ECS765P - Big Data Processing - 2022/23

Course Work

Name: Waleed ul hassan

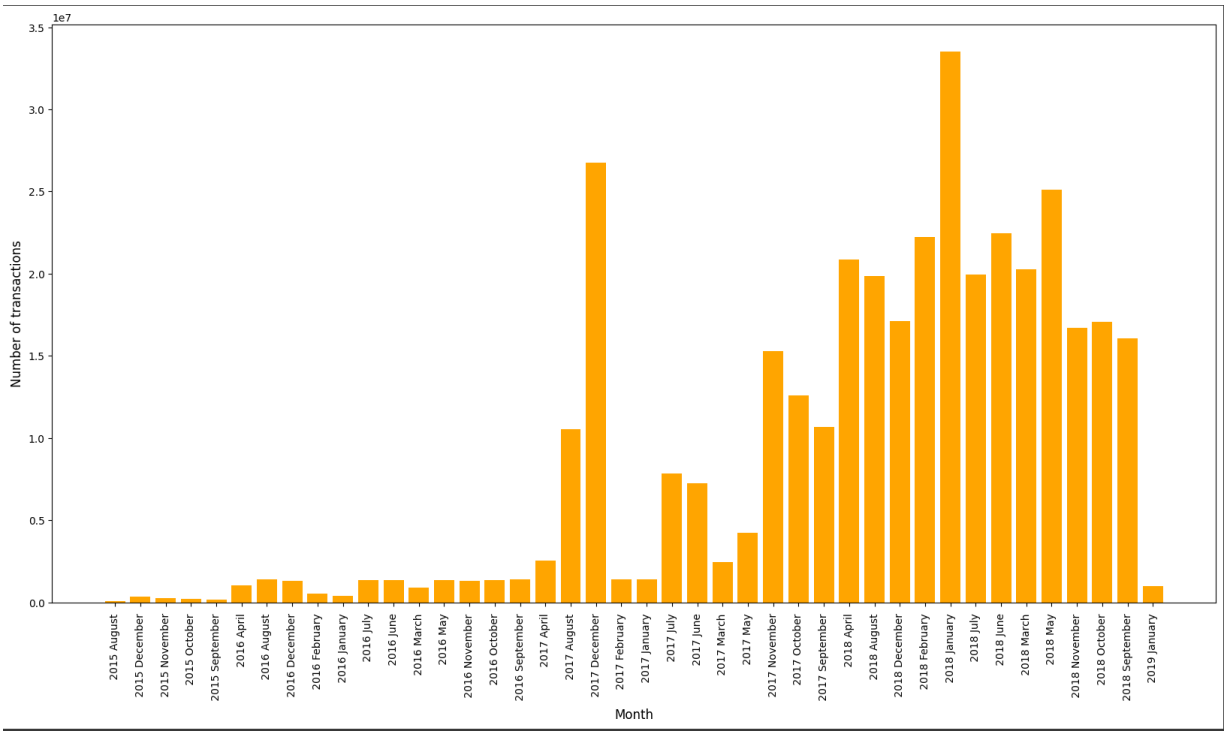
ID: 220906955

Task A

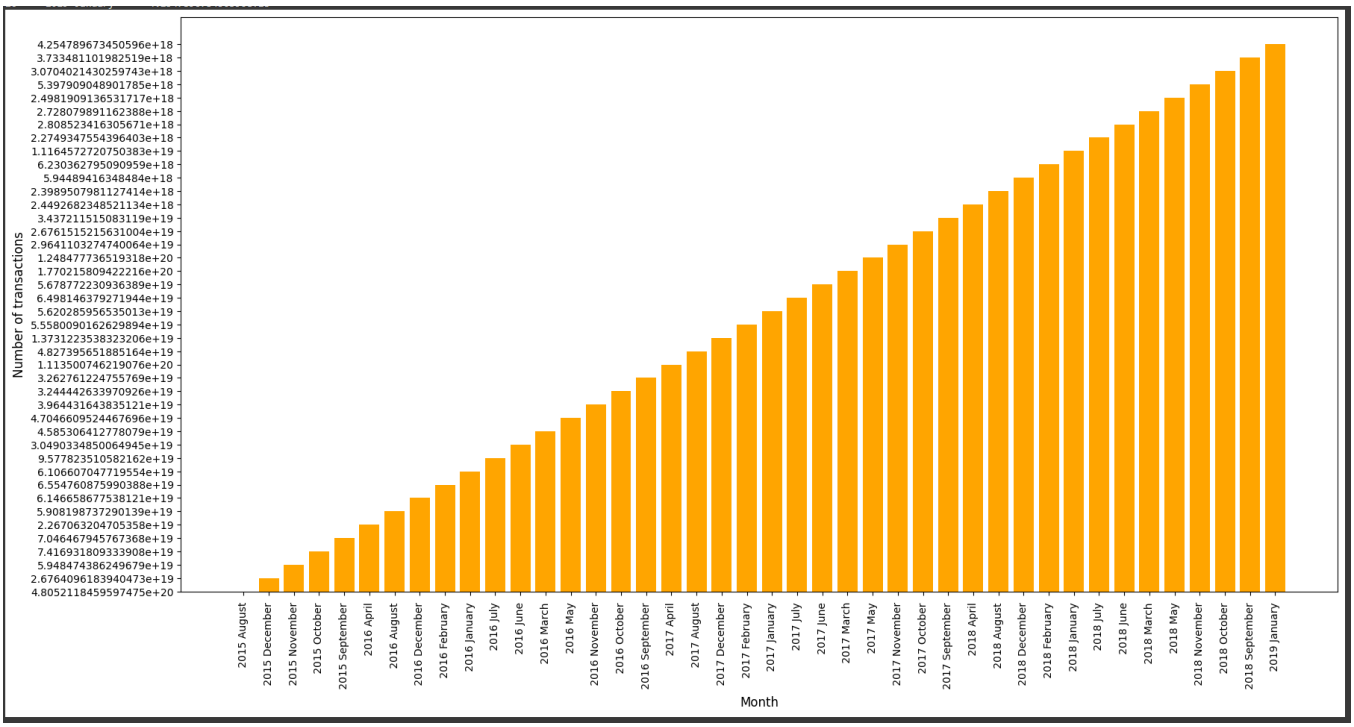
Time Analysis (25%)

1. First, I used the function ValidDateFilter() to get all the clean lines from dataset. Then I mapped the 11th index of the line when we split it by (‘,’) which is the time stamp with value 1 so that we can calculate the total values for the same month. I formatted the time stamp into time string as ‘Month – year’ so that we get value for month and not each day. Then I used reduceByKey() function to sum the values for same month and year. After that I used Colab to plot the bar plot using pandas and matplotlib using the transactions.txt file that I saved all the data I got from running the spark job.

Plot



2. First, I cleaned the dataset using filter function then I mapped time stamp , value of transaction and ‘1’ for counting the transactions then I used the reduceByKey function to calculate count for each month and adding up values for each month. After that I used (value of transaction/Count) by mapping. After that I joined each entry with ‘,’ and saved the resulting dataset in avarage.txt , I used Colab to plot the bar plot using pandas and matplotlib.



Task B

Top Ten Most Popular Services (25%)

First of all, I loaded both the transaction and contracts csv files and then I filtered and mapped both of the datasets according to the fields I need, for example choosing to_address and value from transactions dataset. Then I joined both the datasets using .join() to find out smart contracts and after that I mapped the dataset for address and value only and used takeOrdered() function to get top 10 smart contracts and saved it in the top10SmartContracts.txt file

Result:

[“Address of the contract”, Value of contract]

[["0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444", 84155363699941767867374641],
["0x7727e5113d1d161373623e5f49fd568b4f543a9e", 45627128512915344587749920],
["0x209c4784ab1e8183cf58ca33cb740efbf3fc18ef", 42552989136413198919298969],
["0xbfc39b6f805a9e40e77291aff27aee3c96915bdd", 21104195138093660050000000],
["0xe94b04a0fed112f3664e45adb2b8915693dd5ff3", 15543077635263742254719409],
["0xabbb6bebfa05aa13e908eaa492bd7a8343760477", 10719485945628946136524680],
["0x341e790174e3a4d35b65fdc067b6b5634a61caea", 83790007519177556240575000],
["0x58ae42a38d6b33a1e31492b60465fa80da595755", 29027091871057365328638180],
["0xc7c7f6660102e9a1fee1390df5c76ea5a5572ed3", 12380861145200420000000000],
["0xe28e72fcf78647adce1f1252f240bbfaebd63bcc", 11724264325158231427145820]]

Task C

Top Ten Most Active Miners (10%)

So, I loaded the block.csv file and mapped the dataset for address field of miner and size of the block after cleaning the dataset with filter function. Then I used reduceByKey function to sum up the total size of block mined by a miner and then used takeOrdered() function to get top 10 miners and saved it to top10Mners.txt

Result:

[“Address of the Miner”, Total Block size]

[["0xea674fdde714fd979de3edf0f56aa9716b898ec8", 17453393724],
["0x829bd824b016326a401d083b33d092293333a830", 12310472526],
["0x5a0b54d5dc17e0aadc383d2db43b0a0d3e029c4c", 8825710065],
["0x52bc44d5378309ee2abf1539bf71de1b7d7be3b5", 8451574409],
["0xb2930b35844a230f00e51431acae96fe543a0347", 6614130661],
["0x2a65aca4d5fc5b5c859090a6c34d164135398226", 3173096011],
["0xf3b9d2c81f2b24b0fa0acaaa865b7d9ced5fc2fb", 1152847020],
["0x4bb96091ee9d802ed039c4d1a5f6216f90f81b01", 1134151226],
["0x1e9939daaad6924ad004c2560e90804164900341", 1080436358],
["0x61c808d82a3ac53231750dadc13c777b59310bd9", 692942577]]

Task D

Data exploration (40%)

Wash Trading

Explanation:

First I calculated the total number of transactions that are unique by cleaning the data using a filter and mapping it as from_address and to_address to get number of transactions made from each address to every other address. Then I calculated the average number of transaction a single address has made to the same address by using (Unique transactions.count() / Transaction.count()) (I didn’t include the code for calculating this because I had to run the spark job again and

again to save runtime the average was around 2.9) so I selected a random big values of 1000 to check if two addresses have more then 1000 transactions to the same addresses to decrease the dataset so I can load it to nx.DiGraph which is used to create a directed graph, to save memory as we cannot load the whole dataset to the graph. After this when I have all the unique addresses that have more then 1000 transactions between them I saved that dataset to wash.txt

Creating the graph and detecting cycles:

I loaded the json data in colab from wash.txt and created a graph instance using nx.DiGraph() and then added all the transactions as edges of the graph using “g.add_edge(jsonData[i][0][0], jsonData[i][0][1])” after that I used built in function of Networkx nx.recursive_simple_cycles(g) to get all the cycles in the data set to see the wash trades between multiple addresses. It returned a list of address in which some of them are wash trading with themselves as the receivers and some of them are using complex cycles using around 5 to 6 address.

Result:

[['0x2984581ece53a4390d1f568673cf693139c97049']

, ['0x3b981de0da45b790a0748e7be05cdeaee68fb162']

, ['0x7ed1e469fcb3ee19c0366d829e291451be638e59']

, ['0xfb6ca40775bff429004297e675166286c3391700']

, ['0xa64c326d61bfcc4f9f9a5d937095b0ddb7dd63ae']

, ['0x62600d264c2f76b40469b6d55f357361b0229469']

, ['0xad68942a95fdd56594aa5cf862b358790e37834c']

, ['0x28b1724617c607c9be1bf3e2ba240f6fd766b781']

, ['0x001f31e16de65bc780742a590d955d7526ce38f9']

, ['0x02459d2ea9a008342d8685dae79d213f14a87d43']

, ['0x55ae3f67039c332f55be00c1d33d989d2da108c5']

, ['0x203154b886e9c6948bddbce27484bf2ffd402ea1']

, ['0x82817c5528f8593e9e88a6a583bf5e77326bdb55']

, ['0x3c540be890df69eca5f0099bbedd5d667bd693f3']

, ['0x715160df73fedef0699de0631d78d34590757361']

, ['0xe5801f3145dfc88dbe34c0872821849141faccc2']

, ['0x6201e492e99f23ff42b7677ef1790ccecea8f12a']

, ['0xcf398a8363fda31ef003ccbbb9ff66f9533f279f']

, ['0xe711fa745e7ef32a87a91c15322886aa0567ea5a']

, ['0x01bd54587e981aed414edb8a4609c0f767239533']

, ['0xf3a71cc1be5ce833c471e3f25aa391f9cd56e1aa']

, ['0xf0f7e8b1c9f5a5deae033e33b6ba22bdab3f61c3']

, ['0xf661e08b763d4906457d54c302669ec5e8a24e37']

, ['0x5a289646bfb958c3158a33b1e4b8a8c254b3b5aa']

, ['0x0c5437b0b6906321cca17af681d59baf60afe7d6']

, ['0x32362fbfff69b9d31f3aae04faa56f0edee94b1d']

, ['0x9091be2568df2bed63731ee2c8e7a9f0330ebf9b']

, ['0x4c34ae54dc716808e94af3d1d638b8ea3a23fa9b']

, ['0xf77d8ffca7e769226a8365c35a40715c38c3cbdf']

, ['0x1dec1b49cdde549c25255a3afc90f1bd51094089']

, ['0xe614ddd849ff3ad5a1487a88c33eed9ff4872d04']

, ['0x5173829b2daeac6ffbccc10e6b351d8380c14a9e']

, ['0xdd3e4522bdd3ec68bc5ff272bf2c64b9957d9563']

, ['0x08a373747fc44b0ad2f1d02f74b7c794b2ea802d']

, ['0x7d942f90a8d9c9b3a0b540abc8f5c33314db296c']

, ['0xc257274276a4e539741ca11b590b9447b26a8051',
'0x22b84d5ffea8b801c0422afe752377a64aa738c2', '0x47e847fd1c3129d59990046aade26e3ccc7caf48',
'0x876eabf441b2ee5b5b0554fd502a8e0600950cfa',
'0xb795ea294044bbcc4b61c661c3cbe9acb374eaca']

, ['0xc257274276a4e539741ca11b590b9447b26a8051',
'0x22b84d5ffea8b801c0422afe752377a64aa738c2', '0x47e847fd1c3129d59990046aade26e3ccc7caf48',
'0x876eabf441b2ee5b5b0554fd502a8e0600950cfa',
'0x70faa28a6b8d6829a4b1e649d26ec9a2a39ba413',
'0x0bee799393a298a2e2b08ba47c28bdcd95889bef']

, ['0xc257274276a4e539741ca11b590b9447b26a8051',
'0x22b84d5ffea8b801c0422afe752377a64aa738c2', '0x47e847fd1c3129d59990046aade26e3ccc7caf48',
'0x876eabf441b2ee5b5b0554fd502a8e0600950cfa',
'0x70faa28a6b8d6829a4b1e649d26ec9a2a39ba413',
'0xba798e819217bbc90b2c4126f059b68b8b4909f6']

, ['0x0c462ed55a7d553867b78e845d40216d2b501181',
'0xa073cfdc059caf115fe12c938884fd447b62086a']

, ['0xda1e5d4cc9873963f788562354b55a772253b92f',
'0x876eabf441b2ee5b5b0554fd502a8e0600950cfa',
'0x70faa28a6b8d6829a4b1e649d26ec9a2a39ba413']

, ['0x876eabf441b2ee5b5b0554fd502a8e0600950cfa',
'0x8bbb73bcb5d553b5a556358d27625323fd781d37']

, ['0x86c9981b0d85e1cd6f42b10b8305ffd88f64f55e',
'0x211e96f7a2dfbc2512331f69976445b28f8e63e6']

, ['0x86c9981b0d85e1cd6f42b10b8305ffd88f64f55e',
'0x211e96f7a2dfbc2512331f69976445b28f8e63e6', '0xf8165fe1c2cc5360049e2b9c6bd88432a01c0d24']

, ['0x86c9981b0d85e1cd6f42b10b8305ffd88f64f55e',
'0xf8165fe1c2cc5360049e2b9c6bd88432a01c0d24', '0x211e96f7a2dfbc2512331f69976445b28f8e63e6']

, ['0x86c9981b0d85e1cd6f42b10b8305ffd88f64f55e',
'0xf8165fe1c2cc5360049e2b9c6bd88432a01c0d24']

, ['0x86c9981b0d85e1cd6f42b10b8305ffd88f64f55e',
'0xf8165fe1c2cc5360049e2b9c6bd88432a01c0d24', '0x3143e1cabc3547acde8b630e0dea3b1dfebb3cb0',
'0x211e96f7a2dfbc2512331f69976445b28f8e63e6']

, ['0x211e96f7a2dfbc2512331f69976445b28f8e63e6',
'0xf8165fe1c2cc5360049e2b9c6bd88432a01c0d24']

, ['0x211e96f7a2dfbc2512331f69976445b28f8e63e6',
'0xf8165fe1c2cc5360049e2b9c6bd88432a01c0d24',
'0x3143e1cabc3547acde8b630e0dea3b1dfebb3cb0']

, ['0xd0aedab61b09d85fbbfaa0d3bd9797c1c13dd5af',
'0x3d98cda129757329c82edbc090234cfa77478dc1']

, ['0xf8165fe1c2cc5360049e2b9c6bd88432a01c0d24',
'0x3143e1cabc3547acde8b630e0dea3b1dfebb3cb0']]

Task D

Data exploration (40%)

Gas Guzzlers

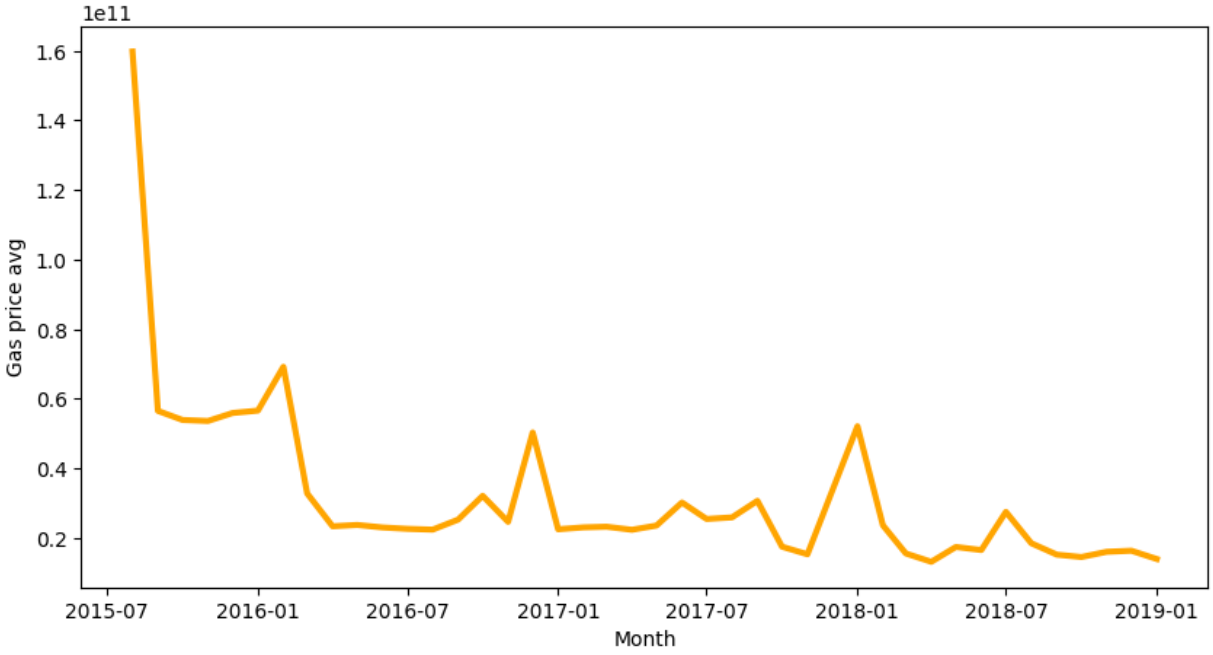
Explanation:

Loaded both transactions and contracts files and filtered them with .filter() after that I mapped Gas price with 1 and using Date as key to count total number of transactions and added price for each month to get average for every month using reduceByKey()

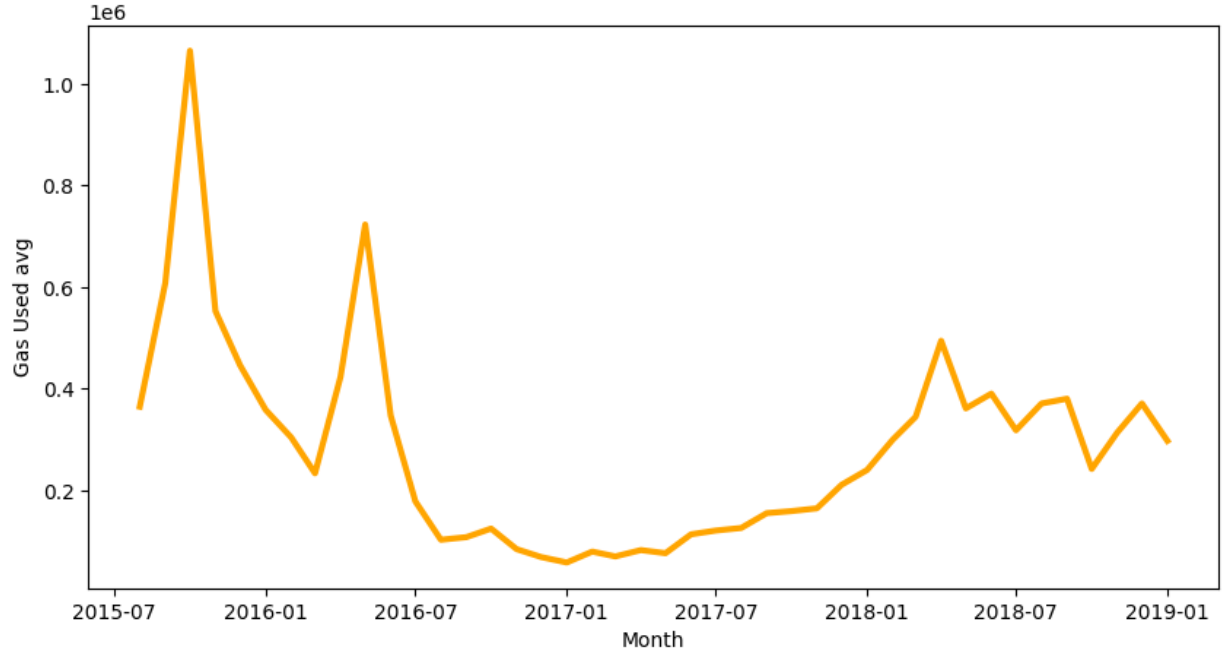
For Avg gas used I joined contract and transactions using the to_address field and calculated the avg used gas by getting a count of contracts count using reduceByKey and sorted it according to Date

After getting these two files for avgGasused.txt and avgGasprice.txt I used colab to plot graphs for both of them using pandas and matplotlib

Gas Price Avg


1e11
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
Gas price avg
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
Month

Gas Used Avg


1e6
1.0
0.8
0.6
0.4
0.2
Gas Used avg
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
Month

Note: I included the notebook for all the plotting logic and graph logic for wash trading with name CourseWork.ipynb