[Mohammad Hamza Ibrahim]
[M00737296]

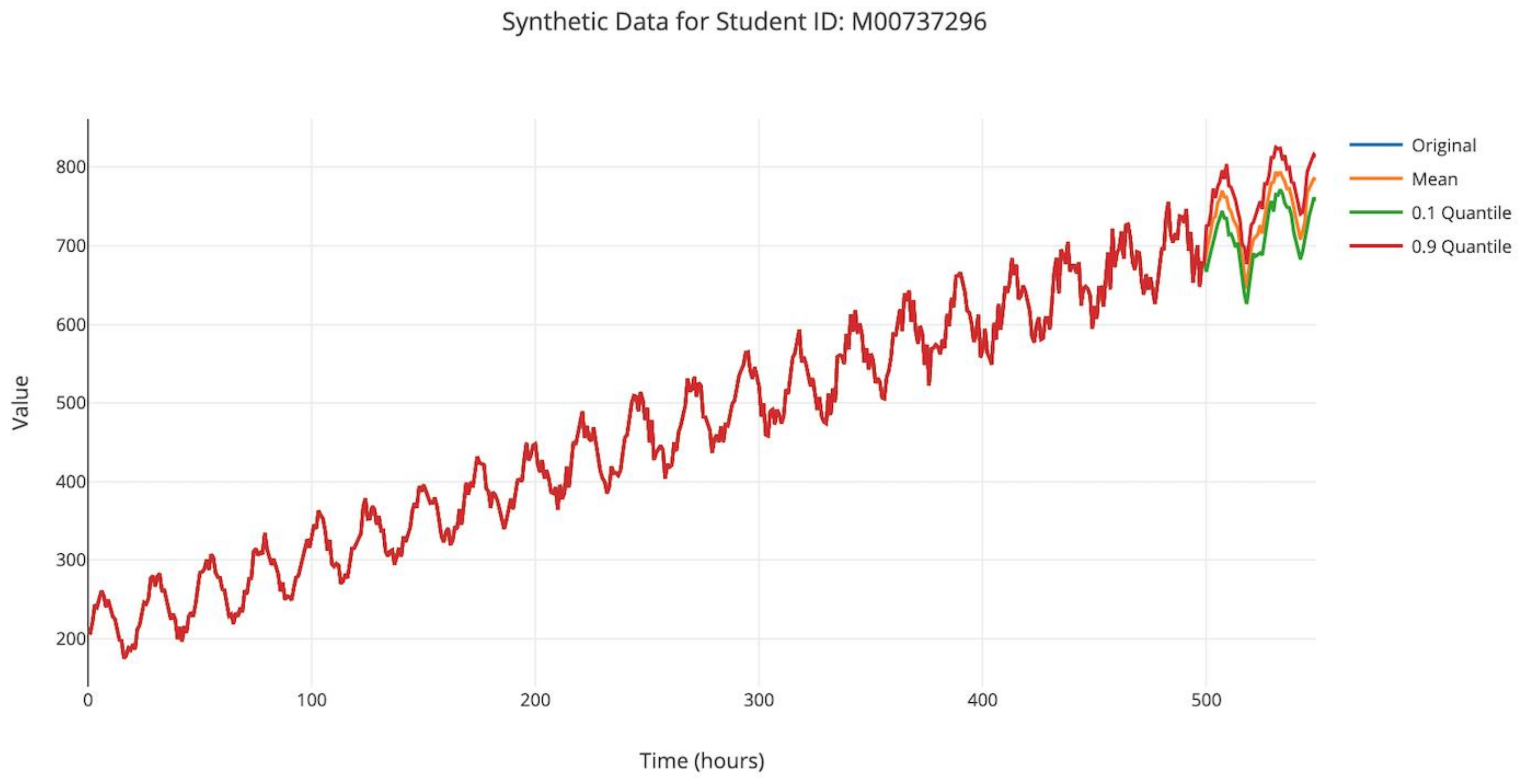
# PROJECT REPORT

[Coursework 2 / Data Visualization Website]

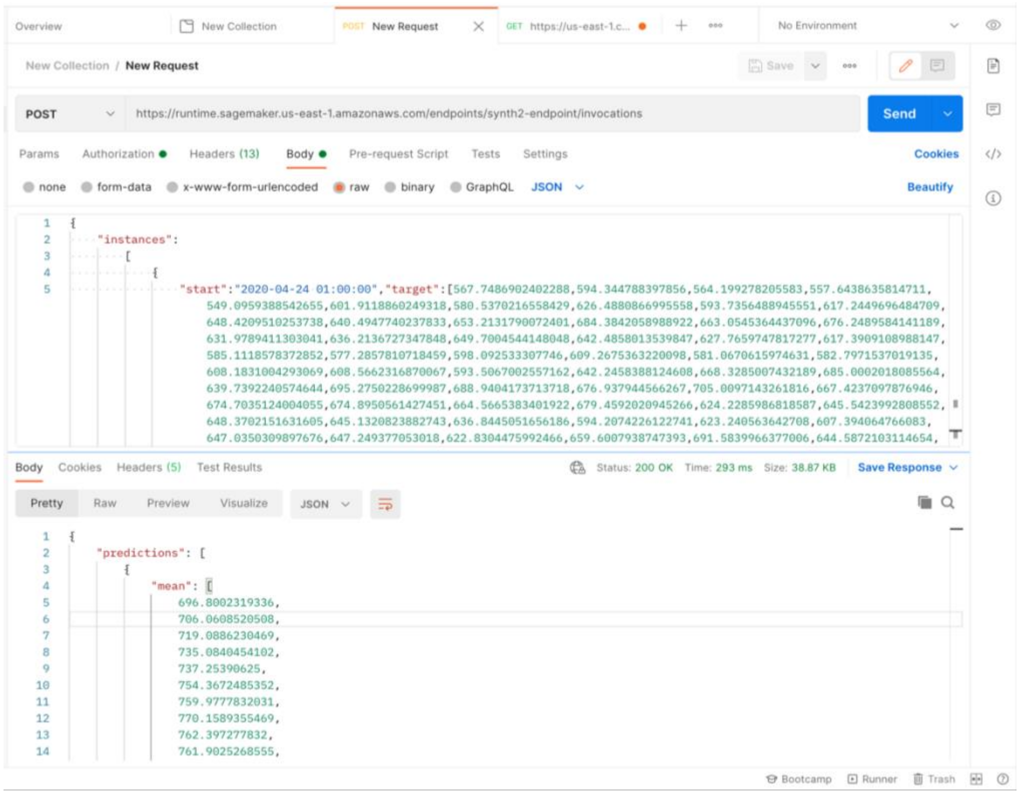
## 1. Summary

Within this project I have produce a crypto data visualization website. This website includes 5 crypto currencies (BTC, ETH, LUNA, DOGE, XLM). Both numerical and sentimental data are displayed using Plotly. The numerical data is represented in a graph format showing crypto rates (in GBP) alongside the time (in mm/dd/yyyy format). the numerical data was obtained using third party web service cryptocompare (https://www.cryptocompare.com/ ). The sentiment data for all 5 crypto currencies are shown using a pie chart showing positive, negative, neutral, and mixed results from tweets about the crypto. The front end of the website was created using html, css, and javascript. This is hosted in the cloud using Amazon S3. The back end is also run in the cloud with serverless technology.

