

# CIS 6930: Blockchain: Optimization and Applications

## Homework 2

Venkata Sindhu Kandula (UF ID: 1914 5414)

### Contents:

- I. *Tools Used*
- II. *Data Trimming*
- III. *Installation Setup and Table creation*
- IV. *PART-1 Solutions*
- V. *PART-2 Solutions*

### I. Tools Used

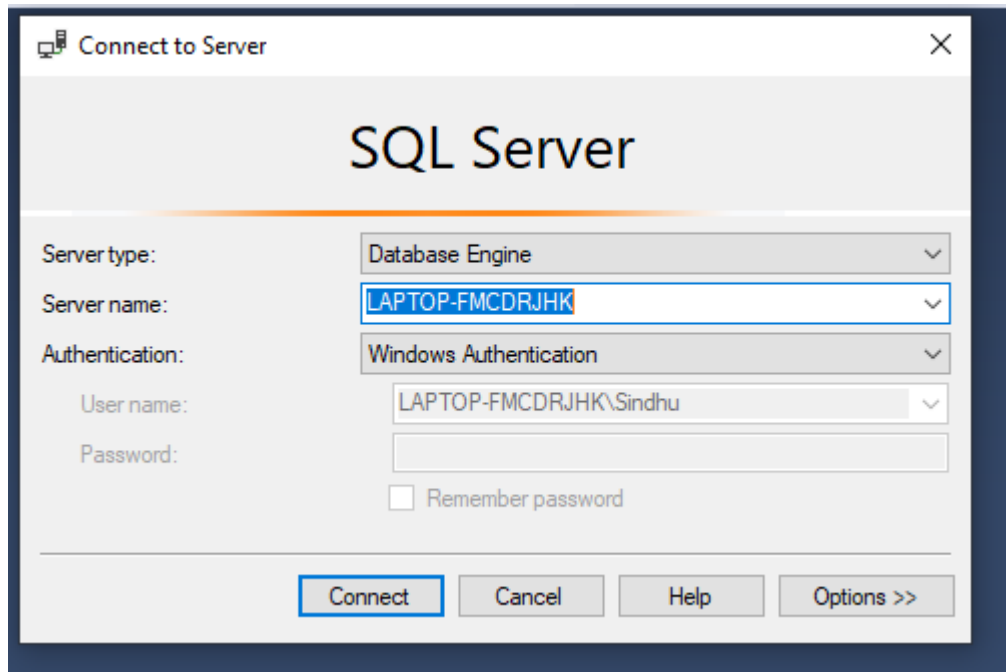
- Microsoft SQL Server Management Studio (MS SQL Server)
- SQL Server 2019 Configuration Manager (SSCM)
- For data trimming GITHUB links - <https://github.com/dkondor/bitcoin/tree/0.16>, <https://github.com/dkondor/scs32s>
- 7zip to extract the dat files
- Windows Power shell script - for importing data from csv files to SQL Server tables
- SQL Queries to perform on the data

