MODUL 6
PROJECTION DAN AGGREGATION

6.1. Tujuan Praktikum

1. Mahasiswa mampu menggunakan query projection
2. Mahasiswa mampu menggunakan query Aggregation

6.2. Dasar Teori

6.2.1. Update dan Save Document MongoDB

Metode update() dan save() MongoDB digunakan untuk memperbarui dokumen menjadi koleksi. Metode update() memperbarui nilai dalam dokumen yang ada

Sintaks dasar metode update() adalah sebagai berikut:

```
>db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)
```

6.2.2. Remove Document MongoDB

Metode remove() MongoDB digunakan untuk menghapus dokumen dari koleksi. metode remove() menerima dua parameter. Salah satunya adalah kriteria penghapusan dan yang kedua adalah bendera justOne.

- kriteria penghapusan: (Opsional) kriteria penghapusan menurut dokumen akan dihapus.
- justOne: (Opsional) jika disetel ke true atau 1, maka hapus hanya satu dokumen.

Sintaks dasar metode remove() adalah sebagai berikut:

```
>db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)
```

Jika ada beberapa record dan Anda hanya ingin menghapus record pertama, maka atur hanya satu parameter dalam metode remove().

```
>db.COLLECTION_NAME.remove(DELETION_CRITERIA,1)
```

Pada mongoDB 6.0 perintah Remove diganti menjadi Delete. Dan variasinya adalah deleteOne() untuk menghapus satu dokumen saja dan deleteMany() untuk menghapus banyak dokumen sekaligus.

6.2.3. Limit Record MongoDB

Untuk membatasi record di MongoDB, Anda perlu menggunakan metode limit(). Metode ini menerima satu argumen tipe angka, yang merupakan jumlah dokumen yang ingin Anda tampilkan.

Sintaks dasar metode limit() adalah sebagai berikut:

```
>db.COLLECTION_NAME.find().limit(NUMBER)
```

Selain metode limit(), ada satu lagi metode skip() yang juga menerima argumen tipe angka dan digunakan untuk melewatkan jumlah dokumen. Sintaks dasar metode skip() adalah sebagai berikut:

```
>db.COLLECTION_NAME.find().limit(NUMBER).skip(NUMBER)
```

6.2.4. Indexing MongoDB

Indeks mendukung resolusi query yang efisien. Tanpa indeks, MongoDB harus memindai setiap dokumen koleksi untuk memilih dokumen yang cocok dengan pernyataan query. Pemindaian ini sangat tidak efisien dan membuat MongoDB harus memproses data dalam jumlah besar.

Indeks adalah struktur data khusus, yang menyimpan sebagian kecil kumpulan data dalam bentuk yang mudah dilintasi. Indeks menyimpan nilai bidang tertentu atau kumpulan bidang, diurutkan berdasarkan nilai bidang sebagaimana ditentukan dalam indeks. Untuk membuat indeks, Anda perlu menggunakan metode createIndex() dari MongoDB. Sintaks dasar metode **createIndex()** adalah sebagai berikut:

```
>db.COLLECTION_NAME.createIndex({KEY:1})
```

Kuncinya adalah nama file yang ingin Anda buat indeksnya, nilai 1 untuk urutan menaik dan nilai -1 untuk membuat indeks dalam urutan menurun. Contoh:

```
>db.mycol.createIndex({"title":1})
```

Dalam metode **createIndex**() Anda dapat mengirim banyak parameter, untuk membuat indeks pada banyak field. Contoh:

```
>db.mycol.createIndex({"title":1,"description":-1})
```

Metode **createIndex()** juga memiliki sejumlah opsi (yang bersifat opsional). Berikut ini adalah daftarnya:

