HW2: Exploring Bitcoin transactions - Code
Ramandeep Singh
8019-7991

Below code can be run in Jupyter notebook. Installation details for the same is mentioned in the report document.

```
import pandas as pd
from pandasql import sqldf

def pysqldef(q):
   return sqldf(q,globals())
```

Part 1: Transactions analysis

1.
```
import pandas as pd
block_hash = pd.read_csv ('bh.csv', delimiter = '\t')
address = pd.read_csv ('addresses.csv', delimiter = ‘\t’)

pysqldef ("select sum(n_txs) from block_hash”)

pysqldef ("select count(addrID) from address”)
```

2.
```
import pandas as pd
address = pd.read_csv ('addresses.csv', delimiter = '\t')
transaction_in = pd.read_csv ('txin.csv', delimiter = '\t')
transaction_out = pd.read_csv ('txout.csv', delimiter = ‘\t')

pysqldef ( "select sval, address FROM (select row_number() over (ORDER BY
(IFNULL(tout.sout,0)-IFNULL(tin.sin,0)) DESC)  row_num, (IFNULL(tout.sout,0)-
IFNULL(tin.sin,0)) sval, ad.address, ad.addrid from address ad, (select
IFNULL(sum(sum),0) sout, addrID from transaction_out group by addrID) tout NATURAL
LEFT OUTER JOIN (select IFNULL(sum(sum),0) sin, addrID from transaction_in group
by addrID) tin where tout.addrID = ad.addrID) where row_num = 1" )
```

3.
```
import pandas as pd
address = pd.read_csv ('addresses.csv', delimiter = '\t')
transaction_in = pd.read_csv ('txin.csv', delimiter = '\t')
transaction_out = pd.read_csv ('txout.csv', delimiter = '\t')

pysqldef ( "select  ROUND(AVG (IFNULL(tout.sout,0)-IFNULL(tin.sin,0)), 0) Balance from (select
IFNULL(sum(sum),0) sout, addrID from transaction_out group by addrID) tout NATURAL LEFT
OUTER JOIN (select IFNULL(sum(sum),0) sin, addrID from transaction_in group by addrID)
tin" )
```

4.
```
import pandas as pd
```

```
address = pd.read_csv ('addresses.csv', delimiter = '\t')
transaction_in = pd.read_csv ('txin.csv', delimiter = '\t')
transaction_out = pd.read_csv ('txout.csv', delimiter = '\t')

pysqldef ( "select AVG(incnt) from (select count(distinct txid) incnt from transaction_in group by
addrid)" )

pysqldef ( "select avg(outcnt) from (select count(distinct txid) outcnt from transaction_out
group by addrid)" )

pysqldef ( "select AVG(tout.outcnt + tin.incnt) from (select count(distinct txid) outcnt, addrid
from transaction_out group by addrid) tout, (select count(distinct txid) incnt , addrid from
transaction_in group by addrid) tin where tout.addrID = tin.addrID" )
```

5. import pandas as pd

```
transaction_hash = pd.read_csv ('txh.csv', delimiter = '\t')
transaction_in = pd.read_csv ('txin.csv', delimiter = '\t')
transaction_out = pd.read_csv ('txout.csv', delimiter = ‘\t')

pysqldef ( " select txgrp.txcnt, txhash.hash FROM transaction_hash txhash, (select
row_number() over (ORDER BY count(txid) DESC)  row_num, count(txid) txcnt, txid from
transaction_in group by txid) txgrp where txgrp.txid = txhash.txid and row_num = 1  " )
```

6. import pandas as pd

```
transaction_hash = pd.read_csv ('txh.csv', delimiter = '\t')
transaction_in = pd.read_csv ('txin.csv', delimiter = '\t')
transaction_out = pd.read_csv ('txout.csv', delimiter = ‘\t')

pysqldef ( " select substr((sout/cout) * 1.0, 1, instr((sout/cout) * 1.0, '.') + 1) avg from (select
sum(sum) sout,count(distinct txid) cout from transaction_out)  " )
```

7. import pandas as pd

```
transaction_overview = pd.read_csv ('tx.csv', delimiter = ‘\t')

pysqldef ( "select count(*) 'Coinbase Transaction' from transaction_overview where n_inputs =
0" )
```

