Big Data Processing
ECS765P

220058151
Benita Ashley

Semester 1
(January 2022 Intake)

MSc Big Data Science with Machine Learning Systems

Coursework:
Analysis of Ethereum Transactions and Smart Contracts

PART A : TIME ANALYSIS (20%)

JOB 1: TOTAL NUMBER OF TRANSACTIONS

PROBLEM STATEMENT: Graphical representation in the form of bar plots of total number of transactions in each month between the start and end of dataset.

DATASET: Transactions

SOLUTION:

- The maper maps the timestamp from field 6 and segregates the month and year from it and yields the month, year and field 1 which is the from_address.
- The reducer counts the total number of transactions per month and yields that.

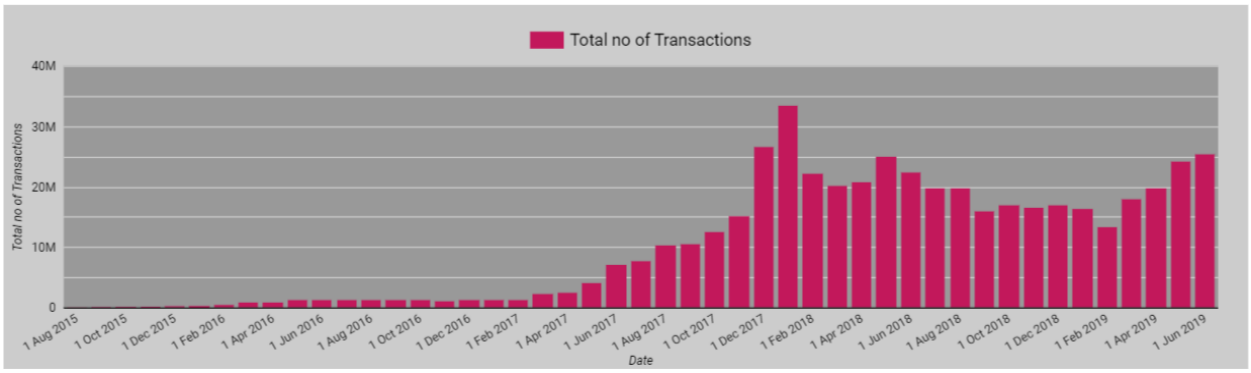
JOB ID:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1637317090236_22726/
Running job: job_1637317090236_22726

SAMPLE OUTPUT:

["01", "17"]	1409664
["01", "19"]	16569597
["02", "16"]	520040
["02", "18"]	22231978
["03", "17"]	2426471
["03", "19"]	18029582
["04", "16"]	1023096
["04", "18"]	20876642
["05", "17"]	4245516
["05", "19"]	24332475
["06", "16"]	1351536
["06", "18"]	22471788

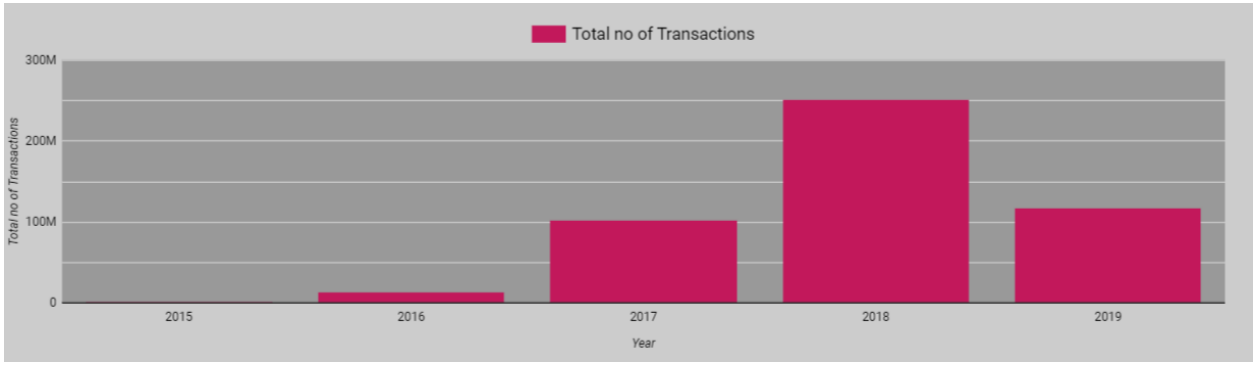
PLOT: Here, Google studio is used to plot the graphs and analyze the data obtained from the graphs. The graphs are further simplified into year wise and month-year wise bar plots to see the trends over the years and during the particular year. The values are sorted in the ascending order of months.

1) Total number of transactions in each month between the start and end of the dataset.



