

# **IMAT3104: Database Management and Programming Assignment**

Georgios Karseras  
P2424630

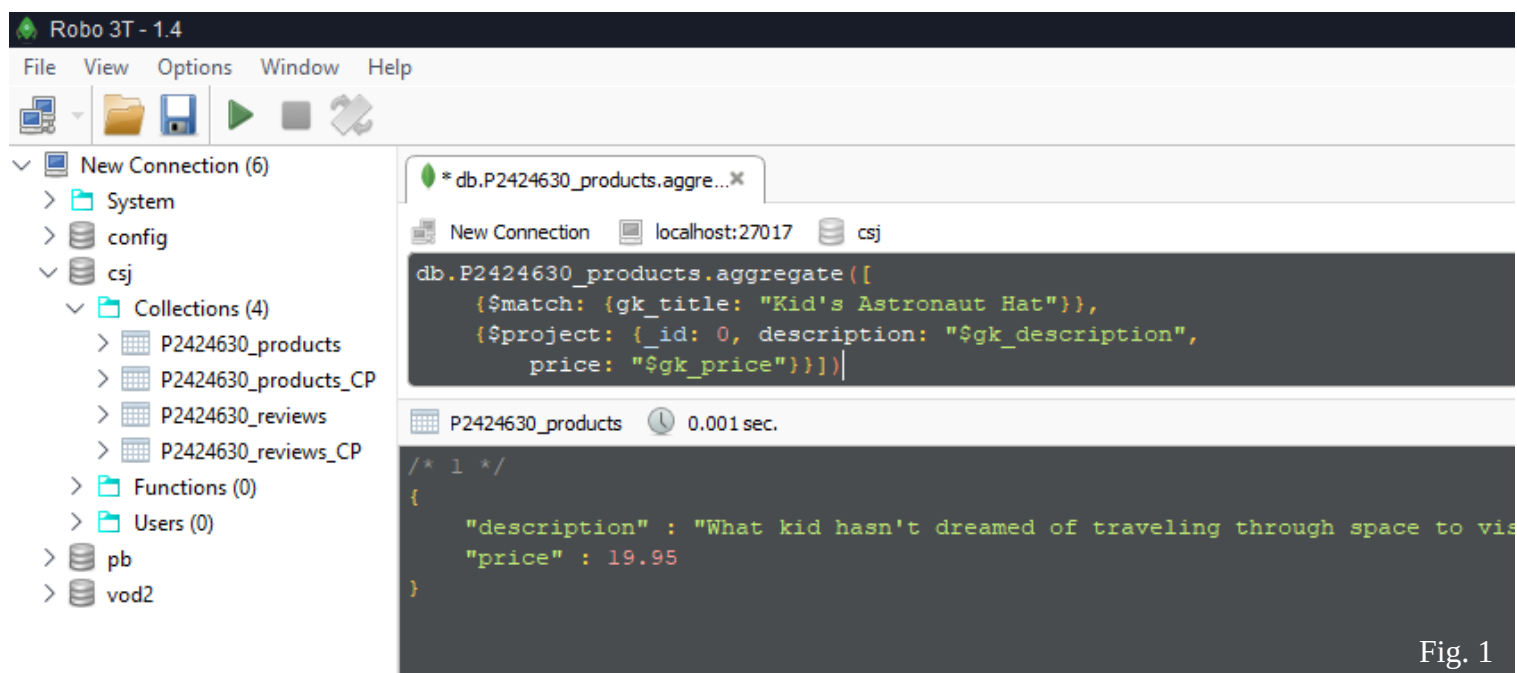
## Table of Contents

Questions:.....	3
Q1: Find the description and price of the product entitled “Kid's Astronaut Hat”. No other product details are required.....	3
Q2: Find all details of products that are priced at either \$7.99 or \$14.99.....	4
Q3: Count the number of products that have a price but do not have a description. Clearly show the number you receive as your answer.....	5
Q4: List the product IDs of products that were reviewed before the 15 <sup>th</sup> June 2000. All calculations should be performed by MongoDB. Only list the product IDs and do not show any duplicates.....	6
Q5: List all of the reviewers who have written at least 3 reviews. Show the reviewer ID, reviewer name and the number of reviews for each one. List the names of reviewers in alphabetical order. [Hint: it is possible to solve using aggregate pipeline method].....	7
Q6: Count the number of related products that were also viewed with the product “Mens Timex”. Only give the number of related products as your answer. [Hint: it is possible to solve using aggregate pipeline method].....	8
Q7: List the product ID and title of all products in the “T-Shirts” category. No other product details are required. [Hint: it is possible to solve using aggregate pipeline method].....	9
Q8: Find the highest rating by the reviewer identified as “A2B3TVWTZ7609A” by using map reduce programming.....	10
Q9: Using both collections, find the title and description of products reviewed by the reviewer identified by “A2C65Q9IDNCSR”. No other product details are required.....	11
E1: Implement one index that would improve the querying of the database based on one or more of the queries (Q1-Q9). Identify the chosen query or queries and explain and justify your choice of index. Present, explain and compare execution plans to support your choice and summarise your findings.....	12
D1: Write code in MongoDB to automatically embed the details of reviews from the reviews collection with their corresponding product in the product collection. Therefore, the products collection will contain both the product data and review data in a single collection of products.	16

## Questions:

**Q1: Find the description and price of the product entitled “Kid's Astronaut Hat”. No other product details are required.**

```
db.P2424630_products.aggregate([
  {$match: {gk_title: "Kid's Astronaut Hat"}},
  {$project: {_id: 0, description: "$gk_description",
    price: "$gk_price"}}])
```



**Q2: Find all details of products that are priced at either \$7.99 or \$14.99.**

```
db.P2424630_products.aggregate([
  {$match: {gk_price: {$in: [7.99, 14.99]}}}]])
```

The screenshot shows the Robo 3T - 1.4 interface. On the left, the 'Collections (4)' list includes 'P2424630\_products'. The main window displays the aggregation query: `db.P2424630_products.aggregate([{$match: {gk_price: {$in: [7.99, 14.99]}}}]])`. Below the query, the results are shown in a table with columns 'Key', 'Value', and 'Type'.

