POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Third EICTA Project

Author(s): **Bamhaoud Younes - 10767795**
Calcara Antonio - 11095781
Foini Lorenzo - 10828129
Martinelli Francesco - 10767793

Group Number: **7**

Academic Year: 2024-2025

Contents

1 | Introduction

The third Enterprise ICT Architectures project is focused on the definition of a MongoDB database based on entities and relationships defined during the first Enterprise ICT Architectures project.
For this request, the used tool is MongoDB Compass for the definition of the MongoDB database and for subsequent querying of it.

2 | MongoDB database introduction

In this chapter, we report the entities and relationships selected for the creation of the MongoDB database and documents.

2.1. Chosen entities from the ER model

As required by the project delivery, five entities of the ER model were selected, subsequently proceeding with the creation of documents within the MongoDB database. The previously defined ER model is shown in Figure 2.1.

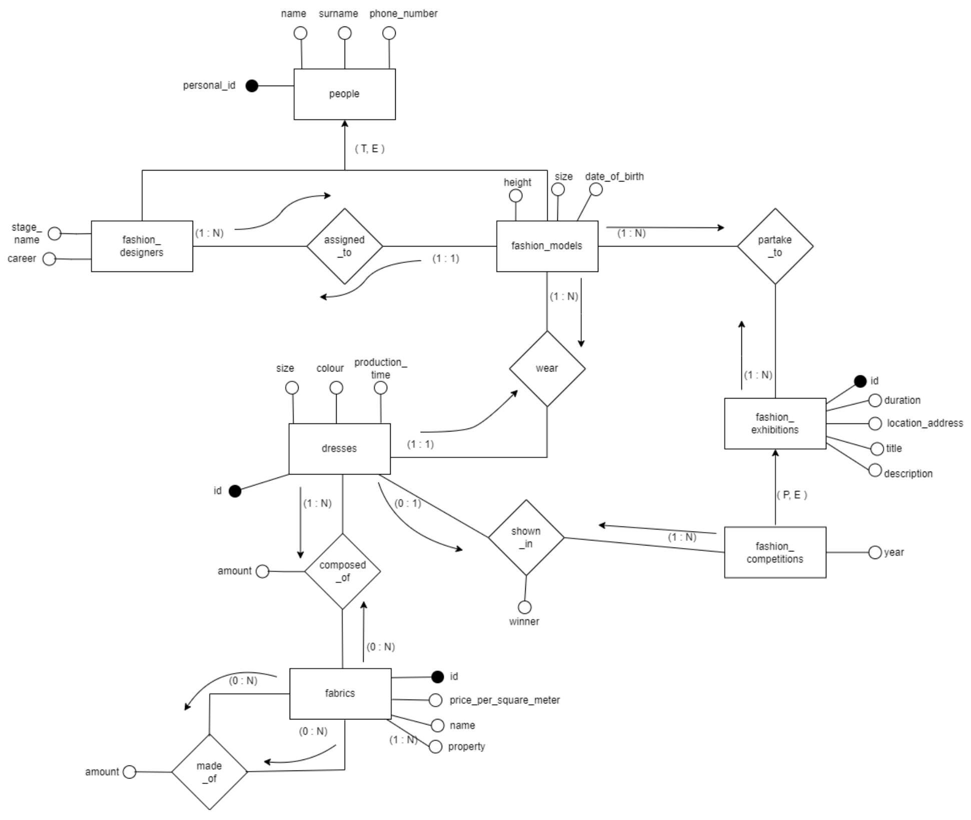
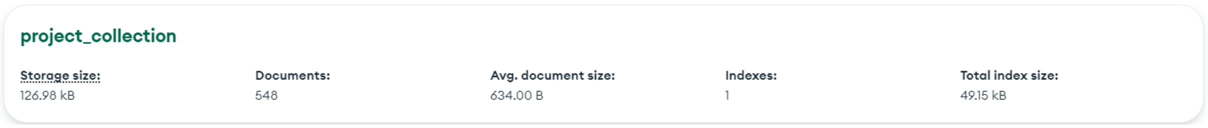


Figure 2.1: ER model from the first project.

Among the entities present, we have selected the following: *fashion_designers, fashion_models, dresses, fabrics, fashion_exhibitions*.
Then the following relationships have been considered: *assigned_to, wear, composed_of, partake_to*.
In the next chapter we specify which entities and relationships constitute the documents of the defined collection and which have been inserted as sub-documents.

2.2. MongoDB database's structure overview

To collect all the documents, a single collection was created, called *project_collection*, whose data can be seen in Figure 2.2.


project_collection

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
126.98 kB	548	634.00 B	1	49.15 kB

Figure 2.2: Collection defined within the MongoDB database.

This collection contains all the "different types" (documents that have very different structures) of documents created, more information about these documents are given in chapter 3.

3 | Structure of the documents

This chapter introduces the structure of the documents inserted in the MongoDB collection.

3.1. *fashion_designers* documents

Each *fashion_designer* document stores the data of a designer, also adding an array containing the sub-documents with the data of the models assigned to him. In addition, the model document itself contains an array of sub-documents with the data of the dresses worn by such model.

The structure of this type of document and an example are shown in Figure 3.1 and Figure 3.2, respectively.

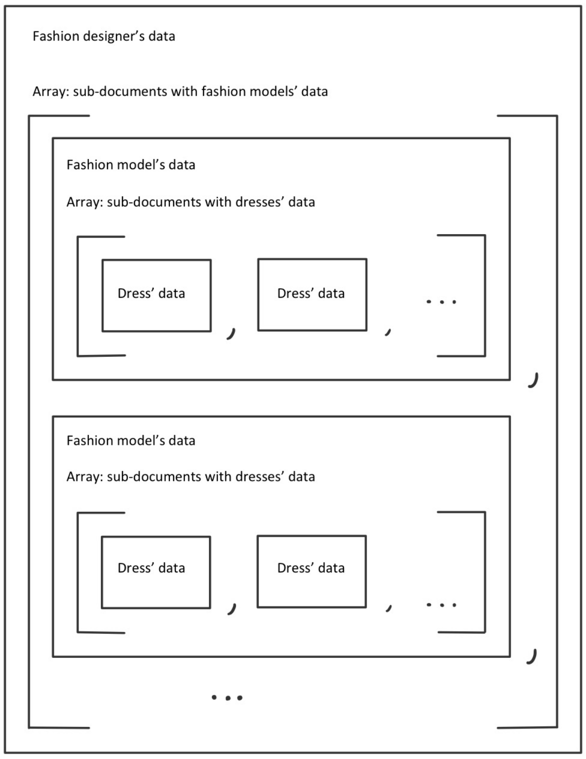
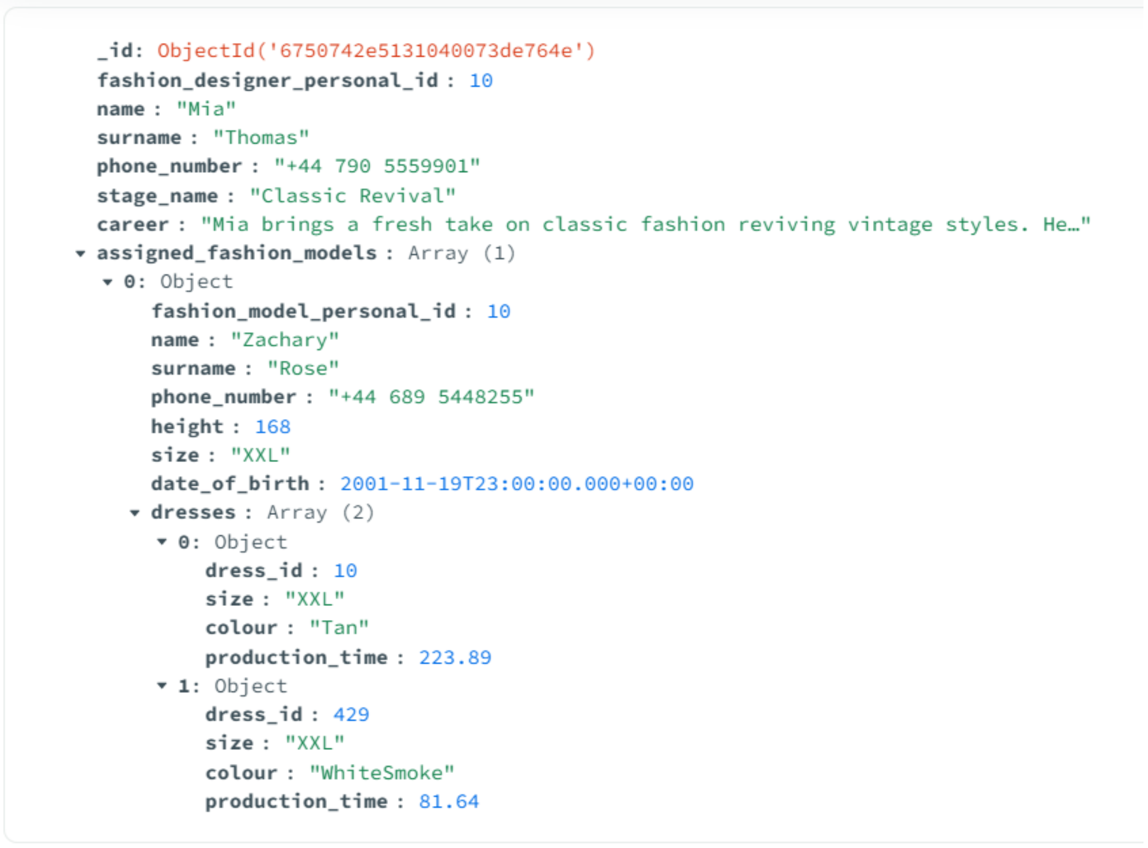

Fashion designer's data
Array: sub-documents with fashion models' data
Fashion model's data
Array: sub-documents with dresses' data
Dress' data
Dress' data
Fashion model's data
Array: sub-documents with dresses' data
Dress' data
Dress' data

Figure 3.1: *fashion_designer* document structure.



```
_id: ObjectId('6750742e5131040073de764e')
fashion_designer_personal_id : 10
name : "Mia"
surname : "Thomas"
phone_number : "+44 790 5559901"
stage_name : "Classic Revival"
career : "Mia brings a fresh take on classic fashion reviving vintage styles. He…"
▾ assigned_fashion_models : Array (1)
  ▾ 0: Object
      fashion_model_personal_id : 10
      name : "Zachary"
      surname : "Rose"
      phone_number : "+44 689 5448255"
      height : 168
      size : "XXL"
      date_of_birth : 2001-11-19T23:00:00.000+00:00
    ▾ dresses : Array (2)
      ▾ 0: Object
          dress_id : 10
          size : "XXL"
          colour : "Tan"
          production_time : 223.89
      ▾ 1: Object
          dress_id : 429
          size : "XXL"
          colour : "WhiteSmoke"
          production_time : 81.64
```

Figure 3.2: *fashion_ designer* document example.

The necessary information to better understand all the fields present in a *fashion_ designer* document are now reported.

Document's fields:

- ***_ id*:** automatically generated unique identifier of the document (generated by MongoDB).
- ***fashion_ designer_ personal_ id***: unique integer identifier of the designer (generated by us).
- ***name***: string containing the designer's name.
- ***surname***: string containing the designer's surname.
- ***phone_ number***: string containing the designer's phone number.
- ***stage_ name***: string containing the designer's stage name.
- ***career***: text description of the designer's career.

- ***assigned_fashion_models***: array of sub-documents containing the data of the models assigned to the designer. Each sub-document has the following fields:
  - ***fashion_model_personal_id***: unique integer identifier of the model (generated by us).
  - ***name***: string containing the model's name.
  - ***surname***: string containing the model's surname.
  - ***phone_number***: string containing the model's phone number.
  - ***height***: integer value in centimeters representing the model's height.
  - ***size***: string containing the model's size.
  - ***date_of_birth***: model's date of birth.
  - ***dresses***: array of sub-documents containing the data of the dresses worn by the model. Each sub-document has the following fields:
    - ***dress_id***: unique integer identifier of the dress (generated by us).
    - ***size***: string containing the size of the dress.
    - ***colour***: string containing the colour of the dress.
    - ***production_time***: decimal value of the time required to product the dress, expressed in minutes.

3.2. *dresses_with_fabrics* documents

Each *dress_with_fabrics* document stores the data of the dress and the fabrics used in it, also including their properties. The fabrics are inserted as an array of sub-documents and the properties of a fabric are inserted in the sub-document as an array of properties.

The structure of this type of document and an example are shown in Figure 3.3 and Figure 3.4, respectively.

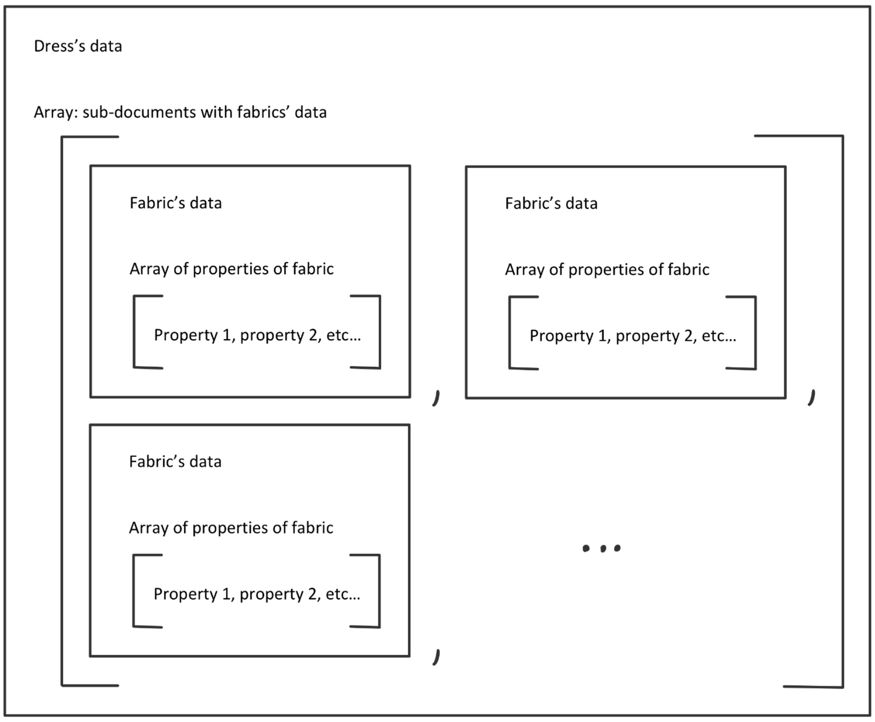
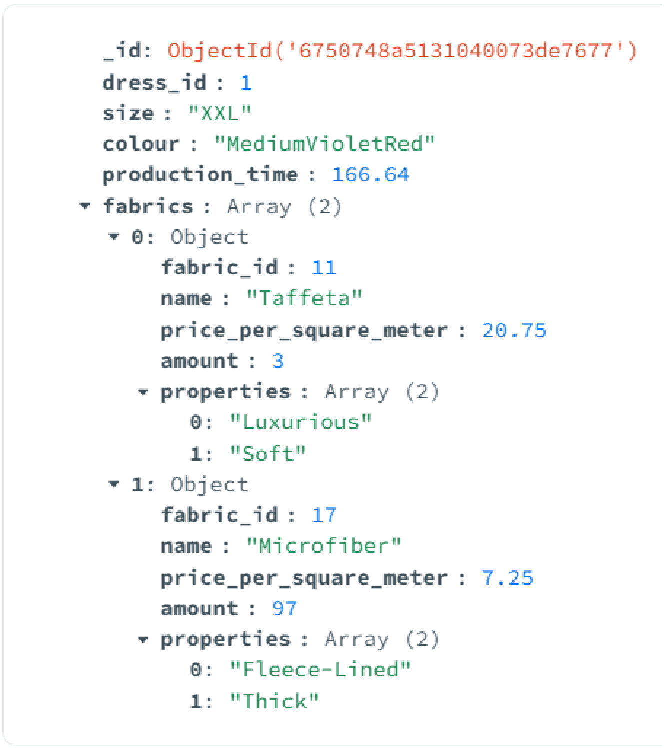

Dress's data
Array: sub-documents with fabrics' data
Fabric's data
Array of properties of fabric
Property 1, property 2, etc...
Fabric's data
Array of properties of fabric
Property 1, property 2, etc...
Fabric's data
Array of properties of fabric
Property 1, property 2, etc...
...