Parameter	Type	Description
Unique	Boolean	Membuat indeks unik sehingga koleksi tidak akan menerima penyisipan dokumen di mana kunci atau kunci indeks cocok dengan nilai yang ada di indeks. Tentukan true untuk membuat indeks unik. Nilai defaultnya salah.
Name	String	Nama indeks. Jika tidak ditentukan, MongoDB menghasilkan nama indeks dengan menggabungkan nama bidang yang diindeks dan mengurutkannya.
Sparse	Boolean	Jika bernilai **true**, indeks hanya mereferensikan dokumen dengan bidang tertentu. Indeks ini menggunakan lebih sedikit ruang tetapi berperilaku berbeda dalam beberapa situasi (terutama pengurutan). Nilai defaultnya **false**.
expireAfterSeconds	Integer	Menentukan nilai, dalam hitungan detik, sebagai TTL untuk mengontrol berapa lama MongoDB menyimpan dokumen dalam kumpulan ini.
V	Index Version	Nomor versi indeks. Versi indeks default bergantung pada versi MongoDB yang berjalan saat membuat indeks.
Weights	Document	Bobotnya adalah angka mulai dari 1 hingga 99.999 dan menunjukkan pentingnya field yang terikat terhadap field lain yang diindeks dalam bentuk skor.
default_language	String	Untuk indeks teks, bahasa yang digunakan untuk menentukan kata yang digunakan untuk

		menghentikan proses dan memberikan aturan untuk stemmer dan tokenizer. Nilai default adalah **english**.
language_override	String	Untuk indeks teks, tentukan nama field dalam dokumen yang berisi kata yang akan diganti bahasanya. Nilai default adalah **language**.

6.2.5. Aggregation MongoDB

Operasi agregasi memproses data record dan mengembalikan hasil yang dihitung. Operasi agregasi mengelompokkan nilai dari beberapa dokumen secara bersamaan, dan dapat melakukan berbagai operasi pada data yang dikelompokkan untuk mengembalikan satu hasil. Dalam SQL, perintah count(*) dan group by setara dengan agregasi mongodb. Sintaks dasar metode **aggregate()** adalah sebagai berikut:

```
>db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

Berikut adalah daftar ekspresi agregasi yang tersedia:

Expression	Description
$sum	Menjumlahkan nilai yang ditentukan dari semua dokumen dalam koleksi.
$avg	Menghitung rata-rata semua nilai yang diberikan dari semua dokumen dalam koleksi.
$min	Mendapatkan nilai minimum yang sesuai dari semua dokumen dalam koleksi.
$max	Mendapatkan nilai maksimum yang sesuai dari semua dokumen dalam koleksi.
$push	Menyisipkan nilai ke array dalam dokumen yang dihasilkan.
$addToSet	Menyisipkan nilai ke array dalam dokumen yang dihasilkan tetapi tidak membuat duplikat.

$first	Mendapat dokumen pertama dari dokumen sumber sesuai dengan pengelompokannya. Biasanya ini hanya digunakan bersama dengan beberapa tahap "$sort" yang diterapkan sebelumnya.
$last	Mendapat dokumen terakhir dari dokumen sumber sesuai dengan pengelompokannya. Biasanya ini hanya digunakan bersama dengan beberapa tahap "$sort" yang diterapkan sebelumnya.

Pipeline Concept. Dalam perintah UNIX, Pipeline berarti kemungkinan untuk menjalankan operasi pada beberapa input dan menggunakan output sebagai input untuk perintah berikutnya dan seterusnya. MongoDB juga mendukung konsep yang sama dalam kerangka agregasi. Ada satu set tahapan yang mungkin dan masing-masing diambil sebagai satu set dokumen sebagai input dan menghasilkan satu set dokumen yang dihasilkan (atau dokumen JSON yang dihasilkan akhir di akhir pipeline). pipeline ini kemudian dapat digunakan untuk tahap berikutnya dan seterusnya.

Expression	Description
$project	Digunakan untuk memilih beberapa bidang tertentu dari koleksi.
$match	Ini adalah operasi penyaringan dan dengan demikian dapat mengurangi jumlah dokumen yang diberikan sebagai input ke tahap berikutnya.
$group	Ini melakukan agregasi aktual seperti yang dibahas di atas.
$sort	Mengurutkan dokumen.
$skip	Dengan ini, dimungkinkan untuk melewati daftar dokumen untuk sejumlah dokumen tertentu.
$limit	Ini membatasi jumlah dokumen untuk dilihat, dengan nomor yang diberikan mulai dari posisi saat ini.
$unwind	Ini digunakan untuk melepas dokumen yang menggunakan array. Saat menggunakan larik, data yang sudah digabungkan sebelumnya dengan menjalankan operasi ini akan dibatalkan, sehingga dokumen akan kembali ke dokumen individual lagi.

	Maka dari itu dengan perintah ini kita akan menambah jumlah dokumen untuk tahap selanjutnya.

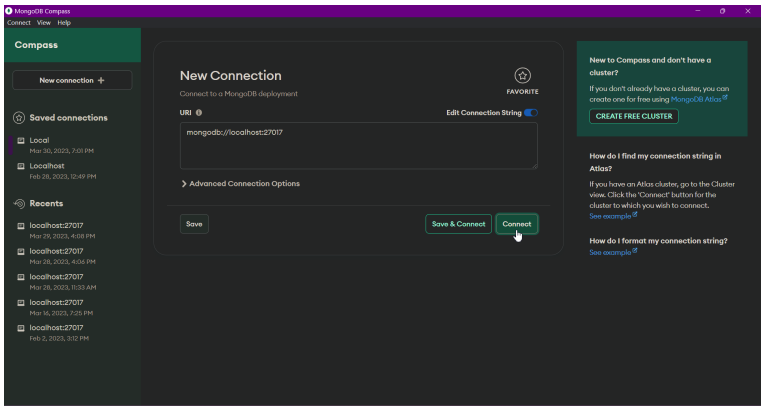
6.4. *Software*

1. MongoDB
2. Compass
3. MongoDB Shell

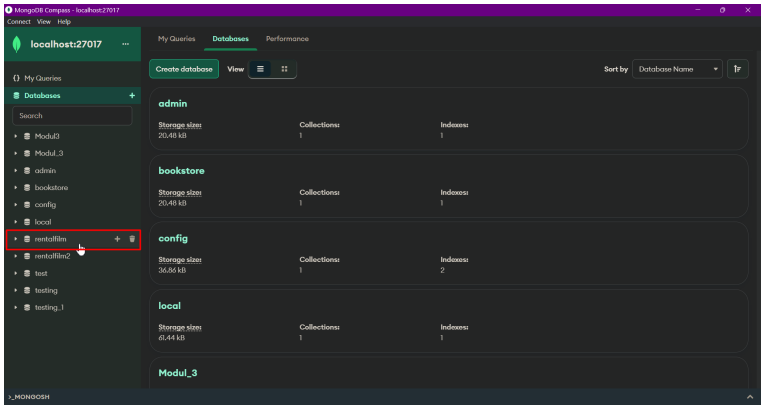
6.5. Tahapan Kerja

6.4.1. Menjalankan MongoDB di Compass dan Command Prompt

1. Buka Aplikasi MongoDB Compass kemudian klik “Connect”



2. Untuk melanjutkan ketahapan kerja selanjutnya pastikan database “Rentalfilm” sudah ada di MongoDB.



3. Buka MongoShell dengan melakukan perintah “mongosh” di dalam command prompt.

