MODERN DATA ARCHITECTURE

Crypto & Conventional Assets : A Comparative Analysis

Group 6

OVERVIEW

1. PROBLEM
2. DATA & SOURCES
3. DATA INGESTION
4. DATA STORAGE
5. DATA PROCESSING

PROBLEM & CONTEXTUALIZATION

In light of the current level of **acceptance of cryptocurrencies**, we are undertaking an assessment of the behaviour of top cryptocurrencies in comparison to other asset classes and commodities. The objective is to ascertain whether there exists a correlation among these various financial instruments.

Analysis: Pairs Correlation:

To **derive meaningful insights** from this analysis, we will be using Pairs Correlation **to demonstrate the similarities or differences in the behaviour of these asset classes** and observe the market trends and sentiments by analysing how different these asset types move together or diverge during various market conditions.

DATA AND SOURCES

- **Source 1: Binance Website**

Our primary objective centers around acquiring valuable market data for cryptocurrencies from Binance, with a keen emphasis on facilitating in-depth analysis. To achieve this, we employ the use of Apache NiFi as our designated data ingestion tool for the purpose of web scraping. Through this tool, we aim to extract pertinent information directly from our predefined source, ensuring a streamlined and efficient process for gathering cryptocurrency market data from Binance. This strategic approach sets the stage for comprehensive analysis, enabling us to derive actionable insights from the data we collect

Cryptocurrencies we have collected:

Bitcoin (BTC) | Etherium (ETH) | Binance Coin (BNB) | Ripple (XRP) | Solana (SOL)

DATA AND SOURCES

Source 2: API for Different Asset Classes

We have collected data from different asset classes, we have selected the top performing stocks and commodities such as gold and silver. In collecting the relevant data for these assets, we have used an API called MarketStack to extract csv files of these stocks and commodities.

Stocks:

NVIDIA. amazon Alphabet Microsoft

Commodities:

GOLD SILVER

DATA INGESTION

SOURCES

BINANCE

We collected data on different dates for analysis, mainly using JSON files. Real-time market data was obtained through Binance SPOT's Websocket API, which delivers information in JSON format for human readability and machine parsing.

The ingestion process started from connecting to our data source, in our case we have a streaming data source from the Binance website.

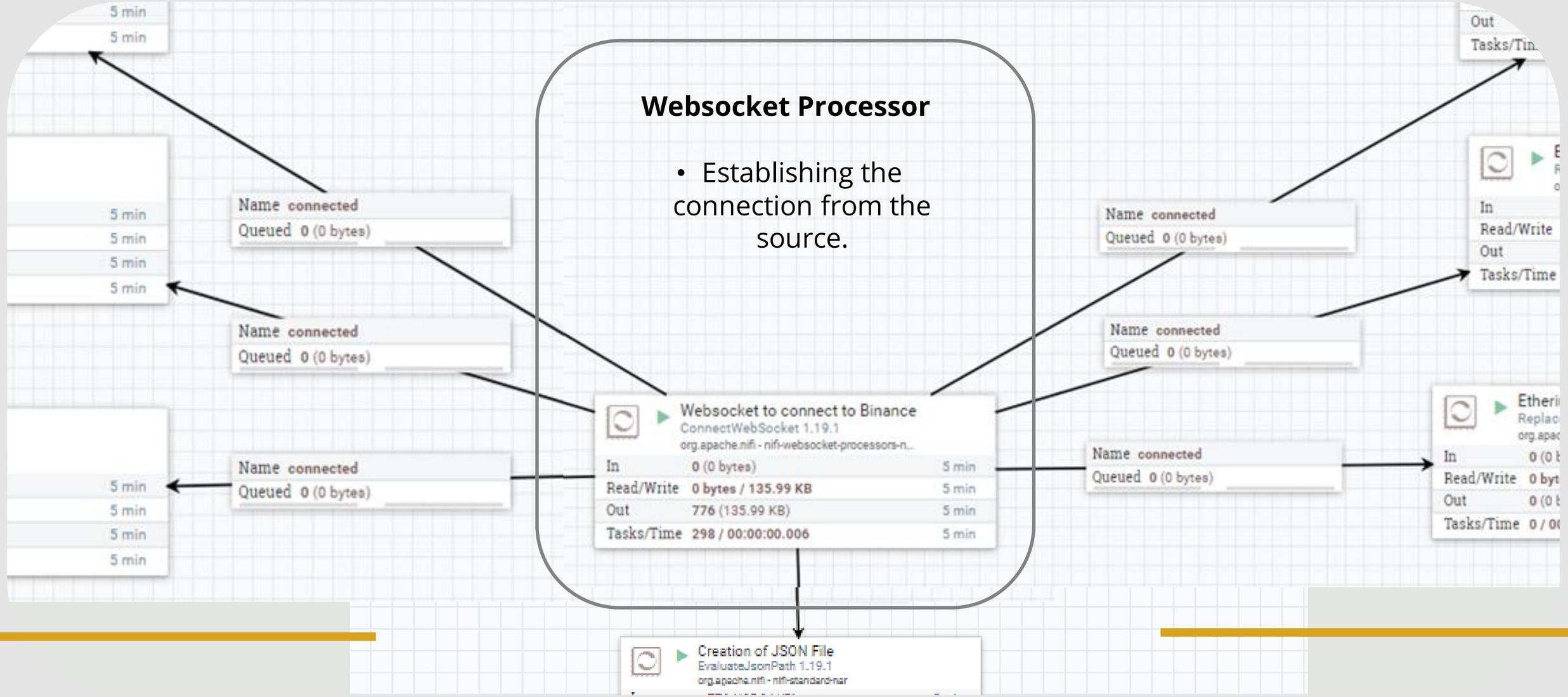
After the connection has been established, we are now ready to create the data flow by connecting processors in a specific order to achieve a series of actions in our Nifi workflow.