Figure 3.3: *dress_with_fabrics* document structure.



```
_id: ObjectId('6750748a5131040073de7677')
dress_id : 1
size : "XXL"
colour : "MediumVioletRed"
production_time : 166.64
▾ fabrics : Array (2)
  ▾ 0: Object
      fabric_id : 11
      name : "Taffeta"
      price_per_square_meter : 20.75
      amount : 3
    ▾ properties : Array (2)
        0: "Luxurious"
        1: "Soft"
  ▾ 1: Object
      fabric_id : 17
      name : "Microfiber"
      price_per_square_meter : 7.25
      amount : 97
    ▾ properties : Array (2)
        0: "Fleece-Lined"
        1: "Thick"
```

Figure 3.4: *dress_with_fabrics* document example.

The information necessary to better understand all the fields present in a *dress_with_fabrics* document are now reported.

Document's fields:

- ***_id:*** automatically generated unique identifier of the document (generated by MongoDB).
- ***dress_id***: unique integer identifier of the dress (generated by us).
- ***size***: string containing the size of the dress.
- ***colour***: string containing the colour of the dress.
- ***production_time***: decimal value of the time required to product the dress, expressed in minutes.
- ***fabrics***: array of sub-documents containing the data of the fabrics used in the dress. Each sub-document has the following fields:
  - ***fabric_id***: unique integer identifier of the fabric (generated by us).
  - ***name***: string containing the name of the fabric.
  - ***price_per_square_meter***: decimal value of the price per square meter of the fabric, with the format "euro.cents" and without the currency.
  - ***amount***: integer representing the percentage of fabric present in the dress.
  - ***properties***: array containing the properties of the fabric.

3.3. *fashion_exhibitions* documents

Each *fashion_exhibition* document stores the data of a fashion exhibition and the fashion models that partook in it.

The structure of this type of document and an example are shown in Figure 3.7 and Figure 3.8, respectively.

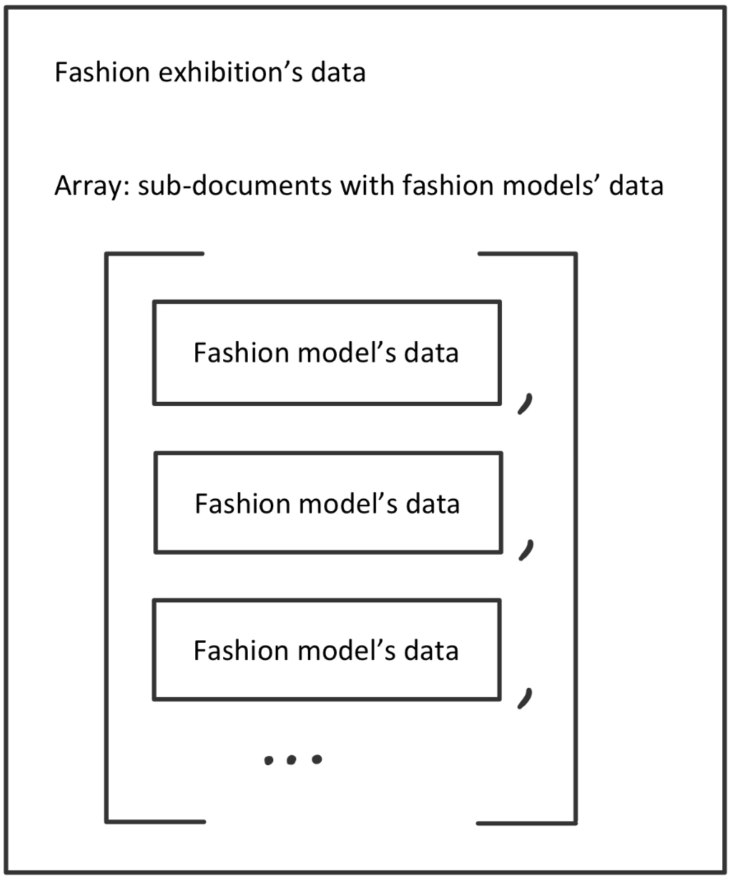
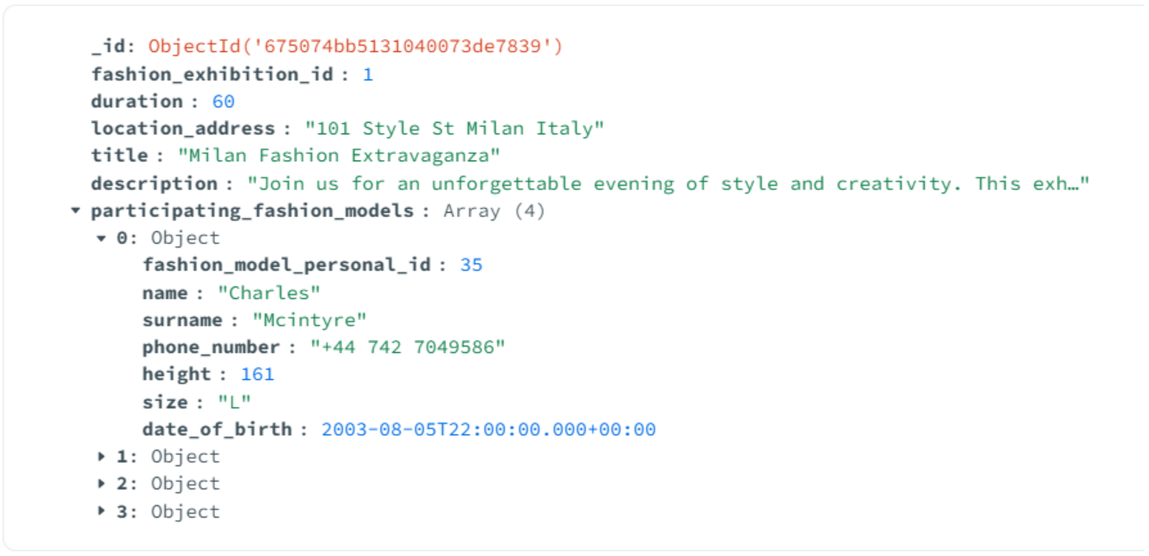

Fashion exhibition's data
Array: sub-documents with fashion models' data
Fashion model's data ,
Fashion model's data ,
Fashion model's data ,
...

Figure 3.5: *fashion_ exhibition* document structure.



```
  _id: ObjectId('675074bb5131040073de7839')
  fashion_exhibition_id : 1
  duration : 60
  location_address : "101 Style St Milan Italy"
  title : "Milan Fashion Extravaganza"
  description : "Join us for an unforgettable evening of style and creativity. This exh…"
▾ participating_fashion_models : Array (4)
  ▾ 0: Object
      fashion_model_personal_id : 35
      name : "Charles"
      surname : "Mcintyre"
      phone_number : "+44 742 7049586"
      height : 161
      size : "L"
      date_of_birth : 2003-08-05T22:00:00.000+00:00
  ▸ 1: Object
  ▸ 2: Object
  ▸ 3: Object
```

Figure 3.6: *fashion_ exhibition* document example.

The information necessary to better understand all the fields present in a *fashion_ exhibition* document are now reported.

Document's fields:

- ***_ id*:** automatically generated unique identifier of the document (generated by MongoDB).
- ***fashion_ exhibition_ id***: unique integer identifier of the fashion exhibition (generated by us).
- ***duration***: decimal value of the duration of the exhibition, with the format "minutes.seconds".
- ***location_ address***: string containing the address of where the exhibition is located.
- ***title***: string containing the title of the exhibition.
- ***description***: text containing a description of the exhibition.
- ***participating_ fashion_ models***: array of sub-documents containing the data of the models that partook in the exhibition. Each sub-document has the following fields:
  - ***fashion_ model_ personal_ id***: unique integer identifier of the model (generated by us).
  - ***name***: string containing the model's name.
  - ***surname***: string containing the model's surname.
  - ***phone_ number***: string containing the model's phone number.
  - ***height***: integer value in centimeters representing the model's height.
  - ***size***: string containing the model's size.
  - ***date_ of_ birth***: model's date of birth.

4 | Queries

In this chapter we present the queries used to interrogate the MongoDB database, dividing them by the type of request.

4.1. Create queries

Below we report the two queries used to create documents.

CREATE QUERY 1:

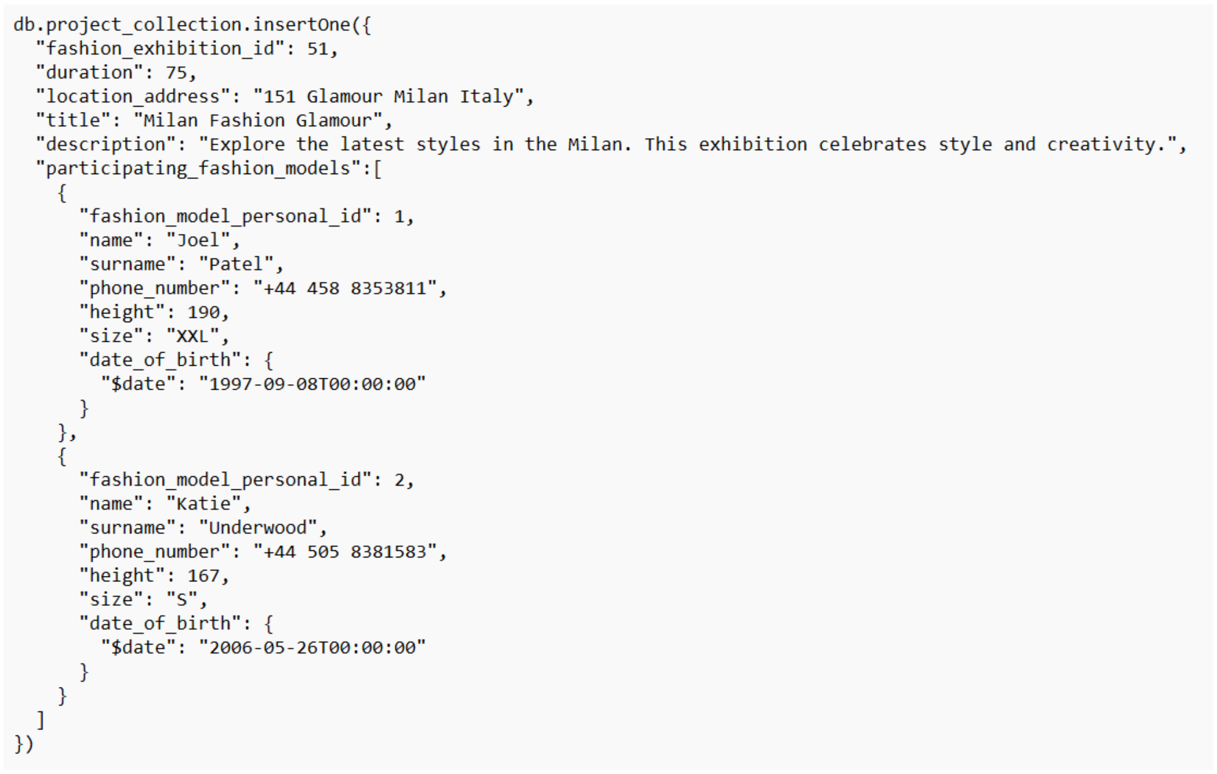
Request:

Create a new document of a fashion exhibition with the following data:

- ***fashion_exhibition_id***: 51
- ***duration***: 75
- ***location_address***: "151 Glamour Milan Italy"
- ***title***: "Milan Fashion Glamour"
- ***description***: "Explore the latest styles in the Milan. This exhibition celebrates style and creativity."

The fashion models with *personal_id* equal to 1 and 2 participated in the exhibition.

MongoDB code:

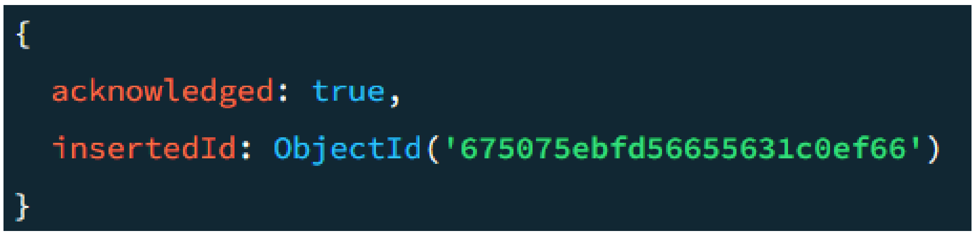


```
db.project_collection.insertOne({
  "fashion_exhibition_id": 51,
  "duration": 75,
  "location_address": "151 Glamour Milan Italy",
  "title": "Milan Fashion Glamour",
  "description": "Explore the latest styles in the Milan. This exhibition celebrates style and creativity.",
  "participating_fashion_models":[
    {
      "fashion_model_personal_id": 1,
      "name": "Joel",
      "surname": "Patel",
      "phone_number": "+44 458 8353811",
      "height": 190,
      "size": "XXL",
      "date_of_birth": {
        "$date": "1997-09-08T00:00:00"
      }
    },
    {
      "fashion_model_personal_id": 2,
      "name": "Katie",
      "surname": "Underwood",
      "phone_number": "+44 505 8381583",
      "height": 167,
      "size": "S",
      "date_of_birth": {
        "$date": "2006-05-26T00:00:00"
      }
    }
  ]
})
```

Figure 4.1: MongoDB code of "CREATE QUERY 1".

