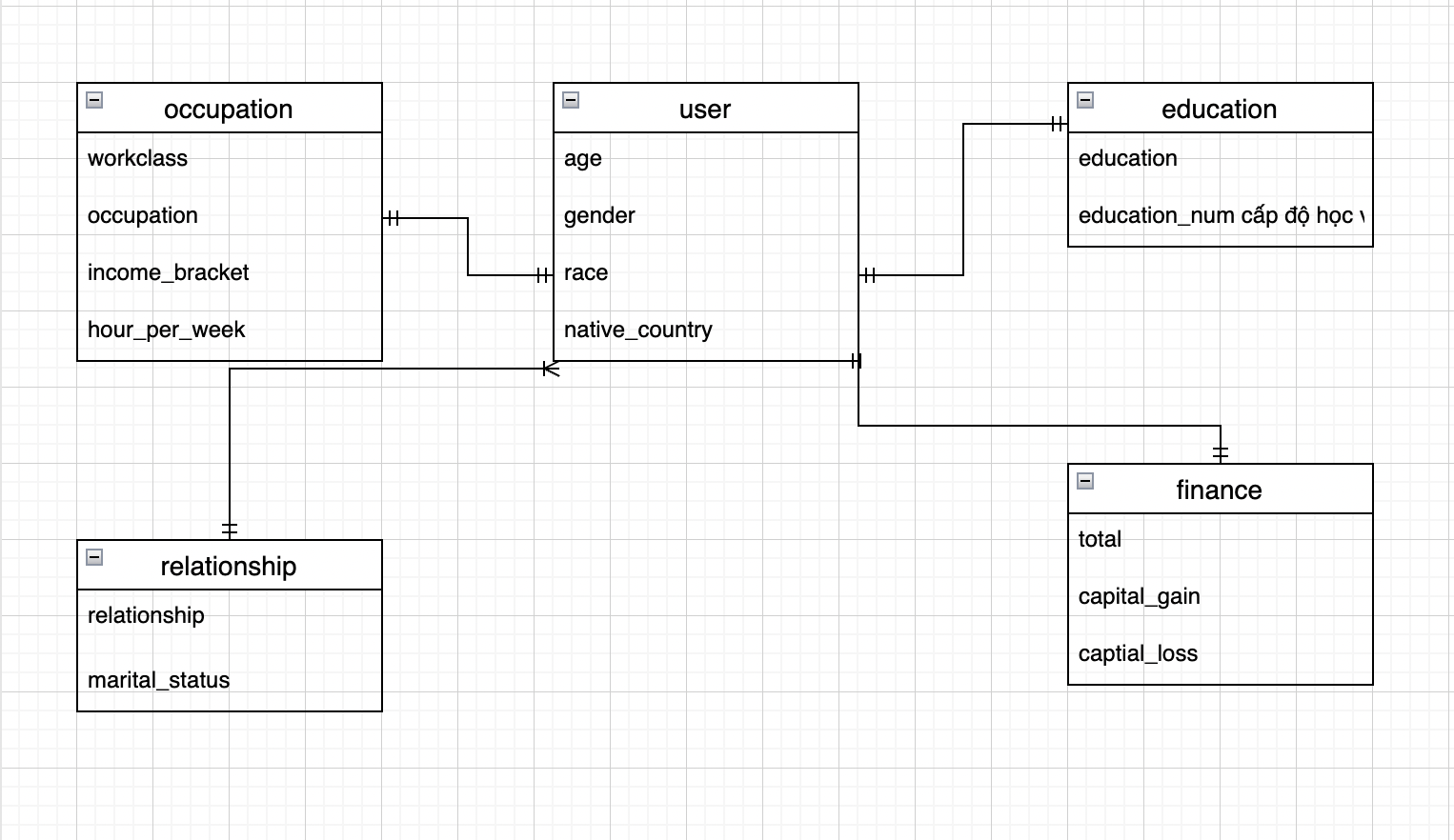
**XÂY DỰNG CƠ SỞ DỮ LIỆU MONGODB**

**NGUYENNNFX10238@FUNIX.EDU.VN**

1. **ERD Diagram**

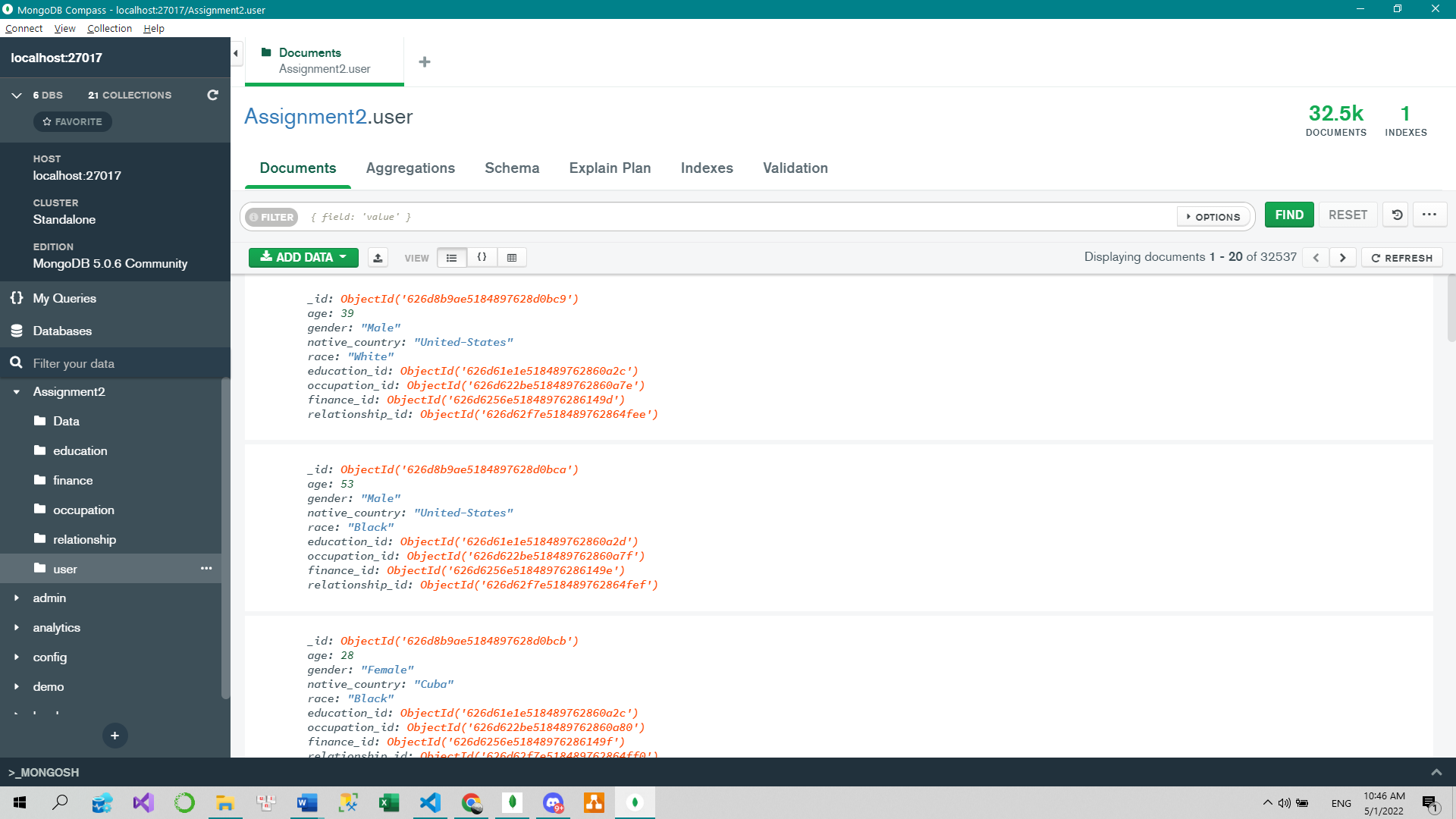


Ta extract data từ User Collection gốc thành 5 Collection con có liên kết với nhau, mỗi Collection sẽ gồm các trường đặc trưng như ERD ở trên:

* Collection **user**: chứa các thông tin cơ bản như tuổi, giới tính, chủng tộc, quốc tịch.
* Collection **education** chứa các thông tin liên quan đến học vấn.
* Collection **occupation** chứa các thông tin liên quan về nghề nghiệp,
* Collection **relationship** chứa các thông tin về tình trạng hôn nhân.
* Collection **finance** chứa các thông tin liên quan đến vấn đề tài chính.

