**Explanation of Key Corrections and Structure:**

* **Index Creation with Proper Syntax**: The syntax for createIndex is corrected for single indexes, and compound indexes are specified with correct key-value pairs.
* **Unique Index Creation**: The unique index on email now ensures each email is unique.
* **Dropping Specific Indexes**: Only existing indexes are dropped after verification with getIndexes().

// 1. Connect to MongoDB and switch to the UniversityDB database

use UniversityDB

// 2. Insert multiple student records into the 'students' collection

db.students.insertMany([

{ name: "Alice", major: "Computer Science", gpa: 3.9, email: "alice@example.com" },

{ name: "Bob", major: "Mathematics", gpa: 3.5, email: "bob@example.com" },

{ name: "Charlie", major: "Computer Science", gpa: 3.7, email: "charlie@example.com" },

{ name: "David", major: "Mathematics", gpa: 3.2, email: "david@example.com" },

{ name: "Eve", major: "Physics", gpa: 3.8, email: "eve@example.com" }

])

// 3. View all documents in the 'students' collection

db.students.find().pretty()

// 4. Create an index on the 'name' field to optimize queries by student name

db.students.createIndex({ name: 1 })

// 5. Create a compound index on 'major' (ascending) and 'gpa' (descending) to optimize

// queries by major and also allow sorting by GPA within each major

db.students.createIndex({ major: 1, gpa: -1 })

// 6. Create a unique index on the 'email' field to enforce unique email addresses for each student

db.students.createIndex({ email: 1 }, { unique: true })

// 7. Check the current indexes on the 'students' collection

db.students.getIndexes()

// 8. Drop the index on 'email' (if no longer needed)

db.students.dropIndex({ email: 1 })

// 9. Drop the compound index on 'major' and 'gpa' (if no longer needed)

db.students.dropIndex({ major: 1, gpa: -1 })

// 10. Verify the remaining indexes

db.students.getIndexes()

// 11. View index sizes for monitoring the memory usage of each index

db.students.stats().indexSizes