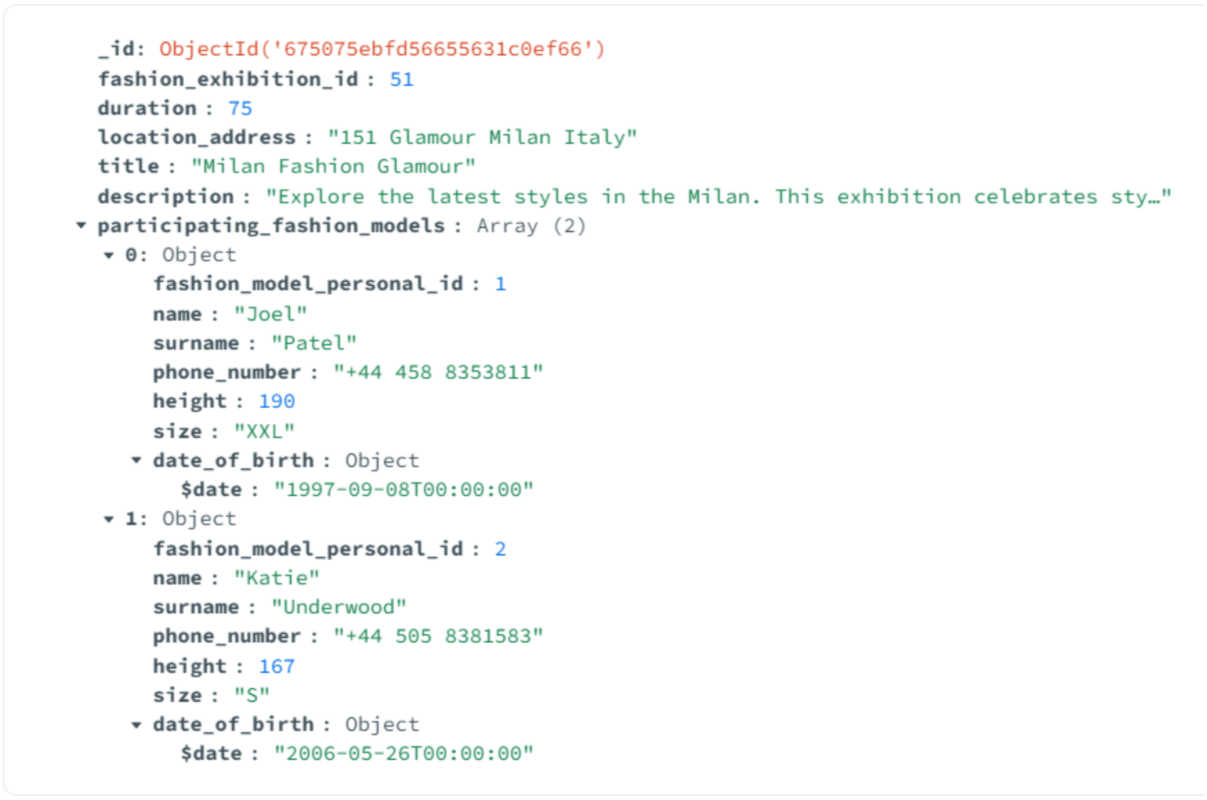
Output:

The new document is correctly added in the collection, as shown in Figure 4.2 and Figure 4.3.



```
{
  acknowledged: true,
  insertedId: ObjectId('675075ebfd56655631c0ef66')
}
```

Figure 4.2: MongoDB Shell acknowledge.



```
_id: ObjectId('675075ebfd56655631c0ef66')
fashion_exhibition_id : 51
duration : 75
location_address : "151 Glamour Milan Italy"
title : "Milan Fashion Glamour"
description : "Explore the latest styles in the Milan. This exhibition celebrates sty…"
▾ participating_fashion_models : Array (2)
  ▾ 0: Object
      fashion_model_personal_id : 1
      name : "Joel"
      surname : "Patel"
      phone_number : "+44 458 8353811"
      height : 190
      size : "XXL"
    ▾ date_of_birth : Object
        $date : "1997-09-08T00:00:00"
  ▾ 1: Object
      fashion_model_personal_id : 2
      name : "Katie"
      surname : "Underwood"
      phone_number : "+44 505 8381583"
      height : 167
      size : "S"
    ▾ date_of_birth : Object
        $date : "2006-05-26T00:00:00"
```

Figure 4.3: "CREATE QUERY 1" output.

CREATE QUERY 2:

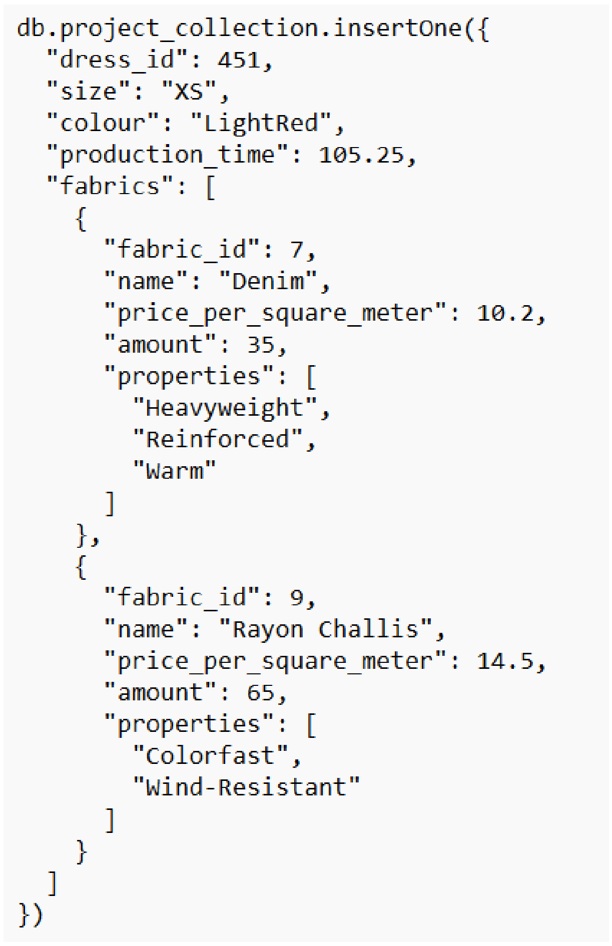
Request:

Create a new document of a dress with the following data:

- ***dress_id***: 151
- ***size***: "XS"
- ***colour***: "LightRed"
- ***production_time***: 105.25

The fabrics used to create the dress have *fabric_id* equal to 7 and 9, they constitute 35% and 65% of the dress, respectively.
This dress is not yet assigned to any model, just insert the document regarding the dress and the fabrics used.

MongoDB code:

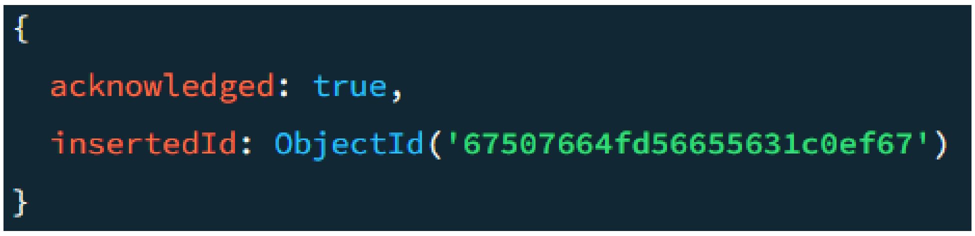


```
db.project_collection.insertOne({
  "dress_id": 451,
  "size": "XS",
  "colour": "LightRed",
  "production_time": 105.25,
  "fabrics": [
    {
      "fabric_id": 7,
      "name": "Denim",
      "price_per_square_meter": 10.2,
      "amount": 35,
      "properties": [
        "Heavyweight",
        "Reinforced",
        "Warm"
      ]
    },
    {
      "fabric_id": 9,
      "name": "Rayon Challis",
      "price_per_square_meter": 14.5,
      "amount": 65,
      "properties": [
        "Colorfast",
        "Wind-Resistant"
      ]
    }
  ]
})
```

Figure 4.4: MongoDB code of "CREATE QUERY 2".

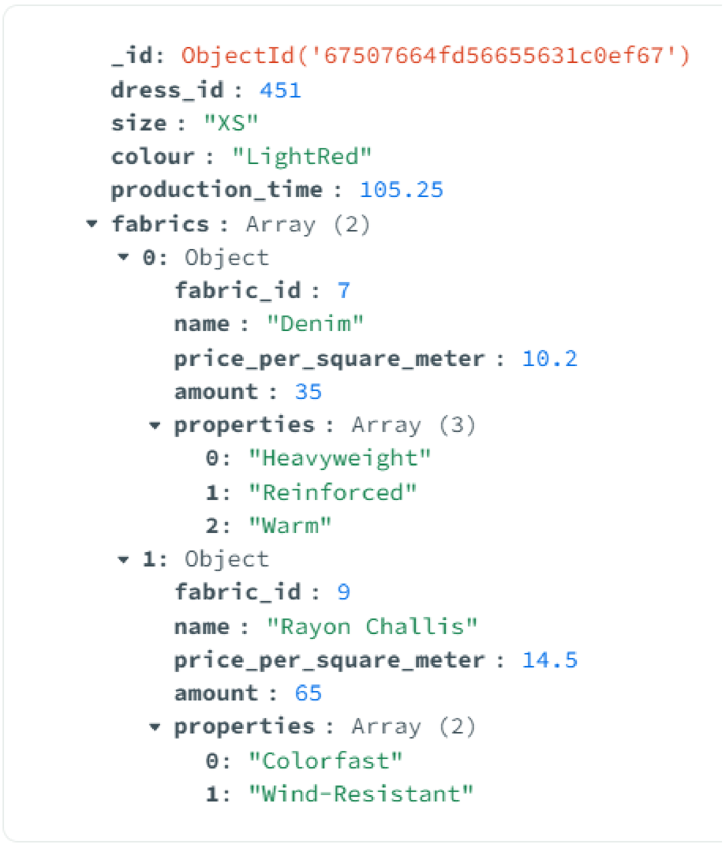
Output:

The new document is correctly added in the collection, as shown in Figure 4.5 and Figure 4.6.



```
{
  acknowledged: true,
  insertedId: ObjectId('67507664fd56655631c0ef67')
}
```

Figure 4.5: MongoDB Shell acknowledge.



```
_id: ObjectId('67507664fd56655631c0ef67')
dress_id : 451
size : "XS"
colour : "LightRed"
production_time : 105.25
▾ fabrics : Array (2)
  ▾ 0: Object
      fabric_id : 7
      name : "Denim"
      price_per_square_meter : 10.2
      amount : 35
    ▾ properties : Array (3)
        0: "Heavyweight"
        1: "Reinforced"
        2: "Warm"
  ▾ 1: Object
      fabric_id : 9
      name : "Rayon Challis"
      price_per_square_meter : 14.5
      amount : 65
    ▾ properties : Array (2)
        0: "Colorfast"
        1: "Wind-Resistant"
```

Figure 4.6: "CREATE QUERY 2" output.

4.2. Update queries

Below we report the two queries used to update documents' fields.

UPDATE QUERY 1:

Request:

> Update the *location_ address* of the fashion exhibition whose *fashion_ exhibition_ id* equal to 22. The new *location_ address* is "174 Fashion St London UK".

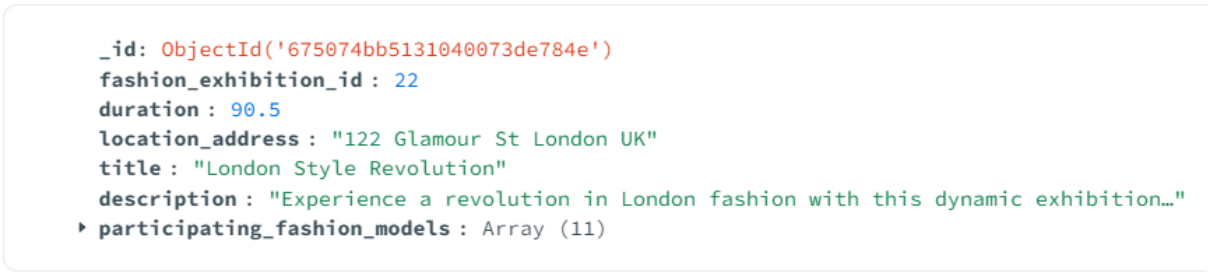
MongoDB code:

```
db.project_collection.updateOne(
  {"fashion_exhibition_id": 22},
  {"$set": {"location_address": "174 Fashion St London UK"}}
)
```

Figure 4.7: MongoDB code of "UPDATE QUERY 1".

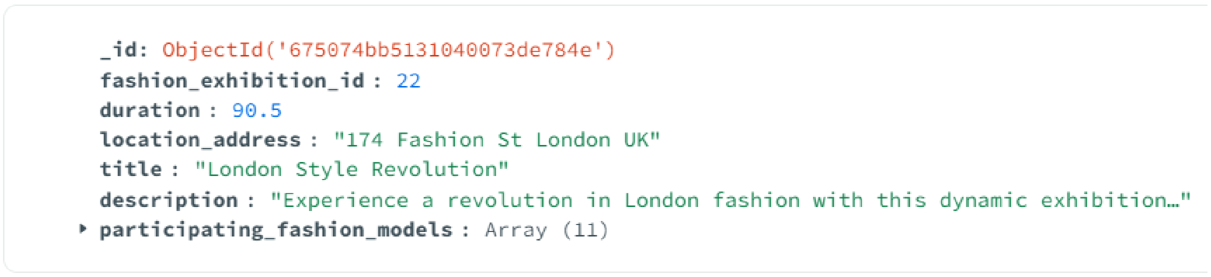
Output:

The two figures below show the *location_ address* of the document of the exhibition before executing the query (Figure 4.8, old *location_ address*: "122 Glamour St London UK") and after executing the query (Figure 4.9, new *location_ address*: "174 Fashion St London UK").



```
_id: ObjectId('675074bb5131040073de784e')
fashion_exhibition_id : 22
duration : 90.5
location_address : "122 Glamour St London UK"
title : "London Style Revolution"
description : "Experience a revolution in London fashion with this dynamic exhibition…"
▸ participating_fashion_models : Array (11)
```

Figure 4.8: Before the query.



```
_id: ObjectId('675074bb5131040073de784e')
fashion_exhibition_id : 22
duration : 90.5
location_address : "174 Fashion St London UK"
title : "London Style Revolution"
description : "Experience a revolution in London fashion with this dynamic exhibition…"
▸ participating_fashion_models : Array (11)
```

Figure 4.9: After the query.