1. **Giải thích mối quan hệ giữa các Collection và cấu trúc của các Document**

Cấu trúc của một Doccument sau khi hoàn thành chỉnh sửa sẽ như hình dưới đây:



**User Collection sẽ lưu các thông tin cơ bản của từng người trong từng Doccument về:**

**\_id**: ID của từng cá nhân

**Age**: Tuổi

**Gender**: giới tính

**Native\_country:** Quốc tịch

**Race**: chủng tộc

**Education\_id**: Data sẽ được liên kết với Education Colletion bằng khoá Object\_id

Lưu trữ các thông tin về Education như education, education\_num

**Occupation\_id**: liên kết với Occupation Collection bằng khoá Object\_id.

Lưu trữ các thông tin gồm các trường con workclass, occupation, hours\_per\_week, income\_bracket

**Finance\_id:** Lưu trữ Object\_Id liên kết với Finance Collection chứa các thông tin về total, capital\_gian,capital\_loss

**Relationship\_id:** Liên kết với Relationship Collection bằng khoá Object\_id

Lưu trữ các thông tin vể Relationship như relationship, martial\_status

1. **Các câu lệnh để tạo Database**

Đầu tiên ta dùng câu lệnh **mongoimport --type csv -d Assignment2 -c Data –headerline --<url file csv>** để load dữ liệu từ file csv vào Collection Data.

Sau đó ta lần lượt dùng các Function bên dưới để lấy dữ liệu từ Collection tổng Data thêm vào các collection con như **education, finance, occupation, relationship**

* 1. **Load dữ liệu từ Data Collection vào Education Collection**

function loadEducation() {

const bulkInsert = db.education.initializeUnorderedBulkOp();

// Get all Documents in 'Data' Collection

const documents = db.Data.find({});

// Process each document

documents.forEach(function (doc) {

const element = {

education: doc.education,

education\_num: doc.education\_num

};

// Upsert into education Document

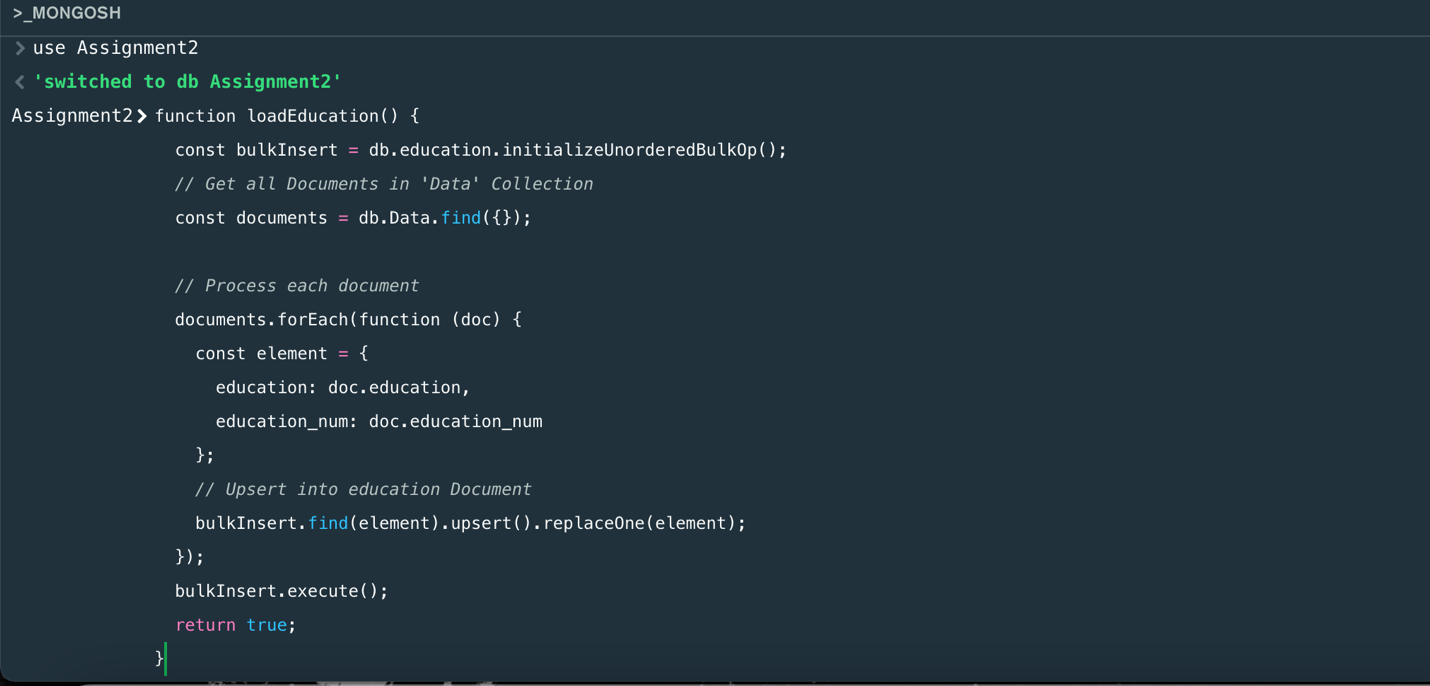
bulkInsert.find(element).upsert().replaceOne(element);

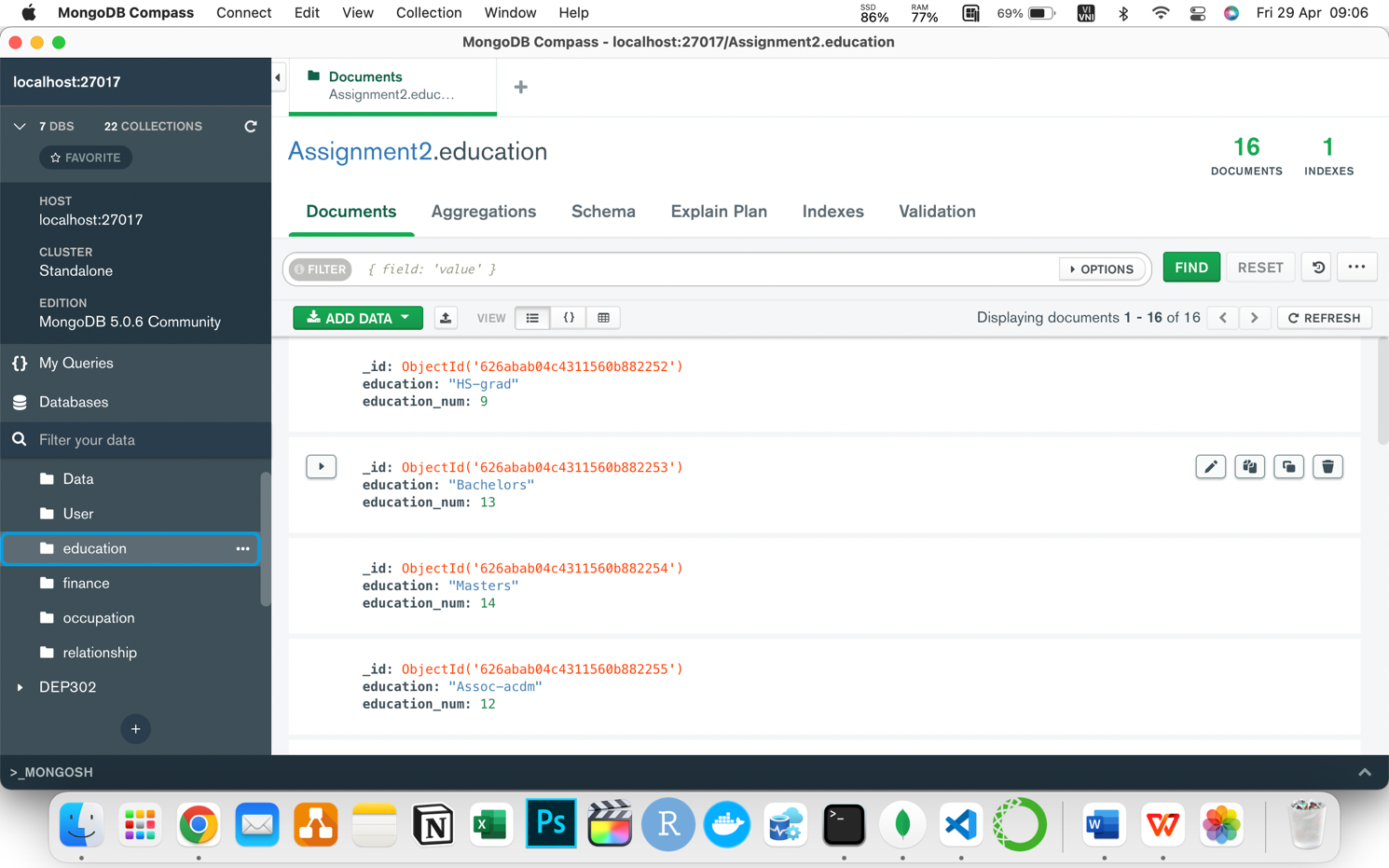
});

