Sprints

– Task 26 –

MongoDB CRUD Operations and Data Aggregation Task

Name: Basel Amr Barakat
Email: baselamr52@gmail.com

Sprints

Task Overview

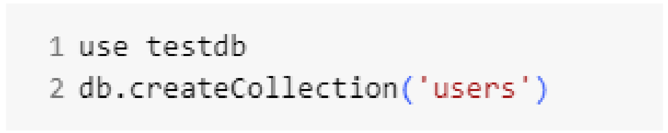
- Objective: Utilize MongoDB's CRUD operations and aggregation framework to manage and analyze data effectively.
- Key Goals:
  - Perform CRUD Operations (Create, Read, Update, Delete)
  - Perform Aggregation Tasks (Average Age, Sorting, Filtering)
  - Gain Insights (Youngest and Oldest Group)

Sprints

Requirement One

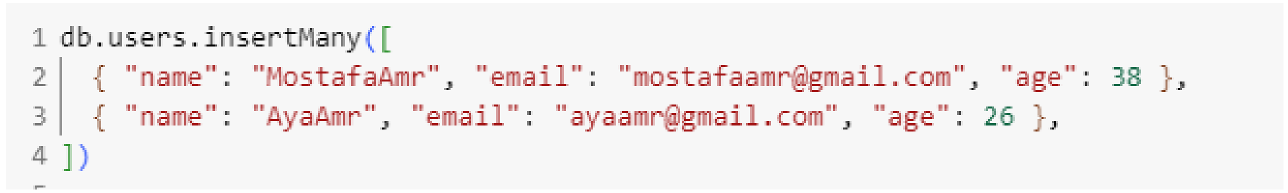
MongoDB Setup and CRUD Operations

1.1 Database and collection Creation



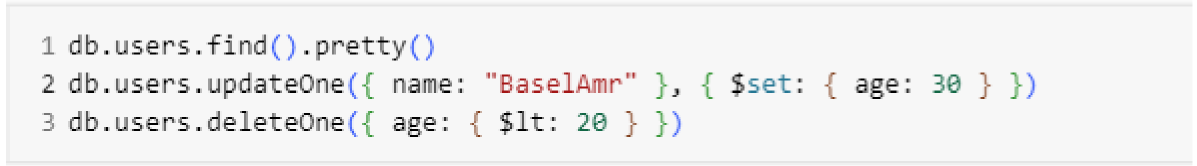
```
use testdb
db.createCollection('users')
```

1.2 Insert Sample Data



```
db.users.insertMany([
  { "name": "MostafaAmr", "email": "mostafaamr@gmail.com", "age": 38 },
  { "name": "AyaAmr", "email": "ayaamr@gmail.com", "age": 26 },
])
```

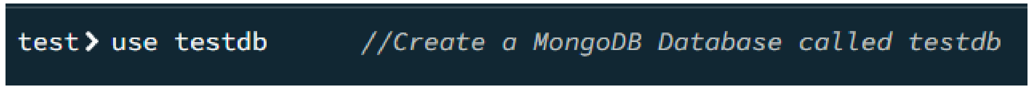
1.3 Read, Update and Delete Data



```
db.users.find().pretty()
db.users.updateOne({ name: "BaselAmr" }, { $set: { age: 30 } })
db.users.deleteOne({ age: { $lt: 20 } })
```

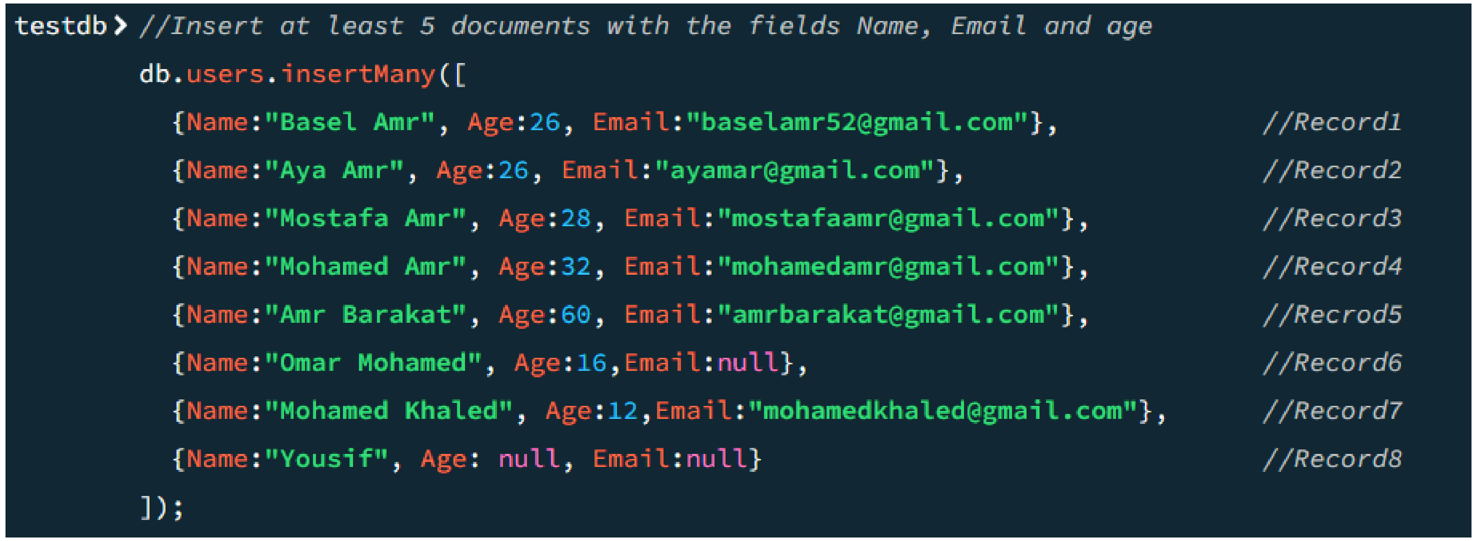
Sprints

1.1 Database and collection Creation

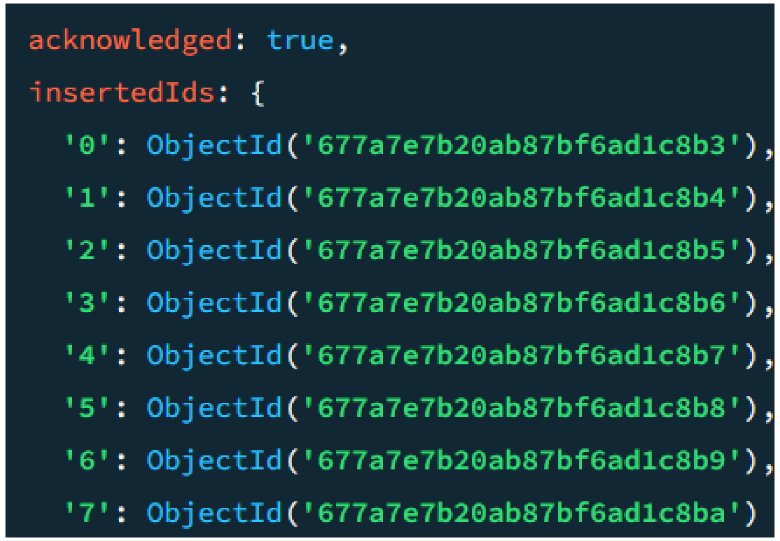


```
test> use testdb        //Create a MongoDB Database called testdb
```