## 2. Machine learning



Within above figure it shows the prediction of my synthetic data that was given to me. To get this prediction, first a script was written to get the dataset and partitioned into test and train set. (train has 400 datapoints and test has entire dataset) I then Upload those train and test data to S3. A training job was created to build a model from the S3 data. Sagemaker was used to build, train and deploy machine learning models. After model was created, I deployed it to an endpoint. I Used built in algorithm DeepAR to estimate future state of time series. To get the prediction I Queried the endpoint by using postman: -



New Collection / New Request

```
POST https://runtime.sagemaker.us-east-1.amazonaws.com/endpoints/synth2-endpoint/invocations
```

```
{
    "instances":
        [
            {
                "start":"2020-04-24 01:00:00","target":[567.7486902402288,594.344788397856,564.199278205583,557.6438635814711,
                    549.0959388542655,601.9118860249318,580.5370216558429,626.4880866995558,593.7356488945551,617.2449696484709,
                    648.4209510253738,640.4947740237833,653.2131790072401,684.3842058988922,663.0545364437096,676.2489584141189,
                    631.9789411303041,636.2136727347848,649.7004544148048,642.4858013539847,627.7659747817277,617.3909108988147,
                    585.1118578372852,577.2857810718459,598.092533307746,609.2675363220098,581.0670615974631,582.7971537019135,
                    608.1831004293069,608.5662316870067,593.5067002557162,642.2458388124608,668.3285007432189,685.0002018085564,
                    639.7392240574644,695.2750228699987,688.9404173713718,676.937944566267,705.0097143261816,667.4237097876946,
                    674.7035124004055,674.8950561427451,664.5665383401922,679.4592020945266,624.2285986818587,645.5423992808552,
                    648.3702151631605,645.1320823882743,636.8445051656186,594.2074226122741,623.240563642708,607.394064766083,
                    647.0350309897676,647.249377053018,622.8304475992466,659.6007938747393,691.5839966377006,644.5872103114654,
```

Status: 200 OK Time: 293 ms Size: 38.87 KB

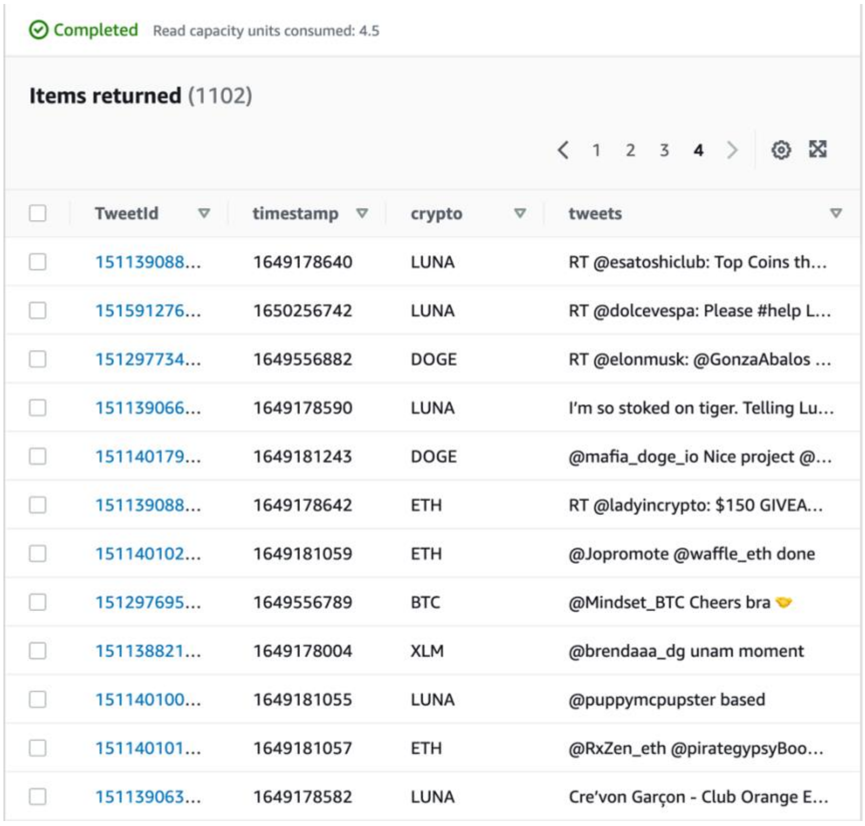
```
{
    "predictions": [
        {
            "mean": [
                696.8002319336,
                706.0608520508,
                719.0886230469,
                735.0840454102,
                737.25390625,
                754.3672485352,
                759.9777832031,
                770.1589355469,
                762.397277832,
                761.9025268555,
```

I then simply visualised the predictions using Potly. And this can be seen here: - https://chart-studio.plotly.com/~Hamza_Ibrahim/0/synthetic-data-for-student-id-m00737296/#/

# 3. Sentiment data and analysis

To produce sentiment, I used the Twitter API to download tweets based on the crypto currencies. By using typescript, I was able to store the correct crypto data into my DynamoDB table (with 1102 data points):-

**Partition Key** – TweetId
**Sort Key** – timestamp



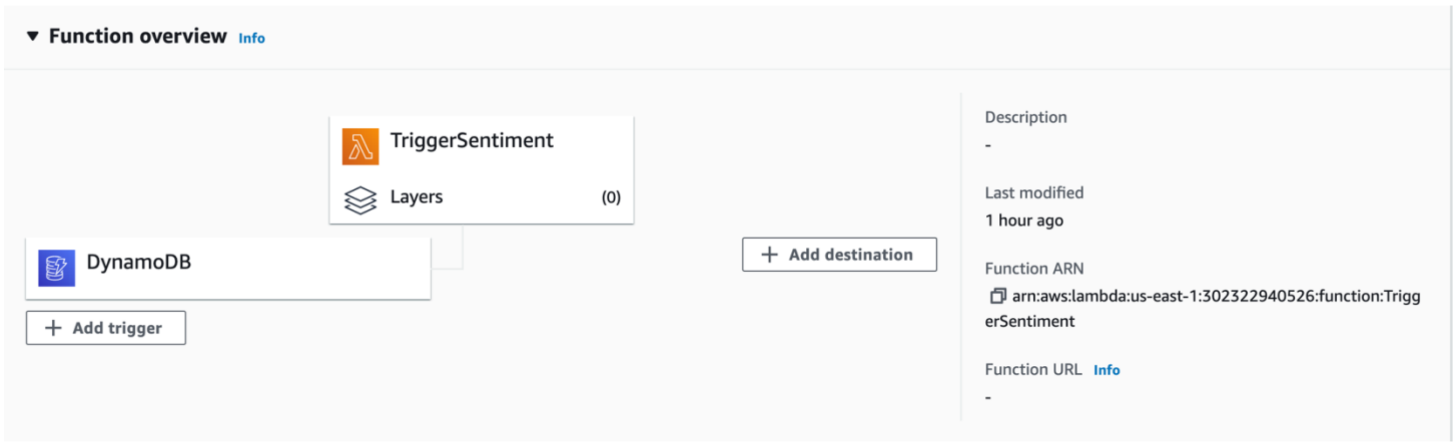
Completed Read capacity units consumed: 4.5