bulkInsert.execute();

return true;

}





* 1. **Load Occupation Collection**

function loadOccupation(){

const bulkInsert = db.occupation.initializeUnorderedBulkOp();

// Load all Data from Data Collection to Doccument

const doccuments = db.Data.find({});

// Process each doccument

doccuments.forEach(function(doc){

const element = {

workclass : doc.workclass,

occupation : doc.occupation,

hours\_per\_week: doc.hours\_per\_week,

income\_bracket: doc.income\_bracket

};

// Upsert each doc to Occupation Collection

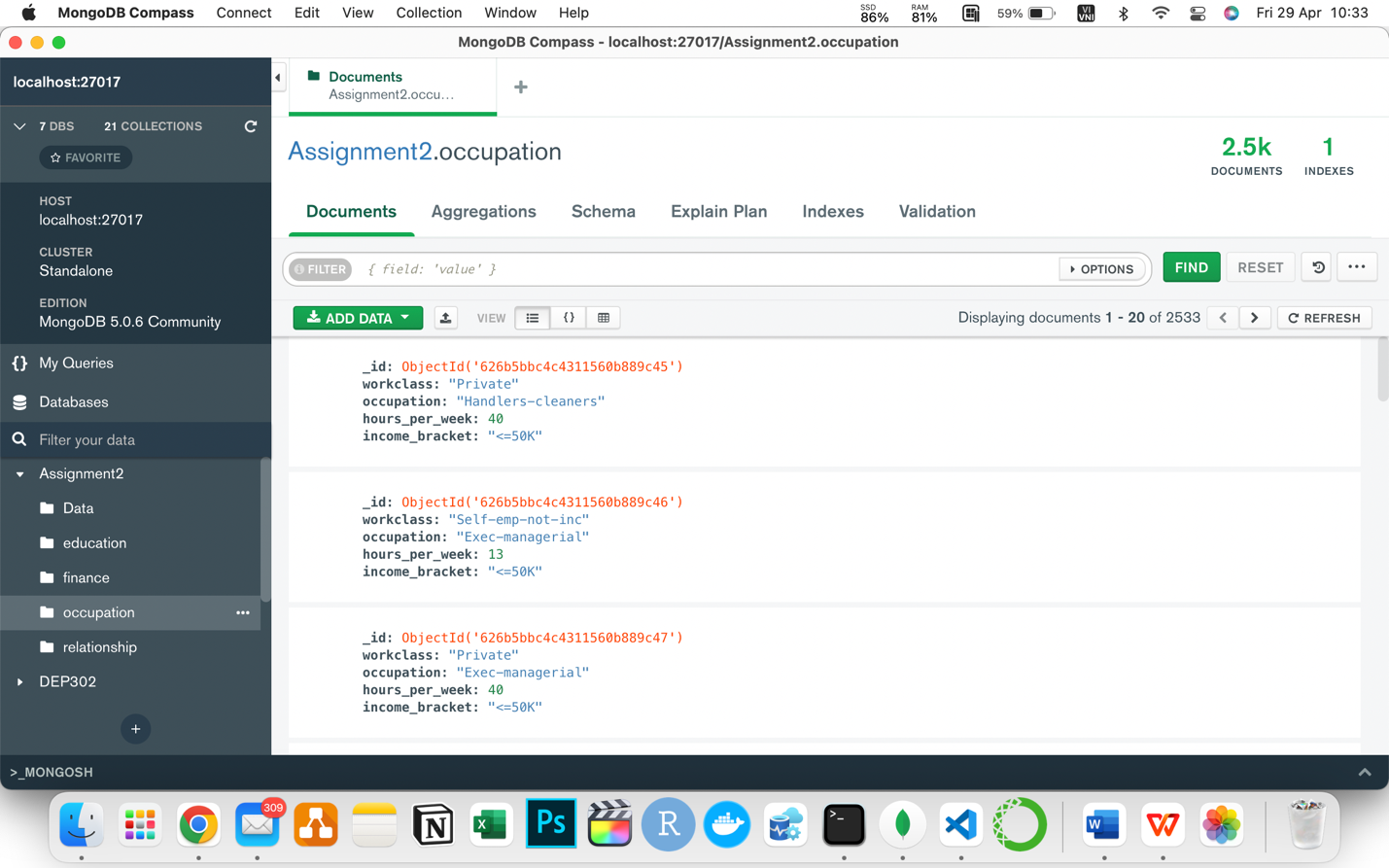
bulkInsert.find(element).upsert().replaceOne(element);

});

bulkInsert.execute();

return true;

};



* 1. **Load Finance Collection**

function loadFinance(){

const bulkInsert = db.finance.initializeUnorderedBulkOp();

// Load all Data from Data Collection to doccuments variable

const doccuments = db.Data.find({});

//Process each doccument

doccuments.forEach(function(doc){

const element = {

total: doc.total,

capital\_gain: doc.capital\_gain,

capital\_loss: doc.capital\_loss

};

