GitHub

Title python, sqlite3 and 3 million rows

December 16 2019

This project has broad applications. I need help. But maybe this project is in the selfish interest of your project and your team. Consider these scenarios.

How many cars go through the intersection in the afternoon (say between 2 and 3pm) compared to the morning , (say between 10 and 11 am.)

How does the heart rate of the patient vary from evening, (say between 6 and 7) and to sleeping (say between 2am and 3am.)

How much more electricity does the apartment building use during a 24 period in January compared to the same time slot in May?

What is the impact of the temperature on the efficiency of the wind turbine. This implies a temperature input and a column with that value.

What is the temperature average for the month of June over the last 15 years. (this implies 15 years worth of data.)

These are all about “what happens over time”. The first column is a UNIX datetime stamp.

Let’s do this together. That is what GitHub is all about

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I have set up an experiment and it has produced about three million rows of data in about 25, sqlite3 databases. The databases and tables are setup with the exact same code; they are identical. There is one table per database. GitHub has the database file [test\_database] The only table is [first\_table] It has about 300K lines and was created with an automatic Arduino C++ experiment. This is an sqlite3 platform and has no complications regarding permissions.

first\_table has 6 columns:

column\_one::: datetime stamp; example 2019-03-02 18:54:44.991000

column\_two::: data from pendulum one example 1876598. This is the period of the pendulum in millionths of a second. So, the period is about 1.8 seconds.

column\_three::: data from pendulum two

column\_four::: data from pendulum three.

column\_five:: the data is sorted into the column by size. Column\_five is the “does not fit any criteria” column

column\_six::: extra column. For comments and the Python ‘start’ signal.

The pendulums were designed to have about the same period so they could be nicely graphed against one x axis. Periods are approximately 1817300, 183500 and 182100. The standard deviation is about 100.

Is there anything interesting is all these numbers? This seems like a sucker bet. But that is the theme.

Measurement out to a millionth of a second is powerful. And there are three pendulums.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I am going to set myself up as project manager. I am going to create the system analysis and set up specific Python objectives. This partly to attract developers, but also because this needs to be done anyway! I am not an autocrat; I am trying to attract help. I need help.

Other database issues: For about 6 months the experiment was under a Windows 10 OS and the numbers sorted nicely into the three columns. About February 2019, I moved the project to a Linux Mint OS. The Linux sent a Unicode to the Python and the value refused to sort. I fussed with this at some length but then decided not to worry. But the later data is all in column\_five and not sorted (and in a Unicode format). It, of course, includes the datetime column.

I am leaving out the unnecessary code. If you are interested in the C++ (Arduino), let me know. If you are interested in the insert under Python, let me know. The project is summarized in this video.

https://youtu.be/PH7FHUDZ4PM

Arduino C++ code and Python INSERT can be found here: https://app.box.com/s/l9q2wysaaxxpzy8jn8owiuxbr7svyyyg

I feel that this application has a broad appeal. There is lots of data in Sqlite3 databases. It has been automatically generated over time. How do we look at all these numbers?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I am the project manager. Let’s do it: Structure of the software project.

**Let’s start with some simplifications;**

The software will not be interactive. That is, there will be no input from the user. This means all modifications will made within the code. The code will be written with this in mind.

For example; does the pendulum’s period differ from day to night? Does the pendulums period differ from summer to winter? These different would be different programs, as opposed to a result of interaction with the operator.

There are 25 databases. Let’s leave these as is.

For example, when the software is complete, the databases will be fed in, manually and chronologically, one at a time. The output will go into one database, but there will a labor-intensive stage in the beginning. (this is, of course, negotiable)

The data will be checked as it is pulled from the databases. If there are problems then those rows are omitted with no comment, no note, no nothing. Just thrown away. There are too many complications to fuss with this. And the data is mostly pristine. And the row always includes the datetime stamp.

**Projects and their status:**

**Fetch from sqlite3 by time stamp;**

All data presentation has the datetime stamp on the x axis. The fetchall is always based on the time stamp in column\_one. This code is complete at least as a beta prototype. There is a major trick and I think it will work: example pseudocode

c.execute (“SELECT \* FROM first\_table where column\_one > some time interval”)

fetchmany(1000)

for example, to search all data one day at a time: The ‘time interval’ would be the first row for that day. And the ‘fetchmany(value) would be less than the number of records for a day . And the “c.execute” would go in a loop, cycling through the days, adding a row per loop with the row having the average and standard deviation. That is every “c.execute” would generate one row in the final database.

The databases would be fed into this one at a time, but the result would go into the same sqlite3 database. The result would be a table with about 500 rows (for each of 500 days) with the average and standard deviation.

I am working on this now and making progress.

**Sanitize data first:**

The whole needs to be automated and problems in the data will screw up everything.

The ‘fetch’ brings in a row of data and it will be used for mathematical manipulation.

In the test\_database the zero fields are “notNull” These all need to be changed to mathematical zero.

The values are strings. They need to be changed to integers.

Anything not numeric needs to be discarded. ( column\_five has some comments. )

The numbers need to be cleaned of garbage. That is, the data is quite precise. Set a broad limit and throw out outliers. Maybe comments could go into a different table. But a simplification is eliminating the junk and ignore it.

**Statistics**

The statistics are pretty simple: average and standard deviation. I assume that this means putting the number in a list and, using Pandas or numpy return the information. I am not sure if these are separate projects.

**Combine databases**

A simplification is to just use the 25 databases, but it would be cool to have them combined. The total data is not much more than 500MB.

This project includes

GitHub\_read\_me\_first. This document in Word format.

Test\_database This is file name of the database that is referenced in the Python program.

master.py This is master branch. I am fairly new, but that may be for the best. A veteran

Python programmer will find my simple code easy to follow.

(I think this is easy for someone better than I)

[all pendulum data] this folder has much of the data. The name is the date of start.

Each database is about 6 weeks. (a week is about 42K rows)

And this project implies the creation of a new database to hold the summary data. That has not been done yet. Any thoughts?!

That is all. Pull the master.py and look at it, please. There is lots of documentation in the program and spaces, with notation, for the various sections.

If you are interested, there are references above that will flesh out the larger project.