**How to Ingest FullStory Data Export Extracts with Python**

If you are interested in user tracking on your website, [FullStory](https://www.fullstory.com/features/) is a pretty good option. You can sign up for the free version [here](https://www.fullstory.com/free/). The [free version](https://help.fullstory.com/using-ref/free) includes heaps of cool features. When you first sign up, you can try all the Pro Edition features for 2 weeks, too.

From data analytics perspective, FullStory gives us the user behaviour data as it collects clicks on the html elements. Analysing data will give us more insights into customer experience on your website and more. The first step is to ingest the data from it.

FullStory has a simple REST API to extract the click data. First, you retrieve a list with data export ids after a target date. By using the id, you can extract the actual click data. Once you make the GET request, the data comes in the gz format. So, all you have to do it to write it to a gz file from the results of the API request. Here is the [official API documentation](https://help.fullstory.com/develop-rest/data-export-api).

In this example, we are loading JSON data into Postgres database (as described [here](https://help.fullstory.com/using/170703-where-can-i-find-my-api-key?from_search=17327579)). You need to have Postgres table with one column with the jsonb data type as the target table.

[cc lang="sql" tab\_size="4" lines="-1" theme="mac-classic"]

Create Table fullstory.data\_export (data jsonb);

[/cc]

**Code**

The code is written in Python 3.

1. **Required modules**

First of all, let’s import all the required modules.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

1. **Set required variables**

API credential needs to be created (see the instruction here). We are going to set variables for Postgres connections.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

1. **Converting local date to UNIX Epoch**

This function converts local date to the epoch timestamp format. For further timestamp conversion into epoch with Python, you can check out the reference [here](https://www.mydatahack.com/how-to-convert-non-utc-timestamp-into-unix-epoch-time-in-python/).

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

1. **Obtain the list of Data Export.**

You first need to get the list of Data Export after the target date.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

1. **Get Data Export**

The function below uses the Data Export Id to retrieve the data and then creates a gz file. Once the gz file is created, it creates a JSON file that is ready for database upload. I am adding the list id to all the records. This may help for troubleshooting or testing.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

1. **Upload JSON file to Postgres**

The function simply copies the JSON file to Postgres.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

1. **Put it all together**

The get\_list() function retrieves a list of Data Exports. In the main function, we will loop the list.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

**Next Step**

You can make the ingestion pattern as incremental. For example, you can write the list to the database and retrieve the max stop time and pass it as the target date. A little bit of customisation to the code above will enable you to do incremental ingestion. See how you go!