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

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

class Player {

constructor(name, scores) {

this.name = name;

this.scores = scores;

}

}

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(playerName, playerScores);

this.players.push(player);

}

getAverageScore() {

let totalScore = 0;

let numPlayers = this.players.length;

let numScores = 0; // Define numScores outside the loop

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

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

numScores = playerScores.length; // Assign value to numScores

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;

}

}

const uri = "mongodb+srv://mershikau:12345@cluster0.inmyorb.mongodb.net/?retryWrites=true&w=majority"; // Replace with your MongoDB connection string

const client = new MongoClient(uri);

async function connect() {

try {

await client.connect();

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

return client.db('yourDatabaseName'); // Replace 'yourDatabaseName' with your actual database name

} catch (err) {

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

process.exit(1);

}

}

async function createPlayer(player) {

const db = await connect();

const playersCollection = db.collection('players');

try {

const result = await playersCollection.insertOne(player);

console.log('Player created successfully');

return result.insertedId;

} catch (err) {

console.error('Failed to create player', err);

return null;

}

}

async function getPlayers() {

const db = await connect();

const playersCollection = db.collection('players');

try {

const players = await playersCollection.find().toArray();

return players;

} catch (err) {

console.error('Failed to get players', err);

return [];

}

}

async function updatePlayer(playerId, updates) {

const db = await connect();

const playersCollection = db.collection('players');

try {

const result = await playersCollection.updateOne({ \_id: playerId }, { $set: updates });

console.log('Player updated successfully');

return result.modifiedCount;

} catch (err) {

console.error('Failed to update player', err);

return 0;

}

}

async function deletePlayer(playerId) {

const db = await connect();

const playersCollection = db.collection('players');

try {

const result = await playersCollection.deleteOne({ \_id: playerId });

console.log('Player deleted successfully');

return result.deletedCount;

} catch (err) {

console.error('Failed to delete player', err);

return 0;

}

}

async function main() {

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());

// Save players to MongoDB

const players = team.players;

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

const player = players[i];

await createPlayer(player);

}

// Retrieve players from MongoDB

const retrievedPlayers = await getPlayers();

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

// Perform update and delete operations

if (retrievedPlayers.length > 0) {

const playerId = retrievedPlayers[0].\_id;

await updatePlayer(playerId, { name: "Updated Player" });

await deletePlayer(playerId);

}

// Retrieve players again after update and delete operations

const updatedPlayers = await getPlayers();

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

}

// Call the main function

main()