Context and MongoDB

Recitation / 6104 / Fall 2025

today

1. Assignment 4a: Exercise 0
    a. Taking the time to setup and debug that it works
2. run `deno test -A` from root
    a. LikertSurvey should run; check MongoDB Atlas cluster
3. clone https://github.com/61040-fa25/mongo-recitation
    a. or http://tinyurl.com/mongo-recitation

Context: lecture recap

- One document = the entire context
- Markdown-based, no special syntax
- cli tool: `ctx prompt file.md`
- Include any files using links with @ sign in the description:
    - `[@prompt.md](prompt.md)`
    - `[@MyConcept.ts](/src/MyConcept.ts)`

including files

- you can drag-drop inside Obsidian, or copy from outside
- relative link:
  - `[@prompt.md](prompt.md)`
- root-based (starts with /):
  - `[@MyConcept.ts](/src/MyConcept.ts)`
  - useful when using relative links in VSCode

try exercise 0!

tips:

- most terminals have auto complete when using `./ctx`
- Obsidian has a copy relative link right/ctrl click menu too
    - useful for getting link to paste into `./ctx`
- `context` directory also has metadata, like token usage

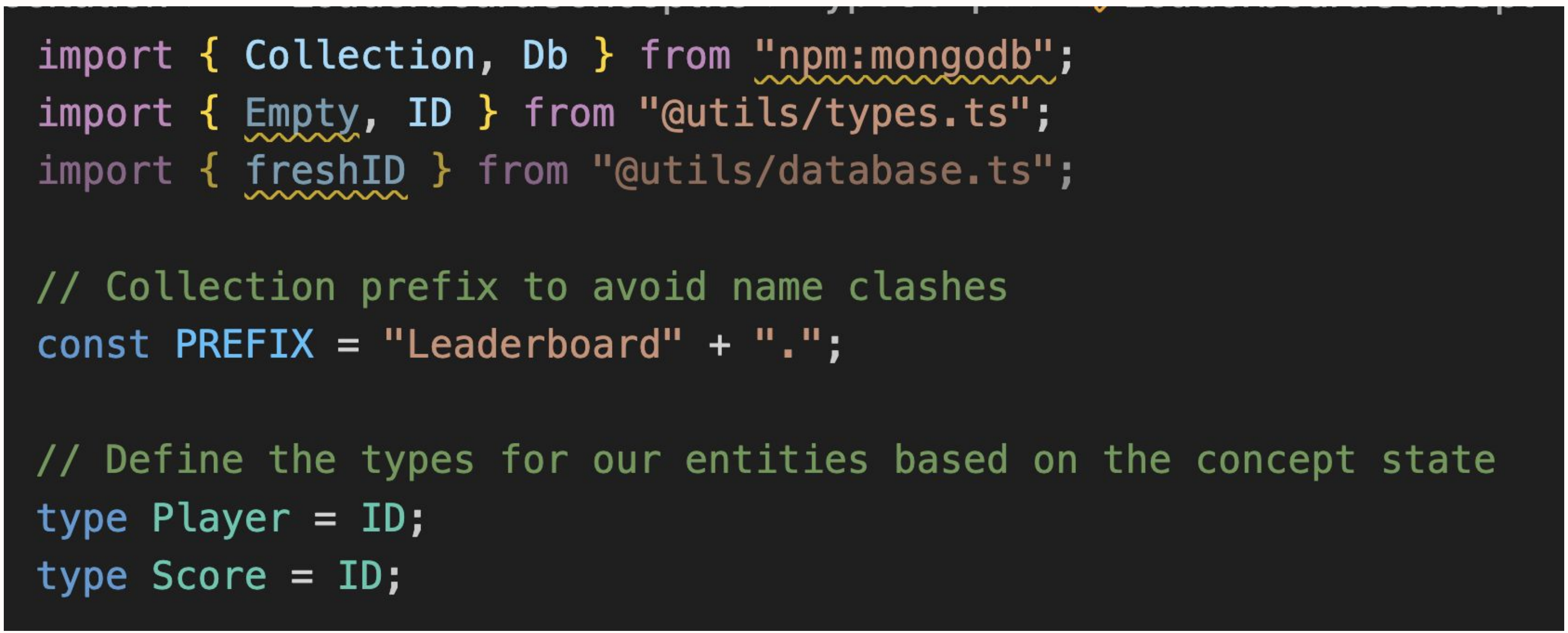
MongoDB

- a NoSQL, document-based database
- documents can be heterogeneous (different schemas)
- organized by collections