// Upsert each doccument to Finance Collection

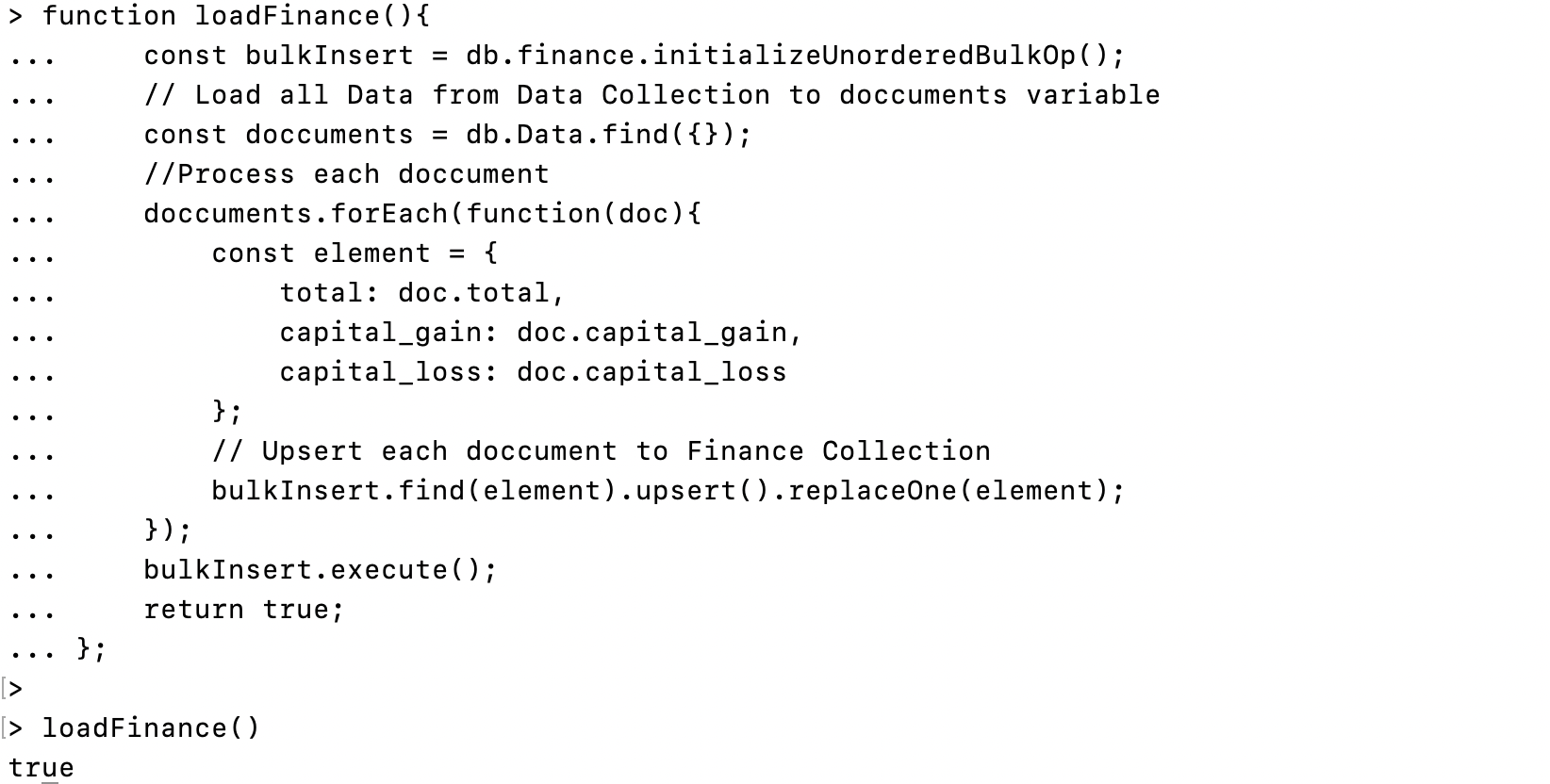
bulkInsert.find(element).upsert().replaceOne(element);

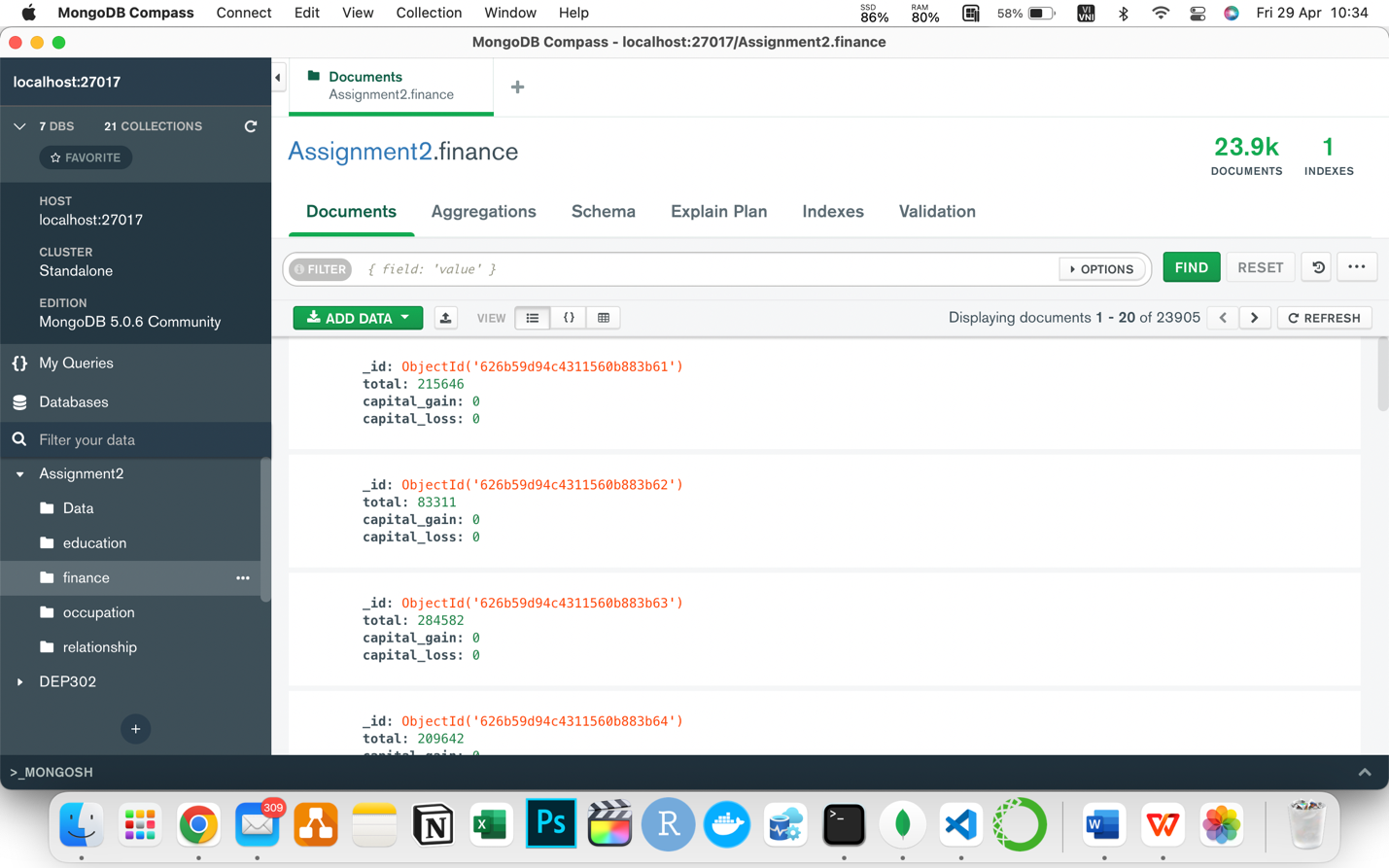
});

bulkInsert.execute();

return true;

};

****

****

* 1. **Load Relationship Collection**

function loadRelationship(){

const bulkInsert = db.relationship.initializeUnorderedBulkOp();

// Load all Data from Data Collection to Doccument

const doccuments = db.Data.find({});

// Process each doccument

doccuments.forEach(function(doc){

const element = {

relationship : doc.relationship,

marital\_status : doc.marital\_status

};

// Upsert each doc to Occupation Collection

bulkInsert.find(element).upsert().replaceOne(element);

});

bulkInsert.execute();

return true;

};

* 1. **Load User Collection**

function loadUser() {

const bulkInsert = db.user.initializeUnorderedBulkOp();

// Get all Documents in Data Collection

const documents = db.Data.find({});

// Process each document

documents.forEach(function (doc) {

const element = {

age: doc.age,

gender: doc.gender,

native\_country: doc.native\_country,

race: doc.race,

};

//Get education PK

// Find the education\_doc and data\_doc(element) load its \_id to each element of User Collection

const education = db.education.findOne({

education : doc.education,

education\_num:doc.education\_num

});

element.education\_id = education.\_id

// Get occupation PK

const occupation = db.occupation.findOne({

occupation: doc.occupation,

workclass: doc.workclass,

hours\_per\_week: doc.hours\_per\_week,

income\_bracket: doc.income\_bracket

});

element.occupation\_id = occupation.\_id;

