Cryptocurrency Market Dynamics: A Big Data Perspective

Rahul Kailasa, Raghav Madderla, Sai Krishna Kathika, Yashaswini Madineni, Varun Reddy Bhumi Reddy, Dinesh Kumar Tirupanapati,

San José State University

Big Data Technologies and Applications

Prof. Sangjin Lee

Dec 2, 2024

Table of contents

Abstract **2**
Problem Statement **2**
Introduction **3**
Data Overview **3**
Source 3
Columns in data 3
Methodology **4**
Data Extraction 4
Data Cleaning 4
Implementation 4
Analysis by queries and Results 6
Visualizations 13
Challenges **17**
Conclusion **18**

Abstract

This project focuses on how to tackle the challenges of analyzing vast cryptocurrency data to uncover essential market insights that can guide investment decisions. It focuses on key aspects like spread analysis, identifying the most traded symbols, pinpointing periods of peak trading activity, and studying daily, weekly, and monthly volatility trends. The analysis utilized a dataset of cryptocurrency trading records from Binance, originally 95 GB in size sourced from Hugging Face. For this study, 48 GB of the data was processed and analyzed. The project involved thorough data processing, analysis, and visualization to extract meaningful insights. These findings shed light on market behavior, offering a deeper understanding of cryptocurrency investment dynamics. Additionally, the outcomes of this research could be further developed to support predictive modeling for cryptocurrency markets, enhancing decision-making strategies for investors.

Problem Statement

Understanding cryptocurrency market volatility is essential for making well-informed investment decisions. Analyzing market trends and identifying optimal investment opportunities in various cryptocurrencies demand highly detailed data. However, this granularity generates enormous amounts of trading data, requiring robust big data technologies for efficient management and

processing. By utilizing these advanced tools, investors can uncover valuable insights into market behavior, paving the way for smarter decisions and more strategic approaches to cryptocurrency investments.

Introduction

Investing in cryptocurrency can be done by fully understanding the crypto market. To do this one needs to understand historical data spanning several years. It is not easy to understand such data without employing big data technologies, as understanding market volatility and its patterns relies on analyzing highly granular data, which is substantial in size. Therefore, this project leverages big data tools and visualization techniques to process the data and derive meaningful insights into market behavior.

Data Overview

Source

The dataset used in this analysis originates from Binance's Klines trading data, available publicly via the Hugging Face. This is a huge tabular data of size 95GB which consists of around 1.3B rows of trading data of various cryptocurrencies from 2020-2024. The data consists of two types of trades one is **spot** and the other is **um**. Only spot trading data was taken according to the project requirements which sizes 50GB approximately.

Columns in data

- ➔ **index**: A unique identifier for each row in the dataset.
- ➔ **open_price**: The price of the asset at the start of the time interval.
- ➔ **high_price**: The highest price of the asset within the time interval.
- ➔ **low_price**: The lowest price of the asset within the time interval.
- ➔ **close_price**: The price of the asset at the end of the time interval.
- ➔ **volume**: The total volume of the asset traded during the time interval.
- ➔ **close_timestamp**: The timestamp representing the end of the time interval (usually in Unix or ISO 8601 format).
- ➔ **quote_asset_volume**: The volume of the quote currency traded (e.g., in BTC/USDT, this would represent the USDT volume).
- ➔ **number_of_trade**: The total number of trades that occurred during the time interval.
- ➔ **tbbav**: (**Taker Buy Base Asset Volume)** – the volume of the base asset bought by takers.
- ➔ **tbqav**: (**Taker Buy Quote Asset Volume)** – the volume of the quote asset spent by takers.
- ➔ **ignore**: A placeholder column, possibly unused or reserved for future use.

➔ **symbol**: The symbol of the asset being traded (e.g., BTCUSDT, ETHUSDT).
➔ **type**: The type of market data indicates whether the data belongs to **spot** or **um**. (Only spot was the type of data taken for analysis as it is the real market data)
➔ **interval**: The time interval of the data (e.g., 1h, 30m, 15m, 5m).
➔ **missing**: A flag indicating whether data for the interval is missing or incomplete.
➔ **open_timestamp**: The timestamp representing the start of the time interval.

Methodology

Data Extraction

- The dataset was extracted from the source (Huggingface) using two libraries namely datasets and hugguingface_hub.
- After Installing these libraries using python in Jupyter notebook, calling a predefined function load_dataset() loads the data in a specified directory (default ~/.cache/huggingface/datasets.).
- This function downloads the data into the local system. As our data was in Arrow file format, the data got downloaded into multiple splits of arrow files.
- Further sharing of these files is also done and parquet files of the data are given as the end result.
- These parquet files were converted into CSV using pandas dataframe to process the data.

Data Cleaning

Data cleaning was carried out by removal of null values in the data and reassigning of the index column to make sure that the rows are numbered correctly. No further cleaning process was required as the dataset was very clean and clear.

Implementation

EC2 Setup and Data Conversion:

In this project, we are tackling the task of setting up an AWS EC2 instance for converting our huge cryptocurrency trading data from CSV to Parquet format with snappy compression. The dataset consists of approximately 48GB of trading data spread across eight files, representing different time granularities of market data.

For processing our huge dataset, we chose to create an instance type r5.2xlarge because of the volume of data that I was working on and the amount of processing it required made me choose

this instance type because it comes with 64GB RAM and 8 vCPUs, hence giving enough memory to efficiently handle our 48GB dataset. The storage was configured with a 20GB root volume for the operating system and applications and a larger 150GB data volume for our processing needs. This liberal allowance for storage meant we had adequate space for the source files, as well as the temporary files created during the conversion process.

