribute workload
- Caching frequent queries such as finding the avg battery life per medical condition
- Aggregation pipelines for real-time analytics

E: Privacy and Security Measures

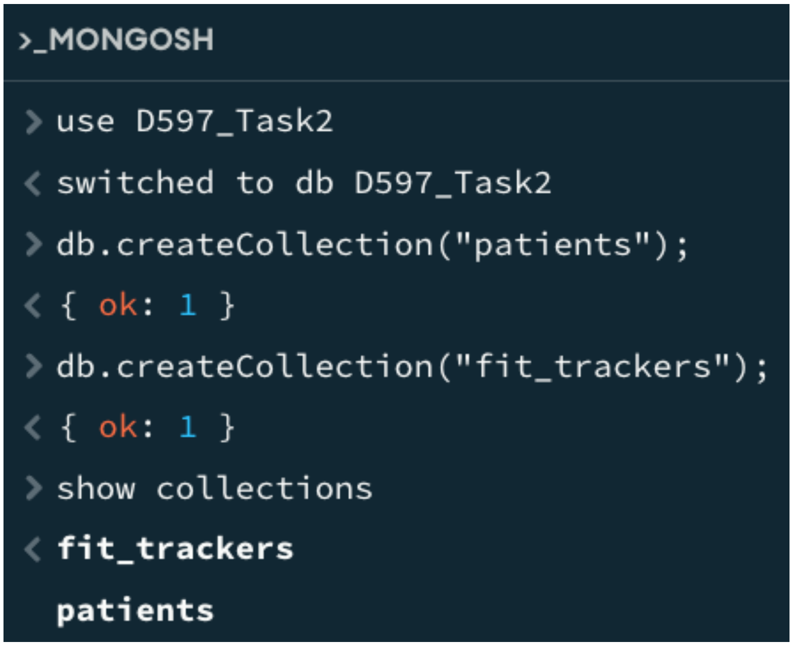
To protect patient and device data:

- Encryption at rest and in transit
- Role-based access control (RBAC): clinicians can query data; admins can update
- Audit logs for all data access
- Tokenization or anonymization of personally identifiable information (PII)
- Compliance with HIPAA standards

Part 2: Implementation

F. Implement the proposed database design in MongoDB

F1 - Write script to create a database named “D597 Task 2” and its respective collections (patients and fitness trackers)



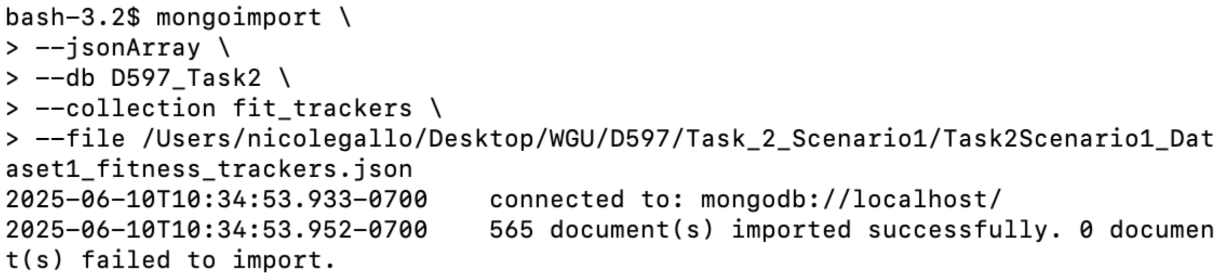
```
>_MONGOSH
> use D597_Task2
< switched to db D597_Task2
> db.createCollection("patients");
< { ok: 1 }
> db.createCollection("fit_trackers");
< { ok: 1 }
> show collections
< fit_trackers
  patients
```

F2 - Write script to insert the data records from JSON file into the database (fitness_trackers.json and medical.json)

Fitness Trackers

Using the terminal & bash$ to import datasets

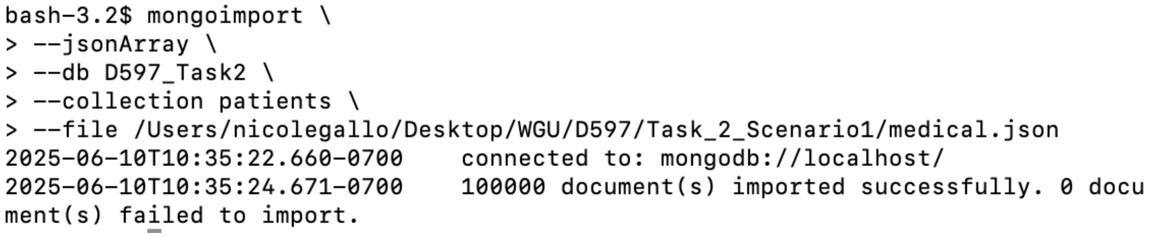
```
mongoimport \
--jsonArray \
--db D597_Task2 \
--collection fit_trackers \
--file
/Users/nicolegallo/Desktop/WGU/D597/Task_2_Scenario1/Task2Scenario1_Dataset1_fitness_
trackers.json
```



```
bash-3.2$ mongoimport \
> --jsonArray \
> --db D597_Task2 \
> --collection fit_trackers \
> --file /Users/nicolegallo/Desktop/WGU/D597/Task_2_Scenario1/Task2Scenario1_Dat
aset1_fitness_trackers.json
2025-06-10T10:34:53.933-0700    connected to: mongodb://localhost/
2025-06-10T10:34:53.952-0700    565 document(s) imported successfully. 0 documen
t(s) failed to import.
```

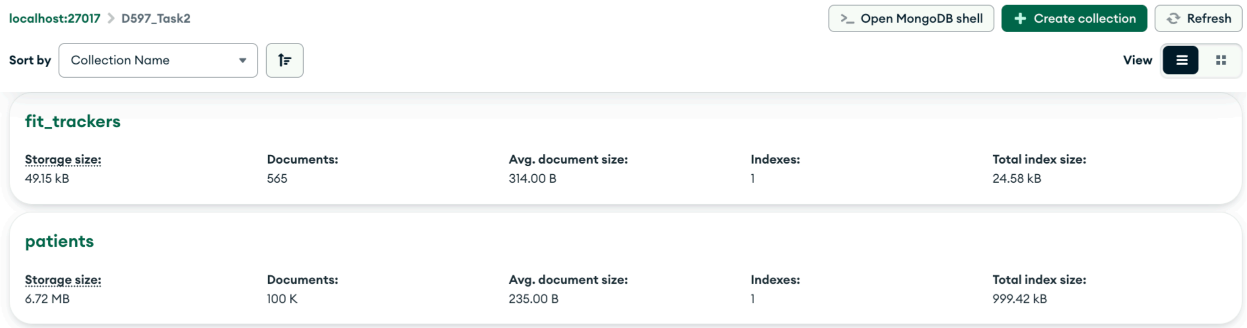
Patients

```
mongoimport \
--jsonArray \
--db D597_Task2 \
--collection patients \
--file /Users/nicolegallo/Desktop/WGU/D597/Task_2_Scenario1/medical.json
```



```
bash-3.2$ mongoimport \
> --jsonArray \
> --db D597_Task2 \
> --collection patients \
> --file /Users/nicolegallo/Desktop/WGU/D597/Task_2_Scenario1/medical.json
2025-06-10T10:35:22.660-0700    connected to: mongodb://localhost/
2025-06-10T10:35:24.671-0700    100000 document(s) imported successfully. 0 docu
ment(s) failed to import.
```

MongoDB Compass UI Screenshots of Imported Data

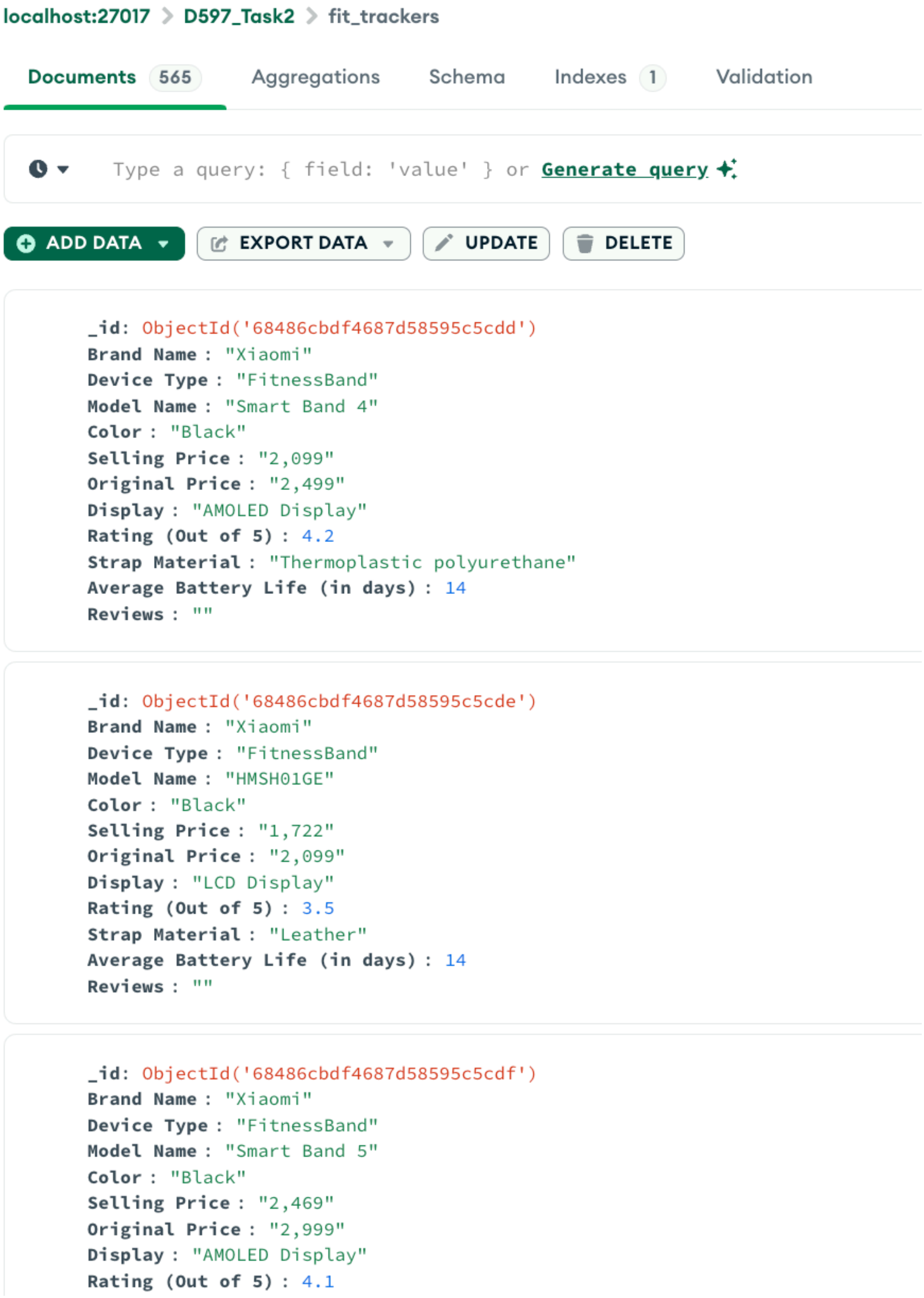


fit_trackers

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
49.15 kB	565	314.00 B	1	24.58 kB