8. import pandas as pd

```
transaction_overview = pd.read_csv ('tx.csv', delimiter = ‘\t')

pysqldef ( "select AVG(incnt) from (select count(distinct txid) incnt from transaction_overview
group by blockid)" )
```

Part 2: Address de-anonymization

1.
```
import pandas as pd
joint_serial = pd.read_csv ('addr_sccs_joint_serial.csv', delimiter = ‘\t’)

pysqldef ( "select count(distinct userid) from joint_serial" )
```

2.1.
```
import pandas as pd

joint_serial = pd.read_csv ('addr_sccs_joint_serial.csv', delimiter = '\t')
transaction_in = pd.read_csv ('txin.csv', delimiter = '\t')
transaction_out = pd.read_csv ('txout.csv', delimiter = '\t')

pysqldef ( "select sumsval, userid from (select row_number() over (ORDER BY
(sum(xyz.sval)) DESC)  row_num, sum(xyz.sval) sumsval, xyz.userid from (select
addr.addrid, addr.userid, abc.sval from joint_serial addr, (select (IFNULL(tout.sout,0)-
IFNULL(tin.sin,0)) sval, tout.addrID addrid from (select IFNULL(sum(sum),0) sout,
addrID from transaction_out group by addrID) tout NATURAL LEFT OUTER JOIN
(select IFNULL(sum(sum),0) sin, addrID from transaction_in group by addrID) tin ) abc
where addr.addrid = abc.addrid) xyz group by xyz.userid) where row_num = 1" )
```

2.2.
```
import pandas as pd
joint_serial = pd.read_csv ('joint_serial.csv', delimiter = '\t')
transaction_in = pd.read_csv ('txin.csv', delimiter = '\t')
transaction_out = pd.read_csv ('txout.csv', delimiter = '\t')

pysqldef ( "select ROUND (AVG(ssval),0) from (select sum(xyz.sval) ssval FROM (select
addr.addrid, addr.userid, abc.sval from joint_serial addr, (select (IFNULL(tout.sout,0)-
IFNULL(tin.sin,0)) sval, tout.addrID addrid from (select IFNULL(sum(sum),0) sout,
addrID from transaction_out group by addrID) tout NATURAL LEFT OUTER JOIN
(select IFNULL(sum(sum),0) sin, addrID from transaction_in group by addrID) tin) abc
where addr.addrid = abc.addrid) xyz group by xyz.userid)" )
```

2.3.
```
import pandas as pd

joint_serial = pd.read_csv ('joint_serial.csv', delimiter = '\t')
transaction_in = pd.read_csv ('txin.csv', delimiter = '\t')
transaction_out = pd.read_csv ('txout.csv', delimiter = '\t')

pysqldef ( "select avg(abc.incnt) from joint_serial addr, (select count(distinct tin.txid)
incnt ,tin.addrid from transaction_in tin group by tin.addrid) abc where abc.addrid =
addr.addrid" )

pysqldef ( "select avg(abc.outcnt) from joint_serial addr, (select count(distinct tout.txid)
outcnt ,tout.addrid from transaction_out tout group by tout.addrid) abc where
abc.addrid = addr.addrid" )
```

```
pysqldef ( "select avg(abc.outcnt + abc.incnt) from joint_serial addr, ((select
count(distinct tout.txid) outcnt ,tout.addrid from transaction_out tout group by
tout.addrid) NATURAL LEFT OUTER JOIN (select count(distinct tin.txid)
incnt ,tin.addrid from transaction_in tin group by tin.addrid)) abc where abc.addrid =
addr.addrid" )
```

3. import pandas as pd

```
joint_serial = pd.read_csv ('addr_sccs_joint_serial.csv', delimiter = '\t')
transaction_in = pd.read_csv ('txin.csv', delimiter = '\t')
transaction_out = pd.read_csv ('txout.csv', delimiter = '\t')
transaction_hash = pd.read_csv ('txh.csv', delimiter = ‘\t')

pysqldef (  "select hash from transaction_hash where txid = (select txid from (select
row_number() over (ORDER BY (sum) DESC)  row_num, txid, sum, addrid from transaction_in
where addrid IN (select addrid from joint_serial where userid = 12461805)) where row_num =
1)" )
```