MongoDB: IDs

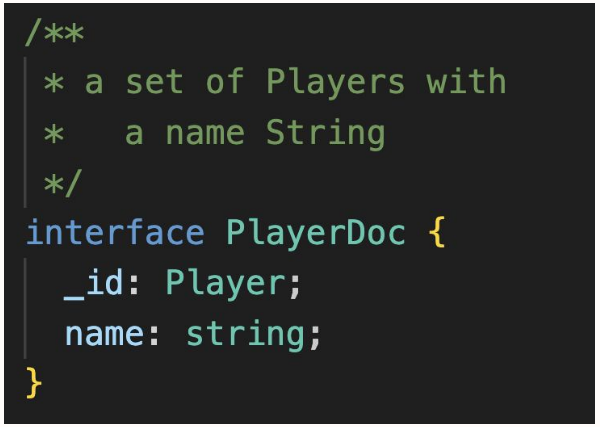
- MongoDB internally wants to use `ObjectId`
  - problem: lots of extra conversion, requires wrappers
  - solution: use `string`s instead, and **type branding**
- type branding: create a type `ID` that's just a string
  - but complains when you assign string -> ID

MongoDB: setup

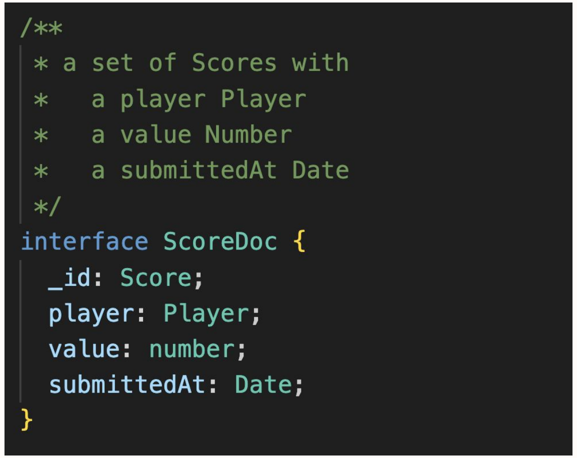


```
import { Collection, Db } from "npm:mongodb";
import { Empty, ID } from "@utils/types.ts";
import { freshID } from "@utils/database.ts";

// Collection prefix to avoid name clashes
const PREFIX = "Leaderboard" + ".";

// Define the types for our entities based on the concept state
type Player = ID;
type Score = ID;
```

MongoDB: state



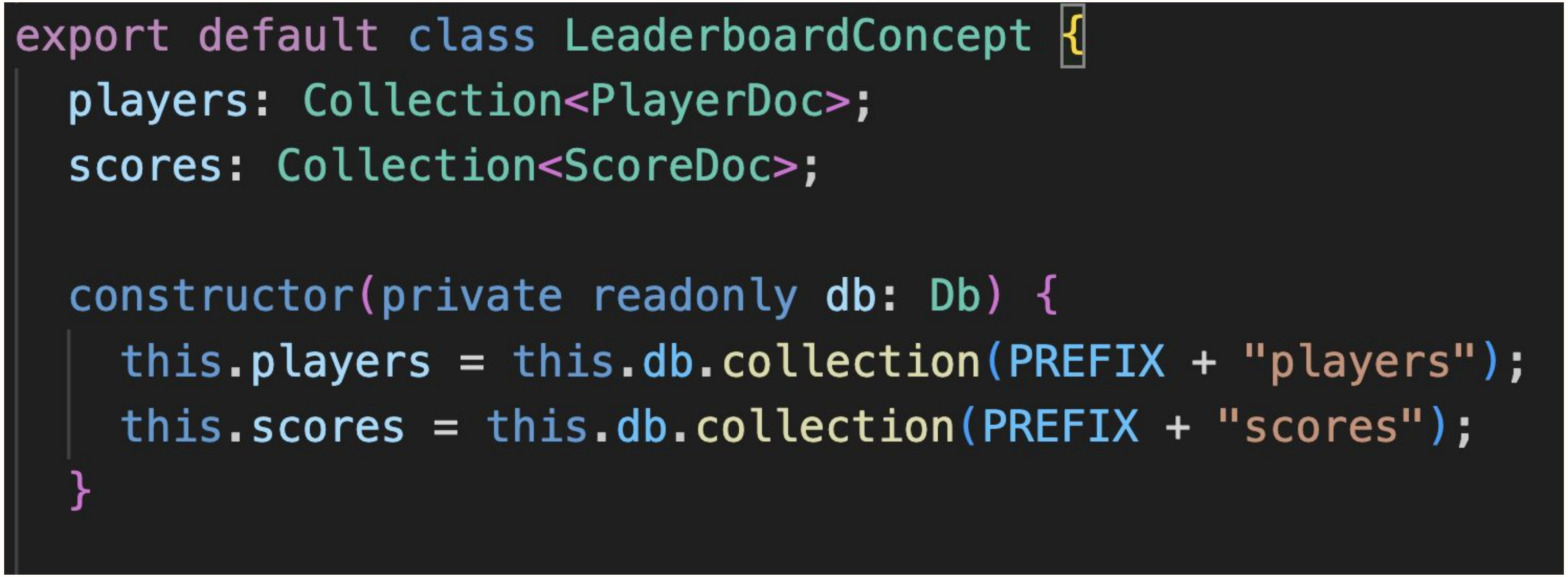
```
/**
 * a set of Players with
 *   a name String
 */
interface PlayerDoc {
  _id: Player;
  name: string;
}
```



```
/**
 * a set of Scores with
 *   a player Player
 *   a value Number
 *   a submittedAt Date
 */
interface ScoreDoc {
  _id: Score;
  player: Player;
  value: number;
  submittedAt: Date;
}
```