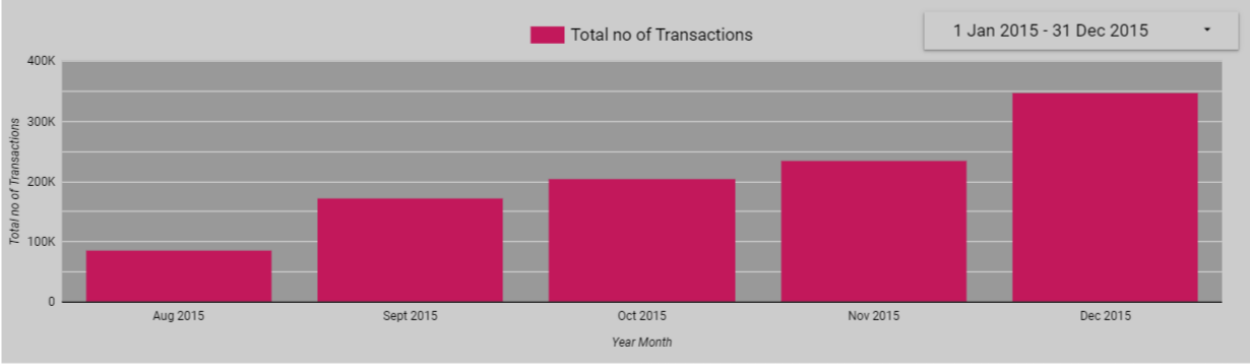
Observation: February 2016 has the highest number of transactions whereas August 2015 has the lowest number of transactions.

2) **Total number of transactions (Year wise)**

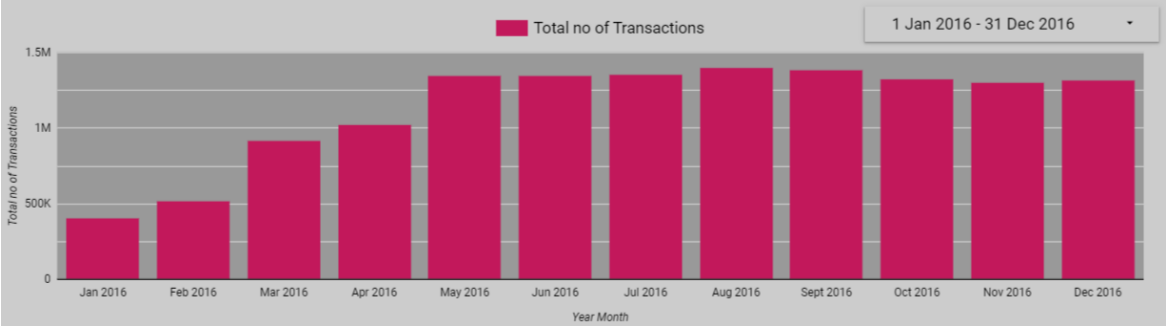


Observation: Year 2018 has the highest number of transactions whereas 2015 has the lowest.

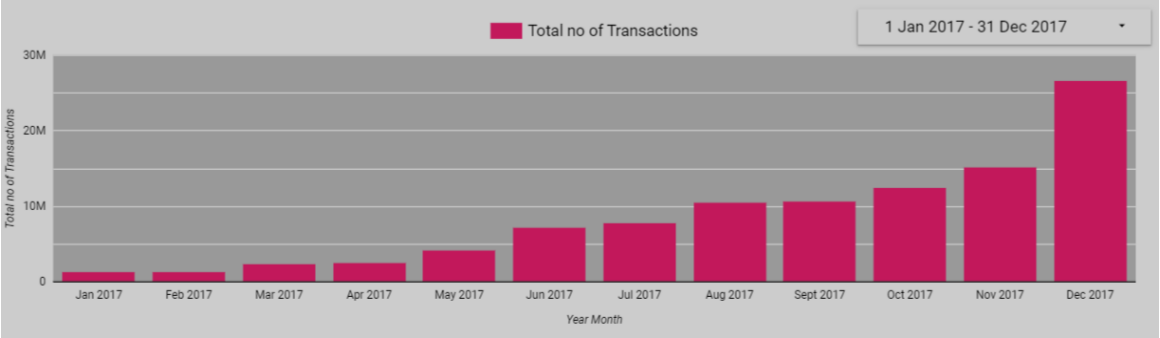
3) **Total number of transactions for the year 2015**



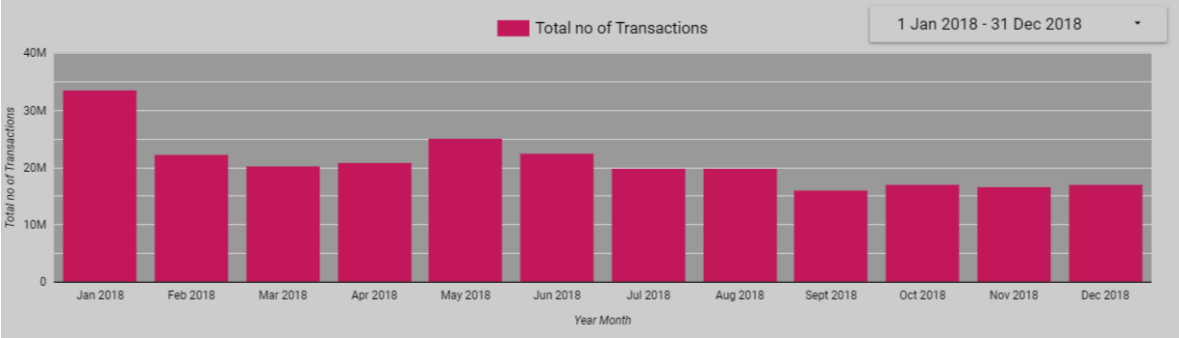
4) Total number of transactions for the year 2016