UPDATE QUERY 2:

Request:

Update the *phone_ number* of the designer with *fashion_ designer_ personal_ id* equal to 5. The new *phone_ number* is +1 219 7486432.

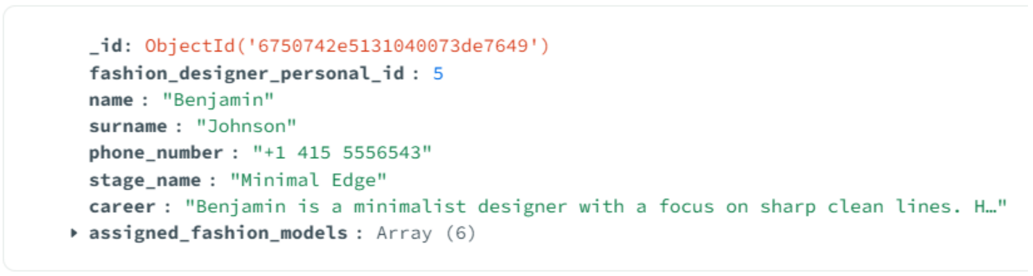
MongoDB code:

```
db.project_collection.updateOne(
  {"fashion_designer_personal_id": 5},
  {"$set": {"phone_number": "+1 219 7486432"}}
)
```

Figure 4.10: MongoDB code of "UPDATE QUERY 2".

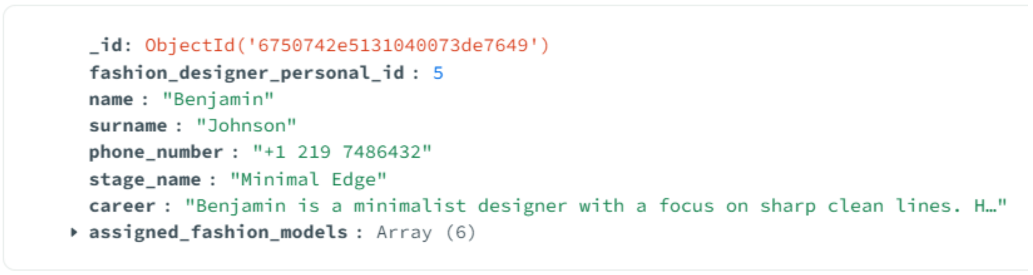
Output:

The two figures below show the *phone_ number* of the document of the designer before executing the query (Figure 4.11, old *phone_ number*: +1 415 5556543) and after executing the query (Figure 4.12, new *phone_ number*: +1 219 7486432).



```
_id: ObjectId('6750742e5131040073de7649')
fashion_designer_personal_id : 5
name : "Benjamin"
surname : "Johnson"
phone_number : "+1 415 5556543"
stage_name : "Minimal Edge"
career : "Benjamin is a minimalist designer with a focus on sharp clean lines. H…"
▸ assigned_fashion_models : Array (6)
```

Figure 4.11: Before the query.



```
_id: ObjectId('6750742e5131040073de7649')
fashion_designer_personal_id : 5
name : "Benjamin"
surname : "Johnson"
phone_number : "+1 219 7486432"
stage_name : "Minimal Edge"
career : "Benjamin is a minimalist designer with a focus on sharp clean lines. H…"
▸ assigned_fashion_models : Array (6)
```

Figure 4.12: After the query.

4.3. Delete queries

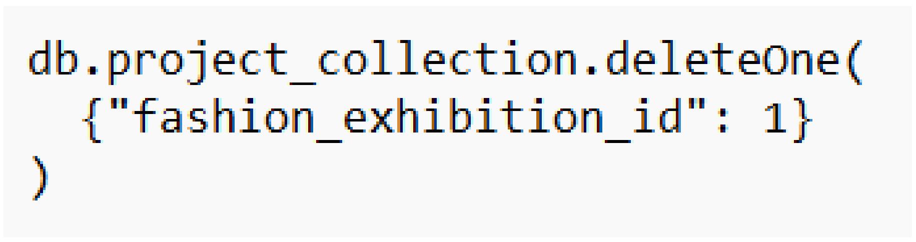
Below we report the two queries used to delete documents.

DELETE QUERY 1:

Request:

> Delete the document of the fashion exhibition with *fashion_exhibition_id* equal to 1.

MongoDB code:

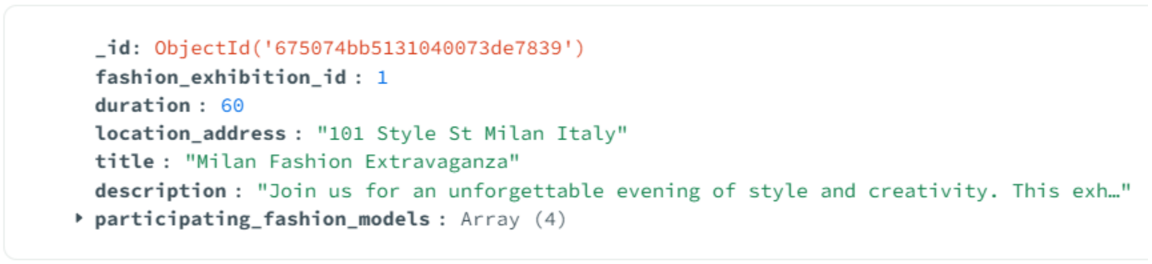


```
db.project_collection.deleteOne(
  {"fashion_exhibition_id": 1}
)
```

Figure 4.13: MongoDB code of "DELETE QUERY 1".

Output:

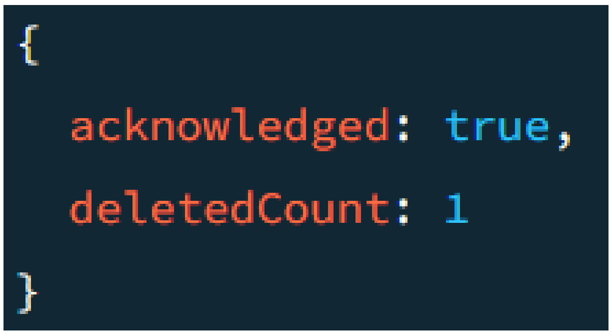
> As we can see in Figure 4.14, the document regarding the exhibition is present in the collection before the execution of the query.



```
  _id: ObjectId('675074bb5131040073de7839')
  fashion_exhibition_id : 1
  duration : 60
  location_address : "101 Style St Milan Italy"
  title : "Milan Fashion Extravaganza"
  description : "Join us for an unforgettable evening of style and creativity. This exh…"
▸ participating_fashion_models : Array (4)
```

Figure 4.14: Before the query.

> After running the query, that document is no longer present in the collection, as can be seen from the MongoDB Shell output shown in Figure 4.15.



```
{
  acknowledged: true,
  deletedCount: 1
}
```

Figure 4.15: MongoDB Shell acknowledge.

DELETE QUERY 2:

Request:

Delete all documents regarding fashion exhibitions located in Toronto.

MongoDB code:

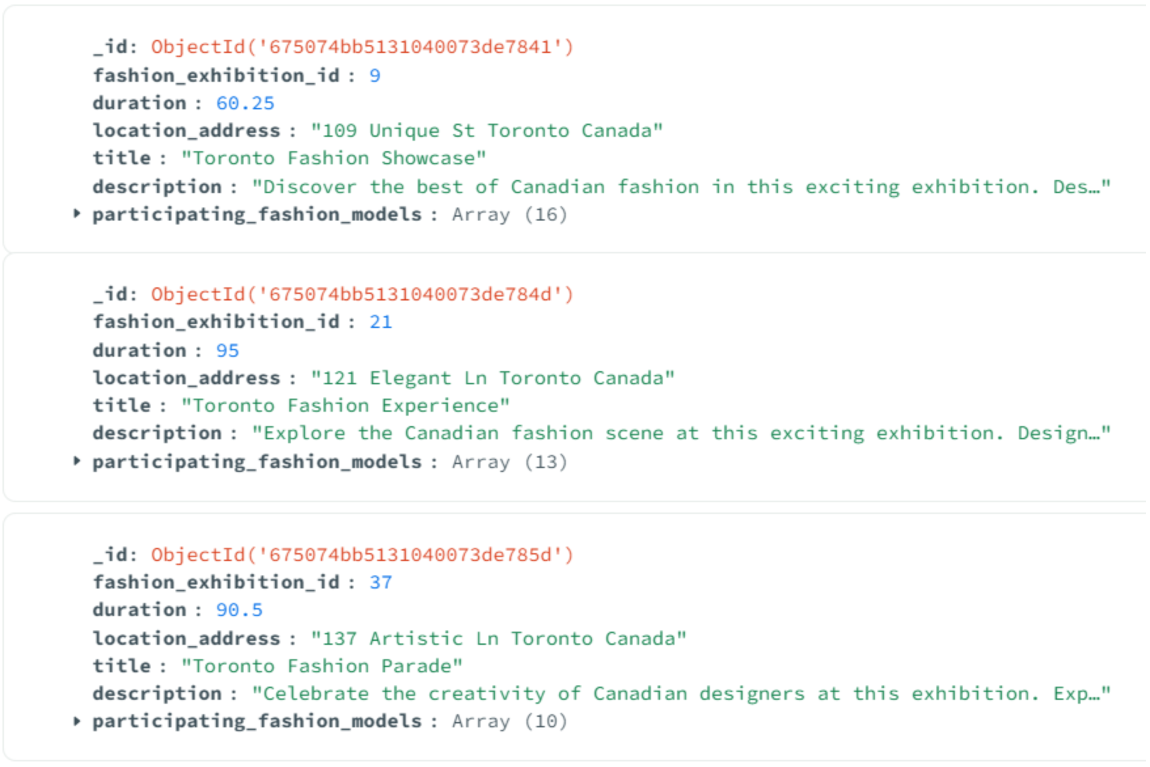
```
db.project_collection.deleteMany({
    "location_address": { "$regex": "Toronto", "$options": "i" }
})
```

Figure 4.16: MongoDB code of "DELETE QUERY 2".

Output:

This query deletes all documents within the specified collection where the *location_address* field contains the word "Toronto". The filter uses a regular expression (*$regex*) to match partial matches and the *$options: "i"* to make the search case-insensitive.

As we can see in Figure 4.17, before executing the query there were three documents that satisfy the delete condition.



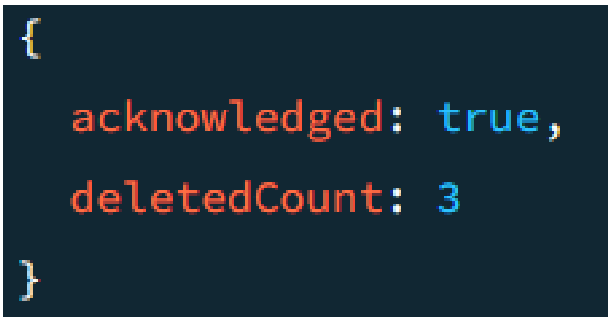
```
  _id: ObjectId('675074bb5131040073de7841')
  fashion_exhibition_id : 9
  duration : 60.25
  location_address : "109 Unique St Toronto Canada"
  title : "Toronto Fashion Showcase"
  description : "Discover the best of Canadian fashion in this exciting exhibition. Des…"
▸ participating_fashion_models : Array (16)

  _id: ObjectId('675074bb5131040073de784d')
  fashion_exhibition_id : 21
  duration : 95
  location_address : "121 Elegant Ln Toronto Canada"
  title : "Toronto Fashion Experience"
  description : "Explore the Canadian fashion scene at this exciting exhibition. Design…"
▸ participating_fashion_models : Array (13)

  _id: ObjectId('675074bb5131040073de785d')
  fashion_exhibition_id : 37
  duration : 90.5
  location_address : "137 Artistic Ln Toronto Canada"
  title : "Toronto Fashion Parade"
  description : "Celebrate the creativity of Canadian designers at this exhibition. Exp…"
▸ participating_fashion_models : Array (10)
```

Figure 4.17: Before the query.

Since the *deleteMany* keyword is used, all three documents are correctly deleted, as can be seen in Figure 4.18.



```
{
  acknowledged: true,
  deletedCount: 3
}
```

Figure 4.18: MongoDB Shell acknowledge.

4.4. Queries with given complexities

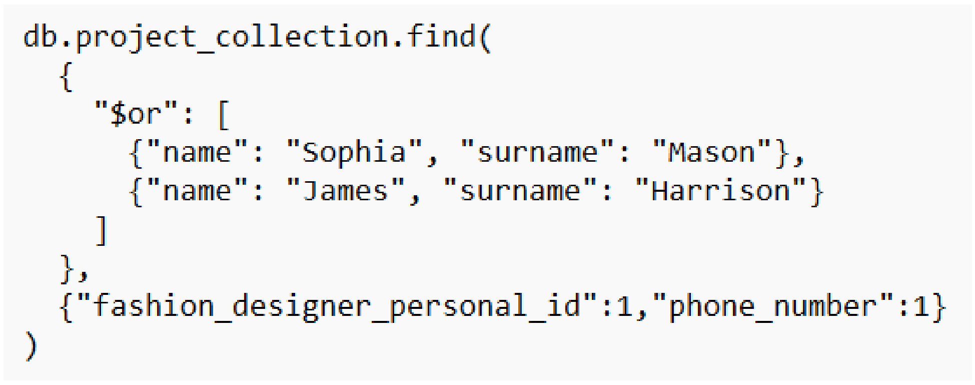
Below we report ten queries, with different types of complexity, used to interrogate the MongoDB database.

QUERY 1: Projections and logical query operators.

Request:

Find the *fashion_designer_personal_id* and *phone_number* of the following fashion designers: Sophie Mason and James Harrison.

