PART A: TIME ANALYSIS

A.1: Number of transactions per month

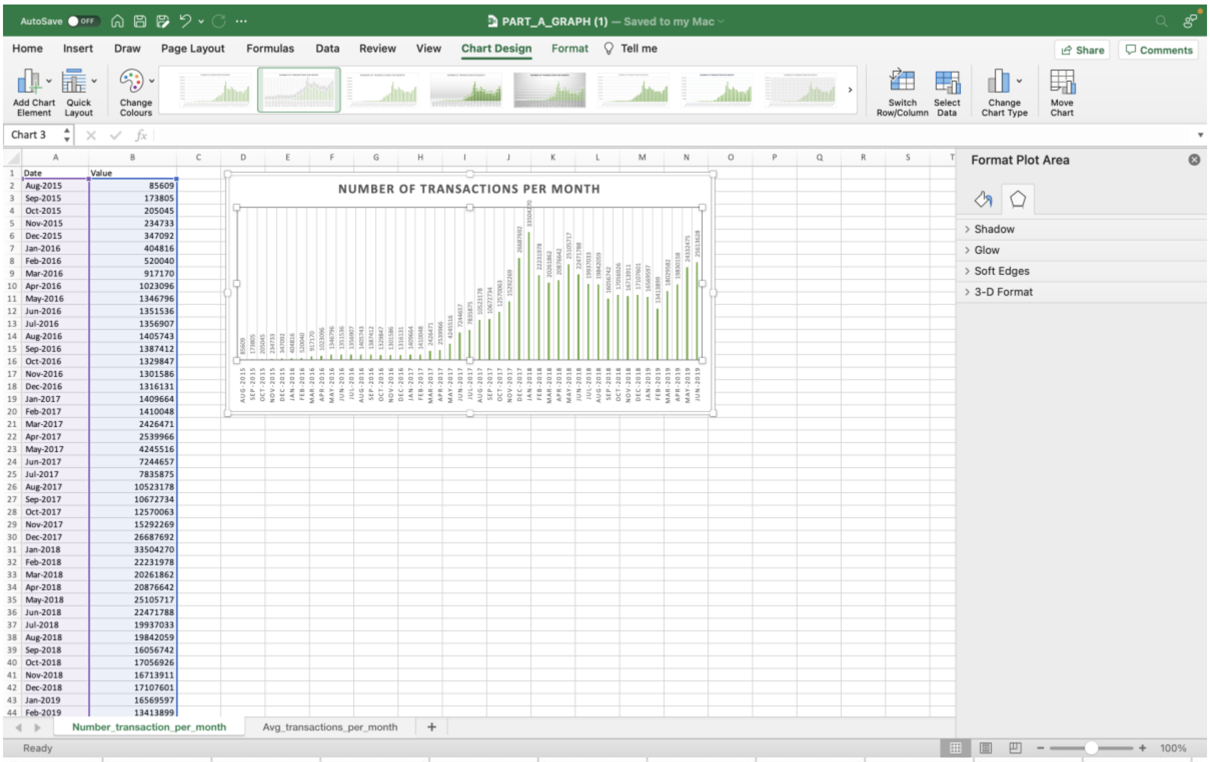
To get the monthly transaction total, we used the mapper's 'datetime.utcfromtimestamp' functionality to extract the date in the desired format, which includes month and year.

In the reducer section, the counts for the various keys are then added together.

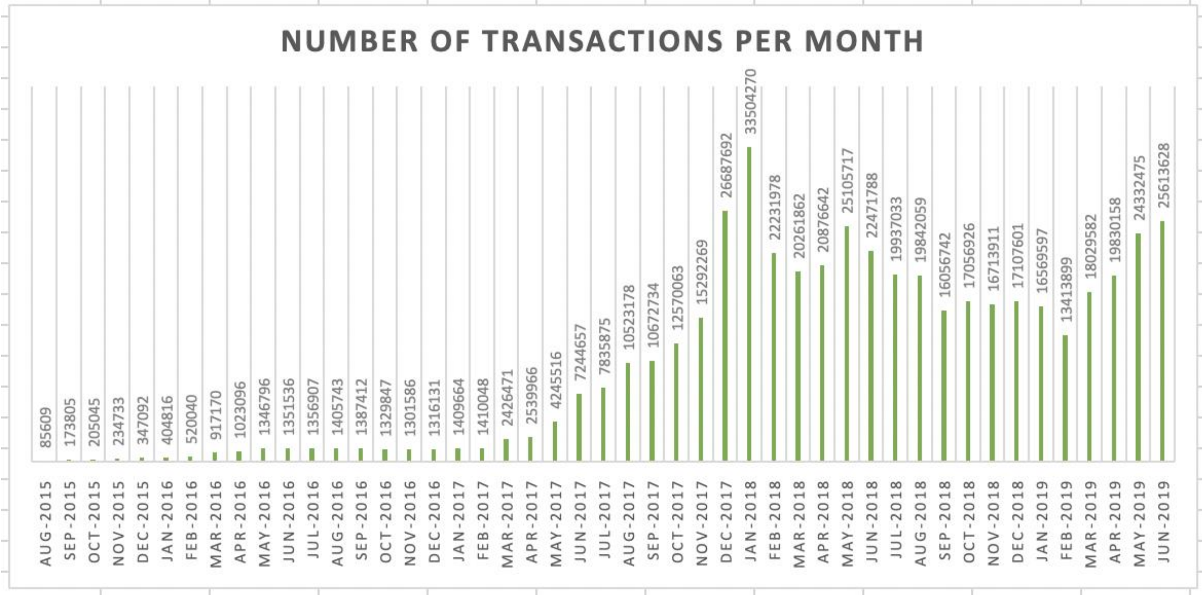
Execution command: python no_trans.py -r hadoop --output-dir trans_per_month --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/part-*.csv

```
jmp01@itl211 ~/ecs765/ass1/partA> cat no_trans.py
from mrjob.job import MRJob
from datetime import datetime
import time
class no_trans(MRJob):

        def mapper(self, _, line):
                try:
                        fields = line.split(',')
                        if len(fields) == 7:
                                day = (datetime.utcfromtimestamp(int(fields[6])).strftime('%Y-%m'))
                                value = int(fields[3])
                                yield (day,1)

                except :
                        pass
        def reducer(self,u,t):
                yield u, sum(t)

if __name__ == '__main__':
        no_trans.run()


# python no_trans.py -r hadoop --output-dir trans_per_month --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/part-*.csv

# job: http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_6967/
  Job job_1648683650522_6967 completed successfully
  Output directory: hdfs:///user/jmp01/trans_per_month
Counters: 55
        File Input Format Counters
                Bytes Read=65309532836
        File Output Format Counters
                Bytes Written=860
        File System Counters
                FILE: Number of bytes read=321522456
                FILE: Number of bytes written=888562378
                FILE: Number of large read operations=0
                FILE: Number of read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=65309715788
                HDFS: Number of bytes read erasure-coded=0
                HDFS: Number of bytes written=860
                HDFS: Number of large read operations=0
                HDFS: Number of read operations=3277
                HDFS: Number of write operations=4
        Job Counters
                Data-local map tasks=986
                Launched map tasks=1089
                Launched reduce tasks=2
                Rack-local map tasks=103
                Total megabyte-milliseconds taken by all map tasks=63529036800
                Total megabyte-milliseconds taken by all reduce tasks=12193628160
                Total time spent by all map tasks (ms)=12408015
                Total time spent by all maps in occupied slots (ms)=62040075
                Total time spent by all reduce tasks (ms)=2381568
                Total time spent by all reduces in occupied slots (ms)=11907840
                Total vcore-milliseconds taken by all map tasks=12408015
                Total vcore-milliseconds taken by all reduce tasks=2381568
        Map-Reduce Framework
                CPU time spent (ms)=6358430
                Combine input records=0
                Combine output records=0
                Failed Shuffles=0
                GC time elapsed (ms)=153877
                Input split bytes=182952
                Map input records=486522454
                Map output bytes=5838256380
                Map output materialized bytes=321556430
                Map output records=486521365
                Merged Map outputs=2178
                Peak Map Physical memory (bytes)=668643328
                Peak Map Virtual memory (bytes)=2965000192
                Peak Reduce Physical memory (bytes)=1137086464
                Peak Reduce Virtual memory (bytes)=3019788288
                Physical memory (bytes) snapshot=675127648256
                Reduce input groups=47
                Reduce input records=486521365
                Reduce output records=47
                Reduce shuffle bytes=321556430
                Shuffled Maps =2178
                Spilled Records=973042730
                Total committed heap usage (bytes)=694780690432
                Virtual memory (bytes) snapshot=2982787223552
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
```