- documents are the unit of storage in MongoDB
- when using your own IDs, you specify with `_id`

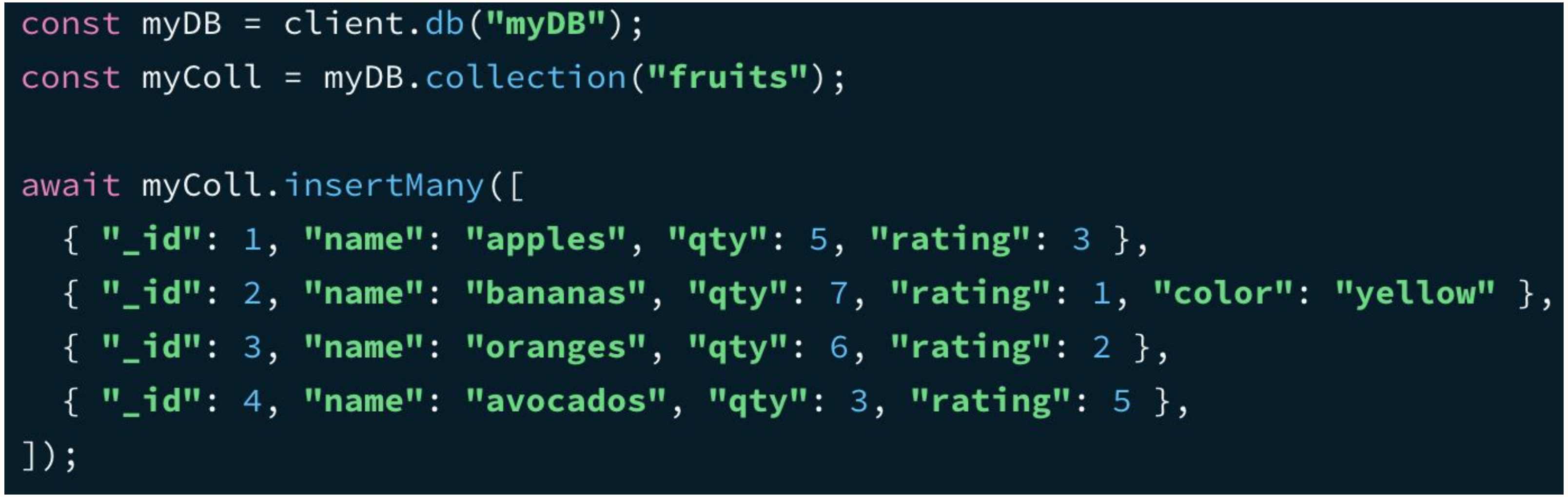
MongoDB: collections



```
export default class LeaderboardConcept {
  players: Collection<PlayerDoc>;
  scores: Collection<ScoreDoc>;

  constructor(private readonly db: Db) {
    this.players = this.db.collection(PREFIX + "players");
    this.scores = this.db.collection(PREFIX + "scores");
  }
```

