

Data Modeling

Homework #2

March 2022

# 

# 

Data 228 - Big Data Technologies and Applications

Department of Applied Data Science

San Jose State University

**Table of Contents**

[**HW #2: Data Modeling**](#_jil8hn3kox3g) **1**

[**Data Model Description**](#_ug8xel6g8tbk) **2**

[Readings](#_7gyzr78osol6) 3

[Variables\_info](#_31pzj2mzzb2h) 3

[Sites\_info](#_rnsu4pkyjv5m) 3

[Instrument\_info](#_uxiujl3atjc7) 4

[Methods\_info](#_fesoznlbzejj) 4

[Files\_info](#_7136gneptrk2) 4

[Relationships](#_wzu0odhgc6r8) 4

[Avoiding Repetition](#_kgq5eowjpkhw) **5**

[Scalable Model](#_rt4immc8alsr) **5**

[Overview of the Software Technology](#_s0z8gxiq3res) **5**

[How we could make it easier to get data into and out](#_6jq78miq4jz4) **6**

[Design to facilitate querying and retrieval of subsets of data](#_4h6ck6zfvfqc) **6**

[Entity-relationship Diagram](#_4yrt4ss66lka) **6**

[Creating Tables](#_qe4otwgivjr6) **8**

[Inserting Data](#_83mft9a9z5zf) **9**

[SITES\_INFO](#_hyf6x6v90009) 9

[INSTRUMENTS\_INFO](#_ajb1k1ps87ga) 9

[METHODS\_INFO](#_do8ba0y09jv6) 10

[VARIABLES\_INFO](#_q2ruvd5oyt1m) 10

[FILES\_INFO](#_3ktcvvedipkg) 10

[READINGS\_INFO](#_a6lf1fa2bks1) 11

# HW #2: Data Modelling

Researchers recently deployed several instruments to continuously monitor streamflow and water quality at two sites in the Little Bear River, Cache Valley, Utah. The instruments report measurements in numerous [datalogger and metadata files](https://sjsu.instructure.com/courses/1474209/files/66957549/download) that the researchers must now regularly retrieve, organize, quality-control/quality-check, aggregate, and manipulate before they can perform further analysis.

* Design a data model to represent the data and its associated metadata
* Use MySQL Workbench to create an entity-relationship diagram of your data model
* Upload a one-page briefing report along with a full-page entity-relationship diagram that shows your logical model design:

–Describe the entities and relationships that you have included in your data model.

–Explain how you will structure the metadata to avoid repetition.

–Overview of the software technology, file formats, etc. you will use to organize the data and implement your data model.

–Describe how you could make it easier to get data into and out of your data model.

–Specify whether your data model design will facilitate querying and retrieval of subsets of data.

–Provide an entity-relationship diagram that shows the entities needed to describe the data, their attributes, and the relationships between them.

# 

# 

# Data Model Description

Our data model contains six tables: Readings, Sites\_info, Variables\_info, Files\_info, Instruments\_info, and Methods\_info. The main table that will be changed more frequently is Readings. It contains the different values (Readings) for each variable (VariableID) of every site (SiteID) taken at distinct times. The tables are described as follows:

## Readings

The values for this table are taken from the different readings files for each site. Each reading has an identification number that is implemented using a surrogate key. It also has the identifications for the variable and the site. Any other information about these two attributes is residing in different tables. The VariableID contains the code for the variable for which we are taking the measurement. An example of these values is Batt\_Volt\_Min. The reading is also related to a specific site that is why we included the SideID to differentiate readings from different sites. The time is also crucial to know at which time each reading was taken, so we included Data\_time in this table. Reading is the column that will contain the value read from the instrument. The VariableID is the foreign key pointing to the Variable\_info table, and the SiteID is the foreign key pointing to the SiteID table.

## Variables\_info

This table contains the information about each variable. The VariableID will get the column name for different variables taken from the files. The VariableID will be used to identify each variable. This ID will be used as a primary key that will be referred to by the Readings table. Information for each variable was captured from the Metadata. Each variable from the table has the following information: VariableID, Variable\_Name, Units, Sample Medium, Value Type, Time\_Support, Data\_Type, No\_Data\_Value, InstrumentID, and MethodID. The Instrument\_ID is the key given for the material used to take the measurement for a particular variable, and the MethodID is the key given for the method used to collect the data. Instrument\_ID and MethodID are the foreign keys pointing to the tables Instruments\_info and Methods\_info respectively.

## Sites\_info

This table contains the information about each site. The information for each column can be found in the metadata file. A number was assigned to each site to be able to use it as a primary key referred to from the Readings table and the Files\_info table. The information for the site is SiteID, Site\_Name, Latitude, Longitude, Elevation, State, County, and Comments.

## Instrument\_info

This table contains information about the instruments used for the measurements. Each instance of the table contains an InstrumentID, an InstrumentName, and the brand. This table has a primary key InstrumentID that is referred to by the Variables\_info table.

## Methods\_info

This table contains information about the different methods used to collect data. This table has two columns: MethodID and MethodDescription. The methodID was created as a primary key to be referred to by the Variables\_info table.