Security was a key consideration while setting up the cluster. We configured a focused security group configuration, allowing SSH connection only from a specific IP address range. This was particularly useful as we needed to access the instance from different locations as we worked on the project. We set up the security group rules to be easily modifiable, allowing us to update the allowed IP addresses whenever my location changed, maintaining high security.

Once the instance is up and running, we set up a proper directory structure (/mnt/data) with one folder for our source CSVs and one for the Parquet-converted files. Installed all required packages: pandas, pyarrow, and boto3 using pip. The installation was pretty smooth, though there were one or two dependency issues to fight my way through. The data conversion work involved a huge amount of processing files that ranged from 794MB to 20GB. We worked with both spot and um-usd margin futures, each coming in different time intervals: 5 minutes, 15 minutes, 30 minutes, and 1 hour. In our approach we performed chunked processing to handle those huge files, and Snappy compression is used for Parquet files. This strategy is utilized to manage memory usage effectively and also maintain good processing speed.

Also, we created an appropriate IAM role, EC2-S3-Access-Role, with proper permissions to access S3 for seamless interaction with S3. With this, our conversion process was able to upload the converted files directly to our S3 bucket without storage of access credentials on the instance.

Databricks Cluster Configuration:

Reason for using databricks cluster is, when we tried to load the csv file to the pandas dataframe our kernel crashed then we decided to use databricks for data analysis. We utilized a Databricks cluster which is optimized for analytical workloads enough for our data. The cluster was configured with Databricks Runtime 15.4 LTS, this includes optimized versions of Apache Spark and other data processing tools. We have implemented autoscaling with a minimum of 2 and maximum of 8 workers to easily handle varying computational demands which we will encounter during our analysis, particularly when processing complex window functions and aggregations across our huge cryptocurrency dataset with around 220 million rows. This setup provided us with the perfect balance between performance and cost-effectiveness for our project analytical requirements.

Market Overview and Data Scope

Our analysis begins with understanding the scope of our data. Our cryptocurrency trading dataset contains the comprehensive market data ranging from January 2020 to July 2024, structured across 17 detailed columns. Temporal information is represented via 'open_timestamp' and 'close_timestamp', thus setting an exact time for every trading interval. Price changes are recorded through four essential price points: 'open_price', 'high_price', 'low_price', and 'close_price', providing full information about the price movement during every interval. In total, trading activity is provided within a multidimensional basis: volume is for base asset volume, quote_asset_volume, and the count of individual trades. Then it includes advanced market metrics, like 'tbbav', or Taker Buy Base Asset Volume, and 'tbqav' standing for Taker Buy Quote Asset Volume to indicate market direction and liquidity. The identification of the market is done via the 'symbol' column, which specifies the trading pair, and the 'interval' column, which defines the granularity of the timeframe that ranges from 5 minutes to 1 hour. Such a rich data structure allows for deep analysis of market behavior, liquidity patterns, and trading dynamics across different cryptocurrencies and timeframes.

Analysis by queries and Results

Intraday Volatility Patterns:

```
SELECT symbol, date_format(open_timestamp, 'yyyy-MM-dd') as trade_date, date_format(open_timestamp, 'HH') as hour_of_day, ((max(high_price) - min(low_price)) / avg(close_price)) * 100 as hourly_volatility, avg(number_of_trade) as avg_trades, avg(volume) as avg_volume FROM spot_data GROUP BY symbol, date_format(open_timestamp, 'yyyy-MM-dd'), date_format(open_timestamp, 'HH') ORDER BY trade_date, hour_of_day
```

This query analyzes the hourly pricing dynamics of cryptocurrency trading. It incorporates date and hour-level analyses in determining how volatility changes during the day. The calculation of hourly volatility as a percentage of the average price normalizes the price movements and thus makes it comparable for any pair, irrespective of their absolute price levels.

The analysis reveals intriguing patterns in hourly trading behavior. VITEUSDT exhibits notable volatility during the first hour of trading, with a 2.60% average price range and relatively low trading frequency of 8.42 trades per hour. This suggests potential opportunities for early-hour trading strategies, though the lower liquidity needs to be considered in trade execution.

In contrast, major pairs like ETHUSDT show more stable patterns with 0.39% hourly volatility but much higher trading frequency at 527.15 trades per hour. This highlights the trade-off between volatility and liquidity that traders must consider. XRPUSDT follows a similar pattern with 0.40% volatility and 269.05 average trades, positioning it as a medium-liquidity option with moderate price movement.

Daily Volatility Metrics:

```
WITH daily_prices AS ( SELECT symbol, date_format(open_timestamp, 'yyyy-MM-dd') as trade_date, max(high_price) as daily_high, min(low_price) as daily_low, first(open_price) as daily_open, last(close_price) as daily_close, avg(close_price) as avg_price FROM spot_data GROUP BY symbol, date_format(open_timestamp, 'yyyy-MM-dd') )
```

This query will reveal the daily price analysis framework by using CTE. First, it will start the aggregation of data at each day-to-day level to capture the required price metric every day of trading, such as high, low, opening, and closing price and the average price. This would give good insight into the daily movement of the pair prices. A GROUP BY clause ensures we have a separate analysis for each trading pair on each day.

Looking at the early 2020 period, we see fascinating patterns in price movements across different trading pairs. BATUSDT demonstrated significant daily volatility with a price range of 9.76% and a substantial open-to-close movement of 8.52%. This indicates active intraday trading and potentially profitable opportunities for day traders. In contrast, HBARUSDT showed a wider price range of 8.14% but more modest open-to-close volatility of 1.27%, suggesting more intraday price swings but ultimately more stable daily closures.