MongoDB code:



```
db.project_collection.find(
  {
    "$or": [
      {"name": "Sophia", "surname": "Mason"},
      {"name": "James", "surname": "Harrison"}
    ]
  },
  {"fashion_designer_personal_id":1,"phone_number":1}
)
```

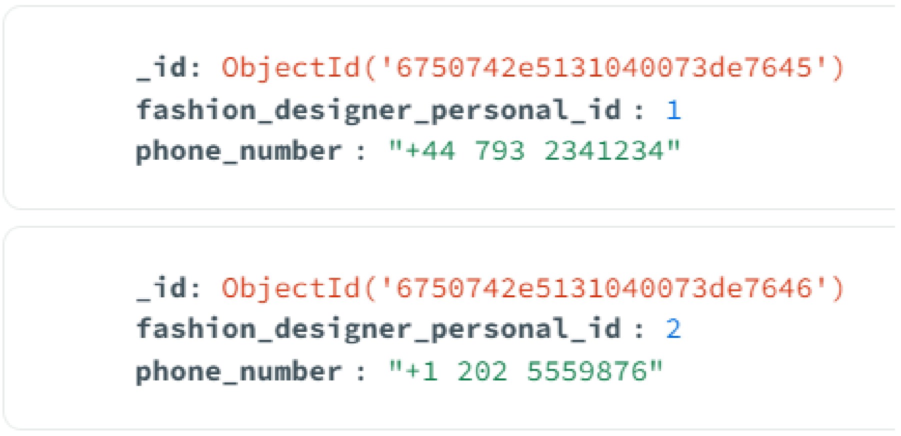
Figure 4.19: MongoDB code of "QUERY 1".

Description:

This query searches the *project_collection* for documents where either the combination of *name* as "Sophia" and *surname* as "Mason" matches, or the combination of *name* as "James" and *surname* as "Harrison" matches.
This query uses the *$or* operator to specify these alternative conditions.
The result of the query will include only the *fashion_designer_personal_id* and *phone_number* fields for the matching documents, as specified in the projection.

Output:

As can be seen in the Figure 4.20, only two documents satisfy the query conditions. In detail, the document with *fashion_designer_personal_id* equal to 1 refers to the designer Sophie Mason, while the document with *fashion_designer_personal_id* equal to 2 refers to the designer James Harrison.
As specified by the projections, only the *fashion_designer_personal_id* and *phone_number* fields are shown.



```
_id: ObjectId('6750742e5131040073de7645')
fashion_designer_personal_id : 1
phone_number : "+44 793 2341234"

_id: ObjectId('6750742e5131040073de7646')
fashion_designer_personal_id : 2
phone_number : "+1 202 5559876"
```

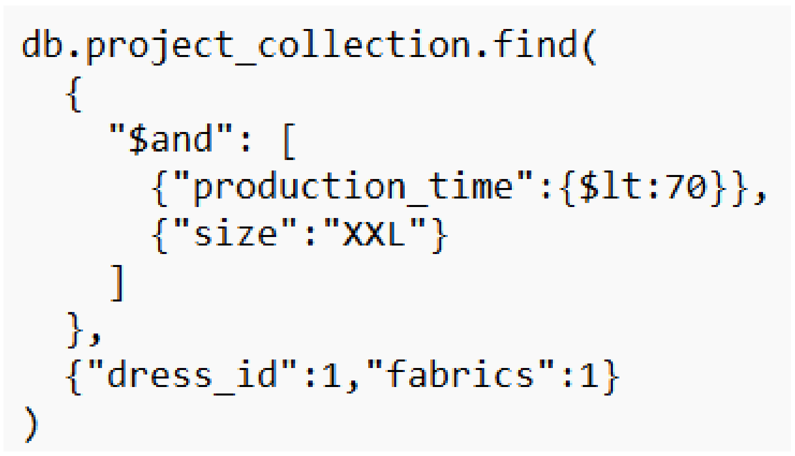
Figure 4.20: "QUERY 1" output.

QUERY 2: Projections and logical query operators.

Request:

> Find the *dress_ id* and *fabrics* of the dresses that have *production_ time* less than 70 and *size* XXL.

MongoDB code:



```
db.project_collection.find(
  {
    "$and": [
      {"production_time":{$lt:70}},
      {"size":"XXL"}
    ]
  },
  {"dress_id":1,"fabrics":1}
)
```

Figure 4.21: MongoDB code of "QUERY 2".

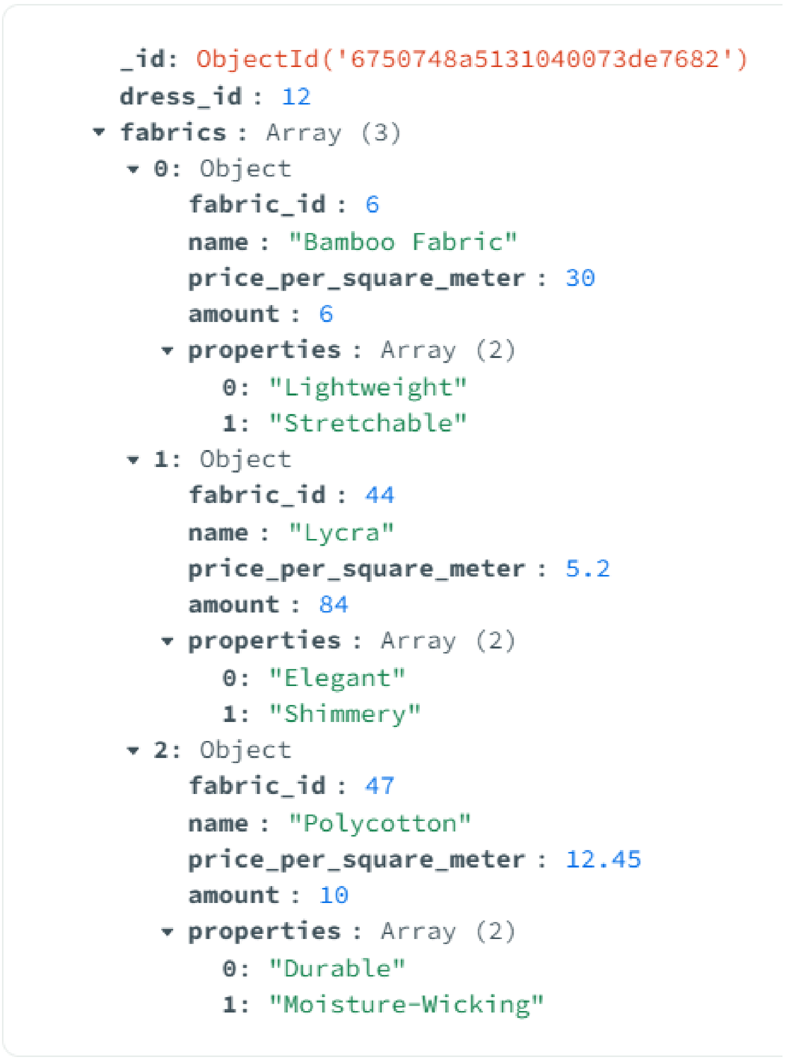
Description:

> This query searches the *project_ collection* for documents where the *production_ time* is less than 70 and the *size* is "XXL", using the *$and* operator to combine these conditions.

The query retrieves only the *dress_ id* and *fabrics* fields from the matching documents, as specified in the projection.

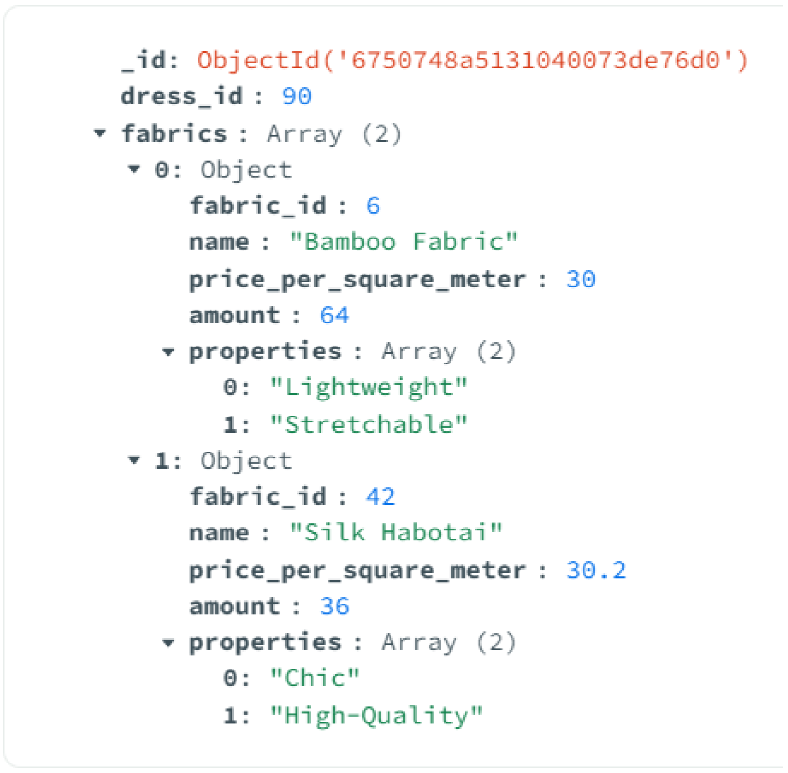
Output:

As can be seen in Figure 4.22 and Figure 4.23, only two documents satisfy the query conditions, that means only those two documents have *production_ time* less than 70 and *size* "XXL".
As specified by the projections, only the *dress_ id* and *fabrics* fields are shown.



```
    _id: ObjectId('6750748a5131040073de7682')
    dress_id : 12
  ▾ fabrics : Array (3)
    ▾ 0: Object
        fabric_id : 6
        name : "Bamboo Fabric"
        price_per_square_meter : 30
        amount : 6
      ▾ properties : Array (2)
          0: "Lightweight"
          1: "Stretchable"
    ▾ 1: Object
        fabric_id : 44
        name : "Lycra"
        price_per_square_meter : 5.2
        amount : 84
      ▾ properties : Array (2)
          0: "Elegant"
          1: "Shimmery"
    ▾ 2: Object
        fabric_id : 47
        name : "Polycotton"
        price_per_square_meter : 12.45
        amount : 10
      ▾ properties : Array (2)
          0: "Durable"
          1: "Moisture-Wicking"
```

Figure 4.22: "QUERY 2" first document returned.



```
_id: ObjectId('6750748a5131040073de76d0')
dress_id : 90
▾ fabrics : Array (2)
  ▾ 0: Object
      fabric_id : 6
      name : "Bamboo Fabric"
      price_per_square_meter : 30
      amount : 64
    ▾ properties : Array (2)
        0: "Lightweight"
        1: "Stretchable"
  ▾ 1: Object
      fabric_id : 42
      name : "Silk Habotai"
      price_per_square_meter : 30.2
      amount : 36
    ▾ properties : Array (2)
        0: "Chic"
        1: "High-Quality"
```

Figure 4.23: "QUERY 2" second document returned.

QUERY 3: Limit, sort, and comparison query operators.

Request:

Find the documents of the three fashion exhibitions with the shortest *duration* in which the model with *fashion_model_personal_id* equal to 1 partook in.

MongoDB code:

```
db.project_collection.find({
  "participating_fashion_models.fashion_model_personal_id": { "$eq": 1}
}).sort({
  "duration":1
}).limit(3)
```

Figure 4.24: MongoDB code of "QUERY 3".

Description:

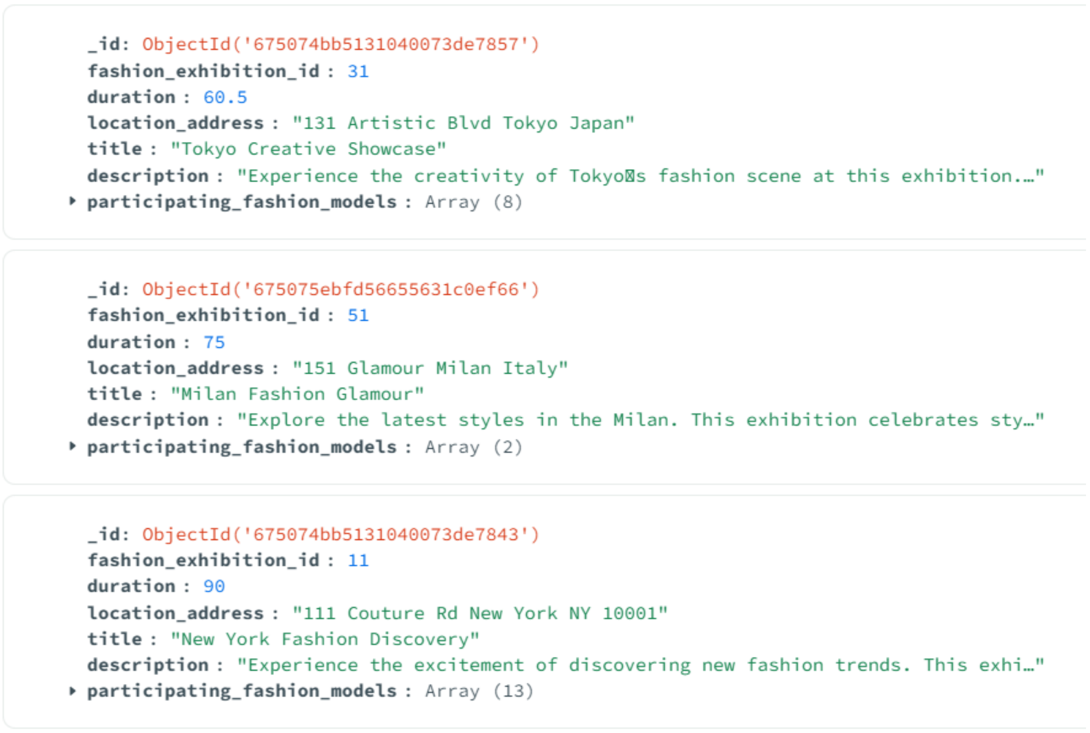
This query searches the *project_collection* for documents with a sub-document in the *participating_fashion_models* array that has *fashion_model_personal_id* equal to 1.

The results are sorted in ascending order based on the *duration* field of the exhibitions, and the query limits the output to the first three matching documents.

Output:

As can be seen in Figure 4.25, the three fashion exhibitions with the shortest *duration* in which the model with *fashion_model_personal_id* equal to 1 partook in are returned as result, sorted in ascending order based on the *duration* field. The three exhibitions have the following *duration*:

- Exhibition with *fashion_exhibition_id* equal to 31 has *duration* 60.5.
- Exhibition with *fashion_exhibition_id* equal to 51 has *duration* 75.
- Exhibition with *fashion_exhibition_id* equal to 11 has *duration* 90.



```
_id: ObjectId('675074bb5131040073de7857')
fashion_exhibition_id : 31
duration : 60.5
location_address : "131 Artistic Blvd Tokyo Japan"
title : "Tokyo Creative Showcase"
description : "Experience the creativity of Tokyo�s fashion scene at this exhibition.…"
▸ participating_fashion_models : Array (8)

_id: ObjectId('675075ebfd56655631c0ef66')
fashion_exhibition_id : 51
duration : 75
location_address : "151 Glamour Milan Italy"
title : "Milan Fashion Glamour"
description : "Explore the latest styles in the Milan. This exhibition celebrates sty…"
▸ participating_fashion_models : Array (2)

_id: ObjectId('675074bb5131040073de7843')
fashion_exhibition_id : 11
duration : 90
location_address : "111 Couture Rd New York NY 10001"
title : "New York Fashion Discovery"
description : "Experience the excitement of discovering new fashion trends. This exhi…"
▸ participating_fashion_models : Array (13)
```

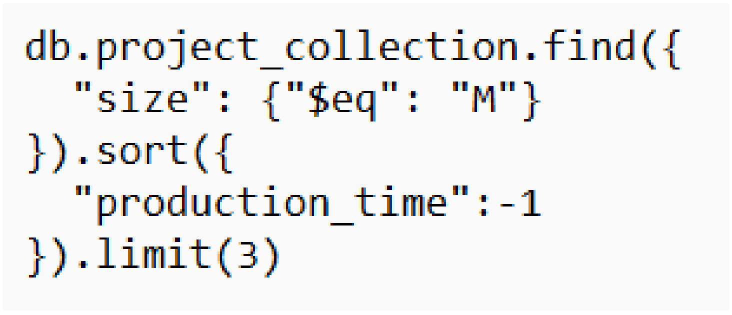
Figure 4.25: "QUERY 3" output.

