

Nama : Muhammad Alfian 212310017

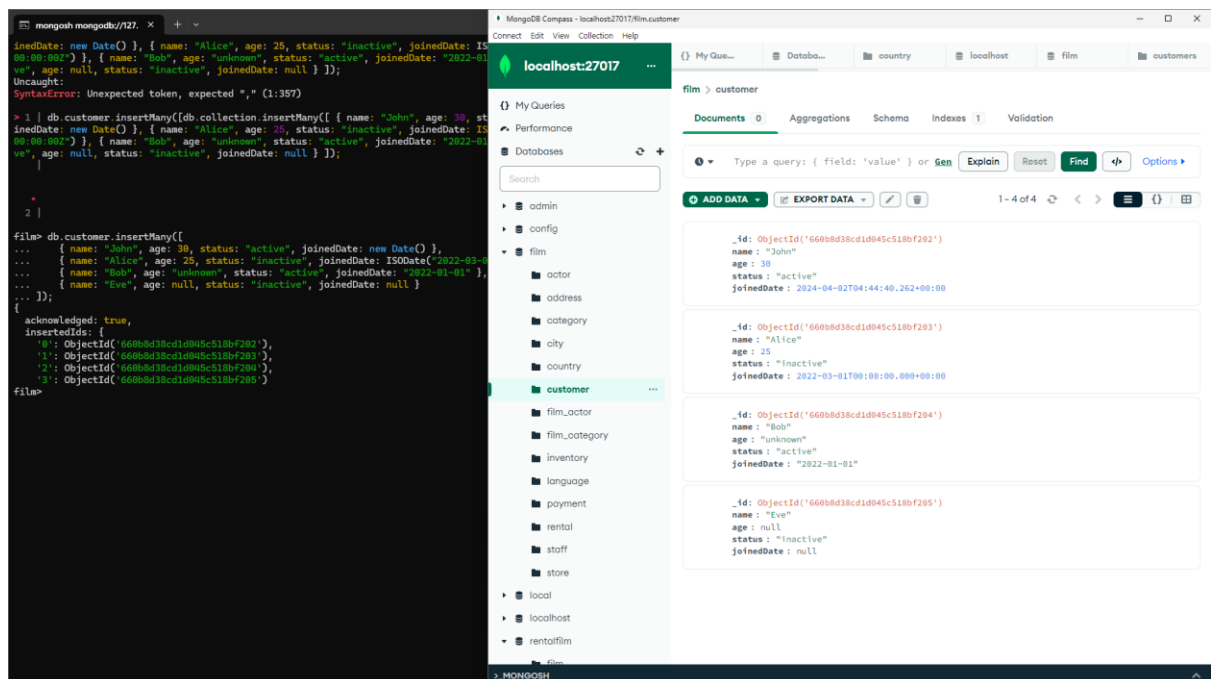
Muhammad Angga Parulian Harahap 212310043

Fathurahman AL Faridzi 212310018

Dindha Achmad Maulana 212310050

Tugas Lab Basis Data | Modul 4

1. Buatlah serta jalankan query untuk membuat lebih dari 1 document baru yang menggunakan semua jenis tipe data.



2. Buatlah collection baru serta tambahkan validator dengan minimal 2 field.

```
mongosh mongodb://127.0.0.1:27021/
2 |
film> db.customer.insertMany([
...   { name: "John", age: 30, status: "active", joinedDate: new Date() },
...   { name: "Alice", age: 25, status: "inactive", joinedDate: ISODate("2022-03-01T00:00:00Z") },
...   { name: "Bob", age: "unknown", status: "active", joinedDate: "2022-01-01" },
...   { name: "Eve", age: null, status: "inactive", joinedDate: null }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('660b8d38cd1d045c518bf202'),
    '1': ObjectId('660b8d38cd1d045c518bf203'),
    '2': ObjectId('660b8d38cd1d045c518bf204'),
    '3': ObjectId('660b8d38cd1d045c518bf205')
  }
}
film> db.createCollection('users', {
...   validator: {
...     $jsonSchema: {
...       bsonType: 'object',
...       required: ['name', 'email'],
...       properties: {
...         name: { bsonType: 'string' },
...         email: { bsonType: 'string', pattern: '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$' }
...       }
...     }
...   }
... });
{ ok: 1 }
```

```
mongosh mongodb://127.0.0.1:27021/
film> db.createCollection('users', {
...   validator: {
...     $jsonSchema: {
...       bsonType: 'object',
...       required: ['name', 'email'],
...       properties: {
...         name: { bsonType: 'string' },
...         email: { bsonType: 'string', pattern: '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$' }
...       }
...     }
...   }
... });
{ ok: 1 }
film> db.createCollection('present', {
...   validator: {
...     $jsonSchema: {
...       bsonType: 'object',
...       required: ['name', 'email'],
...       properties: {
...         name: { bsonType: 'string', description: "'name' harus dalam rupa string dan wajib diisi" },
...         email: { bsonType: 'string', pattern: '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$' }
...       }
...     }
...   }
... });
{ ok: 1 }
film>
```