The data reveals an interesting pattern where certain trading pairs consistently show higher volatility during specific market phases. For example, NEOUSDT maintained a moderate volatility profile with a 4.18% range and 2.03% open-to-close movement, making it potentially more suitable for systematic trading strategies. These patterns suggest that different trading pairs might require distinct trading approaches and risk management strategies.

Weekly Volatility Patterns:

```
weekly_vol = spark.sql(""" WITH weekly_prices AS ( SELECT symbol, date_trunc('week', open_timestamp) as week_start, weekofyear(open_timestamp) as week_number, year(open_timestamp) as year, max(high_price) as weekly_high, min(low_price) as weekly_low, first(open_price) as weekly_open, last(close_price) as weekly_close, avg(close_price) as avg_price FROM spot_data GROUP BY symbol, date_trunc('week', open_timestamp), weekofyear(open_timestamp), year(open_timestamp) ) SELECT symbol, week_start, year, week_number, ((weekly_high - weekly_low) / avg_price) * 100 as weekly_range_volatility, ((weekly_close - weekly_open) / weekly_open) * 100 as weekly_return_volatility FROM weekly_prices ORDER BY year, week_number """) print("Weekly Volatility Analysis:")
```

This query gives us insight about weekly summary of how each cryptocurrency performs. It's like taking a weekly snapshot of each coin's journey - noting its highest point, lowest point, where it started (open), and where it ended (close). This helps us understand how prices move over longer periods, rather than just day by day.

Looking at the first week of 2020, we saw some interesting price movements. TOMOUSDT, for example, had quite a rollercoaster week - prices moved up and down by about 21% during the week, but interestingly, only ended up 0.61% higher than where it started. It's like a car that drove all over the city but parked just a block away from where it started.

WAVESUSDT tells a different story - it had even bigger price swings (23% range) but actually lost value, dropping by 12%. This is similar to a stock that had lots of trading activity but ultimately closed lower, showing that high activity doesn't always mean positive returns.

Monthly Volatility Patterns:

```
WITH monthly_prices AS ( SELECT symbol, date_format(open_timestamp, 'yyyy-MM') as year_month, max(high_price) as monthly_high, min(low_price) as monthly_low, first(open_price) as monthly_open, last(close_price) as monthly_close, avg(close_price) as avg_price FROM spot_data GROUP BY symbol, date_format(open_timestamp, 'yyyy-MM') ) SELECT symbol, year_month, ((monthly_high - monthly_low) / avg_price) * 100 as monthly_range_volatility, ((monthly_close - monthly_open) / monthly_open) * 100 as monthly_return_volatility FROM monthly_prices ORDER BY year_month
```

Here we're checking how each cryptocurrency performed over entire months. We're measuring both how much the price bounced around within each month (range volatility) and how much it actually gained or lost by the end of the month (return volatility).

Looking at January 2020, we see some fascinating monthly patterns. THETAUSDT showed a monthly range volatility of 37.2% and ended the month with a strong positive return of 21.9%. This is like a student who worked hard all month and finished with good grades.

LTCUSDT had an even more dramatic month with a 59.6% range volatility and finished with an impressive 64.7% return. This shows that sometimes, high volatility can lead to good returns, though it's not always the case.

On the flip side, TCTUSDT had a massive price range of 99.9% but ended up with a negative 34.7% return. This is like someone who put in a lot of effort but still ended up with disappointing results.

Year-wise performance analysis:

```
# Year-wise performance for all cryptocurrencies yearly_analysis = spark.sql(""" WITH yearly_stats AS ( SELECT symbol, YEAR(open_timestamp) as trade_year, MIN(open_price) as year_start_price, MAX(close_price) as year_end_price, AVG(volume) as avg_daily_volume, AVG((high_price - low_price)/open_price) * 100 as avg_volatility FROM spot_data GROUP BY symbol, YEAR(open_timestamp) ) SELECT trade_year, symbol, ROUND(((year_end_price - year_start_price)/year_start_price * 100), 2) as yearly_return, ROUND(avg_volatility, 2) as avg_volatility, ROUND(avg_daily_volume, 2) as avg_volume FROM yearly_stats ORDER BY trade_year DESC, yearly_return DESC """) print("Year-wise Performance Analysis:") yearly_analysis.show(10)
```

The yearly analysis query looks at the long-term trading patterns by aggregating price data on an annual basis, calculating key metrics like yearly highs, lows, opening and closing prices, and average prices for each trading pair. The query calculates two key measures of volatility: range volatility, or the maximum possible movement in price, and return volatility, or actual year-over-year performance. This gives a big-picture look at how different cryptocurrencies perform over longer periods of time.

This is important from the perspective of trading because analysis is required for long-term cycles of the market, the consideration of investments, and assessment of the pattern of risk over years, not days or months. For example, comparing the yearly range volatility against actual returns allows traders to understand better how much turbulence they might have to endure in the quest for possible gains and help in pinpointing which tokens have a regular yearly pattern instead of those that are erratic long-term. This information is all the more valuable for portfolio management and a long-term investment strategy, allowing one to differentiate between temporary market noise and real long-term trends.

Trading Volume Distribution:

```
SELECT symbol, date_format(open_timestamp, 'HH:00:00') as hour_of_day, avg(volume) as avg_volume, avg(quote_asset_volume) as avg_quote_volume, avg(number_of_trade) as avg_trades FROM spot_data GROUP BY symbol, date_format(open_timestamp, 'HH:00:00') ORDER BY avg_volume DESC
```

The query analyzes trading volume distribution by:

1. Breaking down trading activity by hour of day.
2. Calculating average volumes and trade counts.
3. Including both raw volume and quote asset volume for a complete perspective.
4. Ordering results to identify peak trading periods.