patients

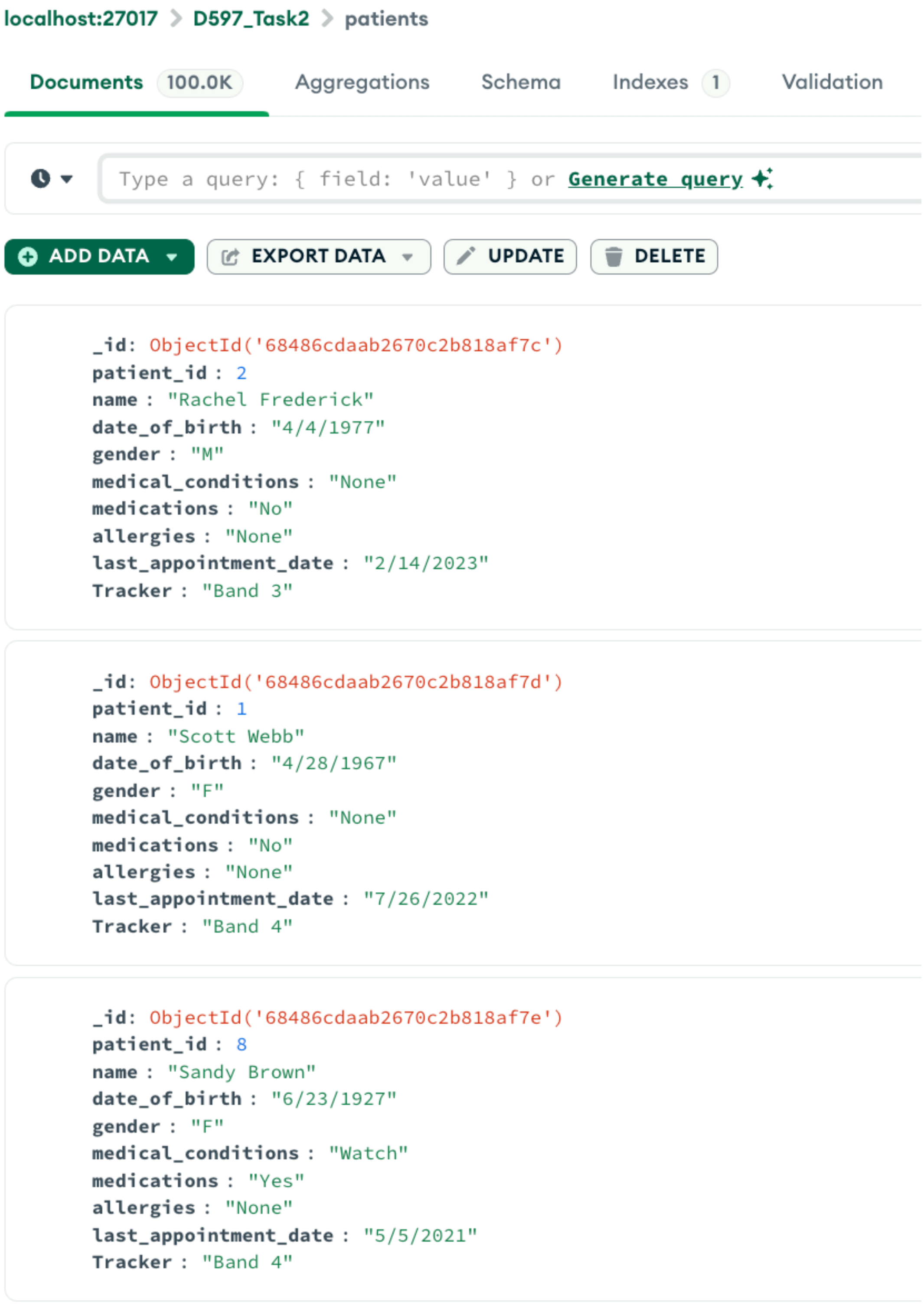
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
6.72 MB	100 K	235.00 B	1	999.42 kB


localhost:27017 > D597_Task2 > fit_trackers
Documents 565
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
ADD DATA
EXPORT DATA
UPDATE
DELETE

```
_id: ObjectId('68486cbdf4687d58595c5cdd')
Brand Name : "Xiaomi"
Device Type : "FitnessBand"
Model Name : "Smart Band 4"
Color : "Black"
Selling Price : "2,099"
Original Price : "2,499"
Display : "AMOLED Display"
Rating (Out of 5) : 4.2
Strap Material : "Thermoplastic polyurethane"
Average Battery Life (in days) : 14
Reviews : ""
```

```
_id: ObjectId('68486cbdf4687d58595c5cde')
Brand Name : "Xiaomi"
Device Type : "FitnessBand"
Model Name : "HMSH01GE"
Color : "Black"
Selling Price : "1,722"
Original Price : "2,099"
Display : "LCD Display"
Rating (Out of 5) : 3.5
Strap Material : "Leather"
Average Battery Life (in days) : 14
Reviews : ""
```

```
_id: ObjectId('68486cbdf4687d58595c5cdf')
Brand Name : "Xiaomi"
Device Type : "FitnessBand"
Model Name : "Smart Band 5"
Color : "Black"
Selling Price : "2,469"
Original Price : "2,999"
Display : "AMOLED Display"
Rating (Out of 5) : 4.1
```


localhost:27017 > D597_Task2 > patients
Documents 100.0K
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
ADD DATA
EXPORT DATA
UPDATE
DELETE

```
_id: ObjectId('68486cdaab2670c2b818af7c')
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"
```

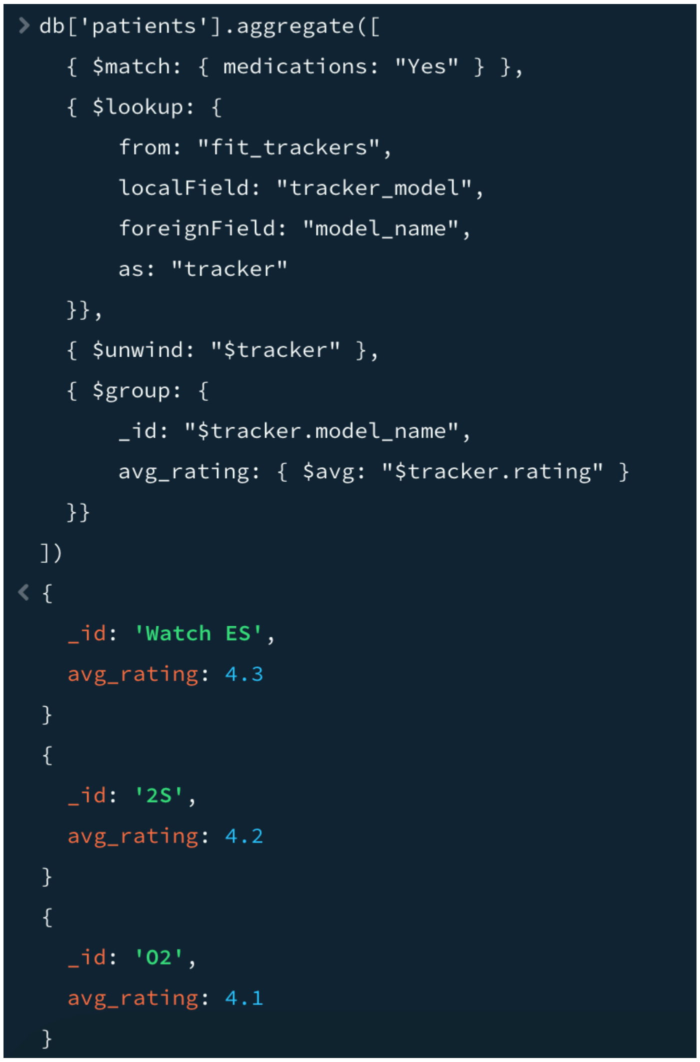
```
_id: ObjectId('68486cdaab2670c2b818af7d')
patient_id : 1
name : "Scott Webb"
date_of_birth : "4/28/1967"
gender : "F"
medical_conditions : "None"
medications : "No"
allergies : "None"
last_appointment_date : "7/26/2022"
Tracker : "Band 4"
```

```
_id: ObjectId('68486cdaab2670c2b818af7e')
patient_id : 8
name : "Sandy Brown"
date_of_birth : "6/23/1927"
gender : "F"
medical_conditions : "Watch"
medications : "Yes"
allergies : "None"
last_appointment_date : "5/5/2021"
Tracker : "Band 4"
```

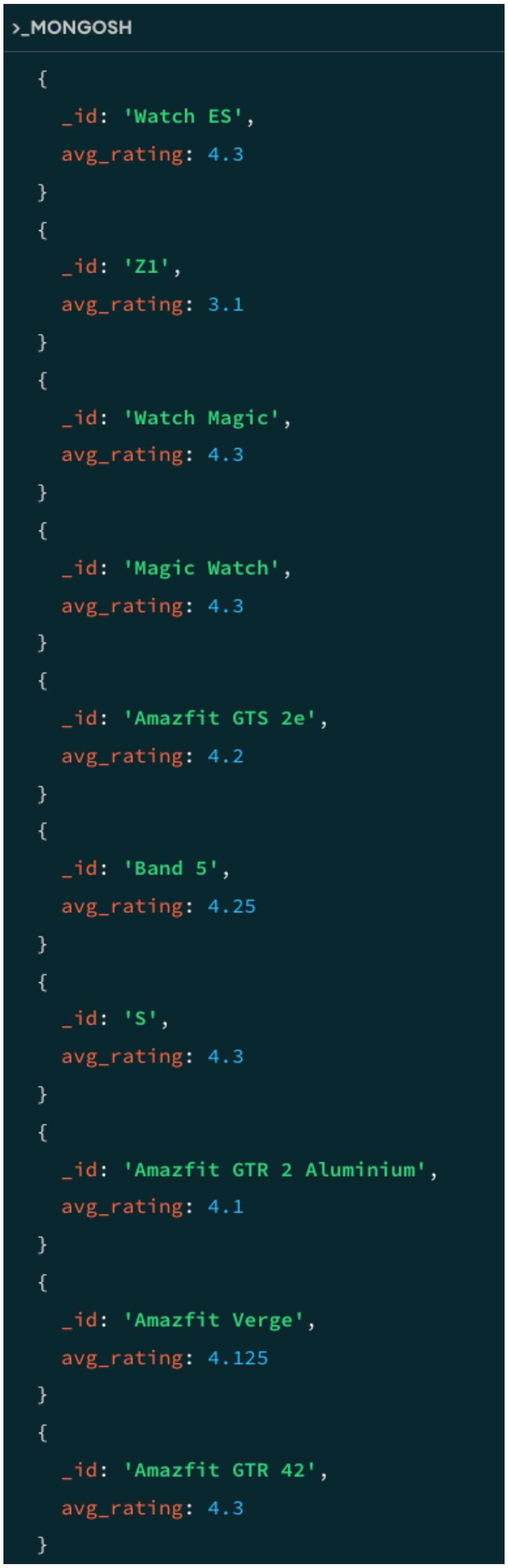
F3 - Write script for **three** queries to solve the identified business problem

Query #1 - Average Rating by Tracker Model for Medicated Patients

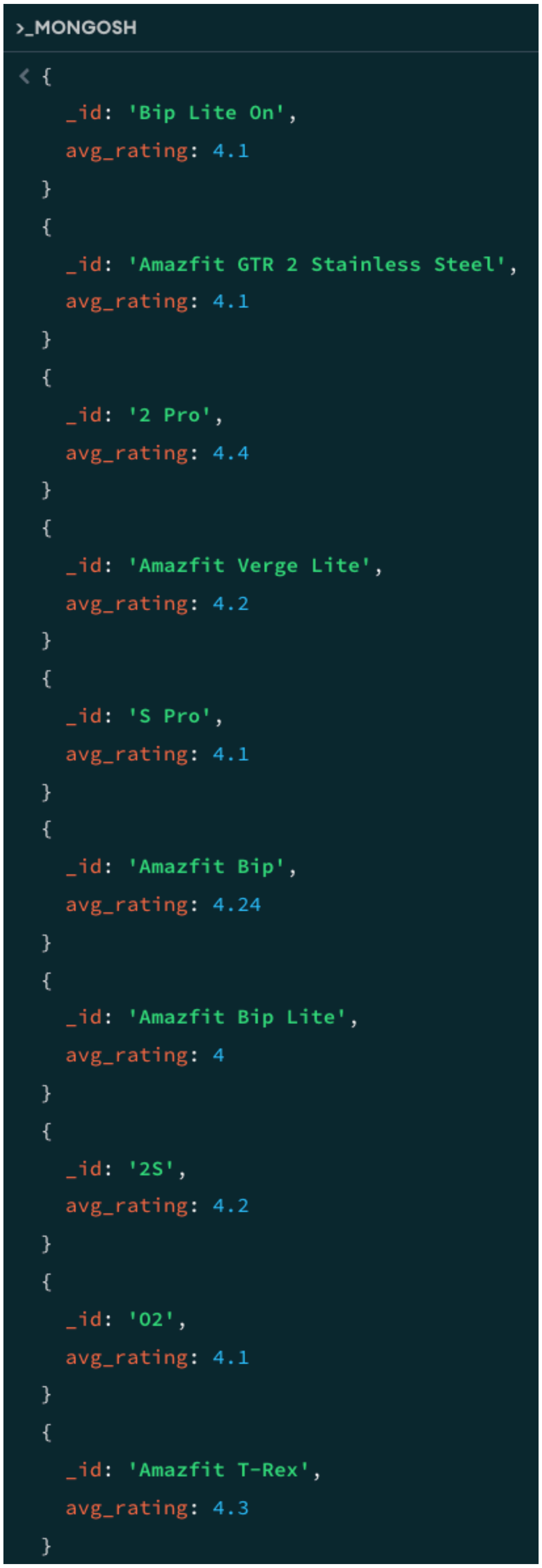
```
db['patients'].aggregate([
  { $match: { medications: "Yes" } },
  { $lookup: {
      from: "fit_trackers",
      localField: "tracker_model",
      foreignField: "model_name",
      as: "tracker"
  }},
  { $unwind: "$tracker" },
  { $group: {
      _id: "$tracker.model_name",
      avg_rating: { $avg: "$tracker.rating" }
  }}
])
```



```
> db['patients'].aggregate([
    { $match: { medications: "Yes" } },
    { $lookup: {
        from: "fit_trackers",
        localField: "tracker_model",
        foreignField: "model_name",
        as: "tracker"
    }},
    { $unwind: "$tracker" },
    { $group: {
        _id: "$tracker.model_name",
        avg_rating: { $avg: "$tracker.rating" }
    }}
  ])
< {
    _id: 'Watch ES',
    avg_rating: 4.3
  }
  {
    _id: '2S',
    avg_rating: 4.2
  }
  {
    _id: 'O2',
    avg_rating: 4.1
  }
```



