

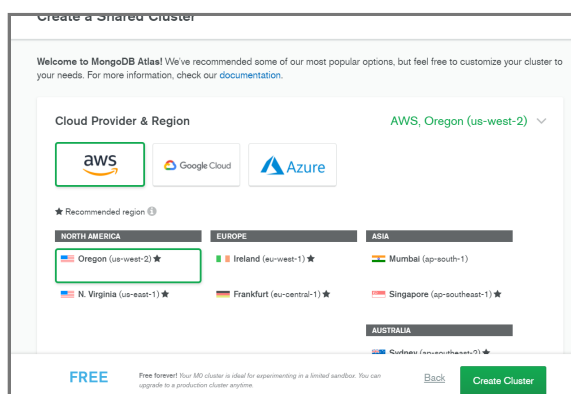
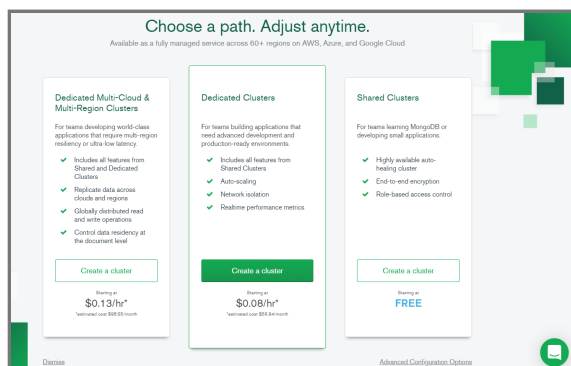
HW4: NoSQL

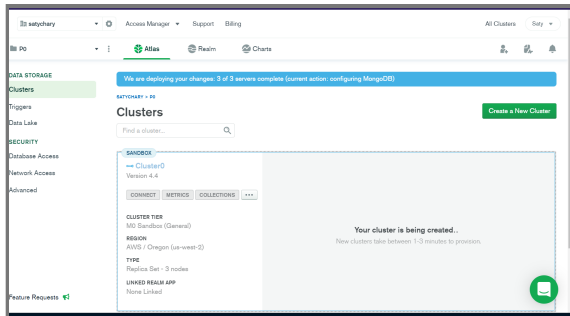
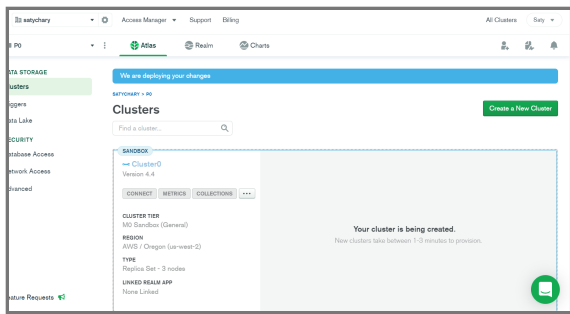
Total points: 6

This HW is going to get you familiar with working with NoSQL, ie. use JSON data! While the data we play with are small, the same software, and steps, apply to data that might be a billion times (!) or more bigger.

Specifically, you will be using MongoDB Atlas - a cloud-based installation of MongoDB, which means there is nothing to install :) You'll work within the free tier, which gets you a cluster of 3 nodes, 500G of storage space, etc, which is quite adequate for this HW and beyond (please DO continue learning more Mongo commands after the course).

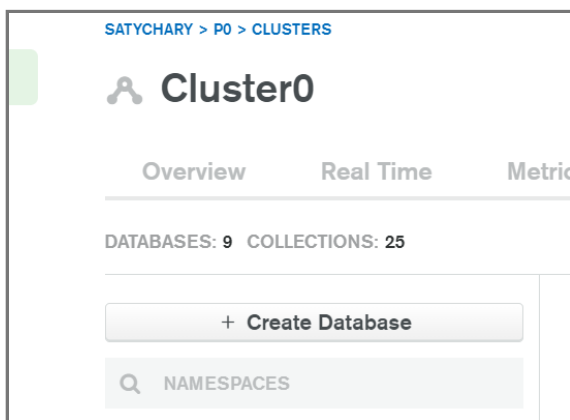
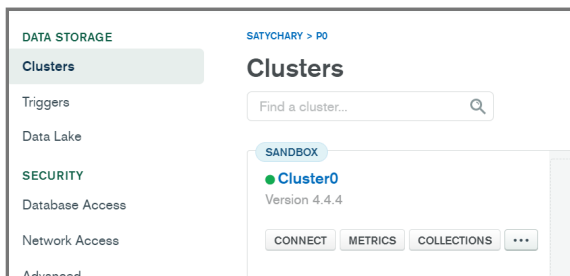
Start by signing up: <https://www.mongodb.com/cloud/atlas> ['Try Free' at the top right, or, 'Start free'].