1.2 Insert Sample Data

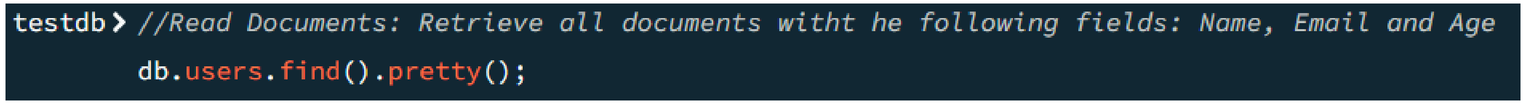


```
testdb> //Insert at least 5 documents with the fields Name, Email and age
        db.users.insertMany([
          {Name:"Basel Amr", Age:26, Email:"baselamr52@gmail.com"},              //Record1
          {Name:"Aya Amr", Age:26, Email:"ayamar@gmail.com"},                    //Record2
          {Name:"Mostafa Amr", Age:28, Email:"mostafaamr@gmail.com"},            //Record3
          {Name:"Mohamed Amr", Age:32, Email:"mohamedamr@gmail.com"},            //Record4
          {Name:"Amr Barakat", Age:60, Email:"amrbarakat@gmail.com"},            //Recrod5
          {Name:"Omar Mohamed", Age:16,Email:null},                              //Record6
          {Name:"Mohamed Khaled", Age:12,Email:"mohamedkhaled@gmail.com"},       //Record7
          {Name:"Yousif", Age: null, Email:null}                                 //Record8
        ]);
```

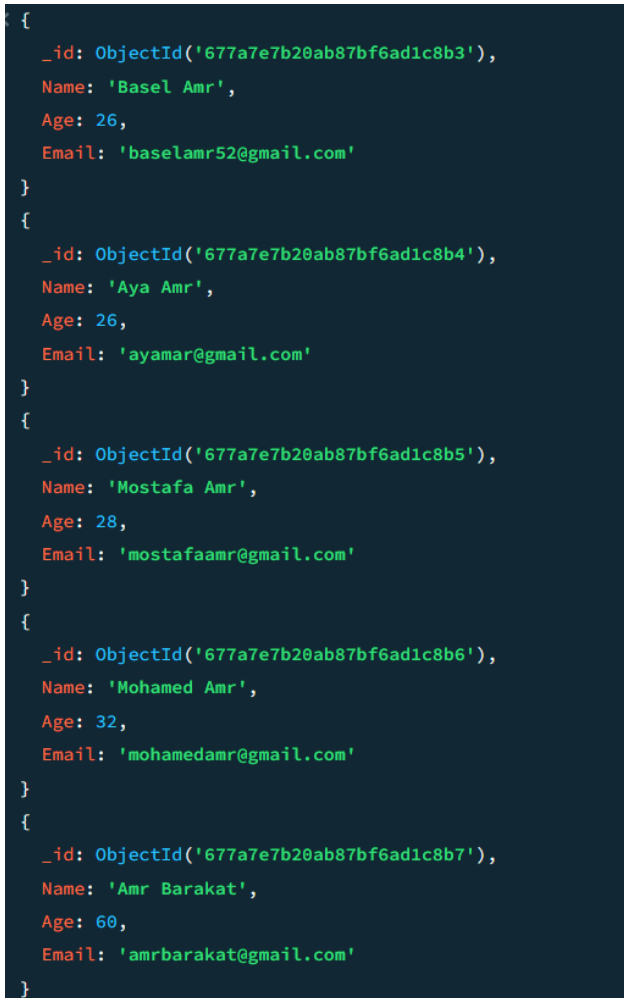


```
acknowledged: true,
insertedIds: {
  '0': ObjectId('677a7e7b20ab87bf6ad1c8b3'),
  '1': ObjectId('677a7e7b20ab87bf6ad1c8b4'),
  '2': ObjectId('677a7e7b20ab87bf6ad1c8b5'),
  '3': ObjectId('677a7e7b20ab87bf6ad1c8b6'),
  '4': ObjectId('677a7e7b20ab87bf6ad1c8b7'),
  '5': ObjectId('677a7e7b20ab87bf6ad1c8b8'),
  '6': ObjectId('677a7e7b20ab87bf6ad1c8b9'),
  '7': ObjectId('677a7e7b20ab87bf6ad1c8ba')
```

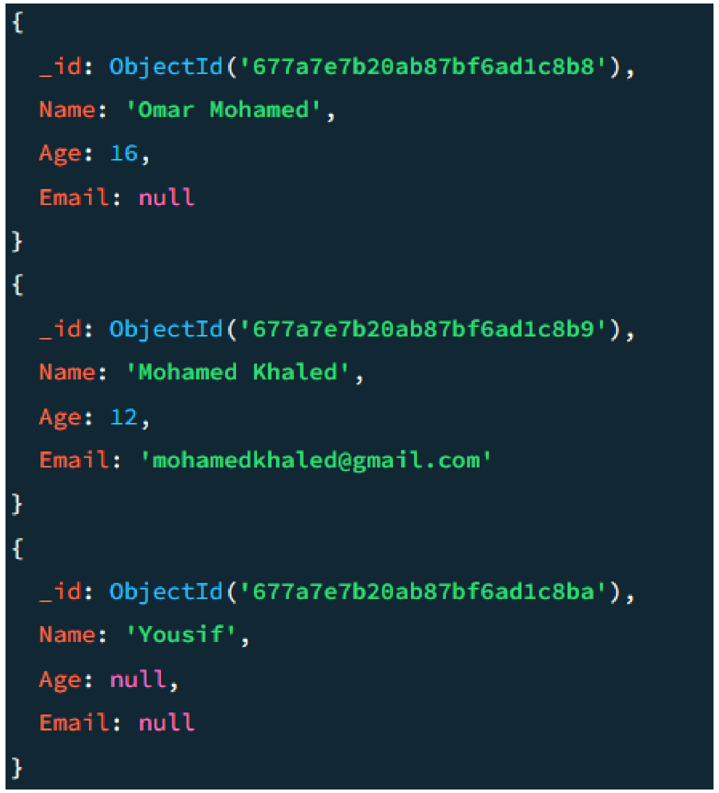
1.3 Retrieve all documents from the users' collections



```
testdb> //Read Documents: Retrieve all documents witht he following fields: Name, Email and Age
        db.users.find().pretty();
```

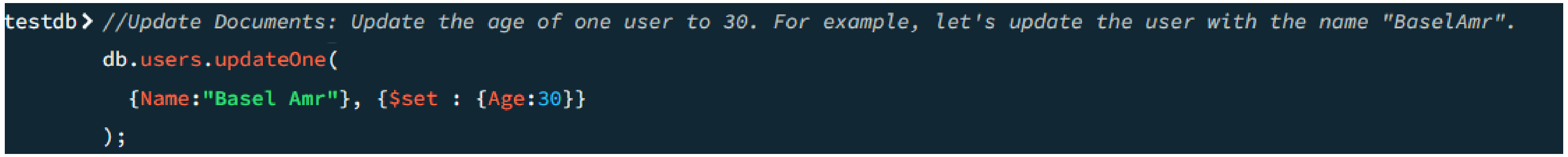


```
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8b3'),
  Name: 'Basel Amr',
  Age: 26,
  Email: 'baselamr52@gmail.com'
}
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8b4'),
  Name: 'Aya Amr',
  Age: 26,
  Email: 'ayamar@gmail.com'
}
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8b5'),
  Name: 'Mostafa Amr',
  Age: 28,
  Email: 'mostafaamr@gmail.com'
}
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8b6'),
  Name: 'Mohamed Amr',
  Age: 32,
  Email: 'mohamedamr@gmail.com'
}
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8b7'),
  Name: 'Amr Barakat',
  Age: 60,
  Email: 'amrbarakat@gmail.com'
}
```