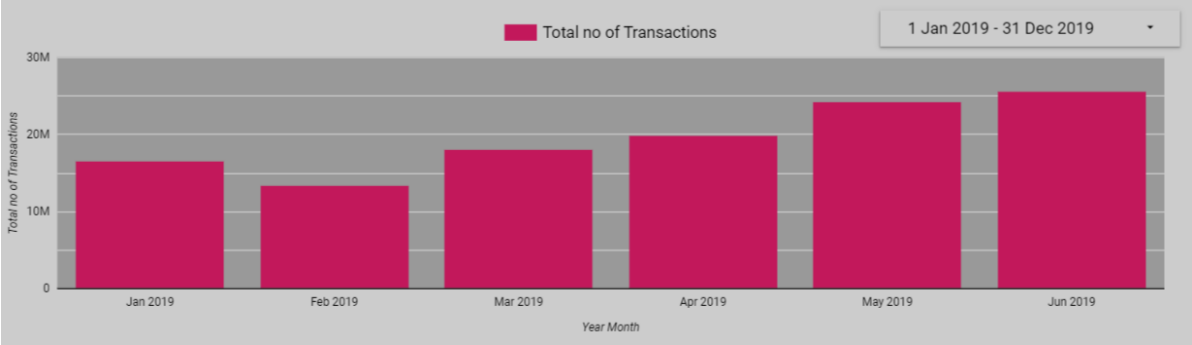
5) Total number of transactions for the year of 2017



6) Total number of transactions for the year of 2018



7) **Total number of transactions for the year of 2019**



JOB 2: AVERAGE OF TRANSACTIONS

PROBLEM STATEMENT: Graphical plot of average value of transactions in each month between the start and the end of the dataset.

DATASET: Transactions

SOLUTION:

- The mapper gets the timestamp i.e field 6 of the dataset from which the month and year are calculated along with the gas price i.e field 5. The months, year and field 1 are yielded.
- Reducer counts the total number of transactions per month.
- A combiner is used to count the number of transactions and the sum which is yielded by the reducer after the average was calculated.

JOB ID:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1637317090236_22968/

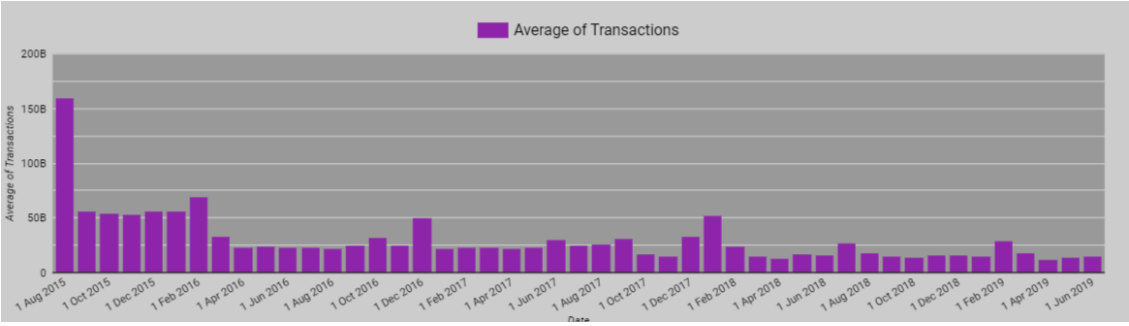
SAMPLE OUTPUT:

["01", "17"]	22507570808
["01", "19"]	14611816446
["02", "16"]	69180681134
["02", "18"]	23636574204
["03", "17"]	23232253601
["03", "19"]	18083035520
["04", "16"]	23361180503

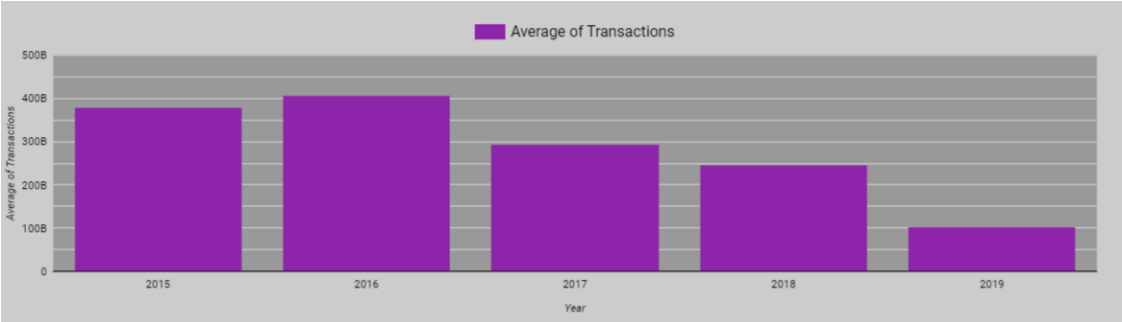
["04", "18"]	13153739248
["05", "17"]	23572314972
["05", "19"]	14479858768
["06", "16"]	23021251390

PLOTS: Here, Google studio is used to plot the graphs and analyze the data obtained from the graphs. The graphs are further simplified into year wise and month-year wise bar plots to see the trends over the years and during the particular year.