Key	Value	Type
(1) ObjectId("5d0b6d1cd7367de7f58b497f")	{ 8 fields }	Object
_id	ObjectId("5d0b6d1cd7367de7f58b497f")	ObjectId
gk_asin	7842953562	String
gk_title	Blessed By Pope Benedetto XVI Our Lady of Charity Medal- Caridad D...	String
gk_price	14.99	Double
gk_imUrl	http://ecx.images-amazon.com/images/I/41C3QDHCWML_SY300_.jpg	String
gk_related	{ 1 field }	Object
gk_salesRank	{ 1 field }	Object
gk_categories	[ 2 elements ]	Array
(2) ObjectId("5d0b6d1cd7367de7f58b4980")	{ 7 fields }	Object
_id	ObjectId("5d0b6d1cd7367de7f58b4980")	ObjectId
gk_asin	7842955867	String
gk_title	Blessed By Pope Benedetto XVI Padre Pio Rose Scented Bracelet &am...	String
gk_price	7.99	Double
gk_imUrl	http://ecx.images-amazon.com/images/I/41EvNYKS8L_SY300_.jpg	String
gk_salesRank	{ 1 field }	Object
gk_categories	[ 2 elements ]	Array
(3) ObjectId("5d0b6d1cd7367de7f58b49ea")	{ 10 fields }	Object
_id	ObjectId("5d0b6d1cd7367de7f58b49ea")	ObjectId
gk_asin	B00000755K	String
gk_related	{ 3 fields }	Object
gk_title	Girls Will Be Girls	String
gk_price	14.99	Double
gk_salesRank	{ 1 field }	Object
gk_imUrl	http://ecx.images-amazon.com/images/I/51B5jGH2CuL_SX300_.jpg	String
gk_brand		String
gk_categories	[ 3 elements ]	Array
gk_description	Shot in High Definition. Yes it's true, girls will be girls. But at the end o...	String
(4) ObjectId("5d0b6d1cd7367de7f58b4a80")	{ 7 fields }	Object
_id	ObjectId("5d0b6d1cd7367de7f58b4a80")	ObjectId
gk_asin	B00005KJXQ	String
gk_title	Rubie's Howling Ghost Costume	String
gk_price	7.99	Double
gk_imUrl	http://ecx.images-amazon.com/images/I/31n-113Dr7L_SY445_.jpg	String
gk_brand	Rubie&#39;s Costume Co	String
gk_categories	[ 1 element ]	Array

Fig. 2

**Q3: Count the number of products that have a price but do not have a description. Clearly show the number you receive as your answer.**

```
db.P2424630_products.aggregate([
  {$match: {gk_price: {$exists: true},
    gk_description: {$exists: false}}},
  {$count: "hasPriceAndNoDescription"}])
```

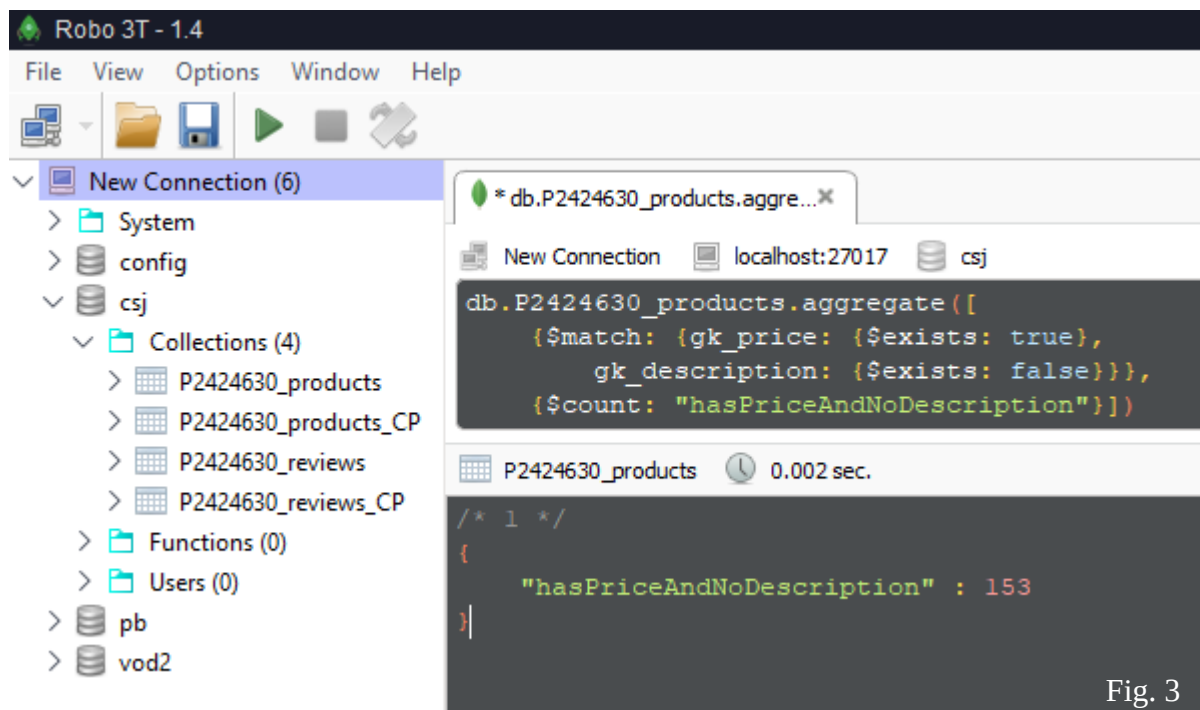


Fig. 3

**Q4: List the product IDs of products that were reviewed before the 15<sup>th</sup> June 2000. All calculations should be performed by MongoDB. Only list the product IDs and do not show any duplicates.**

```
var to_test = ISODate("2000-06-15T00:00:00.000Z").getTime() / 1000
db.P2424630_reviews.aggregate([
  {$match: {gk_unixReviewTime: {$lt: to_test}}},
  {$group: {_id: "$gk_asin"}}])
```