Bargraph:



Date	Value
Aug-2015	85609
Sep-2015	173805
Oct-2015	205045
Nov-2015	234733
Dec-2015	347092
Jan-2016	404816
Feb-2016	520040
Mar-2016	917170
Apr-2016	1023096
May-2016	1346796
Jun-2016	1351536
Jul-2016	1356907
Aug-2016	1405743
Sep-2016	1387412
Oct-2016	1329847
Nov-2016	1301586
Dec-2016	1316131
Jan-2017	1409664
Feb-2017	1410048
Mar-2017	2426471
Apr-2017	2539966
May-2017	4245516
Jun-2017	7244657
Jul-2017	7835875
Aug-2017	10523178
Sep-2017	10672734
Oct-2017	12570063
Nov-2017	15292269
Dec-2017	26687692
Jan-2018	33504270
Feb-2018	22231978
Mar-2018	20261862
Apr-2018	20876642
May-2018	25105717
Jun-2018	22471788
Jul-2018	19937033
Aug-2018	19842059
Sep-2018	16056742
Oct-2018	17056926
Nov-2018	16713911
Dec-2018	17107601
Jan-2019	16569597
Feb-2019	13413899

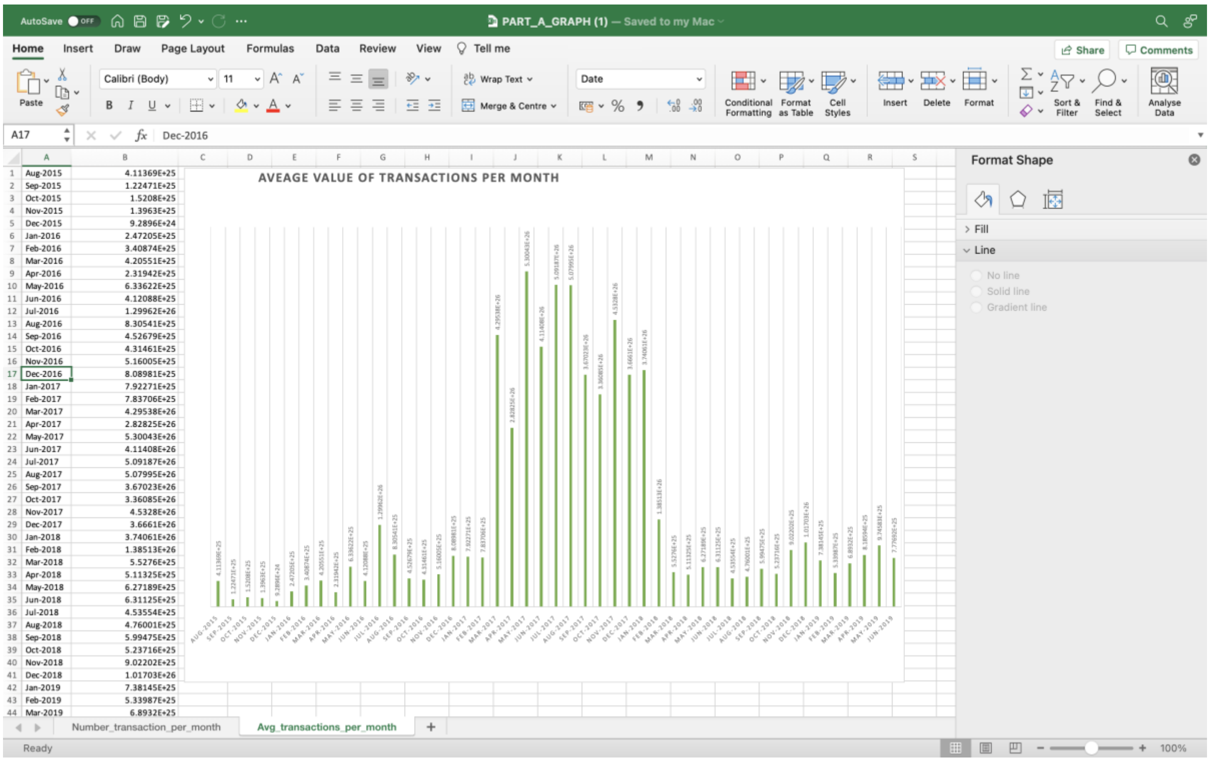


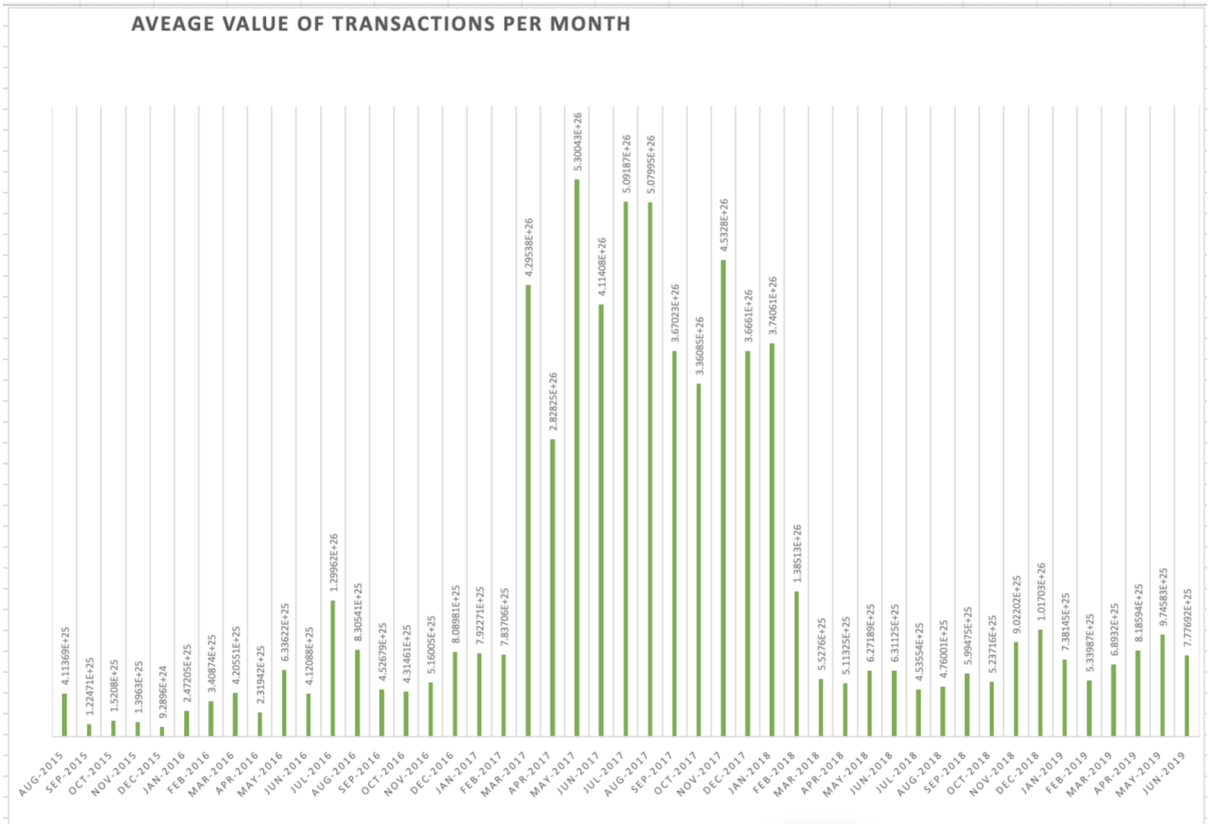
A.2: Average value of transactions per month

In the mapper, we acquired the relevant 'value' field from the transaction's schema along with the date and cumulated these in the reducer phase, like the prior code.

Execution command: python trans_value.py -r hadoop --output-dir each_month_transaction --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/part-*.csv

```
jmp01@itl211 ~/ecs765/ass1/partA> cat trans_value.py
from mrjob.job import MRJob
from mrjob.step import MRStep
import fileinput
import sys
from time import gmtime , strftime , struct_time
from datetime import datetime

class trans_value(MRJob):

 def mapper(self, _,line):
  try:
   fields = line.split(',')
   if len(fields)==7:
    timestamp_val=(datetime.utcfromtimestamp(int(fields[6])).strftime(" %b,%Y "))
    value=int(fields[3])
    yield(timestamp_val,value)

  except:
   pass

 def combiner(self,timestamp_val,value):
  yield(timestamp_val,sum(value))

 def reducer(self,timestamp_val,value):
  yield(timestamp_val, sum(value))


if __name__ == "__main__":

 trans_value.run()


