1.

For storing sensor measurements from a two-dimensional grid with time-based sequences of readings, a time-series database is a better fit than a traditional relational database or MongoDB. Time-series databases are optimized for handling time-series data and provide better performance and scalability for time-based queries. In summary, a time-series database would be the preferred choice for this specific use case.

2.

- For an IoT application, a document-oriented database like MongoDB would be suitable due to the large volumes of unstructured data generated by IoT devices.
- For an e-commerce application, a relational model database would be suitable due to the structured data with well-defined relationships between tables, such as customer information, orders, and products.
- For a gaming application, a document-oriented database like MongoDB would be suitable due to the large volumes of unstructured data generated by game events, player data, and chat messages.
- For a finance application, a relational model database would be suitable due to the structured data with well-defined relationships between tables, such as account information, transactions, and balances.

3.

- Create and insert data

```
test> use students
switched to db students
students> db.students.insertMany([   {"name":"Ramesh","subject":"maths","marks":87},   {"name":"Ramesh","subjec
t":"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("642995e6648162f01d13a1d9"),
    '1': ObjectId("642995e6648162f01d13a1da"),
    '2': ObjectId("642995e6648162f01d13a1db"),
    '3': ObjectId("642995e6648162f01d13a1dc"),
    '4': ObjectId("642995e6648162f01d13a1dd"),
    '5': ObjectId("642995e6648162f01d13a1de"),
    '6': ObjectId("642995e6648162f01d13a1df"),
    '7': ObjectId("642995e6648162f01d13a1e0"),
    '8': ObjectId("642995e6648162f01d13a1e1"),
    '9': ObjectId("642995e6648162f01d13a1e2"),
    '10': ObjectId("642995e6648162f01d13a1e3"),
    '11': ObjectId("642995e6648162f01d13a1e4"),
    '12': ObjectId("642995e6648162f01d13a1e5")
  }
}
```

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

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

- Find the maximum marks scored in each subject.

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

- Find the minimum marks scored by each student.

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

- Find the top two subjects based on average marks.

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

Github link for MongoDB: https://github.com/breezjirasak/MongoDB-HW