## Files\_info

This table contains information about the data files containing all the data. A FIleID is assigned with each file. In addition to FIleID, this table also has the columns FileName and SiteID. The FileName is the name of the file containing all the data, and SiteID is the id assigned to the Site. This SiteID column is a foreign key referring to the Sites\_info table.

## Relationships

The relationship between SITES\_INFO and READINGS\_INFO is 1:M

The relationship between VARIABLES\_INFO and READINGS\_INFO is 1:M

The relationship between METHODS\_INFO and VARIABLES\_INFO is 1:M

The relationship between INSTRUMENTS\_INFO andVARIABLES\_INFO is 1:M

The relationship between SITES\_INFO and FILES\_INFO is 1:M

# Avoiding Repetition

We modeled the metadata to make sure each value is not repeated to avoid inconsistency problems. For instance, we noticed that many variables are measured using the same instrument, so instead of repeating the name of the instrument for each variable we created a table to hold only the instruments used overall. The variable will only refer to the instrument used using the foreign key InstrumentID.

Another example is the methods table that was created to have all the methods used for measurement. The variable will only refer to the method used stored in MethodID since many variables have the same method.

# Scalable Model

Our model is adjustable to any element addition. For instance, If there is a new variable that needs to be added, we only need to add a row to the Variables\_info table and refer to it from the readings table when values are inserted for the new variable.

Like adding variables, this model is built to be able to add new files, sites, instruments, and methods without the need to change the whole model. If we had fixed columns with the existent variables, for example, adding a new variable would be costly!

# Overview of the Software Technology

We Used Draw.io to create our model and build the relationships between the entities. Then we implemented the database using MySQL Workbench. We created all the different tables with their relationships using SQL Scripts shared later in this document. We manually entered data for the metadata since it is a readme file. For the datafiles, we used Python to clean the .dat files for both sites, transferred the clean files into a .csv file, connected to the HYDROINFORMATICS database, which we created on MySQL, using the mysql.connector library, and loaded the data into the Readings table using a python script we implemented.

# How we could make it easier to get data into and out

We could make it easier to get data into and out of your data model by using Microsoft Azure Data Factory. Here we will be creating scheduled workflows (Pipelines) that will execute python scripts (stored in Azure Blob Storage container) to directly retrieve data from the relevant sites and inject the data into our Data Warehousing system.

# Design to facilitate querying and retrieval of subsets of data

This design will allow data to be easily retrieved as different information of different objects is stored in separate tables. For instance, if we want to print the list of all the used variables, or used methods, we only need to query the appropriate table. Also because the VariableID is now a row and not a column we can query each variable by retrieving and doing statistics on one variable at a time without worrying to transfer whole instances to memory with all different variables as columns. The query will be fast and effective.

# Entity-relationship Diagram

In the initial phase, we utilized draw.io (free online diagram software) to design our data model to represent the data and metadata as described in Figure 1. Once our model is designed, we created the database and its respective tables in MySQL and used the reverse engineering technique to generate an entity-relationship diagram as mentioned in below Figure 2.