### II. Data Trimming

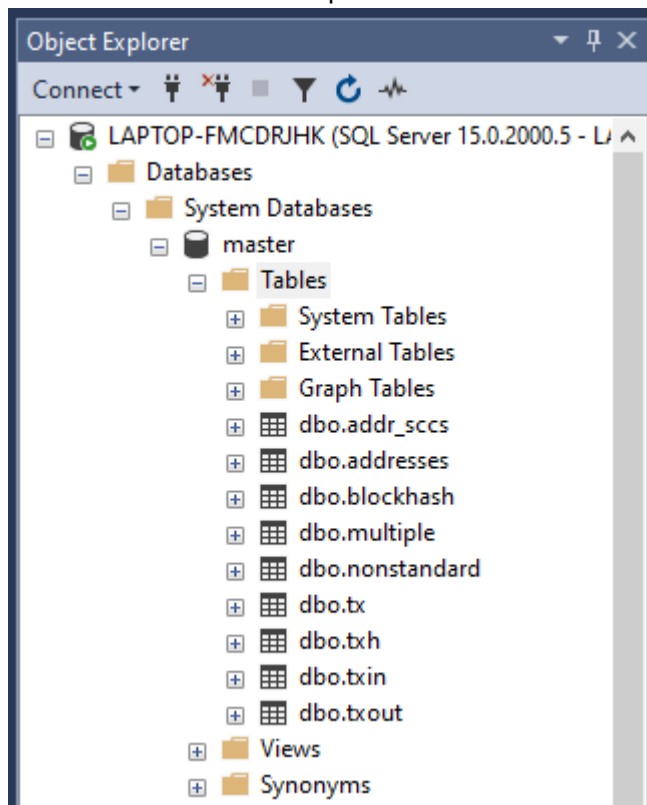
- Downloaded <https://github.com/dkondor/bitcoin/tree/0.16>, installed dependencies and followed the steps mentioned in README.md file.
- Run bitcoind script with argument inputs -DUMP, -DUMP\_outdir, -DUMP\_bmax=212576 (blocks specified for trimmed data – 0 to 212575), -DUMP\_txout, -DUMP\_txin, -DUMP\_tx, -DUMP\_txx, -DUMP\_bh, -DUMP\_missing, -DUMP\_multiple, -DUMP\_nonstandard, -DUMP\_addresses to get dat.gz files of tx, txin, txout, bh, txh, multiple, nonstandard, addresses
- Now use 7zip to extract the dat files
- For getting addr\_scs file, downloaded <https://github.com/dkondor/scs32s> code and performed the steps in readme with the trimmed txin inputs and generated addr\_scs.dat file which has the user ids generated after joint control and their corresponding addresses.
- Convert the all dat files to csv
- Place all the csv files in a folder (Folder I have used: C:\Users\Sindhu\Documents\UF\SEM2\blockchain\hw2\data\)
- Place the power shell script(attached in the source code zip) also in the same directory as csv files

### III. Installation Setup and Table Creation

- Install MS SQL Server and SSCM on local machine
- Open Microsoft SQL Server Management Studio and connect to the local machine



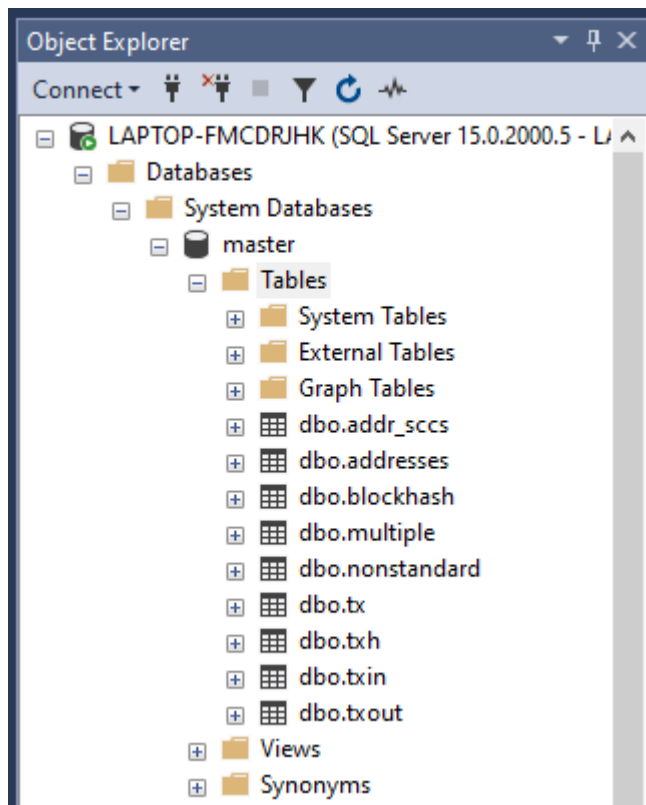
- Create tables in the below path:



- Commands used for table creation:

- `create table addresses (ADDR_ID int, ADDRESS varchar(100));`
- `create table tx(TX_ID int, BLOCK_ID int, N_INPUTS int, N_OUTPUTS int);`

- `create table txin(TX_ID int, INPUT_SEQ int, PREVIOUS_TX_ID int, PREV_OUTPUT_SEQ int, ADDR_ID int, SUM_IN bigint);`
- `create table txout(TX_ID int, OUTPUT_SEQ int, ADDR_ID int, SUM_OUT bigint);`
- `create table multiple(TX_ID int, OUTPUT_SEQ int, ADDR_ID int);`
- `create table nonstandard( TX_ID int, OUTPUT_SEQ int);`
- `create table blockhash (BLOCK_ID int, HASH_VALUE varchar(100), BLOCK_TIMESTAMP varchar(100), N_TXS int);`
- `create table txh (TX_ID int, HASH_VALUE varchar(100));`
- `create table addr_sccs( ADDR_ID int, USER_IDG int);`
- *Tables created:*



- Now open windows PowerShell as administrator and navigate to the directory where all the csv files and ps1 script are present.



ps1-ttemp.ps1

- Edit the below \$Database variables in the ps1 file and run the file (attached in source folder) in windows PowerShell.

