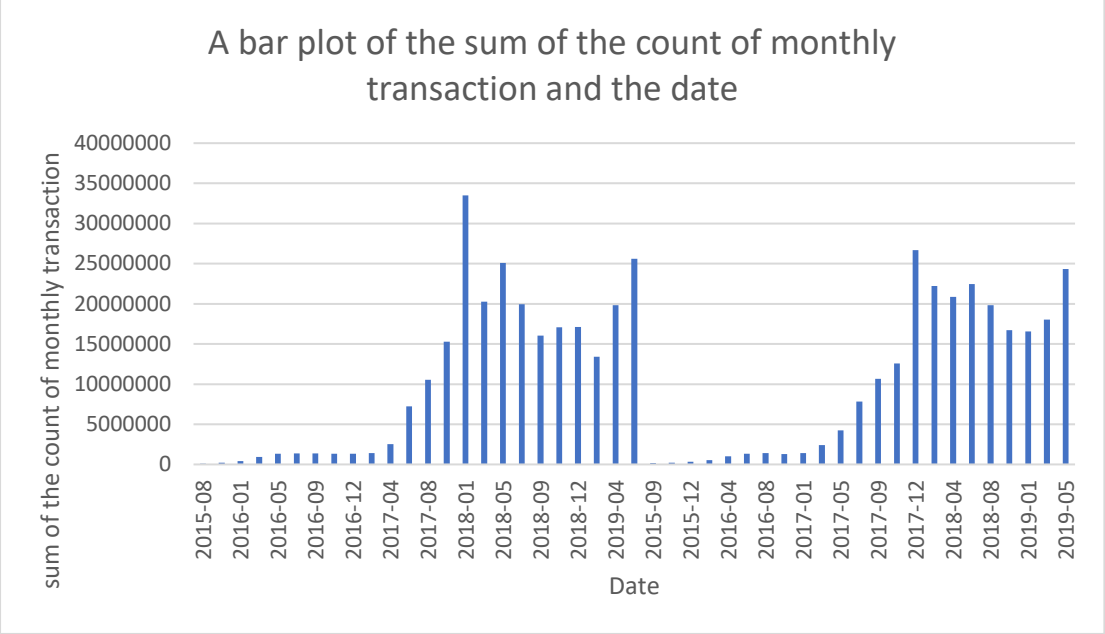
*A file called new.py was used to remove the quotation marks and brackets from the txt file.

Part A:

count of monthly transactions- job_1606730688641_7273

In the mapper the columns of transaction dataset were split via a comma. Then, the data “block_timestamp” was converted into a float and stored. To convert the time stamps into a format of months and years, time.strftime("%Y-%m",time.gmtime(time_epoch)) was used. A count per month-year in the dataset and the month-year(month variable) were yielded.

In the reducer, the month-year(month variable) and the count were taken as inputs. The sum of the count and the month-year(month variable) were yielded. The results were then taken and used in excel to create a bar plot.

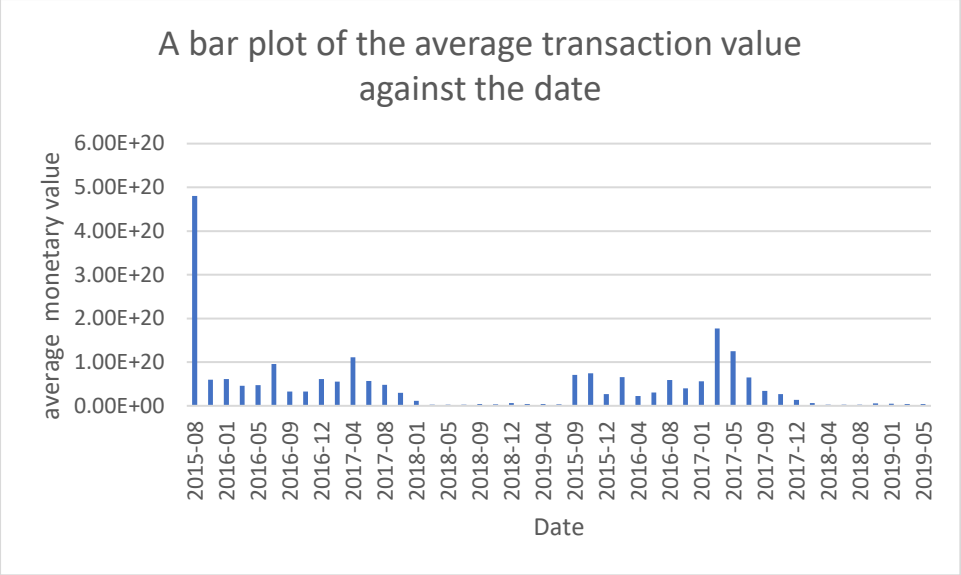


Part A2:

average monthly transaction value- job_1606730688641_7298

The same method was used in the mapper as in the “count of monthly transaction” was used. Also, the “value” column from the transaction dataset was converted to a float and stored. The date (month variable) and the value (mon-value variable) and a count of 1 were yielded.

To find the average transactional value(avg_value variable), the count and total were calculated. The total and count were calculated in a for loop by iterating themselves and adding the value assigned from the mapper as v[0] for the value(mon_value) and v[1] for the count. The average transaction value(avg_value) was calculated by diving the total by the count. The month and average transaction value were then yield. The results were then taken and used in excel to create a bar plot.



Part B:

JOB 1 - INITIAL AGGREGATION- job_1607539937312_4399

The same method was used in the mapper as in “part A’s the average monthly transaction value” was used. Instead of the month being initialised and emitted, the “to_address” column in the transaction dataset was used. No count was yielded.