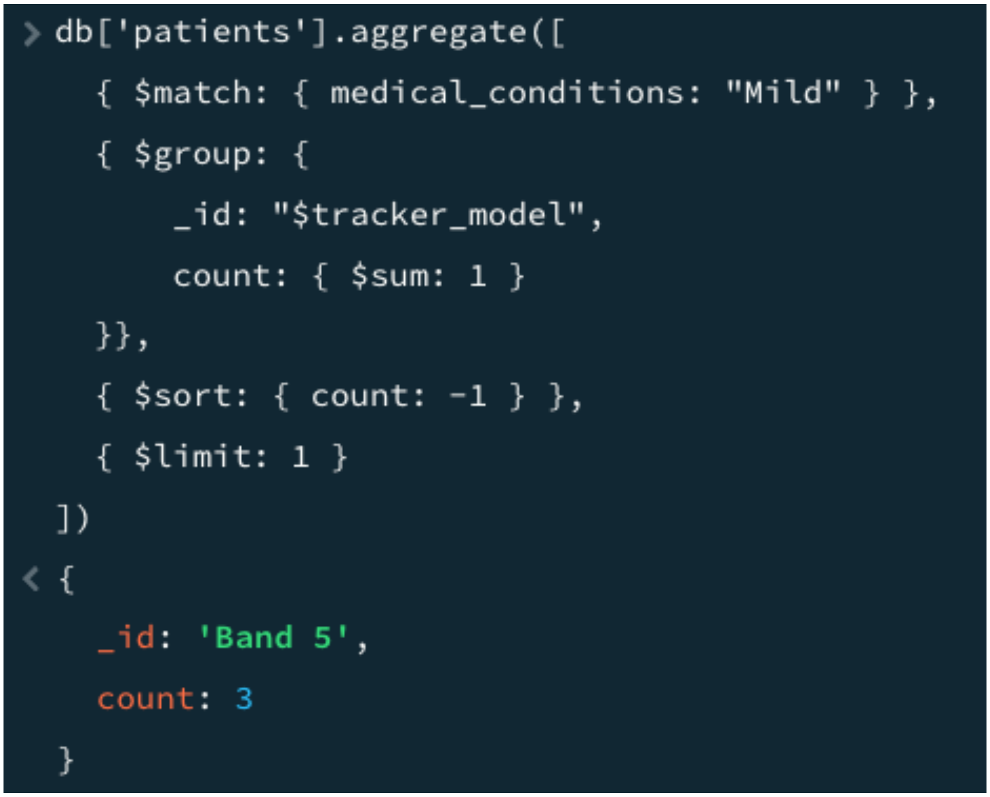
```
{
  _id: 'Watch ES',
  avg_rating: 4.3
}
{
  _id: 'Z1',
  avg_rating: 3.1
}
{
  _id: 'Watch Magic',
  avg_rating: 4.3
}
{
  _id: 'Magic Watch',
  avg_rating: 4.3
}
{
  _id: 'Amazfit GTS 2e',
  avg_rating: 4.2
}
{
  _id: 'Band 5',
  avg_rating: 4.25
}
{
  _id: 'S',
  avg_rating: 4.3
}
{
  _id: 'Amazfit GTR 2 Aluminium',
  avg_rating: 4.1
}
{
  _id: 'Amazfit Verge',
  avg_rating: 4.125
}
{
  _id: 'Amazfit GTR 42',
  avg_rating: 4.3
}
```



```
< {
    _id: 'Bip Lite On',
    avg_rating: 4.1
  }
  {
    _id: 'Amazfit GTR 2 Stainless Steel',
    avg_rating: 4.1
  }
  {
    _id: '2 Pro',
    avg_rating: 4.4
  }
  {
    _id: 'Amazfit Verge Lite',
    avg_rating: 4.2
  }
  {
    _id: 'S Pro',
    avg_rating: 4.1
  }
  {
    _id: 'Amazfit Bip',
    avg_rating: 4.24
  }
  {
    _id: 'Amazfit Bip Lite',
    avg_rating: 4
  }
  {
    _id: '2S',
    avg_rating: 4.2
  }
  {
    _id: 'O2',
    avg_rating: 4.1
  }
  {
    _id: 'Amazfit T-Rex',
    avg_rating: 4.3
  }
```

Query #2 - Most Common Tracker Among Patients with MILD Conditions

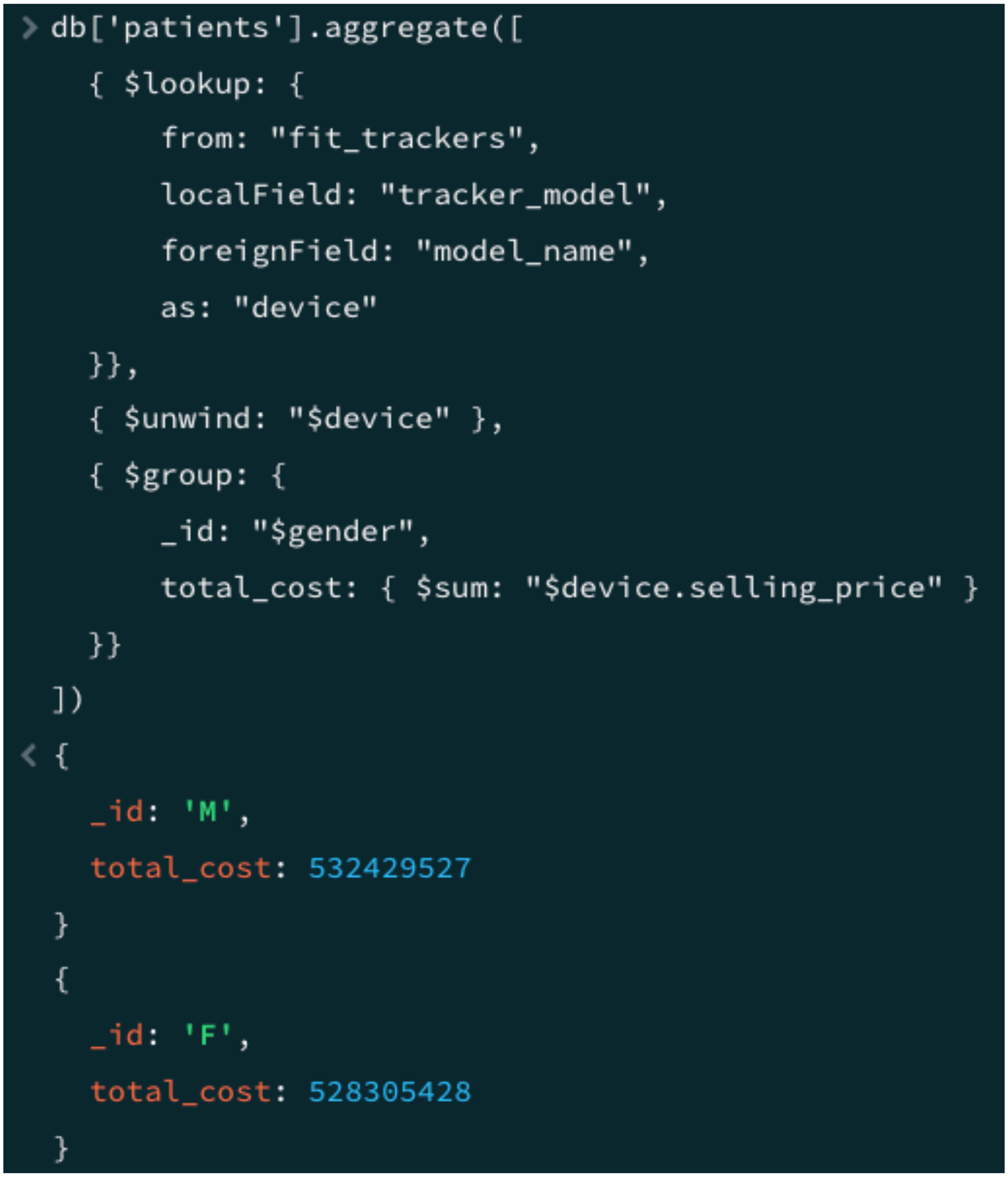
```
db['patients'].aggregate([
  { $match: { medical_conditions: "Mild" } },
  { $group: {
      _id: "$tracker_model",
      count: { $sum: 1 }
  }},
  { $sort: { count: -1 } },
  { $limit: 1 }
])
```



```
> db['patients'].aggregate([
    { $match: { medical_conditions: "Mild" } },
    { $group: {
        _id: "$tracker_model",
        count: { $sum: 1 }
    }},
    { $sort: { count: -1 } },
    { $limit: 1 }
  ])
< {
    _id: 'Band 5',
    count: 3
  }
```

Query #3 - Total Cost of Devices Assigned to Each Gender

```
db['patients'].aggregate([
  { $lookup: {
      from: "fit_trackers",
      localField: "tracker_model",
      foreignField: "model_name",
      as: "device"
  }},
  { $unwind: "$device" },
  { $group: {
      _id: "$gender",
      total_cost: { $sum: "$device.selling_price" }
  }}
])
```

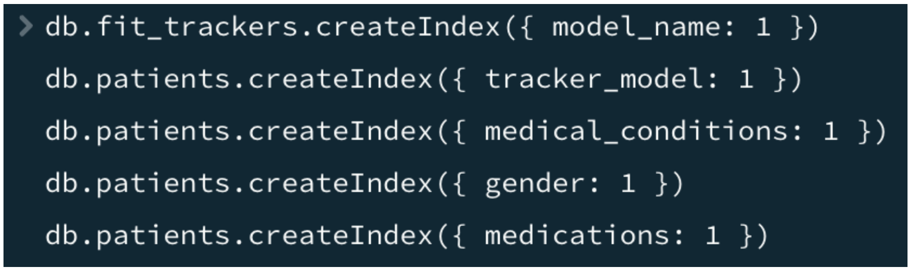


```
> db['patients'].aggregate([
    { $lookup: {
        from: "fit_trackers",
        localField: "tracker_model",
        foreignField: "model_name",
        as: "device"
    }},
    { $unwind: "$device" },
    { $group: {
        _id: "$gender",
        total_cost: { $sum: "$device.selling_price" }
    }}
  ])
< {
    _id: 'M',
    total_cost: 532429527
  }
  {
    _id: 'F',
    total_cost: 528305428
  }
```