QUERY 4: Limit, sort, and comparison query operators.

Request:

Find the documents of the three S-sized dresses with the greatest *production_time*. Show the documents in descending order of time.

MongoDB code:



```
db.project_collection.find({
  "size": {"$eq": "M"}
}).sort({
  "production_time":-1
}).limit(3)
```

Figure 4.26: MongoDB code of "QUERY 4".

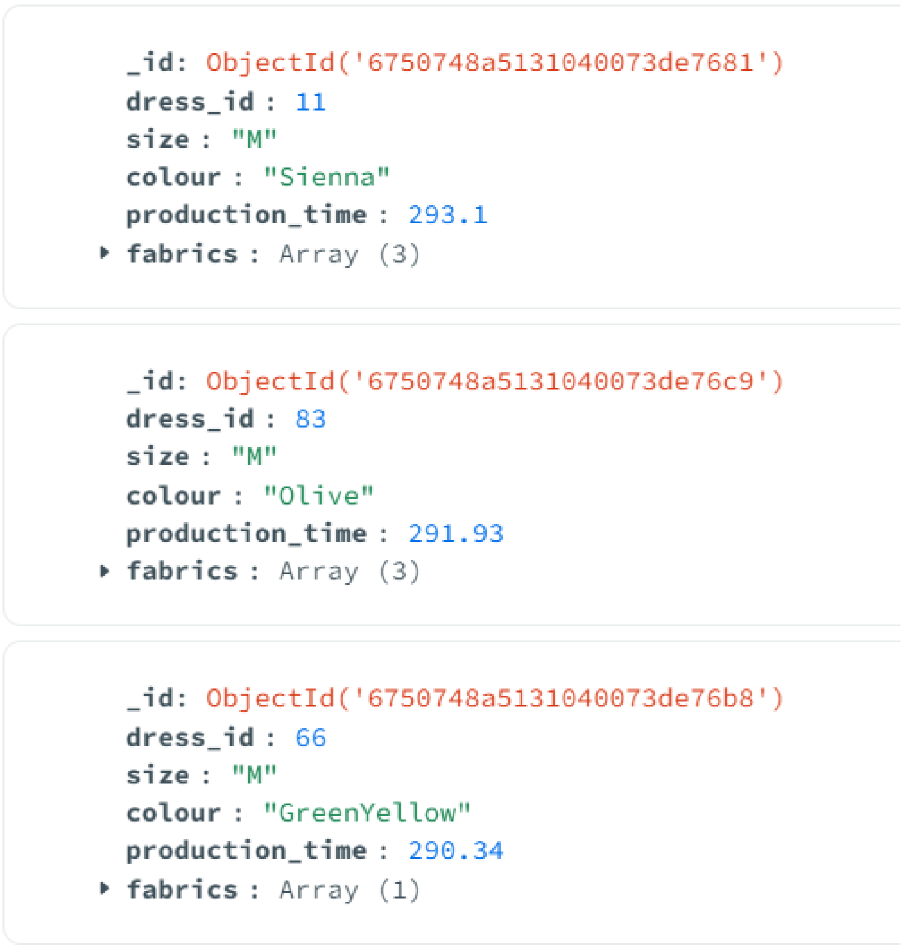
Description:

This query searches the *project_ collection* for documents with the *size* field equals to "S".
The results are sorted in descending order based on the *production_ time* field, and the query limits the output to the first three matching documents.

Output:

As can be seen in Figure 4.27, the three dresses with the greatest *production_ time* field are returned as result, sorted in descending order. The three dresses have the following *production_ time*:

- Dress with *dress_ id* equal to 11 has *production_ time* 293.1.
- Dress with *dress_ id* equal to 83 has *production_ time* 291.93.
- Dress with *dress_ id* equal to 66 has *production_ time* 290.34.



```
_id: ObjectId('6750748a5131040073de7681')
dress_id : 11
size : "M"
colour : "Sienna"
production_time : 293.1
fabrics : Array (3)
```

```
_id: ObjectId('6750748a5131040073de76c9')
dress_id : 83
size : "M"
colour : "Olive"
production_time : 291.93
fabrics : Array (3)
```

```
_id: ObjectId('6750748a5131040073de76b8')
dress_id : 66
size : "M"
colour : "GreenYellow"
production_time : 290.34
fabrics : Array (1)
```

Figure 4.27: "QUERY 4" output.

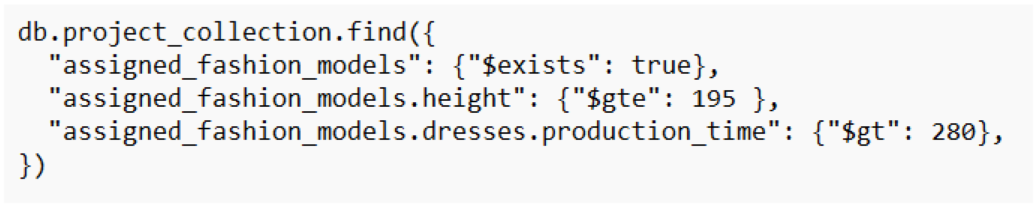
QUERY 5: Element query operators.

Request:

Find documents related to fashion designers that meet the following conditions:

- The *assigned_fashion_models* field exists.
- At least one assigned model is at least 195 centimeters tall.
- At least one assigned model wears a dress with a *production_time* greater than 280.

MongoDB code:



```
db.project_collection.find({
  "assigned_fashion_models": {"$exists": true},
  "assigned_fashion_models.height": {"$gte": 195 },
  "assigned_fashion_models.dresses.production_time": {"$gt": 280},
})
```

Figure 4.28: MongoDB code of "QUERY 5".

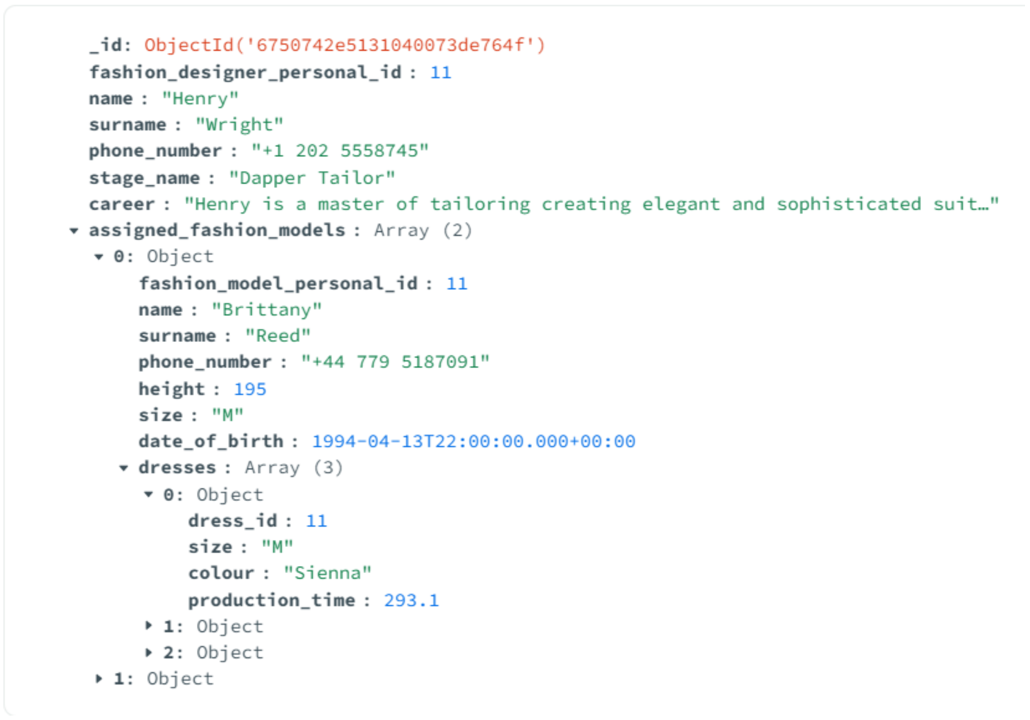
Description:

This query retrieves documents from the *project_ collection* where the *assigned_fashion_ models* field exists.
Additionally, it filters for documents where at least one fashion model in the *assigned_fashion_ models* array has a *height* of 195 or greater and where the *production_ time* for at least one dress in the *assigned_fashion_ models.dresses* subfield is greater than 280.

Output:

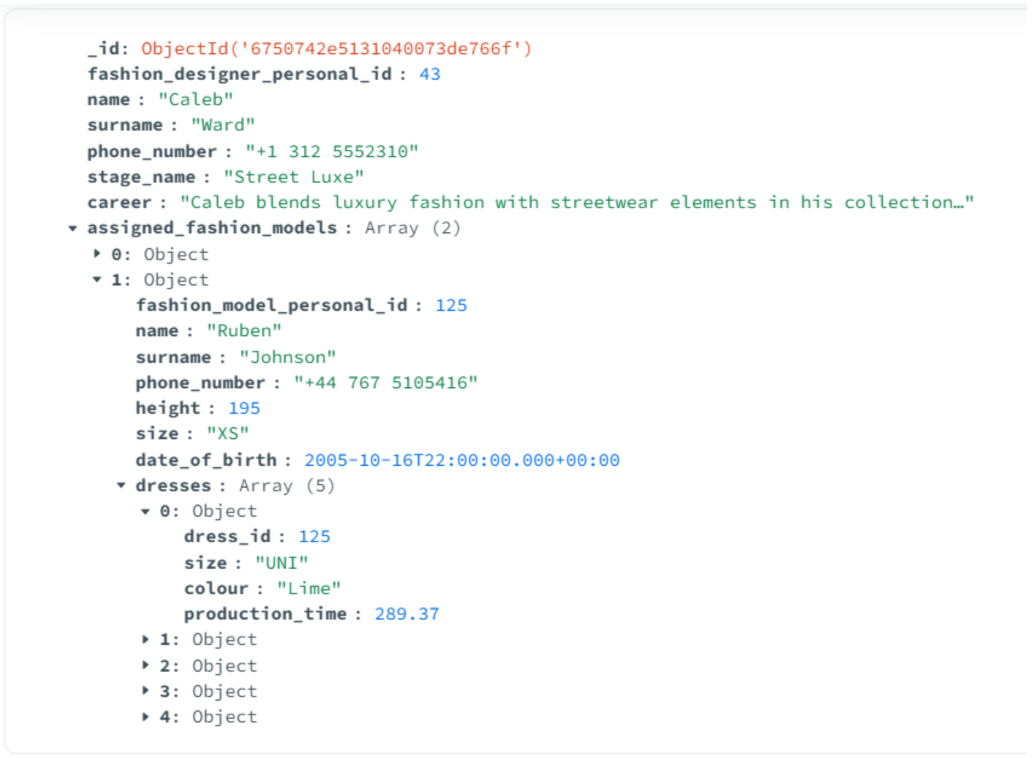
As can be seen in Figure 4.29 and Figure 4.30, only two documents satisfy the query conditions.
In detail:

- The designer with *fashion_ designer_ personal_ id* equal to 11 manages the model with *fashion_ model_ personal_ id* equal to 11, who has a *height* of 195, satisfying the second condition. Additionally, such models also wears a dress with *production_ time* equal to 293.1, satisfying the third condition.
- The designer with *fashion_ designer_ personal_ id* equal to 43 manages the model with *fashion_ model_ personal_ id* equal to 125, who has a *height* of 195, satisfying the second condition. Additionally, such models also wears a dress with *production_ time* equal to 289.37, satisfying the third condition.



```
_id: ObjectId('6750742e5131040073de764f')
fashion_designer_personal_id : 11
name : "Henry"
surname : "Wright"
phone_number : "+1 202 5558745"
stage_name : "Dapper Tailor"
career : "Henry is a master of tailoring creating elegant and sophisticated suit…"
▾ assigned_fashion_models : Array (2)
  ▾ 0: Object
      fashion_model_personal_id : 11
      name : "Brittany"
      surname : "Reed"
      phone_number : "+44 779 5187091"
      height : 195
      size : "M"
      date_of_birth : 1994-04-13T22:00:00.000+00:00
    ▾ dresses : Array (3)
      ▾ 0: Object
          dress_id : 11
          size : "M"
          colour : "Sienna"
          production_time : 293.1
      ▸ 1: Object
      ▸ 2: Object
  ▸ 1: Object
```

Figure 4.29: "QUERY 5" first document returned.



```
_id: ObjectId('6750742e5131040073de766f')
fashion_designer_personal_id : 43
name : "Caleb"
surname : "Ward"
phone_number : "+1 312 5552310"
stage_name : "Street Luxe"
career : "Caleb blends luxury fashion with streetwear elements in his collection…"
▾ assigned_fashion_models : Array (2)
  ▸ 0: Object
  ▾ 1: Object
      fashion_model_personal_id : 125
      name : "Ruben"
      surname : "Johnson"
      phone_number : "+44 767 5105416"
      height : 195
      size : "XS"
      date_of_birth : 2005-10-16T22:00:00.000+00:00
    ▾ dresses : Array (5)
      ▾ 0: Object
          dress_id : 125
          size : "UNI"
          colour : "Lime"
          production_time : 289.37
      ▸ 1: Object
      ▸ 2: Object
      ▸ 3: Object
      ▸ 4: Object
```

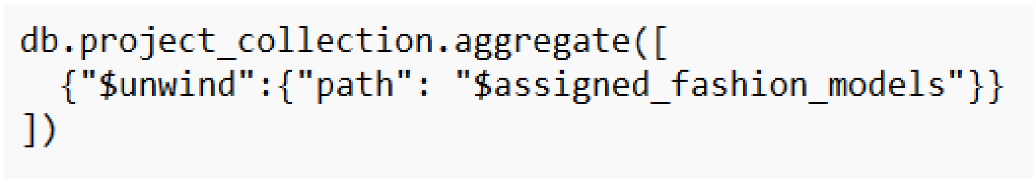
Figure 4.30: "QUERY 5" second document returned.