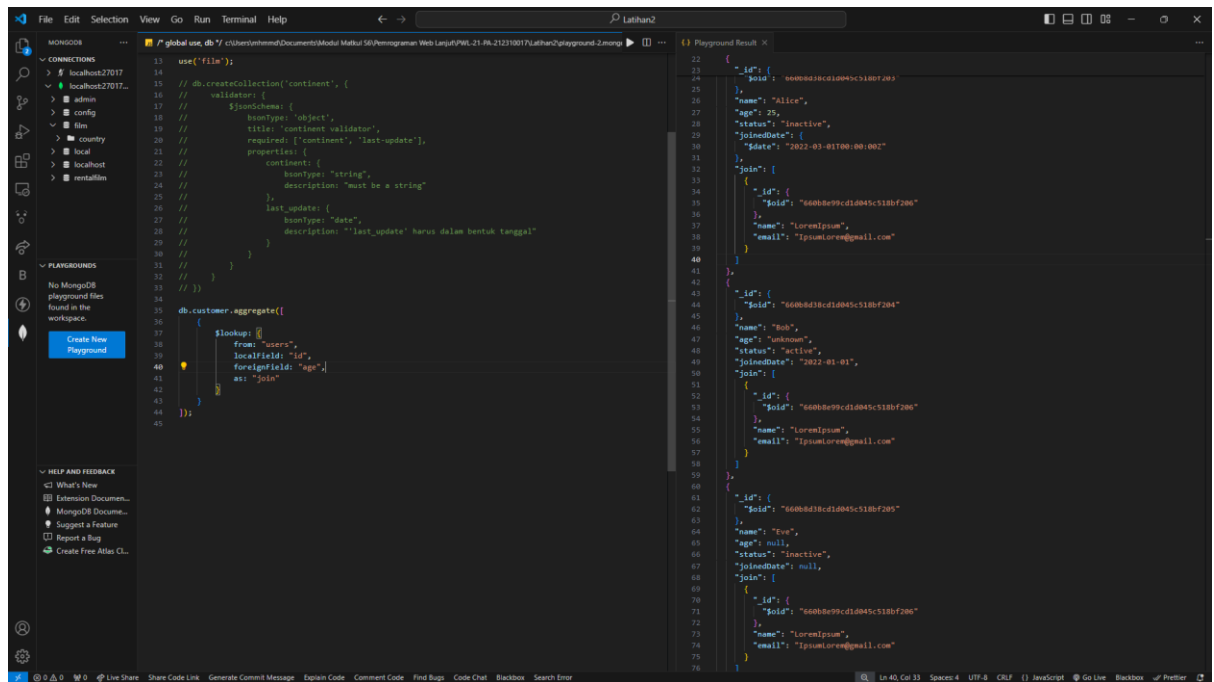
3. Buat dan jalankan query untuk menambahkan data yang valid kedalam collection yang sudah ada validatornya.

```
mongosh mongodb://127.0.0.1:27027
...   validator: {
...     $jsonSchema: {
...       bsonType: 'object',
...       required: ['name', 'email'],
...       properties: {
...         name: { bsonType: 'string' },
...         email: { bsonType: 'string', pattern: '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$' }
...       }
...     }
...   };
film> db.createCollection('present', {
...   validator: {
...     $jsonSchema: {
...       bsonType: 'object',
...       required: ['name', 'email'],
...       properties: {
...         name: { bsonType: 'string', description: "'name' harus dalam rupa string dan wajib diisi" },
...         email: { bsonType: 'string', pattern: '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$' }
...       }
...     }
...   };
... });
{ ok: 1 }
film> db.users.insertOne({ name: "LoremIpsum", email: "IpsumLorem@gmail.com" });
{
  acknowledged: true,
  insertedId: ObjectId('660b8e99cd1d045c518bf206')
}
film>
```

