6410545444 Thanida Chaiwongnon

1.) You're creating a database to contain a set of sensor measurements from a two-dimensional grid. Each measurement is a time-sequence of readings, and each reading contains ten labeled values.Should you use the relational model or MongoDB? Please justify your answer

: I think using MongoDB will be better choice for this data because it's easy to access data and flexible schema allow for easy updates and scaling.

2) For each of the following applications

a. IoT, b. E-commerce, c. Gaming, d. Finance

Propose an appropriate Relational Model or MongoDB database schema. For each application, clearly justify your choice of database.

a. IoT application that involves collecting data from many devices, MongoDB may be a good choice due to its ability to handle semi-structured, unstructured data and flexible schema allows for easy updates and scaling as new devices are added.
b. E-commerce: For an e-commerce application that involves storing product information and customer orders, a relational database may be a good choice due to the structured nature of the data and allow for efficient querying and data integrity using constraints.
c. Gaming: For a gaming application that involves tracking player data and game events, MongoDB may be a good choice due to its ability to handle semi-structured data, scalability and indexing and querying capabilities could be used to efficiently retrieve player data and game stats.
d. Finance: For a finance application that involves tracking financial transactions and user data, a relational database may be a good choice due to the structured nature of the data and the need for data integrity and allow for efficient querying and the use of constraints to enforce data integrity.

3.)

1. Open the MongoDB shell by running the command "mongo" in the terminal.
2. Create a new database by running the command "use student" in the MongoDB shell.
3. Create a new collection called "marks" by running the command "db.createCollection('std')" in the MongoDB shell.
4. Insert the given documents into the "std" collection by running the following commands:

-

```
< 'switched to db <students>'
> db.std.insertMany([
    {"name":"Ramesh","subject":"maths","marks":87},
    {"name":"Ramesh","subject":"english","marks":59},
    {"name":"Ramesh","subject":"science","marks":77},
    {"name":"Rav","subject":"maths","marks":62},
    {"name":"Rav","subject":"english","marks":83},
    {"name":"Rav","subject":"science","marks":71},
    {"name":"Alison","subject":"maths","marks":84},
    {"name":"Alison","subject":"english","marks":82},
    {"name":"Alison","subject":"science","marks":86},
    {"name":"Steve","subject":"maths","marks":81},
    {"name":"Steve","subject":"english","marks":89},
    {"name":"Steve","subject":"science","marks":77},
    {"name":"Jan","subject":"english","marks":0,"reason":"absent"}])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("64250666a6e3c7fadcf3ab94"),
      '1': ObjectId("64250666a6e3c7fadcf3ab95"),
      '2': ObjectId("64250666a6e3c7fadcf3ab96"),
      '3': ObjectId("64250666a6e3c7fadcf3ab97"),
      '4': ObjectId("64250666a6e3c7fadcf3ab98"),
      '5': ObjectId("64250666a6e3c7fadcf3ab99"),
      '6': ObjectId("64250666a6e3c7fadcf3ab9a"),
      '7': ObjectId("64250666a6e3c7fadcf3ab9b"),
      '8': ObjectId("64250666a6e3c7fadcf3ab9c"),
      '9': ObjectId("64250666a6e3c7fadcf3ab9d"),
      '10': ObjectId("64250666a6e3c7fadcf3ab9e"),
      '11': ObjectId("64250666a6e3c7fadcf3ab9f"),
      '12': ObjectId("64250666a6e3c7fadcf3aba0")
    }
```

- Find the total marks for each student across all subjects.

```
> db.std.aggregate([{$group: {_id: "$name", total_marks: {$sum: "$marks"}}}])
< {
    _id: 'Alison',
    total_marks: 252
  }
  {
    _id: 'Rav',
    total_marks: 216
  }
  {
    _id: 'Jan',
    total_marks: 0
  }
  {
    _id: 'Ramesh',
    total_marks: 223
  }
  {
    _id: 'Steve',
    total_marks: 247
  }
<students>
```

- Find the maximum marks scored in each subject.

```
  }
> db.std.aggregate([{$group: {_id: "$subject", max_marks: {$max: "$marks"}}}])
< {
    _id: 'maths',
    max_marks: 87
  }
  {
    _id: 'english',
    max_marks: 89
  }
  {
    _id: 'science',
    max_marks: 86
  }
<students>
```

- Find the minimum marks scored by each student.

```
> db.std.aggregate([{$group: {_id: "$name", minMarks: {$min: "$marks"}}}])
< {
    _id: 'Rav',
    minMarks: 62
  }
  {
    _id: 'Jan',
    minMarks: 0
  }
  {
    _id: 'Alison',
    minMarks: 82
  }
  {
    _id: 'Steve',
    minMarks: 77
  }
  {
    _id: 'Ramesh',
    minMarks: 59
  }
<students>
```

- Find the top two subjects based on average marks.

```
> db.std.aggregate([{$group: {_id: "$subject", avgMarks: {$avg: "$marks"}}},
  {$sort: {avgMarks: -1}},
  {$limit: 2}])
< {
    _id: 'maths',
    avgMarks: 78.5
  }
  {
    _id: 'science',
    avgMarks: 77.75
  }
<students>
```