**Items returned** (1102)

| TweetId | timestamp | crypto | tweets |
|---|---|---|---|
| 151139088... | 1649178640 | LUNA | RT @esatoshiclub: Top Coins th... |
| 151591276... | 1650256742 | LUNA | RT @dolcevespa: Please #help L... |
| 151297734... | 1649556882 | DOGE | RT @elonmusk: @GonzaAbalos ... |
| 151139066... | 1649178590 | LUNA | I'm so stoked on tiger. Telling Lu... |
| 151140179... | 1649181243 | DOGE | @mafia_doge_io Nice project @... |
| 151139088... | 1649178642 | ETH | RT @ladyincrypto: $150 GIVEA... |
| 151140102... | 1649181059 | ETH | @Jopromote @waffle_eth done |
| 151297695... | 1649556789 | BTC | @Mindset_BTC Cheers bra 🤝 |
| 151138821... | 1649178004 | XLM | @brendaaa_dg unam moment |
| 151140100... | 1649181055 | LUNA | @puppymcpupster based |
| 151140101... | 1649181057 | ETH | @RxZen_eth @pirategypsyBoo... |
| 151139063... | 1649178582 | LUNA | Cre'von Garçon - Club Orange E... |

**CryptoTweet:** stores twitter posts for each crypto

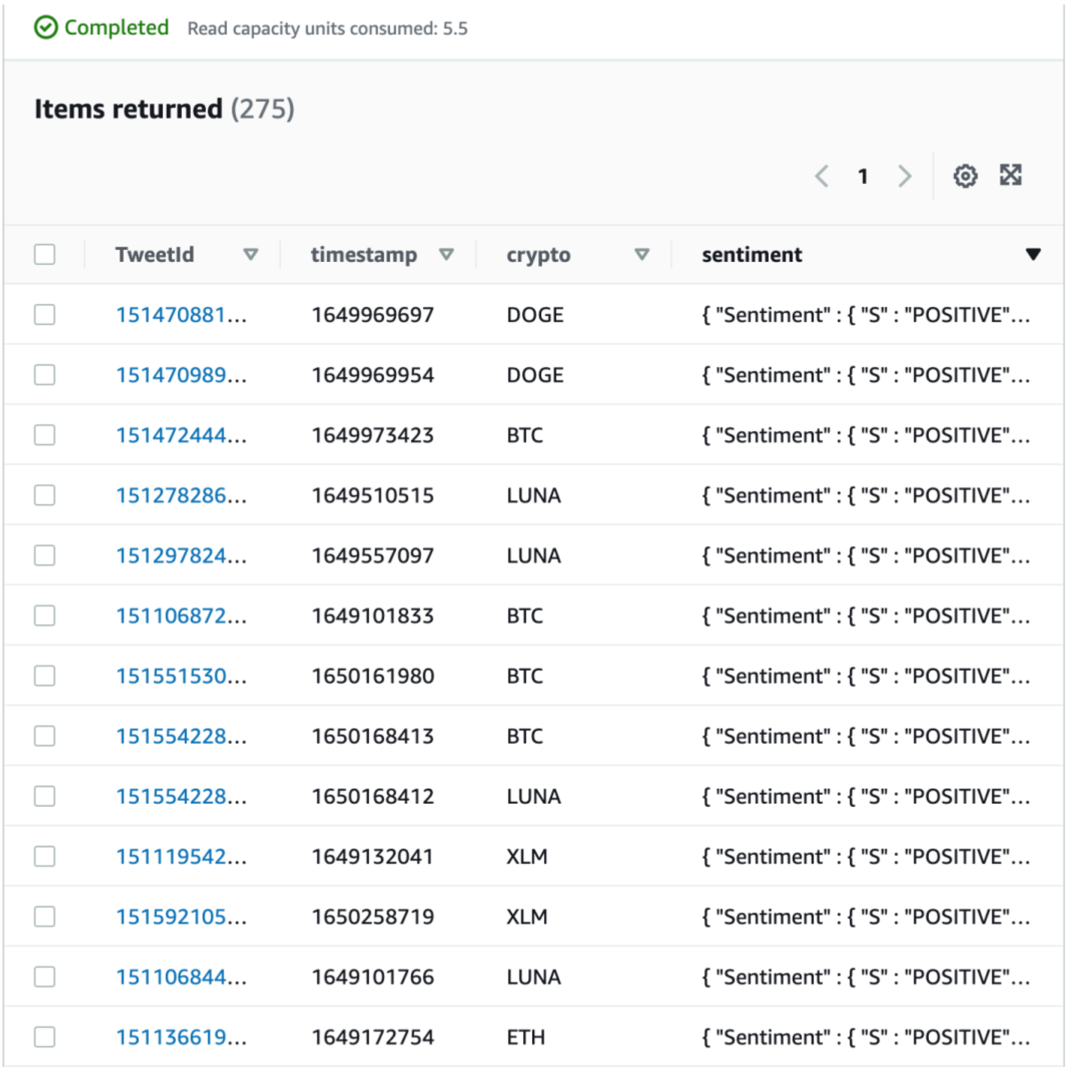
A lambda function was used to generate sentiment for those tweets by using Amazon Comprehend, which uses machine learning to find relationships in text for sentiment analysis. This lambda function also contains a DynamoDB trigger in which the lambda function runs when an items is inserted into the table (CryptoTweet table above):-



Description
-
Last modified
1 hour ago
Function ARN
arn:aws:lambda:us-east-1:302322940526:function:TriggerSentiment
Function URL Info
-

When an item is inserted the lambda function does the sentiment analysis on the tweets and then inserts the results into another DynamoDB table which holds TweetId, timestamp, crypto and results for sentiment analysis: -

