1. We should use the non-relational database model since we have to input data consistently and each sensor can have variable data inputs.

2.1 IoT

Database: MongoDB

Reason: I think we should use MongoDB because the values inputted from different sensors and devices can be varied and we need some scalability for when we want to add more devices.

Schema:

- Collection storing data on devices containing device id, name, location.
- Collection storing data from sensors containing sensor id, device id, values, time inputted.

2.2 E-commerce

Database: Relational

Reason: I think we should use a relational database because data used in e-commerce tends to be rigid and same for every entry into the database and do not require the flexibility of MongoDB.

Schema:

- User(user_id, name, email, address)
- Product(product_id, name, description, price, stock)
- Orders(user_id, product_id, bought_date, payment_type, shipping_type)

2.3 Gaming

Database: MongoDB

Reason: I think we should use MongoDB here because the data in games such as items can be highly varied and diverse so we would need the flexibility of a non-relational database.

Schema:

- Collection for storing player data containing player id, username, password, email, in game name.
- Collection for storing item data containing item_id, effects, rarity, tradeable.

2.4 Finance

Database: Relational database

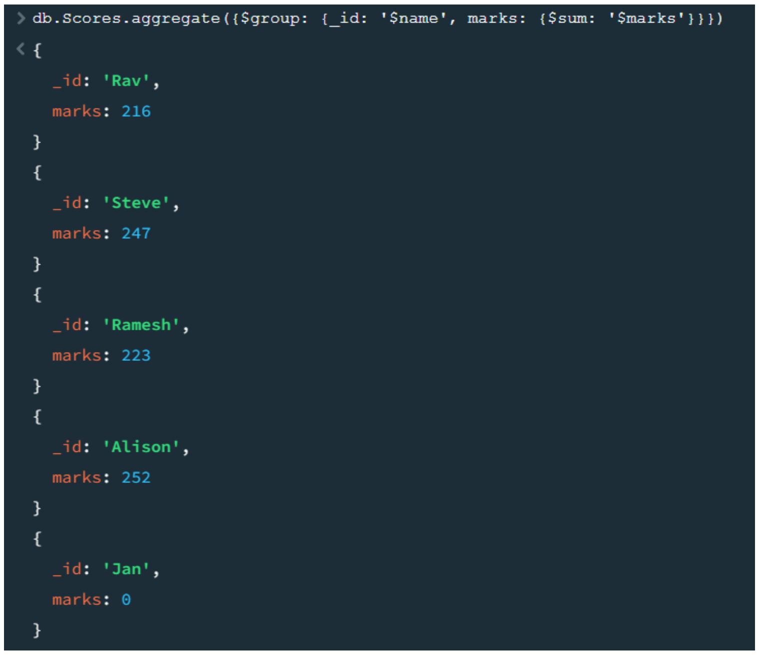
Reason: I think we should use relational database because the data is very structured and we can put them in separate unique columns in a relational database.

Schema:

- User(user_id, name, email, address)
- Account(account_id, owner_id, account_type, balance)
- Transaction(sender_id, receiver_id, amount, timestamp)

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

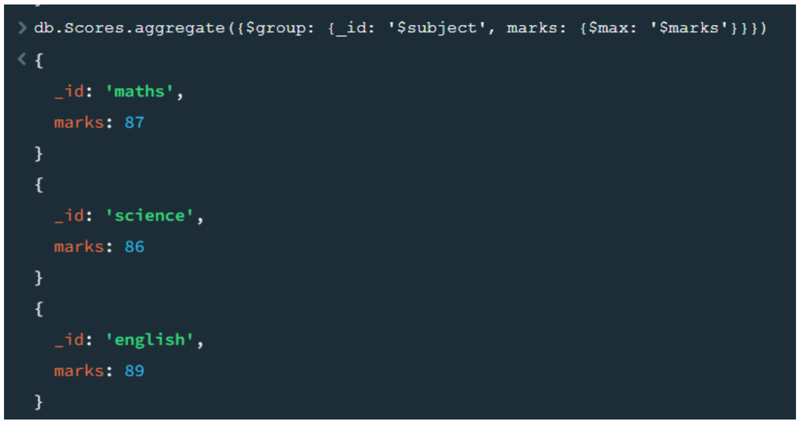
db.Scores.aggregate({$group: {_id: '$name', marks: {$sum: '$marks'}}})