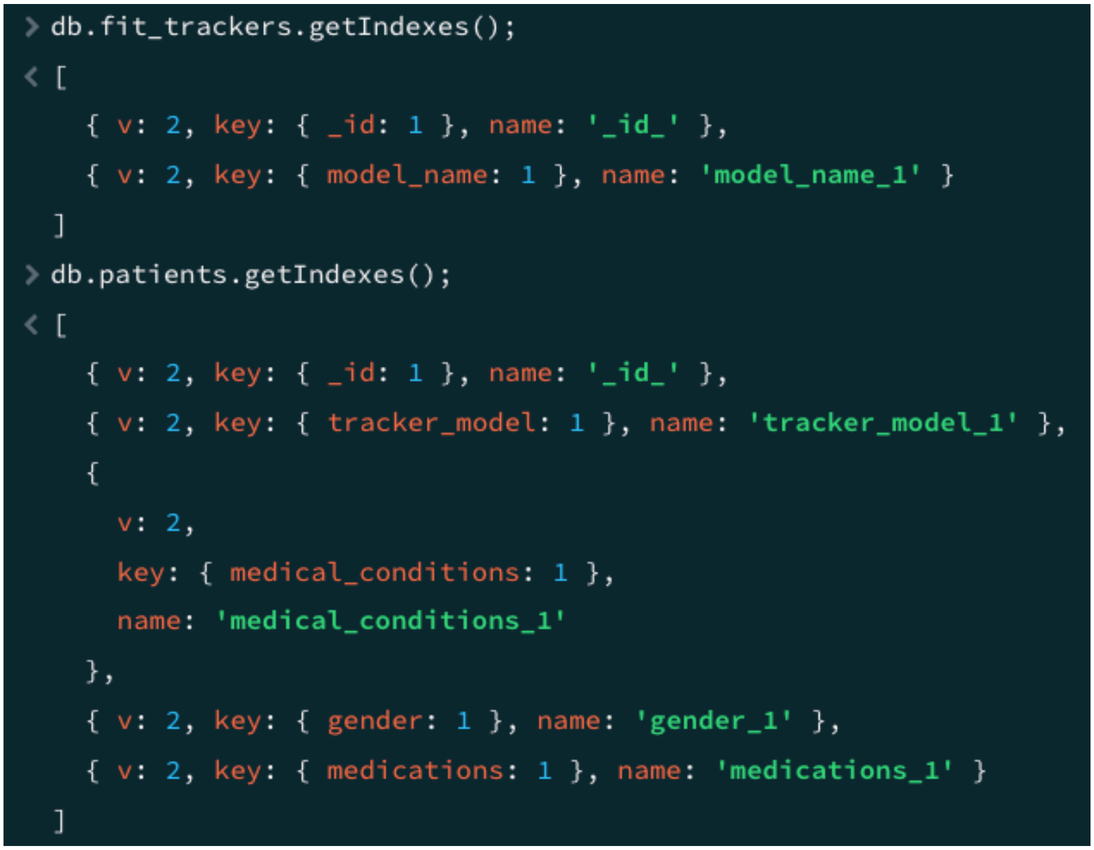
F4 – Optimization Techniques to improve runtime with Indexes

Creating indexes on `model_name`, `tracker_model`, `medical_conditions`, `medications`, and `gender`.

```
db.fit_trackers.createIndex({ model_name: 1 })

db.patients.createIndex({ tracker_model: 1 })

db.patients.createIndex({ medical_conditions: 1 })

db.patients.createIndex({ gender: 1 })

db.patients.createIndex({ medications: 1 })
```



```
> db.fit_trackers.createIndex({ model_name: 1 })
  db.patients.createIndex({ tracker_model: 1 })
  db.patients.createIndex({ medical_conditions: 1 })
  db.patients.createIndex({ gender: 1 })
  db.patients.createIndex({ medications: 1 })
```

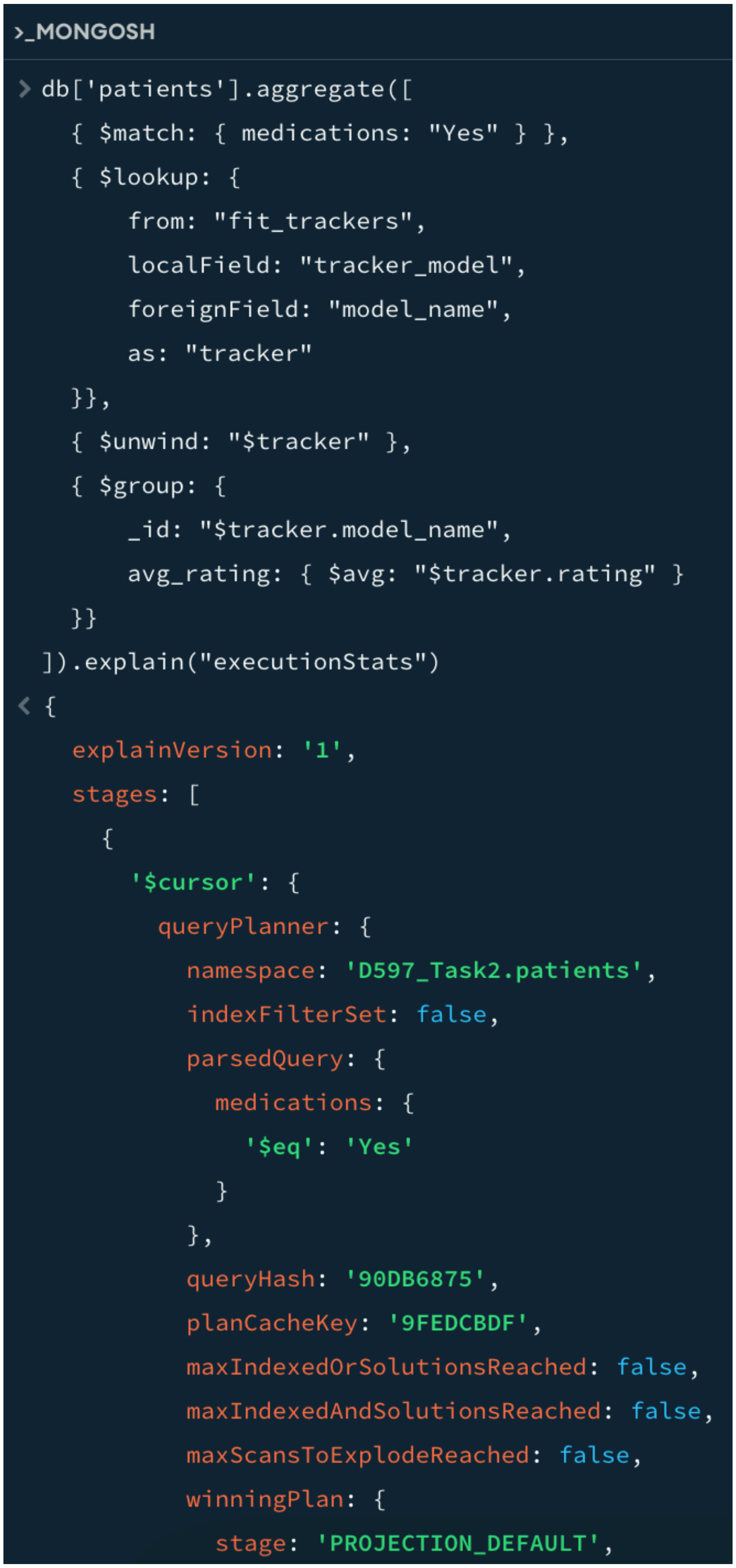


```
> db.fit_trackers.getIndexes();
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { model_name: 1 }, name: 'model_name_1' }
  ]
> db.patients.getIndexes();
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { tracker_model: 1 }, name: 'tracker_model_1' },
    {
      v: 2,
      key: { medical_conditions: 1 },
      name: 'medical_conditions_1'
    },
    { v: 2, key: { gender: 1 }, name: 'gender_1' },
    { v: 2, key: { medications: 1 }, name: 'medications_1' }
  ]
```

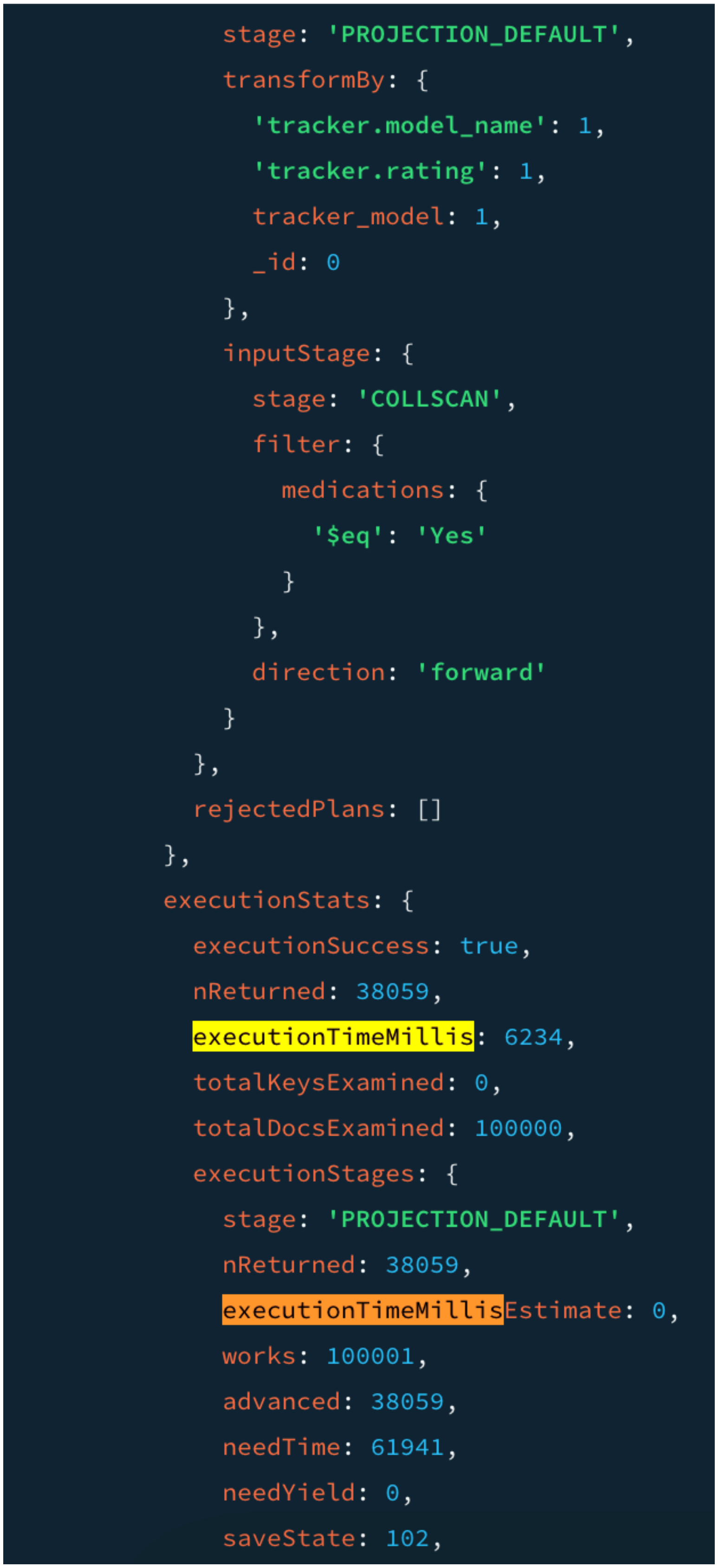
To test before and after execution runtime metrics, we can use `.explain("executionStats")`.