4. Buat dan jalankan query untuk menambahkan data yang tidak valid kedalam collection yang sudah ada validatornya.

```
mongosh mongodb://127.0.0.1:27027
film> db.users.insertOne({ name: "Lorem", email: "invalid-email" });
Uncaught:
MongoServerError: Document failed validation
Additional information: {
  failingDocumentId: ObjectId('660b8ee6cd1d045c518bf207'),
  details: {
    operatorName: '$jsonSchema',
    schemaRulesNotSatisfied: [
      {
        operatorName: 'properties',
        propertiesNotSatisfied: [
          {
            propertyName: 'email',
            details: [
              {
                operatorName: 'pattern',
                specifiedAs: {
                  pattern: '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
                },
                reason: 'regular expression did not match',
                consideredValue: 'invalid-email'
              }
            ]
          }
        ]
      }
    ]
  }
}
film>
```

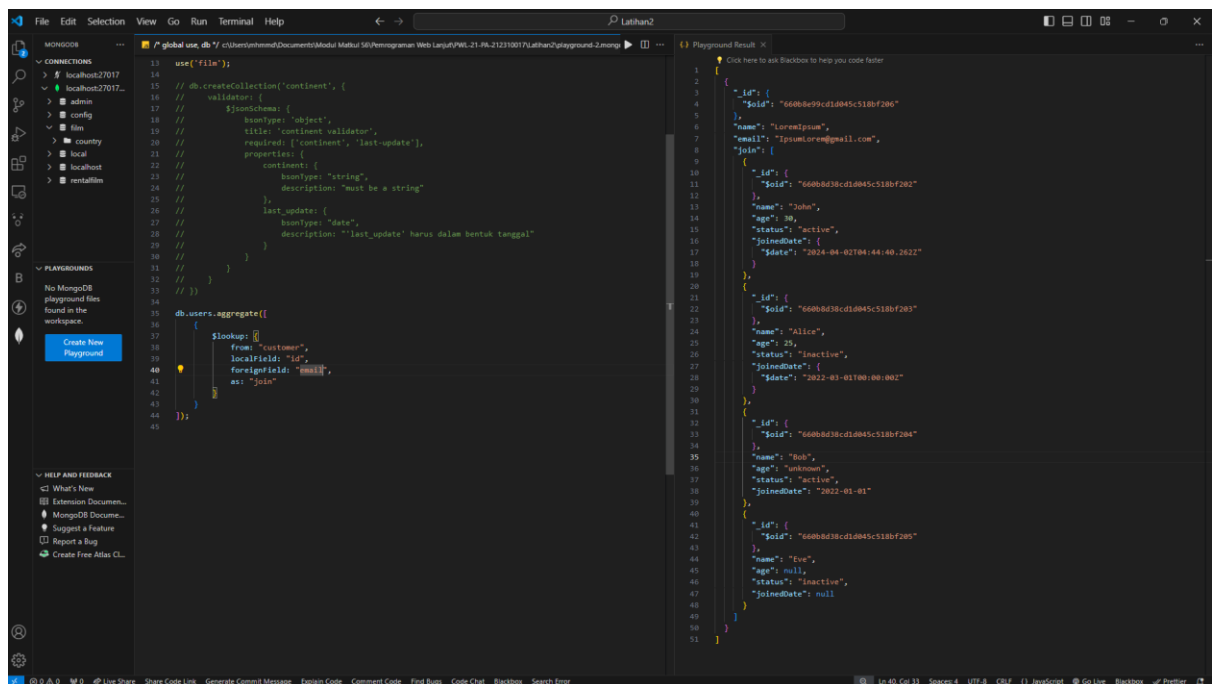
5. Buat dan jalankan dua buah query lookup pada database rentalfilm yang telah dibuat selain pada collection country dan city.