QUERY 6: Simple aggregation.

Request:

Write a query to *"unwind"* the *assigned_fashion_models* array, so that each element of the array becomes a separate document.

MongoDB code:



```
db.project_collection.aggregate([
  {"$unwind":{"path": "$assigned_fashion_models"}}
])
```

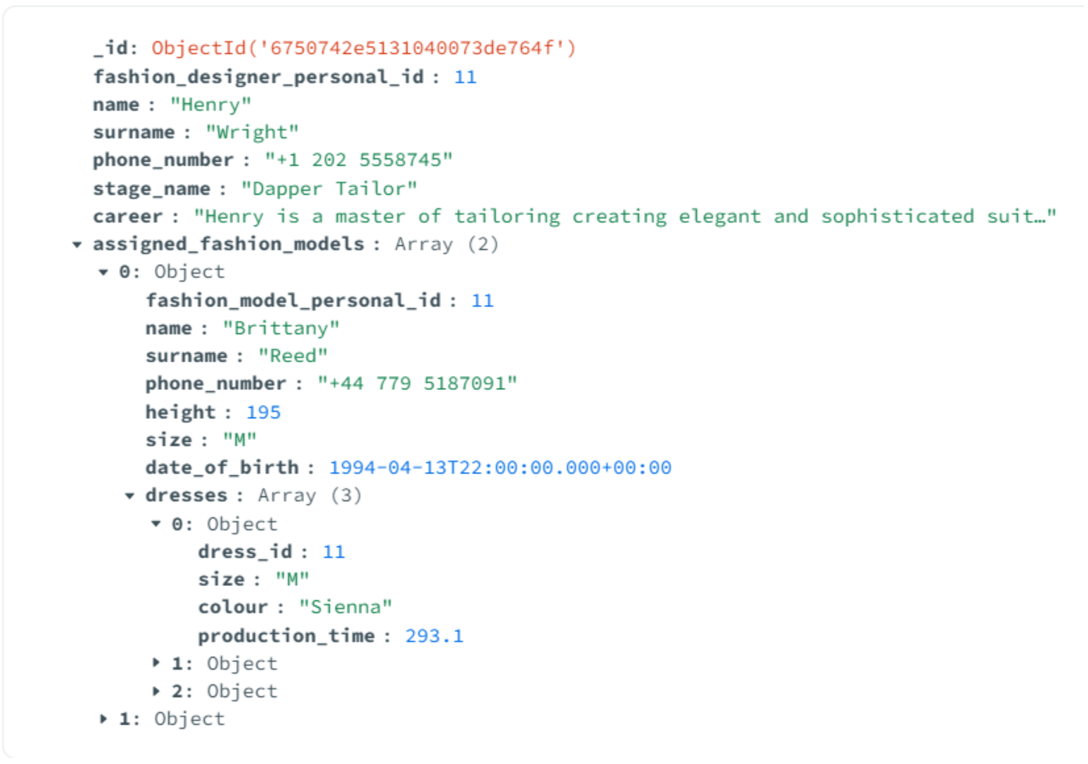
Figure 4.31: MongoDB code of "QUERY 6".

Description:

This query uses *aggregate* and *$unwind* to deconstruct the *assigned_fashion_models* array. It creates a separate document for each fashion model sub-document in the array.

Output:

As can be seen in Figure 4.32 (showing only an example document), each element of the array is output as a separate document, with the remaining fields from the original document preserved.
The complete output is a set of documents where each document contains a single fashion model from the *assigned_fashion_models* array.



```
_id: ObjectId('6750742e5131040073de764f')
fashion_designer_personal_id : 11
name : "Henry"
surname : "Wright"
phone_number : "+1 202 5558745"
stage_name : "Dapper Tailor"
career : "Henry is a master of tailoring creating elegant and sophisticated suit…"
▾ assigned_fashion_models : Array (2)
  ▾ 0: Object
      fashion_model_personal_id : 11
      name : "Brittany"
      surname : "Reed"
      phone_number : "+44 779 5187091"
      height : 195
      size : "M"
      date_of_birth : 1994-04-13T22:00:00.000+00:00
    ▾ dresses : Array (3)
      ▾ 0: Object
          dress_id : 11
          size : "M"
          colour : "Sienna"
          production_time : 293.1
      ▸ 1: Object
      ▸ 2: Object
  ▸ 1: Object
```

Figure 4.32: "QUERY 6" first document returned.

QUERY 7: Unwind and an aggregation.

Request:

Count the total number of models for each model *size*.

MongoDB code:

```
db.project_collection.aggregate([
  {"$unwind":{"path": "$assigned_fashion_models"}},
  {
    "$group":{
        "_id": {
          "model_size": "$assigned_fashion_models.size",
        },
        "num_models_per_model_size": {
          "$sum": 1,
        }
      }
  }
])
```

Figure 4.33: MongoDB code of "QUERY 7".

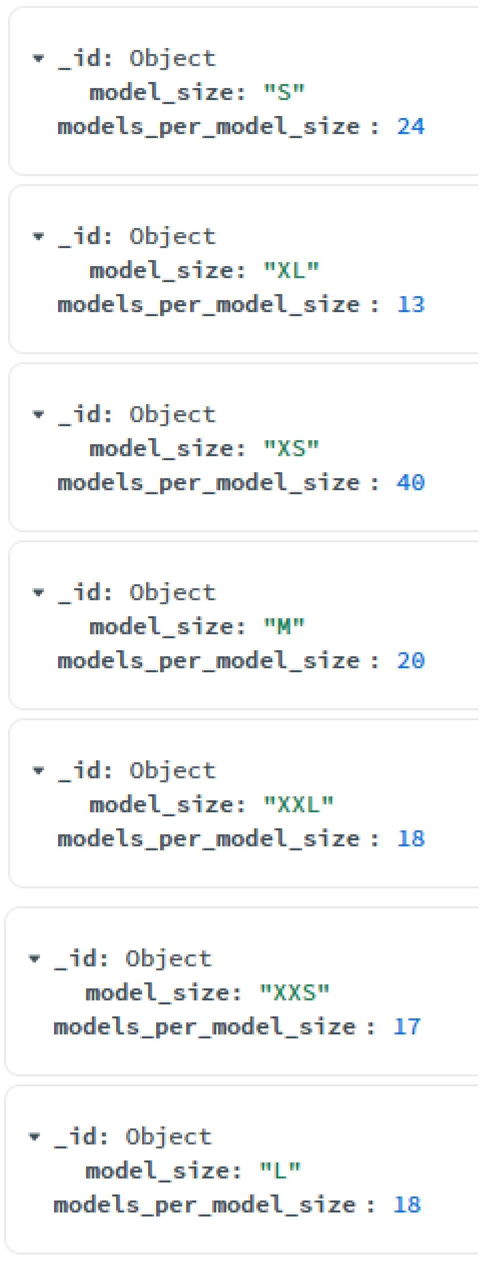
Description:

This query searches the *project_ collection* and executes the *$unwind* for each fashion designers document on the *assigned_fashion_ models* array, creating a separate document for each fashion models in the array.
Subsequently, the *$group* stage groups the documents based on the *size* field within each *assigned_fashion_ models* sub-document. The *_ id* field defines the unique key for each group, which is the *assigned_fashion_ models.size* value.
Finally, the *models_ per_ model_ size* field is computed using *$sum*, incrementing by 1 for each document in the group.

Output:

As can be seen in Figure 4.34, we now report the occurrences of the models for each size:

- *size* "S": 24 models.
- *size* "XL": 13 models.
- *size* "XS": 40 models.
- *size* "M": 20 models.
- *size* "XXL": 18 models.
- *size* "XXS": 17 models.
- *size* "L": 18 models.



```
▾ _id: Object
    model_size: "S"
  models_per_model_size : 24

▾ _id: Object
    model_size: "XL"
  models_per_model_size : 13

▾ _id: Object
    model_size: "XS"
  models_per_model_size : 40

▾ _id: Object
    model_size: "M"
  models_per_model_size : 20

▾ _id: Object
    model_size: "XXL"
  models_per_model_size : 18

▾ _id: Object
    model_size: "XXS"
  models_per_model_size : 17

▾ _id: Object
    model_size: "L"
  models_per_model_size : 18
```

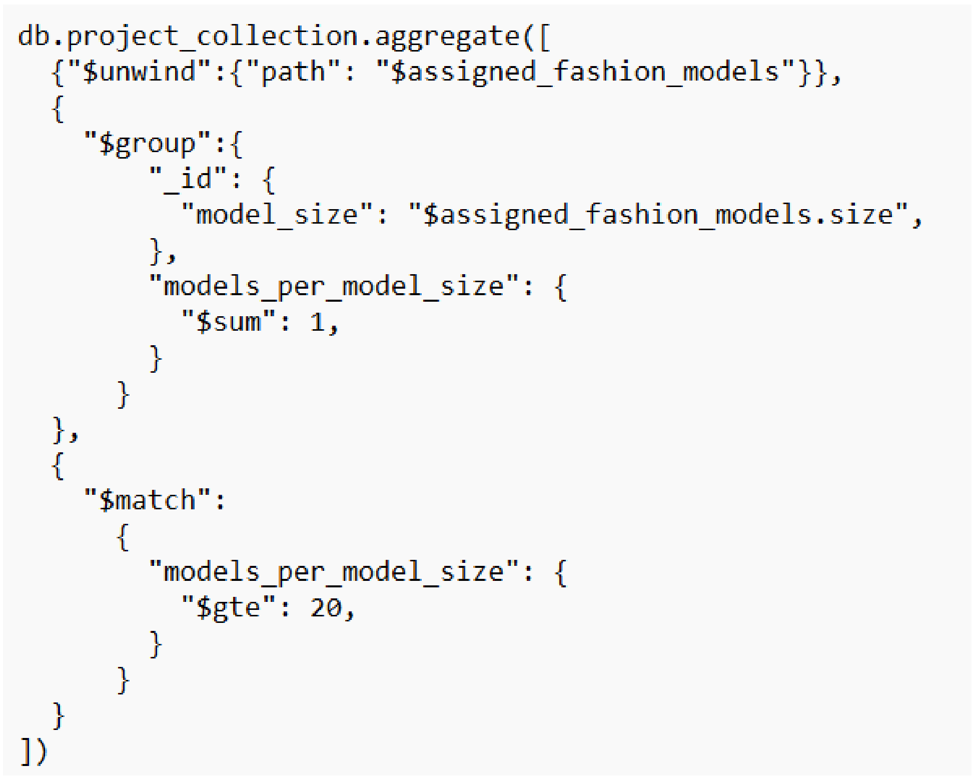
Figure 4.34: "QUERY 7" output.

QUERY 8: Unwind, aggregation, and followed by a filter.

Request:

> Count the total number of models for each model *size*, then keep only the groups with at least 20 occurrences.

MongoDB code:



```
db.project_collection.aggregate([
  {"$unwind":{"path": "$assigned_fashion_models"}},
  {
    "$group":{
        "_id": {
          "model_size": "$assigned_fashion_models.size",
        },
        "models_per_model_size": {
          "$sum": 1,
        }
      }
  },
  {
    "$match":
      {
        "models_per_model_size": {
          "$gte": 20,
        }
      }
  }
])
```

Figure 4.35: MongoDB code of "QUERY 8".

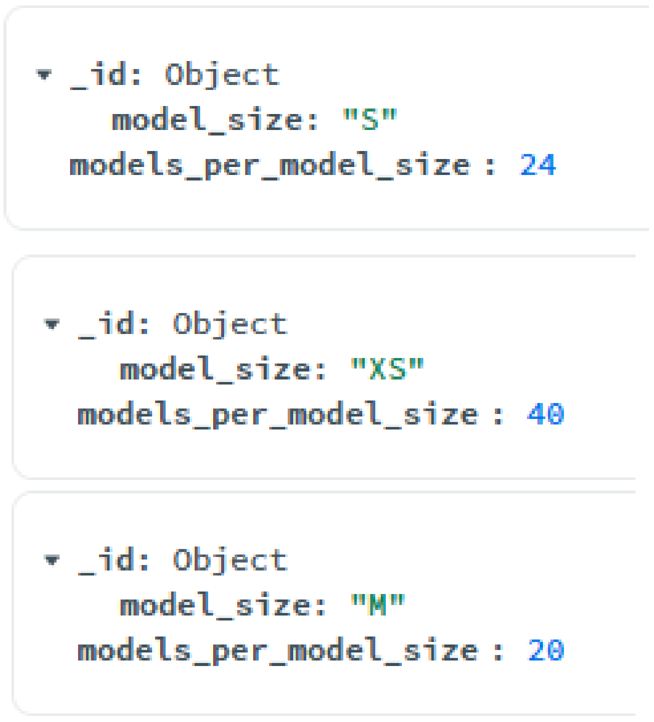
Description:

This query searches the *project_ collection* and executes the *$unwind* for each fashion designers document on the *assigned_ fashion_ models* array, creating a separate document for each fashion models in the array.
Subsequently, the *$group* stage groups the documents based on the *size* field within each *assigned_ fashion_ models* sub-document. The *_ id* field defines the unique key for each group, which is the *assigned_ fashion_ models.size* value.
After, the *models_ per_ model_ size* field is computed using *$sum*, incrementing by 1 for each document in the group.
Lastly, the *$match* stage filters the grouped results, keeping only those where *models_ per_ model_ size* is greater than or equal to 20.

Output:

As can be seen in Figure 4.36, we now report the occurrences of the models for the groups that satisfy the filter condition:

- *size* "S": 24 models.
- *size* "XS": 40 models.
- *size* "M": 20 models.



```
▾ _id: Object
    model_size: "S"
  models_per_model_size : 24

▾ _id: Object
    model_size: "XS"
  models_per_model_size : 40

▾ _id: Object
    model_size: "M"
  models_per_model_size : 20
```

Figure 4.36: "QUERY 8" output.

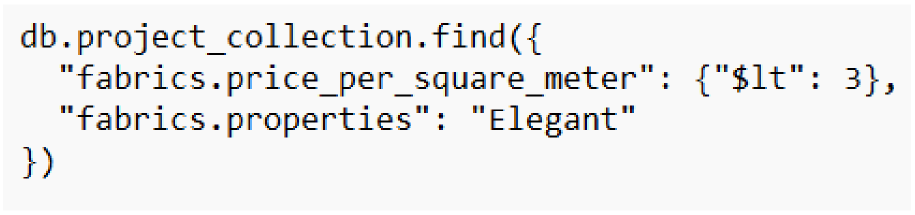
QUERY 9: Two conditions, evaluated separately, on a sub document of an array.