Peak Trading Activity:

1. PEPEUSDT Shows Highest Activity:
    - 16:00 UTC: 473.3B volume, 4,016 average trades
    - 14:00 UTC: 463.0B volume, 3,823 average trades
    - Clear pattern of afternoon trading dominance

Trading Activity Patterns:

```
SELECT symbol, date_format(open_timestamp, 'EEEE') as day_of_week, date_format(open_timestamp, 'HH') as hour_of_day, avg(number_of_trade) as avg_trades, avg(volume) as avg_volume FROM spot_data GROUP BY symbol, date_format(open_timestamp, 'EEEE'), date_format(open_timestamp, 'HH') ORDER BY avg_volume DESC
```

This query examines trading patterns by:

1. Combining day of week and hour analysis
2. Calculating average trade metrics for each time slot
3. Identifying both daily and hourly patterns
4. Ranking periods by trading activity

Finding the Best Trading Hours:

```
SELECT symbol, date_format(open_timestamp, 'HH') as hour_of_day, avg((high_price - low_price) / open_price * 100) as avg_price_range_percent, avg(number_of_trade) as avg_trades, avg(volume) as avg_volume, avg(quote_asset_volume) as avg_quote_volume FROM spot_data GROUP BY symbol, date_format(open_timestamp, 'HH') ORDER BY avg_price_range_percent DESC
```

Here we tried looking at every hour of the day to find out when trading is most active and when prices move the most, finding the best hour of the sales.

The data shows some fascinating patterns. The most dramatic action happens with UNIDOWNUSDT during midnight hours (00:00), where prices typically swing by a massive 1,956% - though this extreme movement suggests high risk. It's like a midnight flash sale where prices go crazy.

During more normal hours, we see IOUSDT being very active around lunch time (12:00), with lots of trades (about 6,034 per hour) and price changes around 52%. This is more like the busy lunch hour at a popular restaurant - lots of activity but more manageable.

Understanding Risk Levels of Different Cryptocurrencies:

```
SELECT symbol, count(*) as total_intervals, avg((high_price - low_price) / open_price * 100) as avg_price_range_percent, max((high_price - low_price) / open_price * 100) as max_price_range_percent, avg(volume) as avg_volume, avg(number_of_trade) as avg_trades_per_interval FROM spot_data GROUP BY symbol HAVING avg_volume > 0
```

We're looking at how wildly prices change (like checking how unpredictable a car's performance is), how often people trade it (like checking how popular the car model is), and what the worst-case scenarios have been (like checking the worst accident records).

Looking at the results, we found some cryptocurrencies that are like roller coasters and others that are more like a steady train ride. UNIDOWNUSDT, for example, is the cryptocurrency equivalent of an extreme sport - prices typically swing by 83% on average, and in its wildest moment, it moved by an incredible 3,952,775%! This is definitely not for the faint-hearted.

On the calmer side, we have pairs like BTCUSDT and ETHUSDT, which are more like taking a regular train ride - they move up and down, but in a more predictable way. They also have many more people trading them (higher average trades), which usually means it's easier to buy and sell when you want to.

Market Liquidity Study:

Here we're checking how easy it is to buy or sell different cryptocurrencies

```
SELECT symbol, date_format(open_timestamp, 'yyyy-MM-dd') as trade_date, avg(quote_asset_volume/number_of_trade) as avg_trade_size, sum(volume) as total_volume, sum(quote_asset_volume) as total_quote_volume, count(*) as intervals_count FROM spot_data WHERE number_of_trade > 0
```

The data shows that USDPUSDT is like a wholesale market - each trade is typically worth around $63,263, which is quite large. This suggests that bigger players (like professional traders or institutions) are active in this market.

Meanwhile, other cryptocurrencies might have smaller average trade sizes but more frequent trading - like a busy retail store with many smaller purchases. This information is crucial for traders because it tells them how much they can buy or sell without significantly affecting the price.

Market Making Potential Analysis:

This query is like analyzing a marketplace to find good business opportunities. Imagine you're

checking which products have the biggest difference between wholesale and retail prices (that's the spread), how many items are sold (volume), and whether more people are buying or selling. This helps identify where you could make money by being the middleman.

```
SELECT symbol, date_format(open_timestamp, 'yyyy-MM-dd') as trade_date, avg((high_price - low_price) / low_price * 100) as avg_spread_percent, avg(volume) as avg_volume, avg(number_of_trade) as avg_trades, sum(tbbav)/sum(volume)*100 as buyer_percentage FROM spot_data WHERE volume > 0
```

The results tell an interesting story about trading opportunities. BNBDOWNUSDT shows massive price differences - on one particular day (April 18, 2021), the price difference was over 71,000%! This is like finding a product that someone's willing to sell for $1 that others would buy for $710. However, these extreme cases usually come with high risks.

Looking at more normal trading pairs, we see most have spreads between 1-5%, which is like a regular store's markup. The buyer percentage hovering around 45-55% tells us there's usually a good balance between buyers and sellers, making it easier to match trades.

Most Popular Trading Pairs:

This query is like taking attendance at a very large party - we're counting how many people traded each cryptocurrency (total trades), how much was traded (volume), and what the average price was. It helps us understand which cryptocurrencies are the most popular and actively traded.

```
SELECT symbol, SUM(volume) as total_volume, SUM(quote_asset_volume) as total_quote_volume, SUM(number_of_trade) as total_trades, AVG(close_price) as avg_price FROM spot_data GROUP BY symbol ORDER BY total_quote_volume DESC
```

Bitcoin (BTC/USDT) stands out as the superstar of the crypto world - it's like the iPhone of cryptocurrencies. It had a whopping 13.89 billion trades and an average price of $32,591. Ethereum (ETH/USDT) follows as a strong second, with 5.49 billion trades and an average price of $1,888.