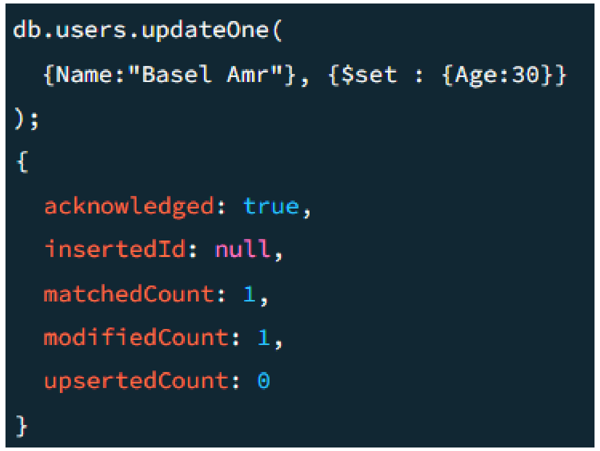


```
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8b8'),
  Name: 'Omar Mohamed',
  Age: 16,
  Email: null
}
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8b9'),
  Name: 'Mohamed Khaled',
  Age: 12,
  Email: 'mohamedkhaled@gmail.com'
}
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8ba'),
  Name: 'Yousif',
  Age: null,
  Email: null
}
```

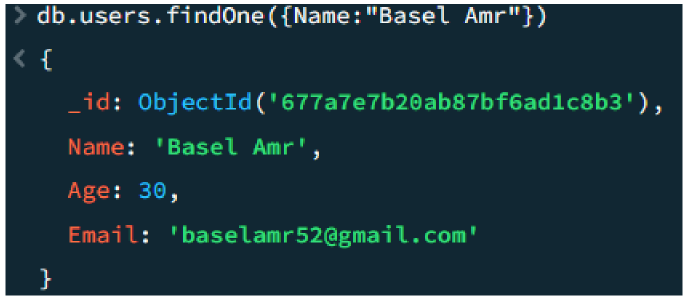
1.4 Update the age of one user to be 30



```
testdb> //Update Documents: Update the age of one user to 30. For example, let's update the user with the name "BaselAmr".
        db.users.updateOne(
          {Name:"Basel Amr"}, {$set : {Age:30}}
        );
```

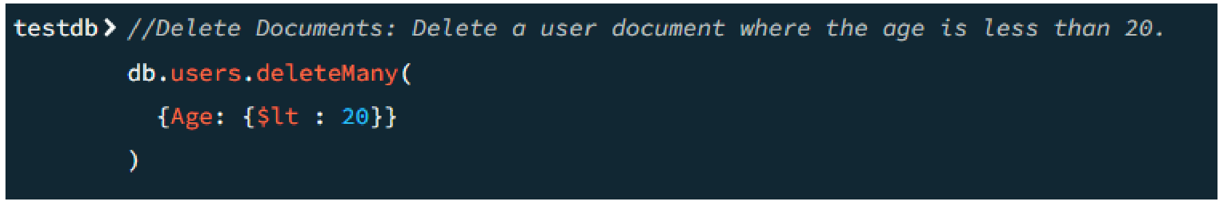


```
db.users.updateOne(
  {Name:"Basel Amr"}, {$set : {Age:30}}
);
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

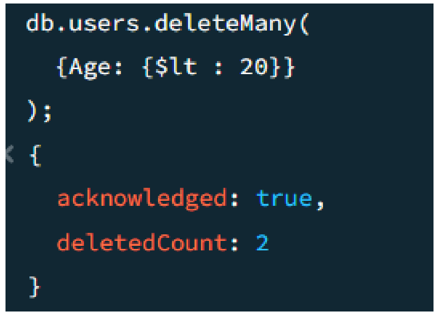


```
> db.users.findOne({Name:"Basel Amr"})
< {
    _id: ObjectId('677a7e7b20ab87bf6ad1c8b3'),
    Name: 'Basel Amr',
    Age: 30,
    Email: 'baselamr52@gmail.com'
  }
```

1.5 Delete all users based on a specific condition (age<20)



```
testdb> //Delete Documents: Delete a user document where the age is less than 20.
        db.users.deleteMany(
          {Age: {$lt : 20}}
        )
```

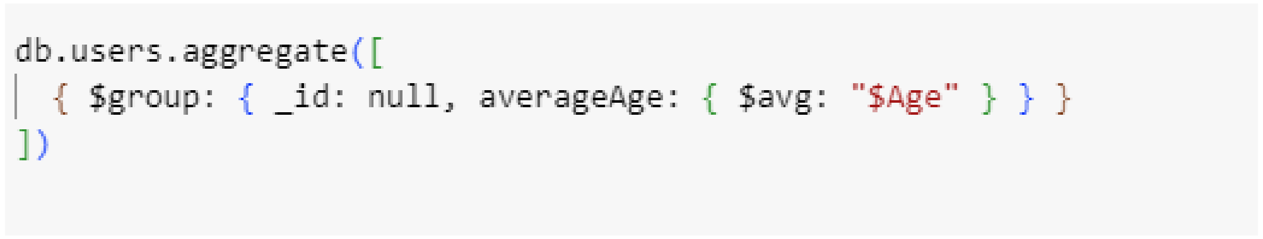