`$sqlserver = "LAPTOP-FMCDRJHK"`

`$database = "master"`

`$table = "addr_sccs"`

`$csvfile = "C:\Users\Sindhu\Documents\UF\SEM2\blockchain\hw2\data\addr_sccs.csv"` – change the values everytime you run for updating table. Do this for all the csv files.

- The values are populated in the tables of SQL server now in the local machine.

#### IV. PART-1 Solutions

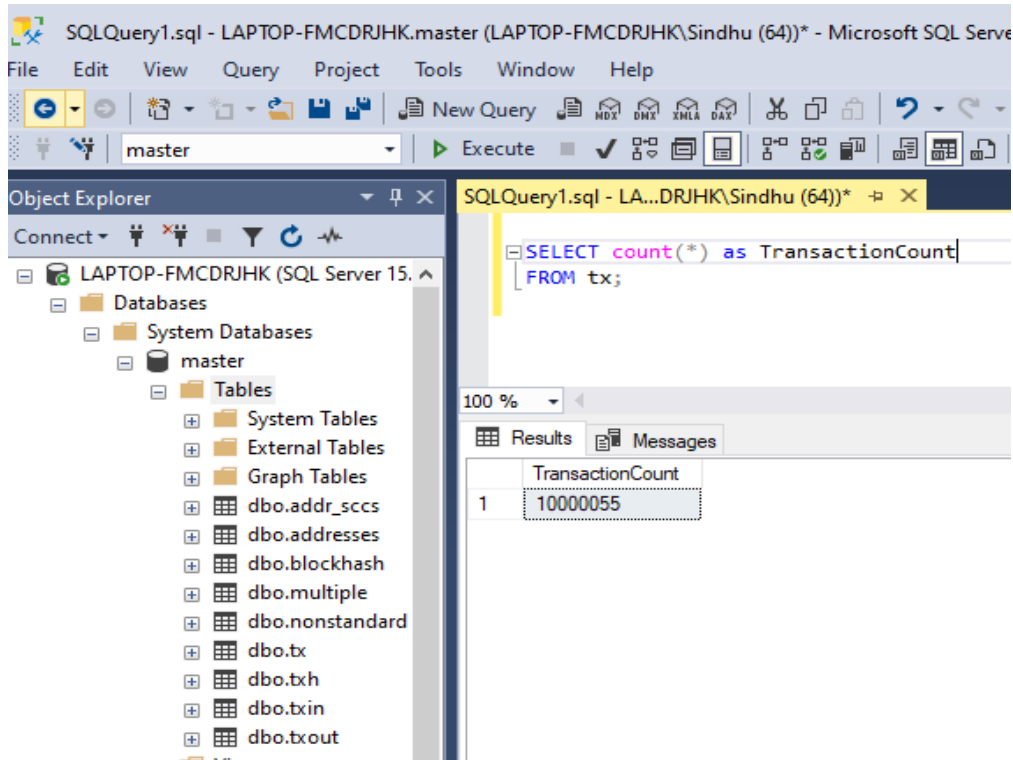
##### ➤ **Question 1:**

**Number of Transactions in the dataset: 10000055**

**Number of Addresses in the dataset: 8385065**

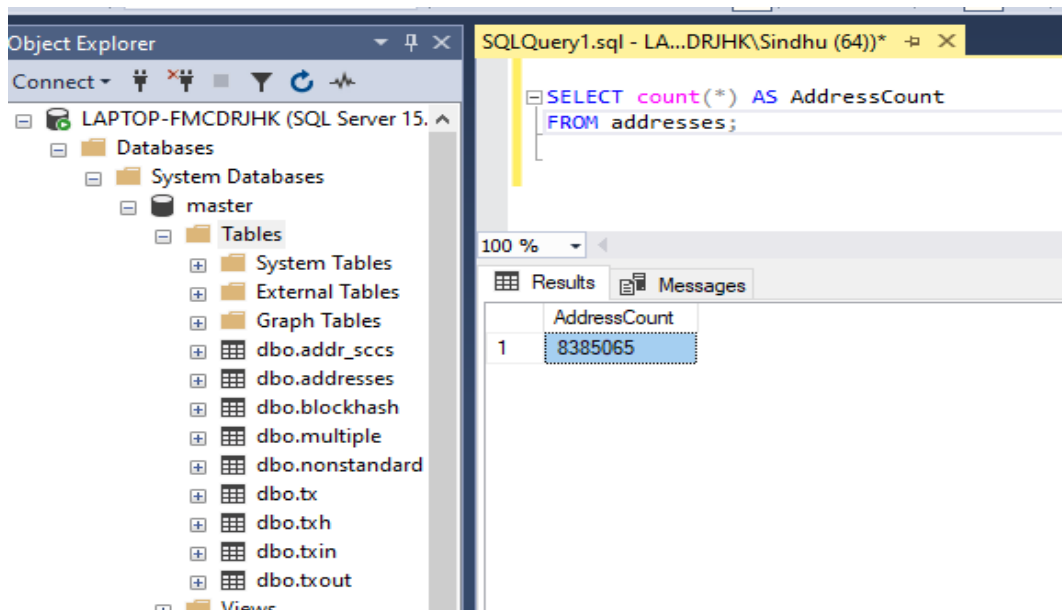
SQL Query: `SELECT count(*) as TransactionCount FROM tx;`

Screenshot:



SQL Query: `SELECT count(*) AS AddressCount FROM addresses;`

Screenshot:



➤ **Question 2:**

What is the Bitcoin address that is holding the greatest amount of bitcoins - 1083442  
How much is that exactly – 11111100000000satoshi

**SQL Query:**

```
SELECT SUM([SUM_OUT]) UTXO, ADDR_ID FROM [master].[dbo].[txout] TXOUT WHERE
TX_ID NOT IN (
    SELECT PREVIOUS_TX_ID
    FROM txin
    WHERE PREV_OUTPUT_SEQ = TXOUT.OUTPUT_SEQ
    AND PREVIOUS_TX_ID = TXOUT.TX_ID
) GROUP BY ADDR_ID
ORDER BY UTXO DESC;
```

**Screenshots:**

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure for 'master' (dbo). The central pane shows the SQL query being executed. The bottom pane displays the results of the query, which is a table with two columns: UTXO and ADDR\_ID. The first row shows the highest UTXO value (11111100000000) for address 1083442.

	UTXO	ADDR_ID
1	11111100000000	1083442
2	7995704110000	268522
3	6010245309814	6850395
