

# Bitcoin decision helper

SKDUE



# Team Members



Natchanon  
Manatphaiboon



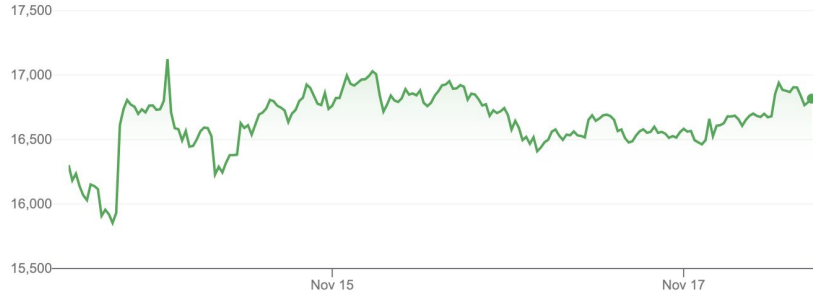
Fredric Jan Michael  
JÖRNELIUS



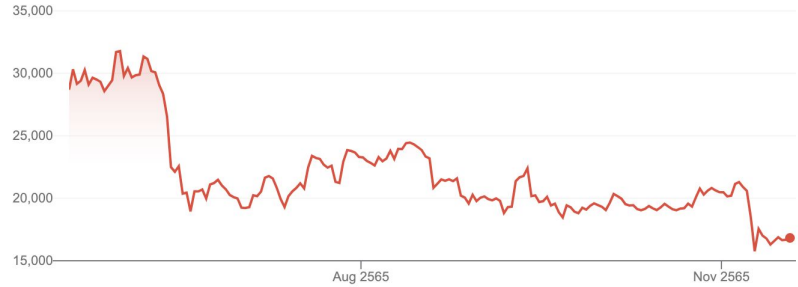
Patkamon  
Awaiwanont

# Purpose

To help users make decisions in today's Bitcoin market.  
This is done by providing them with useful information,  
such as the one below.