INGESTION

APACHE nifi

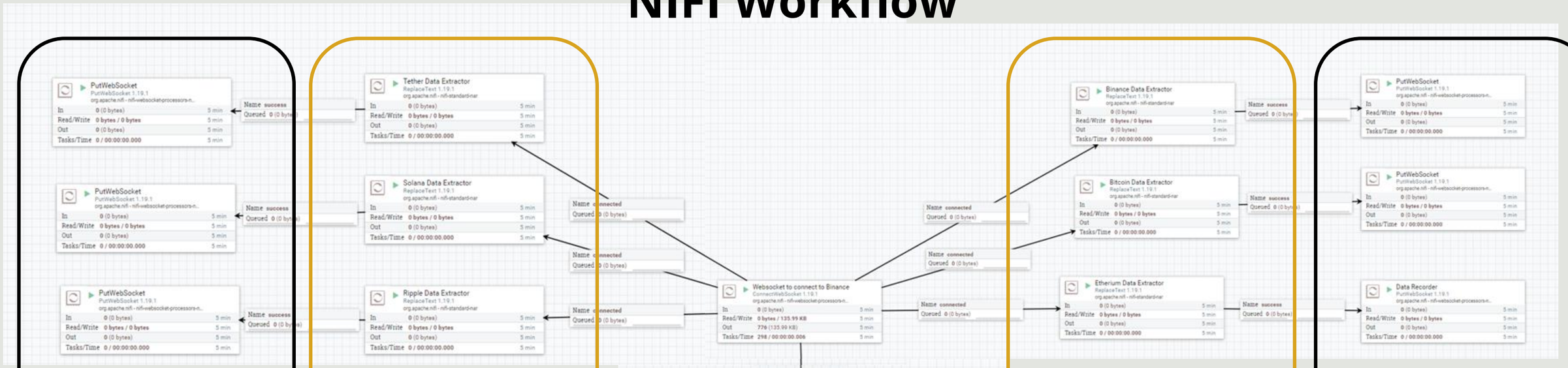
DATA INGESTION

NIFI Workflow



DATA INGESTION

NIFI Workflow



PutWebsocket

- Receives the data from the ReplaceText Processor .
- Sends it back to the Websocket to ingest in our storage system.

Data Extractors

- ReplaceText Processor available in our Nifi Canvas
- Obtain data from the source.
- Specify data you want to extract

Data Extractors

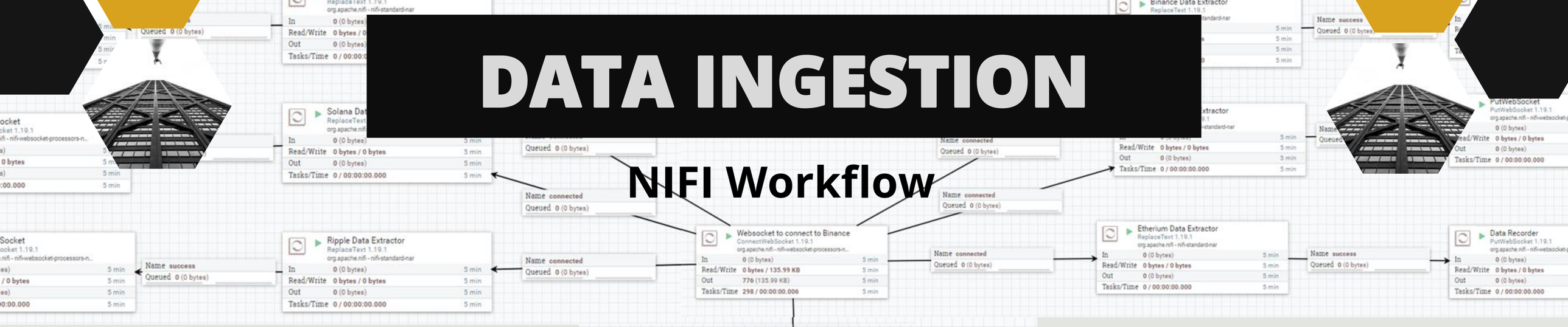
- ReplaceText Processor available in our Nifi Canvas
- Obtain data from the source.
- Specify data you want to extract

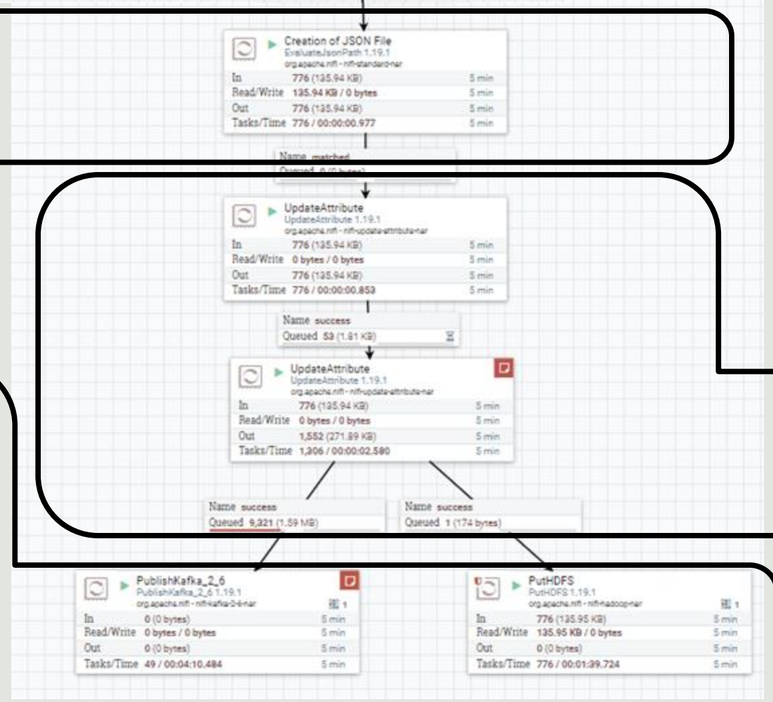
PutWebsocket

- Receives the data from the ReplaceText Processor .
- Sends it to back to Websocket to ingest in our storage system..

DATA INGESTION

NIFI Workflow





EvaluateJSON Path

Processor that creates the JSON files. Receives the data from the Websocket and arranges it into a JSON File

Storage

Once the file has been updated then it will be ready for storage.