// Get finance PK

const finance = db.finance.findOne({

total: doc.total,

capital\_gain: doc.capital\_gain,

capital\_loss: doc.capital\_loss,

});

element.finance\_id = finance.\_id;

// Get relationship PK

const relationship = db.relationship.findOne({

relationship: doc.relationship,

marital\_status: doc.marital\_status

});

element.relationship\_id = relationship.\_id;

// Upsert into user collection

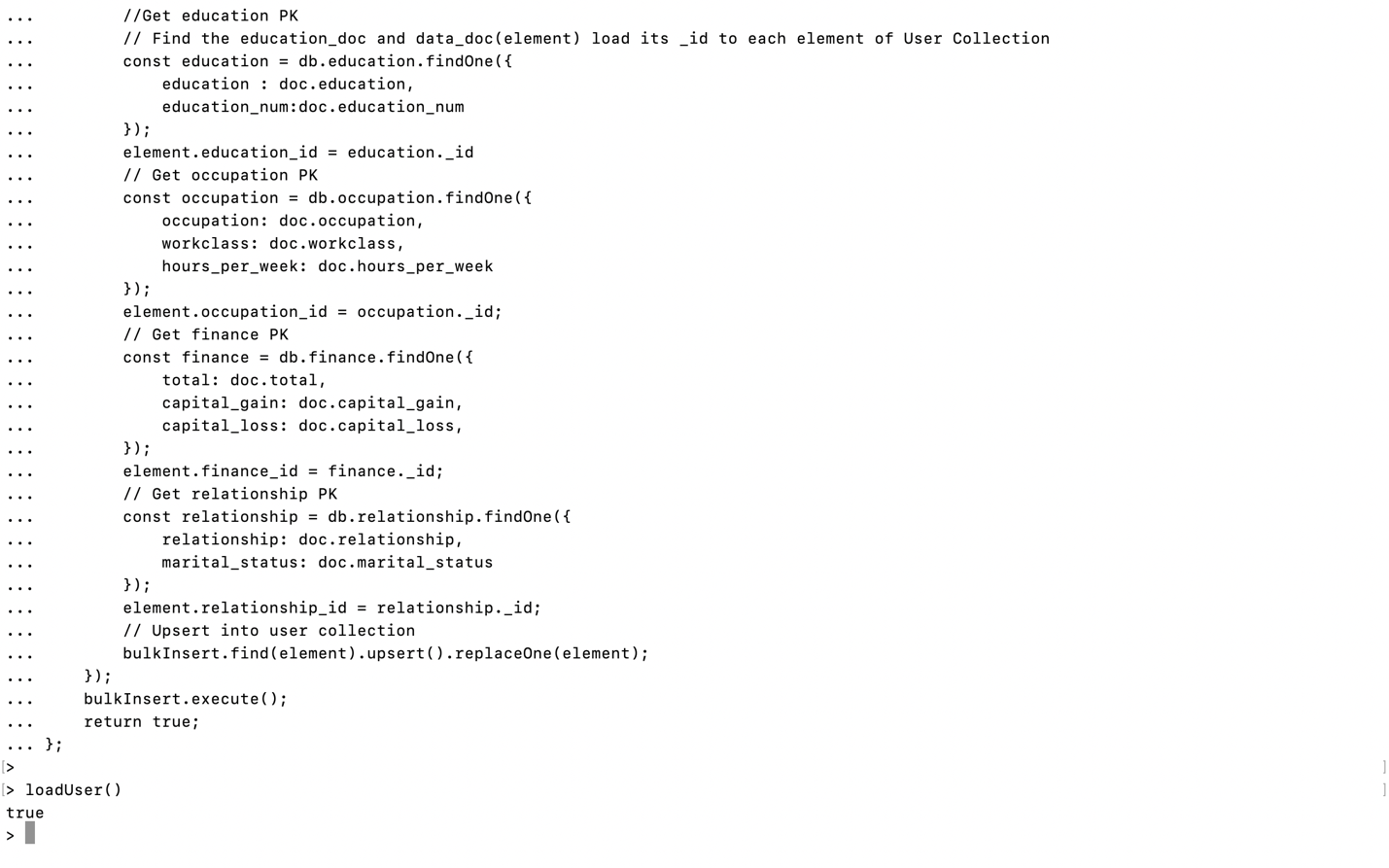
bulkInsert.find(element).upsert().replaceOne(element);

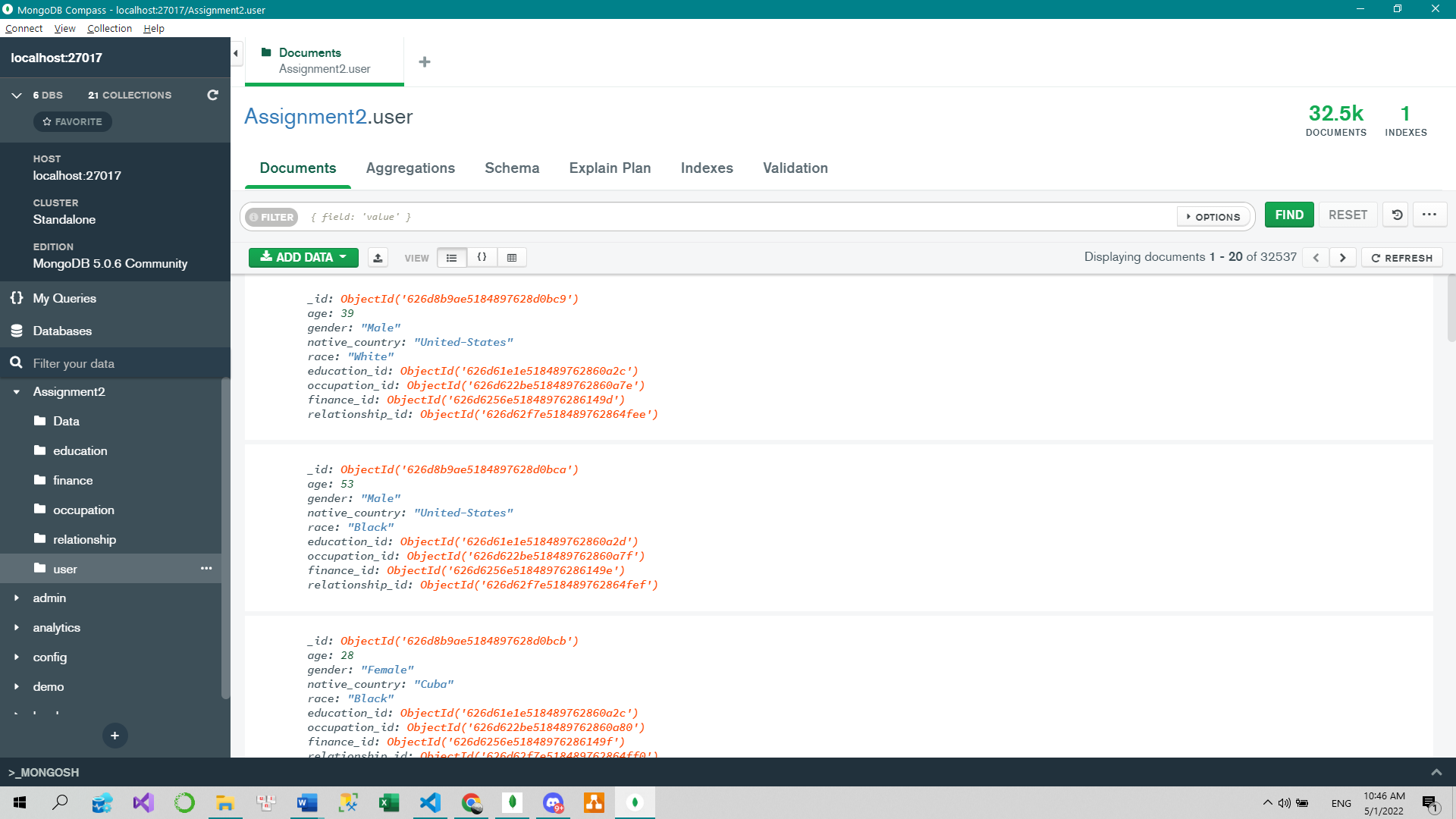
});

bulkInsert.execute();

return true;