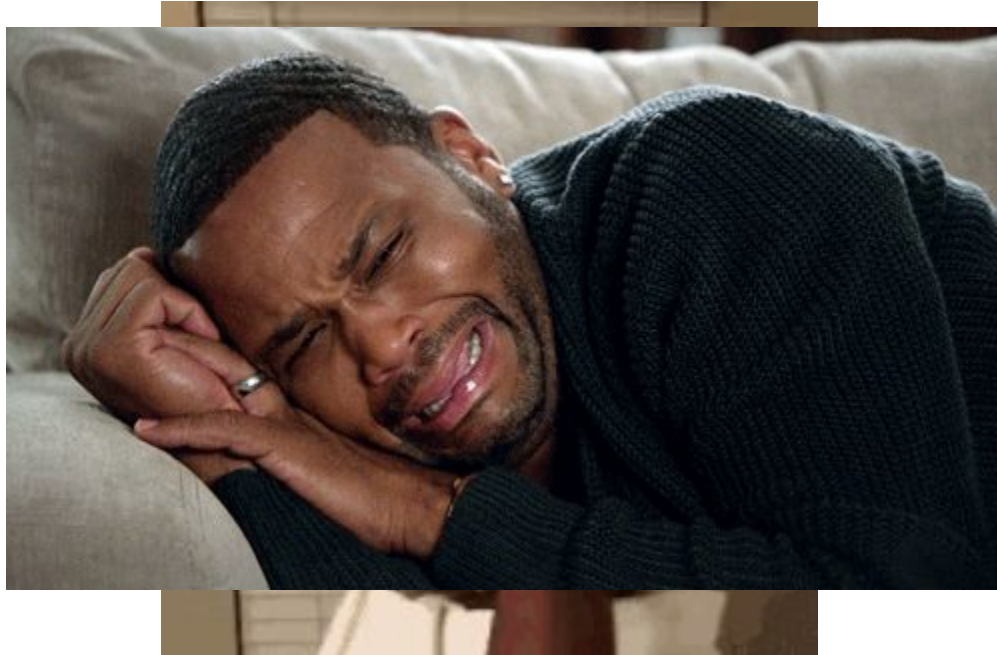
5 days graph



6 month graph

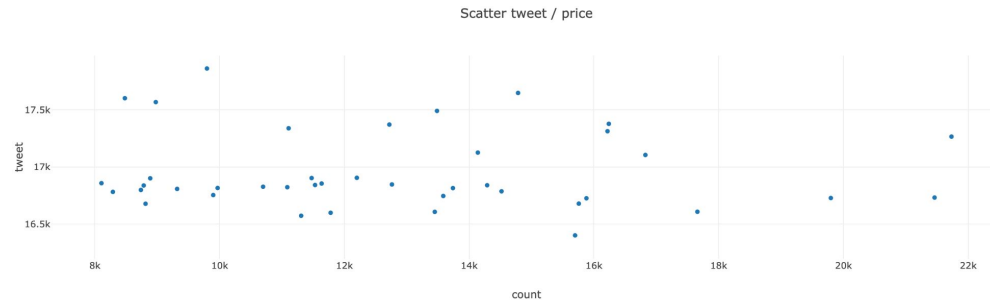
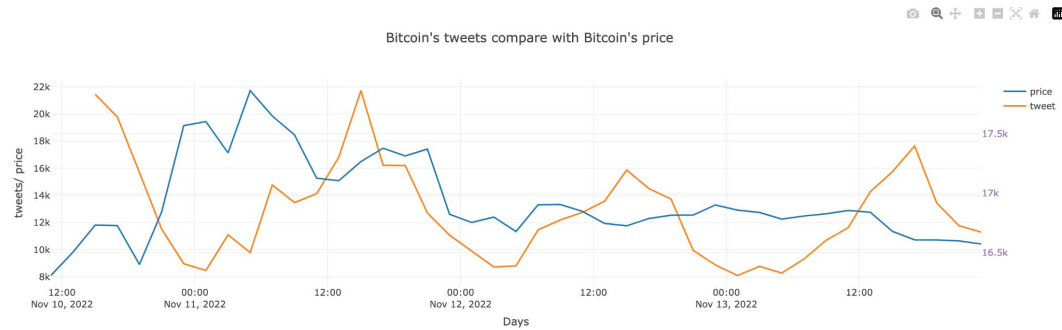
# Purpose

It might be life changing.