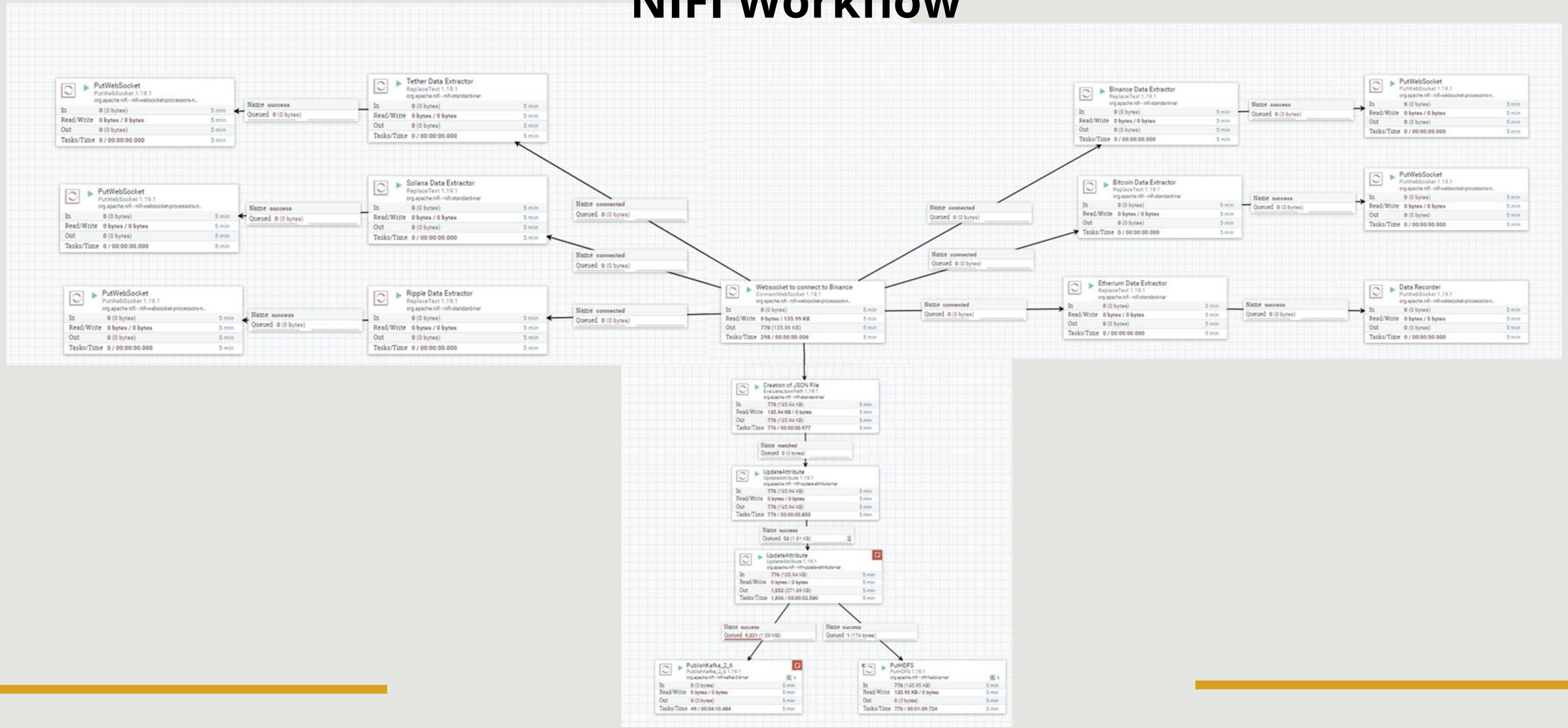
- Publish Kafka
- PutHDFS

Update Processors

Arranges the collected data into timestamps.

DATA INGESTION

NIFI Workflow



DATA STORAGE

INGESTION

Now that we've established our data ingestion flow that stores information in a distributed manner. This is achieved through the PutHDFS processor, which divides the files into smaller blocks.

Directory: /datalake/raw/binance/bitcoin/2023/12/13

STORAGE

/datalake/raw/binance/bitcoin/2023/12/13 Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	osbdet	hadoop	1.64 MB	Dec 13 14:46	1	128 MB	2023121300.json
-rw-r--r--	osbdet	hadoop	295.94 KB	Dec 13 14:59	1	128 MB	2023121314.json
-rw-r--r--	osbdet	hadoop	1.24 MB	Dec 13 16:00	1	128 MB	2023121315.json
-rw-r--r--	osbdet	hadoop	1.73 MB	Dec 13 17:00	1	128 MB	2023121316.json
-rw-r--r--	osbdet	hadoop	1.71 MB	Dec 13 18:00	1	128 MB	2023121317.json
-rw-r--r--	osbdet	hadoop	854.34 KB	Dec 13 18:59	1	128 MB	2023121318.json
-rw-r--r--	osbdet	hadoop	256.54 KB	Dec 13 19:23	1	128 MB	2023121319.json

Showing 1 to 7 of 7 entries Previous 1 Next

DATA PROCESSING

STORAGE

Setting up the **SparkSession**, which is the entry point for Spark functionality in an application. This will drive our processes as it provides a unified interface for reading data, applying transformations, and executing actions on the data.

Use Spark's data source API to load data from our collected data from HDFS, into a distributed collection of data frame.

Apply various transformations to the DataFrame to prepare the data for analysis. Spark provides a rich set of transformations such as filter, groupBy, join, agg, and custom transformations.

PROCESSING

DATA PROCESSING

EXTRACT

- Nifi Ingestion for Cryptocurrencies
- Used API to obtain and store stocks & commodities data
- Stored data in CSV due to daily API call limit

LOAD

- Load Cryptocurrency data from Hadoop
- Load stocks and commodities CSV

TRANSFORM

- Select relevant data for Cryptos and store in a dataframe
- Transformed stock & commodities data into a useful dataframe
- Merged all asset data into one dataframe

DATA PROCESSING

Codes:

```python
import requests
import pandas as pd

# Replace 'your_access_key_here' with your actual Marketstack API key
api_key = 'f8231fd49456f47512903d99ec6dd6b9'

# Replace 'AAPL' with the desired symbol
symbol = 'GOLD'

# Replace 'target_date_from' and 'target_date_to' with the specific date range you're interested in
target_date_from = '2023-12-04'
target_date_to = '2023-12-14'

# Marketstack API endpoint for intraday data
url = f'http://api.marketstack.com/v1/intraday?access_key={api_key}&symbols={symbol}&date_from={target_date_from}&date_to={target_date_to}'

# Make a GET request to the API
response = requests.get(url)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Parse and extract relevant data from the JSON response
    golddata = response.json()['data']

    # Create a DataFrame from the extracted data
    gold = pd.DataFrame(golddata)

    # Convert 'date' column to pandas datetime object
    gold['date'] = pd.to_datetime(gold['date'])

    # Print the DataFrame
    print(gold)
```

```python
# Define the path to the JSON file
json_file_path = "hdfs://localhost:9000/datalake/raw/binance/bitcoin/2023/*/*/*"
```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	key_0	index	open	high	low	last	close	volume	date	symbol	exchange	hour	level_0	
2	2023-12-1	0	16.28	17.065	16.26	17.06	16.340159	131874	2023-12-1	GOLD	IEXG	20		
3	2023-12-1	0	16.28	17.065	16.26	17.06	16.339955	131874	2023-12-1	GOLD	IEXG	20		
4	2023-12-1	0	16.28	17.065	16.26	17.06	16.340207	131874	2023-12-1	GOLD	IEXG	20		
5	2023-12-1	0	16.28	17.065	16.26	17.06	16.340488	131874	2023-12-1	GOLD	IEXG	20		
6	2023-12-1	0	16.28	17.065	16.26	17.06	16.339924	131874	2023-12-1	GOLD	IEXG	20		
7	2023-12-1	0	16.28	17.065	16.26	17.06	16.339924	131874	2023-12-1	GOLD	IEXG	20		
8	2023-12-1	0	16.28	17.065	16.26	17.06	16.340506	131874	2023-12-1	GOLD	IEXG	20		
9	2023-12-1	0	16.28	17.065	16.26	17.06	16.340246	131874	2023-12-1	GOLD	IEXG	20		
10	2023-12-1	0	16.28	17.065	16.26	17.06	16.339849	131874	2023-12-1	GOLD	IEXG	20		
11	2023-12-1	0	16.28	17.065	16.26	17.06	16.34017	131874	2023-12-1	GOLD	IEXG	20		
12	2023-12-1	0	16.28	17.065	16.26	17.06	16.339851	131874	2023-12-1	GOLD	IEXG	20		
13	2023-12-1	0	16.28	17.065	16.26	17.06	16.339850	131874	2023-12-1	GOLD	IEXG	20		
14	2023-12-1	0	16.28	17.065	16.26	17.06	16.340077	131874	2023-12-1	GOLD	IEXG	20		