```
db.users.deleteMany(
  {Age: {$lt : 20}}
);
{
  acknowledged: true,
  deletedCount: 2
}
```

Requirement Two

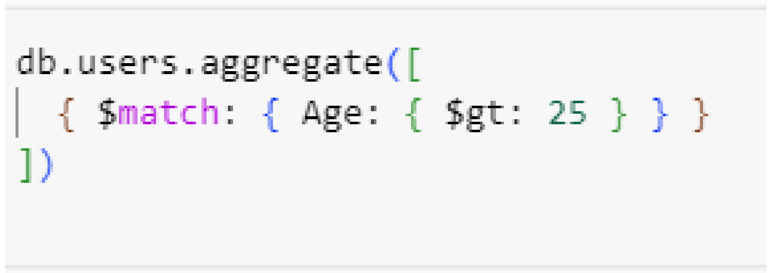
Aggregation Tasks

2.1 Use MongoDB's aggregation framework to calculate avgAge



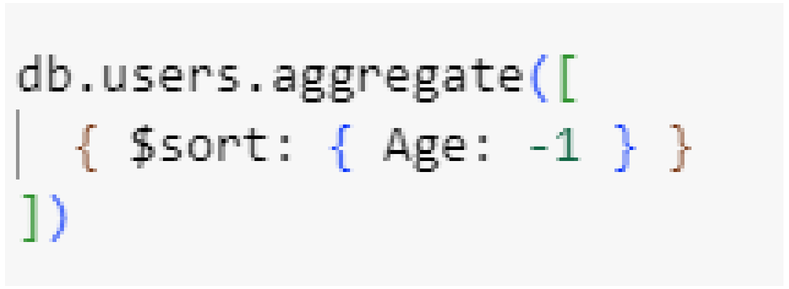
```
db.users.aggregate([
  { $group: { _id: null, averageAge: { $avg: "$Age" } } }
])
```

2.2 Use the aggregation framework to find users > 25 years old



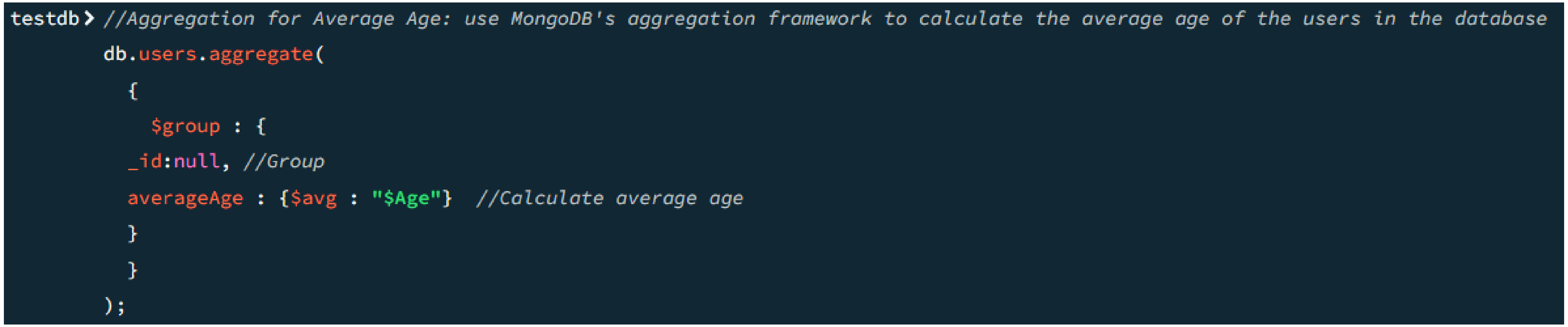
```
db.users.aggregate([
  { $match: { Age: { $gt: 25 } } }
])
```

2.3 Sort the users in descending order of their age

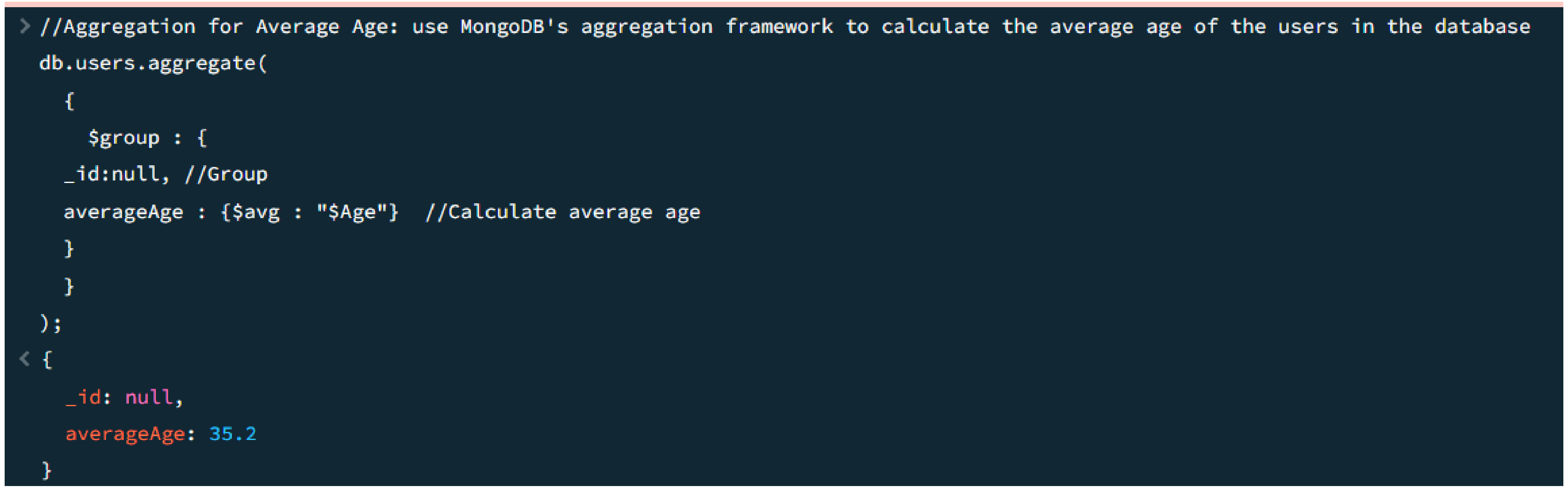