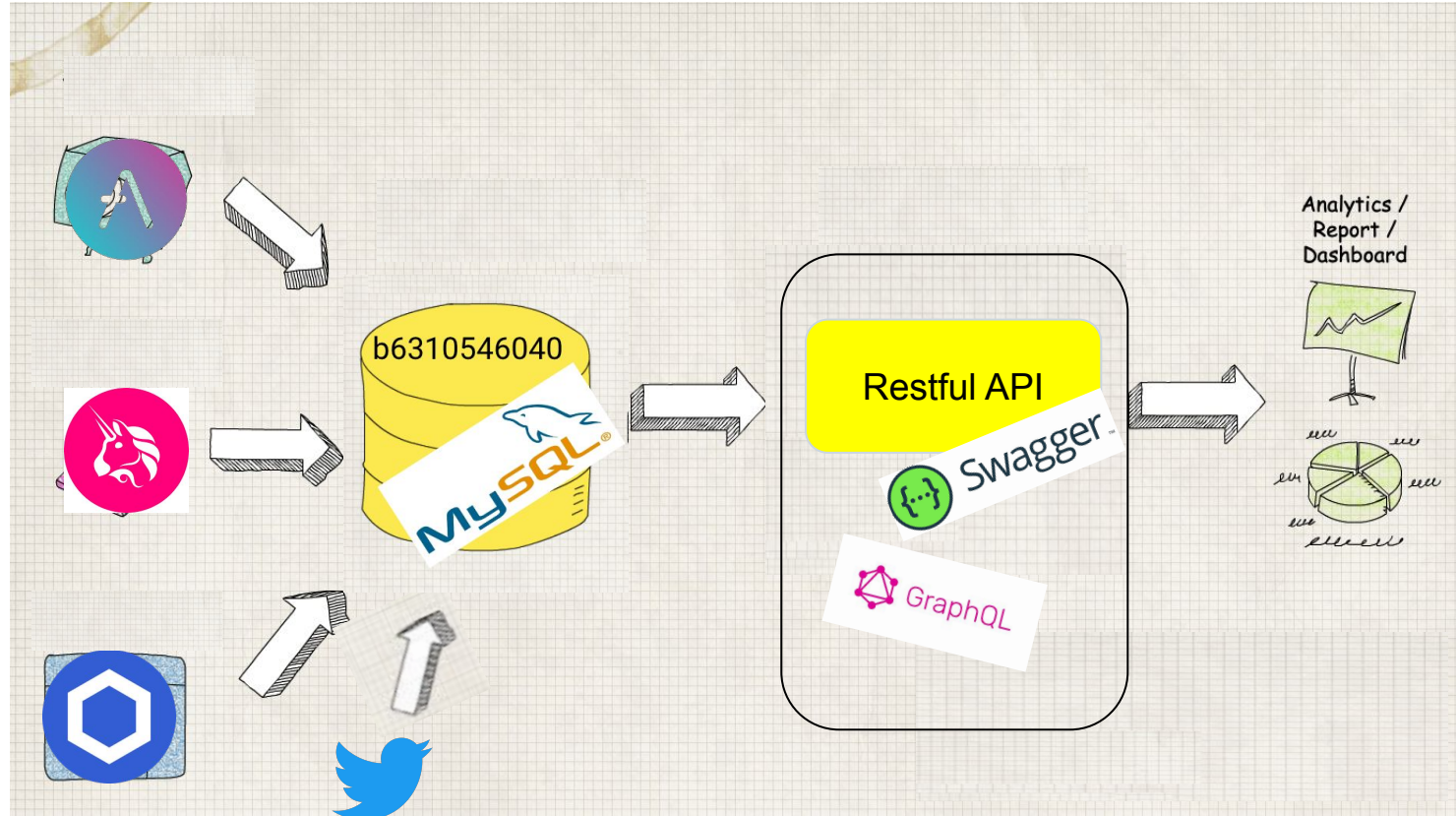


# Purpose

We offer different types of values and visual measurements.



# Overall Architecture



# API USED



Twitter Api



Aave



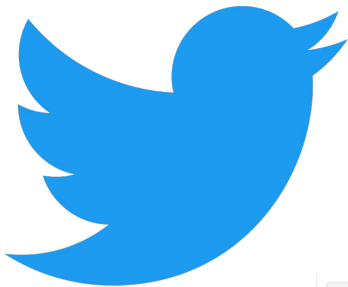
Uniswap



Chainlink



The graph



# Twitter Api

## used to count tweets mention bitcoin

GET <https://api.twitter.com/2/tweets/search/recent?query=Bitcoin> [Send](#)

Params Authorization Headers (8) Body Pre-request Script Tests Settings [Cookies](#)

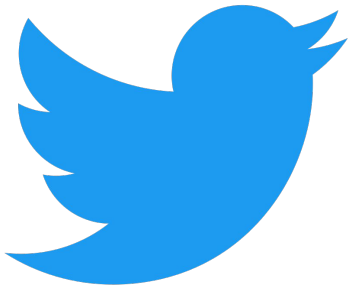
KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> query	Bitcoin	Required. Query for matching Tweets. Up to 512 char...		

Body [Cookies \(4\)](#) [Headers \(20\)](#) [Test Results](#) [Status: 200 OK](#) [Time: 413 ms](#) [Size: 1.77 KB](#) [Save Response](#)

[Pretty](#) [Raw](#) [Preview](#) [Visualize](#) [JSON](#) [...](#)

```
1 {
2   "data": [
3     {
4       "edit_history_tweet_ids": [
5         "1593482419105153025"
6       ],
7       "id": "1593482419105153025",
8       "text": "RT @Nebraskangooner: $100 #Bitcoin giveaway! \n(Day 4 of 14)\n\nAll you have to do is retweet and comment.
          \n\nI'll pick a random winner later to..."
9     },
10    {
11      "edit_history_tweet_ids": [
12        "1593482415099432961"
13      ],
14      "id": "1593482415099432961",
15      "text": "Ep07- Moneyland - Companion Guide For BBC's \"The Missing Cryptoqueen\" Podcast - https://t.co/WiYVPzxpDf \n#bitcoin #cryptocurrency #crypto #nft #news\nhttps://t.co/XminntDEmr &lt;----JOIN https://t.co/n14tbQuRv0"
16    },
17    {
18      "edit_history_tweet_ids": [
```