Querying in MongoDB

- query operators allow for various logical checks
- suppose following data:



```
const myDB = client.db("myDB");
const myColl = myDB.collection("fruits");

await myColl.insertMany([
  { "_id": 1, "name": "apples", "qty": 5, "rating": 3 },
  { "_id": 2, "name": "bananas", "qty": 7, "rating": 1, "color": "yellow" },
  { "_id": 3, "name": "oranges", "qty": 6, "rating": 2 },
  { "_id": 4, "name": "avocados", "qty": 3, "rating": 5 },
]);
```

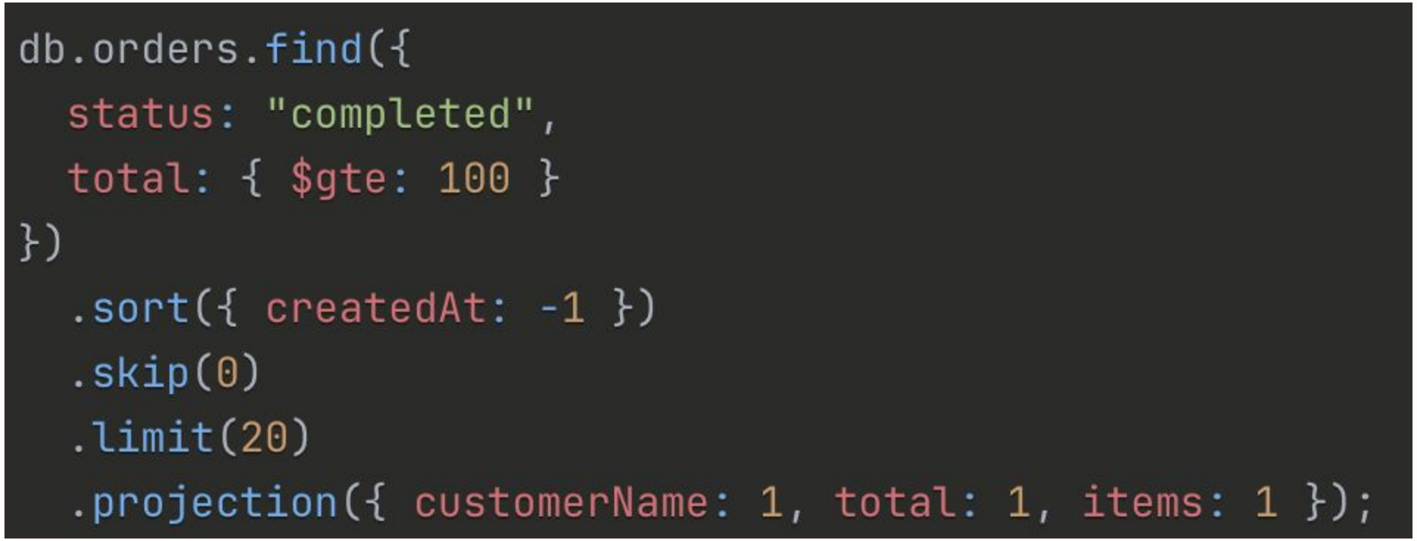
Querying in MongoDB

```
const query = { qty: { $not: { $gt: 5 }}};
const cursor = myColl.find(query);
for await (const doc of cursor) {
  console.dir(doc);
}
```

```
{ "_id": 4, "name": "avocados", "qty": 3, "rating": 5 }
{ "_id": 1, "name": "apples", "qty": 5, "rating": 3 }
```

- many different operators for comparison, logical, etc.
- `find` returns a cursor, which we can navigate further

find cursor



```
db.orders.find({
  status: "completed",
  total: { $gte: 100 }
})
  .sort({ createdAt: -1 })
  .skip(0)
  .limit(20)
  .projection({ customerName: 1, total: 1, items: 1 });
```

- cursor methods like `sort, skip, limit, projection`

  are also powerful at getting what you want

trying it out

- after copying `mongo-recitation` into your repo (more instructions in repo)
- in `src/recitation`
  - edit `LeaderboardConcept.ts`
  - iterate with `deno test -A`
- solve 02...test.ts through 08...test.ts
  - 09...test.ts is a challenge!

getting help and recording notes

- you can use `design/learning/recitation.md` as a template for a good amount of context for prompting
- otherwise, feel free to take notes (and `ctx save` them) about what you learn so you can refer to them later