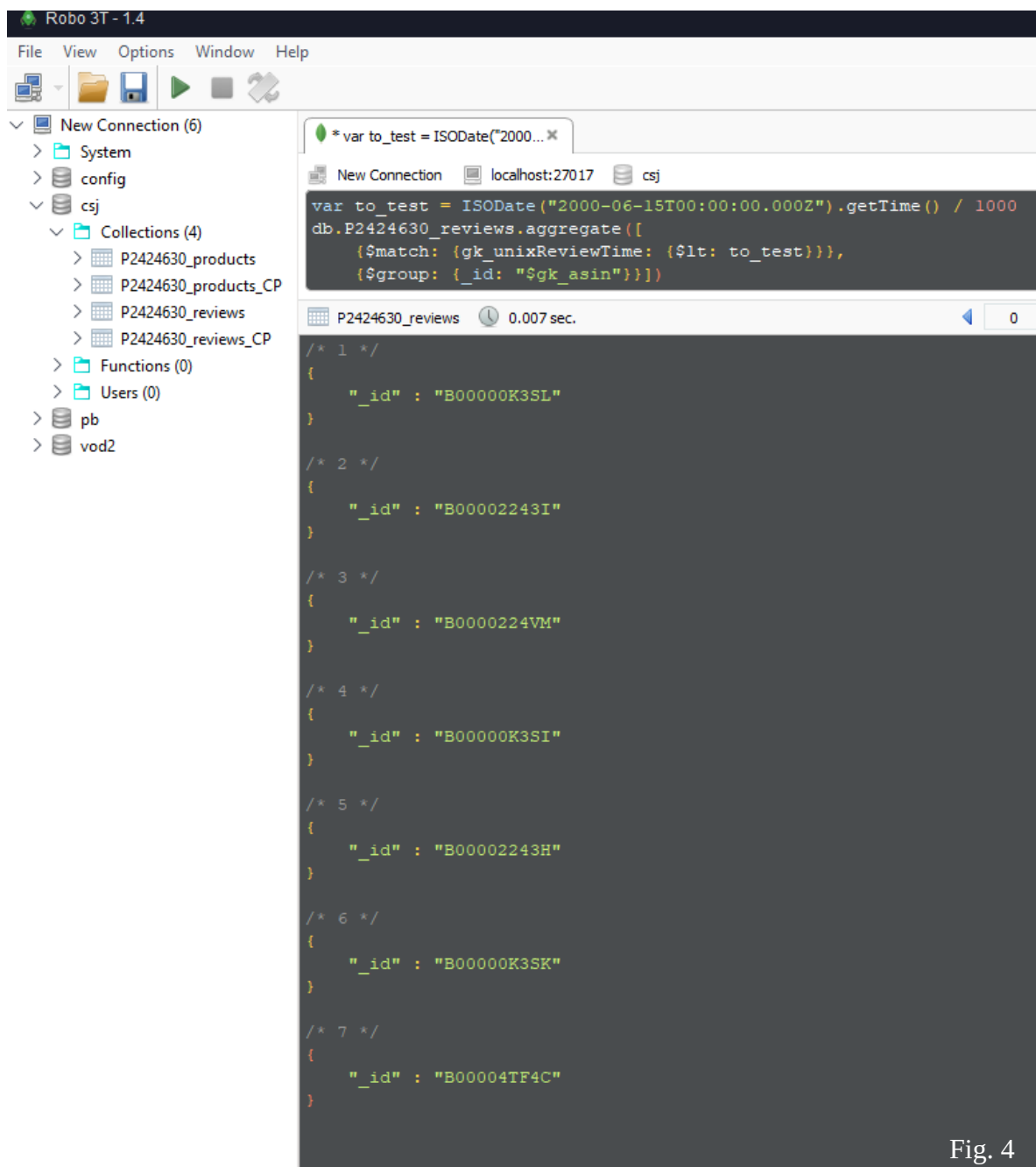


Fig. 4

**Q5: List all of the reviewers who have written at least 3 reviews. Show the reviewer ID, reviewer name and the number of reviews for each one. List the names of reviewers in alphabetical order. [Hint: it is possible to solve using aggregate pipeline method].**

```
db.P2424630_reviews.aggregate([
  {$group: {_id: {reviewerID: "$gk_reviewerID",
    reviewerName: "$gk_reviewerName"}, count: {$sum: 1}}},
  {$match: {count: {$gt: 2}}},
  {$sort: {"_id.reviewerName": 1}},
  {$project: {_id: 0, reviewerID: "$_id.reviewerID",
    reviewerName: "$_id.reviewerName",
    reviewCount: "$count"}}])
```