Interestingly, stablecoins like BUSD/USDT also show high activity with 1.87 billion trades, but their prices barely move - they're like the steady workhorses of the crypto world, mainly used for moving money around rather than for price speculation.

Measuring Cryptocurrency Price Swings:

Think of this query as a heart rate monitor for cryptocurrencies. Just like how a doctor measures

how much your heart rate varies during the day, we're measuring how much cryptocurrency prices bounce up and down. We're looking at both the typical daily swings (average) and the most extreme movements (maximum) to understand how 'excited' each cryptocurrency gets.

```
SELECT symbol, AVG((high_price - low_price)/low_price * 100) as avg_price_range_percent, MAX((high_price - low_price)/low_price * 100) as max_price_range_percent, AVG(number_of_trade) as avg_trades, AVG(volume) as avg_volume FROM spot_data GROUP BY symbol HAVING avg_volume > 0
```

Our analysis shows that different cryptocurrencies have very different personalities. UNIDOWNUSDT is like a highly caffeinated trader - extremely energetic with price swings averaging 91.74% and sometimes going absolutely wild with swings over 3,900%! This is definitely not for someone looking for a calm investment.

In the middle, we found coins like LINKUSDT, with more moderate but still significant movements around 15-16%. This is like someone who's active but not hyperactive - enough movement to create trading opportunities but not so much that it's unmanageable.

Buyer vs Seller Behavior:

Think of this query as a heart rate monitor for cryptocurrencies. Just like how a doctor measures This query is like being a referee in a trading game, counting how often buyers win versus sellers. We're looking at who's more aggressive in the market (buyer dominance), how busy the market is (volume and trades), and how much prices move around. It's similar to watching a tug-of-war and keeping score of which side is pulling harder.

```
SELECT symbol, SUM(tbbav)/SUM(volume) * 100 as buyer_dominance_percentage, AVG(volume) as avg_volume, AVG(number_of_trade) as avg_trades, AVG((high_price - low_price)/low_price * 100) as avg_price_range_percent FROM spot_data WHERE volume > 0 GROUP BY symbol
```

Most cryptocurrencies show a fairly balanced game between buyers and sellers, with the buyer percentage usually staying between 45-55%. This is healthy - like a good sports match where both teams are equally strong.

The really interesting part is seeing how this balance affects prices. When we see periods with higher buyer dominance (above 55%), it often coincides with price increases. It's like watching the crowd at a market - when there are more eager buyers than sellers, prices tend to rise.

Best and worst performance:

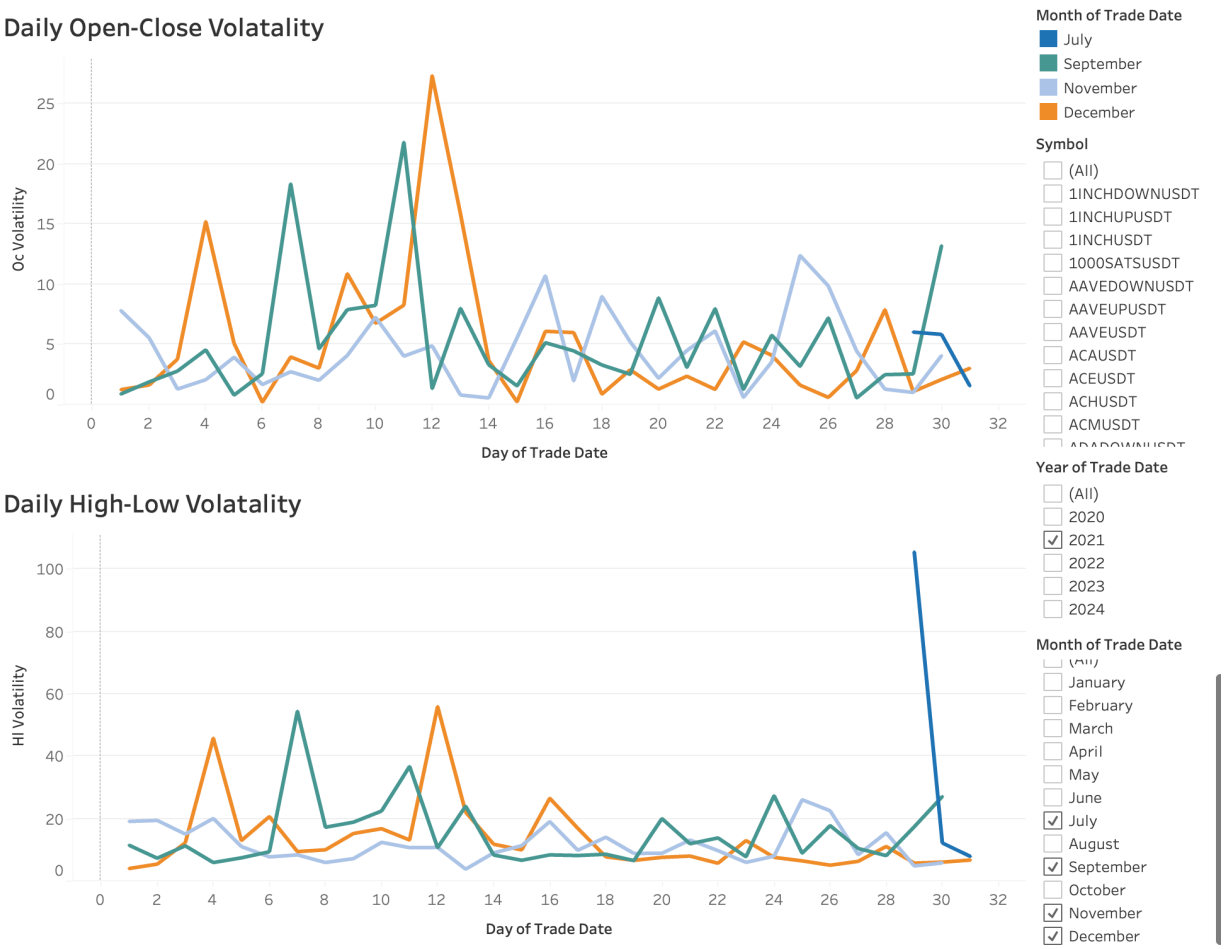
```
best_worst = spark.sql(""" WITH yearly_stats AS ( SELECT symbol, YEAR(open_timestamp) as trade_year, MIN(open_price) as year_start_price, MAX(close_price) as year_end_price, AVG(volume) as avg_daily_volume, AVG((high_price - low_price)/open_price) * 100 as avg_volatility FROM spot_data GROUP BY symbol, YEAR(open_timestamp) ), ranked_performance AS ( SELECT trade_year, symbol, ROUND(((year_end_price - year_start_price)/year_start_price * 100), 2) as yearly_return, ROUND(avg_volatility, 2) as volatility, ROUND(avg_daily_volume, 2) as avg_volume, RANK() OVER (PARTITION BY trade_year ORDER BY ((year_end_price - year_start_price)/year_start_price * 100) DESC) as best_rank, RANK() OVER (PARTITION BY trade_year ORDER BY ((year_end_price - year_start_price)/year_start_price * 100) ASC) as worst_rank FROM yearly_stats ) SELECT trade_year, symbol, yearly_return, volatility, avg_volume, CASE WHEN best_rank = 1 THEN 'Best Performer' WHEN worst_rank = 1 THEN 'Worst Performer' END as performance_label FROM ranked_performance WHERE best_rank = 1 OR worst_rank = 1 ORDER BY trade_year DESC, yearly_return DESC """)

print("\nBest and Worst Performers by Year:") best_worst.show()
```