Once you sign in, you'll get your cluster set up, like shown above. There is no need to set up any further clusters at all, just work with the one that just got set up.

Click on 'COLLECTIONS', then 'Create Database', call it HW4DB (for ex):

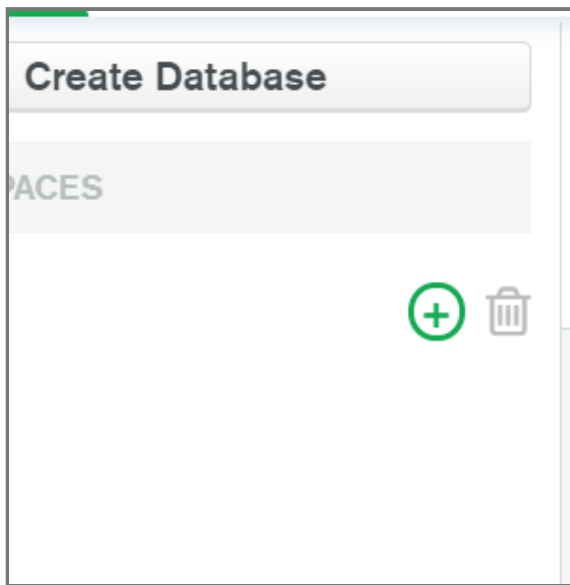


Remember - in NoSQL, a 'database' holds a set of 'collections', and a 'collection' holds a set of 'documents' - this is similar to how, in RDBMS, a database contains a set of tables, each table contains rows. So a NoSQL document (which is a single JSON object)

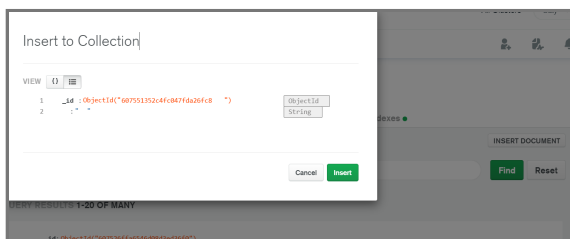
is equivalent to a table row, and is a complex version of a row in fact (can contain nested data, multi-valued data). An array of such documents is what constitutes a collection (ie. a table).

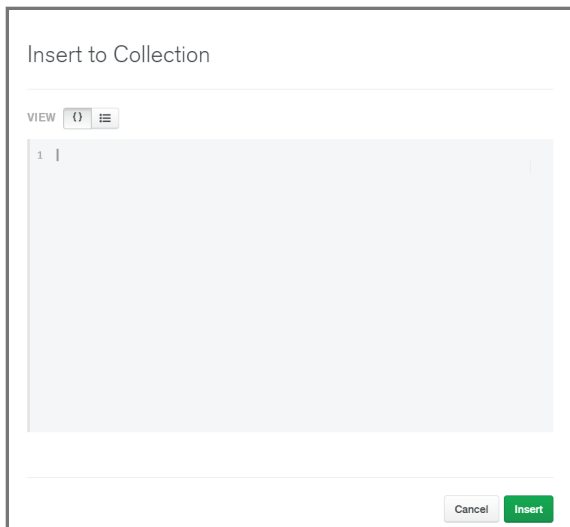
OK, now that you have a blank DB, let's add collections [of documents] to it, and do queries :)

Create a collection ('table') called 'HW3Data' - yes, you're going to (re)use your locations data from HW3 :) The '+' sign to the right of your DB's name, helps you create a collection (we will create several for this HW):



All right, we have a blank collection, next we need to "insert rows" (add documents). All the way over on the right of the page, there is INSERT DOCUMENT, click on it. That brings up the following popup, for inserting (typing in) just one document (Mongo inserts a row-key/PK). You'll be inserting 12 (locations), so, click on the {} JSON icon :) Clean out what's there, to end up with a BLANK slate :)





Insert your data, like so (an array of objects, each object contains a location you collected):

```
[
  {
    "name": "Tommy Trojan",
    "popularity": 100,
    "loc": [34.02, -118.28]
  },
  {
    ...
  },
  {
  }
]
```

As you can see above, you'll supply (create) a 'popularity' key for each location - give it a value between 0-100 :)

Type/paste the data, then click 'Insert' - voila! Your documents are now in the collection.