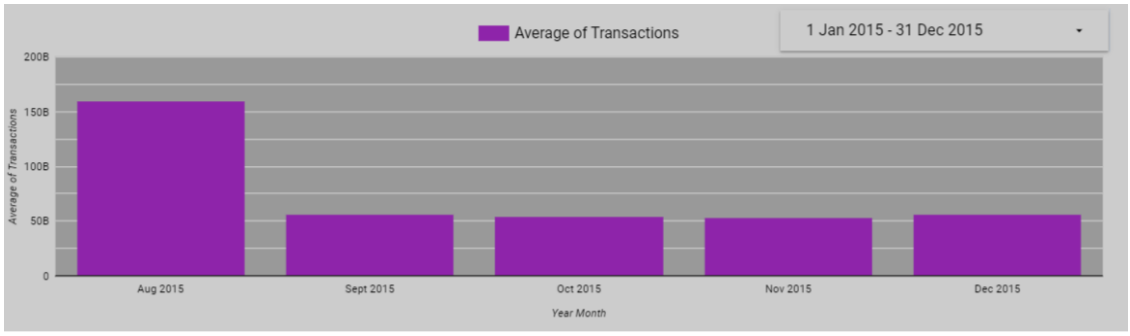
1) **Average of Transactions between the start and end of the dataset**



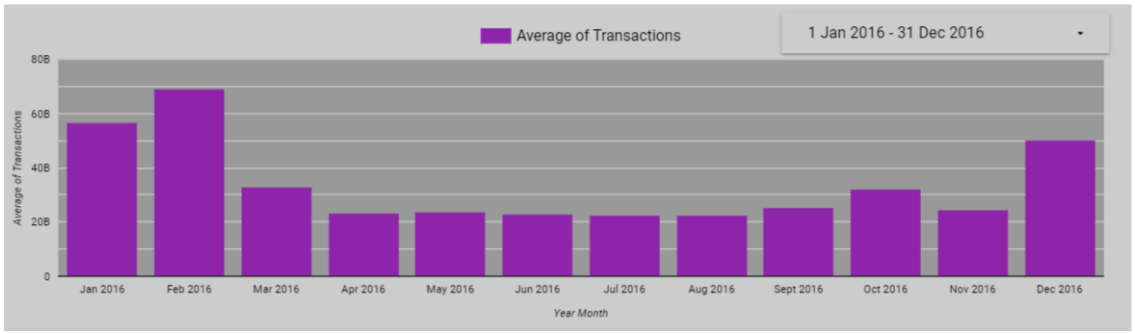
2) **Average of Transactions - Year Wise distribution**



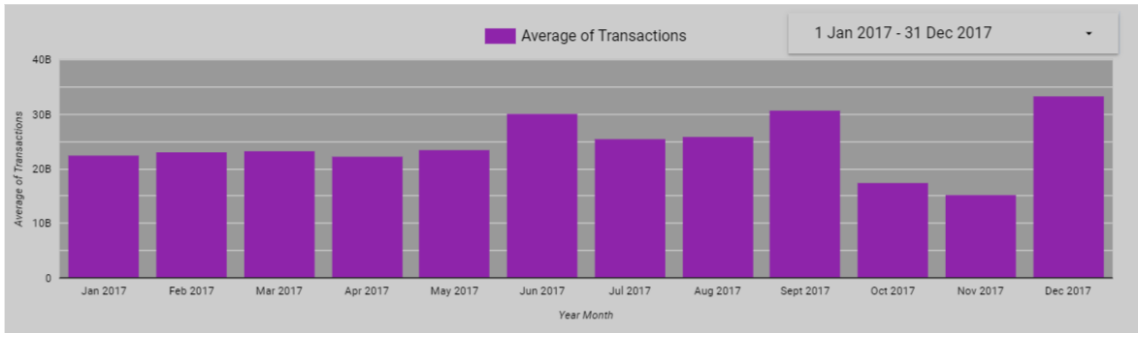
3) **Average of Transactions for the year 2015**



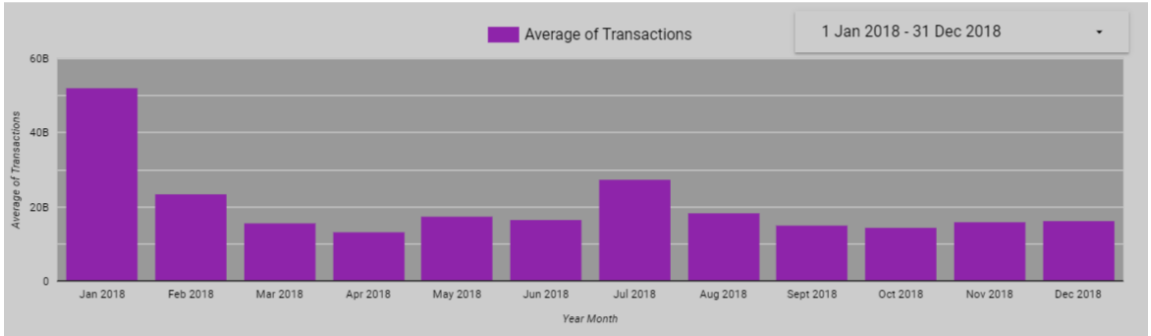
4) Average of Transactions for the year 2016