This query tells us the best and work performed currency year-wise. With an astounding return of 2,319.36%, BANDUSDT became the top-performing trading symbol in 2020. As in the 2023 analysis, the same symbol appears in more than one entry in the same year, each with different metrics. Trading volumes for BANDUSDT varied from 0 to roughly 13,063 units, while the volatility ranged from 0% to 1.31%. This data unpredictability emphasizes the need for query modification in order to aggregate entries and give a more comprehensive picture of annual performance patterns.

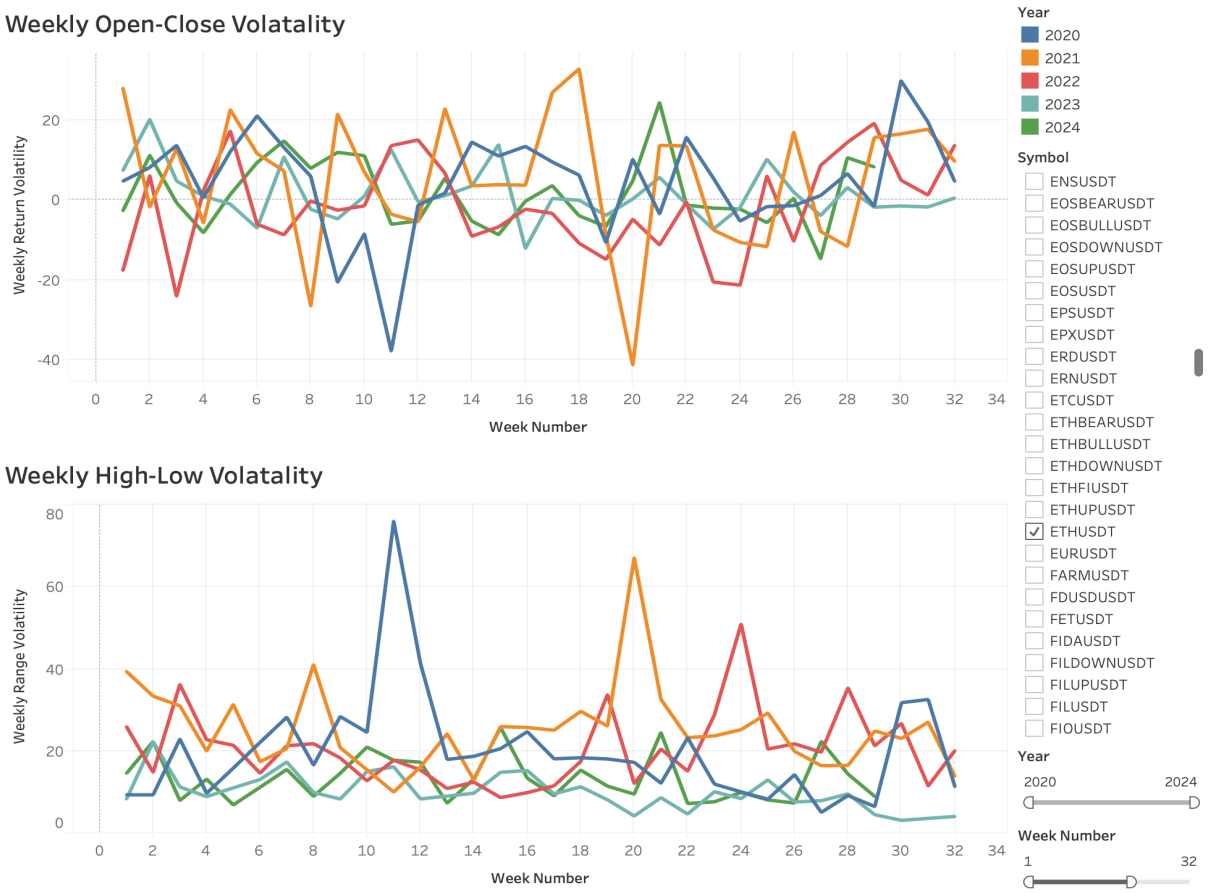
Visualizations

- When it comes to volatility of a coin, it is difficult to know just by observing the query output. Visualization can help in understanding the data patterns easily and get the underlying data story.