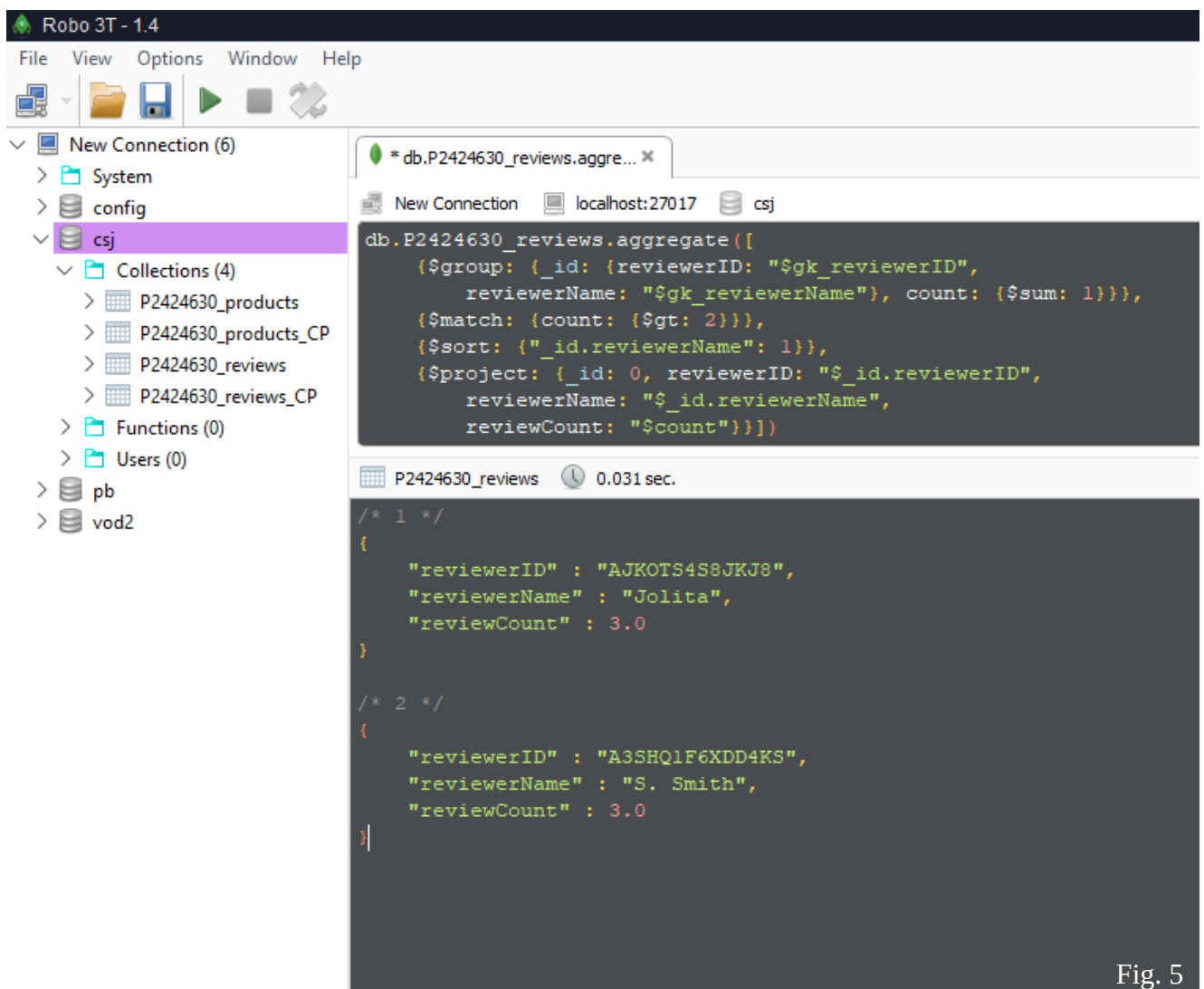


Fig. 5

**Q6: Count the number of related products that were also viewed with the product “Mens Timex”. Only give the number of related products as your answer. [Hint: it is possible to solve using aggregate pipeline method].**

```
db.P2424630_products.aggregate([
  {$match: {gk_title: "Mens Timex"}},
  {$project: {_id: 0, numRelated:
    {$size: "$gk_related.also_viewed"}}}]])
```

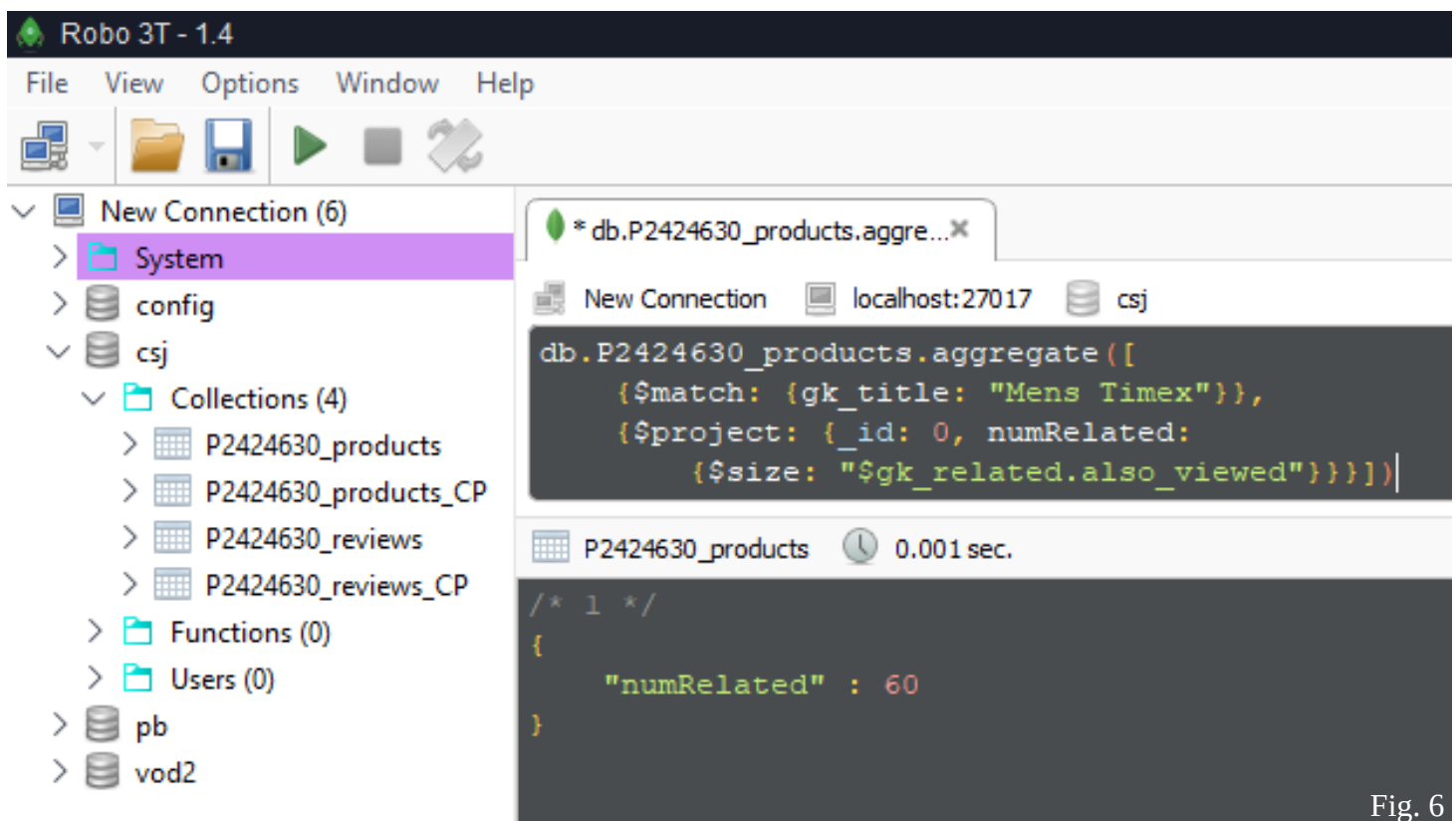


Fig. 6



**Q7: List the product ID and title of all products in the “T-Shirts” category. No other product details are required. [Hint: it is possible to solve using aggregate pipeline method].**

```
db.P2424630_products.aggregate([
  {$unwind: "$gk_categories"},
  {$match: {$expr: {$in: ["T-Shirts", "$gk_categories"]}}},
  {$project: {_id: 0, asin: "$gk_asin", title: "$gk_title"}}])
```

The screenshot shows the Robo 3T - 1.4 application interface. On the left, a sidebar displays the database structure: 'New Connection (6)' with folders for 'System', 'config', and 'csj'. Under 'csj', there are 'Collections (4)' including 'P2424630\_products', 'P2424630\_products\_CP', 'P2424630\_reviews', and 'P2424630\_reviews\_CP'. There are also 'Functions (0)', 'Users (0)', and other databases like 'pb' and 'vod2'.