Load stocks data

```python
[6]: stocks_df = spark.read\
                        .option("header","true")\
                        .option("inferSchema","true")\
                        .csv("./data/merged_stocks_data.csv")
stocks_df.limit(5).toPandas()
```

t[6]:

	key_0	index	open	high	low	last	close	volume	date	symbol	exchange	hour	level_0
0	2023-12-13 20:00:00+00:00	0.0	195.53	197.995	194.92	197.955	194.710133	834937.0	2023-12-13 20:00:00+00:00	AAPL	IEXG	20	NaN
1	2023-12-13 19:59:00+00:00	0.0	195.53	197.995	194.92	197.955	194.709963	834937.0	2023-12-13 20:00:00+00:00	AAPL	IEXG	20	NaN
2	2023-12-13 19:58:00+00:00	0.0	195.53	197.995	194.92	197.955	194.710174	834937.0	2023-12-13 20:00:00+00:00	AAPL	IEXG	20	NaN
3	2023-12-13 19:57:00+00:00	0.0	195.53	197.995	194.92	197.955	194.710409	834937.0	2023-12-13 20:00:00+00:00	AAPL	IEXG	20	NaN
4	2023-12-13 19:56:00+00:00	0.0	195.53	197.995	194.92	197.955	194.709937	834937.0	2023-12-13 20:00:00+00:00	AAPL	IEXG	20	NaN

```python
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType
import json
import re
from pyspark.sql import functions as F

# Create a Spark session
spark = SparkSession.builder.appName("BinanceData").getOrCreate()

# Manually specify schema
custom_schema = StructType([
    StructField("event_type", StringType(), True),
    StructField("event_time", StringType(), True),
    StructField("symbol", StringType(), True),
    StructField("close_price", StringType(), True),
    StructField("open_price", StringType(), True),
    StructField("high_price", StringType(), True),
    StructField("low_price", StringType(), True),
    StructField("total_volume", StringType(), True),
    StructField("total_quote_asset_volume", StringType(), True),
])

# Define the path to the JSON file
json_file_path = "hdfs://localhost:9000/datalake/raw/binance/bitcoin/2023/*/*/*"

# Read JSON data as text
json_text_rdd = spark.sparkContext.textFile(json_file_path)
```

Load Bitcoin data from Hadoop

```python
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, DoubleType, LongType
import json
import re
from pyspark.sql import functions as F
```

EXTRACT

LOAD

DATA PROCESSING

Codes:

```
stocks_df = stocks_df.withColumn(
    "date_time",
    F.to_timestamp("key_0", "MM/dd/yyyy HH:mm")
)
# Order the DataFrame by day
columns_to_drop = ['date_time', 'index', 'level_0', 'date', 'exchange','hour','date_time']
stocks_df = stocks_df.drop(*columns_to_drop)

stocks_df = stocks_df \
    .withColumnRenamed('key_0', 'date_time') \
    .withColumnRenamed('open', 'open_price') \
    .withColumnRenamed('high', 'high_price') \
    .withColumnRenamed('low', 'low_price') \
    .withColumnRenamed('last', 'last_price') \
    .withColumnRenamed('close', 'close_price')

stocks_df = stocks_df.withColumn(
    "date_time",
    F.date_format("date_time", "yyyy-MM-dd HH:mm")
)
stocks_df = stocks_df.withColumn(
    "percentage_revenue",
    F.expr("(close_price - open_price) / open_price")
)

stocks_df = stocks_df.orderBy("date_time")

stocks_df.show(truncate=False)
```

```
# Select relevant columns
crypto_df = df.select("symbol", "event_time", "open_price", "close_price","high_price","low_price")

# Convert event_time to timestamp
crypto_df=crypto_df.withColumn("event_time", (F.col("event_time") / 1000).cast(T.TimestampType()))

# Round the timestamp to the nearest minute
crypto_df = crypto_df.withColumn("minute", F.date_format("event_time", "yyyy-MM-dd HH:mm"))

# Group by symbol and minute, then calculate the maximum close_price and open_price
crypto_df = crypto_df.groupBy("symbol", "minute").agg(
    F.last("event_time").alias("event_time"),
    F.first("open_price").alias("open_price"),
    F.last("close_price").alias("close_price"),
    F.max("high_price").alias("high_price"),
    F.min("low_price").alias("low_price")
).orderBy(F.asc("event_time"))

#Add % revenue
crypto_df = crypto_df.withColumn("percentage_revenue", ((F.col("close_price") - F.col("open_price")) / F.col("open_price")))

crypto_df = crypto_df.drop("event_time")

# Rename the columns
crypto_df = crypto_df.withColumnRenamed("minute", "date_time")

# Show the DataFrame
crypto_df.limit(5).toPandas()
```

```
# Add a column indicating the source DataFrame
crypto_data = crypto_df.withColumn("source_df", lit("crypto"))
stocks_data = stocks_df.withColumn("source_df", lit("stocks"))
commodities_data = commodities_df.withColumn("source_df", lit("commodities"))

# Select and rename columns from crypto_df
crypto_data = crypto_data.selectExpr("symbol", "CAST(date_time AS TIMESTAMP) as date_time", "open_price", "close_price", "percent

# Select and rename columns from stocks_df
stocks_data = stocks_data.selectExpr("symbol", "date_time", "open_price", "close_price", "percentage_revenue", "source_df")

# Select and rename columns from commodities_df
commodities_data = commodities_data.selectExpr("symbol", "CAST(date_time AS TIMESTAMP) as date_time", "open_price", "close_price"

# Union all the data into the empty DataFrame
assets_df = empty_df.union(crypto_data).union(stocks_data).union(commodities_data)

# Show the result
assets_df.toPandas()
```