# Twitter Api

used to count tweets mention bitcoin

```
def sort(list):  
    ans = []  
    total = 0  
    for i in range(len(list)):  
        if i%2==1:  
            total += tweet[i]['tweet_count']  
        else:  
            ans.append([total,  
                        datetime.strptime(tweet[i-1]['start'], '%Y-%m-%dT%H:%M:%S.%f%z')  
                        ])  
            total=0  
    return ans
```

for every hour => for every 2 hours

# The Graph



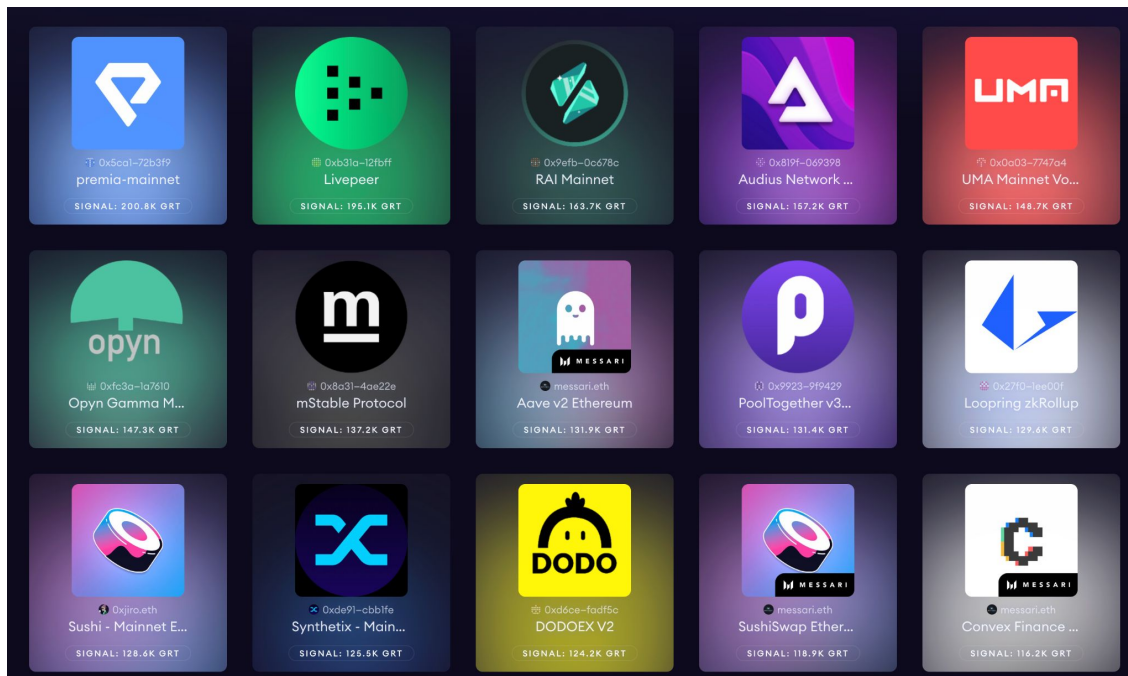
The graph





# The Graph

place for web3 app to put their api in from of graph query  
for everyone to query.