```
C:\Users\Denny>mongosh
```

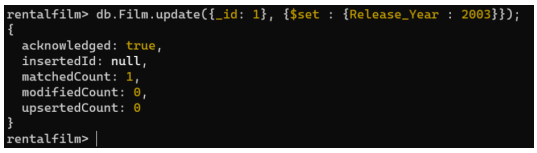
4. Lalu ganti ke database rental film dengan menggunakan perintah “use”.

```
test> use rentalfilm
switched to db rentalfilm
rentalfilm>
```

6.4.2. Update dan save document

1. Update

```
rentalfilm> db.Film.update({_id: 1}, {$set : {Release_Year : 2003}});
```



6.4.3. Remove Document

1. Remove

```
rentalfilm> db.Film.remove({_id: 1})
```

2. Remove One

```
rentalfilm> db.Film.deleteOne({Rating: 9})
```

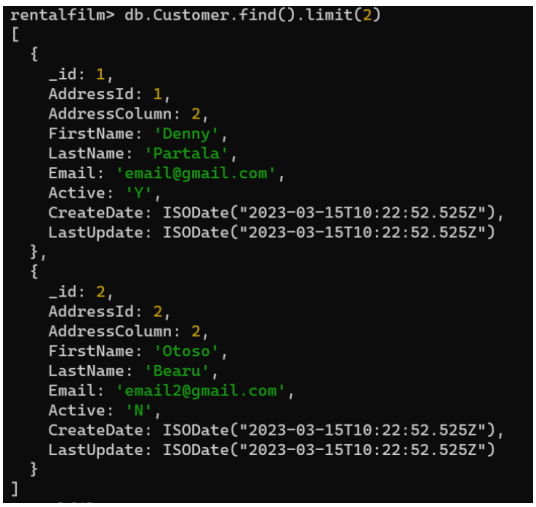
3. Remove All