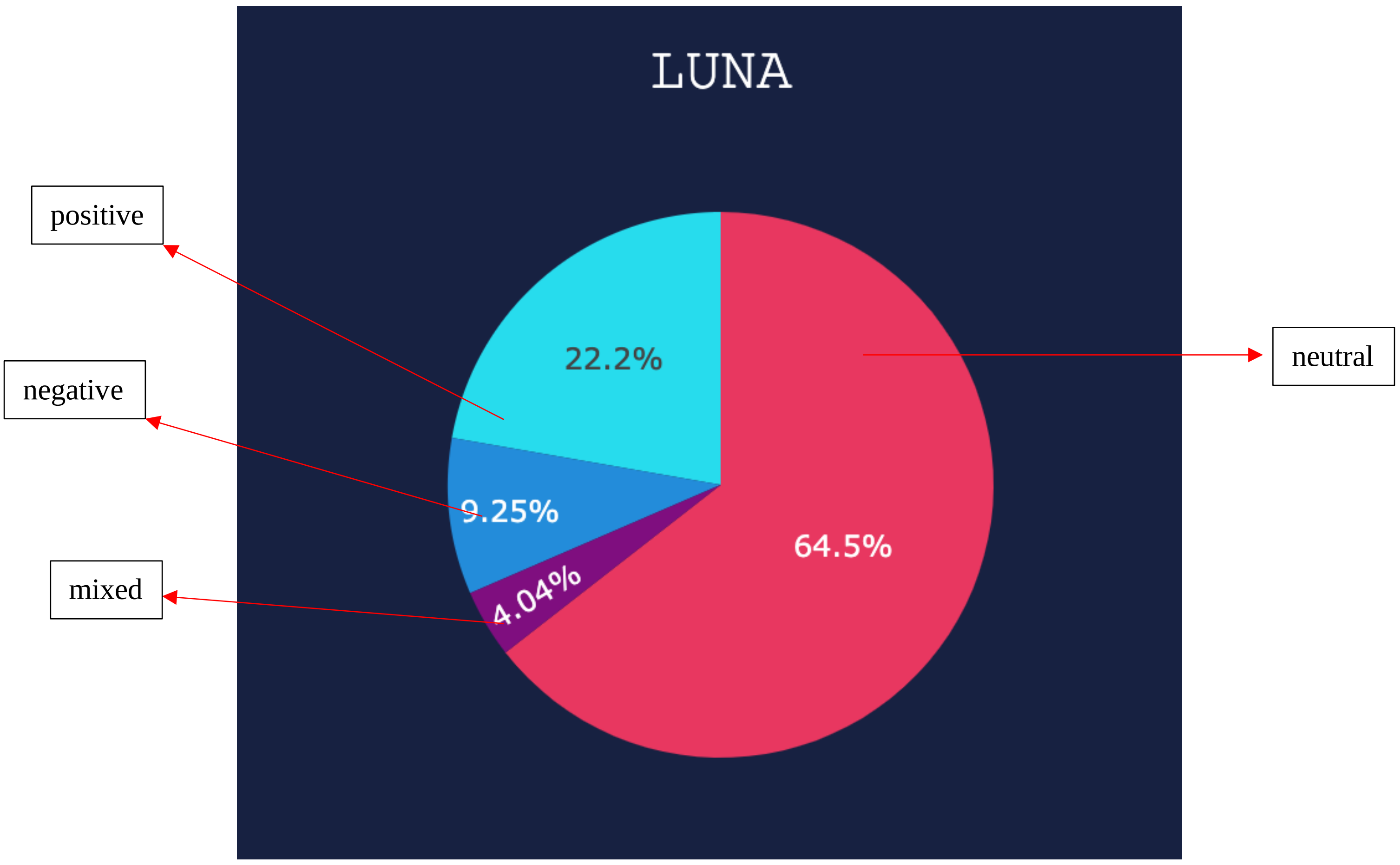
**Partition Key** – TweetId
**Sort Key** – timestamp



| TweetId | timestamp | crypto | sentiment |
|---|---|---|---|
| 151470881... | 1649969697 | DOGE | { "Sentiment" : { "S" : "POSITIVE"... |
| 151470989... | 1649969954 | DOGE | { "Sentiment" : { "S" : "POSITIVE"... |
| 151472444... | 1649973423 | BTC | { "Sentiment" : { "S" : "POSITIVE"... |
| 151278286... | 1649510515 | LUNA | { "Sentiment" : { "S" : "POSITIVE"... |
| 151297824... | 1649557097 | LUNA | { "Sentiment" : { "S" : "POSITIVE"... |
| 151106872... | 1649101833 | BTC | { "Sentiment" : { "S" : "POSITIVE"... |
| 151551530... | 1650161980 | BTC | { "Sentiment" : { "S" : "POSITIVE"... |
| 151554228... | 1650168413 | BTC | { "Sentiment" : { "S" : "POSITIVE"... |
| 151554228... | 1650168412 | LUNA | { "Sentiment" : { "S" : "POSITIVE"... |
| 151119542... | 1649132041 | XLM | { "Sentiment" : { "S" : "POSITIVE"... |
| 151592105... | 1650258719 | XLM | { "Sentiment" : { "S" : "POSITIVE"... |
| 151106844... | 1649101766 | LUNA | { "Sentiment" : { "S" : "POSITIVE"... |
| 151136619... | 1649172754 | ETH | { "Sentiment" : { "S" : "POSITIVE"... |

**TweetSentiment:** stores the sentiment results for each crypto

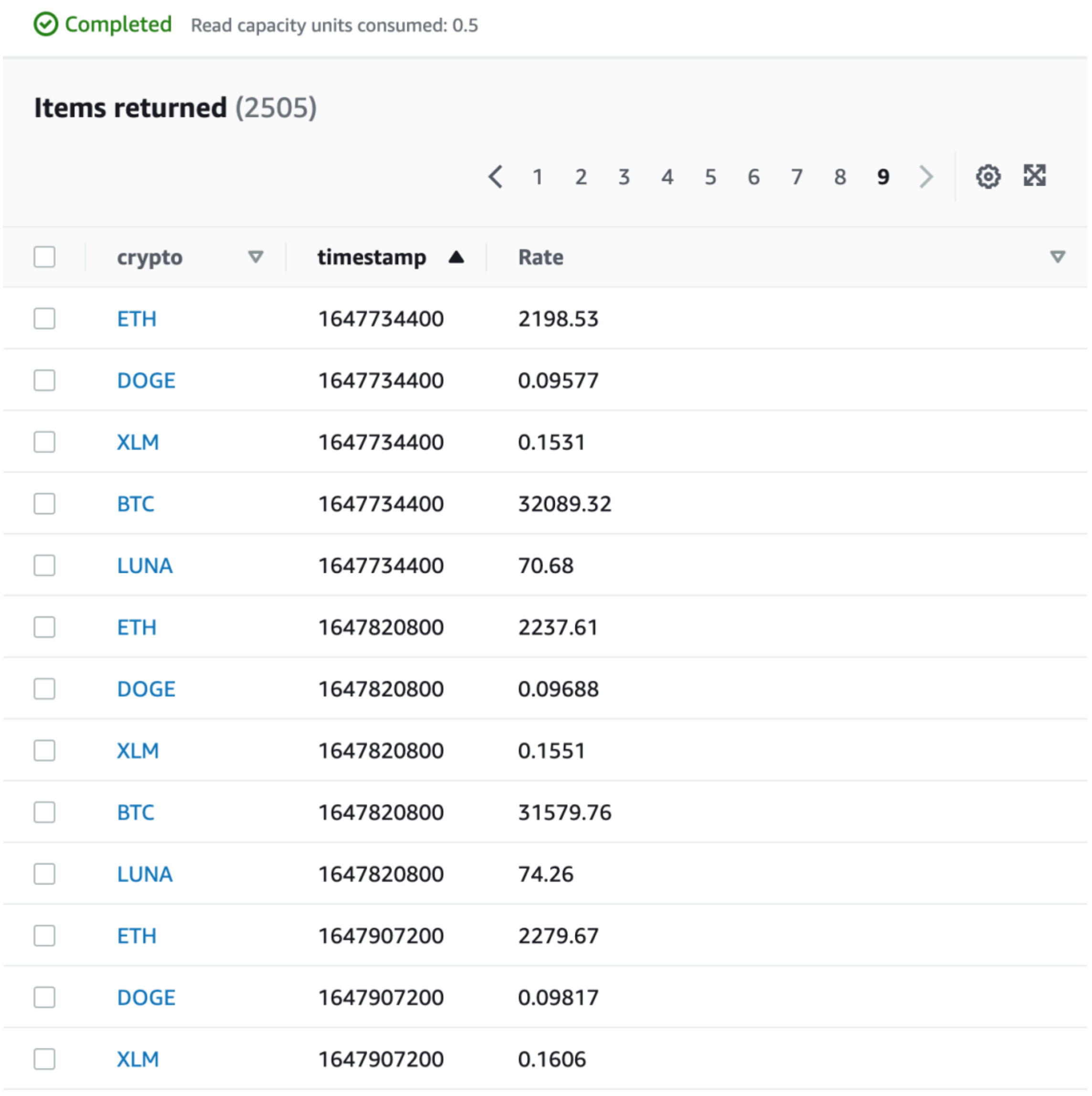
I then visualised the results of sentiment analysis in a pie chart. By overseeing a range of data point with the sentiment analysis results (positive, negative, mixed and neutral). I then calculated the average of the sentiment (positive, negative, mixed and neutral) for each crypto and then displayed it as percentages in the pie chart when hover over the pie chart it shows the sentiments (positive, negative, mixed and neutral): -