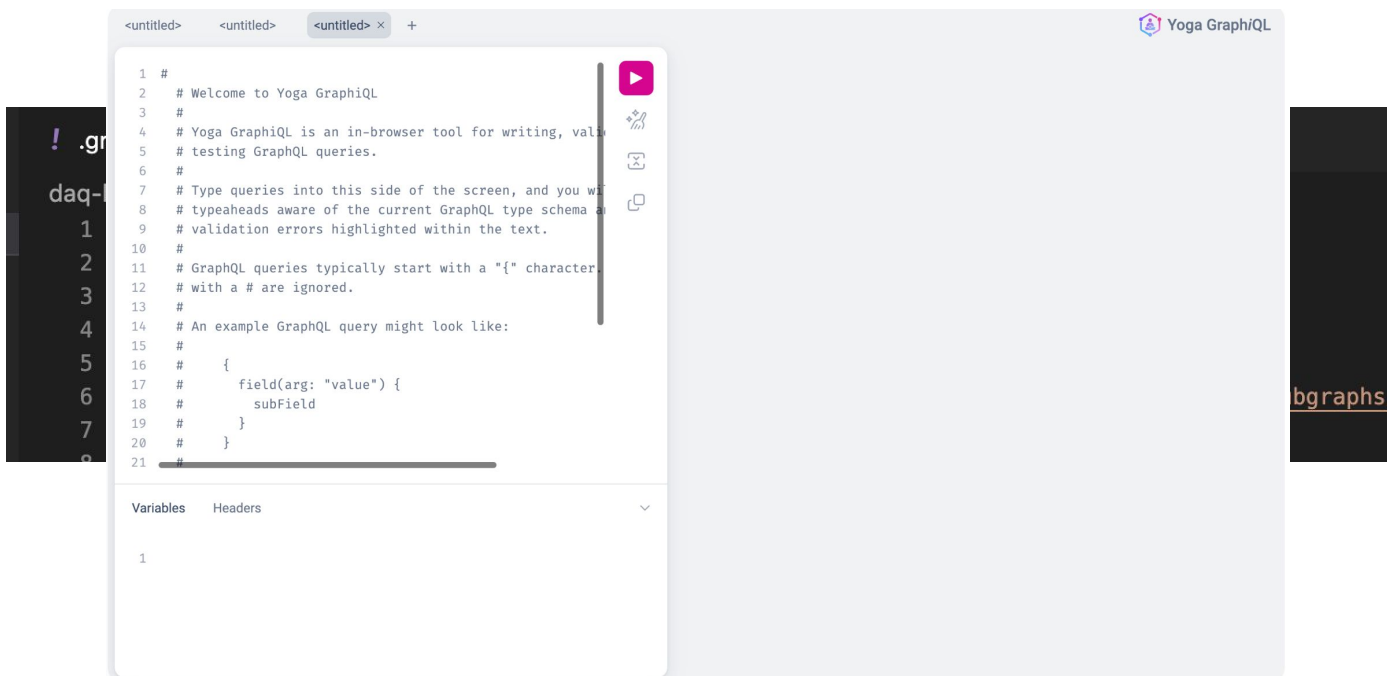




# The Graph

To use the graph

```graphclient serve-dev```






# Aave graph API

Web3 app for lending & borrowing crypto

```
{  
  marketHourlySnapshots( orderBy:  
    timestamp, orderDirection: desc,  
    where: {timestamp_lt: %s ,market:  
      "0x2260fac5e5542a773aa44fbcfedf  
7c193bc2c599",timestamp_gt:1668  
038400}){  
    timestamp  
    totalValueLockedUSD  
    totalBorrowBalanceUSD  
  }  
}
```



```
{  
  "data": {  
    "marketHourlySnapshots": [  
      {  
        "timestamp": "1668750671",  
        "totalValueLockedUSD":  
        "599536862.8041679952221114381720725",  
        "totalBorrowBalanceUSD":  
        "61419620.00536499310015566442368177"  
      },  
      {  
        "timestamp": "1668747551",  
        "totalValueLockedUSD":  
        "599499454.8955163840043374978784085",  
        "totalBorrowBalanceUSD":  
        "61419541.3499048780712549934339718"  
      },  
      {  
        "timestamp": "1668743843",  
        "totalValueLockedUSD":  
        "599663137.2130029757519506830584",  
        "totalBorrowBalanceUSD":  
        "61419447.88652168704356484316745428"  
      },  
      {  
        "timestamp": "1668740135",  
        "totalValueLockedUSD":  
        "599663137.2130029757519506830584",  
        "totalBorrowBalanceUSD":  
        "61419447.88652168704356484316745428"  
      }  
    ]  
  }  
}
```



# Aave graph API