# python trans_value.py -r hadoop --output-dir each_month_transaction --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/part-*.csv

# job: http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_7013/
```





PART B: TOP TEN MOST POPULAR SERVICES

1. Aggregation

The input transaction file is read and split at ',' in the first step of aggregation. We also get the key: to address and the value: value. The entire value for each unique key (to address) is then added.

Execution Command: python agg_trans.py -r hadoop --output-dir agg_trans --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/part-*.csv

```
jmp01@itl211 ~/ecs765/ass1/partB> cat agg_trans.py
from mrjob.job import MRJob
from mrjob.step import MRStep
import fileinput
import sys
from time import gmtime , strftime , struct_time
from datetime import datetime

class trans_per_m(MRJob):

 def mapper(self, _,line):
  try:
   fields = line.split(',')
   if len(fields)==7:
    to_address=fields[2]
    value=int(fields[3])
    yield(to_address,value)

  except:
   pass

 def reducer(self,to_address,value):
  yield(to_address, sum(value))


if __name__ == "__main__":

 trans_per_m.run()

# python agg_trans.py -r hadoop --output-dir agg_trans --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/part-*.csv
```

2. Repartition join between aggregate and contracts

'/user/jmp01/aggregate transactions/' and 'data/ethereum/contracts/' are the two input source directories. We look to split the input in the mapper phase based on whether the delimiter is 't' (in aggregate transactions with 2 fields) OR ',' (in contracts with 5 fields). We also get key(address) and values (block number).

Only when a key(address) receives values from both 'aggregate transactions' and 'contracts' will the reducer yield in the subsequent reducer phase (address, value). As a result, only smart contracts are allowed to pass.

The result includes a list of smart contracts as well as their total values.