};





1. **Liệu kê các Business Query**
2. **Có bao nhiêu người là Nữ và làm việc nhiều hơn 30 tiếng / tuần ?**

db.user.aggregate([

    {

        $lookup: {

            from: 'occupation',

            localField: 'occupation\_id',

            foreignField: '\_id',

            as: 'output'

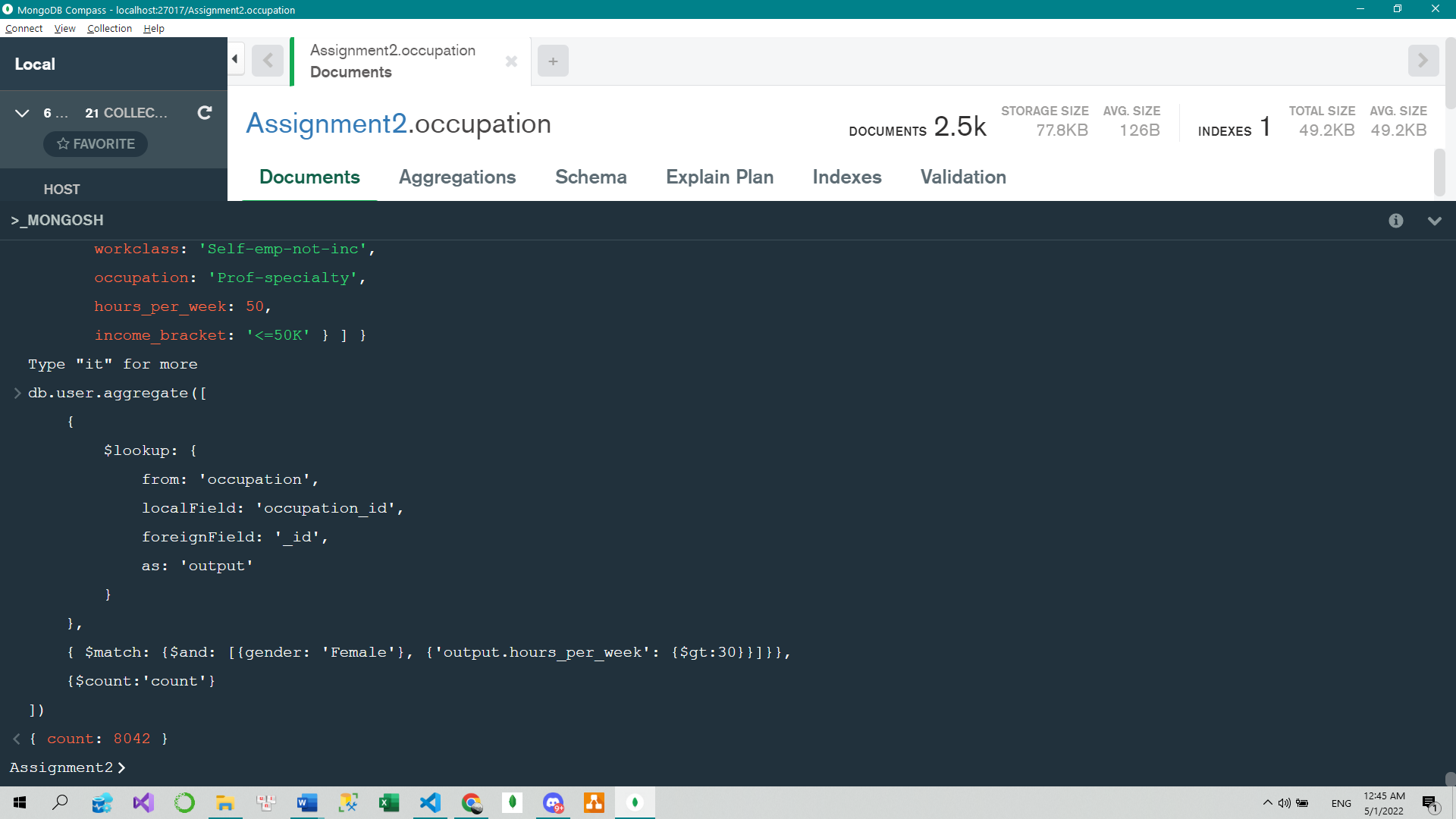
        }

    },

    { $match: {$and: [{gender: 'Female'}, {'output.hours\_per\_week': {$gt:30}}]}},

    {$count:'count'}

])



1. **Có bao nhiêu người ở Mỹ có mức thu nhập > 50K**

db.user.aggregate([

    {

        $lookup: {

            from: 'occupation',

            localField: 'occupation\_id',

            foreignField: '\_id',

            as: 'output'

        }

    },

    {

        $match: {

            $and: [

                { native\_country: 'United-States' },

                {'output.income\_bracket':'>50K'}

        ]}

    },

    {$count:'Count query 2'}

])



1. **Tính tổng số dư tài khoản của những người đang ở Mỹ.**

db.user.aggregate([

    {

        $lookup: {

            from: 'finance',

            localField: 'finance\_id',

            foreignField: '\_id',

            as:"output"

        }

    },

    { $unwind: { path: '$output' } },

    {$project:{native\_country:1,total:{$convert:{input:'$output.total',to:'int'}}}},