```
db.users.aggregate([
  { $sort: { Age: -1 } }
])
```

2.1 Use MongoDB’s aggregation framework to calculate avgAge



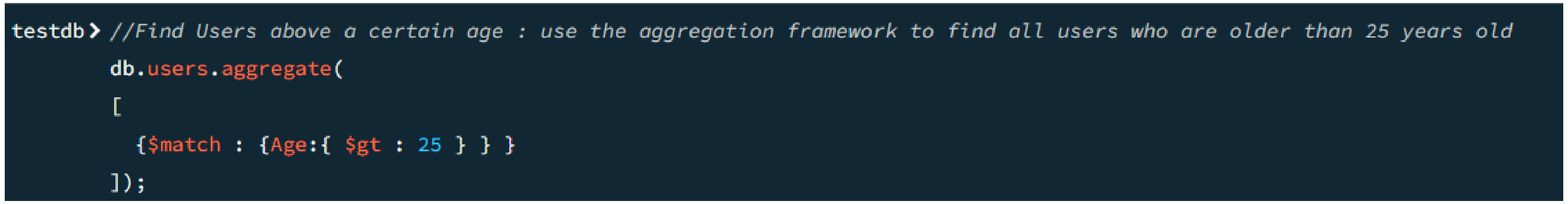
```
testdb> //Aggregation for Average Age: use MongoDB's aggregation framework to calculate the average age of the users in the database
        db.users.aggregate(
          {
            $group : {
          _id:null, //Group
          averageAge : {$avg : "$Age"}  //Calculate average age
          }
          }
        );
```



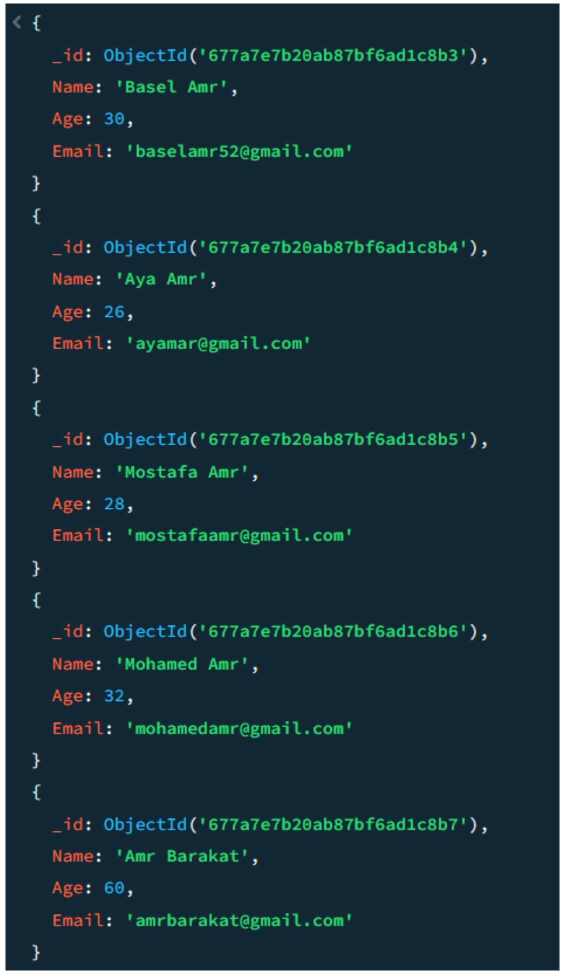
```
> //Aggregation for Average Age: use MongoDB's aggregation framework to calculate the average age of the users in the database
  db.users.aggregate(
    {
      $group : {
    _id:null, //Group
    averageAge : {$avg : "$Age"}  //Calculate average age
    }
    }
  );
< {
    _id: null,
    averageAge: 35.2
  }
```

Sprints

2.2 Use the aggregation framework to find users > 25 years old



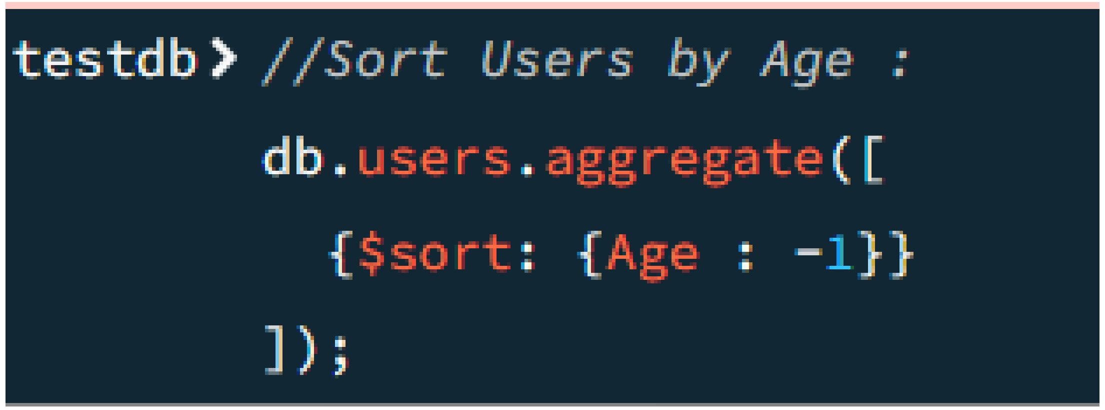
```
testdb> //Find Users above a certain age : use the aggregation framework to find all users who are older than 25 years old
        db.users.aggregate(
        [
          {$match : {Age:{ $gt : 25 } } }
        ]);
```



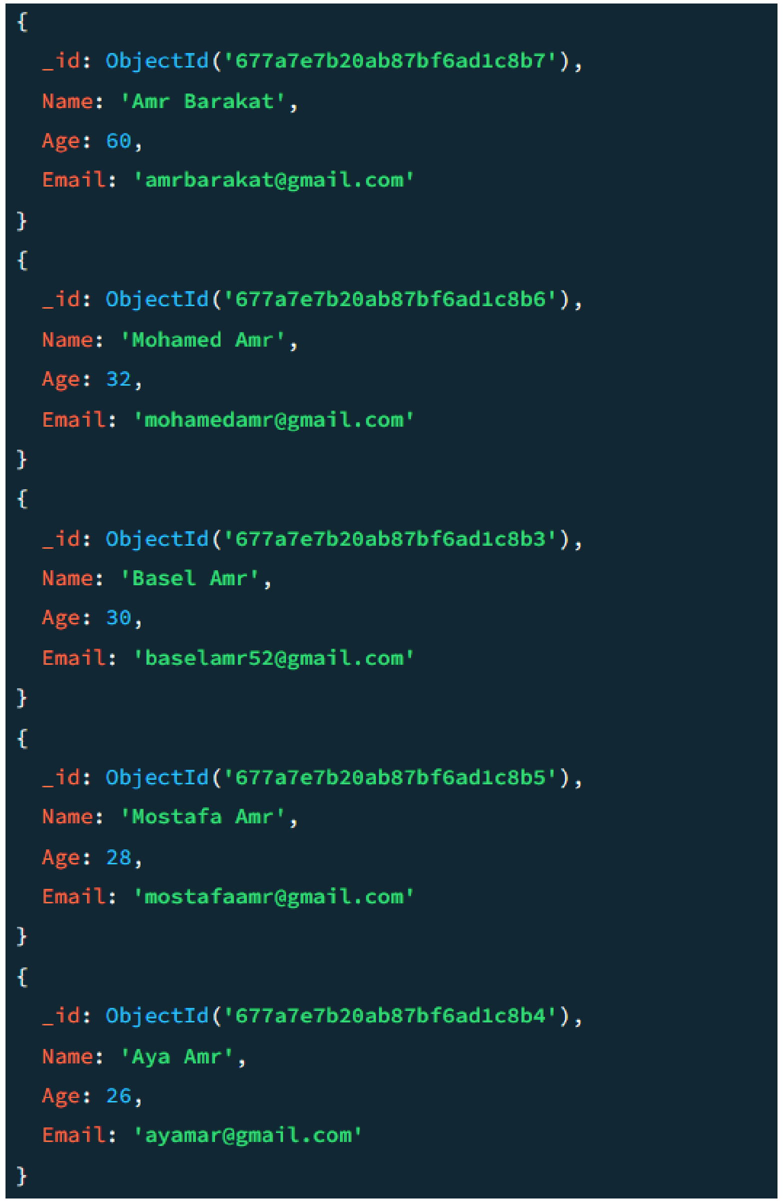
```
< {
    _id: ObjectId('677a7e7b20ab87bf6ad1c8b3'),
    Name: 'Basel Amr',
    Age: 30,
    Email: 'baselamr52@gmail.com'
  }
  {
    _id: ObjectId('677a7e7b20ab87bf6ad1c8b4'),
    Name: 'Aya Amr',
    Age: 26,
    Email: 'ayamar@gmail.com'
  }
  {
    _id: ObjectId('677a7e7b20ab87bf6ad1c8b5'),
    Name: 'Mostafa Amr',
    Age: 28,
    Email: 'mostafaamr@gmail.com'
  }
  {
    _id: ObjectId('677a7e7b20ab87bf6ad1c8b6'),
    Name: 'Mohamed Amr',
    Age: 32,
    Email: 'mohamedamr@gmail.com'
  }
  {
    _id: ObjectId('677a7e7b20ab87bf6ad1c8b7'),
    Name: 'Amr Barakat',
    Age: 60,
    Email: 'amrbarakat@gmail.com'
  }
```

Sprints

2.3 Sort the users in descending order of their age



```
testdb> //Sort Users by Age :
        db.users.aggregate([
          {$sort: {Age : -1}}
        ]);
```