Execution command: python join_task2.py -r hadoop --output-dir part_b_joined_records --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/contracts/ hdfs://andromeda.eecs.qmul.ac.uk/user/jmp01/aggregate_transactions/

```
jmp01@itl211 ~/ecs765/ass1/partB> cat join_task2.py
from mrjob.job import MRJob
from mrjob.step import MRStep
import fileinput
import sys
from time import gmtime , strftime , struct_time
from datetime import datetime

class transaction_per_month(MRJob):

    def mapper(self,_,line):
     try:
      if (len(line.split('\t'))==2):
       fields = line.split('\t')
       join_key = fields[0]
       join_val = int(fields[1])
       yield (join_key,(join_val,1))

      elif(len(line.split(','))==5):
       fields = line.split(',')
       join_key = fields[0]
       join_val = fields[3]
       yield (join_key, (join_val, 2))

     except:
      pass

    def reducer(self, address, values):
     contracts =[]
     aggregate_file = None

     for value in values:
      if value[1]==1:
       aggregate_file=value[0]
      elif value[1]==2:
       contracts.append(value[0])
      if aggregate_file != None and len(contracts)!=0:
       yield (address, aggregate_file)


if __name__ == "__main__":

 transaction_per_month.run()


# python join_task2.py -r hadoop --output-dir joined_records --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/contracts/ hdfs://andromeda.eecs.qmul.ac.uk/user/kg003/aggregate_transactions/
```

3. Finding top 10

If the input file is divided along 't' and has two fields, the mapper reads it line by line from the preceding job (/user/jmp01/part b joined records/). We get value(None) and key(None) (address, total transaction value).

All pairs are sorted in descending order in the reducer phase using the sort function. Finally, we run through the sorted data 10 times before stopping.

Execution Command: python part_B_top10.py -r hadoop --output-dir part_B_top10 --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/user/jmp01/part_b_joined_records

```
[jmp01@itl211 ~/ecs765/ass1/partB> cat part_B_top10.py
from mrjob.job import MRJob

class PartB(MRJob):

    def mapper(self,_,line):
        try:
            fields = line.split('\t')
            if len(fields) == 2:
                add = fields[0]
                agg = float(fields[1])
                yield (None,(add, agg))
        except:
            pass
    def combiner(self,key,val):
        sorted_values = sorted(val,reverse=True, key=lambda tup:tup[1])

        i=0
        for v in sorted_values:
            yield ("Top",v)
            i += 1
            if i >= 10:
                break

    def reducer(self,key,val):
        sorted_values = sorted(val, reverse=True, key=lambda tup: tup[1])

        i = 0
        for v in sorted_values:
            yield ((v[0], v[1]))
            i+=1
            if i>=10:
                break

if __name__=='__main__':
    PartB.run()

#python part_B_top10.py -r hadoop --output-dir part_B_top10 --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/user/jmp01/part_b_joined_records
```

Output:

```
0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444    8.415510080996593e+25
0x3f5ce5fbfe3e9af3971dd833d26ba9b5c936f0be    5.834331302252918e+25
0x32be343b94f860124dc4fee278fdcbd38c102d88    5.432072061064933e+25
0xfa52274dd61e1643d2205169732f29114bc240b3    4.578748448318936e+25
0x7727e5113d1d161373623e5f49fd568b4f543a9e    4.56206240013507e+25
0x209c4784ab1e8183cf58ca33cb740efbf3fc18ef    4.317035609226244e+25
0x876eabf441b2ee5b5b0554fd502a8e0600950cfa    4.015767887861935e+25
0x6fc82a5fe25a5cdb58bc74600a40a69c065263f8    2.706892158201953e+25
0x2910543af39aba0cd09dbb2d50200b3e800a63d2    2.6386022236908247e+25
0xcafb10ee663f465f9d10588ac44ed20ed608c11e    2.3078610109547e+25
```

PARTC: TOP TEN MOST ACTIVE MINERS

The mapper phase reads the file line by line and divides at ',' resulting in key (miner) and value (mapper) fields (size). The size of each miner is aggregated and yielded in the Combiner/Reducer phase.

Command: python partc.py -r hadoop --output-dir part_c_a --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/part-*.csv

```
jmp01@itl211 ~/ecs765/ass1/partC> cat part_c.py
from mrjob.job import MRJob
import time

class partc(MRJob):

    def mapper(self,_,line):
        try:
            fields = line.split(',')
            if len(fields) == 9:
                size = float(fields[4])
                miner = fields[2]
                yield (fields[2], size)
        except:
            pass

    def combiner(self,key,val):
        yield (key,sum(val))

    def reducer(self,key,val):
        yield (key,sum(val))


if __name__=='__main__':
    partc.run()


# python partc.py -r hadoop --output-dir partc --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/part-*.csv

# job: http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_7045/
```

hadoop fs -copyToLocal part_c_a inputfor10.txt

Top 10

If the input files are split along '\t' and have two fields, the mapper phase reads them line by line. We get value(None) and key(None) (miner, size). All pairs are sorted in descending order in the reducer phase using the sort function. We then repeat through the sorted values 10 times, yielding them, before stopping.

Command: python part_c_top_10.py inputfor10.txt > part_c_top10.txt

```
jmp01@itl211 ~/ecs765/ass1/partC> cat part_c_top10.py
from mrjob.job import MRJob

class top10_miner(MRJob):

 def mapper(self, _,line):
  try:
   fields = line.split('\t')
   if len(fields) == 2:
    address = fields[0]
    aggregate = float(fields[1])
    yield (None,(address,aggregate))
  except:
   pass

 def combiner(self, _, values):
  sorted_values = sorted(values, reverse = True, key = lambda x:x[1])
  for idx,value in enumerate(sorted_values):
   yield ("top", value)
   if idx >= 10:
    break


 def reducer(self, _, values):

  sorted_values = sorted(values, reverse = True, key = lambda x:x[1])
  for idx,value in enumerate(sorted_values):
   yield ("{} - {}".format(value[0],value[1]),None)
   if idx >= 10:
    break


if __name__ == '__main__':
 top10_miner.run()

# python part_c_top_10.py inputfor10.txt > part_c_top10.txt
```

Output:

```
jmp01@itl211 ~/ecs765/ass1> cat  part_c_top10.txt
"\"0xea674fdde714fd979de3edf0f56aa9716b898ec8\" - 23989401188.0"        null
"\"0x829bd824b016326a401d083b33d092293333a830\" - 15010222714.0"        null
"\"0x5a0b54d5dc17e0aadc383d2db43b0a0d3e029c4c\" - 13978859941.0"        null
"\"0x52bc44d5378309ee2abf1539bf71de1b7d7be3b5\" - 10998145387.0"        null
"\"0xb2930b35844a230f00e51431acae96fe543a0347\" - 7842595276.0" null
"\"0x2a65aca4d5fc5b5c859090a6c34d164135398226\" - 3628875680.0" null
"\"0x4bb96091ee9d802ed039c4d1a5f6216f90f81b01\" - 1221833144.0" null
"\"0xf3b9d2c81f2b24b0fa0acaaa865b7d9ced5fc2fb\" - 1152472379.0" null
"\"0x1e9939daaad6924ad004c2560e90804164900341\" - 1080301927.0" null
"\"0x61c808d82a3ac53231750dadc13c777b59310bd9\" - 692942577.0"  null
"\"0xbcdfc35b86bedf72f0cda046a3c16829a2ef41d1\" - 610498611.0"  null
jmp01@itl211 ~/ecs765/ass1>
```

