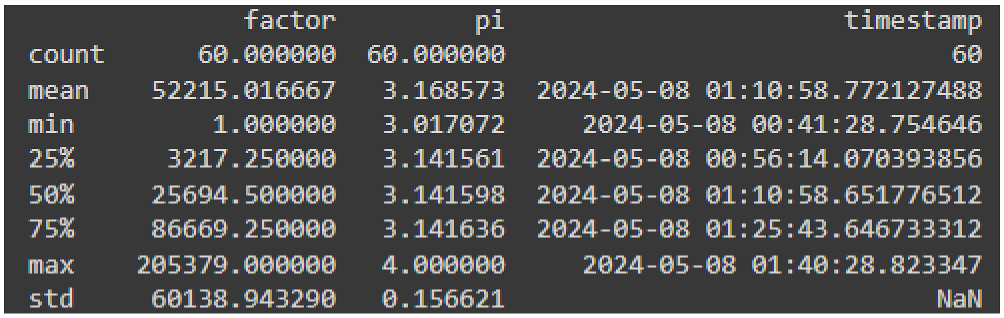
Documentation:
Given the challenge of timing an API call once a minute for an hour, I thought the best way to implement this was to separate the processes for timing and api calls separately to be the most efficient when testing. I first created a for loop to iterate 60 times and make the API call to test my ability to save the data into the local db that I created before then adding in the functionality to make a call once a minute. Afterwards, I added a print statement to verify the calls being made and also to verify that they were being made at the right time. Finally, in order to analyze the data in the db I made a loop to plot every column against the timestamp (excluding timestamp itself) before I realized that timestamp vs time would be an increasing linear plot (obviously) so I excluded that and was able to arrive at my conclusions based on the plots.

Verification techniques:
I utilized two strategies to verify 60 API calls and correct timing. The first method being a simple df.describe() which allowed me to see a few statistics about the data frame:

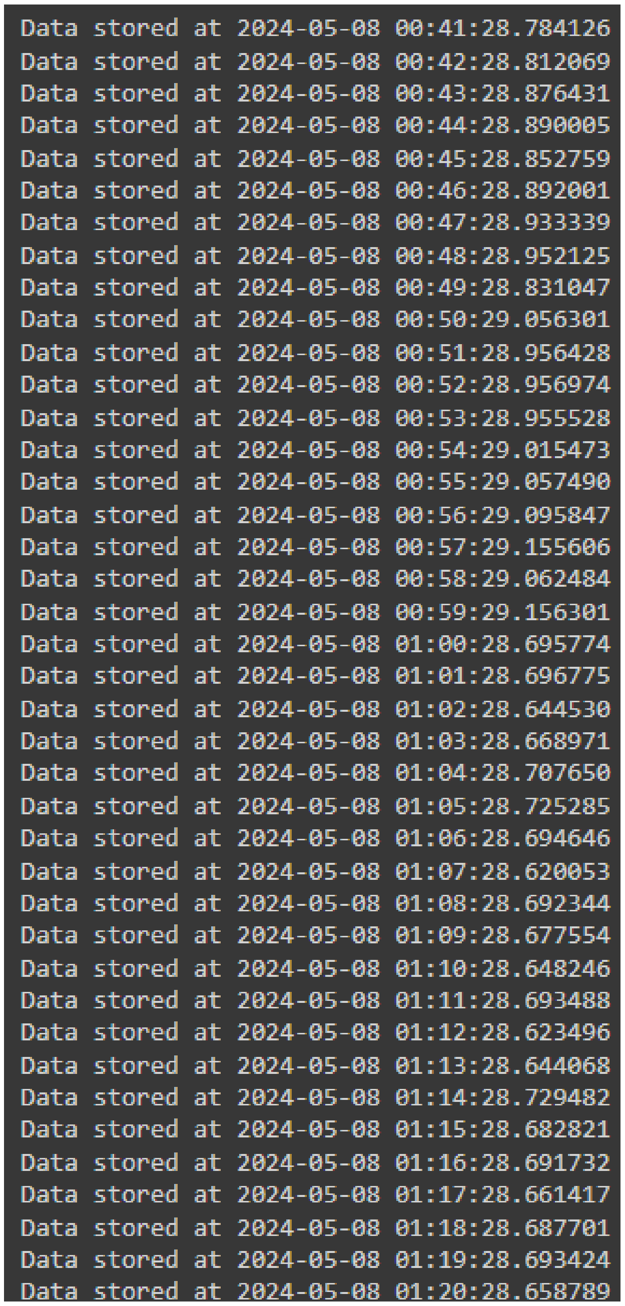


```
                factor         pi                      timestamp
count        60.000000  60.000000                             60
mean      52215.016667   3.168573  2024-05-08 01:10:58.772127488
min           1.000000   3.017072     2024-05-08 00:41:28.754646
25%        3217.250000   3.141561  2024-05-08 00:56:14.070393856
50%       25694.500000   3.141598  2024-05-08 01:10:58.651776512
75%       86669.250000   3.141636  2024-05-08 01:25:43.646733312
max      205379.000000   4.000000     2024-05-08 01:40:28.823347
std       60138.943290   0.156621                            NaN
```