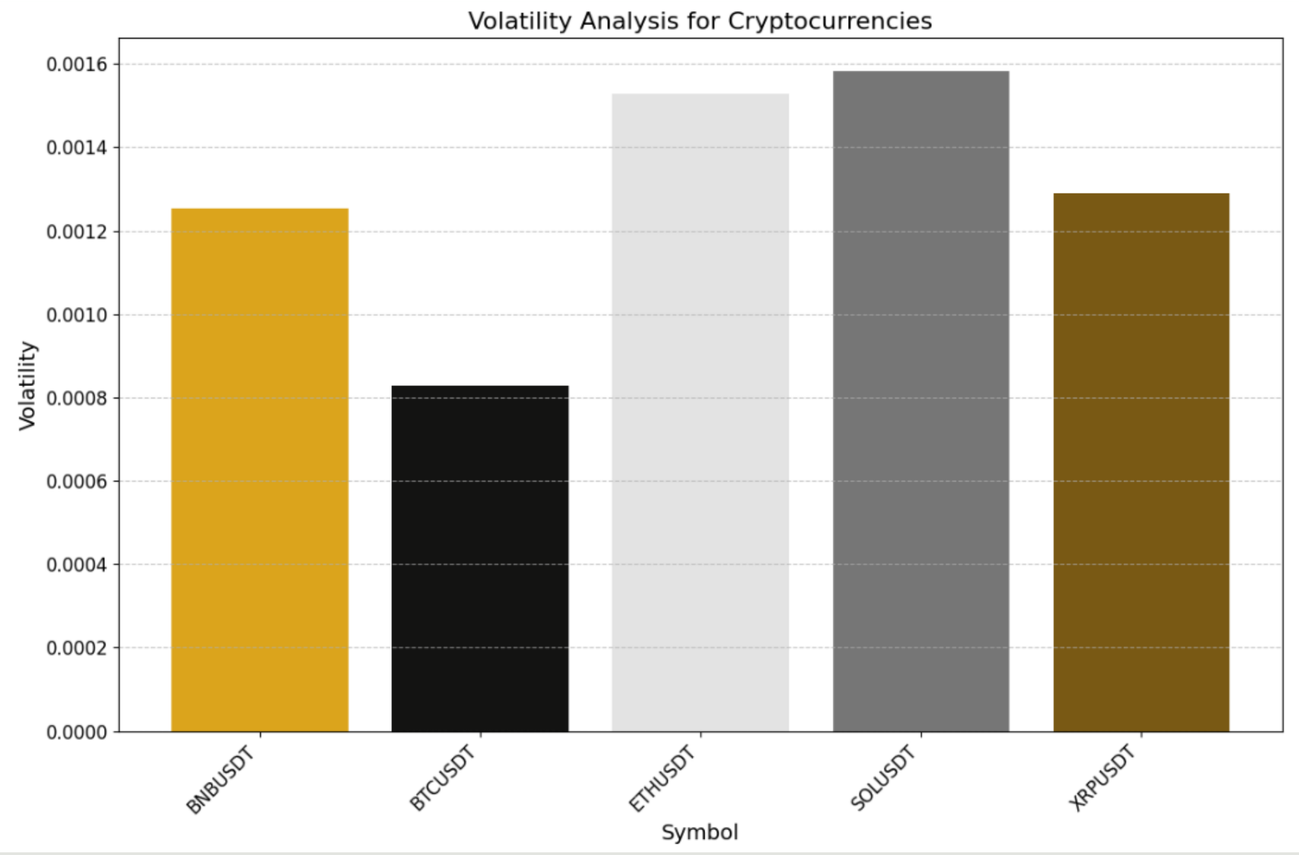
TRANSFORM

DATA PROCESSING

Data Analytics : Cryptocurrency

We have found that opening a position at the beginning of our data ingestion and closing it at the end would have yielded a return of 3.8% for Bitcoin BTC.

symbol	coun	min_event_tim	max_event_tim	avg_close_price	min_close_pric	max_close_pric	total_return
ETHUSDT	8415	12/11/2023 21:27	12/13/2023 15:37	2195.08	2171.07	2218.50	0.001341
BTCUSDT	**31799**	**12/11/2023 22:15**	**12/13/2023 20:38**	**41661.40**	**41034.52**	**42813.08**	**0.038412**
BNBUSDT	18825	12/12/2023 19:21	12/13/2023 18:17	253.44	249.2	256.90	-0.003959
SOLUSDT	19948	12/12/2023 21:33	12/13/2023 18:17	67.61	66.41	69.08	0.006272
XRPUSDT	456	12/12/2023 21:33	12/12/2023 22:05	0.62	0.61	0.62	0.006839



3.8%
BTC Return

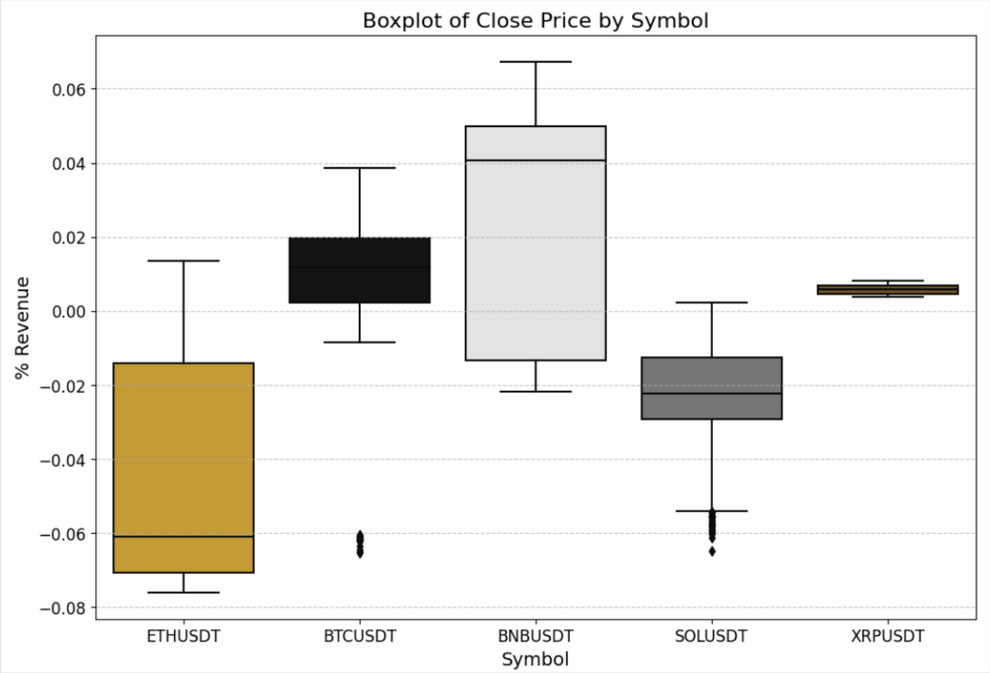
Volatility Wise we have observed that Solana (SOLUSDT) have been the most volatile and Bitcoin (BTCUSDT) has been least one.