PART D. DATA EXPLORATION

Scam Analysis

The transaction dataset and the scams.json file were utilised as input files. Because we're dealing with 'json,' we've also imported the json library. Only if the input data set has fields with a length of 7 do we proceed (transactions data set). We'll use address as the key and value as the value from the transaction dataset. Similarly, we take address from the scams data set obtained from the for loop, assign value to it, and set 1 to differentiate it from transactions. The initial reduction phase determines whether the values come from transactions or json files. The reducer then sends the key value pair to the mapper, which includes categories and total values. After that, the second reduction step generates cumulative counts for each category.

Command: python scam1.py -r hadoop --output-dir scam_part_1 --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/ hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/scams.json/

```
jmp01@itl211 ~/ecs765/ass1/partD> cat scam1.py
from mrjob.job import MRJob
from mrjob.step import MRStep
import json

class scams(MRJob):
        def mapper1(self, _, lines):
                try:
                        fields = lines.split(",")
                        if len(fields) == 7:
                                address1 = fields[2]
                                value = float(fields[3])
                                yield address1, (value,0)

                        else:
                                line = json.loads(lines)
                                keys = line["result"]

                                for i in keys:
                                        record = line["result"][i]
                                        category = record["category"]
                                        addresses = record["addresses"]

                                        for j in addresses:
                                                yield j, (category,1)

                except:
                        pass

        def reducer1(self, key, values):
                tvalue=0
                category=None

                for k in values:
                        if k[1] == 0:
                                tvalue = tvalue + k[0]
                        else:
                                category = k[0]
                if category is not None:
                        yield category, tvalue

        def mapper2(self,key,value):
                yield(key,value)
        def reducer2(self, key, value):
                yield(key,sum(value))

        def steps(self):
                return [MRStep(mapper = self.mapper1, reducer=self.reducer1), MRStep(mapper = self.mapper2,

if __name__ == '__main__':
        scams.run()
```

Output1:

```
jmp01@itl210 ~/ecs765/ass1/partD> cat scamout1.txt
"Scamming"      3.833616286244431e+22
"Fake ICO"      1.35645756688963e+21
"Phishing"      2.699937579408742e+22
"Scam"  0
```

Part 2

Similarly to the previous section, we import the json library and use the input files 'transactions.json' and'scams.json'. To ensure that we are utilising the correct dataset, we check the length of the fields. The first mapper works in the same way as the previous code.

In the first reducer, we utilise address as a key, which is retrieved using a for loop, and category and status of the scam as the value, which is used to distinguish between them. Furthermore, it checks for the values and adds them if they are 1, else, the reducer will add them to categories and status as they are from the json file containing scam categories and values. The second mapper takes the values from the previous reduction and passes them on to the next reducer, which sums them up to tally the overall number of scams in a category.

Command: python scam2.py -r hadoop --output-dir scam_part_2 --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/ hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/scams.json/

```
jmp01@itl210 ~/ecs765/ass1/partD> cat scam2.py
from mrjob.job import MRJob
from mrjob.step import MRStep
import json

class scams2(MRJob):
 def mapper1(self, _, lines):
  try:
   fields = lines.split(",")
   if len(fields) == 7:
    address1 = fields[2]

    yield address1, (1,0)

   else:
    line = json.loads(lines)
    keys = line["result"]

    for i in keys:
     record = line["result"][i]
     category = record["category"]
     addresses = record["addresses"]
     status = record["status"]

     for j in addresses:
      yield j, (2, category,status)

  except:
   pass

 def reducer1(self, key, values):
  tvalue=0
  category=None
  status = None
  for k in values:
   if k[0] == 1:
    tvalue = tvalue + k[0]
   else:
    category = k[1]
    status = k[2]
  if category is not None and status is not None:
   yield (status,category), tvalue

 def mapper2(self,key,value):
  yield(key,value)
 def reducer2(self, key, value):
  yield(key,sum(value))

 def steps(self):
  return [MRStep(mapper = self.mapper1, reducer=self.reducer1), MRStep(mapper = self.mapper2, reducer = self.reducer2)]

if __name__ == '__main__':
        scams2.run()
```

The output below shows the total number of frauds in various categories.
Output2:

```
jmp01@itl210 ~/ecs765/ass1/partD> cat scamout2.txt
["Active", "Scamming"]  88444
["Inactive", "Phishing"]        22
["Offline", "Fake ICO"] 121
["Offline", "Phishing"] 7022
["Offline", "Scam"]     0
["Suspended", "Phishing"]       11
["Active", "Phishing"]  1584
["Offline", "Scamming"] 24692
["Suspended", "Scamming"]       56
jmp01@itl210 ~/ecs765/ass1/partD>
```

Contract types (identify contract types):
Extraction of features Top 5 transaction datasets with the most transactions
We can give award points to addresses that have the highest amount of transactions, and we can refer to them as loyal users.

2 jobs at the same time:

1. Here, the "address" field is yielded, and the reducer aggregates it to the number of transactions. The output is saved in the " Part_D_contracts_part_1" file.

2. In the second task, we used the sorted method in Python to apply the sorting algorithm to find the top 5 transactions.

Command: python Part_D_contracts_part_1.py -r hadoop --output-dir Part_D_contracts_part_1 --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/

```
jmp01@itl210 ~/ecs765/ass1/partD> cat Part_D_contracts_part_1.py
from mrjob.job import MRJob

class number_of_transactions(MRJob):
        def mapper(self, _, line):
                try:
                        fields = line.split(",")
                        if len(fields) == 7:
                                key = fields[2]
                                yield (key, 1)

                except:
                        pass

        def combiner(self, k, l):
                yield k, sum(l)

        def reducer(self, u, t):
                yield (u, sum(t))

if __name__ == '__main__':
        number_of_transactions.run()

jmp01@itl210 ~/ecs765/ass1/partD>
```