```
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8b7'),
  Name: 'Amr Barakat',
  Age: 60,
  Email: 'amrbarakat@gmail.com'
}
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8b6'),
  Name: 'Mohamed Amr',
  Age: 32,
  Email: 'mohamedamr@gmail.com'
}
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8b3'),
  Name: 'Basel Amr',
  Age: 30,
  Email: 'baselamr52@gmail.com'
}
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8b5'),
  Name: 'Mostafa Amr',
  Age: 28,
  Email: 'mostafaamr@gmail.com'
}
{
  _id: ObjectId('677a7e7b20ab87bf6ad1c8b4'),
  Name: 'Aya Amr',
  Age: 26,
  Email: 'ayamar@gmail.com'
}
```

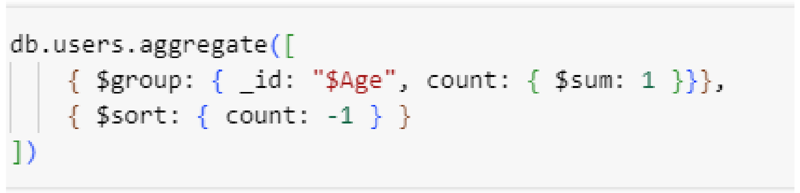
Sprints

Requirement Three

Data Insights and Analysis

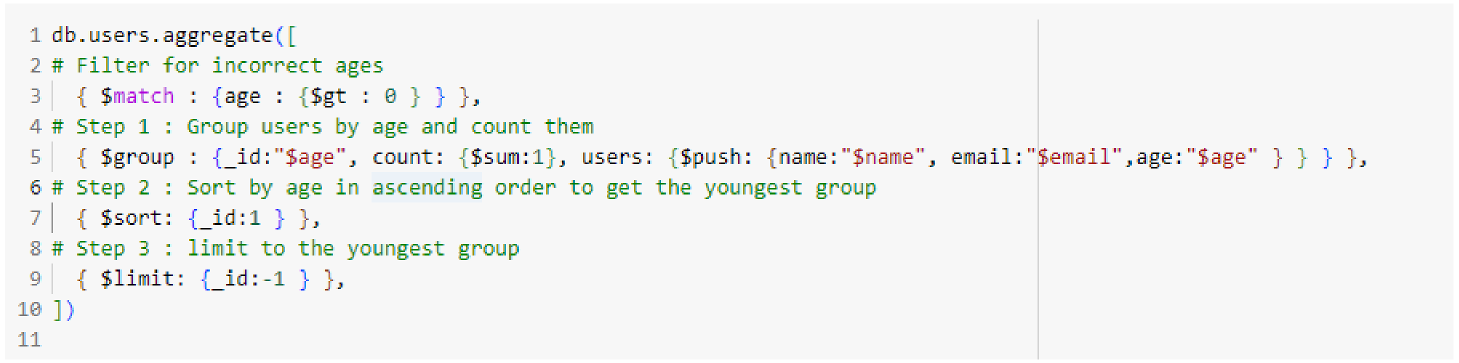
Sprints

3.1 Group the users by age and count how many users in each age group



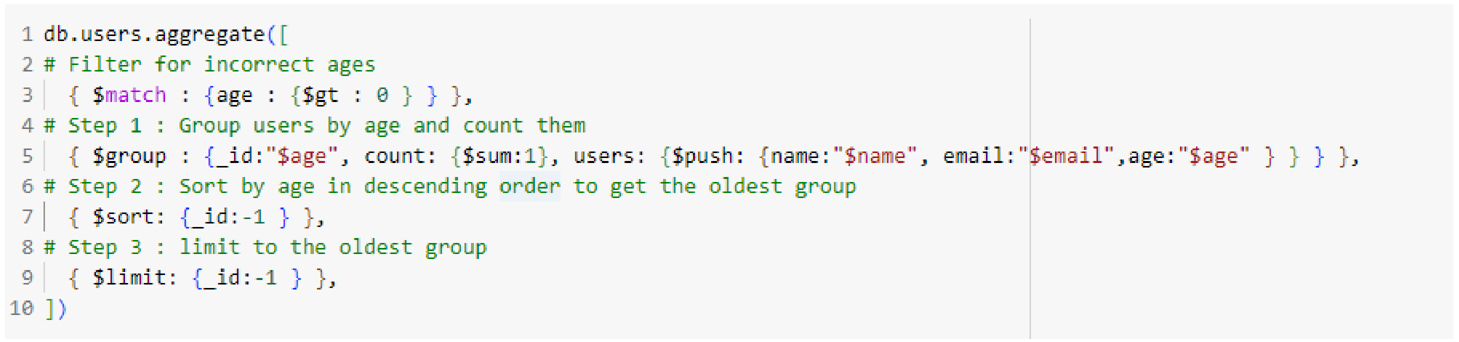
```
db.users.aggregate([
    { $group: { _id: "$Age", count: { $sum: 1 }}},
    { $sort: { count: -1 } }
])
```

3.2 Combine multiple aggregation stages (like $match, $group and $sort) to get more insightful analysis



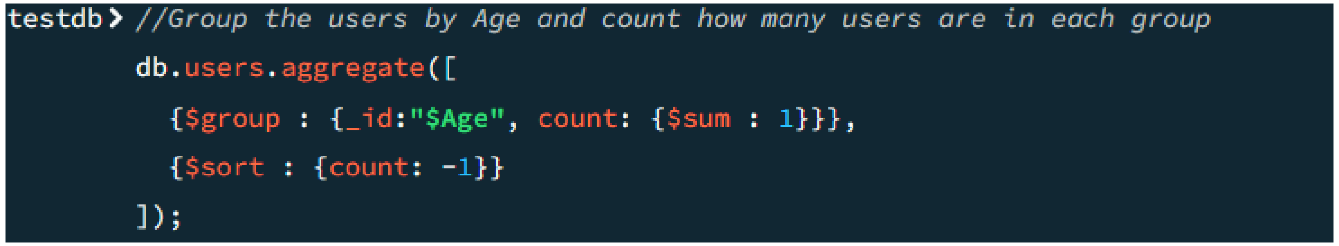
```
db.users.aggregate([
# Filter for incorrect ages
  { $match : {age : {$gt : 0 } } },
# Step 1 : Group users by age and count them
  { $group : {_id:"$age", count: {$sum:1}, users: {$push: {name:"$name", email:"$email",age:"$age" } } } },
# Step 2 : Sort by age in ascending order to get the youngest group
  { $sort: {_id:1 } },
# Step 3 : limit to the youngest group
  { $limit: {_id:-1 } },
])

```

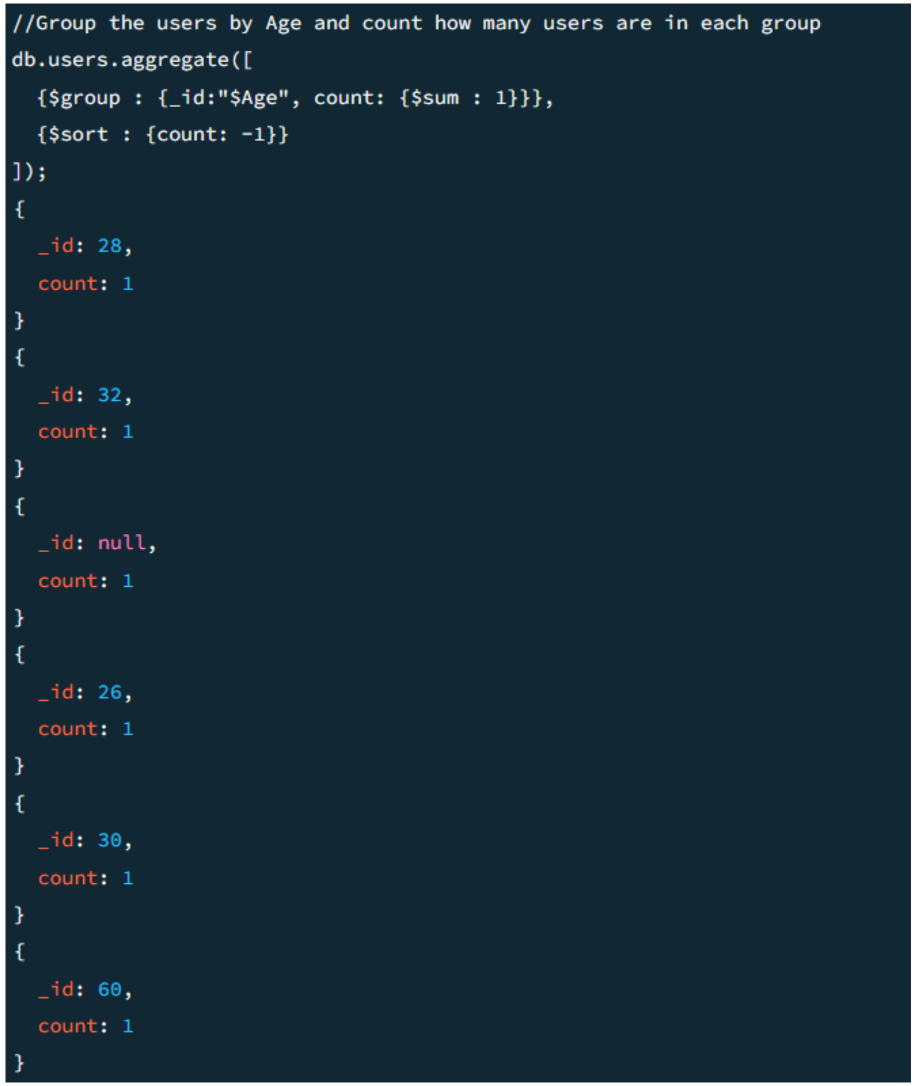