## 4. Numerical data

For numerical data I used CryptoCompare API to download targeted crypto currencies (BTC, ETH, DOGE, LUNA, XLM) data, typescript was used to insert these targeted data in a DynamoDB table. This will then be used as a trigger for a lambda function which will send data to a single client, with the help of WebSocket and API gateway.

**Partition Key** – crypto
**Sort Key** – timestamp


Completed Read capacity units consumed: 0.5

**Items returned** (2505)

< 1 2 3 4 5 6 7 8 **9** >

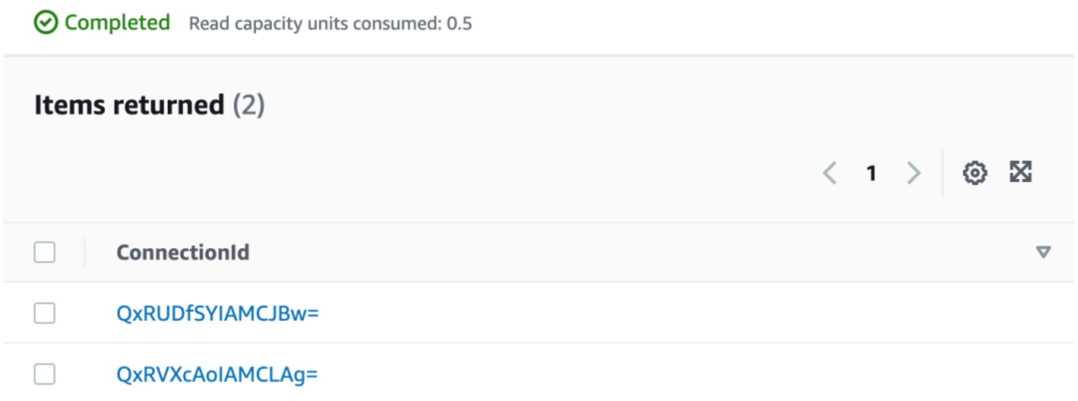
| crypto | timestamp | Rate |
|---|---|---|
| ETH | 1647734400 | 2198.53 |
| DOGE | 1647734400 | 0.09577 |
| XLM | 1647734400 | 0.1531 |
| BTC | 1647734400 | 32089.32 |
| LUNA | 1647734400 | 70.68 |
| ETH | 1647820800 | 2237.61 |
| DOGE | 1647820800 | 0.09688 |
| XLM | 1647820800 | 0.1551 |
| BTC | 1647820800 | 31579.76 |
| LUNA | 1647820800 | 74.26 |
| ETH | 1647907200 | 2279.67 |
| DOGE | 1647907200 | 0.09817 |
| XLM | 1647907200 | 0.1606 |

**CryptoInfo:** stores the numeric data for each crypto

## 5. WebSocket

For my project I used WebSocket to allow me to connect events to lambda function in order to push data to connected clients. The clients connect to an API gateway and this API gateway calls a lambda function to store connection id of the clients in a DynamoDB table: -

Partition Key – ConnectionId



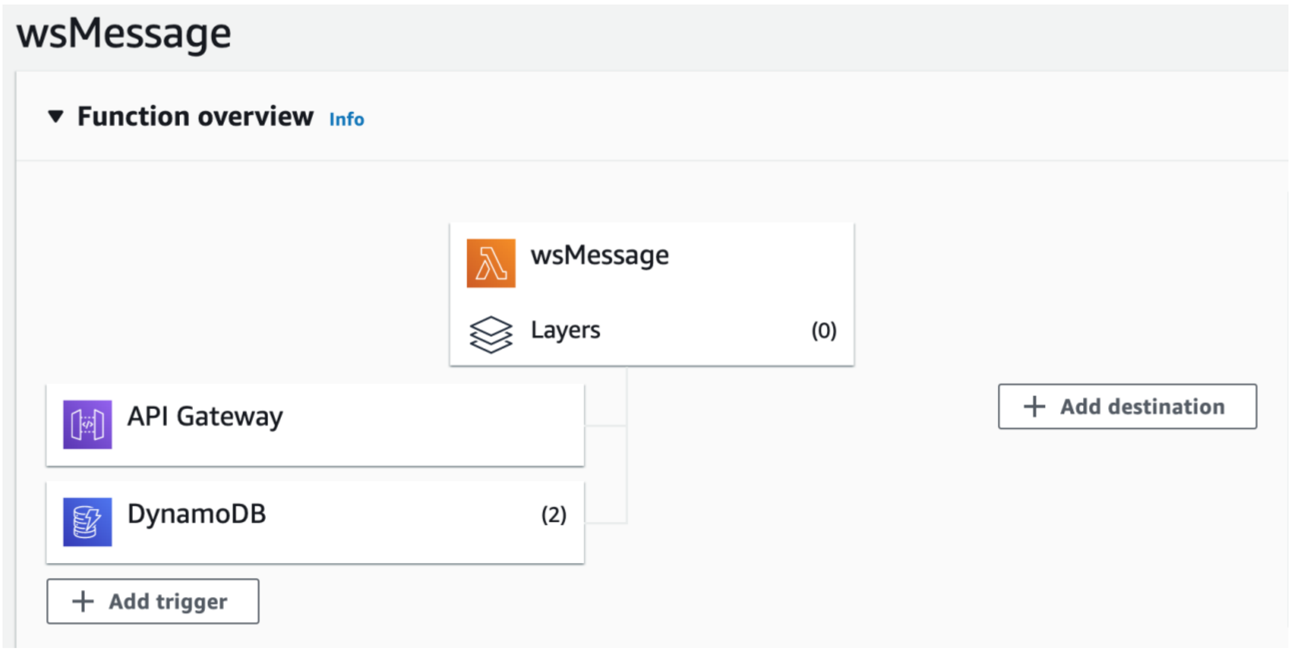
Items returned (2)