Command: python Part_D_contracts_part_2.py -r hadoop --output-dir Part_D_contracts_part_2 --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/user/jmp01/Part_D_contracts_part_1/

```
jmp01@itl210 ~/ecs765/ass1/partD> cat Part_D_contracts_part_2.py
from mrjob.job import MRJob

class number_of_transactions(MRJob):
        def mapper(self)
                try:
                        fields = line.split()
                        key = fields[0]
                        value = int(fields[1])
                        yield (None, (key, value))

                except:
                        pass

        def reducer(self, _, val):
                sorted_values = sorted(val, reverse = True, key = lambda tup: tup[1])
                j =0
                for i in sprted_values:
                        yield i
                        j += 1
                        if j >= 5:
                                break

if _ _name_ _ == '_ _main_ _'
        number_of_transacions.run()
jmp01@itl210 ~/ecs765/ass1/partD>
```

Output:

```
"\"0x8d12a197cb00d4747a1fe03395095ce2a5cc6819\""  11089018
"\"0x2a0cOdbecc7e4d658f48e01e3fa353f44050c208\""   7160876
"\"0x3f5ce5fbfe3e9af3971dd833d26ba9b5c936f0be\""    5952175
"\"0x174bfa6600bf90c885c7c01c7031389ed1461ab9\""  5797049
"\"0x06012c8cf97bead5deae237070f9587f8e7a266d\""   3995252
```

Is_erc721_contract:

Erc721 is an ethereum token that may be used to identify contracts that adhere to the erc721 contract rules. Smart contracts are another name for them. By simply adding a "if" condition that checks for the truth value in the relevant column of the dataset, we can filter the contract by determining if it is an er721 contract.

Command: python Part_D_Is_erc721_contract.py -r hadoop --output-dir Part_D_Is_erc721_contract --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/contracts/

```
jmp01@itl210 ~/ecs765/ass1/partD> cat Part_D_contracts_part_2.py
from mrjob.job import MRJob

class number_of_transactions(MRJob):
        def mapper(self)
                try:
                        fields = line.split()
                        key = fields[0]
                        value = int(fields[1])
                        yield (None, (key, value))

                except:
                        pass

        def reducer(self, _, val):
                sorted_values = sorted(val, reverse = True, key = lambda tup: tup[1])
                j =0
                for i in sprted_values:
                        yield i
                        j += 1
                        if j >= 5:
                                break

if _ _name_ _ == '_ _main_ _'
        number_of_transacions.run()
jmp01@itl210 ~/ecs765/ass1/partD> nano Part_D_Is_erc721_contract.py
jmp01@itl210 ~/ecs765/ass1/partD> cat Part_D_Is_erc721_contract.py
from mrjob.job import MRJob

class erc721_contract(MRJob):
        def mapper(self, _, line):
                try:
                        fields = line.split(',')
                        if len(fields) == 5:
                                if fields[2] == "true":
                                        yield fields [0], 1

                except:
                        pass

        def reducer(self, u, t):
                yield (u,sum(t))

if __name__ == '__main__':
        erc721_contract.run()

jmp01@itl210 ~/ecs765/ass1/partD>
```

Output:

```
"0x000edf42475e7ceb32f82ee11f9733231a67b2be"     1
"0x00136a57574c805312d4ee875b0bc2e56984f00d"     1
"0x003203f31def0aa63b895b8599c9b81ddce8939b"     1
"0x003ad9c18bc279f40632e7e5de2fd213931215d0"     1
"0x0063f8d3537ec9cd23b08357494d3e0ee63a8f4a"     1
"0x00650ea64d5c226755a6a6976a774a6f2fdf8c13"     1
"0x006bf2f5cc930eed50e14c66605ac95767133bec"     1
"0x00b218fd2db515392f2a2d78b658c913f2eebf2d"     1
"0x00bf70e1ddfb8984d0af9af4b29ad3ec40d4b84e"     1
"0x00d2be2c7e509515cdba64051a643a5220f1f241"     1
"0x0111ac7e9425c891f935c4ce54cf16db7c14b7db"     1
"0x01144cad63687f9d06c46854c810366b28a84d73"     1
"0x0131c575be43586c5346435273035c8709a3fd34"     1
"0x01d8a618ba50232e73b7e0c3254bfa0802a75b56"     1
"0x01f96d3376acae2fc4f9077f03fa9d997797ed5c"     1
"0x0211621fb78c85ee44b12d2f04e2e2f16334a9fe"     1
"0x0213c2c85315299c6074fc4a6d779121284e9f42"     1
```

Outflow addresses:

Simply apply wordcount to the transaction dataset to obtain a unique outflow address. This allows us to retrieve all of the transaction dataset's unique addresses.

Command: python Part_D_outflow.py -r hadoop --output-dir Part_D_outflow --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/

```
[jmp01@itl210 ~/ecs765/ass1/partD> cat Part_D_outflow.py
from mrjob.job import MRJob

class contract(MRJob):
        def mapper(self, _, line):
                try:
                        fields = line.split(",")
                        if len(fields) == 7:
                                yield fields[2], 1

                except:
                        pass

        def reducer(self, k, v):
                yield k, sum(v)

if __name__ == '__main__':
        contract.run()

jmp01@itl210 ~/ecs765/ass1/partD>
```

Output:

```
"0x014d3c3de0696c4e1662cd5e913d833e9181af89" 1
"0x014d3c62d7d8d41936773b186be11af.2d2e6ac16" 1
"0x014d3cd32875eb5fab7cdc383398fe8b8.28181d8" 3
"0x014d3cd43567dac810aba9e0eec6ff084bb0159d" 1
"0x014d3d3cd0d2f6334a7d70b26a72038118875df9" 3
"0x014d3d762587bc8f33fd2eff618cc8505d45a290" 5
"Ox014d3dc71911e397602c3991b830bdal7c5a8a71" 1
"0x014d3de457357bcf065d5897e905cac292bc452c" 2
"Ox014d3e9e5216d77067af4Ofce741e0aaf2bedc90" 1
"Ox014d3eb929c0f449345c87dcde4a4e13f91c7246" 1
"0x014d3ed7a70692491a832aac8.2d.241ed967c8401" 1
"Ox014d3f101261d1b7221492946123017d786a5f6d" 1
"0x014d3f2be1c647a725742cdb917f235bcd0e2074" 64
"0x014d405b8dcd1723eec901b078d481928805d103" 15
"Ox014d40d44c31f7f4f1795bf4eb0Oca385cdfa92d" 1
"0x014d40f86e343c298b03446a8d86f7022af1465e" 1
"0x014d42589bfbde32b84190d7fdbce618c4a4d9c2" 1
"Ox014d42bfcbe9c0Ofb956af2b837caf231ffebd8d" 1
"0x014d43a3b18f0944fee6622a5815ba0321d6440a" 9
"Ox014d4517ecd1e4bb9e7c0c7020del6b9f68c7faa" 1
"0x014d45d33b131b4ecda18c3a4924a7ac15c58220" 1
"0x014d46c6eff0e04a2.278f017671f73ea7ed2ca9d" 1
"Ox014d47e551cac04db2ab4980d81f29cb38acl6b3" 1
"0x014d47fc325050fe37347d96e3a5726284ef664a" 1
"0x014d48483f5b111e397974a75322dbbd1a6b0cce" 1
"0x014d495cd68302745445fda7cd42ce1d9e033f29" 1
"Ox014d49efeb2a7fbca43705b4eaf957a02813df19" 2
"0x014d4a2cd11519245ca78d30b68a5e935e518e12" 22
"Ox014d4a2fedlaabe08e82edfa28343c94d3800442" 1
"Ox014d4b916c71cdcb0122d8c5605ff7a1d500a518" 1
"Ox014d4be25067f7dde1187b5dffa9c2ace3e395d6" 81
"0x014d4bf25039f4f47b74ea73db42399b97288ca2" 2
"Ox014d4bfal886b969ede482fa7af5045f333dd59b" 2
"Ox014d4cf0721be83114alac9d321664cc3084cd3c" 2
```

