Kevin VanDelden
10/15/2021

Objective: Write a data pipeline, in Python, that collects daily prices, 24h volume, and mkt cap for all crypto assets listed by CoinGecko. This pipeline should collect the data once per day, perform a variety of checks and transformations on the data, and then load it into a MySQL database.

Database Considerations:

- Made sure to use an exact numeric type available for currencies, DECIMAL(65,30)
- Additional fields “created_at” and “updated_at” were added to help tracking of the data, so there would be record of any updates to records of days if necessary, and exact history of when data was collected.
- The 24H volume and mkt cap figures I found were given in BTC, so one transformation done was to take the value of BTC in usd at time of collection and multiply that to get the 24H volume and mkt cap figures in usd.
- I ran the database every 10 minutes while I was working on this report so I didn’t have to wait 24 hours for it to complete and do testing on the database. The code submitted is written to be run every 24H
- (database password is omitted in the connection profile)

Database Improvements:

- On a larger scale, I think I would create a “staging” and “production” database, where in the staging database I would schedule the report to collect data for the ~hour surrounding the target run time. That database wouldn’t need to store the extra data-indefinitely, it could just be used in the verification process and be a backup of the data collected, and only store x days previous of data.
- Find the largest decimal place that any particular site uses to save storage/trailing 0’s in the decimal place, I saw as many as 15+ decimal places in the data so far so just went with largest possible DECIMAL(65,30) but that was overkill for a handful of currencies.
- A backup check for BTC could be to fetch the usd value from another site or two and verify they are both within an acceptable range of deviation to verify its accuracy before applying to the other fields.

Duplication/Accuracy Considerations:

- The script identifies the number of records that are collected each run, if that number *changes* it could be a signal to collect again from the staging db or that the site has changed.
- Right now the script checks the entire database for any duplicate entries, if there is a problem with the scheduling feature,website, or any other bug it would identify identical records and could add in a DELETE function to remove those records.
- The script also checks the database for any records of the price, and would return any entry where the price of an asset fluctuates greater than 50% between records (10 minutes). That could be adjusted on an individual asset level, ex: If asset X fluctuates greater than 3x the standard deviation of it’s 24H% change, then something is likely off. Or you’re missing out on a rocket and you gotta get in before it goes to the moon.
- I wrote my original job in a Jupyter notebook which makes it super easy to segment out different features of the script, and I would start there when trying to maximize code recycling for other scrapers.

Duplication/Accuracy Improvements:

- Sites like coingecko and coinmarketcap also post the 24H% change and 7d % change. Using those as additional checks if they align with the database to make sure the script is working accurately would add a factor of safety.
- If I wanted super reliable accuracy I would probably set up a half-dozen scrapers on various sites, and have checks-and balances and if *none* of those worked you could ping a paid-for API as the final backup for any days missed, to keep the request usage down based on the usage plan.

Considerations of Different Sources:

- Could use a VPN with a variable IP address to prevent any site from blocking the scraper if that became an issue, or cycle between sites if constant collection from one is blocked. Any sites that require a login would automatically be set up to identify scraping behavior more easily.
- Could scrape a site like https://www.isitdownrightnow.com/ as another check, to check coingecko before even starting the scraper, and even use that as a determining factor between scraping different sites.