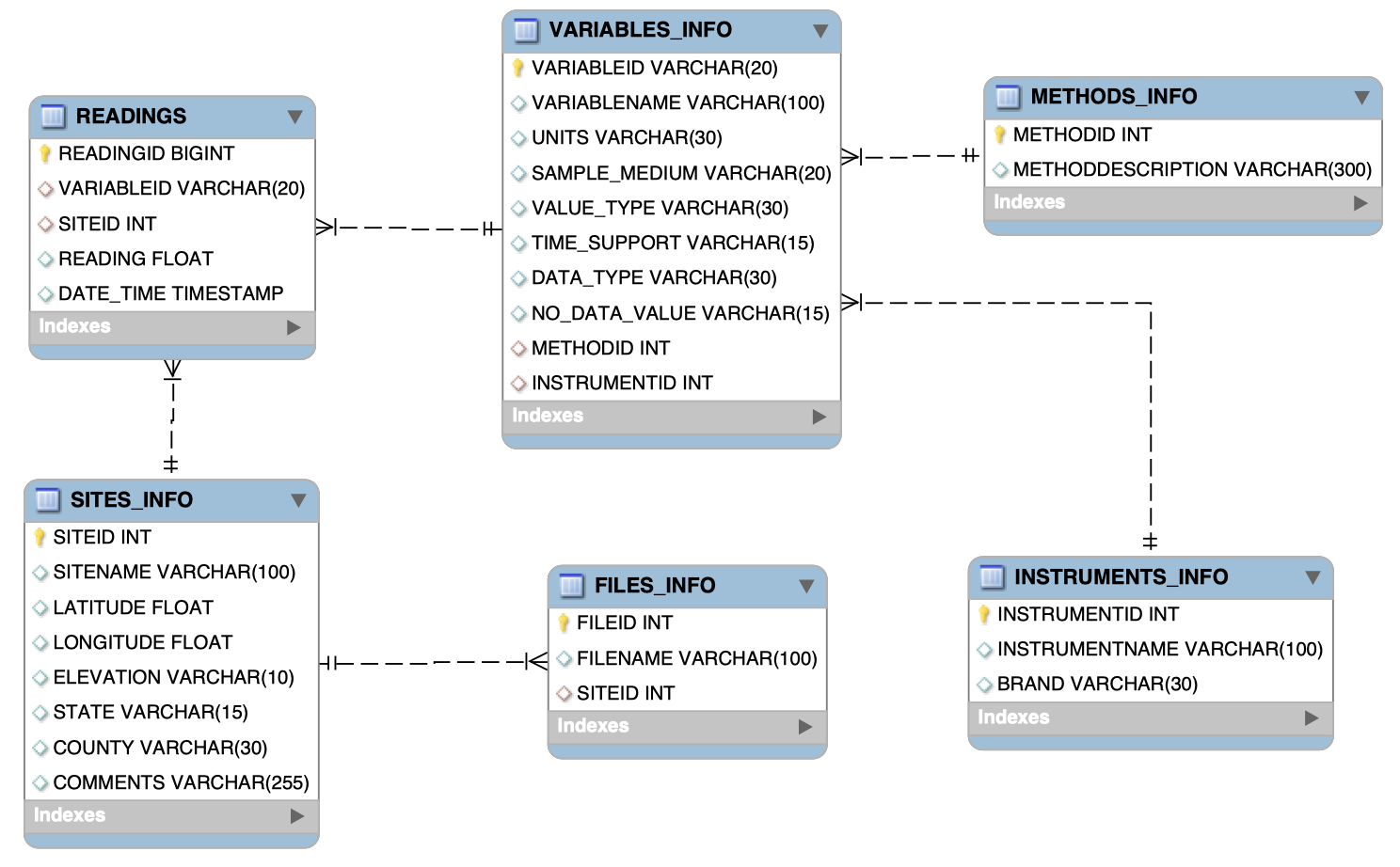
**Figure 1**

*Using Draw.io to design the data model to represent the data and metadata*

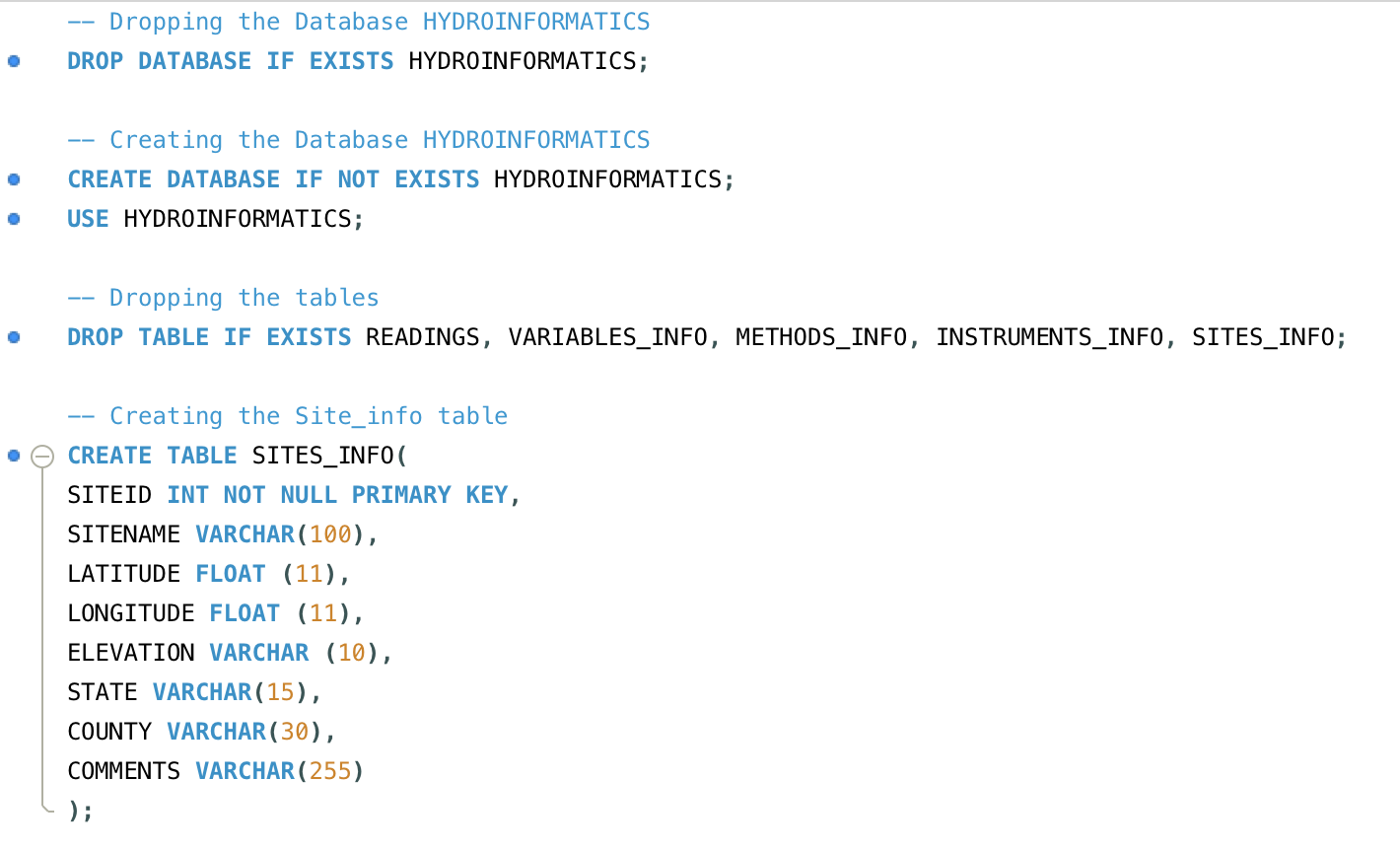