The main window shows a query editor with the following aggregation pipeline:

```
db.P2424630_products.aggregate([
  {$unwind: "$gk_categories"},
  {$match: {$expr: {$in: ["T-Shirts", "$gk_categories"]}}},
  {$project: {_id: 0, asin: "$gk_asin", title: "$gk_title"}}])
```

Below the query editor, the results are displayed in a table. The table has three columns: 'Key', 'Value', and 'Type'. The results are grouped by index (1) through (8). Each group contains two fields: 'asin' and 'title'.

Key	Value	Type
(1)	{ 2 fields }	Object
asin	5555014368	String
title	Top Gear Official Merchandise - Captain Slow T-Shirt	String
(2)	{ 2 fields }	Object
asin	5555014090	String
title	Official Top Gear 'My Dad is The Stig' T-Shirt	String
(3)	{ 2 fields }	Object
asin	6041006041	String
title	Sevendayz Men's The Weeknd Xo Long Sleeve Baseball T-Shirt	String
(4)	{ 2 fields }	Object
asin	555501239X	String
title	Top Gear Official Merchandise - Stig Navy	String
(5)	{ 2 fields }	Object
asin	6041616796	String
title	Muay Thai Kick Boxing T-shirt Thai Tattoo Twin Tigers (XL)	String
(6)	{ 2 fields }	Object
asin	9789841655	String
title	Tommy Bahama Embroidered Port Authority Silk Camp Shirt (Color: Black, Size L)	String
(7)	{ 2 fields }	Object
asin	9820303354	String
title	Tommy Bahama Embroidered Legendary Lures Silk Camp Shirt (Color: Oceanaire Blue, Size XL)	String
(8)	{ 2 fields }	Object
asin	B000072SJ2	String
title	Loosegear Short Sleeve	String

Fig. 7

**Q8: Find the highest rating by the reviewer identified as “A2B3TVWTZ7609A” by using map reduce programming.**

```
var map = function() {
    emit(this.gk_reviewerID, this.gk_overall)}
var reduce = function(key, values) {
    var maxRating = values[0]
    values.forEach(function(value){
        if (maxRating<value){
            maxRating = value
        })
    })
    return maxRating}
db.P2424630_reviews.mapReduce(map, reduce, {
    query: {gk_reviewerID: "A2B3TVWTZ7609A"},
    out: {inline: 1}})
```

The screenshot shows the Robo 3T - 1.4 interface. On the left, the 'New Connection (6)' pane shows a tree view with 'System', 'config', 'csj', 'Collections (4)', 'Functions (0)', 'Users (0)', 'pb', and 'vod2'. The 'Collections (4)' section is expanded, showing 'P2424630\_products', 'P2424630\_products\_CP', 'P2424630\_reviews', and 'P2424630\_reviews\_CP'. The 'P2424630\_reviews' collection is selected.

The main pane shows a mapReduce query being executed. The query is:

```
* var map = function() {
    emit(this.gk_reviewerID, this.gk_overall)}
var reduce = function(key, values) {
    var maxRating = values[0]
    values.forEach(function(value){
        if (maxRating<value){
            maxRating = value
        })
    })
    return maxRating}
db.P2424630_reviews.mapReduce(map, reduce, {
    query: {gk_reviewerID: "A2B3TVWTZ7609A"},
    out: {inline: 1}})
```

The execution time is 0.024 sec.

The results are displayed in a table with three columns: Key, Value, and Type.

Key	Value	Type
(1)	{ 5 fields }	Object
results	[ 1 element ]	Array
[0]	{ 2 fields }	Object
_id	A2B3TVWTZ7609A	String
value	4.0	Double
ok	1.0	Double
_o	{ 2 fields }	Object
_keys	[ 2 elements ]	Array
_db	{ 3 fields }	Object

Fig. 8

**Q9: Using both collections, find the title and description of products reviewed by the reviewer identified by “A2C65Q9IDNCSR”. No other product details are required.**

```
var ASINcursor = db.P2424630_reviews.aggregate([
  {$match: {gk_reviewerID: "A2C65Q9IDNCSR"}},
  {$project: {gk_asin: "$gk_asin"}}])
while (ASINcursor.hasNext()){
  var nextDoc = ASINcursor.next()
  var nextAsin = nextDoc.gk_asin
  var prodCursor = (db.P2424630_products.aggregate([
    {$match: {gk_asin: nextAsin}},
    {$project: {_id: 0, title: "$gk_title",
      description: "$gk_description"}}
  ]))
  while (prodCursor.hasNext()){
    print(prodCursor.next())}}}
```

Robo 3T - 1.4

File View Options Window Help

New Connection (6)

- System
- config
- csj
  - Collections (4)
    - P2424630\_products
    - P2424630\_products\_CP
    - P2424630\_reviews
    - P2424630\_reviews\_CP
  - Functions (0)
  - Users (0)
- pb
- vod2

\* var ASINcursor = db.P2424630\_reviews.aggregate([

New Connection localhost:27017 csj

```
var ASINcursor = db.P2424630_reviews.aggregate([
  {$match: {gk_reviewerID: "A2C65Q9IDNCSR"}},
  {$project: {gk_asin: "$gk_asin"}}])
while (ASINcursor.hasNext()){
  var nextDoc = ASINcursor.next()
  var nextAsin = nextDoc.gk_asin
  var prodCursor = (db.P2424630_products.aggregate([
    {$match: {gk_asin: nextAsin}},
    {$project: {_id: 0, title: "$gk_title",
      description: "$gk_description"}}
  ]))
  while (prodCursor.hasNext()){
    print(prodCursor.next())}}}
```

0.001 sec.

