

1. I recommend using MongoDB because I think we work on semi-structured data so MongoDB should be fulfilled data requirement.
2. I recommend using MongoDB because I think we work on data that unstructured or semi-structured data that doesn't fit the relational model and require the flexibility of a dynamic schema or want more choice over the data model.
3. I recommend using relational model because I think for this data we require a high degree of data integrity, and data is highly structured.

4. Gaming

```
Game
{
  team: {
    team1: {
      name: string,
      member: {
        member1_name: string,
        member2_name: string,
        .....
      },
      score: int
    },
    team2: {
      name: string,
      member: {
        member1_name: string,
        member2_name: string,
        .....
      },
      score: int
    },
    .....
  },
  map: string,
  mode: string
}
```

```

Player
{
    Id: int,
    level: int
    name: string,
    clan: string,
    k/da: float,
    money: int,
}

```

5. Create MongoDB database with following information

```

> db.ExamScore.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":78}, {"name":"Rav","subject":"science","marks":91}, {"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('623b731459215b429645c498'), '1': ObjectId('623b731459215b429645c499'), '2': ObjectId('623b731459215b429645c49a'), '3': ObjectId('623b731459215b429645c49b'), '4': ObjectId('623b731459215b429645c49c'), '5': ObjectId('623b731459215b429645c49d'), '6': ObjectId('623b731459215b429645c49e'), '7': ObjectId('623b731459215b429645c49f'), '8': ObjectId('623b731459215b429645c4a0'), '9': ObjectId('623b731459215b429645c4a1'), '10': ObjectId('623b731459215b429645c4a2'), '11': ObjectId('623b731459215b429645c4a3'), '12': ObjectId('623b731459215b429645c4a4')])

Find the total marks for each student across all subjects.

```

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

```

Find the maximum marks scored in each subject.

```

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

```

Find the minimum marks scored by each student.

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

Find the top two subjects based on average marks.

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