**Figure 2**

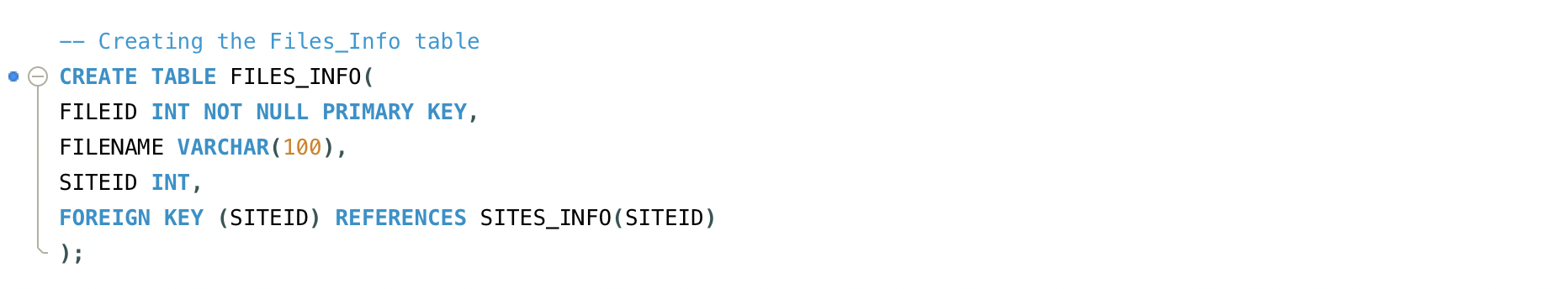
*Using MySQL Workbench to create an entity-relationship diagram for the data model*