```
get_borrow_lend.py / ...
import datetime
import requests
import json
import pandas as pd

testlimit = "1668038400"
timenow = 166803827
query = """
{
  marketHourlySnapshots( orderBy: timestamp, orderDirection: desc, where: {timestamp
    timestamp
    totalValueLockedUSD
    totalBorrowBalanceUSD
  }
) {
  """ % (timenow)
print(datetime.datetime.fromtimestamp(int(timenow)))
url = 'http://localhost:4000/graphql'
r = requests.post(url, json={'query': query})
json_data = json.loads(r.text)
df_data = json_data['data']['marketHourlySnapshots']
df = pd.DataFrame(df_data)
print(datetime.datetime.fromtimestamp(int(df_data[-1]['timestamp'])))
print(df_data)
```

get\_borrow\_lend.py  
for query data

```
17
18 def insert_data(data):
19     sql = "INSERT INTO aave (lend, borrow, time) VALUES (%s, %s, %s)"
20     mycursor.execute(sql, data)
21     mydb.commit()
22
23
24 def sort(list):
25     ans = []
26     starttime = 1668672000
27     endtime = starttime - 7200
28     totalL = 0
29     totalB = 0
30     for i in range(len(list)):
31         if int(list[i]['timestamp']) > endtime and int(list[i]['timestamp']) <= starttime:
32             totalL += float(list[i]['totalValueLockedUSD'])
33             totalB += float(list[i]['totalBorrowBalanceUSD'])
34         else:
35             ans.append([totalL, totalB,
36                         datetime.datetime.fromtimestamp(starttime)
37                         ])
38             starttime = endtime
39             endtime = starttime - 7200
40             totalL = 0
41             totalB = 0
42     return ans
43
44 sort_aave = sort(aave)
45
46 for i in range(2,87):
47     data = (sort_aave[i][0], sort_aave[i][1], sort_aave[i][2])
48     insert_data(data)
49
```


sort\_aave.py  
for write data on sql



# Uniswap graph API

decentralize app for exchange crypto

```
{
  liquidityPools(
    where: {id: "0xcbcdf9626bc03e24f779434178a73a0b4bad62ed"}) {
    inputTokens(first: 2) {
      symbol
    }
    totalValueLockedUSD,
    deposits(where: {timestamp_lt:
%s, timestamp_gt: %s
}, orderBy: timestamp, orderDirection: desc)
    {
      amountUSD
      timestamp
    }
  }
}
```



```
{
  "data": {
    "liquidityPools": [
      {
        "inputTokens": [
          {
            "symbol": "WBTC"
          },
          {
            "symbol": "WETH"
          }
        ],
        "totalValueLockedUSD": "168093376.0770558983221178623891457",
        "deposits": [
          {
            "amountUSD": "2.002721796905720444745097417379009",
            "timestamp": "1668749471"
          },
          {
            "amountUSD": "0.1680970209670822667142730528502963",
            "timestamp": "1668748655"
          },
          {
            "amountUSD": "2.187317791841514859853766844350675",
            "timestamp": "1668747671"
          }
        ]
      }
    ]
  }
}
```



# Uniswap graph API

```
get_withdraw_deposit.py > ...
1 import datetime
2 import requests
3 import json
4 import pandas as pd
5
6 testlimit = "1668038400"
7 timenow = "1668125651"
8 query = """
9     {
10       liquidityPools( where:{id:"0xc000"}
11       inputTokens(first:2){
12         symbol
13       }
14       totalValueLockedUSD,
15       deposits(where:{timestamp_lt: %s, timestamp_gt:%s },orderBy:timestamp
16         amountUSD
17         timestamp
18       )
19     }
20     """ % (timenow, testlimit)
21 print(datetime.datetime.fromtimestamp(int(timenow)))
22 url = 'http://localhost:4000/graphql'
23 r = requests.post(url, json={'query': query})
24 json_data = json.loads(r.text)
25 df_data = json_data['data']['liquidityPools'][0]['deposits']
26 df = pd.DataFrame(df_data)
27 print(datetime.datetime.fromtimestamp(int(df_data[-1]['timestamp'])))
28 print(r.text)
29 print(df_data[-1]['timestamp'] <= testlimit)
30
```

get\_withdraw\_deposit.py  
for query data