Query #1 - Average Rating by Tracker Model for Medicated Patients

Before index – `executionTimeMillis: 6234`

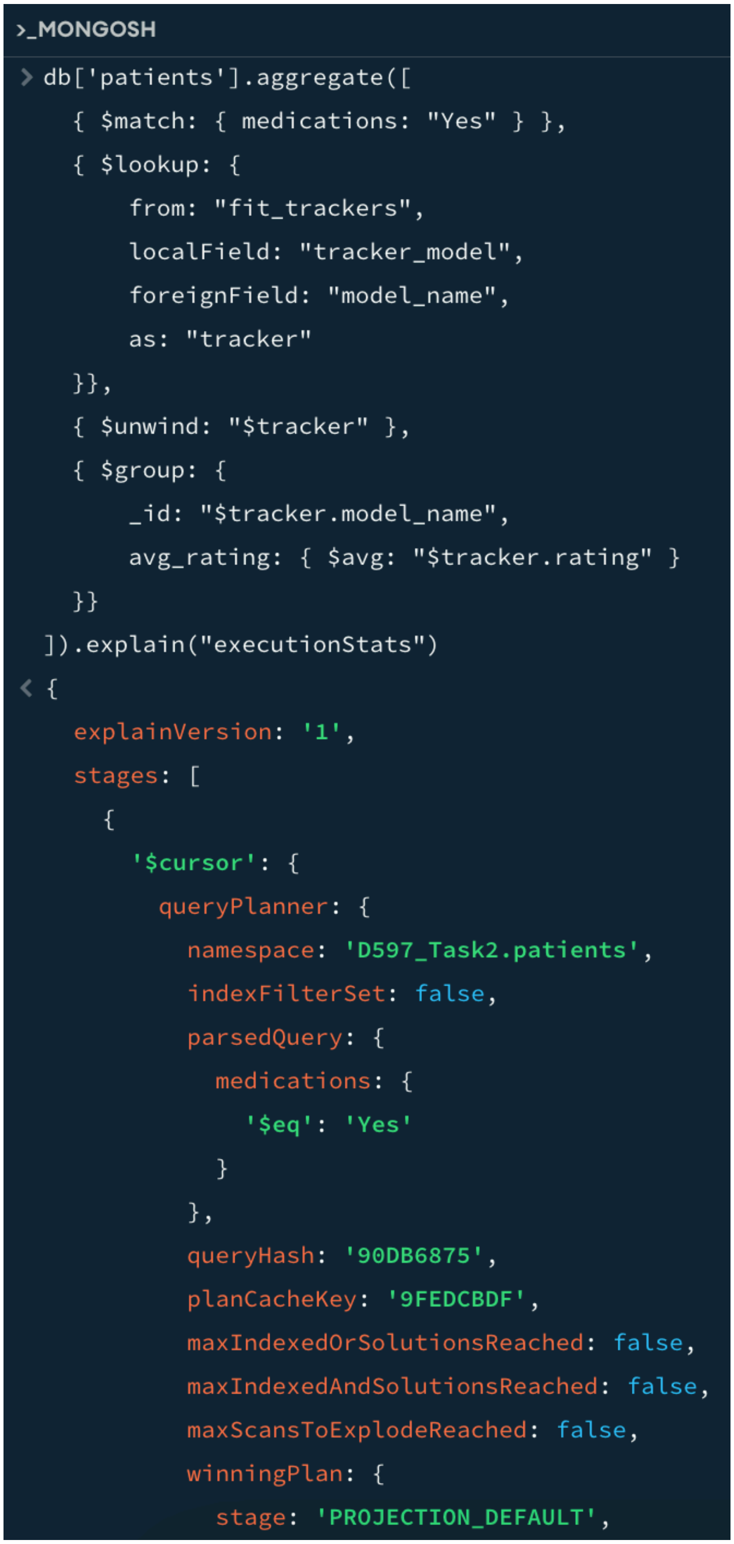


```
> db['patients'].aggregate([
    { $match: { medications: "Yes" } },
    { $lookup: {
        from: "fit_trackers",
        localField: "tracker_model",
        foreignField: "model_name",
        as: "tracker"
    }},
    { $unwind: "$tracker" },
    { $group: {
        _id: "$tracker.model_name",
        avg_rating: { $avg: "$tracker.rating" }
    }}
  ]).explain("executionStats")
< {
    explainVersion: '1',
    stages: [
      {
        '$cursor': {
          queryPlanner: {
            namespace: 'D597_Task2.patients',
            indexFilterSet: false,
            parsedQuery: {
              medications: {
                '$eq': 'Yes'
              }
            },
            queryHash: '90DB6875',
            planCacheKey: '9FEDCBDF',
            maxIndexedOrSolutionsReached: false,
            maxIndexedAndSolutionsReached: false,
            maxScansToExplodeReached: false,
            winningPlan: {
              stage: 'PROJECTION_DEFAULT',
```

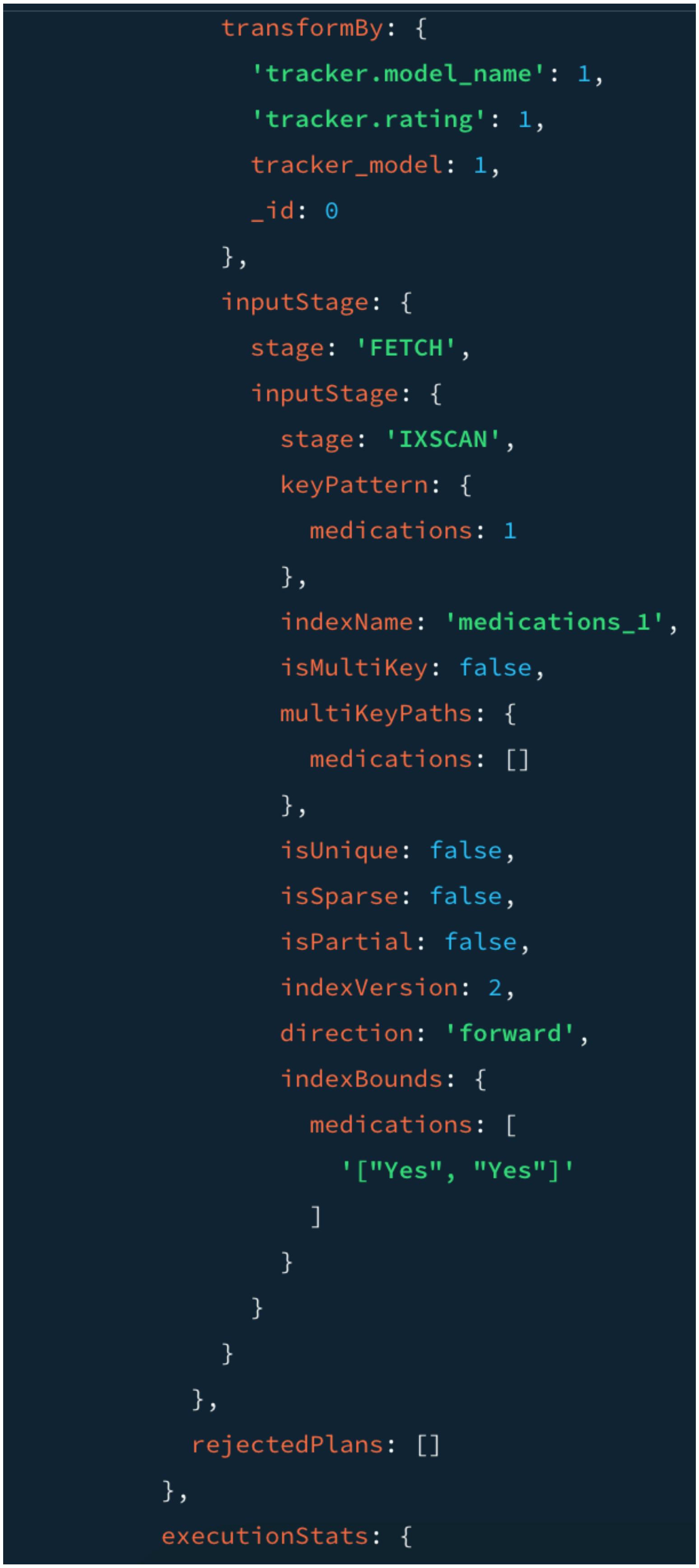


```
        stage: 'PROJECTION_DEFAULT',
        transformBy: {
          'tracker.model_name': 1,
          'tracker.rating': 1,
          tracker_model: 1,
          _id: 0
        },
        inputStage: {
          stage: 'COLLSCAN',
          filter: {
            medications: {
              '$eq': 'Yes'
            }
          },
          direction: 'forward'
        }
      },
      rejectedPlans: []
    },
    executionStats: {
      executionSuccess: true,
      nReturned: 38059,
      executionTimeMillis: 6234,
      totalKeysExamined: 0,
      totalDocsExamined: 100000,
      executionStages: {
        stage: 'PROJECTION_DEFAULT',
        nReturned: 38059,
        executionTimeMillisEstimate: 0,
        works: 100001,
        advanced: 38059,
        needTime: 61941,
        needYield: 0,
        saveState: 102,
```

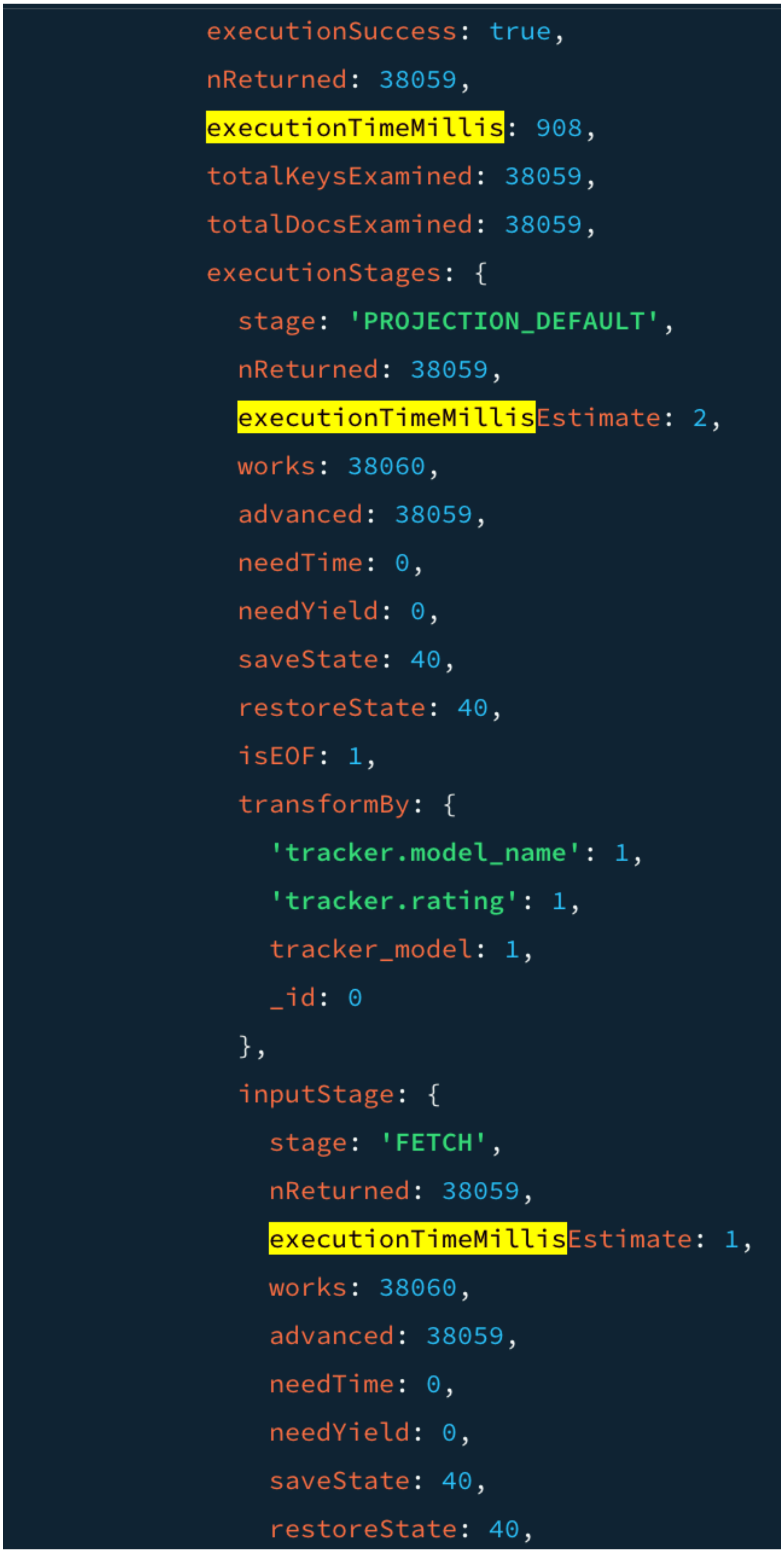
After index – `executionTimeMillis: 908`



```
> db['patients'].aggregate([
    { $match: { medications: "Yes" } },
    { $lookup: {
        from: "fit_trackers",
        localField: "tracker_model",
        foreignField: "model_name",
        as: "tracker"
    }},
    { $unwind: "$tracker" },
    { $group: {
        _id: "$tracker.model_name",
        avg_rating: { $avg: "$tracker.rating" }
    }}
  ]).explain("executionStats")
< {
    explainVersion: '1',
    stages: [
      {
        '$cursor': {
          queryPlanner: {
            namespace: 'D597_Task2.patients',
            indexFilterSet: false,
            parsedQuery: {
              medications: {
                '$eq': 'Yes'
              }
            },
            queryHash: '90DB6875',
            planCacheKey: '9FEDCBDF',
            maxIndexedOrSolutionsReached: false,
            maxIndexedAndSolutionsReached: false,
            maxScansToExplodeReached: false,
            winningPlan: {
              stage: 'PROJECTION_DEFAULT',
```