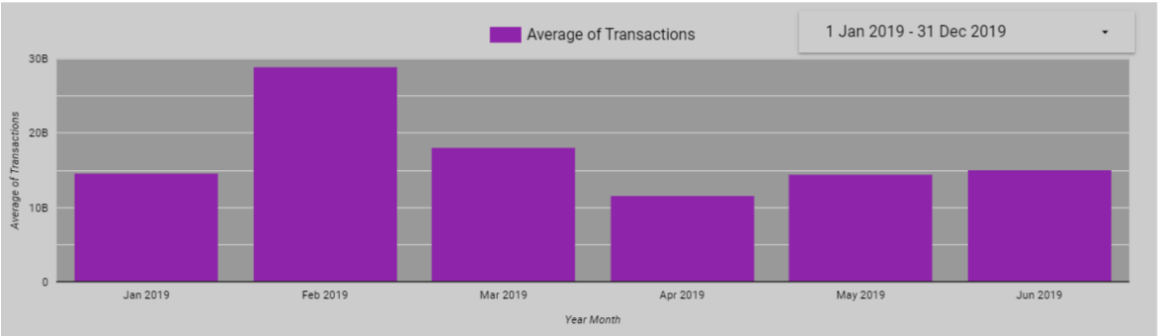
5) Average of Transactions for the year of 2017



6) Average of Transactions for the year of 2018



7) Average of Transactions for the year 2019



PART B: TOP TEN MOST POPULAR SERVICES (25%)

PROBLEM STATEMENT: Evaluate the top 10 miners by the size of blocks mined.

DATASET: Transactions and Contracts

SOLUTION:

- The first mapper takes the value of the addresses from the to_address field from both the Transactions and Contracts dataset as the key.
- The reducer takes the values from the mapper and checks if the address for a given aggregate is not present within the contacts is not considered as a smart contact and hence should be filtered out. Then the sum of all values and keys is yielded.
- In the second mapper the values yielded from the first reducer are taken and the keys and values are combined. The second reducer finds the top 10 contracts by taking 'None' as key and sorts the values in the descending order.

JOB ID:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1637317090236_23848/
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1637317090236_23862/

OUTPUT:

0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444	84155100809965865822726776
0xfa52274dd61e1643d2205169732f29114bc240b3	45787484483189352986478805
0x7727e5113d1d161373623e5f49fd568b4f543a9e	45620624001350712557268573
0x209c4784ab1e8183cf58ca33cb740efbf3fc18ef	43170356092262468919298969
0x6fc82a5fe25a5cdb58bc74600a40a69c065263f8	27068921582019542499882877

0xbfc39b6f805a9e40e77291aff27aee3c96915bdd	2110419513809366005000000
0xe94b04a0fed112f3664e45adb2b8915693dd5ff3	1556239895680211225 4719409
0xbb9bc244d798123fde783fcc1c72d3bb8c189413	11983608729202893846818681
0xabbb6bebfa05aa13e908eaa492bd7a8343760477	11706457177940895521770404
0x341e790174e3a4d35b65fdc067b6b5634a61caea	8379000751917755624057500

PART C: TOP 10 MOST ACTIVE MINERS (15%)

PROBLEM STATEMENT: To find the top 10 active miners.

DATASET: Blocks

SOLUTION:

- We get the miner addresses from field 2 and size from field 4 from the blocks dataset.
- The first mapper maps all the miners to their respective sizes and then the first reducer uses the address as key and sizes as value.
- The second mapper is run with 'None' key and gets the value as miner addresses with total sizes. Sorting is done in decreasing fashion using lambda and the second field is used as the key to sort.
- The reducer yields the addresses and the size corresponding to each address.

JOB ID:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1637317090236_24056/
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1637317090236_24058/

OUTPUT:

0xea674fdde714fd979de3edf0f56aa9716b898ec8	23989401188.0
0x829bd824b016326a401d083b33d092293333a830	15010222714.0
0x5a0b54d5dc17e0aadc383d2db43b0a0d3e029c4c	13978859941.0
0x52bc44d5378309ee2abf1539bf71de1b7d7be3b5	10998145387.0
0xb2930b35844a230f00e51431acae96fe543a0347	7842595276.0
0x2a65aca4d5fc5b5c859090a6c34d164135398226	3628875680.0
0x4bb96091ee9d802ed039c4d1a5f6216f90f81b01	1221833144.0

0xf3b9d2c81f2b24b0fa0acaaa865b7d9ced5fc2fb	1152472379.0
0x1e9939daaad6924ad004c2560e90804164900341	1080301927.0
0x61c808d82a3ac53231750dadc13c777b59310bd9	692942577.0

PART D: DATA EXPLORATION (40%)

1. FORK THE CHAIN (10%)

PROBLEM STATEMENT: To identify one or more forks in Ethereum and see what effect it has on gas price and general usage.

DATASET: Transactions and Blocks

SOLUTION:

- We have taken two major forks into consideration i.e. the Spurious Dragon in November, 2016 and Byzantium in October 2017.
- Using MapReduce the daily price and number of transactions have been derived for both Spurious Dragon as well as Byzantium.
- The mapper reads the line and splits at ‘,’ after this the time stamp is converted into time. It yields the key which is the date of the month and value which is the gas price if the month is 11 for Spurious Dragon and 10 for Byzantium and the year is 2017 for Spurious Dragon and 2017 for Byzantium.
- Then the aggregated count of transactions and gas price has been yielded by the combiner and reducer respectively.
- The output plot is as shown below for both the forks analyzed and to determine the trend in gas prices and number of transactions and the effect of the fork on the same.

JOB ID:

1. **Spurious Dragon:**

http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_7833/

2. **Byzantium:**

http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_7870/

SAMPLE OUTPUT:

Spurious Dragon:(November 2016)

1	[49025, 20278425376.0]
10	[40164, 20000000000.0]
12	[40882, 22167999998.0]
14	[43876, 20800928824.0]
16	[41684, 25000000000.0]
18	[43785, 40000000000.0]
21	[42735, 21000000000.0]
23	[64289, 23090062212.0]
25	[58075, 20000000000.0]
27	[44449, 31000000000.0]

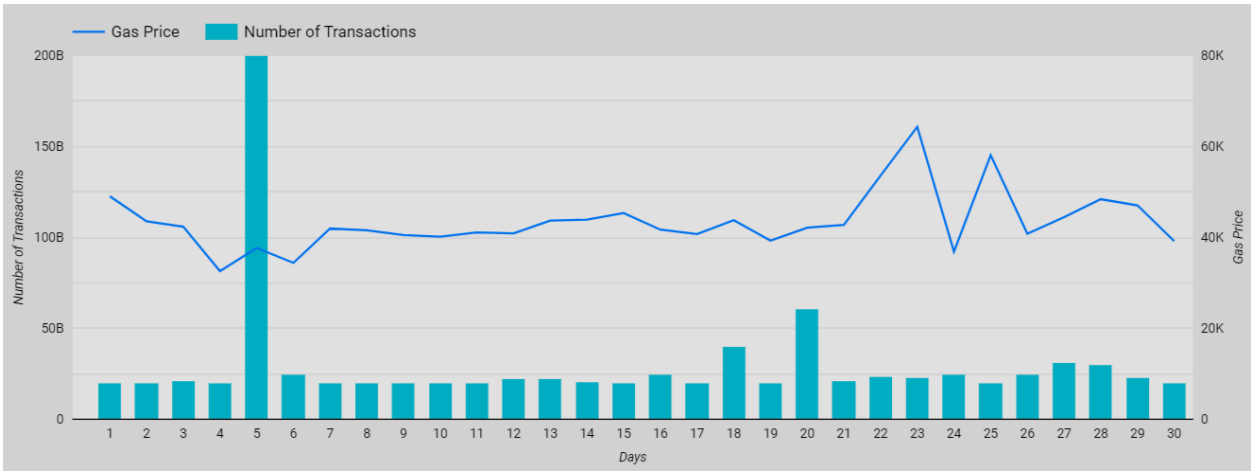
Byzantium: (October 2017)

1	[283871, 100000000000.0]
10	[349605, 20000000000.0]
12	[363411, 28000000000.0]
14	[332169, 25000000000.0]
16	[408297, 31000000000.0]
18	[470449, 4000000000.0]
21	[462902, 20000000000.0]
23	[468168, 4000000001.0]
25	[460419, 150000000.0]
27	[486147, 20000000000.0]

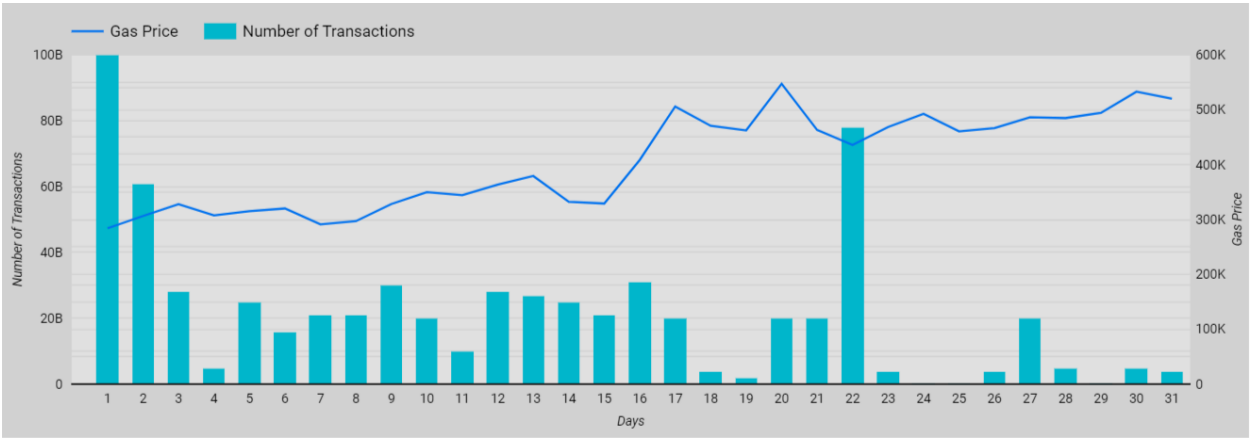
PLOT:

The gas prices vs No of transactions per day for both the forks i.e Spurious Dragon for the month of November 2016 and Byzantium for the month of October 2017 have been plotted.

Spurious Dragon:(November 2016)



Byzantium: (October 2017)



OBSERVATION: The gas prices and number of transactions drastically drop on the days of forking which is October 4,2017 for Byzantium and November 22, 2016 for Spurious Dragon which can be seen from the graphs above. Both these falls are just on the days of the fork and then the gas prices and number of transactions show a considerable hike.

2. GAS GUZZLERS (10%)

PROBLEM STATEMENT: To get average gas price and average gas limit for each month between the start and end of the dataset. Plot a graph to analyze the trends.

DATASET: Transactions, Contracts and Blocks

SOLUTION:

- The mapper yields the key which is the month and year and value which is the Gas Price/Gas Limit respectively for both the jobs.
- The reducer is to aggregate the count of transactions and the total value of Gas Price/Gas Limit in the specific months and divide it by count and then yield the output.

- **GAS PRICE**

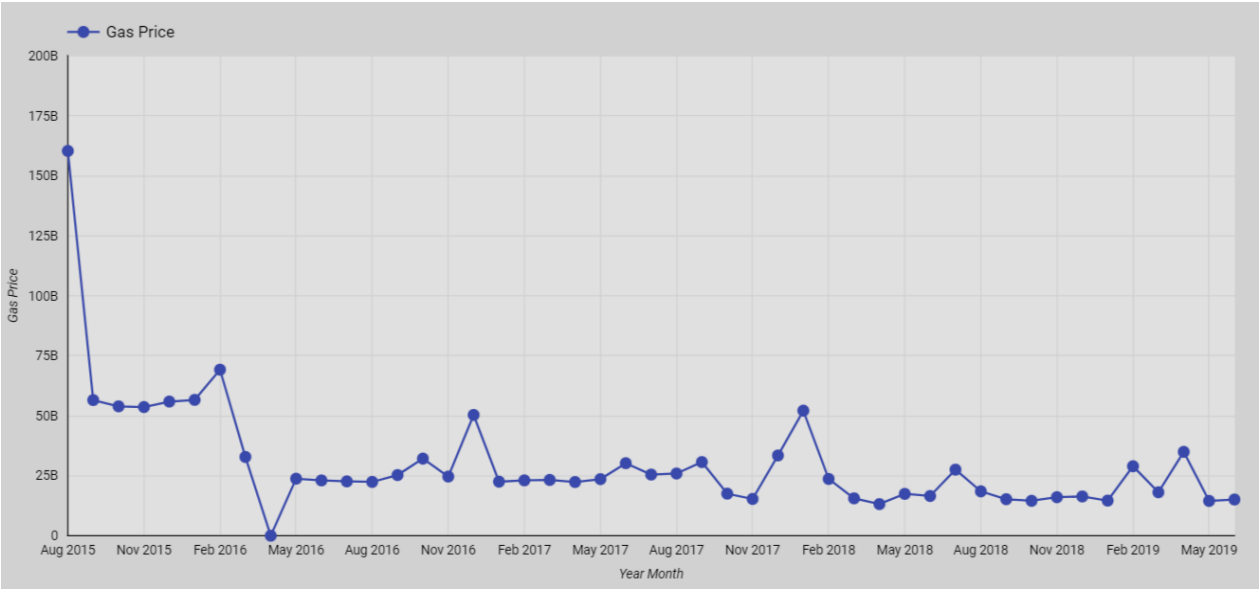
JOB ID:

http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1649894236110_5352/

SAMPLE OUTPUT:

[1, 2017]	22507570808
[1, 2019]	14611816446
[10, 2015]	53898497955
[10, 2017]	17509171845
[11, 2016]	24634294365
[11, 2018]	16034859009
[12, 2015]	55899526672
[12, 2017]	33423472930
[2, 2016]	69180681134
[2, 2018]	23636574204

PLOT:



OBSERVATION: There is a gradual decrease in prices at the start of the dataset and then we can see that there is an upward trend in the gas prices and gradually remains the same except for certain hikes before the major forks.

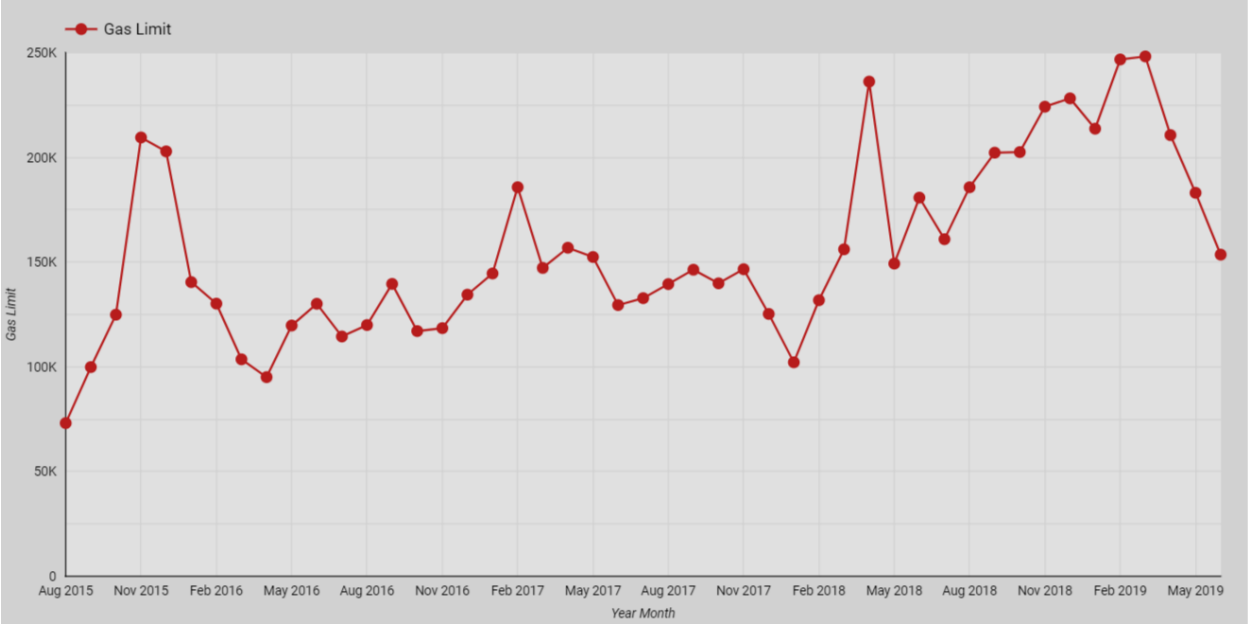
- **GAS LIMIT**

JOB ID:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1637317090236_24257/
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1637317090236_24271/

SAMPLE OUTPUT:

[1, 2017]	144585.8323
[1, 2019]	213715.9605
[10, 2015]	124898.7635
[10, 2017]	139870.0591
[11, 2016]	118474.207
[11, 2018]	224208.9518
[12, 2015]	202883.6323
[12, 2017]	125269.5105
[2, 2016]	130175.5214
[2, 2018]	131824.697
[3, 2017]	147228.1978
[3, 2019]	248217.4293

PLOT:



OBSERVATION: There is an upward trend in the gas limit over the years as shown by the graph above with sudden hikes right before the major forks in Ethereum.

3. POPULAR SCAMS (15%)

PROBLEM STATEMENT:

DATASET: Transactions and scams.json

SOLUTION:

Job1:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_8205/
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_8272/

- The fields of the dataset are checked if the length is equal to 7 and if it is then it is of the transaction dataset and if it is not then it means that it is not our required dataset and then the scams.json file is loaded.
- The mapper takes the address as key and value as value and here 0 is used to distinguish between the scams.json data. When the address has been taken from the json dataset

Out1:

Scamming	3.83361628624443e+22
Fake ICO	1.35645756688963e+21
Phishing	2.6999375794087423e+22
Scam	0

Job2:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1649894236110_5727/

- The fields of the dataset are checked if the length is equal to 7 and if it is then it is of the transaction dataset and if it is not then it means that it is not our required dataset and then the scams.json file is loaded.
- Mapper takes address as the key and 1 is yielded to distinguish between the chosen dataset from transactions and scams.
- The reducer also takes address as a key from the for loop and for the value we take category and status of the scam and 2 will be used to differentiate between the datasets.
- Then the values are checked and the category and status of the scam is taken from the json file.
- The second mapper takes the values from the previous reducers and gives it to the second reducer in which the sum of the total number of scams of any particular category.

Out2:

["Active", "Scamming"]	2.2096952356679117e+22
["Inactive", "Phishing"]	1.488677770799503e+19
["Offline", "Fake ICO"]	1.35645756688963e+21
["Offline", "Phishing"]	2.2451251236751494e+22
["Offline", "Scam"]	0
["Suspended", "Phishing"]	1.63990813e+18
["Active", "Phishing"]	4.531597871497939e+21
["Offline", "Scamming"]	1.6235500337815102e+22
["Suspended", "Scamming"]	3.71016795e+18

4. COMPARATIVE EVALUATION(10%)

PROBLEM STATEMENT: To run the Part B code in Spark and compare the performance on basis of the time taken by both the executions.

DATASET: Transactions and Contracts

JOB ID:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1649894236110_5750_

OUTPUT:

0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444:84155100809965865822726776

0xfa52274dd61e1643d2205169732f29114bc240b3:45787484483189352986478805

0x7727e5113d1d161373623e5f49fd568b4f543a9e:45620624001350712557268573

0x209c4784ab1e8183cf58ca33cb740efbf3fc18ef:43170356092262468919298969

0x6fc82a5fe25a5cdb58bc74600a40a69c065263f8:27068921582019542499882877

0xbfc39b6f805a9e40e77291aff27aee3c96915bdd:21104195138093660050000000

0xe94b04a0fed112f3664e45adb2b8915693dd5ff3:15562398956802112254719409

0xbb9bc244d798123fde783fcc1c72d3bb8c189413:11983608729202893846818681

0xabbb6bebfa05aa13e908eaa492bd7a8343760477:11706457177940895521770404

0x341e790174e3a4d35b65fdc067b6b5634a61caea:8379000751917755624057500

250.65382738176

RESULTS:
For Spark:
1st Run: 250.65382738176
2nd Run: 180.78689638218
3rd Run: 257.78665342835
Average: 229 seconds approximately.

For MapReduce:
1st Run: 13,147 s
2nd Run:11,350 s
3rd Run: 14,517 s
Average: 13,004 seconds approximately

OBSERVATION:

- The working of the program is similar to that of Map Reduce in which we use both transactions and blocks dataset.
- We check the field 7 for transactions and field 5 for contracts and then each address is mapped to their transaction value.
- Time library has been imported to calculate the time taken to run the program.
- Average time taken by Map Reduce is more than that of Spark.