**How to Ingest Data From MongoDB with Node.js**

Data ingestion from a NoSQL database often involves the denormalisation of their schema-less data before loading into a relational database. In this post, we will try to grab the restaurant data from MongoDB, denormalise the collection into a parent and a child table, and load them into Postgres.

This exercise has been previously done with Python. The post includes how to get the restaurants dataset and load it to MongoDB. If you need a little bit of guidance on the preparation, you can refer to [**How to Get Data From MongoDB with Python**](https://www.mydatahack.com/how-to-get-data-from-mongodb-with-python/).

If you need a little bit of MongoDB refresher, check out the posts below:

* [MongoDB Skills Essential Exercise](https://www.mydatahack.com/mongodb-skills-essential-exercise/)
* [User Management with MongoDB](https://www.mydatahack.com/user-management-with-mongodb/)
* [MongoDB CRUD Operation with Node.js](https://www.mydatahack.com/mongodb-crud-operation-with-node-js/)

**Dataset**

We are using the restaurant dataset from [here](https://github.com/OpenKitten/Mongo-Assets/blob/master/primer-dataset.json). Download it as a JSON file and upload it to MongoDB (see [here](https://www.mydatahack.com/how-to-get-data-from-mongodb-with-python/) for detailed instruction).

**Data Model**

We are going to denormalise the dataset into the main restaurants table as a parent and the grades table as a child. Data model is below. For further information including SQL create table statements, see the previous post [here](https://www.mydatahack.com/how-to-get-data-from-mongodb-with-python/).

**Code**

Our ingestion strategy is:

* Get data from MongoDB.
* Transform the format of the timestamp field for the child table.
* Convert it into two csv files
* Use the Postgres copy function to bulk insert data from the csv files.

First of all, we will create a module to perform truncate & load operations. The function is straight-forward. It takes csv file and upload it to a target table. Save it as copyPg.js. At the end of the file, we can export it with the module.export function.

I have covered the topic of bulk-loading data into Postgres with Node.js previously. It discusses the modules used for bulk-loading as well as using the config.json file to define connection details. Check out this post if you need a refresher ([Converting JSON to CSV and Loading it to Postgress with Node.js](https://www.mydatahack.com/converting-json-to-csv-and-loading-it-to-postgres-with-node-js/)).

<pre class=" toolbar:2 lang:js decode:true" >

</pre>

Once, we have the custom module to load data, let’s write the main part. We first connect to MongoDB with the [mongodb](https://www.npmjs.com/package/mongodb) module. Then query the collection. The data from the query results is already a JSON object. Therefore, we can pass it to the json2csv converter without data type conversion.

Before passing it to the converter, we are going to reformat the date field from the ISODate format to the YYY-MM-DD HH:MM:SS format. The code snippet below is looping every date field in every grades array to pass the format functions.

<pre class=" toolbar:2 lang:js decode:true" >

</pre>

To convert JSON into csv, I am using the [json2csv](https://www.npmjs.com/package/json2csv) module. It makes JSON manipulation easier and covers most of the conversion scenarios.

Finally, I am using the callback function to ensure the truncate & load event happens sequentially for two tables. The rest should be relatively simple.

<pre class=" toolbar:2 lang:js decode:true" >

</pre>