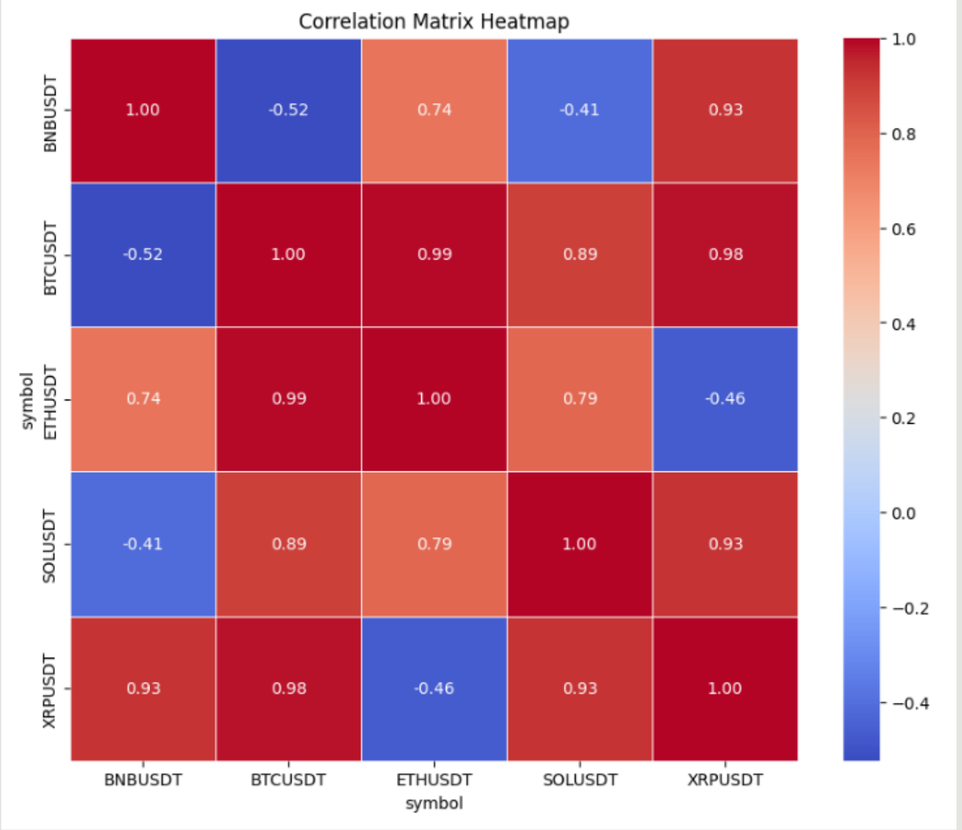
Although BTC has shown the lowest volatility. It has demonstrated that the it follows an upward trend during the data collection process.

DATA PROCESSING

Data Analytics : Cryptocurrency

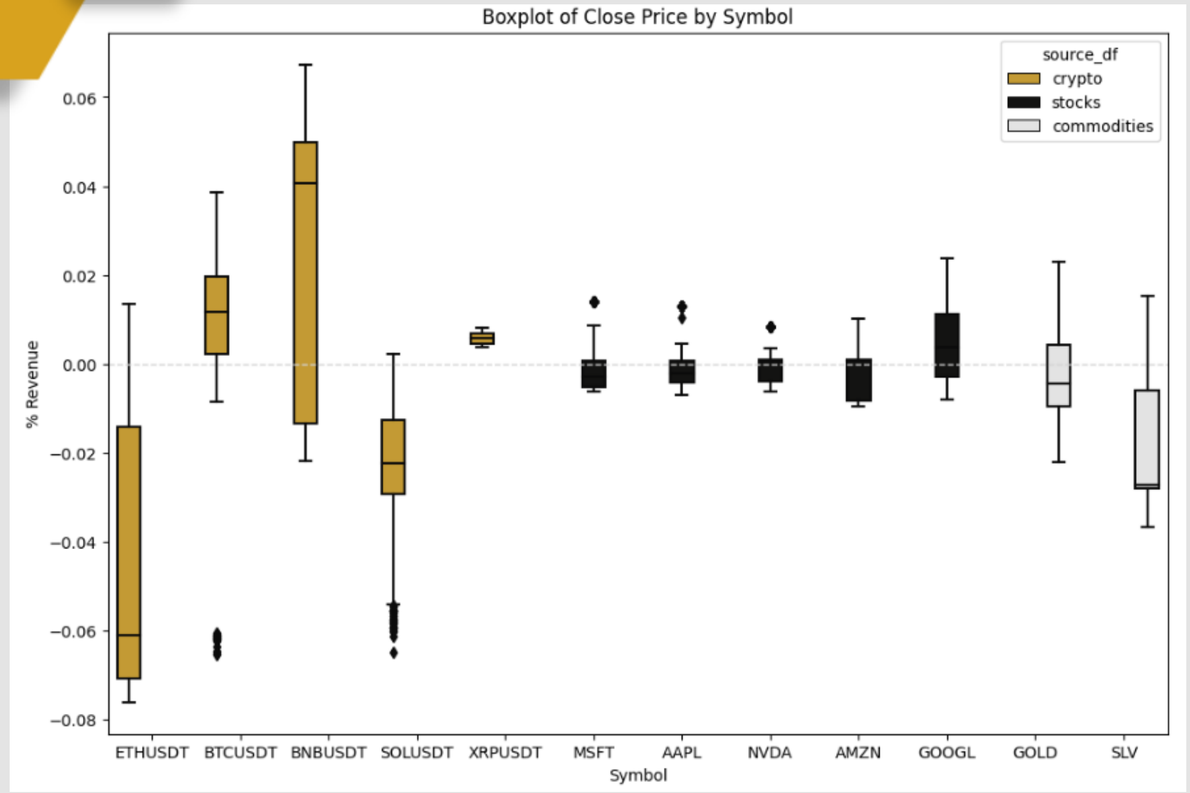
We plotted the return per minute of each Crypto coin. The coin with the highest average return for minute is BNB at 4%. The one with the lowest average return is ETH at -6%.