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

3.2 Find the maximum marks scored in each subject.

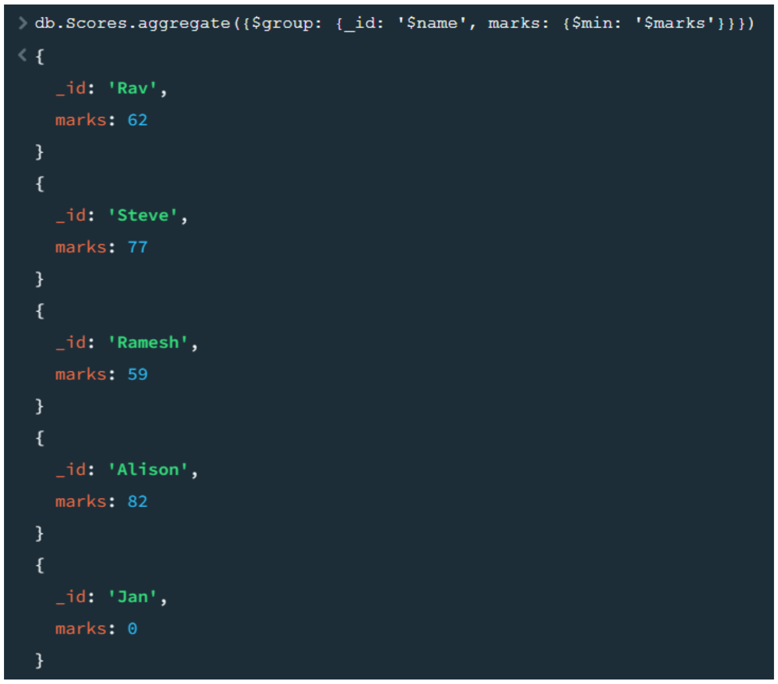
db.Scores.aggregate({$group: {_id: '$subject', marks: {$max: '$marks'}}})



```
> db.Scores.aggregate({$group: {_id: '$subject', marks: {$max: '$marks'}}})
< {
    _id: 'maths',
    marks: 87
  }
  {
    _id: 'science',
    marks: 86
  }
  {
    _id: 'english',
    marks: 89
  }
```

3.3 Find the minimum marks scored by each student.

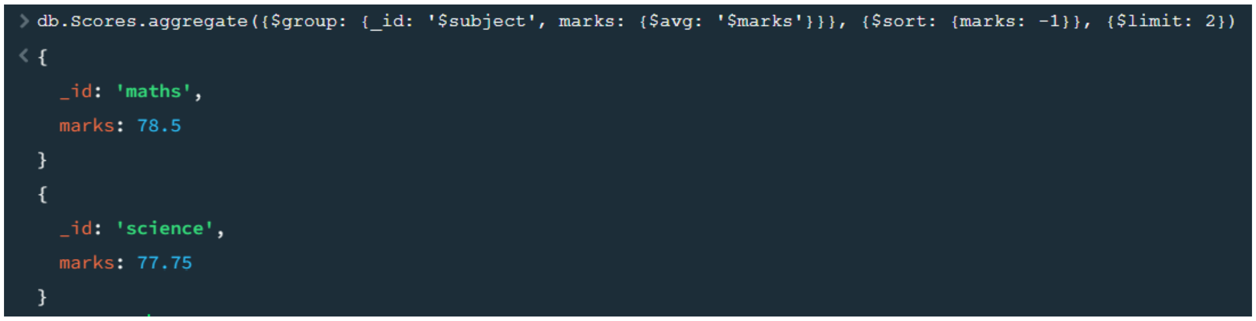
db.Scores.aggregate({$group: {_id: '$name', marks: {$min: '$marks'}}})



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

3.4 Find the top two subjects based on average marks.

db.Scores.aggregate({$group: {_id: '$subject', marks: {$avg: '$marks'}}}, {$sort: {marks: -1}}, {$limit: 2})



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