To query, you'd enter a 'query filter' - see the example below (the data is NOT the data you are using):

FILTER {"filter": "example"}

QUERY RESULTS 1-20 OF MANY

```
_id: ObjectId("607526ffa6546d08d3ed36f0")
type: "Feature"
id: 0
> properties: Object
> geometry: Object
```

```
_id: ObjectId("607526ffa6546d08d3ed36f1")
type: "Feature"
id: 1
> properties: Object
> geometry: Object
```

```
_id: ObjectId("607526ffa6546d08d3ed36f2")
type: "Feature"
id: 2
> properties: Object
> geometry: Object
```

FILTER {"id": 2}

QUERY RESULTS 1-1 OF 1

```
_id: ObjectId("607526ffa6546d08d3ed36f2")
type: "Feature"
id: 2
> properties: Object
> geometry: Object
```

Read up (there are tutorials right there on Mongo's site for ex) on writing queries; look up 'GeoJSON' queries too, that's what you'd use, for Q3, Q5, Q6 below. Mongo queries may be simple-looking but they are powerful!

Q1 (1 point): write a (simple!) query to output locations (documents) with a popularity of ≥ 50 . Take a screenshot of the query and results.

Next, modify your HW3's Spirograph code, to output data in the above format, but without 'popularity' and name, ie. like so:

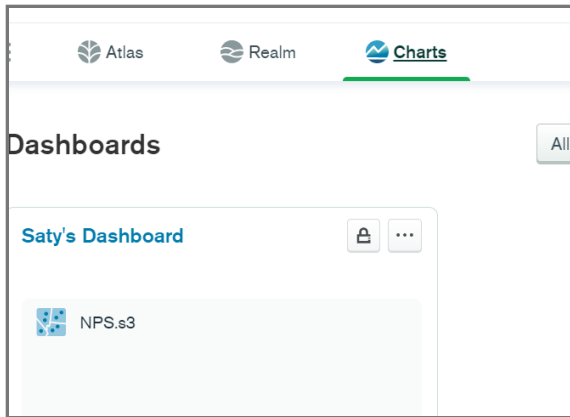
```
[
  {
    "loc": [-118.28886097539089, 34.02121762509463]
  },
  {
    "loc": [-118.28886090156676, 34.02121425075701]
  },
  ...

  {
    "loc": ...
  }
]
```

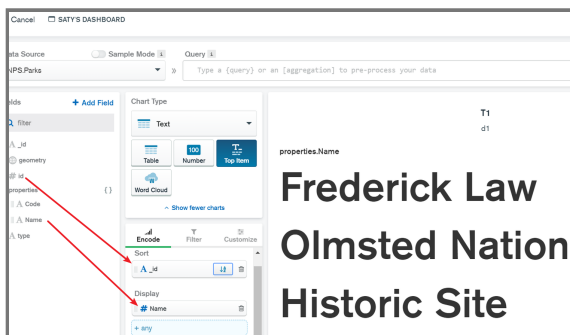
Create a second collection (in the same DB) called 'Spiro', and insert the above data into it [you can have the step size be 0.1 so that it's not too much data - not that Mongo can't handle it (it can!), it's just that you don't need it].

Create a third collection called 'NPS' [<https://www.nps.gov/>], grab this data, add it (just the features array): <https://www.nps.gov/lib/npm.js/4.0.0/examples/data/national-parks.geojson> [optionally, you can paste the JSON on the LHS here, to format it, and to view it as a tree: <https://bytes.usc.edu/~saty/tools/jsoned/index.html>]

Now, we're going to visualize data, and do spatial querying (and see the results visually). Click on 'Charts', then create a dashboard (which will hold your charts, ie visualizations), select it - now you can 'ADD CHART' to create multiple types of visualizations of your documents/rows :)



Q2 (1 point). Visualize your 12 locations (ie the HW3Data collection), take a screenshot. To visualize (create a chart), you'd pick a chart type, then drag and drop document columns (keys) on to the chart's slots, like so:



Q3 (1 point). Write a query (in the chart area) where you would specify a bounding box (lower-left location, upper-right location) to display just the northern half of your 12 locations. Take a screenshot of the query and the result.

Q4 (1 point). Bring your Spirograph data into a new map :) Take a screenshot.

Q5 (1 point). Specify a triangle (three locations) to display just the Spiro coords inside your triangle. Take a screenshot of the query and result (this is a cool capability, to bound spatial search results by specifying an arbitrary polygon).

Q6 (1 point). Bring in all the national parks into another map - cool! All 396 are visible, on a map of the entire US. Write a query to output just the western half locations (west of the Rockies). Create a bounding box to do this (like in Q3). Take a screenshot.

You're done! Submit (in a .zip), Q1.{jpg,png} through Q6.{jpg,png}.

Have fun! Now you know how to insert JSON docs into MongoDB, query, visualize, do spatial queries. Neat :)