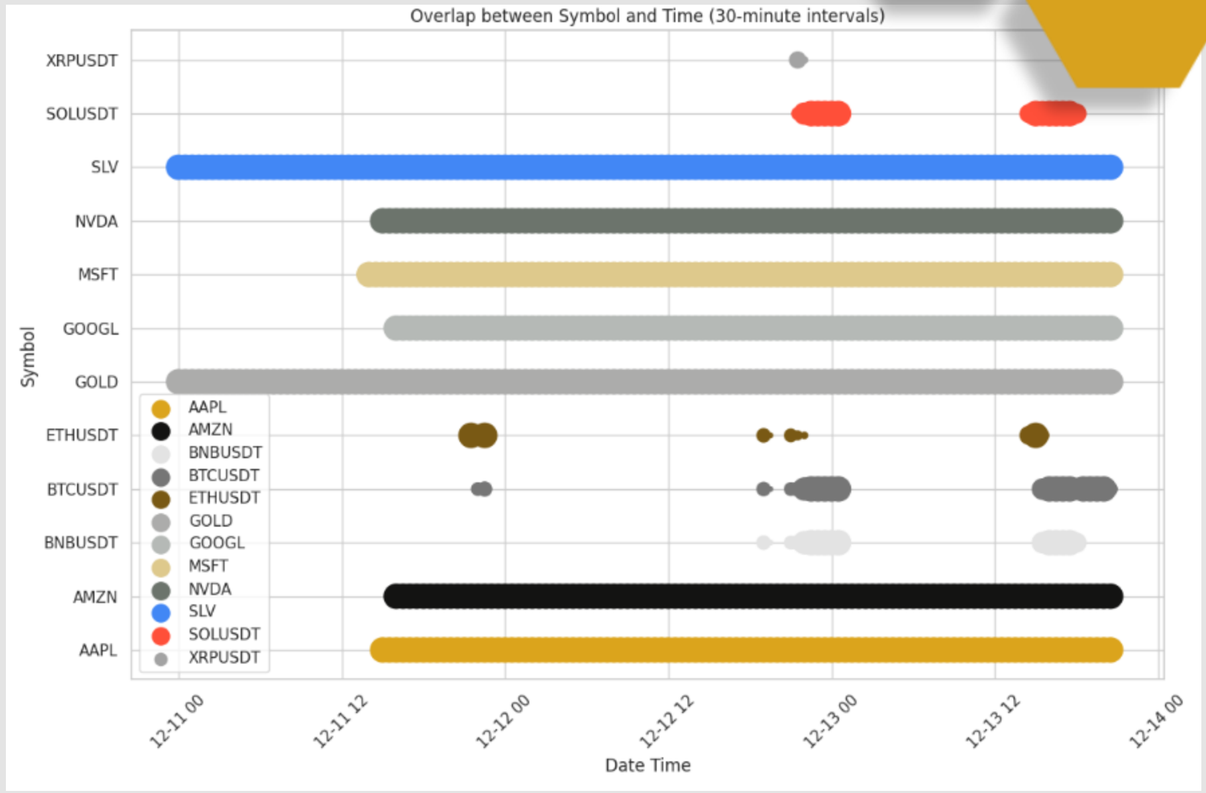




- Correlations among all crypto are high.
- the exceptions are bnb-sol,eth-xrp,bnb-btc
- highest correlation in all data set is btc-eth with .99

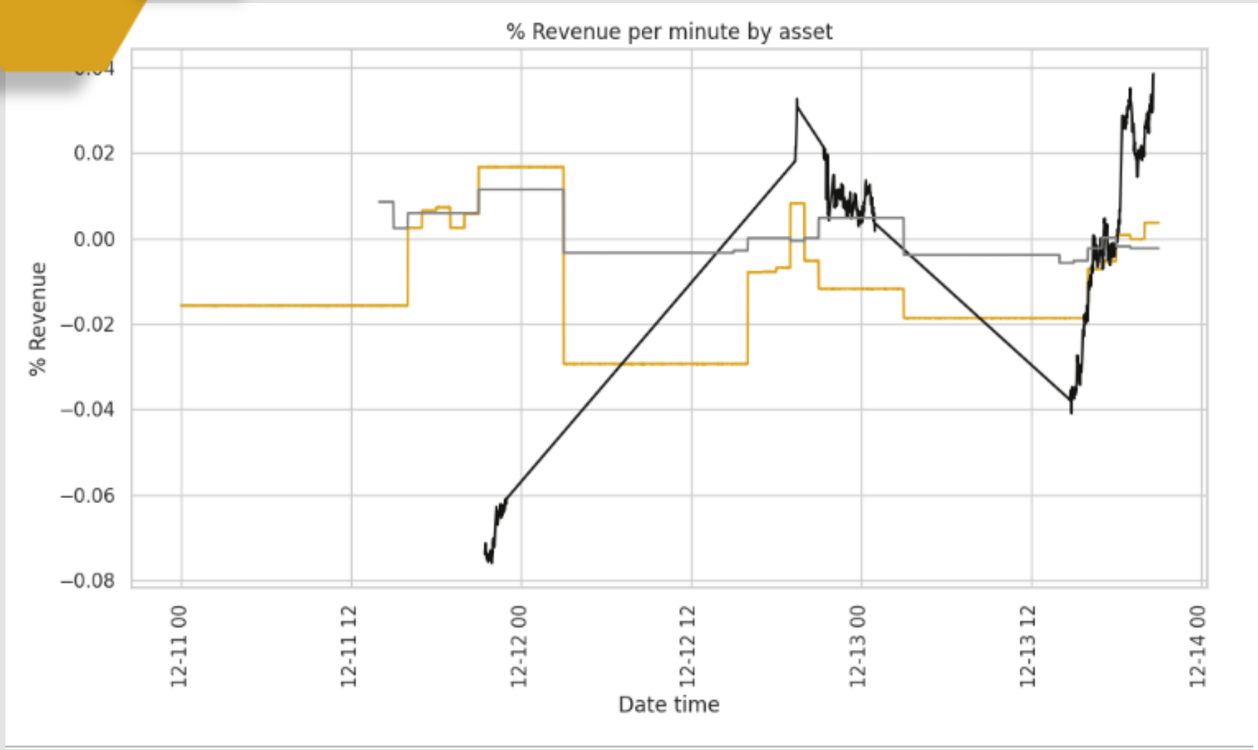
PROJECT RESULTS





Minute-by minute overlap for the data shows us the times we obtained data from Binance. Cryptocurrency data was only obtained while Nifi was running, approximately 3.5 hours of data. For stocks and commodities, we obtained all data from the days where we ran Nifi.