    {

        $match: {native\_country: 'United-States' }

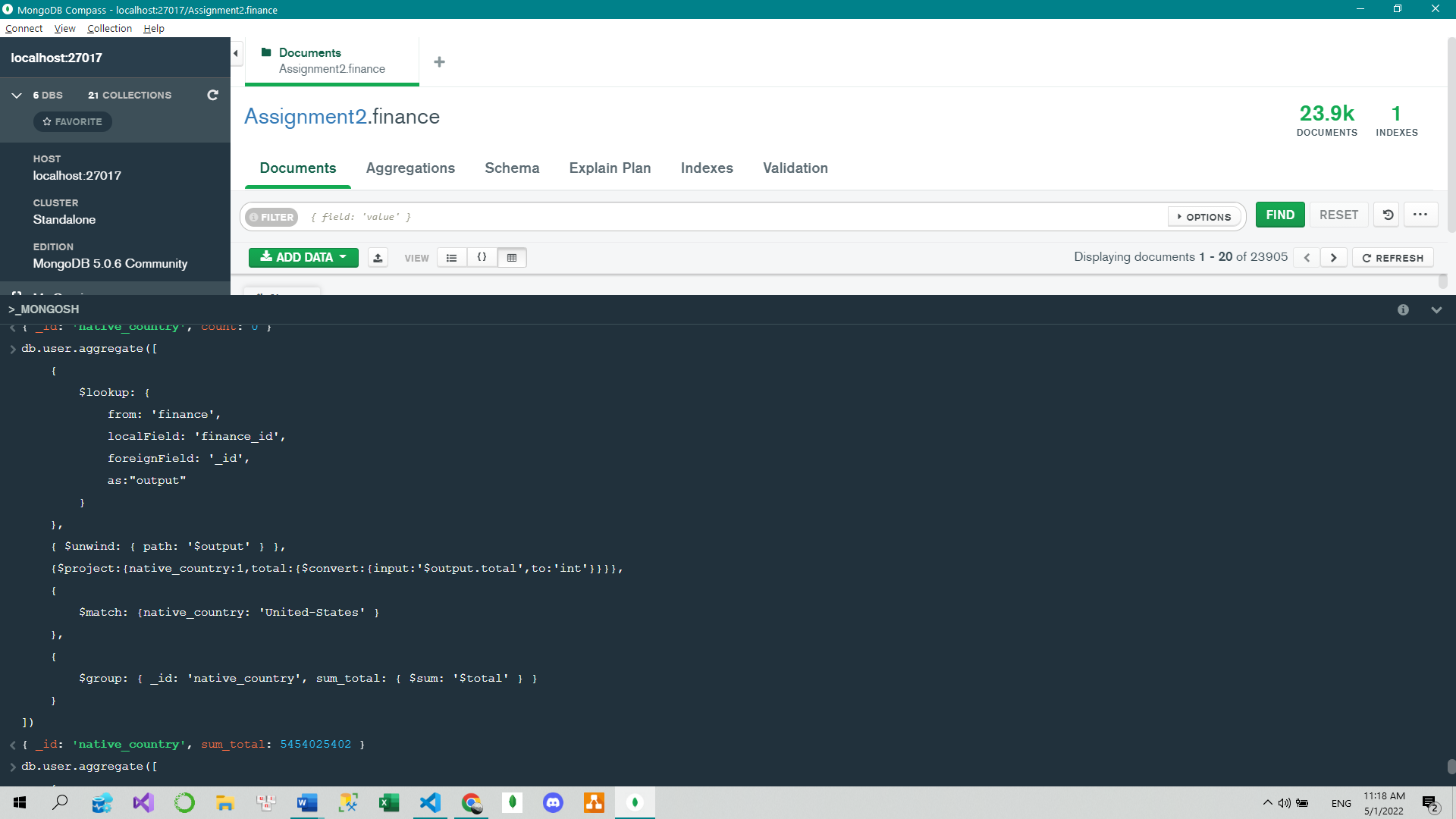
    },

    {

        $group: { \_id: 'native\_country', sum\_total: { $sum: '$total' } }

    }

])



1. **Tính tổng số giờ làm việc một tuần của những người có mức thu nhập <= 50K**

db.occupation.aggregate([

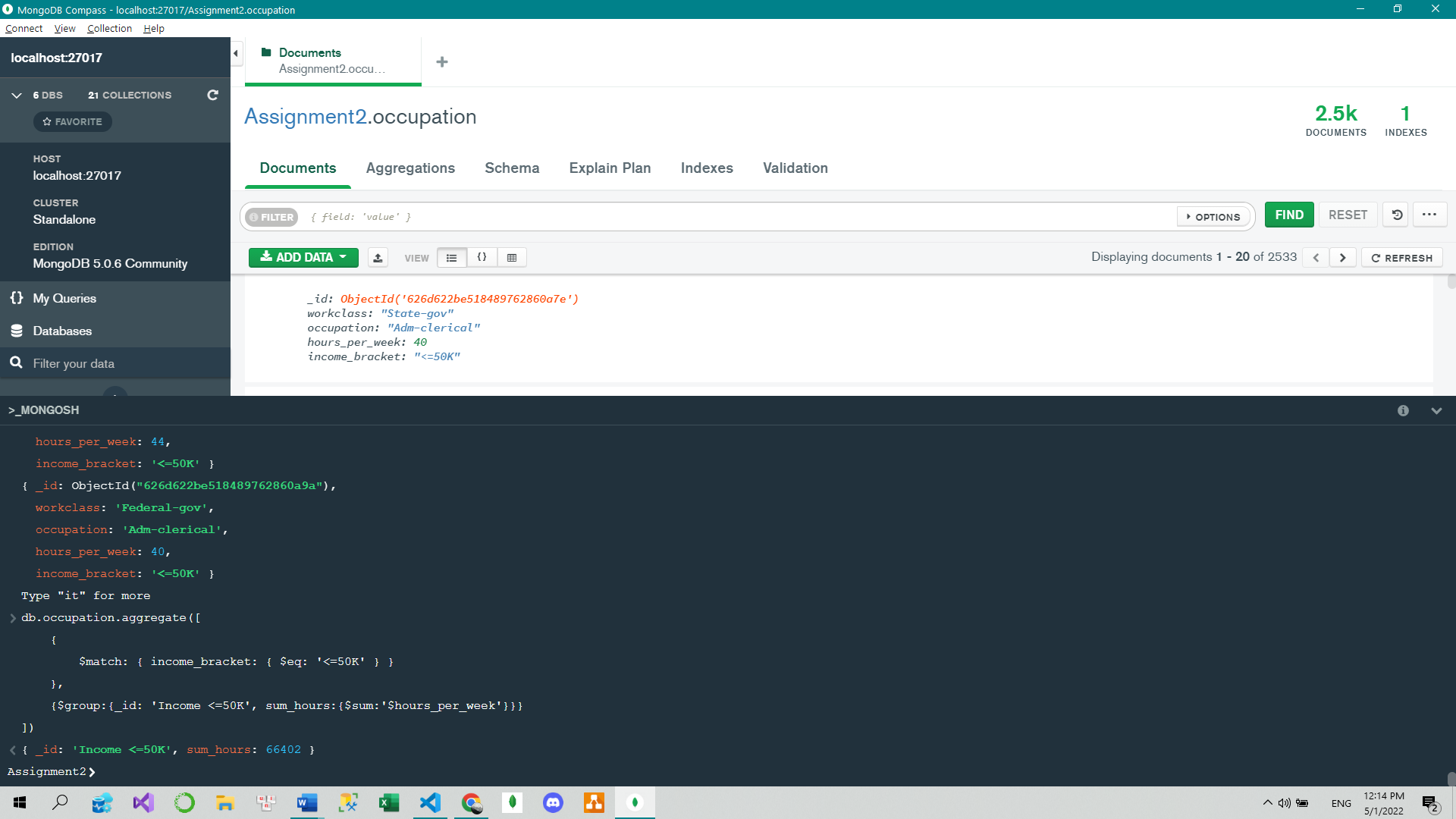
    {

        $match: { income\_bracket: { $eq: '<=50K' } }

    },

    {$group:{\_id: 'Income <=50K', sum\_hours:{$sum:'$hours\_per\_week'}}}

])



1. **Tìm những người có tổng số tiền trong tài khoản > 100000 và có số giờ làm việc hàng tuần < 55.**

db.user.aggregate([

    {

        $lookup: {

            from: 'finance',

            localField: 'finance\_id',

            foreignField: '\_id',

            as:'output1'

        }

    },

    {

        $lookup: {

            from: 'occupation',

            localField: 'occupation\_id',

            foreignField: '\_id',

            as: 'output2'

