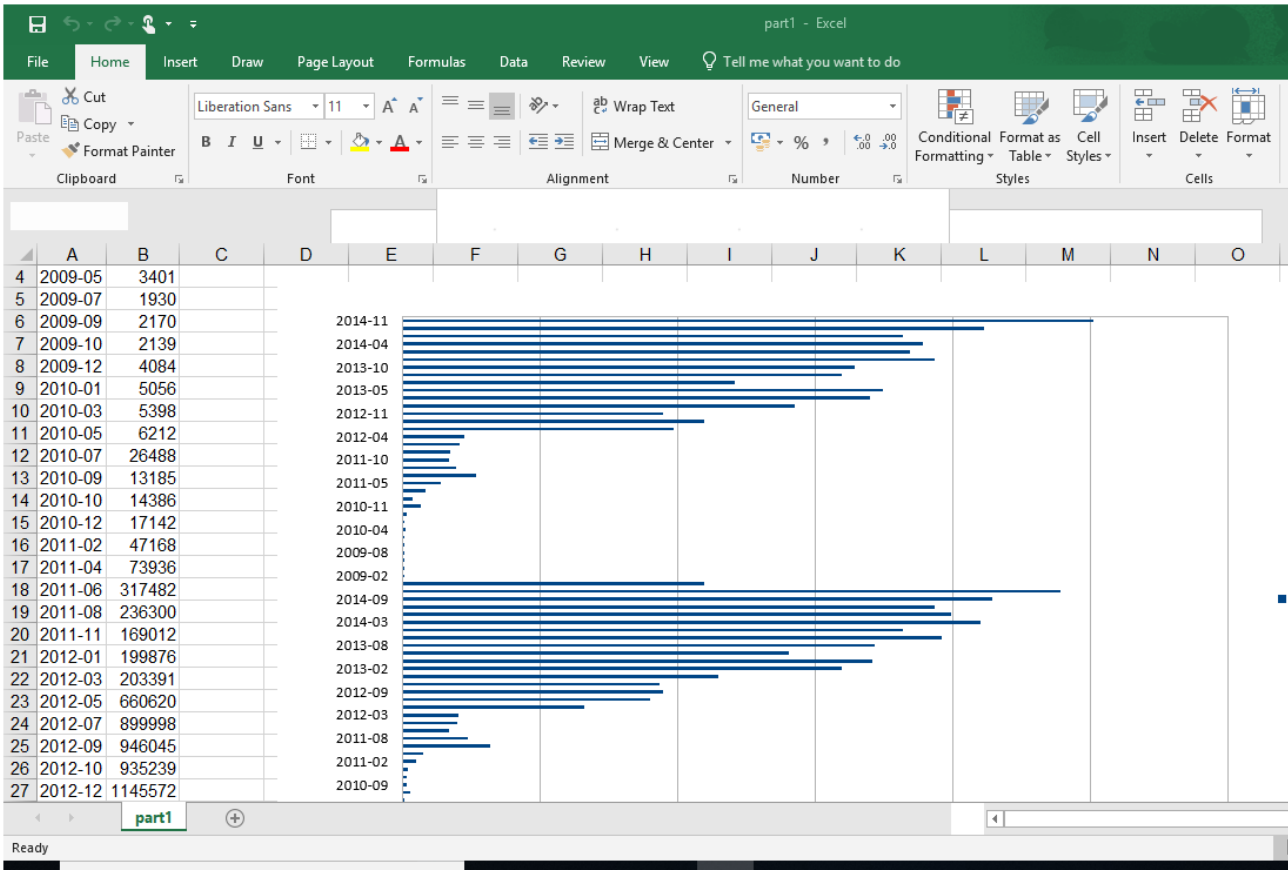
Note :The codes and the outputs obtained from the codes are included .

Part A

1q . observation

- B/w year 2009-02 to 2010 -11 we see that the utilization for bitcoin was very less.
- The bitcon trend peaked at 2014-11.
- Bit coin trend was better during the year 2013 and 2014.



Code explanation,

Mapper

- Line.split(‘,’) will split the line whenever a comma is encountered.
- If the length of the field = 5 . **strftime** function is used to format the output of a date.
- We yield day , 1

Reducer

- We calculate sum of counts
- We yield the sum with day

```
if __name__ == '__main__':
   Part1.run()    Part1.run()
```

The above code runs the mapper and reducer functions.

Part B

Part2_1

We use the vout.csv file for the first part.

- We are looking for a particular wallet id .If the wallet id matches we print the fields from that line .
- We repeat this process for the rest of the data set.
- We save the data in a text file .

Part 2_2

We use vin.csv for this part along with output txt file from previous part

- We make use of sector table to combine 2 data tables.
- In this stage we use 2 mappers
    - First mapper we are finding the key and in the second mapper we are finding the transaction id and match it with key.

Part 2_3

We use vout.csv for this part along with output txt file from previous part

- As v_in only specifies the previous transaction where the bitcoins originated from, this second output must be re-joined with /data/bitcoin/vout.csv.
- We once again make use of sector tables to combine data from 2 different tables.
- In first mapper we are finding [key+val]= val
- If hash + n is present in val we use strip function.

Part2_4

- We use the output file from part 3 and find the top 10 from that file.
- The top 10 code is implemented in the reducer.
  To get top 10 we first sort the values with reverse = true so that it will be arranged in descending order. The output is then formatted.

Part C

Ransomware

I could not find any meaningful data set on the internet so I made the data set myself.

I Got the data from this link below.

https://www.blockchain.com/btc/address/12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw?sort=0

ID of wallet receiving ransom: 12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw

It contains 1839 entries. Among 1839 entries there are entries for a wallet that was used to receive ransom money by installing ransomware on systems by hacking them and demanding money in the form of bitcoins.

An example for ransomware is "wannaCry Ransomware".

I implemented the filtering as we did in part B to extract transaction data for a particular wallet id of our wish. in this case wallet id which has received ransom bitcoins.

Then after filtering I tried to calculate the sum for the entire column. I tried to alter te top 10 code to achieve that in the reducer for-loop but I failed. it is still giving out output of a top10.

That however helped me see which 10 highest ransom payments received by that wallet.

I found 126 entries for the ransom wallet collecting a total of **19.39642985** bitcoins.

What happens to bitcoins after transaction.?

transaction gets included in a "block" which gets attached to the previous block – hence the term "blockchain." Transactions can't be undone or tampered by anyone.

Code explanation for ransomware

rw.py

- We are looking for a particular wallet id .If the wallet id matches we print the fields from that line .
- We repeat this process for the rest of the data set.
- We save the data in a text file .

Rw2.py

- Calculate sum of fields[1] which will give the total extorted amount.

Part C Spark.

Spark	Map/Reduce
Spark doesn't have its own distributed filesystem, but can use HDFS	**MapReduce** is strictly disk-based
Spark uses memory and can use disk for processing	MapReduce writes intermediate results back to Disk and reads it back
Since spark uses memory which is faster than regular storage and will have better performance	
Spark is designed to transform data In-memory and hence reduces time for disk I/O.	
Drawbacks As memory is volatile a power failure means the process is lost	Advantages Since it is processed on the disk which is non-volatile there is no loss of data in case of power failure
The problem arises if there is insufficient memory	Not the case with map/reduce as the process runs on a disk.

The spark program took an average time of 3 mins and 34 seconds.

The above result is far from accurate as ITL systems run the code much faster than my laptop.

And also the load on the machines at a given time play its role in how fast it is processed but I see that spark was lot faster than map/reduce. I do not have times for map/reduce jobs.