Request:

Find the documents of dresses where the *fabrics* array contain one or more fabrics that satisfy the following conditions when evaluated separately:

- At least one fabric in the array must have a *price_per_square_meter* lower than 3.
- At least one fabric in the array must have the property "Elegant".

MongoDB code:



```
db.project_collection.find({
  "fabrics.price_per_square_meter": {"$lt": 3},
  "fabrics.properties": "Elegant"
})
```

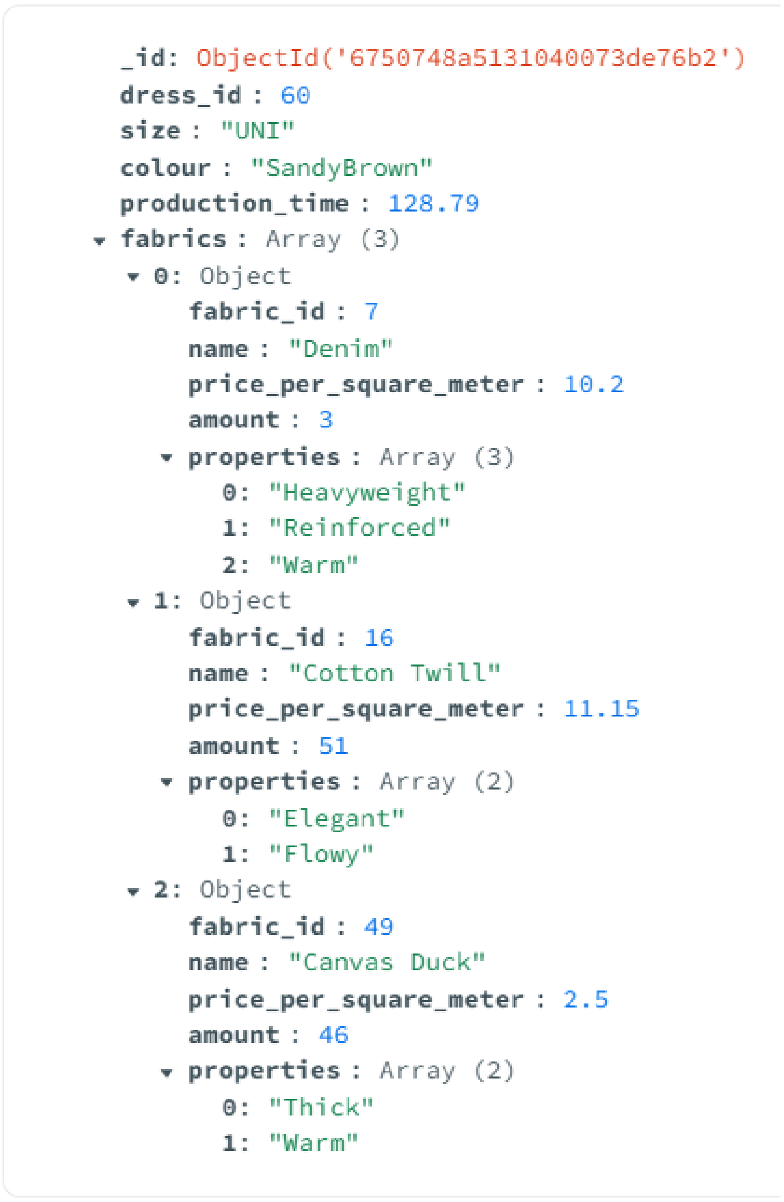
Figure 4.37: MongoDB code of "QUERY 9".

Description:

This query searches the *project_collection* to find all documents of dresses where the conditions on the *fabrics* array are satisfied when evaluated separately.
All the documents included in the result have at least one sub-document in the *fabrics* array that has *price_per_square_meter* lower than 3 and at least one sub-document (can be different because the conditions are evaluated separately) in the *fabrics* array that has "Elegant" in the *properties* array.
Documents that meet these conditions will be included in the returned result.

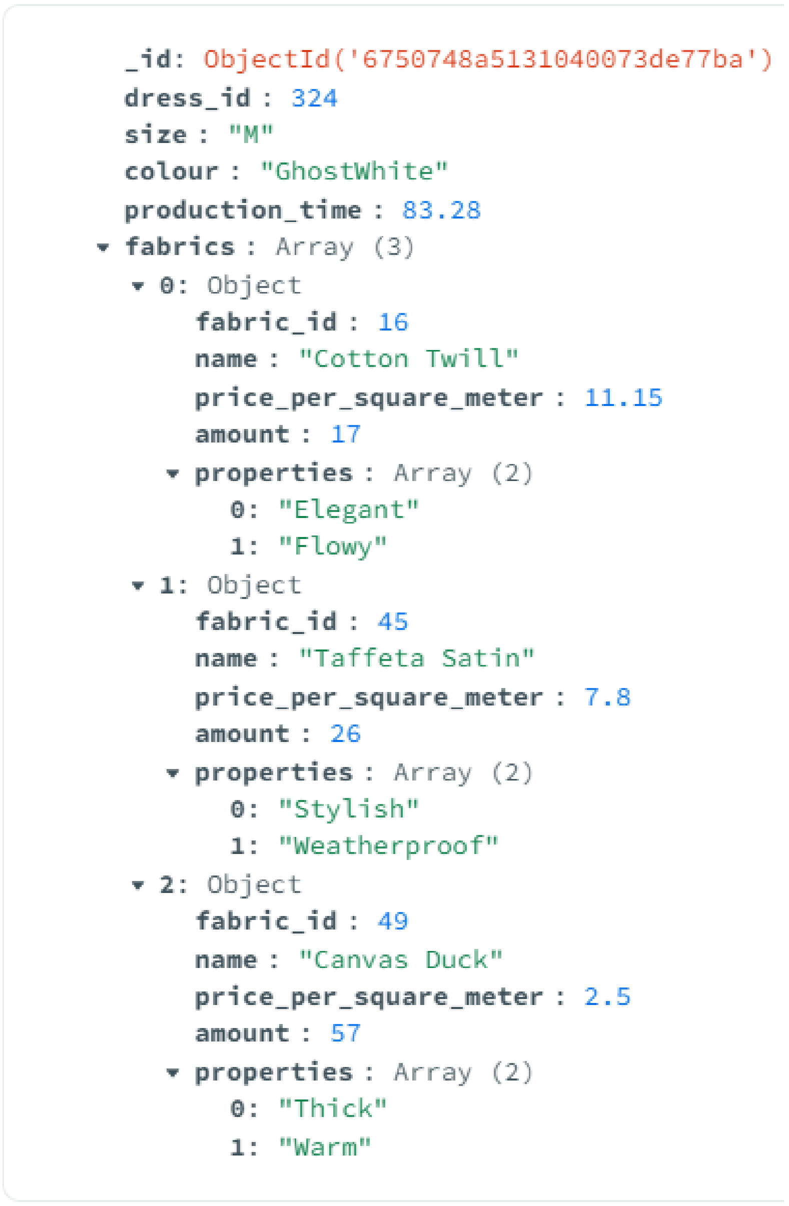
Output:

As can be seen in Figure 4.38 and Figure 4.39, the query has found a total of two documents whose *fabrics* array satisfies the conditions.
This means there are two dresses in which at least one fabric has *price_per_square_meter* lower than 3 and at least one fabric that has "Elegant" in the *properties* array.
The dresses included in the result have *dress_id* equal to 5, 8, 60, 324.



```
_id: ObjectId('6750748a5131040073de76b2')
dress_id : 60
size : "UNI"
colour : "SandyBrown"
production_time : 128.79
▾ fabrics : Array (3)
  ▾ 0: Object
      fabric_id : 7
      name : "Denim"
      price_per_square_meter : 10.2
      amount : 3
    ▾ properties : Array (3)
        0: "Heavyweight"
        1: "Reinforced"
        2: "Warm"
  ▾ 1: Object
      fabric_id : 16
      name : "Cotton Twill"
      price_per_square_meter : 11.15
      amount : 51
    ▾ properties : Array (2)
        0: "Elegant"
        1: "Flowy"
  ▾ 2: Object
      fabric_id : 49
      name : "Canvas Duck"
      price_per_square_meter : 2.5
      amount : 46
    ▾ properties : Array (2)
        0: "Thick"
        1: "Warm"
```

Figure 4.38: "QUERY 9" first document returned.



```
_id: ObjectId('6750748a5131040073de77ba')
dress_id : 324
size : "M"
colour : "GhostWhite"
production_time : 83.28
▾ fabrics : Array (3)
  ▾ 0: Object
      fabric_id : 16
      name : "Cotton Twill"
      price_per_square_meter : 11.15
      amount : 17
    ▾ properties : Array (2)
        0: "Elegant"
        1: "Flowy"
  ▾ 1: Object
      fabric_id : 45
      name : "Taffeta Satin"
      price_per_square_meter : 7.8
      amount : 26
    ▾ properties : Array (2)
        0: "Stylish"
        1: "Weatherproof"
  ▾ 2: Object
      fabric_id : 49
      name : "Canvas Duck"
      price_per_square_meter : 2.5
      amount : 57
    ▾ properties : Array (2)
        0: "Thick"
        1: "Warm"
```

Figure 4.39: "QUERY 9" second document returned.

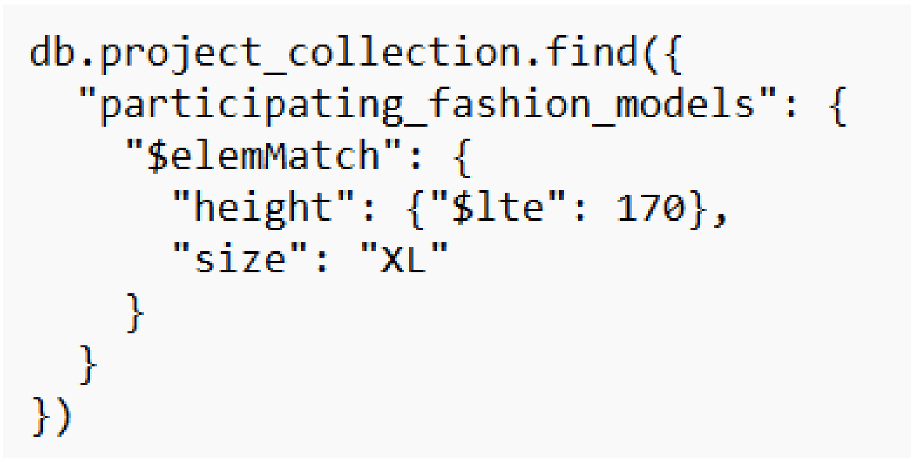
QUERY 10: Two conditions, evaluated simultaneously, on a sub-document of an array.

Request:

Find the documents of the fashion exhibitions where the *participating_fashion_models* array contain a model who satisfies simultaneously the following conditions:

- The model is less than or equal to 170 centimeters tall.
- The model wears *size* "XL".

MongoDB code:



```
db.project_collection.find({
  "participating_fashion_models": {
    "$elemMatch": {
      "height": {"$lte": 170},
      "size": "XL"
    }
  }
})
```

Figure 4.40: MongoDB code of "QUERY 10".

Description:

This query searches the *project_ collection* to find all the documents of fashion exhibitions where the condition on the *participatin_fashion_ models* array is satisfied.
All the documents included in the result have at least one sub-document in the *participating_fashion_ models* array that has:

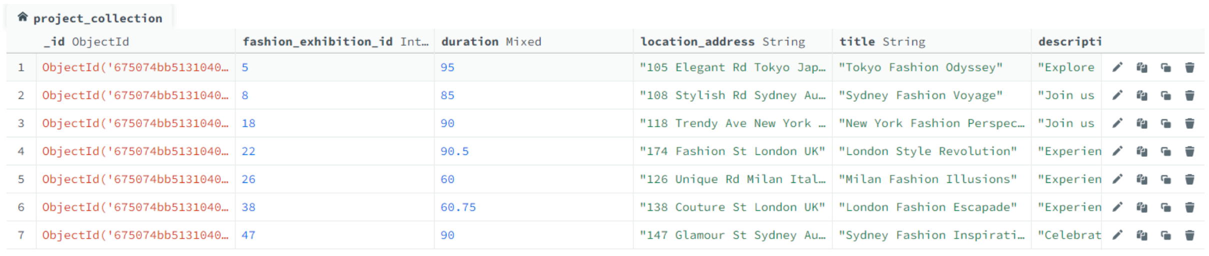
- *height* less than or equal to 170.
- *size* equal to "XL".

The query uses the *$elemMatch* operator to ensure both conditions are applied in the same sub-document of the *participating_fashion_ models* array.
Documents that meet these conditions will be included in the returned result.

Output:

As can be seen in Figure 4.41, the query has found a total of seven documents whose *participating_fashion_ models* array satisfies the conditions.
This means there are seven exhibitions in which at least one model who is less than or equal to 170 centimeters tall and wears *size* "XL" partook in.
The exhibitions included in the result have *fashion_ exhibition_ id* equal to 5, 8, 18, 22, 26, 38 and 47.
The models that satisfy the conditions are:

- Model with *personal_id* equals 143, having *height* equals 164 and *size* "XL".
- Model with *personal_id* equals 72, having *height* equals 170 and *size* "XL".



project_collection

	_id ObjectId	fashion_exhibition_id Int…	duration Mixed	location_address String	title String	descripti
1	ObjectId('675074bb5131040…	5	95	"105 Elegant Rd Tokyo Jap…	"Tokyo Fashion Odyssey"	"Explore
2	ObjectId('675074bb5131040…	8	85	"108 Stylish Rd Sydney Au…	"Sydney Fashion Voyage"	"Join us
3	ObjectId('675074bb5131040…	18	90	"118 Trendy Ave New York …	"New York Fashion Perspec…	"Join us
4	ObjectId('675074bb5131040…	22	90.5	"174 Fashion St London UK"	"London Style Revolution"	"Experien
5	ObjectId('675074bb5131040…	26	60	"126 Unique Rd Milan Ital…	"Milan Fashion Illusions"	"Experien
6	ObjectId('675074bb5131040…	38	60.75	"138 Couture St London UK"	"London Fashion Escapade"	"Experien
7	ObjectId('675074bb5131040…	47	90	"147 Glamour St Sydney Au…	"Sydney Fashion Inspirati…	"Celebrat

Figure 4.41: "QUERY 10" output.

5 | Conclusion

The third Enterprise ICT Architectures project involved defining a MongoDB database, defining a collection and some documents.
We used MongoDB Compass for the creation of the MongoDB database and for querying it, ensuring the accuracy and functionality of the designed queries.
This project was very stimulating and enhanced our understanding of MongoDB databases, providing us with key skills for defining and querying of them.