| ConnectionId |
|---|
| QxRUDfSYIAMCJBw= |
| QxRVXcAoIAMCLAg= |

WebSocketClients: stores the connection id of clients

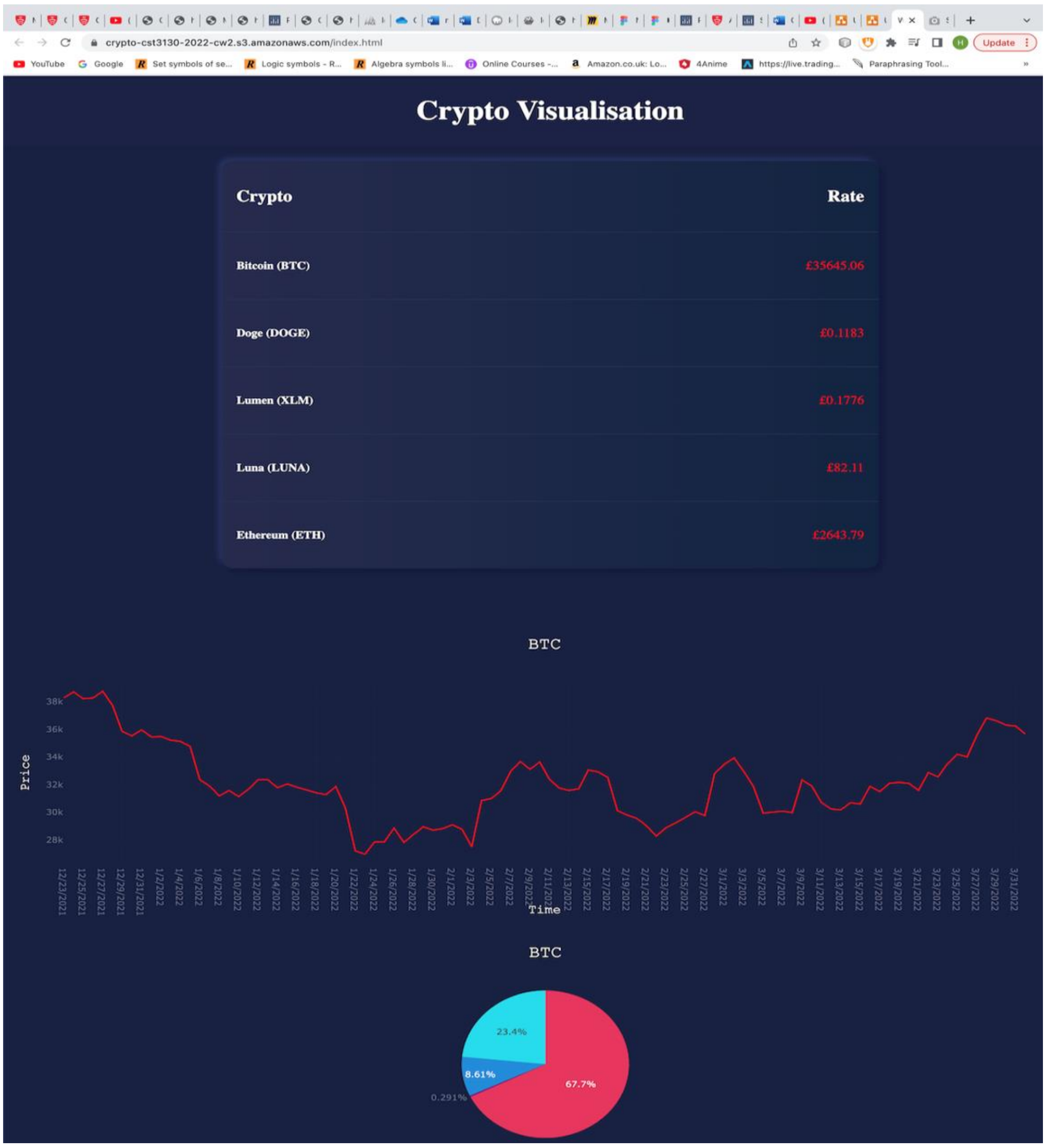
Lambda functions sends data to clients through API Gateway by using stored connection IDs from the DynamoDB table above.

Within my application there is a lambda function to connect new clients and another function to handle messages sent between clients. This lambda function has 2 triggers which determines whether to send data to one client or broadcast to all clients.



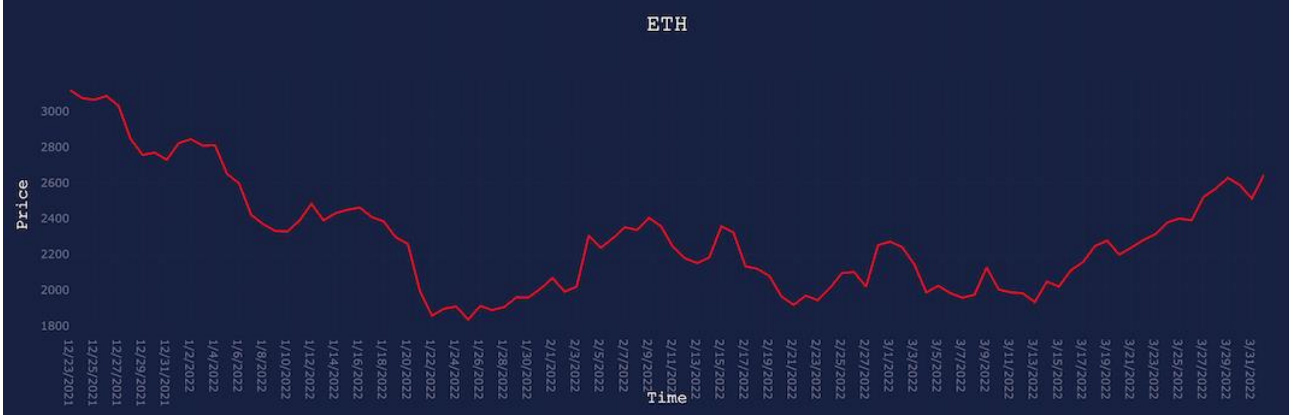
If it is an API gateway trigger it calls a function within the lambda which sends data to a single client however if the trigger is a DynamoDB trigger (numerical and sentiment DynamoDB table) it calls a function to broadcast the data to all clients. I have done this by using "if statement" to check the type of event the lambda function received whether it's an API gateway or DynamoDB. There is also another lambda function which is called when the clients disconnect.

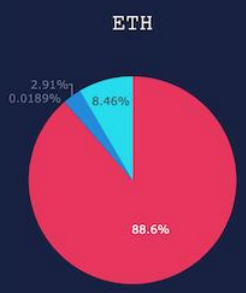
## 6. Screenshot(s) of the front end of website and the data visualization.