```
db.users.aggregate([
# Filter for incorrect ages
  { $match : {age : {$gt : 0 } } },
# Step 1 : Group users by age and count them
  { $group : {_id:"$age", count: {$sum:1}, users: {$push: {name:"$name", email:"$email",age:"$age" } } } },
# Step 2 : Sort by age in descending order to get the oldest group
  { $sort: {_id:-1 } },
# Step 3 : limit to the oldest group
  { $limit: {_id:-1 } },
])
```

3.1 Group the users by age and count how many users in each age group



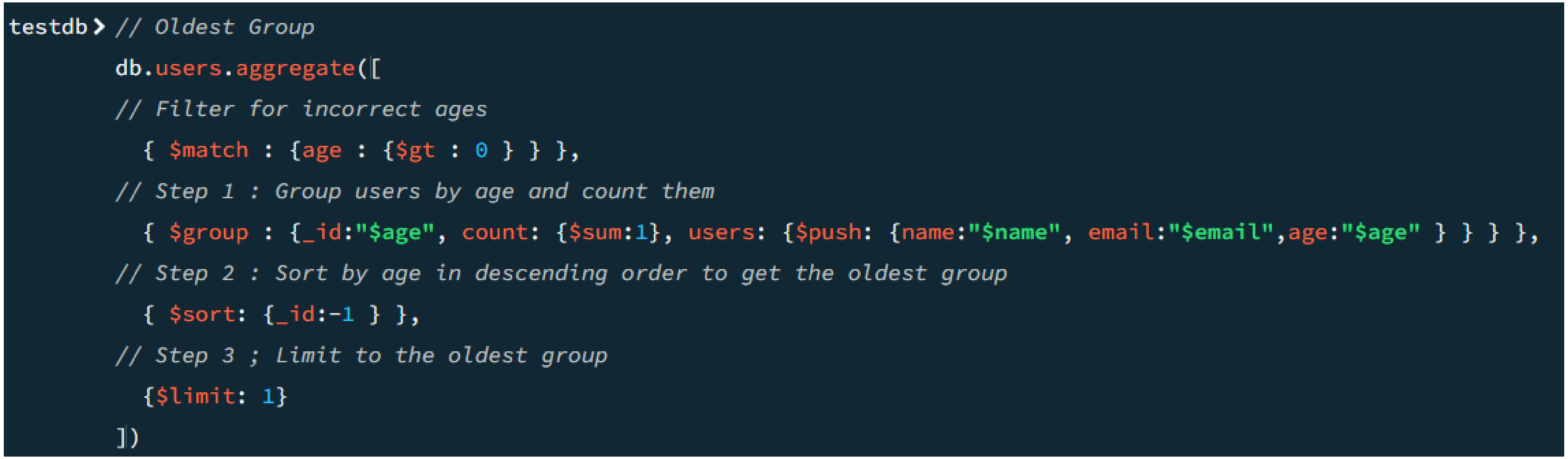
```
testdb> //Group the users by Age and count how many users are in each group
        db.users.aggregate([
          {$group : {_id:"$Age", count: {$sum : 1}}},
          {$sort : {count: -1}}
        ]);
```