```
rentalfilm> db.Film.deleteMany({Length: 90})
```

6.4.4. Limit Document

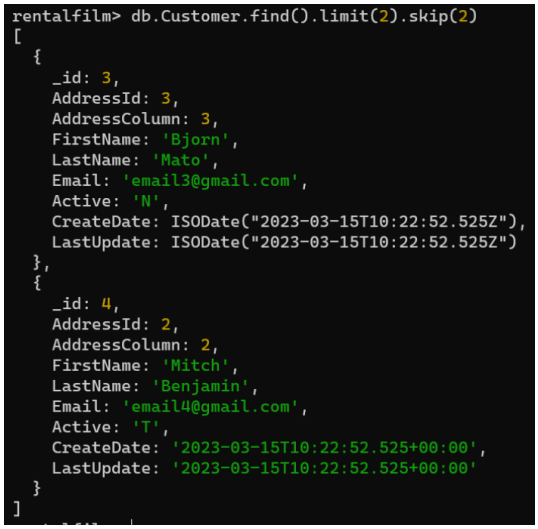
1. Limit

```
rentalfilm> db.Customer.find().limit(2)
```



2. Skip

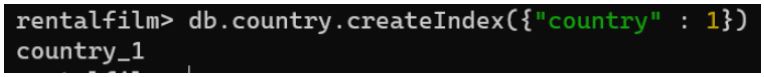
```
rentalfilm> db.Customer.find().limit(2).skip(2)
```



6.4.5. Index

1. ensureIndex()

```
rentalfilm> db.country.createIndex({"country" : 1})
```



2. unique

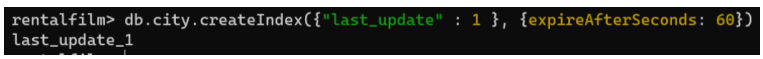
```
rentalfilm> db.country.createIndex({"country" : 1}, {unique : true})
```

3. sparse

```
rentalfilm> db.Film.createIndex({"LanguageID" : 1}, {sparse : true})
```

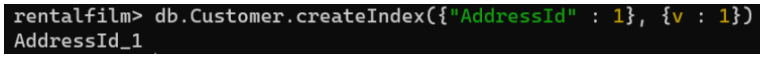
4. expireAfterSeconds

```
rentalfilm> db.city.createIndex({"last_update" : 1}, {expireAfterSeconds :
60})
```



5. v

```
rentalfilm> db.Customer.createIndex({"AddressId" : 1}, {v : 1})
```



6. weights

```
rentalfilm> db.Film.createIndex({Title : "text"}, {weights: {Title : 5}})
```

```
rentalfilm> db.Film.createIndex({Title : "text"}, {weights: {Title : 5}})
Title_text
rentalfilm>
```

7. default_language

```
rentalfilm> db.Customer.createIndex({Email : "text"}, {default_language:
"english"})
```

```
rentalfilm> db.Customer.createIndex({Email : "text"}, {default_language: "english"})
Email_text
rentalfilm>
```

8. language_override

```
rentalfilm> db.Customer.createIndex({FirstName : "text"}, {language_override:
"spanish"})
```

```
rentalfilm> db.Customer.createIndex({FirstName : "text"}, {language_override: "spanish"})
FirstName_text
rentalfilm>
```

6.4.6. Aggregation

1. *$sum*

```
rentalfilm> db.Film.aggregate([ {$group: { _id: null, Total_Replacement_Cost:
{ $sum: '$Replacement_Cost' } } }]);
```

```
rentalfilm> db.Film.aggregate([ { $group: { _id: null, Total_Replacement_Cost: { $sum: '$Replacement_Cost' } } }]);
[ { _id: null, Total_Replacement_Cost: 6000 } ]
rentalfilm>
```

2. *$avg*

```
rentalfilm> db.Film.aggregate([ {$group: { _id: null, Total_Replacement_Cost:
{ $avg: '$Replacement_Cost' } } }]);
```

```
rentalfilm> db.Film.aggregate([ { $group: { _id: null, Total_Replacement_Cost: { $avg: '$Replacement_Cost' } } }]);
[ { _id: null, Total_Replacement_Cost: 1160 } ]
rentalfilm>
```

3. *$min*

```
rentalfilm> db.Film.aggregate([ {$group: { _id: null, Total_Replacement_Cost:
{ $min: '$Replacement_Cost' } } }]);
```