MISCELLANEOUS

Gas Guzzlers:

Using MapReduce, obtain the average Gas Price and Average Gas Limit for each month in the dataset by running the following two tasks.

GasPrice.py

```
jmp01@itl210 ~/ecs765/ass1/partD> cat GasPrice.py
from mrjob.job import MRJob
import time

class Gas(MRJob):

    def mapper(self,_,line):
        try:
            fields = line.split(',')
            val = float(fields[5])
            date  = time.localtime(float(fields[6]))
            if len(fields) == 7:
                yield ((date.tm_mon,date.tm_year),(1,val))    #Number of transaction

        except:
            pass

    def combiner(self,key,val):
        count = 0
        total = 0
        for v in val:
            count+=v[0]
            total+=v[1]
        yield (key,(count,total))

    def reducer(self,key,val):
        count = 0
        total = 0
        for v in val:
            count+=v[0]
            total+=v[1]
        yield (key,(total/count))


if __name__=='__main__':
    Gas.run()
jmp01@itl210 ~/ecs765/ass1/partD>
```

GasLimit.py

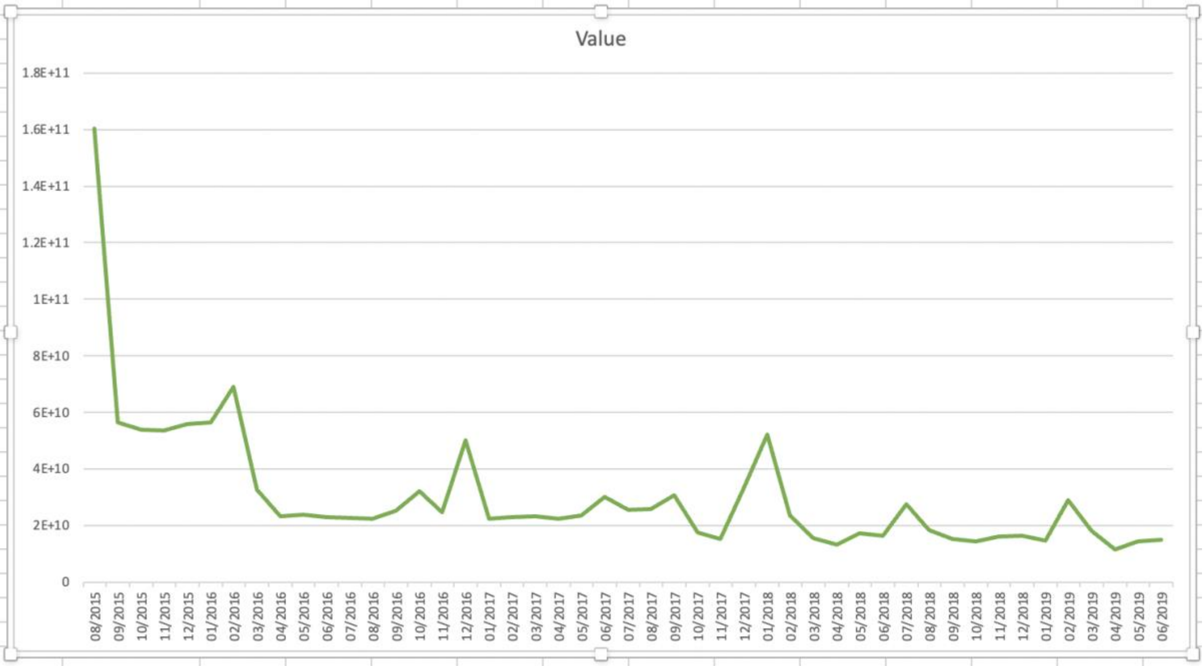
```
jmp01@itl210 ~/ecs765/ass1/partD> cat GasLimit.py
from mrjob.job import MRJob
import time

class Gas_limit(MRJob):

    def mapper(self,_,line):
        try:
            fields = line.split(',')
            val = float(fields[4])
            date  = time.localtime(float(fields[6]))
            if len(fields) == 7:
                yield ((date.tm_mon,date.tm_year),(1,val))    #Number of transaction

        except:
            pass

    def combiner(self,key,val):
        count = 0
        total = 0
        for v in val:
            count+=v[0]
            total+=v[1]
        yield (key,(count,total))

    def reducer(self,key,val):
        count = 0
        total = 0
        for v in val:
            count+=v[0]
            total+=v[1]
        yield (key,(total/count))


if __name__=='__main__':
    Gas_limit.run()

jmp01@itl210 ~/ecs765/ass1/partD>
```