Key	Value	Type
✓ (1)	{ 2 fields }	Object
title	Buzz Lightyear Deluxe - Size: Child S(4-6)	String
description	Kids can get ready to blast off "to infinity and beyond"--or at lea...	String
✓ (2)	{ 2 fields }	Object
title	Elope Kid's Magician Hat With Rabbit, Black/White, One Size	String
description	Presto, change-o! Children can transform themselves into magic...	String

Fig. 9

**E1: Implement one index that would improve the querying of the database based on one or more of the queries (Q1-Q9). Identify the chosen query or queries and explain and justify your choice of index. Present, explain and compare execution plans to support your choice and summarise your findings.**

```
db.P2424630_reviews.createIndex({gk_unixReviewTime: 1})
```

I chose to use unixReviewTime as an index. Using this index I am expecting a significant reduction in time required for my question 4 query. As shown in the figure 4, the best time that I managed to achieve was 0.007 sec. While after creating the index, as shown in the figure below (fig. 10) the best time was reduced to 0.001 sec. Using the `.explain("allPlansExecution")` before the aggregate, it is clearly shown why the speed up is happening, as shown below in figures 11, before, and 12, after, the index was applied, totalDocsExamined were reduced from 7331 to 21, which are the returned documents.

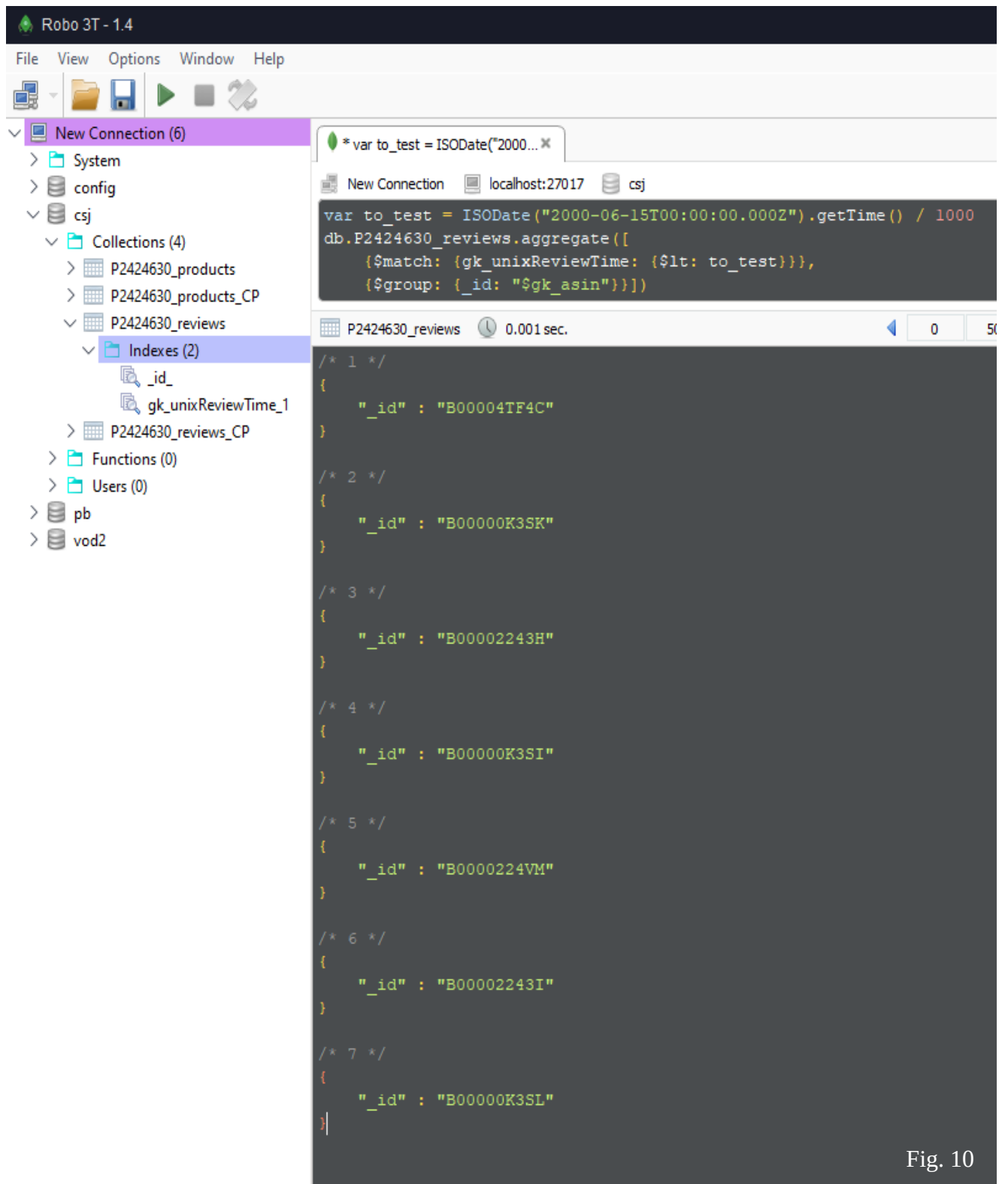


Fig. 10

Robo 3T - 1.4

File View Options Window Help

New Connection (6)

- System
- config
- csj
  - Collections (4)
    - P2424630\_products
    - P2424630\_products\_CP
    - P2424630\_reviews
      - Indexes (1)
        - \_id\_**
      - P2424630\_reviews\_CP
        - Indexes
    - Functions (0)
    - Users (0)
  - pb
  - vod2

\* var to\_test = ISODate("2000...X

New Connection localhost:27017 csj

```
var to_test = ISODate("2000-06-15T00:00:00.000Z").getTime() / 1000
db.P2424630_reviews.explain("allPlansExecution").aggregate([
  {$match: {gk_unixReviewTime: {$lt: to_test}}},
  {$group: {_id: "$gk_asin"}}])
```

0.007 sec.

```
{
  "executionStats" : {
    "executionSuccess" : true,
    "nReturned" : 21,
    "executionTimeMillis" : 5,
    "totalKeysExamined" : 0,
    "totalDocsExamined" : 7331,
    "executionStages" : {
      "stage" : "PROJECTION_SIMPLE",
      "nReturned" : 21,
      "executionTimeMillisEstimate" : 0,
      "works" : 7333,
      "advanced" : 21,
      "needTime" : 7311,
      "needYield" : 0,
      "saveState" : 8,
      "restoreState" : 8,
      "isEOF" : 1,
      "transformBy" : {
        "gk_asin" : 1,
        "_id" : 0
      },
      "inputStage" : {
        "stage" : "COLLSCAN",
        "filter" : {
          "gk_unixReviewTime" : {
            "$lt" : 961027200.0
          }
        }
      },
      "nReturned" : 21,
      "executionTimeMillisEstimate" : 0,
      "works" : 7333,
      "advanced" : 21,
      "needTime" : 7311,
      "needYield" : 0,
      "saveState" : 8,
      "restoreState" : 8,
      "isEOF" : 1,
      "direction" : "forward",
      "docsExamined" : 7331
    }
  },
  "allPlansExecution" : []
}
```

Fig. 11

The screenshot displays the Robo 3T - 1.4 application window. On the left, a sidebar shows a database structure with collections and indexes. The main area on the right shows a MongoDB query being executed, followed by its execution statistics.

