**Module 4: Assignment 1: Setting up MongoDB in Cloud Environment**

Saurabh Verma

Arizona State University

IFT 598: Middleware Programming & Database Security

Professor. Dinesh Sthapit

October 8, 2022

Step 1:

Creating a cluster on MongoDB:

Graphical user interface, text, application

Description automatically generated

Cluster used for MongoDB:

Graphical user interface, email

Description automatically generated

Selecting the MongoDB Compass:

Graphical user interface, text, application

Description automatically generated

Using MongoDB compass to connect with the database:

Graphical user interface, text, application, email

Description automatically generated

After setting up the connection:

Graphical user interface, text, application

Description automatically generated

Now creating a database in MongoDB:

Graphical user interface, text, application, email

Description automatically generated

Naming the database and collection in cluster– CoursesDB and Courses:

Graphical user interface, application

Description automatically generated

Inserting the first document in collection:

Graphical user interface, text, application, Teams

Description automatically generated

Step 2:

Accessing the project in MongoDB:

Graphical user interface, text, application, Teams

Description automatically generated

Deleting the database in MongoDB:

Graphical user interface, application

Description automatically generated

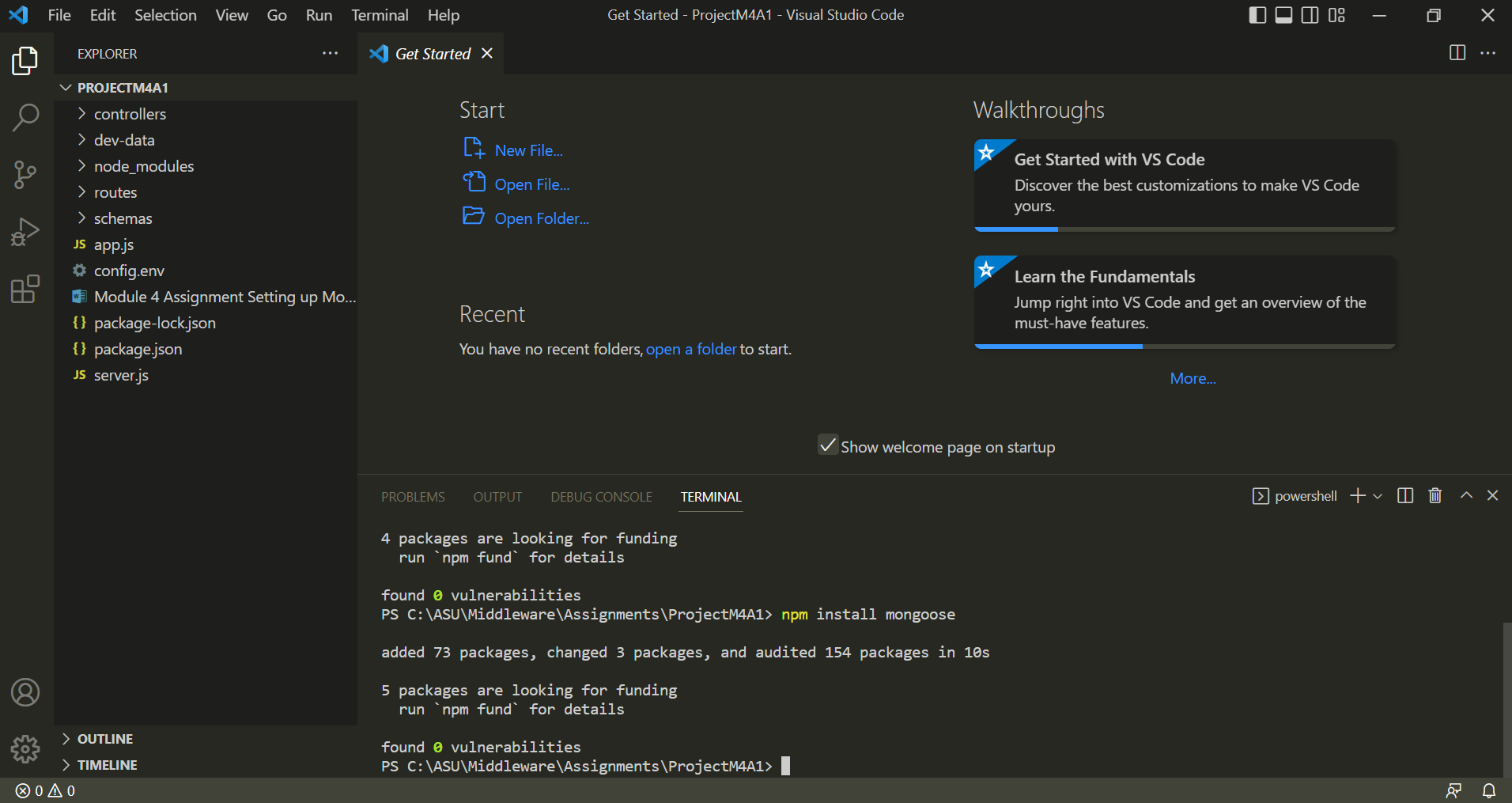
Creating a new database for the LoanDB:

Graphical user interface, application

Description automatically generated

Step 3:

Installing mongoose (npm install mongoose):



Step 4:

Creating schemas repository and loanSchema.js inside the folder. Then, altering the schema based on the loan data (shown in Figure 18)

**Code:**

import mongoose from 'mongoose';

const { Schema } = mongoose;

const loanSchema = new Schema({

  customerName:  String,

  phoneNumber: Number,

  address:   String,

  loanAmount: Number,

  interest: Number,

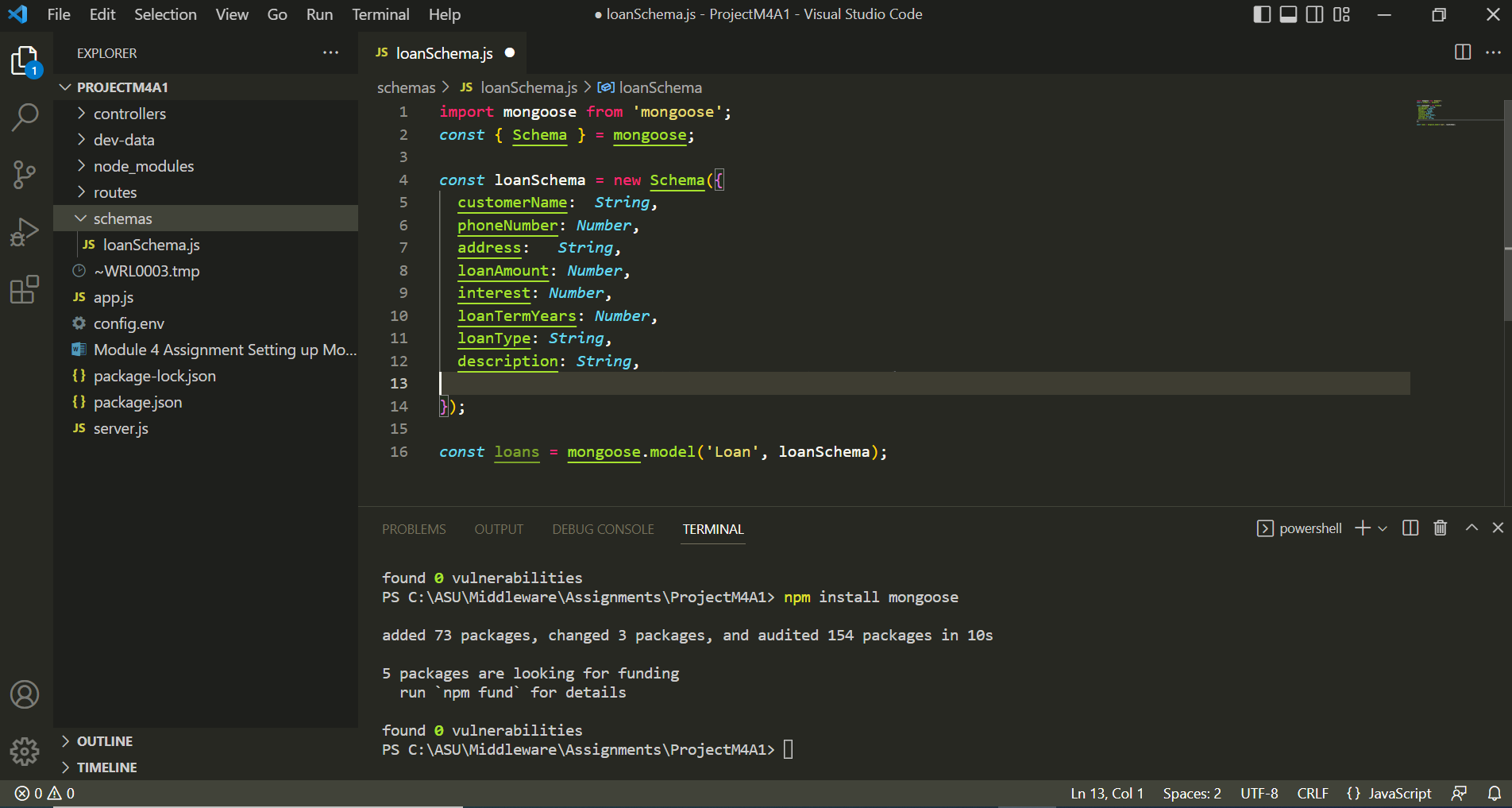
  loanTermYears: Number,

  loanType: String,

  description: String

});

const loans = mongoose.model('Loan', loanSchema);



Step 5:

Schemas are helpful in defining the collection's document's shape. To comprehend the authenticity of the data format given by MongoDB, schemas are crucial. They are required to trigger exceptions if the collections' input is improperly formatted or devoid of the proper data types.

The datatypes that MongoDB supports include:

* String: A string of characters, integers, and special characters would be the data type for this data.
* Number is a data type that would represent values that are numerical.
* Date is a data type that includes various date formats.
* Buffer: Using this data type, you can read built-in or string buffer objects.
* Boolean: Boolean values, such as true and false, are read using this data type.
* Mixed - This data type is used to take into account any kind of data in a field that accepts mixed data.
* ObjectId - The class's objects are identified by their unique identifier, or "ObjectId."
* Array - It accepts arrays of subdocuments and schema types as document input.
* Decimal values up to 128 bits can be entered using Decimal128.
* Map - a tool for entering maps of nested documents with arbitrary keys

Step 6:

The fields createdDate and insertedDate added:

**Code:**

import mongoose from 'mongoose';

const { Schema } = mongoose;