# Creating Tables





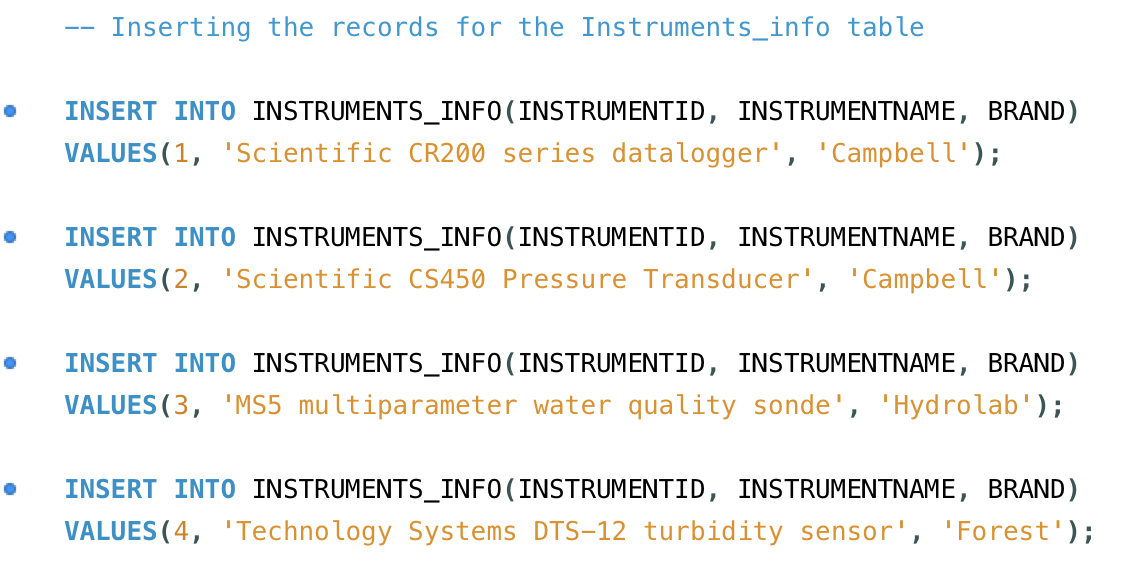
****

# Inserting Data

### SITES\_INFO



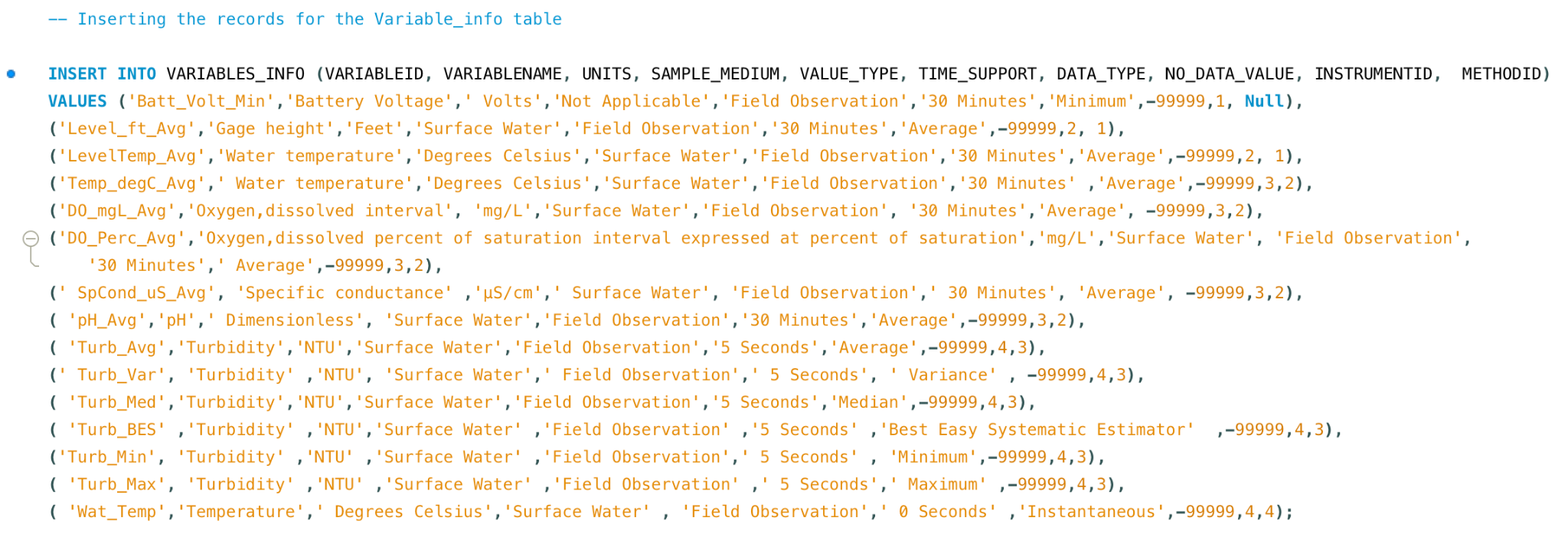