```
//Group the users by Age and count how many users are in each group
db.users.aggregate([
  {$group : {_id:"$Age", count: {$sum : 1}}},
  {$sort : {count: -1}}
]);
{
  _id: 28,
  count: 1
}
{
  _id: 32,
  count: 1
}
{
  _id: null,
  count: 1
}
{
  _id: 26,
  count: 1
}
{
  _id: 30,
  count: 1
}
{
  _id: 60,
  count: 1
}
```

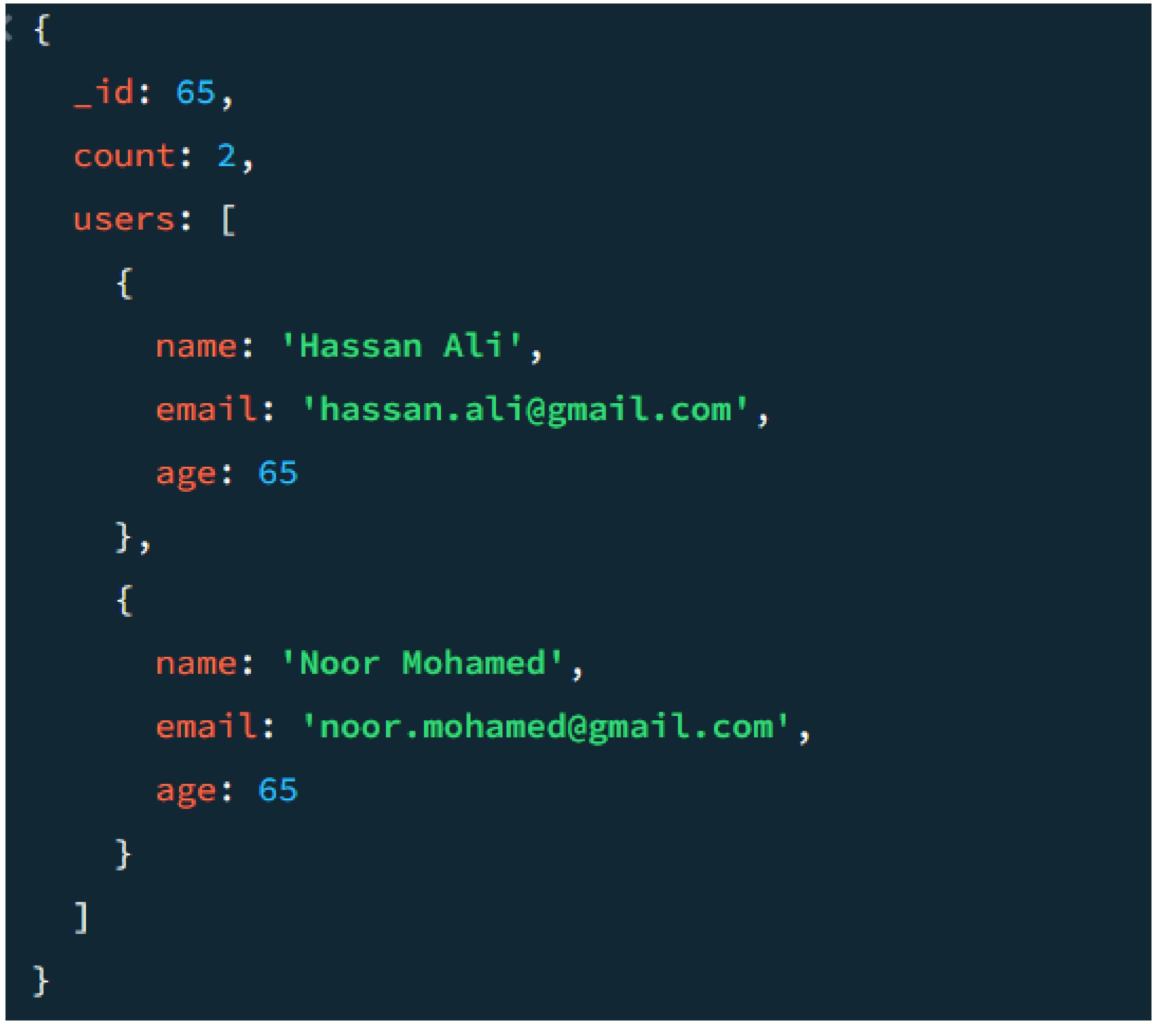
Sprints

3.2 Combine multiple aggregation stages (like $match, $group and $sort) to get more insightful analysis



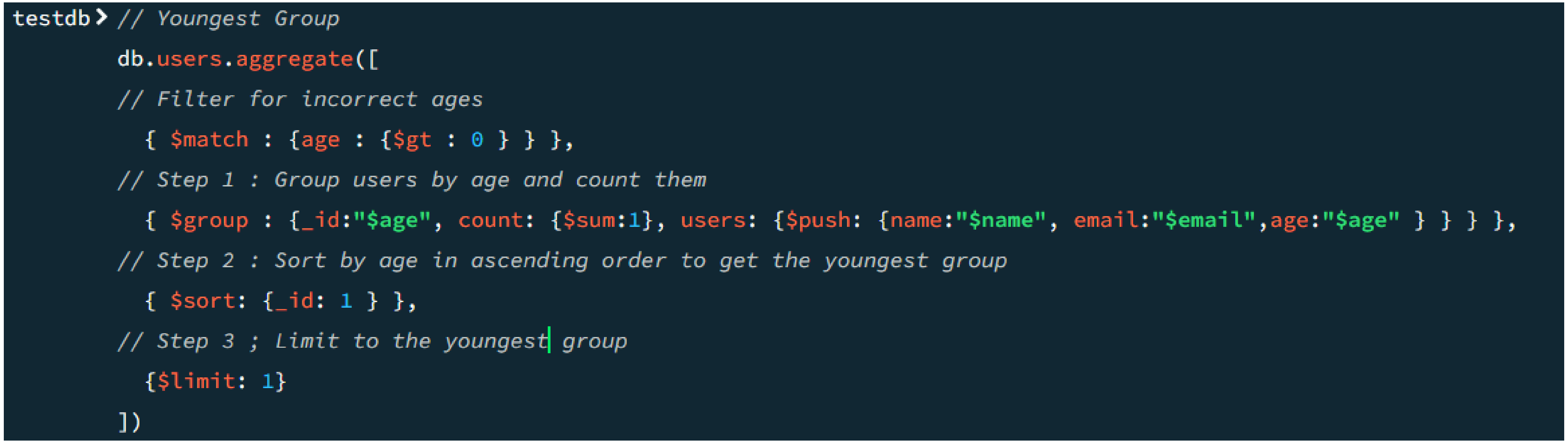
```
testdb> // Oldest Group
        db.users.aggregate([
        // Filter for incorrect ages
          { $match : {age : {$gt : 0 } } },
        // Step 1 : Group users by age and count them
          { $group : {_id:"$age", count: {$sum:1}, users: {$push: {name:"$name", email:"$email",age:"$age" } } } },
        // Step 2 : Sort by age in descending order to get the oldest group
          { $sort: {_id:-1 } },
        // Step 3 ; Limit to the oldest group
          {$limit: 1}
        ])
```

3.2 Combine multiple aggregation stages (like $match, $group and $sort) to get more insightful analysis

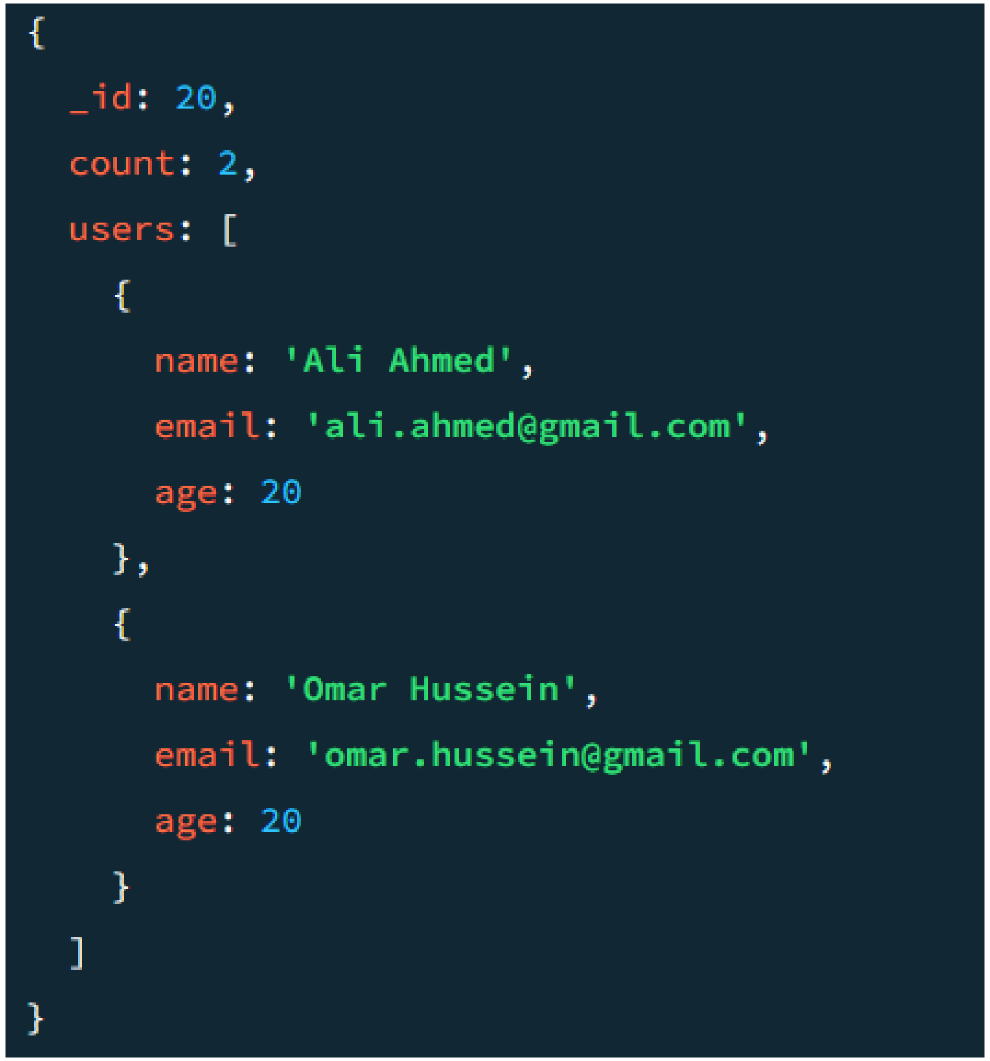


```
{
  _id: 65,
  count: 2,
  users: [
    {
      name: 'Hassan Ali',
      email: 'hassan.ali@gmail.com',
      age: 65
    },
    {
      name: 'Noor Mohamed',
      email: 'noor.mohamed@gmail.com',
      age: 65
    }
  ]
}
```

3.2 Combine multiple aggregation stages (like $match, $group and $sort) to get more insightful analysis


```
testdb> // Youngest Group
        db.users.aggregate([
        // Filter for incorrect ages
          { $match : {age : {$gt : 0 } } },
        // Step 1 : Group users by age and count them
          { $group : {_id:"$age", count: {$sum:1}, users: {$push: {name:"$name", email:"$email",age:"$age" } } } },
        // Step 2 : Sort by age in ascending order to get the youngest group
          { $sort: {_id: 1 } },
        // Step 3 ; Limit to the youngest group
          {$limit: 1}
        ])
```

3.2 Combine multiple aggregation stages (like $match, $group and $sort) to get more insightful analysis



```
{
  _id: 20,
  count: 2,
  users: [
    {
      name: 'Ali Ahmed',
      email: 'ali.ahmed@gmail.com',
      age: 20
    },
    {
      name: 'Omar Hussein',
      email: 'omar.hussein@gmail.com',
      age: 20
    }
  ]
}
```