Microsoft Fabric: Crypto Currency Case Study

By Krishan Patel

1. Map out the product

Goal:

To develop a Live Cryptocurrency Tracker using Microsoft Fabric.

Essential Features:

- **Real-Time Price Updates:** Fetch the latest price of each cryptocurrency with a latency of under 1 minute.
- **Flexible Time Slicing:** Users should be able to select the time period they want to view.
- **Adaptive Candle Aggregation:** For larger time ranges, the tracker should automatically adjust the candle aggregation.

Key Design Principals:

- **Upstream Transformations:** Ensure any data transformations are performed as far upstream as possible.
- **Scalability:** Design the solution with the ability to efficiently handle increasing amounts of data without compromising performance.
- **Optimize Performance:** Ensure the report is fast and responsive.

2. Data source

To build a Cryptocurrency Tracker, we need a **REST API** to fetch real-time market data.

For this we will use the Binance Kline REST API: https://binance-docs.github.io/apidocs/spot/en/#kline-candlestick-data.

Kline/Candlestick Data

`GET /api/v3/klines`

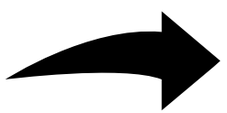
Kline/candlestick bars for a symbol.
Klines are uniquely identified by their open time.

Weight(IP): 2

Parameters:

Name	Type	Mandatory	Description
symbol	STRING	YES	
interval	ENUM	YES	
startTime	LONG	NO	
endTime	LONG	NO	
timeZone	STRING	NO	Default: 0 (UTC)
limit	INT	NO	Default 500; max 1000.

- If startTime and endTime are not sent, the most recent klines are returned.
- Supported values for `timeZone`:
  - Hours and minutes (e.g. `-1:00`, `05:45`)
  - Only hours (e.g. `0`, `8`, `4`)
  - Accepted range is strictly [-12:00 to +14:00] inclusive
- If `timeZone` provided, kline intervals are interpreted in that timezone instead of UTC.
- Note that `startTime` and `endTime` are always interpreted in UTC, regardless of `timeZone`.



```
[
  [
    1499040000000,      // Kline open time
    "0.01634790",       // Open price
    "0.80000000",       // High price
    "0.01575800",       // Low price
    "0.01577100",       // Close price
    "148976.11427815",  // Volume
    1499644799999,      // Kline Close time
    "2434.19055334",    // Quote asset volume
    308,                // Number of trades
    "1756.87402397",    // Taker buy base asset volume
    "28.46694368",      // Taker buy quote asset volume
    "0"                 // Unused field, ignore.
  ]
]
```

NOTE: Each kline has a designated EndTime, but the most recent candle doesn't automatically finalize until the period has ended. Therefore, the tracker must calculate and adjust this latest candle's values manually while it's still live.

Option 1. Use Incremental Refresh

Approach:

Load Historical Data (Fabric Notebook)

Use the Binance API to get 1-minute trading data for each cryptocurrency, starting from the earliest available data to the most recent. Retrieve data in batches of 1,000 rows per API call.

Partition by Symbol and Date

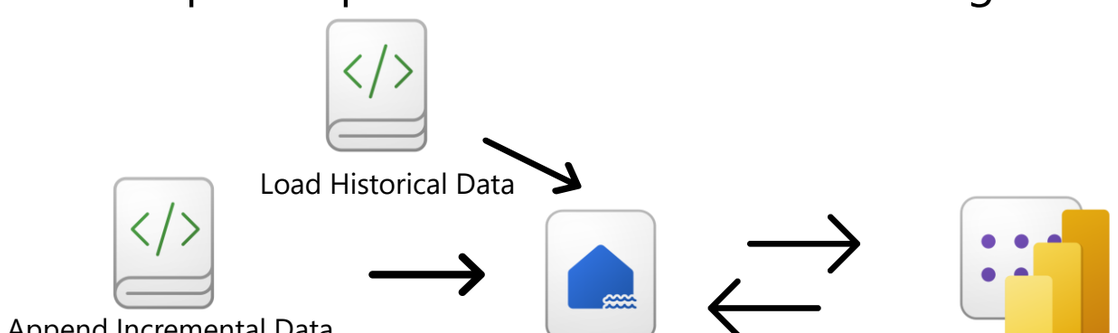
Partitioning by Symbol and Date improves query speed and efficiency by directing queries to relevant data slices, reducing load times and resource use.

Append Incremental Data (Fabric Notebook)

Continuously append new 1-minute interval data for each cryptocurrency, ensuring that the dataset stays up-to-date. Ensure to remove any kline data where the end_time has not concluded.

Manage Small File Problem

A consequence of running frequent incremental loads and data updates is the creation of many small files, which can degrade query performance and storage efficiency; using the OPTIMIZE command with V-order at regular intervals helps compact these small files and mitigate the issue.



Pros and Cons of Incremental Refresh

Pros:

Improved Notebook Performance

Faster data refresh by processing only new or modified data, reducing resource consumption.

Other use cases

The 1-minute trading data can be used outside of Power BI for data science applications, such as predictive modelling, anomaly detection, and advanced analytics.

Cons:

Complex DAX Aggregations:

Aggregations must be performed in DAX measures, which can increase report complexity and may impact performance, especially with large datasets.

Data Loading Challenge:

We need to load 60 million rows of data, as Publish to Web doesn't support Direct Lake, leading to potential performance issues and increased load times.

Option 2. Pre-Aggregated Time Periods

Time Period	Aggregation Level
1H	1m
4H	5m
1D	15m
1W	1H
1M	4H
YTD	YTD
1Y	1D
MAX	1W

Instead of using DAX for complex calculations, import data that's already aggregated (e.g., hourly, daily) for a specific period, saving time and improving report performance.

Approach:

Data Fetching and Parallelization

Define cryptocurrency symbols and timeframes (see table above), then fetch data for each symbol and period simultaneously using ThreadPoolExecutor to enhance efficiency.

Data Aggregation and Transformation

Fetch historical market data from Binance's API and aggregate it by time period, ensuring consistency in timestamps and processing.

Data Preparation for Analysis and Load

Combine the data into a single DataFrame, then convert it to a Spark DataFrame for scalable, efficient analysis, and load the data into a Delta table

Pros and Cons of Pre-Aggregated Time Periods

Pros:

Improved Report Performance

By importing pre-aggregated data, we eliminate the need for complex DAX calculations, resulting in faster report rendering and reduced load times.

Simplified Data Model

No need to create complex aggregations in DAX, making the model simpler and more maintainable.

Efficient Data Processing

Pre-aggregating data reduces the computational load during report generation, as the heavy lifting is done upfront.

Cons:

Loss of Granularity

Pre-aggregated data may not capture the finer details needed for some analyses, such as minute-level or real-time decision-making.

Initial Data Processing Time

Pre-aggregating data upfront may require more processing time during data import, especially for large datasets, to ensure accuracy across multiple periods.

On balance I choose to Option 2: Pre-Aggregated Time Periods