const loanSchema = new Schema({

  customerName:  String,

  phoneNumber: Number,

  address:   String,

  loanAmount: Number,

  interest: Number,

  loanTermYears: Number,

  loanType: String,

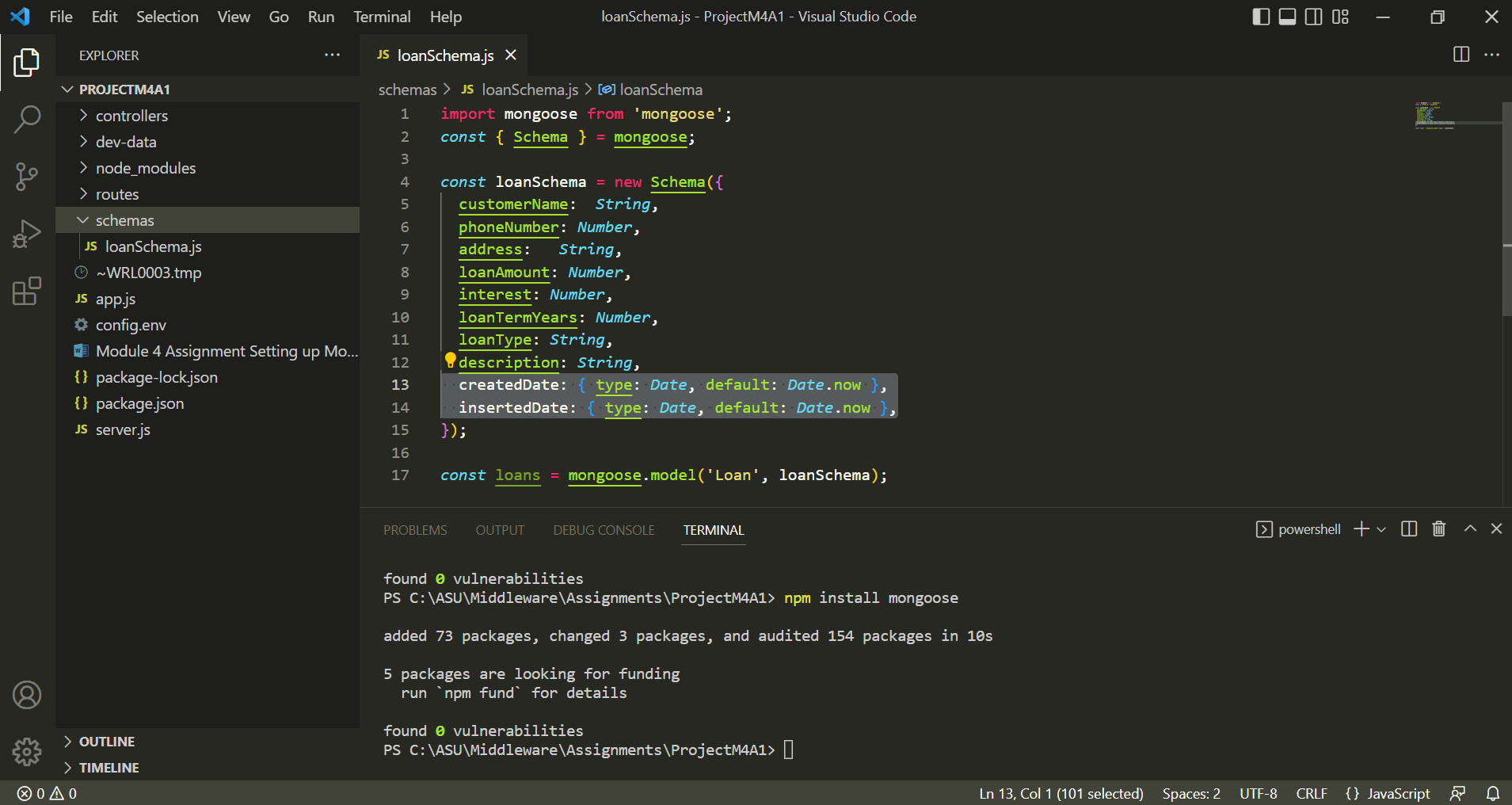
  description: String,

  createdDate: { type: Date, default: Date.now },

  insertedDate: { type: Date, default: Date.now },

});

const loans = mongoose.model('Loan', loanSchema);



Step 7:

Adding the first record:

Graphical user interface, application

Description automatically generated

Added the second record:

Graphical user interface

Description automatically generated

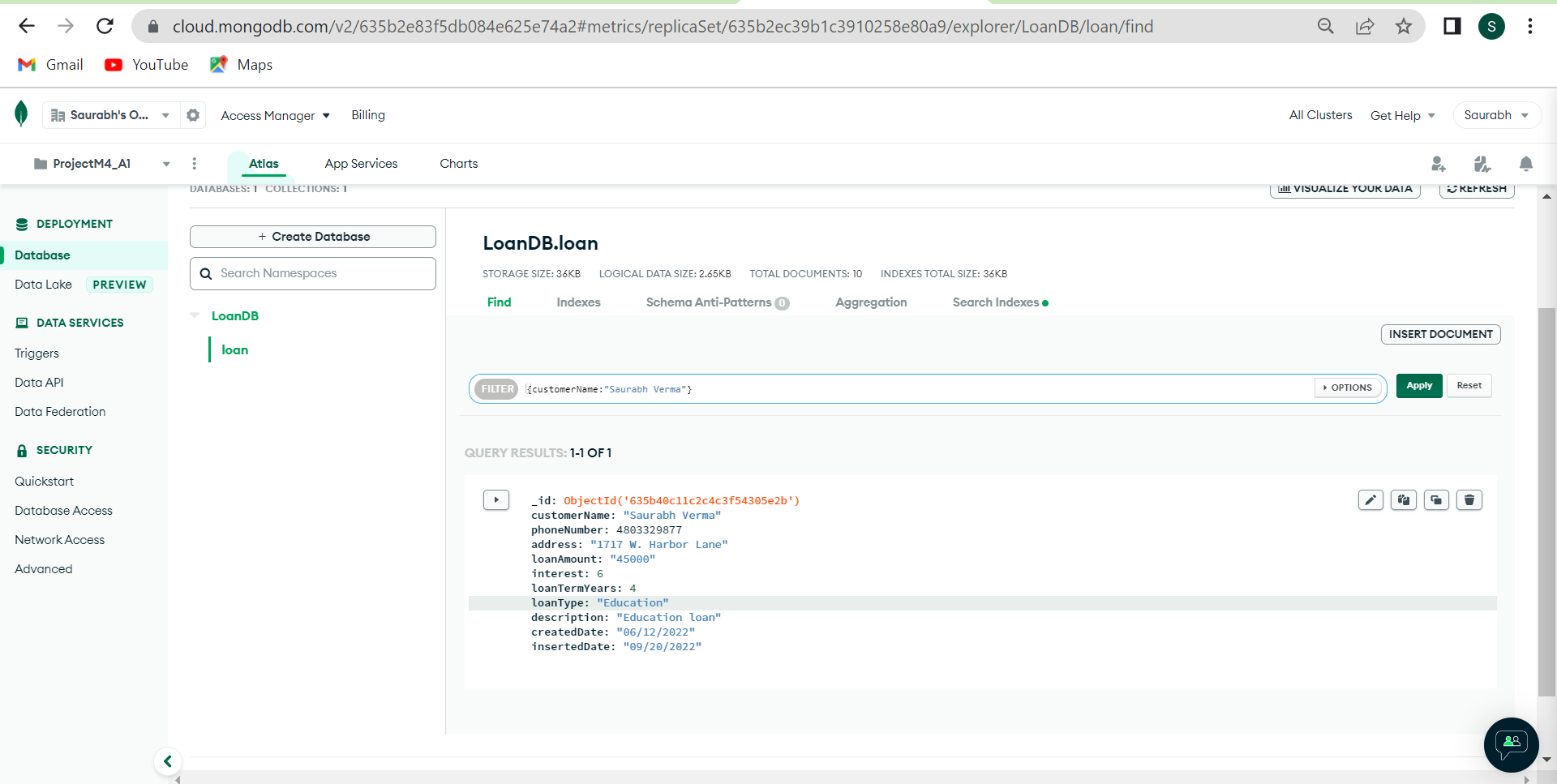
Similarly, added 8 more records:

Graphical user interface, text, application, email

Description automatically generated

Querying:

Query: {customerName:"Saurabh Verma"}



Query: {loanType:"Education"}

Graphical user interface, text, application, email

Description automatically generated

Query: {\_id: ObjectId('635b41851c2c4c3f5431c3f1')}

Graphical user interface, text, application, Teams

Description automatically generated

**References**

[Atlas MongoDB Cloud Offering](https://www.mongodb.com/cloud/atlas/lp/try4?utm_source=bing&utm_campaign=search_bs_pl_evergreen_atlas_core_prosp-brand_gic-null_amers-us_ps-all_desktop_eng_lead&utm_term=atlas%20mongodb&utm_medium=cpc_paid_search&utm_ad=e&utm_ad_campaign_id=415204521&adgroup=1207264237113872&msclkid=834393609b4a11e1048df05965eeeec3)

GitHub link: