Deliverables

The deliverable for this homework is a web site that operates as a front-end to a MongoDB database.

Pair Programming

You are required to work with your assigned partner on this assignment. You must observe the pair programming guidelines outlined in the course syllabus — failure to do so will be considered a violation of the Davidson Honor Code. *Collaboration across teams is prohibited and at no point should you be in possession of any work that was completed by a person other than you or your partner.*

1 Introduction

In this assignment, we will obtain un-normalized data from Jeff Sackmann's ATP Tennis dataset, and we will build a website that performs queries and displays information. Here are the goals:

1. Implement a start-to-finish database system: from data (un-normalized) to presentation (user-facing website);
2. Create a database from a NoSQL perspective using MongoDB.

1.1 Material

The material a set of CSV files that can be obtained from

`https://github.com/JeffSackmann/tennis_atp`

Download the whole set of files, but focus at the `atp_matches_XXXX.csv`, where XXXX is the year from 1968 to 2018.

Each file contains, among other fields, a tournament name and date, surface type (Clay, Hard, etc), and information about the players who won or lost (name, hand, height in cm, country of origin, rank). Each match has a tournament, and date. There's also the score of the match in the "score" column.

2 (40 pts) Importing the dataset into MongoDB

To import the dataset into MongoDB, you should read the information using your preferred language, although this should be more effectively done in a language like Python. With Python you can install the MongoDB driver, and import the data directly to the database using `pymongo`, documented here:

`http://api.mongodb.com/python/current/tutorial.html`

To install `pymongo`, it should be enough to run

```
python -m pip install pymongo
```

in your command terminal.

2.1 Normalizing the Dataset

One particular problem with CSV data sets is that they do not have enough structure to support normalization like a database. You should read the un-normalized dataset from our source and:

1. Identify all tournament names/dates, match surface types and the tournament they refer to, and players together with their information, and create separate collections for each of them, noting that each of these are *entities* in your application.
2. Note that each match refers to a specific surface type, and a specific tournament. You should use the analogous of *foreign keys* in your MongoDB database.
3. Create new IDs for your entities. The CSV files have IDs for players, tournaments, etc. You **will not** use these IDs, but create your own as you add your entities into MongoDB.

Deliverable 1. A program called `DataImporter.py` (or `DataImporter.X`, where X is the extension of your favorite language) that imports the dataset into MongoDB. Note that I'd like this program to insert the information in the database using **one pass**, that is, you should not read any line of the CSV file more than once.

3 (50 pts) A Web Frontend with MongoDB and PHP

Your users would like to obtain statistics about players, tournaments, counties, etc, and they would like to query your database using a website. In this part of the homework, you will exercise:

1. Your ability to learn new tools and languages given only documentation;
2. Your sense of style in designing a front-end application[1]
3. Technically, you will see how websites, databases, and users inter-operate.

3.1 The Tools of The Trade

Webpages are written using HTML. A short, time-efficient tutorial can be found here:

`https://www.w3schools.com/html/default.asp`

Particularly, look at the "HTML Forms" section as they will be quite useful in your website interface. HTML files are hosted in a *web server*, that fetches local files upon request, and transfers them to the other side of the connection, where they are interpreted by *browsers*. The web server can also execute *server-side* scripts contained in these HTML files as they read them, so the files obtain data dynamically from a local or remote database. The web server reads the server-side script, and amends the HTML file with dynamic contents, typically obtained from a database.

A common server-side scripting language is PHP, which is embedded in HTML files as described before. To install PHP on macOS, run in the terminal:

```
brew update
brew upgrade mongodb
brew install php
brew install composer
```

If `brew install php` fails because you have PHP, do a `brew upgrade php` instead.

In the same directory where your PHP files reside, run

```
pecl install mongodb
composer require mongodb/mongodb
```

[1] If you are taking Software Design, this should be familiar.

If your installation of `composer` fails because of a "JIT compilation error", please edit the file `/usr/local/etc/php/7.3/php.ini` (adjust for your PHP version), and add the line "php.jit=0" (no quotes, no spaces) to the beginning of the file. On Windows, you can download PHP from `https://windows.php.net/download`.

3.2 Test your PHP

To test your PHP installation, you should run a web server. Fortunately, PHP7 comes with a web server of its own, and this assignment comes with two PHP examples, one connecting to a MySQL database `test_mysql.php`, and one connecting to MongoDB (`test_mongodb.php`, which you particularly care about).

In the same directory where your PHP files reside, run

```
php -S localhost:3333
```

which will launch a web server at the local machine, port 3333. In your browser, type:

```
http://localhost:3333/test_mongodb.php
http://localhost:3333/test_mysql.php
```

For the MySQL test, you have **first** to setup your Activity 4 database and adjust username and password in the file, if necessary.

For the MongoDB test, you have **first** to create the database and insert some data on it. Please run the following in your `mongo` client to setup your MongoDB for your test file:

```
use TestDB;

db.students.insertMany([
  { name: "Alice", major: "Computer Science" },
  { name: "Bob", major: "Mathematics" },
  { name: "Carlos", major: "History" },
  { name: "Denise", major: "Data Science" },
  { name: "Eduardo", major: "Undeclared" },
  { name: "Fabiana", major: "Physics" }
]);
```

Please inspect these files' syntax - they should be treated as setup documentation.

Here are the PHP tutorials you might need to read:

`https://www.w3schools.com/php7/` (read the "PHP7" and "PHP7 Forms" sections)
`https://www.php.net/manual/en/mongodb.tutorial.library.php`

Please note that there is a whole section concerned with connecting to MySQL, which could be useful for your final project (but not for this one, as you are using MongoDB).

Also, using JavaScript will make your job a lot easier, since it has tools to sort tables, search in tables, etc. Here is your go-to tutorial for JavaScript:

`https://www.w3schools.com/js/default.asp` (read just the "JS Tutorial" part)
`https://www.w3schools.com/howto/howto_js_filter_lists.asp`
`https://www.w3schools.com/howto/howto_js_filter_table.asp`
`https://www.w3schools.com/howto/howto_js_sort_list.asp`
`https://www.w3schools.com/howto/howto_js_sort_table.asp`
`https://www.w3schools.com/howto/howto_js_filter_dropdown.asp`

3.3 Functional Requirements

Your web frontend should use forms and make available for the user the following queries:

1. Display all information[2] from a player, and allow the user to filter/sort by any of these attributes;
2. Display the general attributes of a tournament (name, surface, date);
3. Display all games of a certain tournament, with winner/loser, and score. Each player or tournament is displayed as link that, upon a click, brings the user to the player or tournament information.

Deliverable 2. A directory called **www** containing all the PHP files (containing HTML) that support your webserver. For my test, I will run `php -S localhost:3333` in your directory and connect with my browser to `localhost:3333/index.php`

Please compress the `www` directory and the data importer program into a single file and submit it to Moodle.

[2] As defined in Sec. 1.1

4 (10 pts) Style

The remaining 10 points are granted for meeting the following style guidelines. The writeup is organized and properly written. The relational algebra notation is clear. The SQL scripts are well-indented and organized. The SQL scripts are concise and elegant, and do not use too many unnecessary Cartesian operations.

Good luck,
- Hammurabi