- Visualization was done using **Tableau.**
- Volatility for Daily, Weekly, Monthly, Yearly have been visualized as shown below.
- Each visualization has filters such as symbol, Year, Month, Week to easily understand data at a specific time.



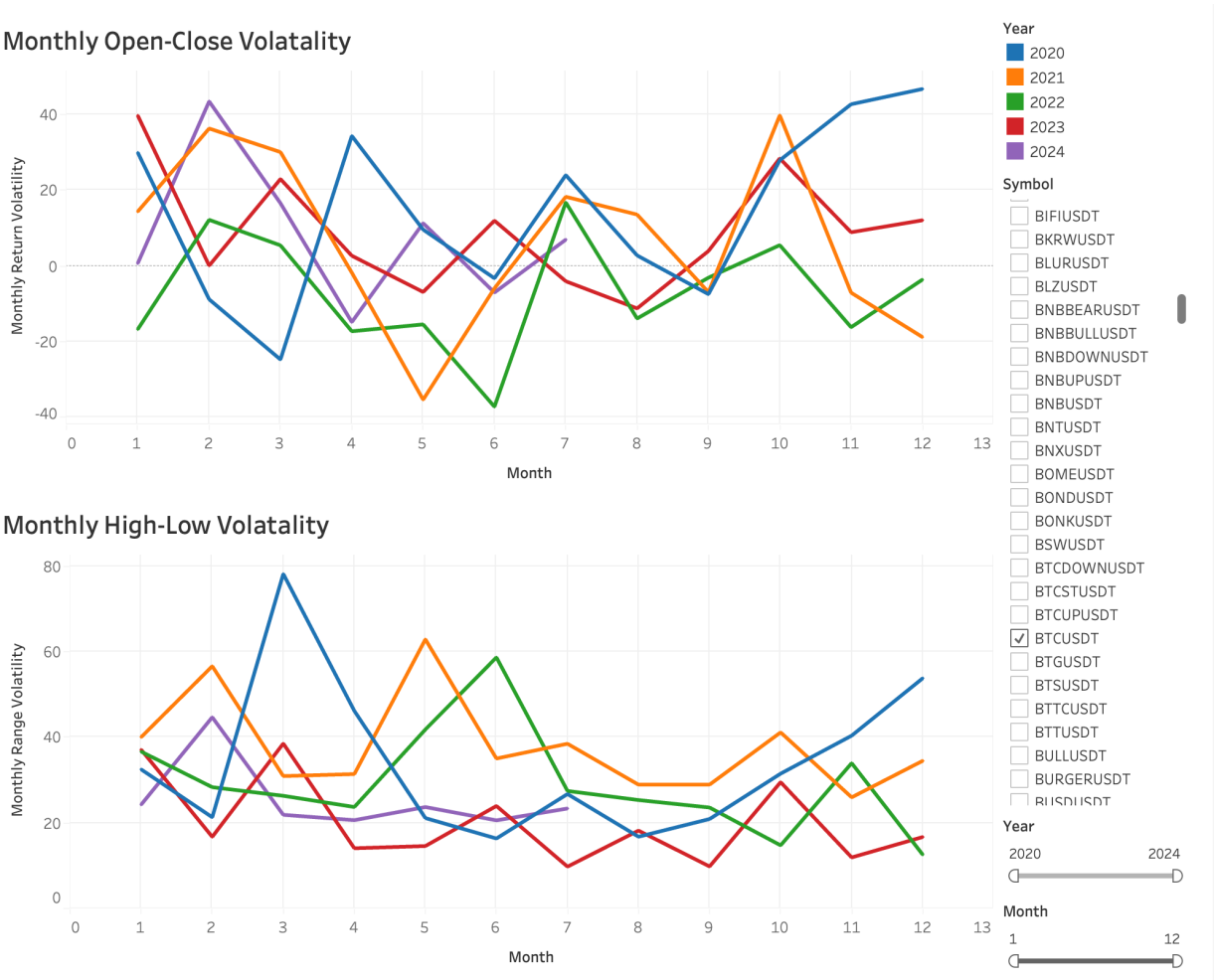
Open-Close Volatility: Daily volatility for selected months (July, September, November, December) in 2021. Peaks occur around specific days of the trade month, indicating high trading activity or price movement.

High-Low Volatility: Emphasizes daily range variations, showing significant peaks on certain days, possibly correlating with market news or events.



