const mongoose = require('mongoose');

const prompt = require('prompt-sync')();

// Define Player schema

const playerSchema = new mongoose.Schema({

name: String,

scores: [Number]

});

// Create Player model

const Player = mongoose.model('Player', playerSchema);

class Team {

constructor() {

this.players = [];

}

addPlayer() {

let playerName = prompt('Enter the name of the player:');

let playerScores = [];

let numScores = parseInt(prompt('Enter the number of scores for the player:'));

for (let i = 1; i <= numScores; i++) {

let score = parseInt(prompt('Enter score ' + i + ' for the player:'));

playerScores.push(score);

}

let player = new Player({ name: playerName, scores: playerScores });

this.players.push(player);

}

getAverageScore() {

let totalScore = 0;

let numPlayers = this.players.length;

let numScores = 0;

for (let i = 0; i < numPlayers; i++) {

let playerScores = this.players[i].scores;

numScores = playerScores.length;

for (let j = 0; j < numScores; j++) {

totalScore += playerScores[j];

}

}

return totalScore / (numPlayers \* numScores);

}

getMinimumScore() {

let minScore = Infinity;

for (let i = 0; i < this.players.length; i++) {

let playerScores = this.players[i].scores;

for (let j = 0; j < playerScores.length; j++) {

if (playerScores[j] < minScore) {

minScore = playerScores[j];

}

}

}

return minScore;

}

getMaximumScore() {

let maxScore = -Infinity;

for (let i = 0; i < this.players.length; i++) {

let playerScores = this.players[i].scores;

for (let j = 0; j < playerScores.length; j++) {

if (playerScores[j] > maxScore) {

maxScore = playerScores[j];

}

}

}

return maxScore;

}

}

async function main() {

// Connect to MongoDB

try {

await mongoose.connect("mongodb+srv://mershikau:12345@cluster0.inmyorb.mongodb.net/?retryWrites=true&w=majority", {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log('Connected to MongoDB');

} catch (err) {

console.error('Failed to connect to MongoDB', err);

process.exit(1);

}

let team = new Team();

// Take input for the number of players

let numPlayers = parseInt(prompt('Enter the number of players:'));

// Take input for each player's name and scores

for (let i = 1; i <= numPlayers; i++) {

team.addPlayer();

}

// Calculate and display the average, minimum, and maximum scores

console.log('Average Score:', team.getAverageScore());

console.log('Minimum Score:', team.getMinimumScore());

console.log('Maximum Score:', team.getMaximumScore());

try {

// Save players to MongoDB

const players = team.players;

for (let i = 0; i < players.length; i++) {

const player = players[i];

await player.save();

}

// Retrieve players from MongoDB

const retrievedPlayers = await Player.find();

console.log('Retrieved Players:', retrievedPlayers);

// Perform update and delete operations

if (retrievedPlayers.length > 0) {

const playerId = retrievedPlayers[0].\_id;

await Player.updateOne({ \_id: playerId }, { name: 'Updated Player' });

await Player.deleteOne({ \_id: playerId });

}

// Retrieve players again after update and delete operations

const updatedPlayers = await Player.find();

console.log('Updated Players:', updatedPlayers);

} catch (err) {

console.error('Error performing CRUD operations', err);

} finally {

// Disconnect from MongoDB

mongoose.disconnect();

console.log('Disconnected from MongoDB');

}

}

// Call the main function

main();