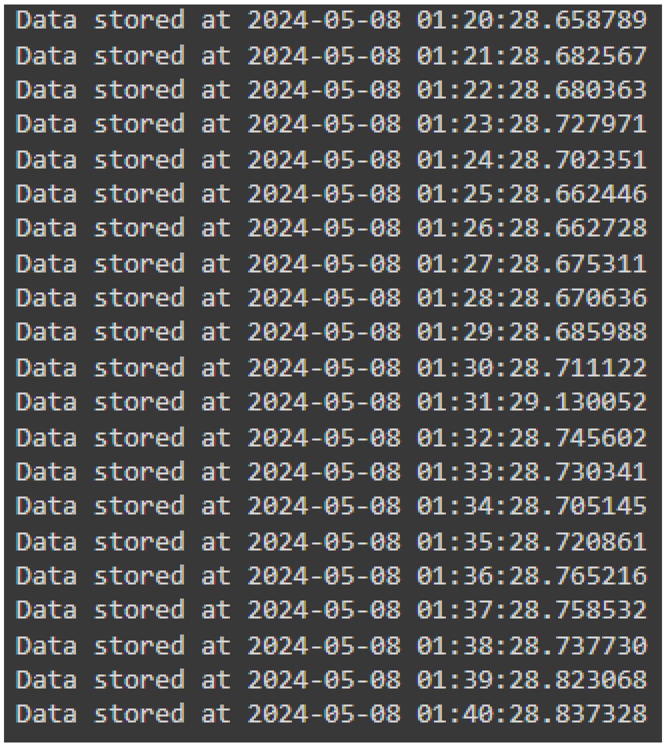
The first insight was that there were indeed 60 values in the db indicating that I had made the correct number of calls properly, the second was that the value of pi was fluctuating from 3.01 to 4, which is a bit interesting when considering it is the value of a ratio which does not change. I also was able to see that “factor” was a value between 1 and 205379 and made an inference that it was directly proportional to the value of pi or somewhat correlated to the time at which the data point was collected.

Next to verify that the timing of the calls were correct I took a look at the log generated by my API call function which contained: print(f"Data stored at {datetime.now()}")

(Screenshot on next page and in 2 parts)



```
Data stored at 2024-05-08 00:41:28.784126
Data stored at 2024-05-08 00:42:28.812069
Data stored at 2024-05-08 00:43:28.876431
Data stored at 2024-05-08 00:44:28.890005
Data stored at 2024-05-08 00:45:28.852759
Data stored at 2024-05-08 00:46:28.892001
Data stored at 2024-05-08 00:47:28.933339
Data stored at 2024-05-08 00:48:28.952125
Data stored at 2024-05-08 00:49:28.831047
Data stored at 2024-05-08 00:50:29.056301
Data stored at 2024-05-08 00:51:28.956428
Data stored at 2024-05-08 00:52:28.956974
Data stored at 2024-05-08 00:53:28.955528
Data stored at 2024-05-08 00:54:29.015473
Data stored at 2024-05-08 00:55:29.057490
Data stored at 2024-05-08 00:56:29.095847
Data stored at 2024-05-08 00:57:29.155606
Data stored at 2024-05-08 00:58:29.062484
Data stored at 2024-05-08 00:59:29.156301
Data stored at 2024-05-08 01:00:28.695774
Data stored at 2024-05-08 01:01:28.696775
Data stored at 2024-05-08 01:02:28.644530
Data stored at 2024-05-08 01:03:28.668971
Data stored at 2024-05-08 01:04:28.707650
Data stored at 2024-05-08 01:05:28.725285
Data stored at 2024-05-08 01:06:28.694646
Data stored at 2024-05-08 01:07:28.620053
Data stored at 2024-05-08 01:08:28.692344
Data stored at 2024-05-08 01:09:28.677554
Data stored at 2024-05-08 01:10:28.648246
Data stored at 2024-05-08 01:11:28.693488
Data stored at 2024-05-08 01:12:28.623496
Data stored at 2024-05-08 01:13:28.644068
Data stored at 2024-05-08 01:14:28.729482
Data stored at 2024-05-08 01:15:28.682821
Data stored at 2024-05-08 01:16:28.691732
Data stored at 2024-05-08 01:17:28.661417
Data stored at 2024-05-08 01:18:28.687701
Data stored at 2024-05-08 01:19:28.693424
Data stored at 2024-05-08 01:20:28.658789
```