**Database Structure (Left Sidebar):**

- New Connection (6)
  - System
  - config
  - csj
    - Collections (4)
      - P2424630\_products
      - P2424630\_products\_CP
      - P2424630\_reviews
        - Indexes (2)
          - \_id\_
          - gk\_unixReviewTime\_1
        - P2424630\_reviews\_CP
          - Indexes
      - Functions (0)
      - Users (0)
      - pb
      - vod2

**Query Execution (Main Area):**

Query: `* var to_test = ISODate("2000-06-15T00:00:00.000Z") / 1000`

Connection: localhost:27017 csj

Query: `var to_test = ISODate("2000-06-15T00:00:00.000Z").getTime() / 1000`  
`db.P2424630_reviews.explain("allPlansExecution").aggregate([`  
`{ $match: { gk_unixReviewTime: { $lt: to_test } } },`  
`{ $group: { _id: "$gk_asin" } } ])`

Execution Time: 0.001 sec.

**Execution Statistics (Right Panel):**

```

    },
    "executionStats" : {
      "executionSuccess" : true,
      "nReturned" : 21,
      "executionTimeMillis" : 0,
      "totalKeysExamined" : 21,
      "totalDocsExamined" : 21,
      "executionStages" : {
        "stage" : "PROJECTION_SIMPLE",
        "nReturned" : 21,
        "executionTimeMillisEstimate" : 0,
        "works" : 22,
        "advanced" : 21,
        "needTime" : 0,
        "needYield" : 0,
        "saveState" : 1,
        "restoreState" : 1,
        "isEOF" : 1,
        "transformBy" : {
          "gk_asin" : 1,
          "_id" : 0
        },
      },
      "inputStage" : {
        "stage" : "FETCH",
        "nReturned" : 21,
        "executionTimeMillisEstimate" : 0,
        "works" : 22,
        "advanced" : 21,
        "needTime" : 0,
        "needYield" : 0,
        "saveState" : 1,
        "restoreState" : 1,
        "isEOF" : 1,
        "docsExamined" : 21,
        "alreadyHasObj" : 0,
        "inputStage" : {
          "stage" : "IXSCAN",
          "nReturned" : 21,
          "executionTimeMillisEstimate" : 0,
          "works" : 22,
          "advanced" : 21,
          "needTime" : 0,
          "needYield" : 0,
          "saveState" : 1,

```

Fig. 12

**D1: Write code in MongoDB to automatically embed the details of reviews from the reviews collection with their corresponding product in the product collection. Therefore, the products collection will contain both the product data and review data in a single collection of products.**

```
var RVcursor = db.P2424630_reviews_CP.find({}, {_id:0})
while (RVcursor.hasNext()){
    var nextRVDoc = RVcursor.next()
    var rv_gk_asin = nextRVDoc.gk_asin
    var rv_gk_reviewerID = nextRVDoc.gk_reviewerID
    var rv_gk_reviewerName = nextRVDoc.gk_reviewerName
    var rv_gk_helpful = nextRVDoc.gk_helpful
    var rv_gk_reviewText = nextRVDoc.gk_reviewText
    var rv_gk_overall = nextRVDoc.gk_overall
    var rv_gk_summary = nextRVDoc.gk_summary
    var rv_gk_unixReviewTime = nextRVDoc.gk_unixReviewTime
    var rv_gk_reviewTime = nextRVDoc.gk_reviewTime
    db.P2424630_products_CP.update({gk_asin: rv_gk_asin}, {$push:
        {gk_reviewerID: rv_gk_reviewerID,
         gk_reviewerName: rv_gk_reviewerName,
         gk_helpful: rv_gk_helpful,
         gk_reviewText: rv_gk_reviewText,
         gk_overall: rv_gk_overall,
         gk_summary: rv_gk_summary,
         gk_unixReviewTime: rv_gk_unixReviewTime,
         gk_reviewTime: rv_gk_reviewTime}}})
    db.P2424630_products_CP.find({gk_asin: "B000005JHK9"},
        {_id:0, gk_title: 1, gk_reviewerID: 1, gk_reviewerName: 1,
         gk_helpful: 1, gk_reviewText: 1, gk_overall: 1,
         gk_summary: 1, gk_unixReviewTime: 1, gk_reviewTime: 1})
}
```