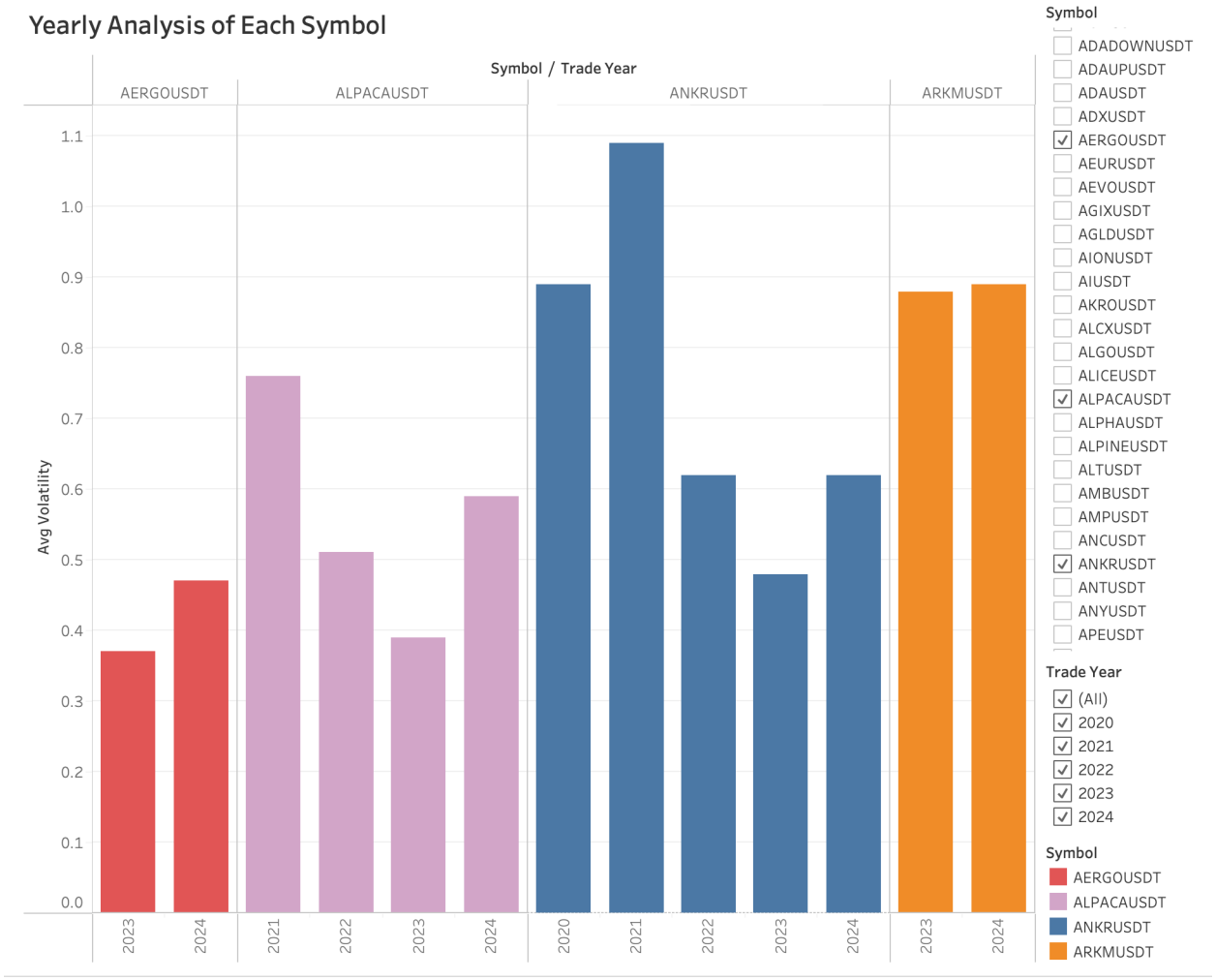
Open-Close Volatility: This plot depicts the fluctuation in weekly returns across different years, 2020-2024. Each cryptocurrency symbol has different trends in volatility, highlighting some notable spikes and drops during certain weeks.

High-Low Volatility: The weekly range volatility is shown with extreme outliers in some weeks, especially for certain symbols, which may indicate high activity in the market or some unexpected events.



Open-Close Volatility: Monthly return trends across years (2020-2024) show patterns in volatility. Specific months have marked increases, possibly reflecting seasonality or recurring market events.

High-Low Volatility: The monthly range volatility captures larger movements in certain months, reflecting high market uncertainty or opportunities.



The yearly average volatility for each cryptocurrency symbol draws into focus how the performance of various symbols changes in different years. For example, some of them show stability, whereas others have sharp upsurges or downfalls, reflecting either market dynamics or symbol-specific influences.

Insights from visualization:

Interesting and common insights from the visualizations of volatility are, whenever the high-low volatility is high, the open-close volatility is low in all the time frames, be it daily,weekly or monthly. These insights tell us what type of crypto to invest and when to invest.

Challenges

Handling Large File: (spot_5m.csv - 20GB) The processing was done in chunks to avoid memory issues while processing the file, reading it in chunks instead of loading it into memory. This solved not only the memory issue but also made the process more robust and reliable.
Uploading to EC2: Due to the breaks in the connection, it was painful to upload the big file to EC2. This is solved by using the rsync command, which made it possible to resume the data transfer from where it stopped when the connection broke.

Changing IP Addresses for Access to EC2: The frequent change of IP addresses made access to the EC2 instance very cumbersome. A simplified procedure for updating security group rules was developed, which allowed seamless access management while maintaining security, even when working from multiple locations.

Conclusion

This project successfully utilized big data technologies to unlock the dynamic cryptocurrency market with actionable insights into market behavior. Using AWS EC2/Databricks and complex query analysis, we were able to process a dataset volume of 48GB of data to discover patterns regarding volatility, trading volumes, and market liquidity for cryptocurrencies. The visualizations that unfold show when and where an investor should invest, due to the major implications related to traders and investors. With such challenges like large file processing, discontinuation of an internet connection, and dynamic IP configuration, the robust methodology supported the processing effectively for meaningful results. These will not only enhance the understanding of the market dynamics but also lay the foundation for predictive modeling to support smarter investment strategies in this evolving cryptocurrency landscape.