### INSTRUMENTS\_INFO



### METHODS\_INFO

### 

### VARIABLES\_INFO

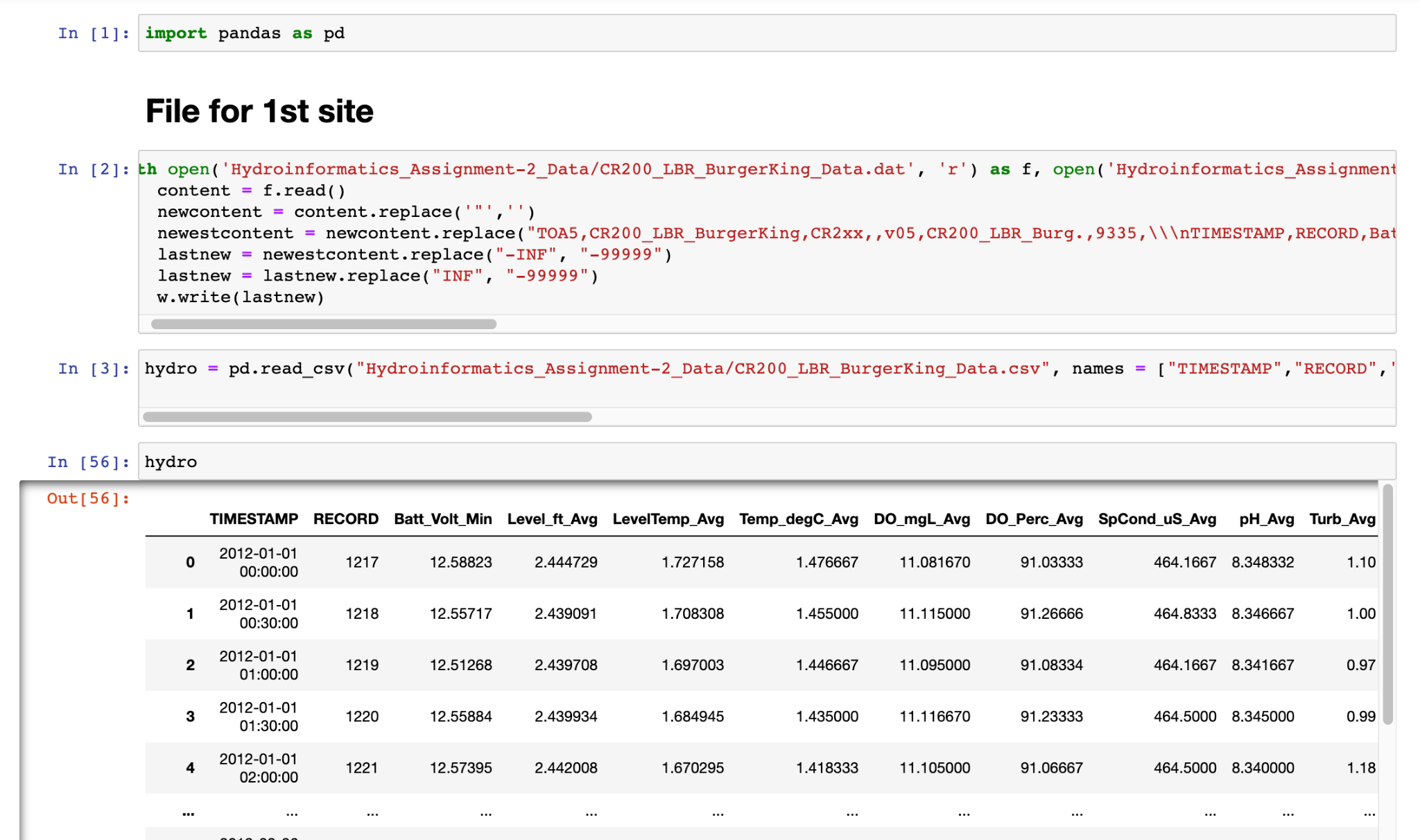