```
rentalfilm> db.Film.aggregate([ { $group: { _id: null, Total_Replacement_Cost: { $min: '$Replacement_Cost' } } }]);
[ { _id: null, Total_Replacement_Cost: 1000 } ]
rentalfilm>
```

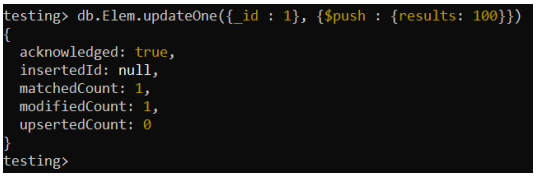
4. *$max*

```
rentalfilm> db.Film.aggregate([ {$group: { _id: null, Total_Replacement_Cost:
{ $max: '$Replacement_Cost' }} }]);
```

```
rentalfilm> db.Film.aggregate([ { $group: { _id: null, Total_Replacement_Cost: { $max: '$Replacement_Cost' } } }]);
[ { _id: null, Total_Replacement_Cost: 1200 } ]
rentalfilm>
```

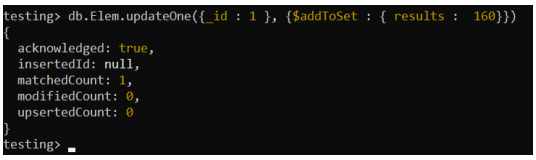
5. *$push*

```
rentalfilm> db.Elem.updateOne({_id : 1}, {$push : {results: 100}})
```



6. *$addToSet*

```
rentalfilm> db.Elem.updateOne({_id : 1}, {$addToSet : {results: 160}})
```



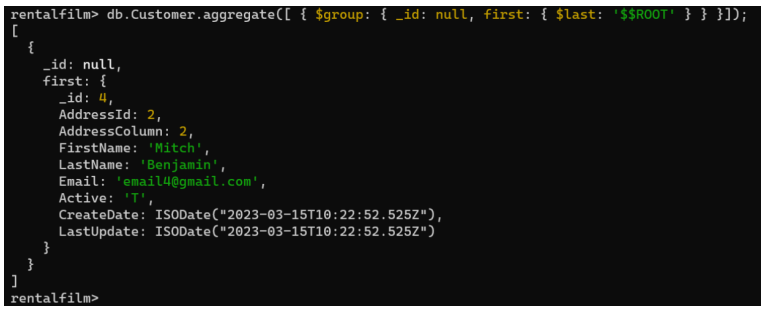
7. *$first*

```
rentalfilm> db.Customer.aggregate({
        {
          $group: {
          _id: null,
          first: {
            $first: '$$ROOT'
          }
         }
        }
]);
[
  {
    _id: null,
    first: {
    _id: 1,
    AddressId: 1,
    AddressColumn: 2,
    FirstName: 'Denny',
    LastName: 'Partala'
    Email: 'email@gmail.com',
    Active: 'Y',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
   }
  }
]
```

‘$$ROOT’ disini mereferensikan ke dokumen yang ada didalam collection itu sendiri.

8. *$last*

```
rentalfilm> db.Customer.aggregate([{$group: {_id: null, first:{ $last:
'$$ROOT' } } } ]);
```



6.4.7. Pipeline

1. *$project*

   Digunakan untuk meneruskan document dengan field yang diminta ke tahapan pipeline selanjutnya.

```
rentalfilm> db.Film.aggregate([
       {
           $project: {
              "Last_Update" : 0
           }
       }
     ])
[
   {
     _id: 2,
     Title: 'Country Bear',
     Description: '-',
     Release_Year: 2023,
     Rental_Duration: 1608,
     Rental_Rate: 10.2,
     Length: 90,
     Replacement_Cost: 1200,
     Rating: 9,
     Special_Features: 'Bears',
     Fulltext: 'Bear Bear Bear 2'
   },
```

   Contoh diatas menunjukan penggunaan $project untuk tidak menampilkan field yang tidak kita perlukan.

2. *$match*

   Melakukan filter dokumen sesuai kondisi yang ditentukan dan diteruskan ke tahap pipeline selanjutnya.