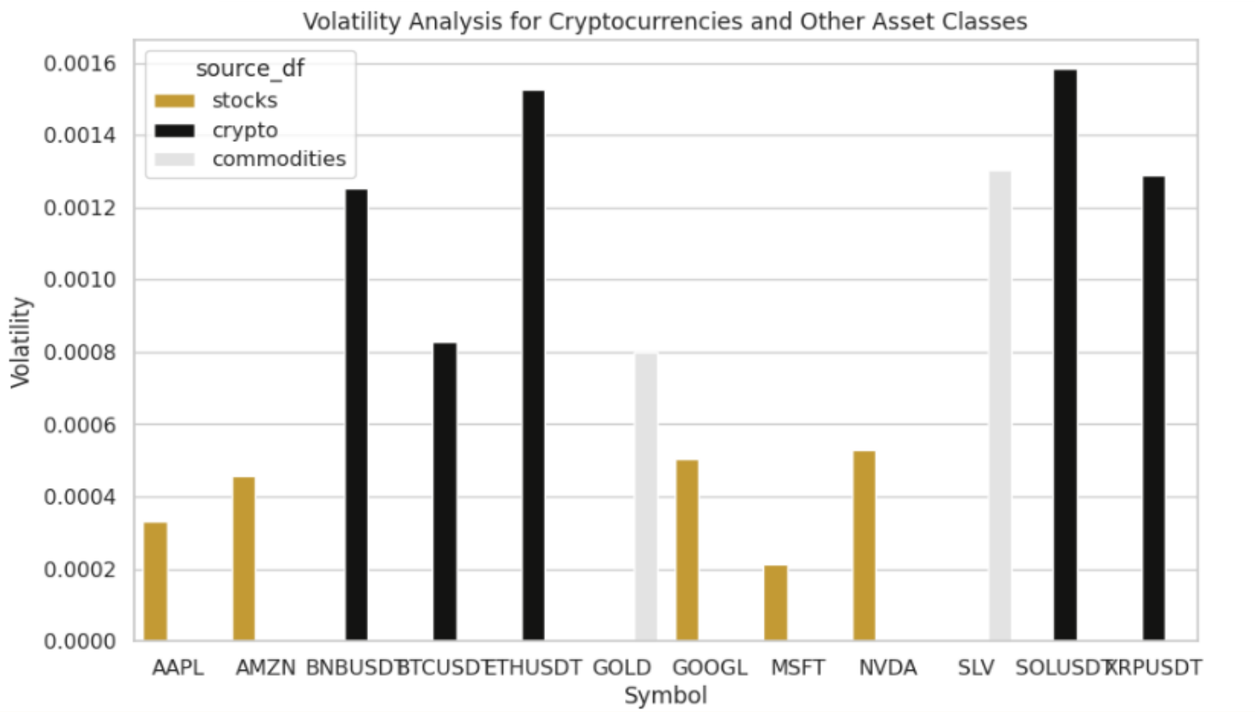
PROJECT RESULTS



Long straight lines represent time where data wasn't obtained.

Stocks and commodities have constant slow growth.

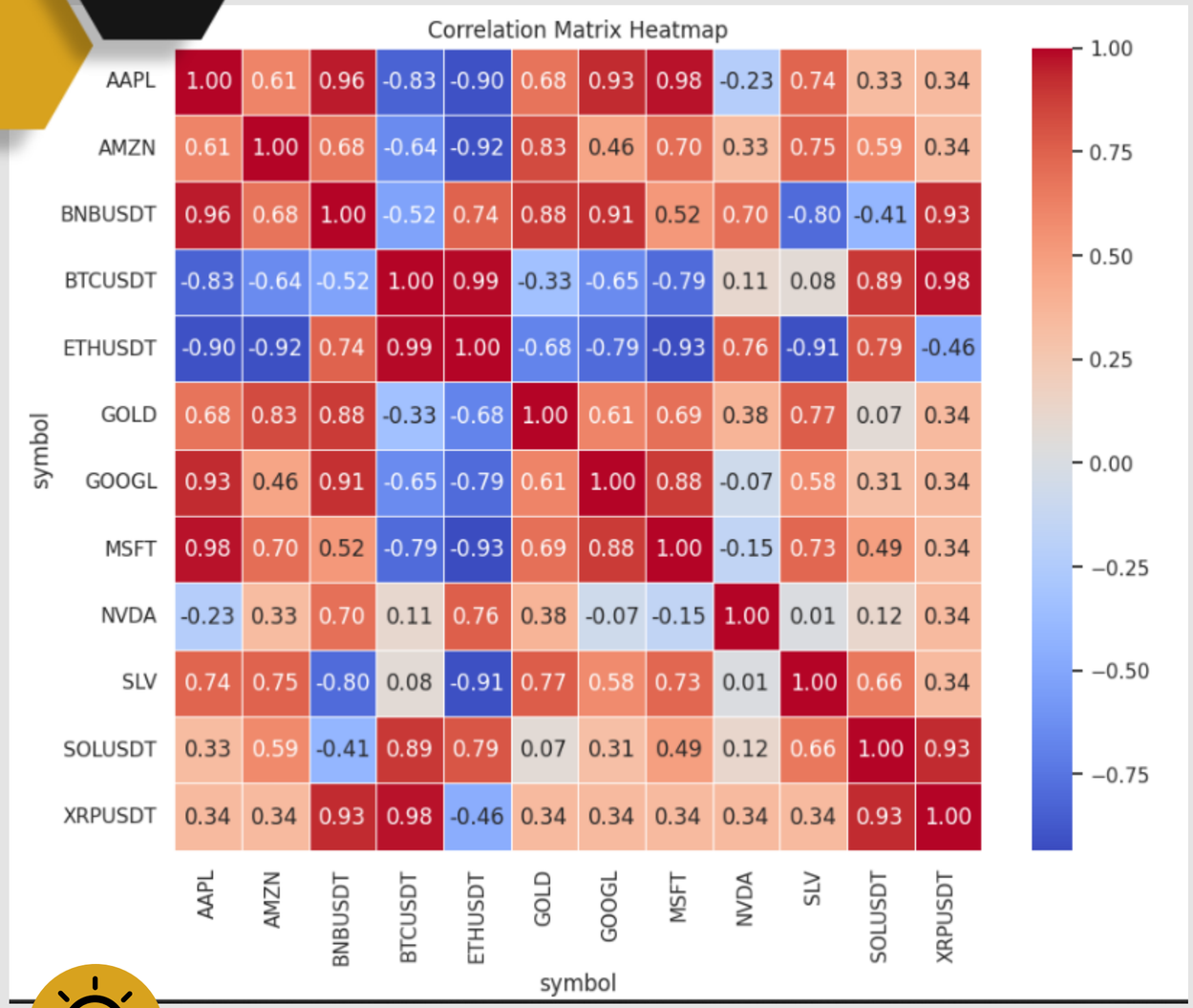
Cryptos are more volatile and have more movement over short periods of time. On the last data collection period for crypto, it showed a 4% increase on the opening vs. closing price.



Cryptos have a higher overall volatility tan the other assets

The highest volatility belongs to SOL, followed closely by ETH

PROJECT RESULTS



Cryptos vs. commodities

- ETH has a negative correlation with commodities.
- BTC has a -.33 correlation with gold.
- BNB has a -.80 correlation with silver.

Cryptos vs. Stock

- BTC has a high negative correlation with 4/5 stocks. Except Nvidia.
- AAPL,AMZN,GOOGL,MSFT have a high negative correlation with BTC.
- BNB has a high correlation with all stock, the highest being AAPL at .96.

General

- For XRP there are too few data points(<1hr) to show meaningful correlation patterns.

DATA VALUE CHAIN

SOURCES	INGESTION	STORAGE	PROCESSING	SERVING
BINANCE marketstack	APACHE nifi	hadoop	APACHE Spark™	seaborn

CONCLUSION

- There is no direct correlation between cryptocurrencies and leading asset stocks and commodities
- Cryptocurrencies have a similar volatile behavior
- BTC is the leading cryptocurrency due to its overall valuation and it establishes a benchmark for the rest of cryptocurrencies
- The behavior and future price of cryptocurrencies is not predictable and dependent on the general sentiment over top leading stocks and commodity valuation
- There are factors such as regulatory developments, circulating supply, technological advances and breaking news around the top crypto exchanges which can shift the behavior of crypto in the future
- The crypto market is relatively young and exhibits greater volatility compared to traditional markets, thus our insights shouldn’t be taken as truths, investing in these assets has inherent risks

THANK YOU

REFERENCES

BINANCE

marketstack