Databases for Data Science

Workshop 09a · 2022-09-26

Today

- Local setup for PSQL and MongoDB
- MongoDB
  - CRUD operations
  - Map-reduce

Local installations: PostgreSQL

We'll do this in the terminal.

- Windows: Set up WSL, then follow the Ubuntu Linux instructions
  - Run `wsl --install -d Ubuntu-20.04` in the terminal to set up WSL, if you haven't already.
    - You may need to change some Windows settings and/or reboot your computer.
    - You may need to enable virtualization in your BIOS.
    - You can install the *Remote-WSL* extension for VSCode to make working with WSL easier.
- Windows non-WSL fallback: https://www.postgresql.org/download/windows/
- Linux / WSL users: via package manager
  - e.g. `sudo apt-get -y install postgresql` for Ubuntu
- Mac users: via Homebrew
  - If you haven't installed Homebrew yet, see <brew.sh>
  - Run `brew install postgresql@14` in the terminal

Starting PostgreSQL server

The server must be started before it can be used.

```
// --- LINUX / WSL ---
// Start the server
sudo service postgresql start

// Create a postgres superuser for yourself
sudo -u postgres createuser YOUR_LOCAL_USERNAME_DONT_TYPE_THIS -s
```

```
// -- MACOS ---
brew services run postgresql
```

Then, verify that your setup works:

```
createdb test
psql test
```

```
-- in PSQL shell
select 1+1;
```

Local installations: MongoDB

- Mac users: via homebrew

```
xcode-select --install
brew tap mongodb/brew
brew update
brew install mongodb-community@6.0
brew services start mongodb/brew/mongodb-community
```

- Windows and Linux: use the installers (you need both of them)
  - Server: https://www.mongodb.com/try/download/community
  - Shell: https://www.mongodb.com/try/download/shell

Testing

Run `mongosh` in your terminal to enter the MongoDB shell.

- This is just a newer version of what we were using on CS1.

Run some javascript to test it out.

MongoDB | Importing data

Let's set up the Airbnb data locally.

To download the file:

```
curl https://raw.githubusercontent.com/neelabalan/mongodb-sample-dataset/main/sample_airbnb/listingsAndReviews.json > listingsAndReviews.json
```

Or, visit https://github.com/neelabalan/mongodb-sample-dataset/blob/main/sample_airbnb/listingsAndReviews.json and click **Download**.

Import / export

- `mongoexport` produces JSON files
- `mongoimport` reads them.

```
//source: https://gist.github.com/john-guerra/1554ce7ca4bb8248b715d484e1db03f3
mongoimport -h localhost:27017 --db sample_airbnb --collection listingsAndReviews --file listingsAndReviews.json
```

MongoDB | Inserting data

Navigate to your own database:

- `> use [YOUR NAME]`

Create a new collection:

- `> db.createCollection(students)`

You can also create a collection *implicitly* by adding data to it!

MongoDB | Inserting data

It's pretty simple:

```
// db.collection.insert( <object> )

> db.students.insert({"name": "Alex");
> db.students.insert({"student_id": "N10234567"});

// Use an array to insert several objects
> db.students.insert([{"name": "Charlie"}, {"name": "Dylan"}])
```

At this point, you can put *anything* in any collection.

- Why might this be helpful?
- Why might this be a problem?

MongoDB | Inserting data

JavaScript can be useful for automating tasks.

```
> var names = ["Alex","Bryce","Charlie","Dylan"];
> var new_students = [];
> for (var name in names) {
  new_students.push({name:"name"});
}
> db.students.insert(new_students);
```

Exercise: Use JS to generate and insert your own set of students. Assign each one a unique email address.

MongoDB | Update

Where we had `.find(filter, projection)`, we now have `.updateMany(filter, command)`.

```
> db.student.updateMany({"name": "Alex"}, { $set: {"gpa": 4.0}});
```

Exercise: Assign random GPAs to all of your students.

MongoDB | Delete

Similarly to a `DELETE FROM tablename WHERE condition`,
we have `.deleteMany(filter)`.

```
// Remove matching records
> db.student.deleteMany({"name": "Bryce"});

// Remove all records
> db.student.deleteMany({});
```

Exercise: Delete all records for students with GPA below 1.0.

MongoDB | Queries

To get documents matching a condition, add a *filter* to the `find()` call:

```
// Match a top-level property
> db.listingsAndReviews.find({property_type: "House"})

// Match a nested property
> db.listingsAndReviews.find({"address.country_code": "US"})

// Match multiple properties
> db.listingsAndReviews.find({property_type: "House", "address.country_code": "US"})
```

- What does this correspond to in SQL?

Exercise: Find all of the two-bedroom apartments.

MongoDB | Query Operators

We've used this syntax for *equality* filters:

```
// { <field>: <value>}
```

We can also use *operators* to define more general filters.

```
// { <field>: { <operator>: <value> } }

// Comparison
{ accommodates: { $gt: 5 } }

// Inclusion
{ "address.country_code": { $in: ["US", "CA"]}}
```

MongoDB | Query Operators

Full list: https://www.mongodb.com/docs/v4.4/reference/operator/query/

Exercise: Find all of the listings with a rating above least 90.

Exercise: Find all of the listings with more than two bedrooms and a weekly price of less than $1000.

Exercise: Find all of the condos and apartments in the US whose host is named "Alex".

MongoDB | Query Operators

Queries on array contents:

```
// Match one array item
db.listingsAndReviews.find({"amenities": "Internet"})

// Match two array items
db.listingsAndReviews.find({"amenities": {$all: ["Internet", "TV"]}})

// Match *exact* array contents (this will be empty!)
db.listingsAndReviews.find({"amenities": ["Internet", "TV"]});

// Match document within array
db.listingsAndReviews.find({"reviews.reviewer_name": "Xavier"});
```

Exercise: Find all of the apartments with TV *or* internet service.

MongoDB | Projection

We can also choose which parts of each document to return.

```
// Get the street address and bedroom count of every listing
db.listingsAndReviews.find({}, {"address.street": 1, bedrooms: 1});

// Get the amenity list and review score for every listing without internet
db.listingsAndReviews.find(
  {"amenities": {$ne: "Internet"} },
  {"amenities": 1,"review_scores.review_scores_rating": 1}
);
```

- What does this correspond to in SQL?

Exercise: Get the host name of every listing in Brazil.

MongoDB | Map

We can apply a function to every item in the output. (Or in an array, etc.)

```
// Get the first letter of each string
['Alex','Charlie'].map(s=>s.substring(0,1))

// Apply a more complicated function
db.listingsAndReviews.find().map(listing => {
  let count = listing.reviews.length;
  return `${listing.name} has ${count} reviews`;
})
```

Exercise: Write a function to decide whether you would consider staying at a given listing. Use `map` to apply this function to all listings.

MongoDB | Documentation

Manual:
https://www.mongodb.com/docs/v6.0/

Cheat sheet:
https://www.mongodb.com/developer/products/mongodb/cheat-sheet/