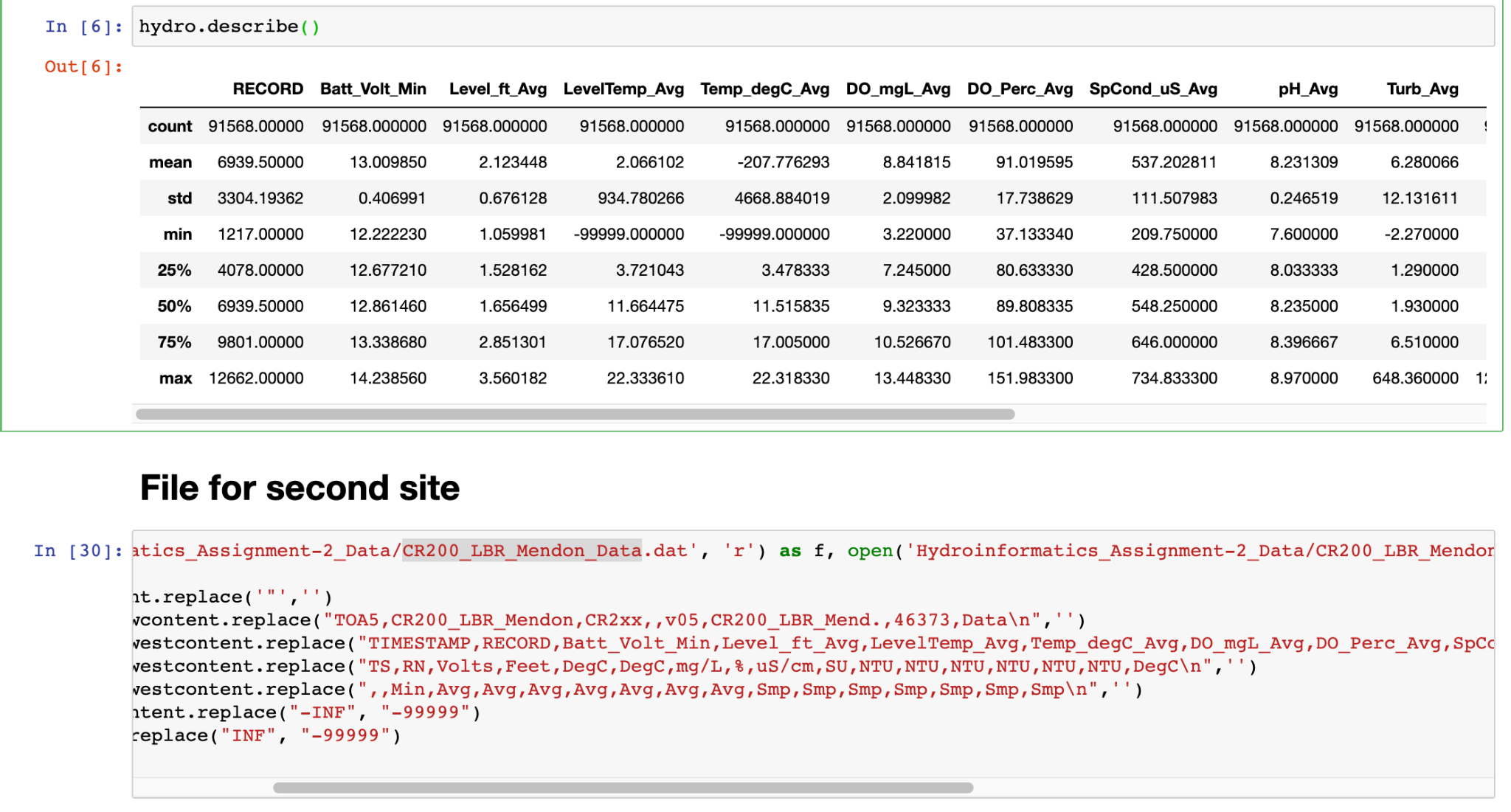
### FILES\_INFO

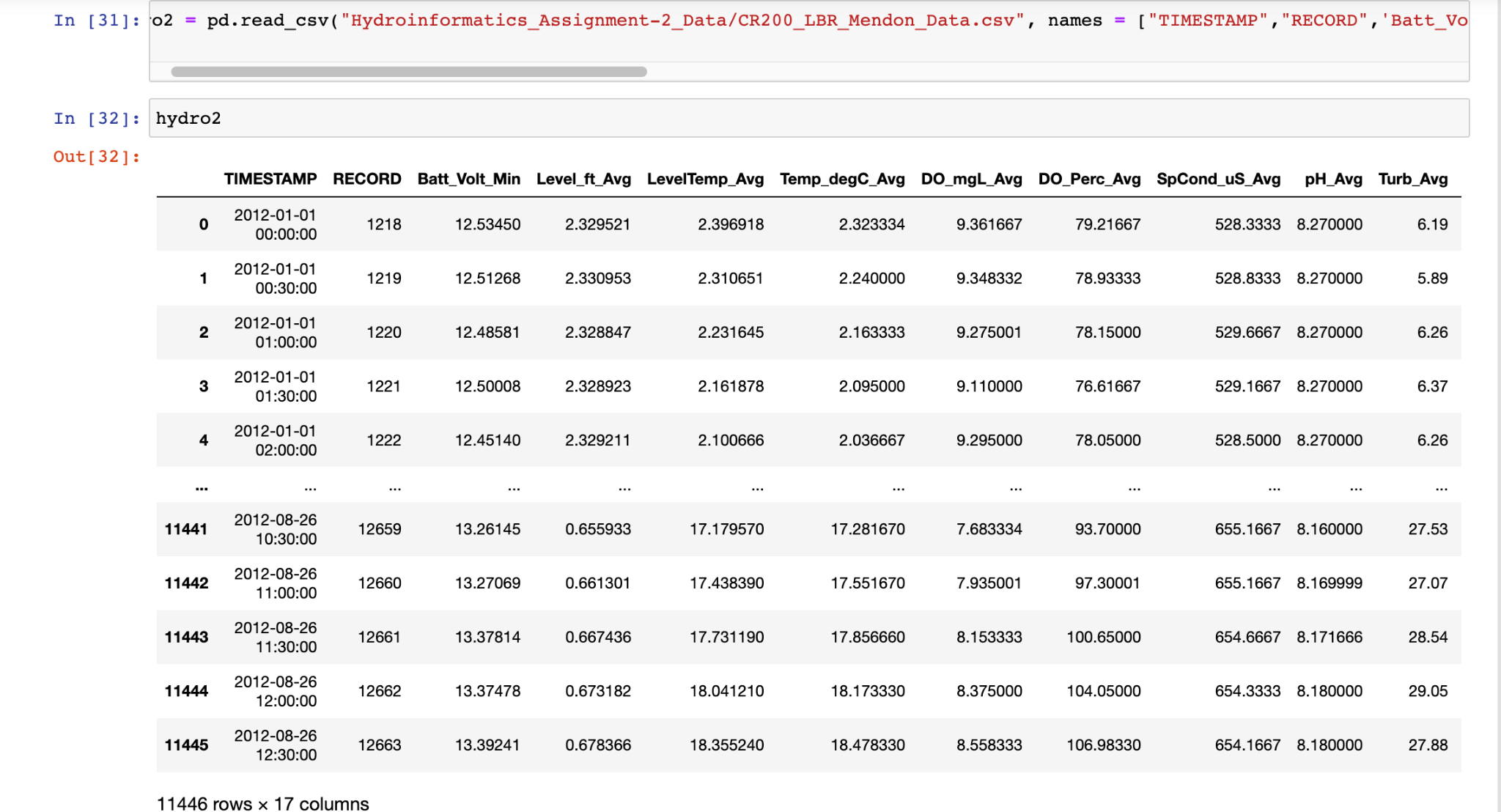


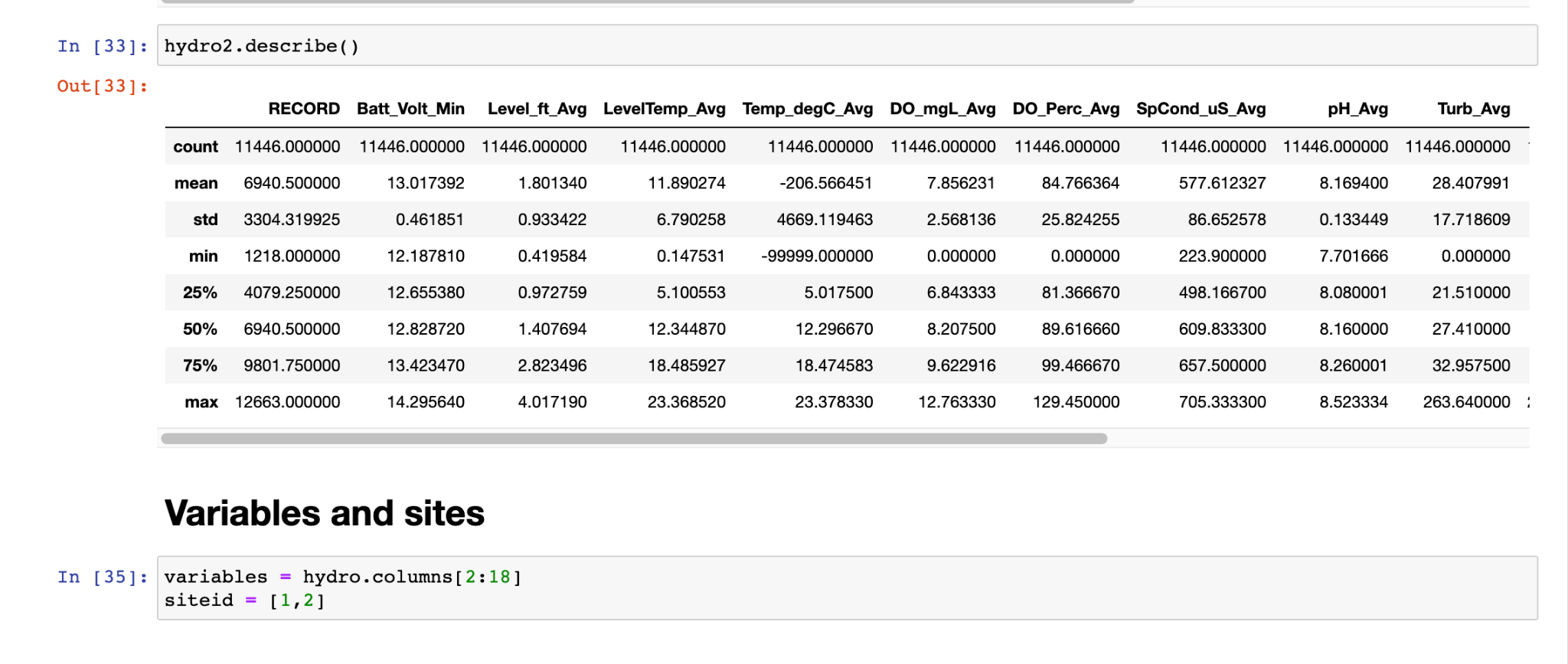