```
rentalfilm> db.Film.aggregate([{$group: {"Last_Update" : 0 } }, { $match: {
Title: "Country Bear" } }]);
```

Contohnya diatas dilakukan $project pada tahap pertama dimana field last_update dihilangkan kemudian diteruskan ketahap kedua dimana $match akan mencari dokumen yang memiliki field ‘Title’ dengan value ‘Country Bear’.

3. *$group*

```
rentalfilm> db.Customer.aggregate([
        {
              $group: {
              _id: '$Active',
              }
        }
  ]);
[ { _id: 'Y' }, { _id: 'T' }, { _id: 'N' }]
```

$Group digunakan untuk mengelompokan field dengan value unik sesuai yang telah ditentukan. $group juga bisa digunakan dengan operator yang lain seperti $last, $avg dll.

4. *$sort*

```
rentalfilm> db.Customer.aggregate([
        {
              $sort: {
                LastName: 1
              }
        }
  ]);
```

```
rentalfilm> db.Customer.aggregate([
...     {
...         $sort: {
...           LastName: 1
...         }
...     }
... ]);
[
  {
    _id: 2,
    AddressId: 2,
    AddressColumn: 2,
    FirstName: 'Otoso',
    LastName: 'Bearu',
    Email: 'email2@gmail.com',
    Active: 'N',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:50.525Z")
  },
  {
    _id: 4,
    AddressId: 2,
    AddressColumn: 2,
    FirstName: 'Mitch',
    LastName: 'Benjamin',
    Email: 'email4@gmail.com',
    Active: 'T',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
```

$sort digunakan untuk melakukan pengurutan data pada field yang diberikan, dan disini bisa digunakanjua weight yang telah ditentukan pada index apabila sudah dibuat.

5. *$limit*

```
rentalfilm> db.Customer.aggregate([
        {
             $Limit: 2
        }
  ])
```

```
rentalfilm> db.Customer.aggregate([
...     {
...         $limit: 2
...     }
... ])
[
  {
    _id: 1,
    AddressId: 1,
    AddressColumn: 2,
    FirstName: 'Denny',
    LastName: 'Partala',
    Email: 'email@gmail.com',
    Active: 'Y',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 2,
    AddressId: 2,
    AddressColumn: 2,
    FirstName: 'Otoso',
    LastName: 'Bearu',
    Email: 'email2@gmail.com',
    Active: 'N',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:50.525Z")
  }
]
rentalfilm>
```

Digunakan untuk membatasi berapa banyak dokumen yang akan diteruskan ketahapan pipeline berikutnya.

6. *$unwind*

```
rentalfilm> db.TipeDataValidasi.aggregate([
        {
              $unwind: {
                 Path: 'Tag',
              }
        }
  ]);
```

```
Modul3> db.TipeDataValidasi.aggregate([
...     {
...         $unwind: {
...           path: '$Tag',
...         }
...     }
... ]);
[
  {
    _id: ObjectId("64227bf92ec0c7264609f7c8"),
    Nama_Produk: 'Mesin Cuci',
    Harga: 10000000,
    Berat_Produk: 10.9,
    Rusak: false,
    Tanggal_Masuk: ISODate("2023-03-28T05:32:41.054Z"),
    Tag: 'Elektronik',
    Dimensi: { tinggi: 60, panjang: 50, lebar: 20 },
    Id_Produk: ObjectId("64227bf92ec0c7264609f7c7")
  },
  {
    _id: ObjectId("64227bf92ec0c7264609f7c8"),
    Nama_Produk: 'Mesin Cuci',
    Harga: 10000000,
    Berat_Produk: 10.9,
    Rusak: false,
    Tanggal_Masuk: ISODate("2023-03-28T05:32:41.054Z"),
    Tag: 'Perabotan',
    Dimensi: { tinggi: 60, panjang: 50, lebar: 20 },
    Id_Produk: ObjectId("64227bf92ec0c7264609f7c7")
  }
]
Modul3>
```

Digunakan ketika ingin dekonstruksi array menjadi field nya masing masing.

6.5. Tugas dan Latihan

1. Buatlah dan jalankan query untuk membuat setiap jenis tipe index pada database rentalfilm yang berbeda dengan contoh pada tahapan kerja.
2. Buatlah dan jalankan setiap query aggregate pada database rentalfilm yang berbeda dengan contoh yang telah diberikan.
3. Buatlah dan jalankan setiap query pipeline pada database rentalfilm yang berbeda dengan contoh yang telah diberikan.
4. Buatlah dan jalankan minimal 3 buah gabungan query pipeline pada database rentalfilm.