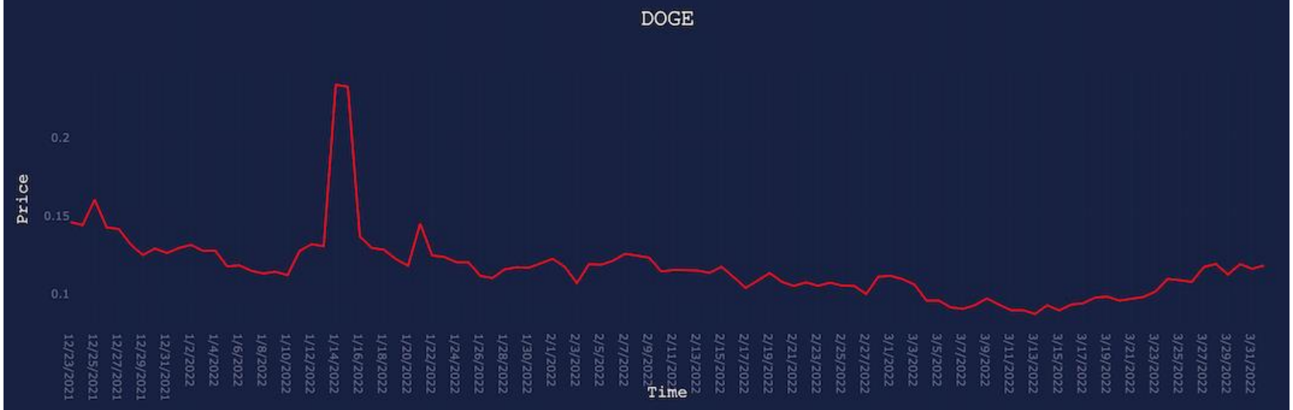


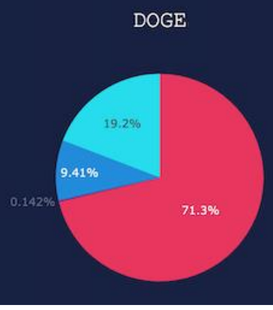
**Crypto Visualisation**

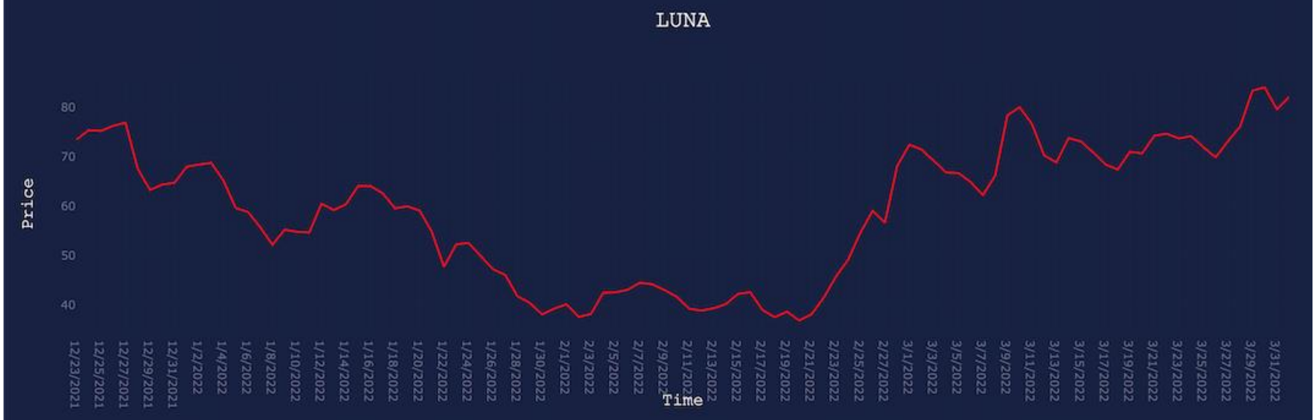
| Crypto | Rate |
|---|---|
| Bitcoin (BTC) | £35645.06 |
| Doge (DOGE) | £0.1183 |
| Lumen (XLM) | £0.1776 |
| Luna (LUNA) | £82.11 |
| Ethereum (ETH) | £2643.79 |