```

9
10 def insert_data(data):
11     sql = "INSERT INTO unibtc (withdraw, deposit, time) VALUES (%s, %s, %s)"
12     mycursor.execute(sql, data)
13     mydb.commit()
14
15
16 def sort(list):
17     ans = []
18     starttime = 1668672000
19     endtime = starttime - 7200
20     total = 0
21     for i in range(len(list)):
22         if int(list[i]['timestamp']) > endtime and int(list[i]['timestamp'])
23             total += float(list[i]['amountUSD'])
24         else:
25             ans.append([total,
26                         datetime.datetime.fromtimestamp(starttime)
27                         ])
28             starttime = endtime
29             endtime = starttime - 7200
30             total = 0
31     return ans
32
33
34 sort_with = sort(withdraw7)
35 sort_deposit = sort(deposit7)
36
37 for i in range(2,87):
38     data = (sort_with[i][0], sort_deposit[i][0], sort_deposit[i][1])
39     insert_data(data)
40
```

sort\_uni.py  
for write data on sql






# Chainlink graph API

web3 app that keep track of crypto price

```
{  
  priceFeeds(where:{tokenSymbol:"BTC", timestamp_gt :  
    1668038400},orderBy:timestamp,orderDirection:desc){  
    tokenPrice    timestamp    } } }
```



```
{  
  "data": {  
    "priceFeeds": [  
      {  
        "tokenPrice": "1679424452100",  
        "timestamp": "1668750923"  
      },  
      {  
        "tokenPrice": "1680628000000",  
        "timestamp": "1668747347"  
      },  
      {  
        "tokenPrice": "1677452000000",  
        "timestamp": "1668745787"  
      },  
      {  
        "tokenPrice": "1685992002168",  
        "timestamp": "1668743747"  
      },  
      {  
        "tokenPrice": "1691228817628",  
        "timestamp": "1668740147"  
      },  
      {  
        "tokenPrice": "1693938167873",
```



# Chainlink graph API

```
get-price.py 2, U X
get-price.py > ...
1 import datetime
2 import requests
3 import json
4 import pandas as pd
5
6 testlimit = "1668038400"
7 timenow = 1668067247
8 query = """
9     {
10         priceFeeds(where:{tokenSymbol:"BTC",timestamp_lt : %s, timestamp_gt : 1668038400
11             tokenPrice
12             timestamp
13         }
14     }
15     """ % (timenow)
16 print(datetime.datetime.fromtimestamp(int(timenow)))
17 url = 'http://localhost:4000/graphql'
18 r = requests.post(url, json={'query': query})
19 json_data = json.loads(r.text)
20 df_data = json_data['data']['priceFeeds']
21 df = pd.DataFrame(df_data)
22 print(datetime.datetime.fromtimestamp(int(df_data[-1]['timestamp'])))
23 print(df_data)
24
```

get\_price.py  
for query data


```
def sort(list):
    ans = []
    starttime = 1668672000
    endtime = starttime - 7200
    total = 0
    c=0
    for i in range(len(list)):
        if int(list[i]['timestamp']) > endtime and int(list[i]['timestamp']) < starttime:
            total += float(list[i]['tokenPrice'])
            c+=1
        else:
            ans.append([int(total/c/10**8),
                        datetime.datetime.fromtimestamp(starttime)])
            starttime = endtime
            endtime = starttime - 7200
            total = 0
            c = 0
    return ans

sort_price = sort(price)
for i in range(2,87):
    data = (sort_price[i][0],sort_price[i][1])
    insert_data(data)
```


sort\_price.py  
for write data on sql

# DATABASE SCHEMA

Twitter


#	Name	Type
1	<b>id</b> 	int(11)
2	<b>count</b>	int(11)
3	<b>time</b>	timestamp

aave


Name	Type
<b>id</b> 	int(11)
<b>lend</b>	float
<b>borrow</b>	float
<b>time</b>	timestamp

# DATABASE SCHEMA

Chainlink

Name	Type
id 	int(11)
price	int(11)
time	timestamp

uniswap

Name	Type	C
id 	int(11)	
withdraw	float	
deposit	float	
time	timestamp	

**Live Demo for  
APIs, Main  
Features, and  
Visualizations.**

**ENOUGH SLIDES**