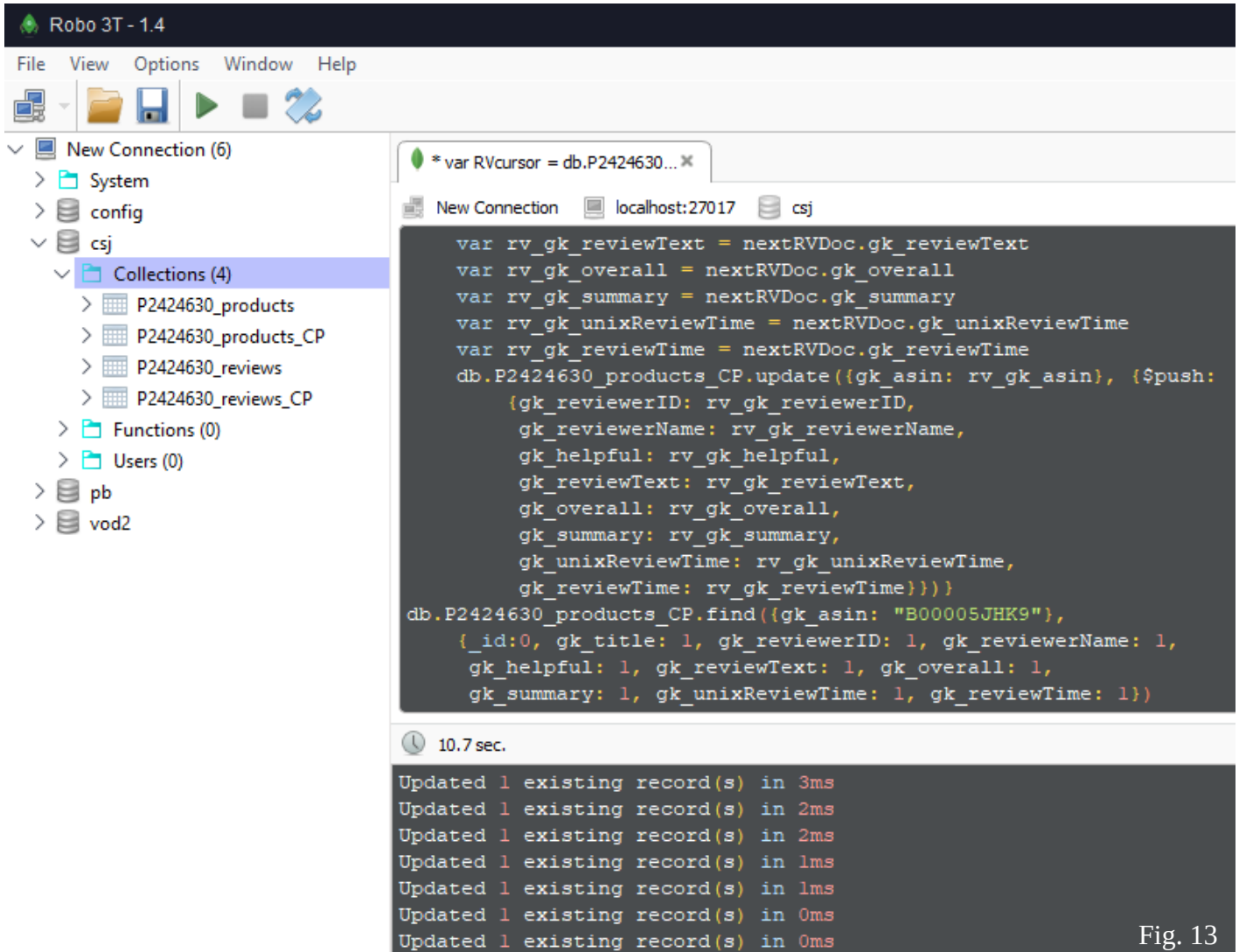


Fig. 13

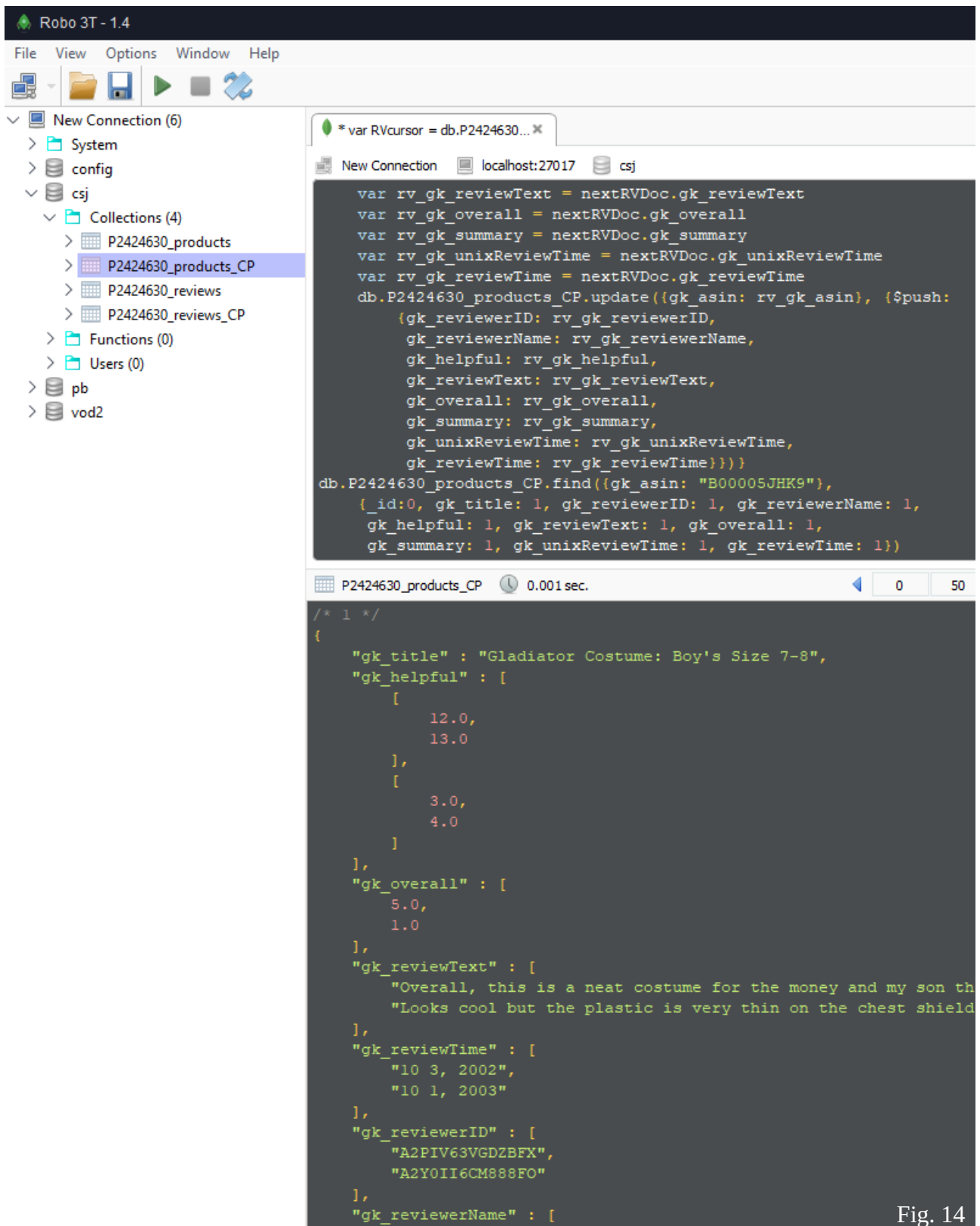


Fig. 14