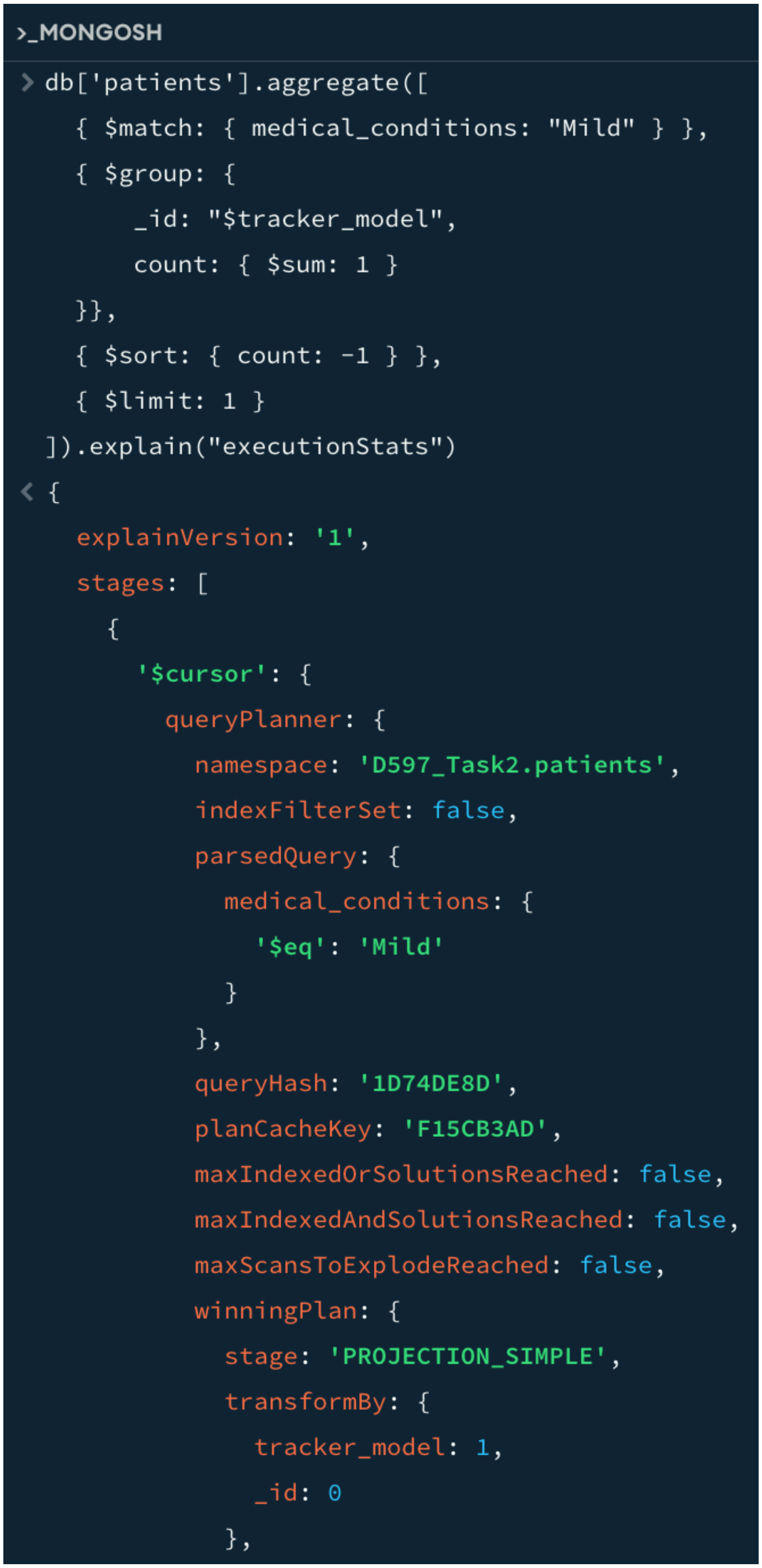
```
        transformBy: {
          'tracker.model_name': 1,
          'tracker.rating': 1,
          tracker_model: 1,
          _id: 0
        },
        inputStage: {
          stage: 'FETCH',
          inputStage: {
            stage: 'IXSCAN',
            keyPattern: {
              medications: 1
            },
            indexName: 'medications_1',
            isMultiKey: false,
            multiKeyPaths: {
              medications: []
            },
            isUnique: false,
            isSparse: false,
            isPartial: false,
            indexVersion: 2,
            direction: 'forward',
            indexBounds: {
              medications: [
                '["Yes", "Yes"]'
              ]
            }
          }
        }
      },
      rejectedPlans: []
    },
    executionStats: {
```



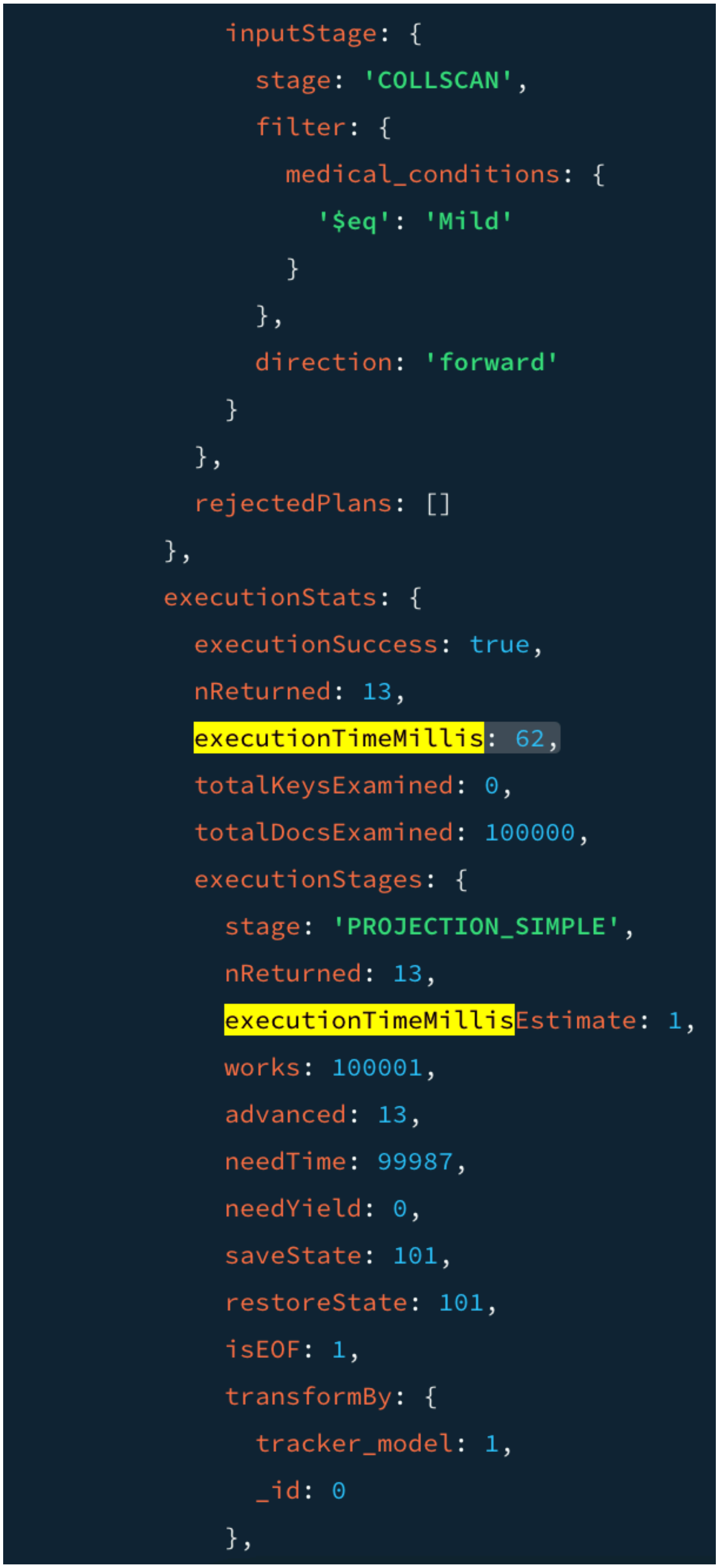
```
executionSuccess: true,
nReturned: 38059,
executionTimeMillis: 908,
totalKeysExamined: 38059,
totalDocsExamined: 38059,
executionStages: {
  stage: 'PROJECTION_DEFAULT',
  nReturned: 38059,
  executionTimeMillisEstimate: 2,
  works: 38060,
  advanced: 38059,
  needTime: 0,
  needYield: 0,
  saveState: 40,
  restoreState: 40,
  isEOF: 1,
  transformBy: {
    'tracker.model_name': 1,
    'tracker.rating': 1,
    tracker_model: 1,
    _id: 0
  },
  inputStage: {
    stage: 'FETCH',
    nReturned: 38059,
    executionTimeMillisEstimate: 1,
    works: 38060,
    advanced: 38059,
    needTime: 0,
    needYield: 0,
    saveState: 40,
    restoreState: 40,
```

Query #2 - Most Common Tracker Among Patients with MILD Conditions

Before index – `executionTimeMillis: 62`

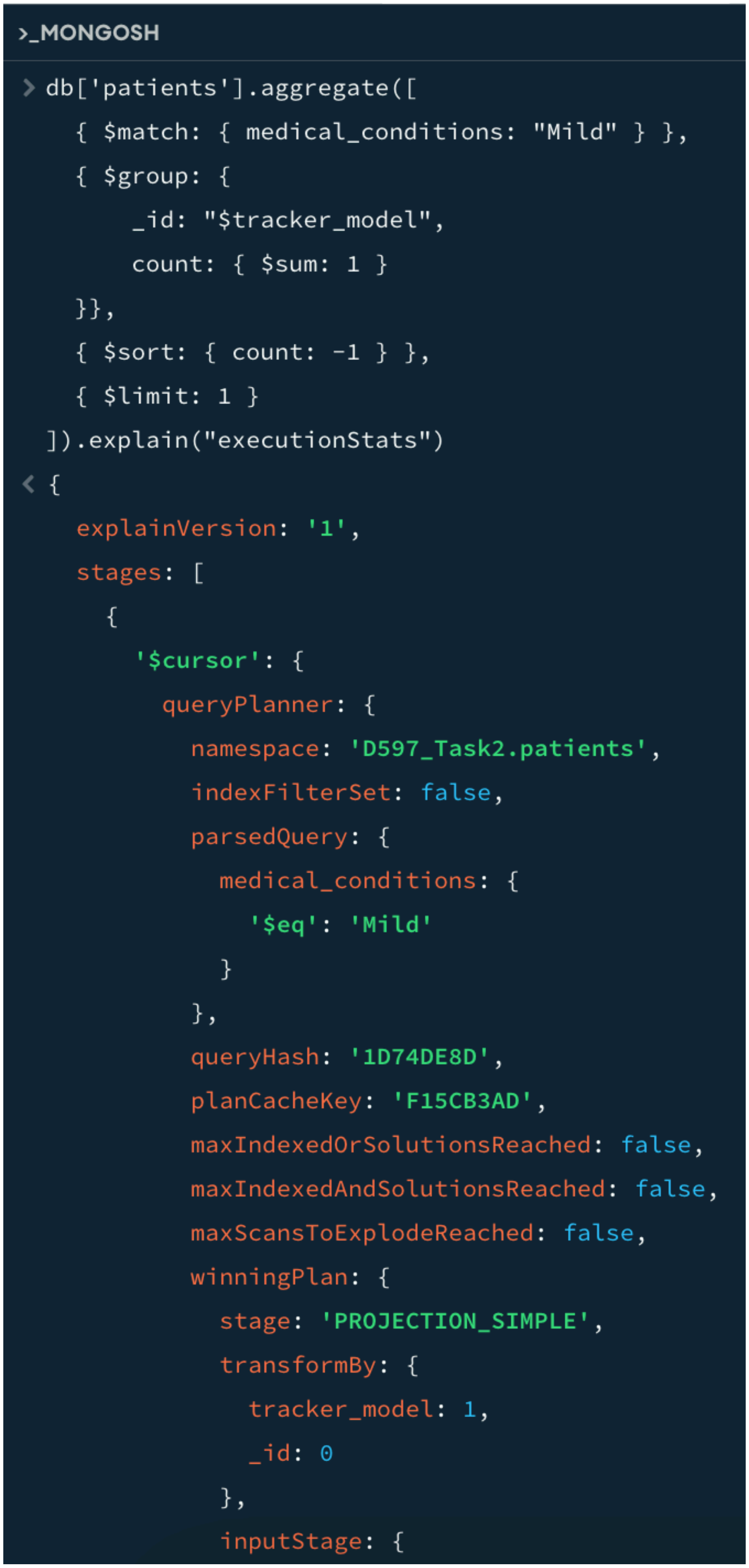


```
> db['patients'].aggregate([
    { $match: { medical_conditions: "Mild" } },
    { $group: {
        _id: "$tracker_model",
        count: { $sum: 1 }
    }},
    { $sort: { count: -1 } },
    { $limit: 1 }
  ]).explain("executionStats")
< {
    explainVersion: '1',
    stages: [
      {
        '$cursor': {
          queryPlanner: {
            namespace: 'D597_Task2.patients',
            indexFilterSet: false,
            parsedQuery: {
              medical_conditions: {
                '$eq': 'Mild'
              }
            },
            queryHash: '1D74DE8D',
            planCacheKey: 'F15CB3AD',
            maxIndexedOrSolutionsReached: false,
            maxIndexedAndSolutionsReached: false,
            maxScansToExplodeReached: false,
            winningPlan: {
              stage: 'PROJECTION_SIMPLE',
              transformBy: {
                tracker_model: 1,
                _id: 0
              },
```