### READINGS\_INFO

**Code to Read the files and fill the Readings Table:**



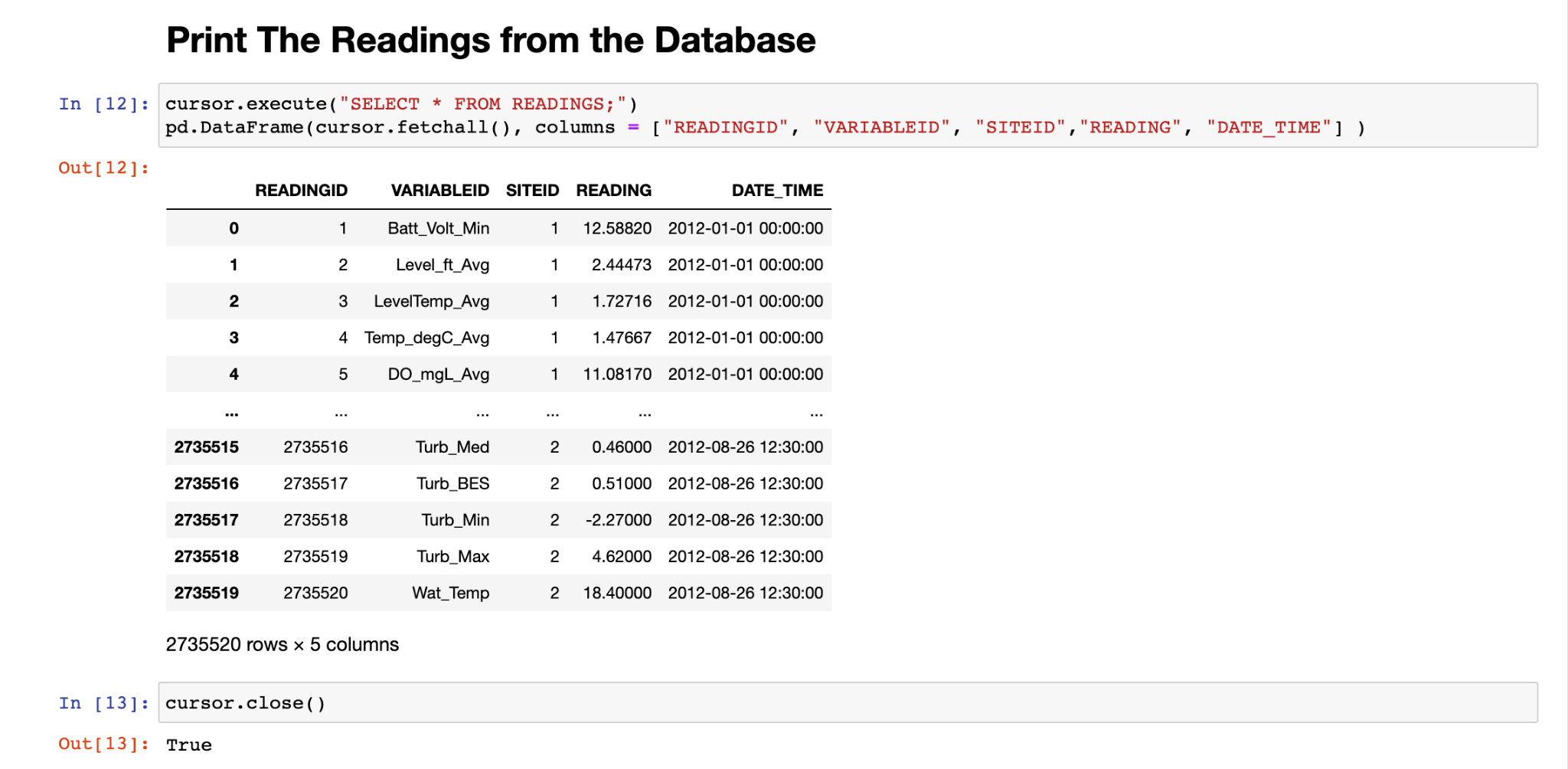












# 

# 

# 