Output:

```
jmp01@itl210 ~/ecs765/ass1/partD> cat Gas_Price_output.txt
[1, 2017]       22507570807.719795
[1, 2019]       14611816445.785261
[10, 2015]      53898497955.07804
[10, 2017]      17509171844.77064
[11, 2016]      24634294365.279037
[11, 2018]      16034859008.681648
[12, 2015]      55899526672.35498
[12, 2017]      33423472930.407898
[2, 2016]       69180681134.38954
[2, 2018]       23636574203.828873
[3, 2017]       23232083087.910202
[3, 2019]       18091340267.2465
[4, 2016]       23359978331.676765
[4, 2018]       13149523170.45877
[5, 2017]       23568661138.035046
[5, 2019]       14480574461.419468
[6, 2016]       23021831286.57488
[6, 2018]       16536952425.540333
[7, 2017]       25463022668.143833
[8, 2016]       22407628763.365406
[8, 2018]       18478650928.737164
[9, 2015]       56512934320.01927
[9, 2017]       30676555381.728436
[1, 2016]       56596270931.316185
[1, 2018]       52106060636.844185
[10, 2016]      32113146198.891758
[10, 2018]      14527572489.59486
[11, 2015]      53607614201.79755
[11, 2017]      15312465314.693539
[12, 2016]      50318068074.687996
[12, 2018]      16338844844.014513
[2, 2017]       23047230327.254387
[2, 2019]       28940599438.1487
[3, 2016]       32805967466.947445
[3, 2018]       15554999714.874079
[4, 2017]       22357075153.737774
[4, 2019]       11573133401.384796
[5, 2016]       23747073761.79113
[5, 2018]       17414613148.43696
[6, 2017]       30201664896.657127
[6, 2019]       15067557451.33386
[7, 2016]       22619213302.825947
[7, 2018]       27520561081.0211
[8, 2015]       160356354969.0592
[8, 2017]       25903650367.89669
[9, 2016]       25262249340.11566
[9, 2018]       15208159827.250359

jmp01@itl210 ~/ecs765/ass1/partD>
```

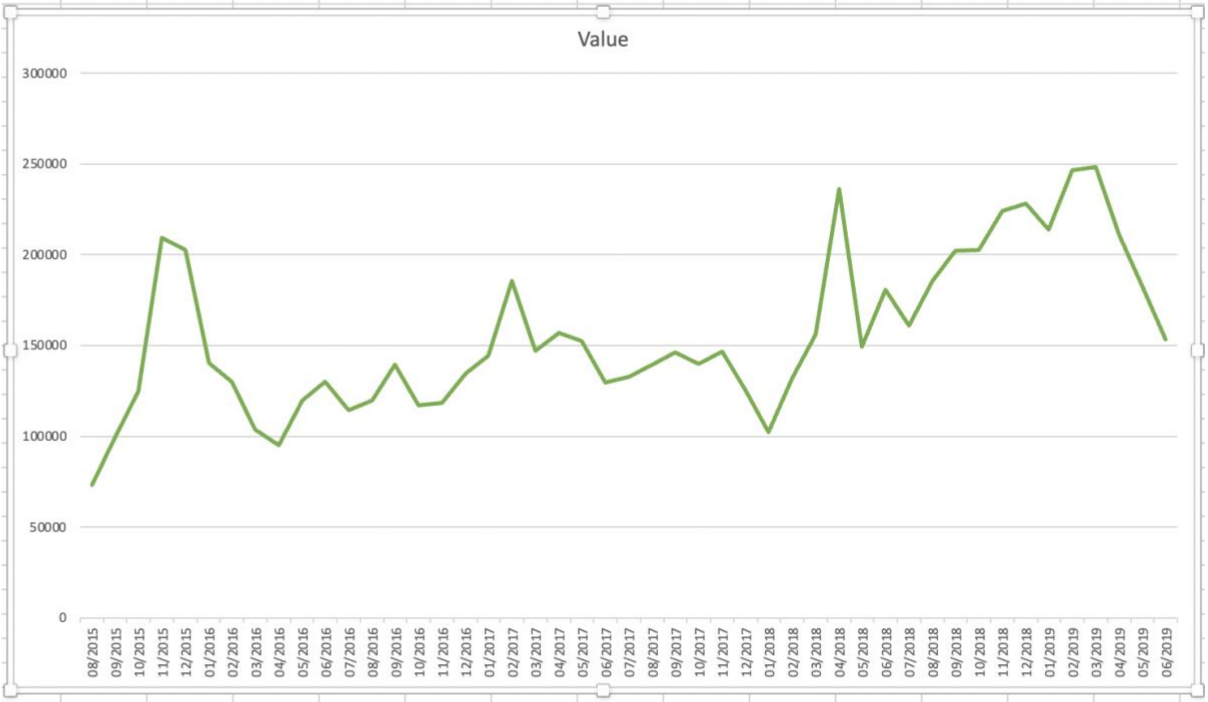
```
jmp01@itl210 ~/ecs765/ass1/partD> cat Gas_Limit_output.txt
[1, 2017]       144585.8323444452
[1, 2019]       213715.9605048934
[10, 2015]      124898.76345065859
[10, 2017]      139870.05913735816
[11, 2016]      118474.20698056064
[11, 2018]      224208.95181702235
[12, 2015]      202883.632316504
[12, 2017]      125269.51047898784
[2, 2016]       130175.52141758327
[2, 2018]       131824.6970378884
[3, 2017]       147228.19778654756
[3, 2019]       248217.42927149552
[4, 2016]       95083.47755021007
[4, 2018]       236204.97904008476
[5, 2017]       152449.08294623252
[5, 2019]       183094.5693197547
[6, 2016]       130112.21626908527
[6, 2018]       180795.02085380582
[7, 2017]       132790.34469405047
[8, 2016]       119958.22676283357
[8, 2018]       185752.46188363043
[9, 2015]       99880.37666969116
[9, 2017]       146371.34922749794
[1, 2016]       140463.7812759377
[1, 2018]       102167.12354317823
[10, 2016]      117078.86856704934
[10, 2018]      202550.24404083
[11, 2015]      209520.94340378218
[11, 2017]      146613.7212948582
[12, 2016]      134463.8302327048
[12, 2018]      228197.0665342265
[2, 2017]       185771.26611789103
[2, 2019]       246776.93062665823
[3, 2016]       103585.96288806947
[3, 2018]       156114.3435117928
[4, 2017]       156866.61651822302
[4, 2019]       210658.60478696763
[5, 2016]       119737.9383389679
[5, 2018]       149300.75953211795
[6, 2017]       129488.39504908556
[6, 2019]       153578.42011967467
[7, 2016]       114490.23118436741
[7, 2018]       160946.32119566188
[8, 2015]       73133.53050098693
[8, 2017]       139510.6310349726
[9, 2016]       139595.43180709172
[9, 2018]       202264.51543298518

jmp01@itl210 ~/ecs765/ass1/partD>
```

Bar charts & Observation
Gas_Price vs Time :



According to the graph above, the average price of gas has reduced from 2015 to 2019, and we can also see that each year, prices have risen in the last few months or the first few months of the year.

Contract_gas vs Time :



Observation from the graph above - Despite some ups and downs, there has been a general pattern of steadily increasing gas requirements (from 80000 to 250000 approx)