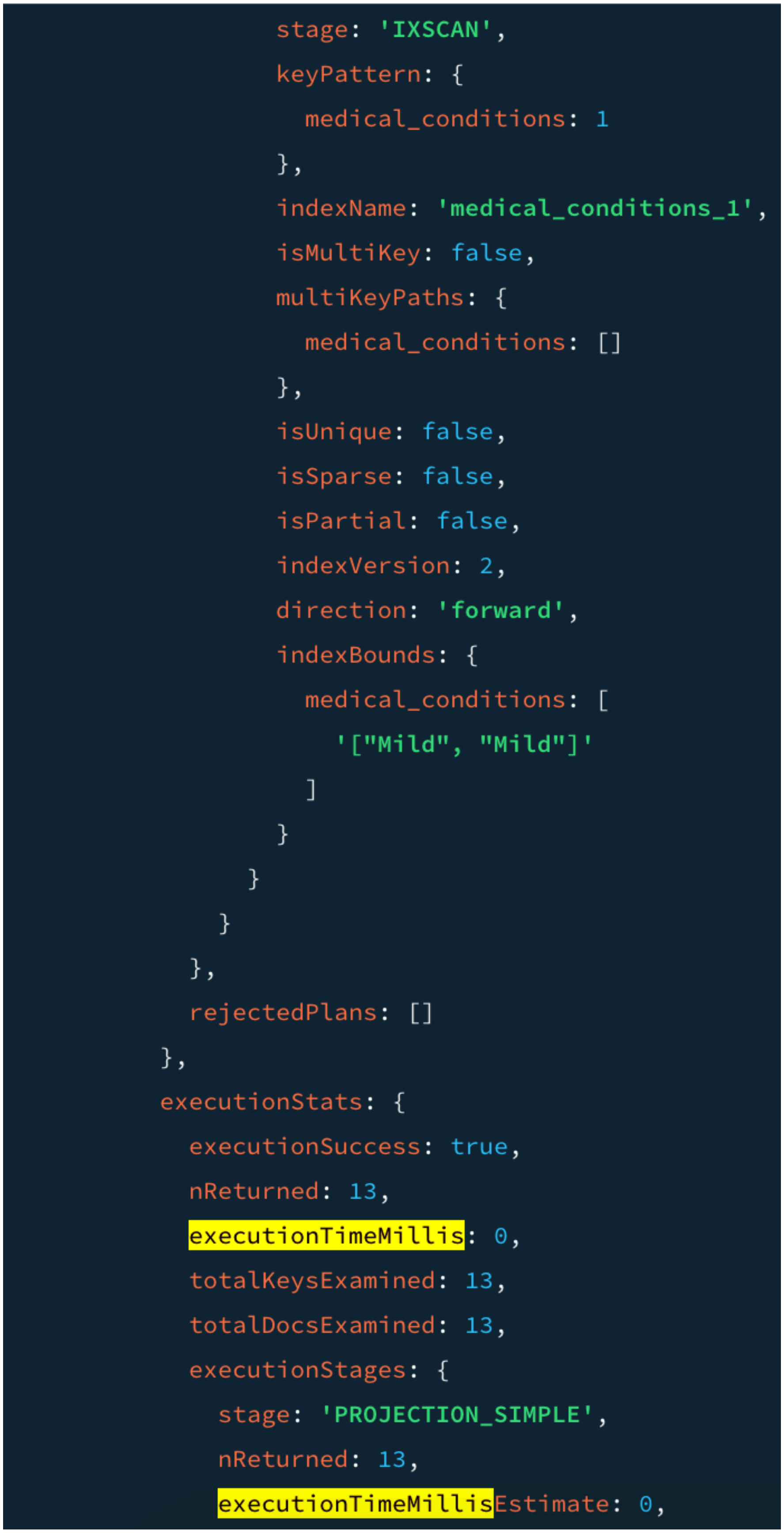


```
        inputStage: {
          stage: 'COLLSCAN',
          filter: {
            medical_conditions: {
              '$eq': 'Mild'
            }
          },
          direction: 'forward'
        }
      },
      rejectedPlans: []
    },
    executionStats: {
      executionSuccess: true,
      nReturned: 13,
      executionTimeMillis: 62,
      totalKeysExamined: 0,
      totalDocsExamined: 100000,
      executionStages: {
        stage: 'PROJECTION_SIMPLE',
        nReturned: 13,
        executionTimeMillisEstimate: 1,
        works: 100001,
        advanced: 13,
        needTime: 99987,
        needYield: 0,
        saveState: 101,
        restoreState: 101,
        isEOF: 1,
        transformBy: {
          tracker_model: 1,
          _id: 0
        },
```

After index – `executionTimeMillis: 0`



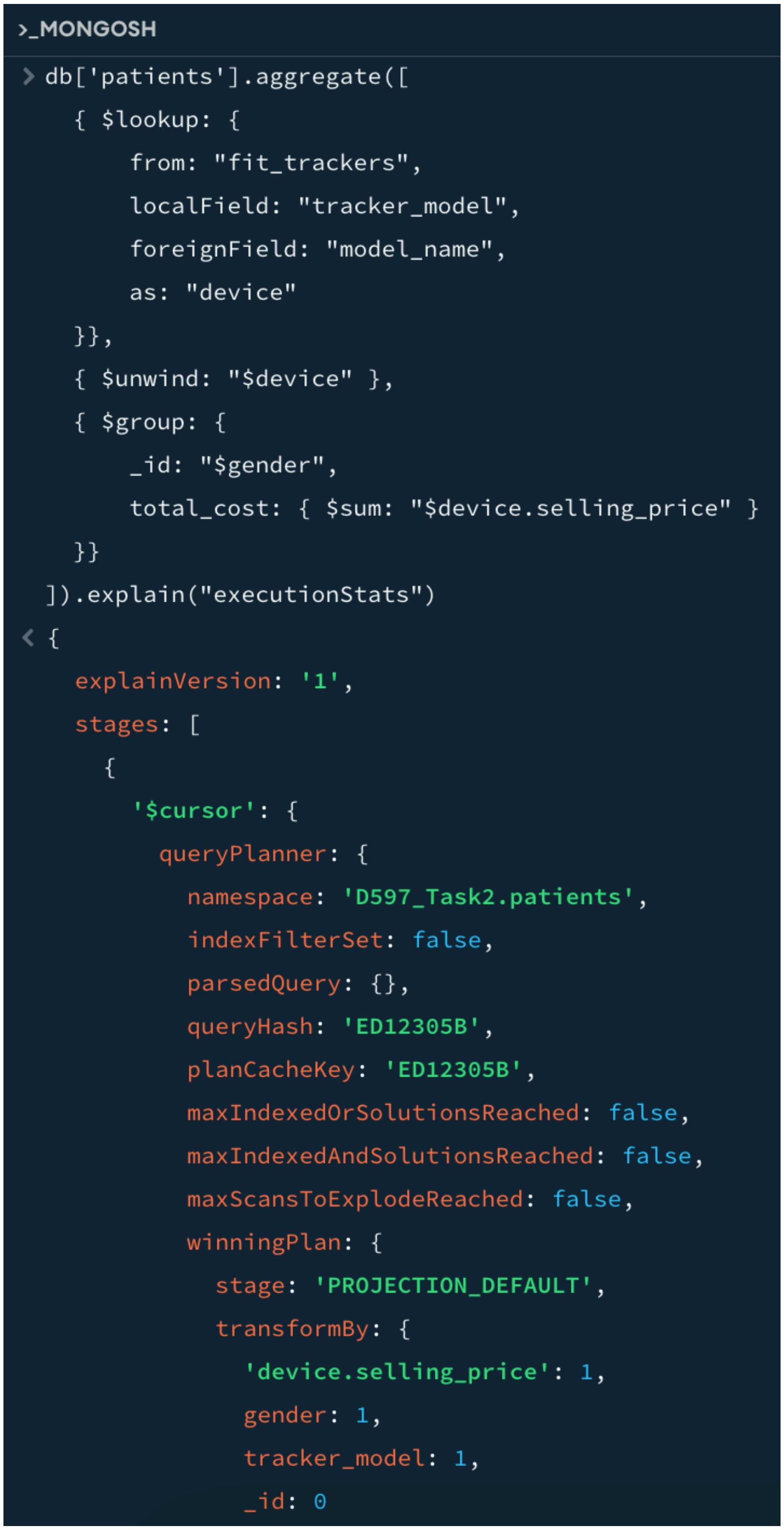
```
>_MONGOSH
> db['patients'].aggregate([
    { $match: { medical_conditions: "Mild" } },
    { $group: {
        _id: "$tracker_model",
        count: { $sum: 1 }
    }},
    { $sort: { count: -1 } },
    { $limit: 1 }
  ]).explain("executionStats")
< {
    explainVersion: '1',
    stages: [
      {
        '$cursor': {
          queryPlanner: {
            namespace: 'D597_Task2.patients',
            indexFilterSet: false,
            parsedQuery: {
              medical_conditions: {
                '$eq': 'Mild'
              }
            },
            queryHash: '1D74DE8D',
            planCacheKey: 'F15CB3AD',
            maxIndexedOrSolutionsReached: false,
            maxIndexedAndSolutionsReached: false,
            maxScansToExplodeReached: false,
            winningPlan: {
              stage: 'PROJECTION_SIMPLE',
              transformBy: {
                tracker_model: 1,
                _id: 0
              },
              inputStage: {
```



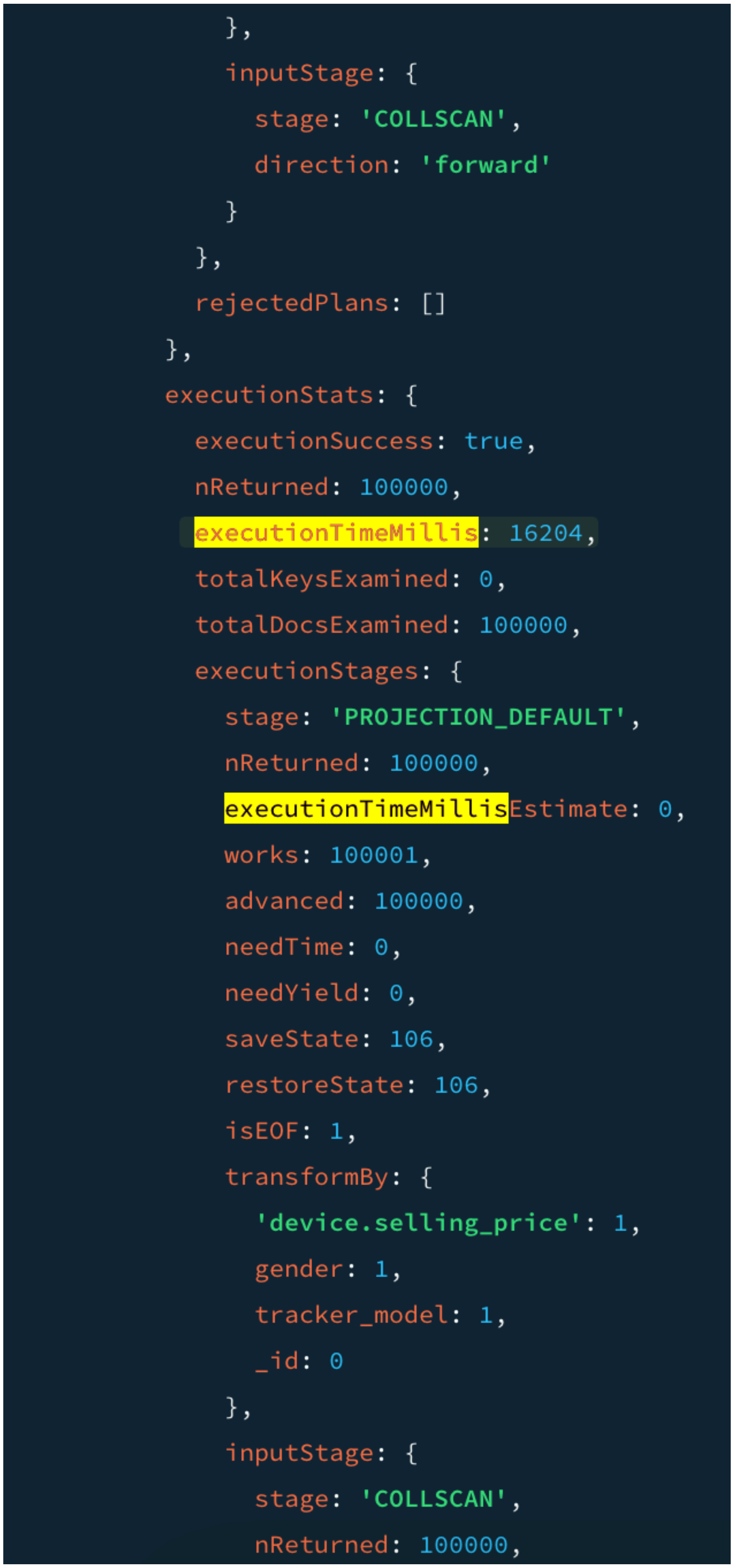
```
        stage: 'IXSCAN',
        keyPattern: {
          medical_conditions: 1
        },
        indexName: 'medical_conditions_1',
        isMultiKey: false,
        multiKeyPaths: {
          medical_conditions: []
        },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: {
          medical_conditions: [
            '["Mild", "Mild"]'
          ]
        }
      }
    }
  },
  rejectedPlans: []
},
executionStats: {
  executionSuccess: true,
  nReturned: 13,
  executionTimeMillis: 0,
  totalKeysExamined: 13,
  totalDocsExamined: 13,
  executionStages: {
    stage: 'PROJECTION_SIMPLE',
    nReturned: 13,
    executionTimeMillisEstimate: 0,
```