    }

    },

    {

        $match: {

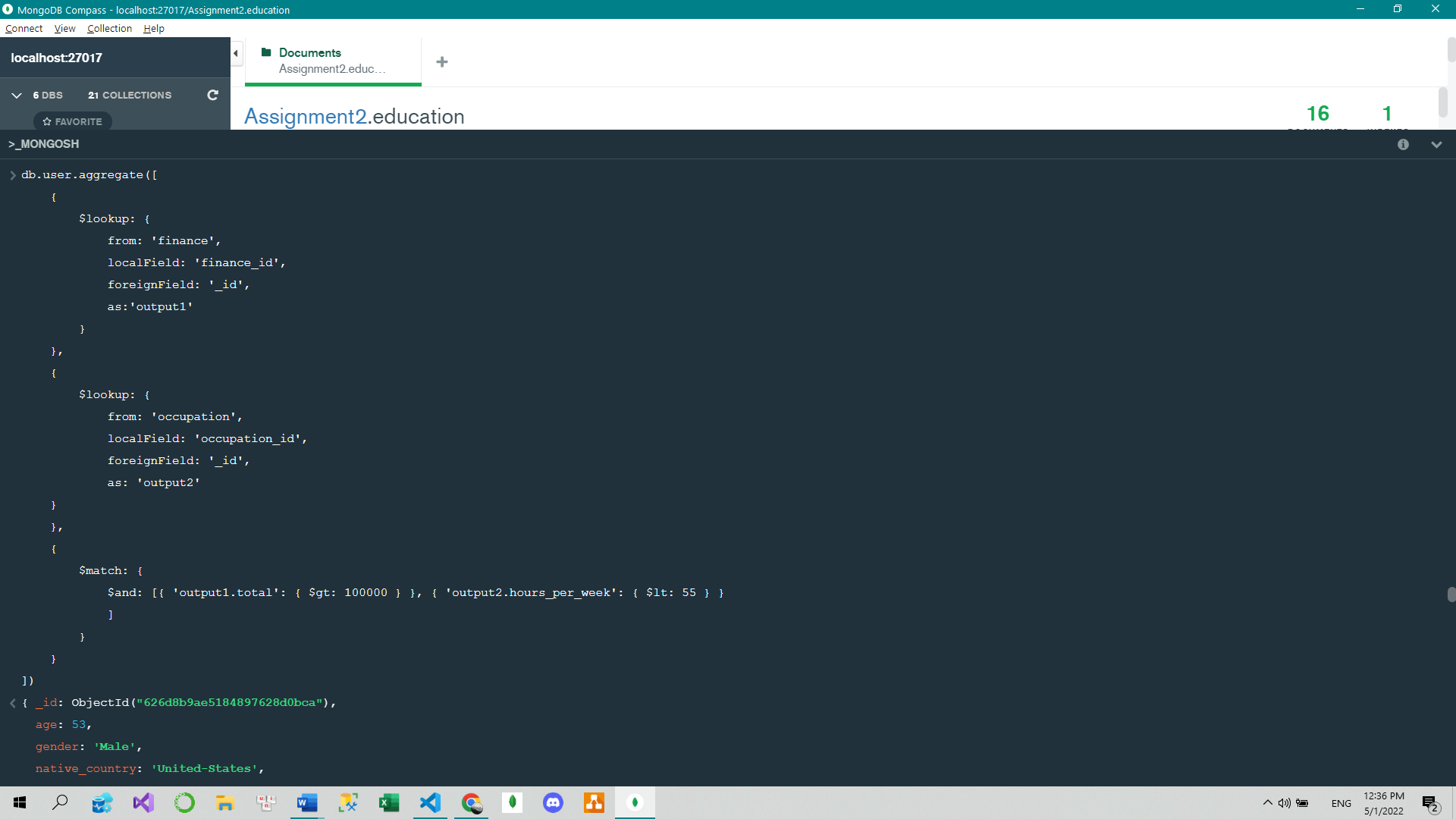
            $and: [{ 'output1.total': { $gt: 100000 } }, { 'output2.hours\_per\_week': { $lt: 55 } }

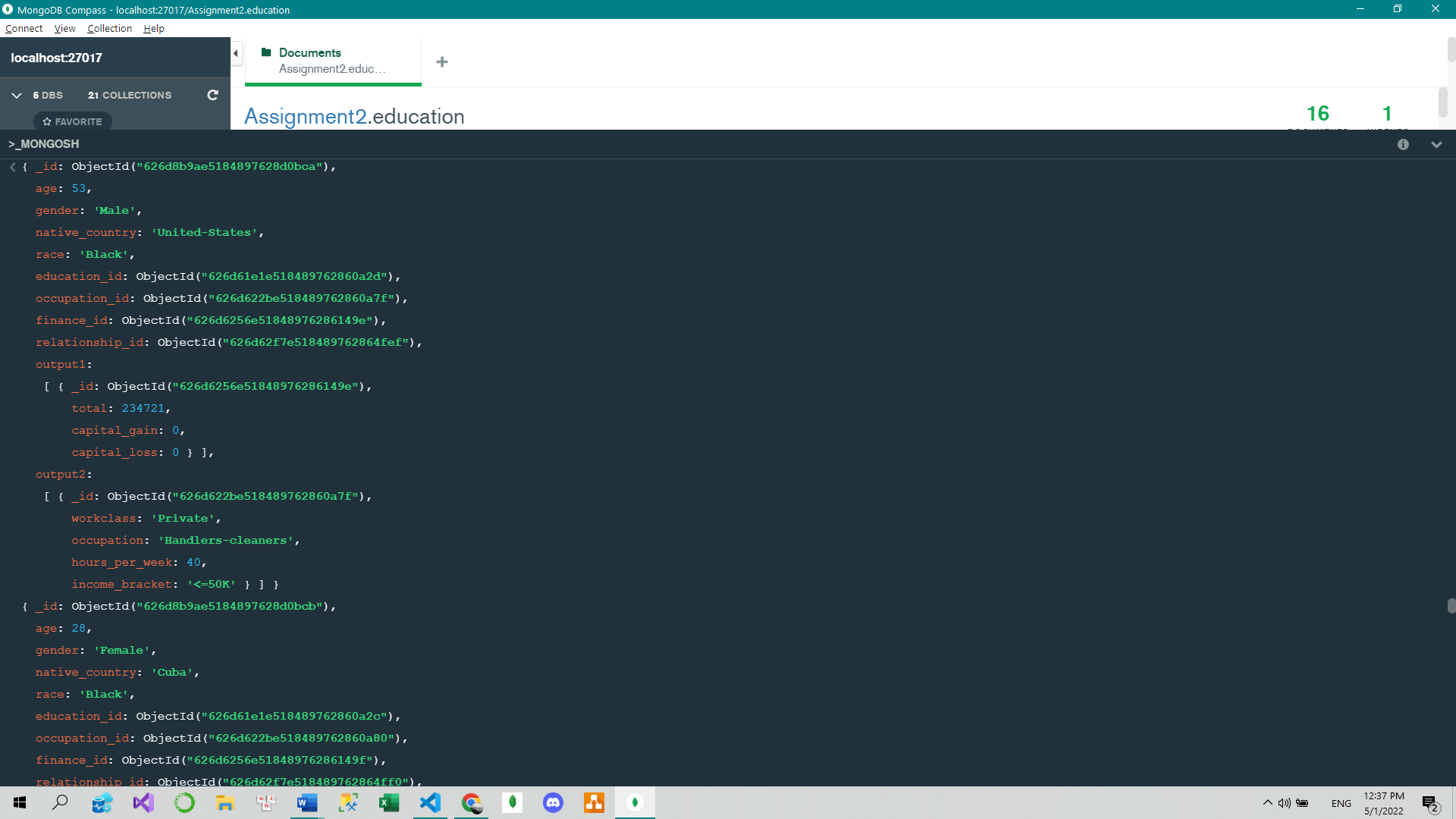
            ]

        }

    }

])





**(Nâng cao)**

1. **Thiết lập index**

Đối với các Database lớn thì việc thiết lập Index rất quan trọng đối với hiệu quả của các truy vấn. Nếu không đánh Index, khi thực hiện truy vấn, Database sẽ scan qua toàn bộ các Doccument để tìm được kết quả truy vấn, và đối với những Collection lớn, việc truy vấn kiểu scan này tốn rất nhiều thời gian.  
 Tuy nhiên, việc đánh index cũng tiêu tốn thêm bộ nhớ trong Database để chứa các bảng Index, và làm giảm tốc độ các câu lệnh thêm, sửa, xóa (các câu lệnh ghi vào Database). Bởi vì với mỗi lần dữ liệu thay đổi MongoDB sẽ phải đánh lại toàn bộ Index, để đảm bảo dữ liệu truy vấn chính xác.

Sau khi xem xét, Data trong các Collection chỉ dung để thực hiện các truy vấn, đặc biệt là các truy vấn nâng cao nên việc Index cho các trường là khá quan trọng.

Vì vậy, việc em quyết định đánh Index cho tất cả các Field của tất cả các Collection( trừ \_id) để tăng hiệu suất truy vấn.