The screenshot shows the MongoDB Playground interface. The left sidebar displays the database structure with collections: admin, config, film, country, local, localhost, and rentalfilm. The main editor contains a JavaScript query using the `aggregate` pipeline with a `$lookup` stage. The query joins the `customer` collection with the `users` collection based on the `id` field. The right pane shows the resulting JSON documents, which include fields for `_id`, `id`, `name`, `age`, `status`, `joinedDate`, and `join`.

```
13 use('film');
14
15 // db.createCollection('continent', {
16 //   validator: {
17 //     $jsonSchema: {
18 //       bsonType: 'object',
19 //       title: 'continent validator',
20 //       required: ['continent', 'last-update'],
21 //       properties: {
22 //         continent: {
23 //           bsonType: 'string',
24 //           description: 'must be a string'
25 //         },
26 //         last_update: {
27 //           bsonType: 'date',
28 //           description: '"last_update" harus dalam bentuk tanggal'
29 //         }
30 //       }
31 //     }
32 //   })
33 // })
34
35 db.customer.aggregate([
36   {
37     $lookup: {
38       from: 'users',
39       localField: 'id',
40       foreignField: 'age',
41       as: 'join'
42     }
43   }
44 ]);
```

```
22 {
23   "_id": {
24     "id": "666b8d38cd18945c318f283"
25   },
26   "name": "Alice",
27   "age": 29,
28   "status": "inactive",
29   "joinedDate": {
30     "date": "2022-03-01T00:00:00Z"
31   },
32   "join": [
33     {
34       "_id": {
35         "id": "666b8d38cd18945c318f284"
36       },
37       "name": "Lorem Ipsum",
38       "email": "Ipsum@lorem@gmail.com"
39     }
40   ]
41 },
42 {
43   "_id": {
44     "id": "666b8d38cd18945c318f284"
45   },
46   "name": "Bob",
47   "age": "unknown",
48   "status": "active",
49   "joinedDate": "2022-01-01",
50   "join": [
51     {
52       "_id": {
53         "id": "666b8d38cd18945c318f286"
54       },
55       "name": "Lorem Ipsum",
56       "email": "Ipsum@lorem@gmail.com"
57     }
58   ]
59 },
60 {
61   "_id": {
62     "id": "666b8d38cd18945c318f285"
63   },
64   "name": "Eve",
65   "age": null,
66   "status": "inactive",
67   "joinedDate": null,
68   "join": [
69     {
70       "_id": {
71         "id": "666b8d38cd18945c318f286"
72       },
73       "name": "Lorem Ipsum",
74       "email": "Ipsum@lorem@gmail.com"
75     }
76   ]
77 }
```



The screenshot shows the MongoDB Playground interface. The left sidebar displays the database structure with collections: admin, config, film, country, local, localhost, and rentalfilm. The main editor contains a JavaScript query using the `aggregate` pipeline with a `$lookup` stage. The query joins the `users` collection with the `customer` collection based on the `id` field. The right pane shows the resulting JSON documents, which include fields for `_id`, `id`, `name`, `age`, `status`, `joinedDate`, and `join`.

```
13 use('film');
14
15 // db.createCollection('continent', {
16 //   validator: {
17 //     $jsonSchema: {
18 //       bsonType: 'object',
19 //       title: 'continent validator',
20 //       required: ['continent', 'last-update'],
21 //       properties: {
22 //         continent: {
23 //           bsonType: 'string',
24 //           description: 'must be a string'
25 //         },
26 //         last_update: {
27 //           bsonType: 'date',
28 //           description: '"last_update" harus dalam bentuk tanggal'
29 //         }
30 //       }
31 //     }
32 //   })
33 // })
34
35 db.users.aggregate([
36   {
37     $lookup: {
38       from: 'customer',
39       localField: 'id',
40       foreignField: 'id',
41       as: 'join'
42     }
43   }
44 ]);
```

```
1
2
3 {
4   "_id": {
5     "id": "666b8d38cd18945c318f286"
6   },
7   "name": "Lorem Ipsum",
8   "email": "Ipsum@lorem@gmail.com",
9   "join": [
10     {
11       "_id": {
12         "id": "666b8d38cd18945c318f282"
13       },
14       "name": "John",
15       "age": 30,
16       "status": "active",
17       "joinedDate": {
18         "date": "2024-04-02T04:44:40.262Z"
19       }
20     }
21   },
22   {
23     "_id": {
24       "id": "666b8d38cd18945c318f283"
25     },
26     "name": "Alice",
27     "age": 29,
28     "status": "inactive",
29     "joinedDate": {
30       "date": "2022-03-01T00:00:00Z"
31     }
32   },
33   {
34     "_id": {
35       "id": "666b8d38cd18945c318f284"
36     },
37     "name": "Bob",
38     "age": "unknown",
39     "status": "active",
40     "joinedDate": "2022-01-01"
41   },
42   {
43     "_id": {
44       "id": "666b8d38cd18945c318f285"
45     },
46     "name": "Eve",
47     "age": null,
48     "status": "inactive",
49     "joinedDate": null
50   }
51 }
```