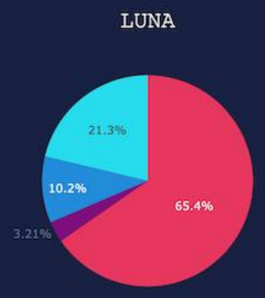

ETH
Price
3000
2800
2600
2400
2200
2000
1800
Time
12/23/2021
3/31/2022

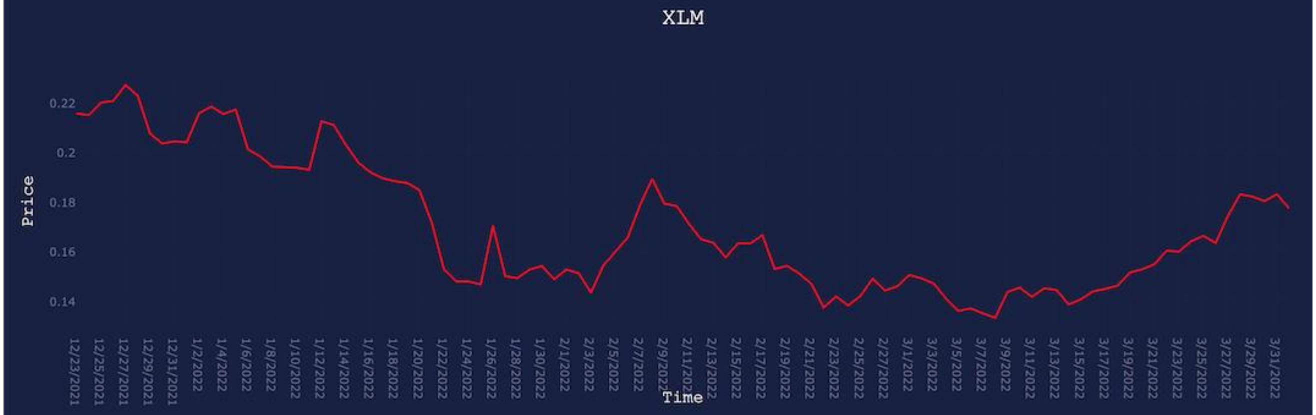

ETH
2.91%
0.0189%
8.46%
88.6%

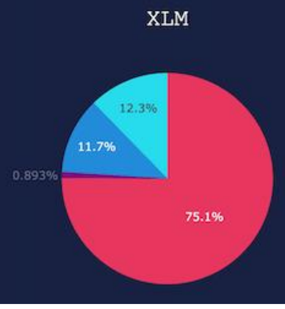

DOGE
Price
0.2
0.15
0.1
Time
12/23/2021
3/31/2022

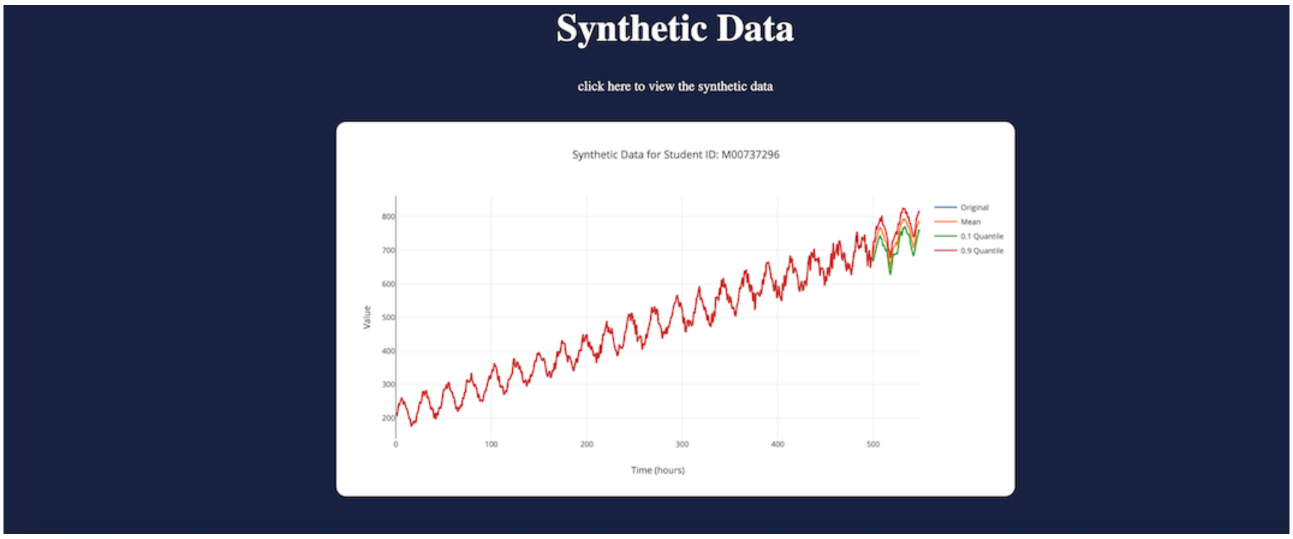

DOGE
19.2%
9.41%
0.142%
71.3%


LUNA
Price
80
70
60
50
40
Time
12/23/2021
12/25/2021
12/27/2021
12/29/2021
12/31/2021
1/2/2022
1/4/2022
1/6/2022
1/8/2022
1/10/2022
1/12/2022
1/14/2022
1/16/2022
1/18/2022
1/20/2022
1/22/2022
1/24/2022
1/26/2022
1/28/2022
1/30/2022
2/1/2022
2/3/2022
2/5/2022
2/7/2022
2/9/2022
2/11/2022
2/13/2022
2/15/2022
2/17/2022
2/19/2022
2/21/2022
2/23/2022
2/25/2022
2/27/2022
3/1/2022
3/3/2022
3/5/2022
3/7/2022
3/9/2022
3/11/2022
3/13/2022
3/15/2022
3/17/2022
3/19/2022
3/21/2022
3/23/2022
3/25/2022
3/27/2022
3/29/2022
3/31/2022


LUNA
21.3%
10.2%
3.21%
65.4%


XLM
Price
0.22
0.2
0.18
0.16
0.14
Time
12/23/2021
12/25/2021
12/27/2021
12/29/2021
12/31/2021
1/2/2022
1/4/2022
1/6/2022
1/8/2022
1/10/2022
1/12/2022
1/14/2022
1/16/2022
1/18/2022
1/20/2022
1/22/2022
1/24/2022
1/26/2022
1/28/2022
1/30/2022
2/1/2022
2/3/2022
2/5/2022
2/7/2022
2/9/2022
2/11/2022
2/13/2022
2/15/2022
2/17/2022
2/19/2022
2/21/2022
2/23/2022
2/25/2022
2/27/2022
3/1/2022
3/3/2022
3/5/2022
3/7/2022
3/9/2022
3/11/2022
3/13/2022
3/15/2022
3/17/2022
3/19/2022
3/21/2022
3/23/2022
3/25/2022
3/27/2022
3/29/2022
3/31/2022

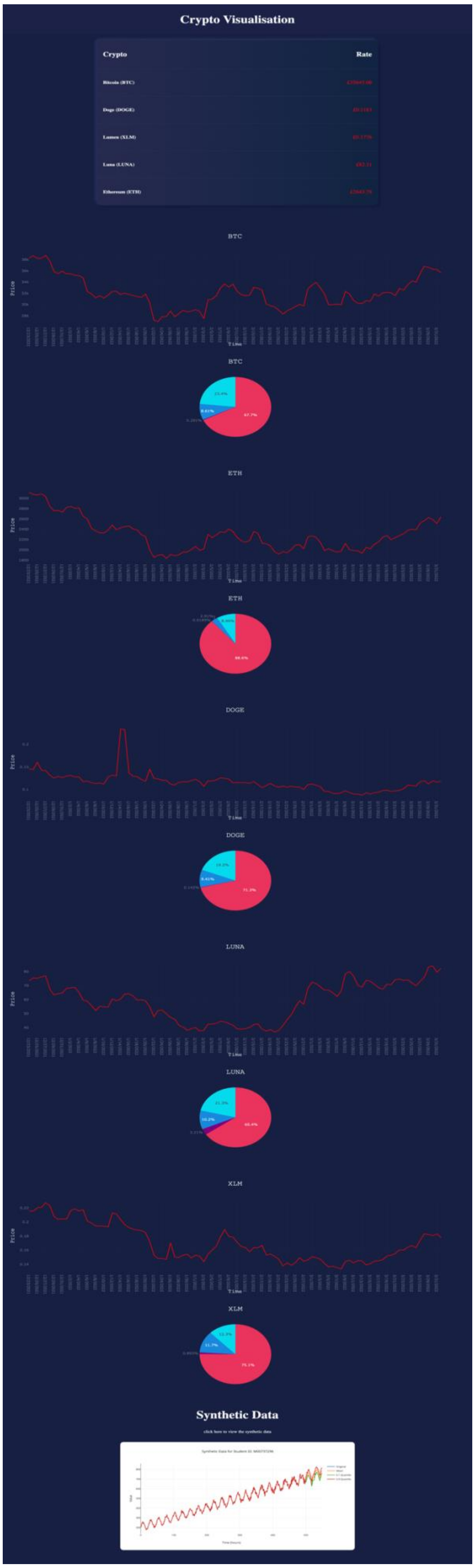

XLM
12.3%
11.7%
0.893%
75.1%



Synthetic Data

click here to view the synthetic data

Live Site : [https://crypto-cst3130-2022-cw2.s3.amazonaws.com/index.html](https://crypto-cst3130-2022-cw2.s3.amazonaws.com/index.html) full page:-



Crypto Visualisation

| Crypto | Rate |
|---|---|
| Bitcoin (BTC) | [illegible] |
| Doge (DOGE) | [illegible] |
| Lumen (XLM) | [illegible] |
| Luna (LUNA) | £82.11 |
| Ethereum (ETH) | [illegible] |

Synthetic Data

click here to view the synthetic data