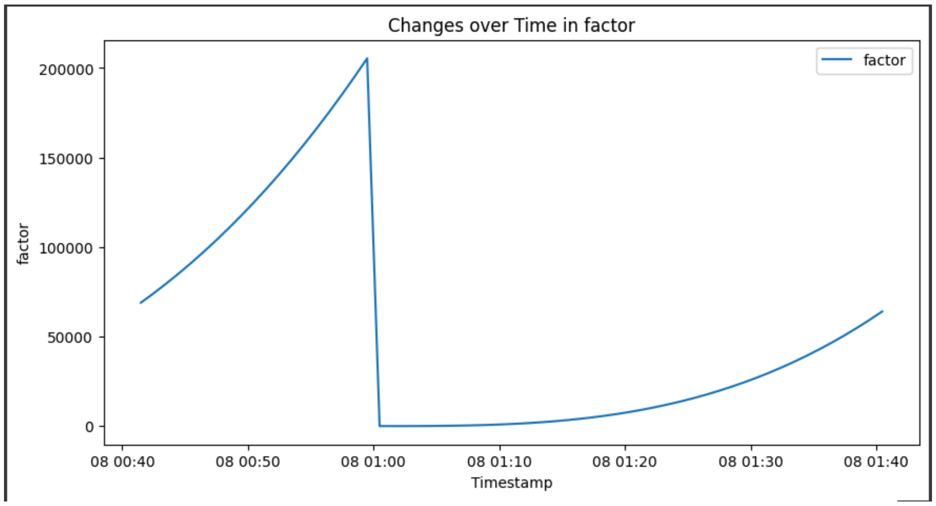


```
Data stored at 2024-05-08 01:20:28.658789
Data stored at 2024-05-08 01:21:28.682567
Data stored at 2024-05-08 01:22:28.680363
Data stored at 2024-05-08 01:23:28.727971
Data stored at 2024-05-08 01:24:28.702351
Data stored at 2024-05-08 01:25:28.662446
Data stored at 2024-05-08 01:26:28.662728
Data stored at 2024-05-08 01:27:28.675311
Data stored at 2024-05-08 01:28:28.670636
Data stored at 2024-05-08 01:29:28.685988
Data stored at 2024-05-08 01:30:28.711122
Data stored at 2024-05-08 01:31:29.130052
Data stored at 2024-05-08 01:32:28.745602
Data stored at 2024-05-08 01:33:28.730341
Data stored at 2024-05-08 01:34:28.705145
Data stored at 2024-05-08 01:35:28.720861
Data stored at 2024-05-08 01:36:28.765216
Data stored at 2024-05-08 01:37:28.758532
Data stored at 2024-05-08 01:38:28.737730
Data stored at 2024-05-08 01:39:28.823068
Data stored at 2024-05-08 01:40:28.837328
```

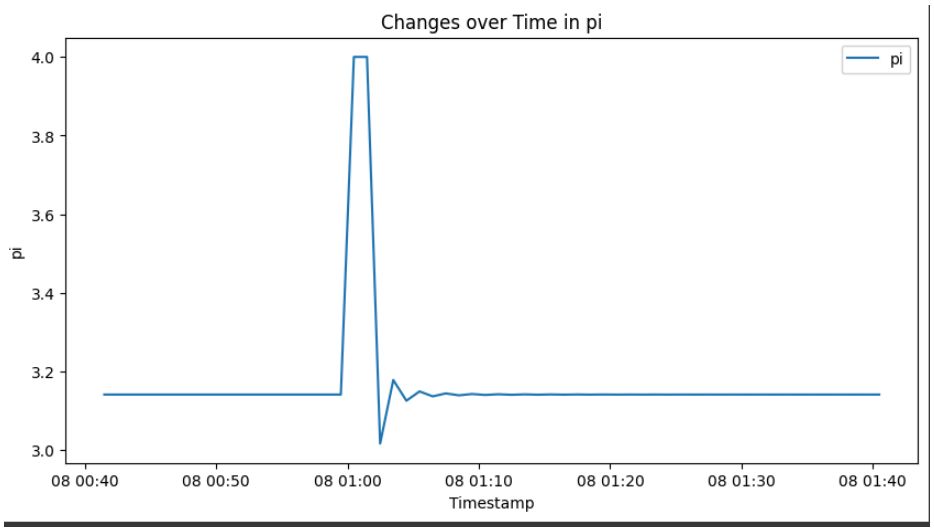
Looking at this generated log we can see that the calls went off about a minute apart from each other which validates that condition for the project.

Graph insights:

The graphs generated after the first part of the project look like the following:



This graph depicts the relationship between "factor" and time and indicates that it is an exponential function that starts at the beginning of each hour and slowly builds up to its maximum value. This changes my initial interpretation that it changes with pi when it instead seems to be influenced by time and plays a role in influencing pi rather than the other way around. This is fairly consistent with the high standard deviation for the factor column.



Finally, this heartbeat shaped graph shows the changes in "pi" over the hour and indicates that it remains at a mostly constant value throughout the hour until the factor reaches its steep dropoff at which point it experiences dramatic changes. This is fairly consistent with the data frame indicating a relatively low standard deviation for the pi column.