

const { MongoClient } = require('mongodb');

const url = 'mongodb://localhost:27017'

// Database Name

const dbName = 'zenportal';

async function run() {

    const client = new MongoClient(url, { useUnifiedTopology: true });

    try {

        await client.connect();

        console.log('Connected successfully to MongoDB server');

        const db = client.db(dbName);

        // Users Collection

        const usersCollection = db.collection('users');

        const users = [

            { name: 'Aria', email: 'aria@gmail.com', role: 'student', joined\_date: new Date('2020-09-01') },

            { name: ‘akhil’, email: 'akhil@gmail.com', role: 'mentor', joined\_date: new Date('2020-08-15') }

        ];

        await usersCollection.insertMany(users);

        // Codekata Collection

        const codekataCollection = db.collection('codekata');

        const codekata = [

            { user\_id: users[0].\_id, problem\_id: 101, date\_solved: new Date('2020-10-20'), status: 'solved' }

        ];

        await codekataCollection.insertMany(codekata);

        // Attendance Collection

        const attendanceCollection = db.collection('attendance');

        const attendance = [

            { user\_id: users[0].\_id, date: new Date('2020-10-15'), status: 'present' }

        ];

        await attendanceCollection.insertMany(attendance);

        // Topics Collection

        const topicsCollection = db.collection('topics');

        const topics = [

            { topic\_name: 'Introduction to MongoDB', date\_taught: new Date('2020-10-14'), instructor\_id: users[1].\_id }

        ];

        await topicsCollection.insertMany(topics);

        // Tasks Collection

        const tasksCollection = db.collection('tasks');

        const tasks = [

            { task\_name: 'Build a REST API', assigned\_to: users[0].\_id, due\_date: new Date('2020-10-20'), submitted: true }

        ];

        await tasksCollection.insertMany(tasks);

        // Company Drivers Collection

        const companyDriversCollection = db.collection('company\_drivers');

        const companyDrivers = [

            {

                company\_name: 'TechCorp',

                drive\_date: new Date('2020-10-25'),

                students\_appeared: [{ user\_id: users[0].\_id, name: users[0].name }]

            }

        ];

        await companyDriversCollection.insertMany(companyDrivers);

        // Mentors Collection

        const mentorsCollection = db.collection('mentors');

        const mentors = [

            {

                mentor\_name: users[1].name,

                email: users[1].email,

                mentees: [{ user\_id: users[0].\_id, name: users[0].name }]

            }

        ];

        await mentorsCollection.insertMany(mentors);

        console.log('Documents inserted successfully');

    } catch (err) {

        console.error(err.stack);

    } finally {

        await client.close();

    }

}

run().catch(console.dir);

QUESTION AND ANSWERS:

**Find all topics and tasks taught in October**:

db.topics.find({ "date\_taught": { $gte: ISODate("2020-10-01"), $lte: ISODate("2020-10-31") } })

db.tasks.find({ "due\_date": { $gte: ISODate("2020-10-01"), $lte: ISODate("2020-10-31") } })

**Find all company drivers that appeared between October 15 and 31, 2020**:

db.company\_drivers.find({ "drive\_date": { $gte: ISODate("2020-10-15"), $lte: ISODate("2020-10-31") } })

**Find all company drivers and students who appeared for placements**:

db.company\_drivers.find({ "students\_appeared": { $exists: true, $not: { $size: 0 } } })

**Find the number of problems solved by a user in Codekata**:

db.codekata.aggregate([

{ $match: { "user\_id": ObjectId("user\_id\_here") } },

{ $group: { \_id: "$user\_id", totalProblemsSolved: { $sum: "$problems\_solved" } } }

])

**Find all mentors with a mentee count greater than 15**:

db.mentors.find({ "mentees": { $exists: true, $size: { $gt: 15 } } })

**Find the number of users who were absent and did not submit tasks between October 15 and 31, 2020**:

db.attendance.aggregate([

{ $match: { "status": "absent", "date": { $gte: ISODate("2020-10-15"), $lte: ISODate("2020-10-31") } } },

{ $lookup: {

from: "tasks",

localField: "user\_id",

foreignField: "user\_id",

as: "tasks"

}

},

{ $unwind: "$tasks" },

{ $match: { "tasks.submitted": false } },

{ $group: { \_id: "$user\_id", count: { $sum: 1 } } }

])