In the reducer the to_address(address variable) and the sum of the monetary values(mon_value) were yielded into a text file. A combiner was used, which had the same operation as the reducer. This was done to decrease the amount data the reducer needed to process. Also 10 reducers were initialised to further reduce the time taken to yield the results.

JOB 2 - JOINING TRANSACTIONS/CONTRACTS AND FILTERING- job_1607539937312_4422

The data in the txt file from job 1 was used in the mapper (by entering the txt file as input in the command line) to get the to_address(address variable) and the sum of the monetary values(mon_value variable) which is assigned a value 1. Also, the “address” column(join_key) from the contract dataset stored and assigned a 2.

In the reducer, we use what was assigned a count of a 1 or 2 from mapper, the values are assigned to variable or a list created in the reducer. If the value is equal to 1 from the mapper it is assigned to variable whereas if the value equals 2 it is assigned to the list. To remove case where value is not present in both list and variable, the value is filter out and not yielded. If the value is present in both the address from the block and the address from the contract, it is yielded.

JOB 3 - TOP TEN- job_1607539937312_4430

The data in the txt file from job 2 was used in the mapper (by entering the txt file as input in the command line). To yield the to_address(address variable) and the contract address(addresses variable).

In the reducer, the values of the addresses were sorted and the top ten were yielded.

PART C. TOP TEN MOST ACTIVE MINERS- job_1607539937312_1995, job_1607539937312_7053

The same method was used in the mapper and the reducer as in “part B’s initial aggregation” was used. Instead, the mapper was used to find the size and miner in the “blocks” dataset. The reducer was used to yield the sum of size and the miner.

Looking at the txt file created, the minimum aggregated size was 692942577 for the miner to be in the top ten. As such a file partC1.py was created using the value as minimum value to be yielded from the reducer.

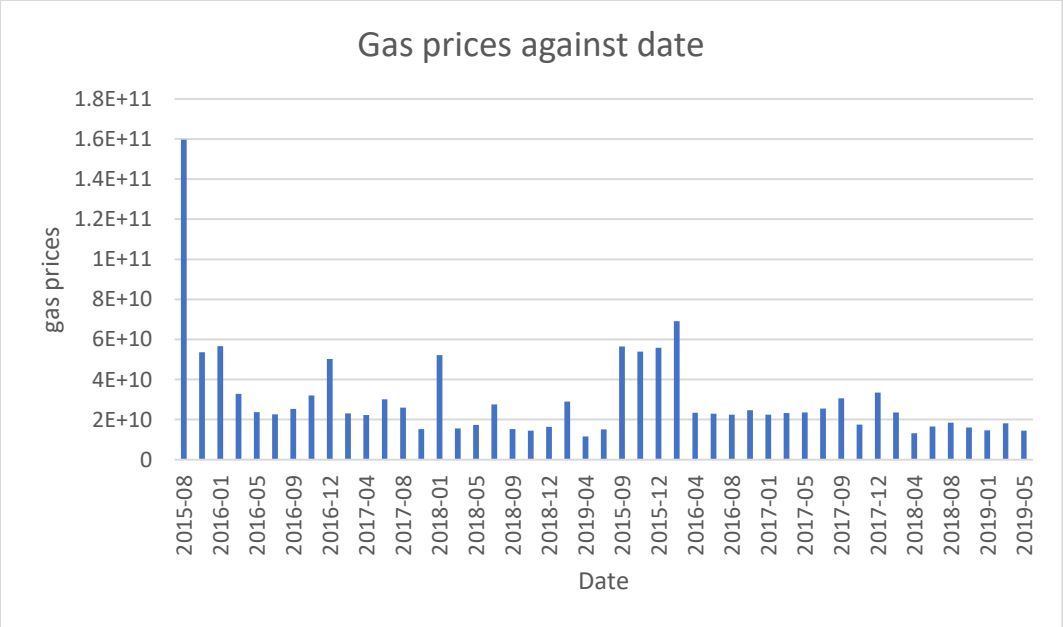
Part D1-gas guzzlers- job_1607539937312_2369

The same method was used in the mapper and the reducer as in “average monthly transaction value” was used. Instead, the mapper was used to find the gas prices(gas_price) and date(month) in the "transaction” dataset. The reducer was used to yield the average price of gas and the month.

Part D1-gas guzzlers2-job_1607539937312_3418

The same method was used in the mapper and the reducer as in “part B’s joining transactions/contracts and filtering” was used. Instead, the mapper used “to_address” and “gas” from the transaction dataset and the “address” from the contract dataset. The address is variable is assigned a 2 while the gas is assigned a 1 in the mapper.

In the reducer, we use what was assigned a count of a 1 or 2 from mapper, the values are assigned to variable or a list created in the reducer. If the value is equal to 1 from the mapper it is assigned to variable whereas if the value equals 2 it is assigned to the list. To remove case where value is not present in both list and variable, the value is filter out and not yielded. If the value is present in both the address from the block and the address from the contract, the gas used and the address in the contract are yielded.



As we can see by the bar plot, the price of gas decreased over times. However, as contracts became more complicated over time more gas was required. This correlates with what was seen in part B as we see the value of contracts increase over time.

Part D2-scams-job_1607539937312_7487

There were multiple mappers used in the scam part. First mapper was used to convert the data in the scam.json file into dictionary to be used. The second mapper was used get the values needed from the transaction dataset and to take the address from the json file. The last mapper was used to yield the information in the scam, the month and the total value of the scam.

The first reducer was used to find the total of the value and yield both the value and the key. The final reducer was used to sort, and the values based on their totals.