HW#XXX: NoSQL & MongoDB

1) I would use the relational model because the attribute count is static. Using MongoDB would just slow the query speed down without any benefit.

2) I would use MongoDB here since the attribute count is uncertain. If I were to use relational model, then I would have to alter the table every time there's an additional attribute.

3.) I would use the relational model because the database needs to be time sensitive, and the attribute count is static.
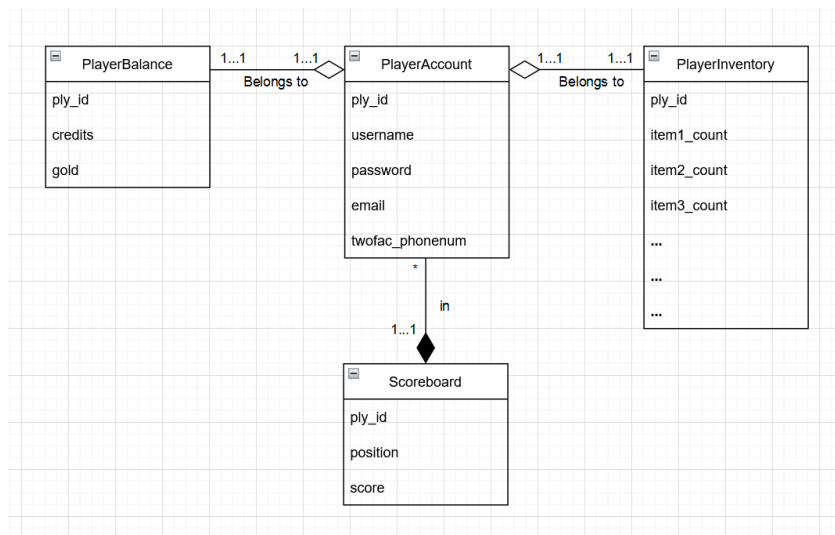
4.) Gaming | MongoDB

PlayerAccount(<u>ply_id</u>, <u>username</u>, password, email, twofac_phonenum)

PlayerBalance(<u>ply_id</u>, credits, gold)

PlayerInventory(<u>ply_id</u>, item1_count, item2_count, ...)

Scoreboard(<u>ply_id</u>, <u>position</u>, score)

5)

Database creation

Connect View Collection Help

Local

HW_6310545418.Stud... ✕ +
Documents

⌄ 5 DBS   3 COLLECTIONS   C   **HW_6310545418.Students**   STORAGE SIZE AVG. SIZE   TOTAL SIZE AVG. SIZE
   ☆ FAVORITE   DOCUMENTS 13   20.5KB 70B   INDEXES 1 36.9KB 36.9KB

>_MONGOSH   ⓘ ⌄

> use HW_6310545418
< 'switched to db HW_6310545418'
> db.createCollection("Students")
< { ok: 1 }
> db.Students.insertMany([{"name":"Ramesh","subject":"maths","marks":87}, {"name":"Ramesh","subject":"english","marks":59}, {"name":"Ramesh","subject":"science","marks":77}])
< { acknowledged: true,
    insertedIds:
     { '0': ObjectId("6232a236c0139692ba268e72"),
       '1': ObjectId("6232a236c0139692ba268e73"),
       '2': ObjectId("6232a236c0139692ba268e74") } }
> db.Students.insertMany([{"name":"Rav","subject":"maths","marks":62}, {"name":"Rav","subject":"english","marks":83}, {"name":"Rav","subject":"science","marks":71}])
< { acknowledged: true,
    insertedIds:
     { '0': ObjectId("6232a26bc0139692ba268e75"),
       '1': ObjectId("6232a26bc0139692ba268e76"),
       '2': ObjectId("6232a26bc0139692ba268e77") } }
> db.Students.insertMany([{"name":"Alison","subject":"maths","marks":84}, {"name":"Alison","subject":"english","marks":82}, {"name":"Alison","subject":"science","marks":86}])
< { acknowledged: true,
    insertedIds:
     { '0': ObjectId("6232a296c0139692ba268e78"),
       '1': ObjectId("6232a296c0139692ba268e79"),
       '2': ObjectId("6232a296c0139692ba268e7a") } }
> db.Students.insertMany([{"name":"Steve","subject":"maths","marks":81}, {"name":"Steve","subject":"english","marks":89}, {"name":"Steve","subject":"science","marks":77}])
< { acknowledged: true,
    insertedIds:
     { '0': ObjectId("6232a2c9c0139692ba268e7b"),
       '1': ObjectId("6232a2c9c0139692ba268e7c"),

       '2': ObjectId("6232a2c9c0139692ba268e7d") } }
> db.Students.insertOne({"name":"Jan","subject":"english","marks":0,"reason":"absent"})
< { acknowledged: true,
    insertedId: ObjectId("6232a2e1c0139692ba268e7e") }
HW_6310545418 >

Queries

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

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

- Find the maximum marks scored in each subject.

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

- Find the minimum marks scored by each student.

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

- Find the top two subjects based on average marks.

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