Query #3 - Total Cost of Devices Assigned to Each Gender

Before index – `executionTimeMillis: 16204`

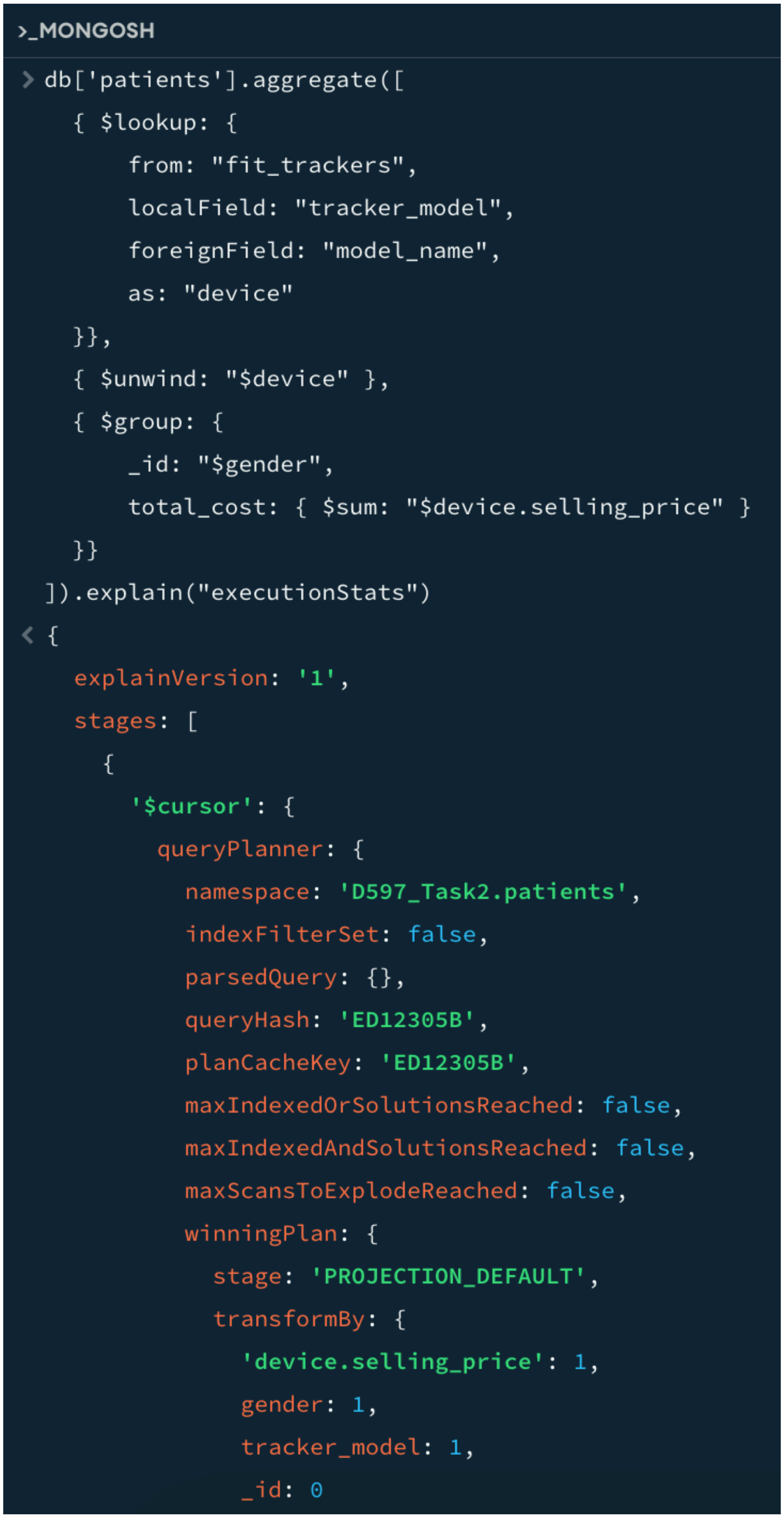


```
> db['patients'].aggregate([
    { $lookup: {
        from: "fit_trackers",
        localField: "tracker_model",
        foreignField: "model_name",
        as: "device"
    }},
    { $unwind: "$device" },
    { $group: {
        _id: "$gender",
        total_cost: { $sum: "$device.selling_price" }
    }}
  ]).explain("executionStats")
< {
    explainVersion: '1',
    stages: [
      {
        '$cursor': {
          queryPlanner: {
            namespace: 'D597_Task2.patients',
            indexFilterSet: false,
            parsedQuery: {},
            queryHash: 'ED12305B',
            planCacheKey: 'ED12305B',
            maxIndexedOrSolutionsReached: false,
            maxIndexedAndSolutionsReached: false,
            maxScansToExplodeReached: false,
            winningPlan: {
              stage: 'PROJECTION_DEFAULT',
              transformBy: {
                'device.selling_price': 1,
                gender: 1,
                tracker_model: 1,
                _id: 0
```



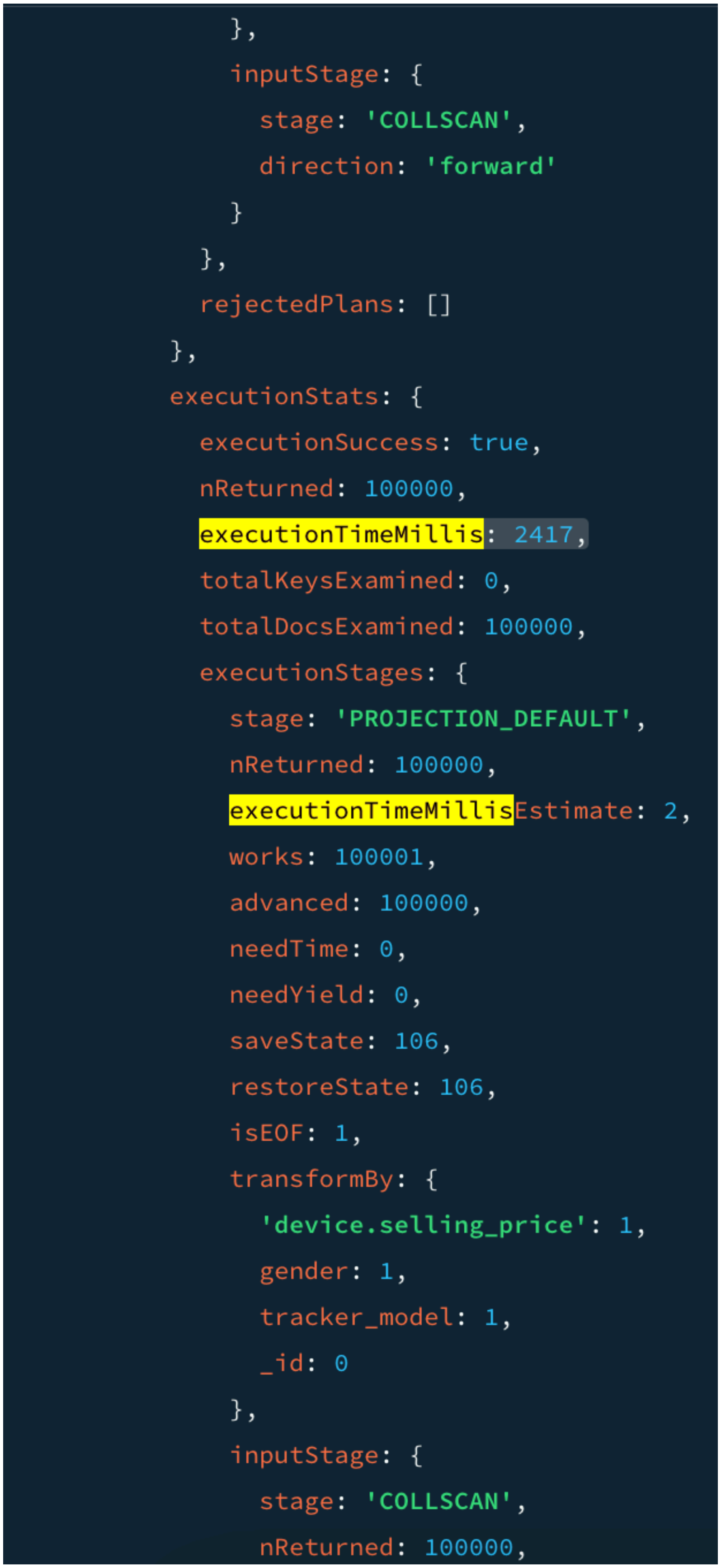
```
        },
        inputStage: {
          stage: 'COLLSCAN',
          direction: 'forward'
        }
      },
      rejectedPlans: []
    },
    executionStats: {
      executionSuccess: true,
      nReturned: 100000,
      executionTimeMillis: 16204,
      totalKeysExamined: 0,
      totalDocsExamined: 100000,
      executionStages: {
        stage: 'PROJECTION_DEFAULT',
        nReturned: 100000,
        executionTimeMillisEstimate: 0,
        works: 100001,
        advanced: 100000,
        needTime: 0,
        needYield: 0,
        saveState: 106,
        restoreState: 106,
        isEOF: 1,
        transformBy: {
          'device.selling_price': 1,
          gender: 1,
          tracker_model: 1,
          _id: 0
        },
        inputStage: {
          stage: 'COLLSCAN',
          nReturned: 100000,
```

After index – `executionTimeMillis: 2417`



>_MONGOSH

```
> db['patients'].aggregate([
    { $lookup: {
        from: "fit_trackers",
        localField: "tracker_model",
        foreignField: "model_name",
        as: "device"
    }},
    { $unwind: "$device" },
    { $group: {
        _id: "$gender",
        total_cost: { $sum: "$device.selling_price" }
    }}
  ]).explain("executionStats")
< {
    explainVersion: '1',
    stages: [
      {
        '$cursor': {
          queryPlanner: {
            namespace: 'D597_Task2.patients',
            indexFilterSet: false,
            parsedQuery: {},
            queryHash: 'ED12305B',
            planCacheKey: 'ED12305B',
            maxIndexedOrSolutionsReached: false,
            maxIndexedAndSolutionsReached: false,
            maxScansToExplodeReached: false,
            winningPlan: {
              stage: 'PROJECTION_DEFAULT',
              transformBy: {
                'device.selling_price': 1,
                gender: 1,
                tracker_model: 1,
                _id: 0
```



```
      },
      inputStage: {
        stage: 'COLLSCAN',
        direction: 'forward'
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 100000,
    executionTimeMillis: 2417,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
      stage: 'PROJECTION_DEFAULT',
      nReturned: 100000,
      executionTimeMillisEstimate: 2,
      works: 100001,
      advanced: 100000,
      needTime: 0,
      needYield: 0,
      saveState: 106,
      restoreState: 106,
      isEOF: 1,
      transformBy: {
        'device.selling_price': 1,
        gender: 1,
        tracker_model: 1,
        _id: 0
      },
      inputStage: {
        stage: 'COLLSCAN',
        nReturned: 100000,
```

Part 3: Presentation

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=6c59dfcd-23ed-442c-8b53-b2f9011b9b1e

H: Citations

- MongoDB documentation: https://www.mongodb.com/docs/manual/
- HIPAA privacy guidance: https://www.hhs.gov/hipaa