4	5690000000000	3804057
5	5300004110000	811394
6	5013004110000	2770907
7	5000004110000	297910
8	5000004110000	2518477
9	4745804110000	3639552
10	4451204110000	3350132

➤ **Question 3:**

What is the average balance per address – 125990554.986894satoshi

**SQL Query:**

```
SELECT convert(FLOAT, p.TOTALBAL) / q.addressses FROM (
    SELECT SUM(T) AS TOTALBAL
    FROM (
        SELECT SUM([SUM_OUT]) T
        FROM [master].[dbo].[txout] r
        WHERE TX_ID NOT IN (
            SELECT PREVIOUS_TX_ID
            FROM txin
            WHERE PREV_OUTPUT_SEQ = r.OUTPUT_SEQ
            AND PREVIOUS_TX_ID = r.TX_ID
        )
    ) s
) p
,(
    SELECT count(*) AS addressses FROM addresses
```

) q;

#### Screenshots:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the 'master' database with tables 'dbo.txin' and 'dbo.txout'. The right pane shows a SQL query in the 'SQLQuery1.sql' file. The query calculates the average number of transactions per address by dividing the total balance by the number of addresses. The results pane at the bottom shows a single row with the value 125990554.986894.

```
SELECT convert(FLOAT, p.TOTALBAL) / q.addressses
FROM (
    SELECT SUM(T) AS TOTALBAL
    FROM (
        SELECT SUM([SUM_OUT]) T
        FROM [master].[dbo].[txout] r
        WHERE TX_ID NOT IN (
            SELECT PREVIOUS_TX_ID
            FROM txin
            WHERE PREV_OUTPUT_SEQ = r.OUTPUT_SEQ
            AND PREVIOUS_TX_ID = r.TX_ID
        )
    ) s
) p
, (
    SELECT count(*) AS addressses
    FROM addresses
) q;
```

(No column name)
1 125990554.986894

#### ➤ Question 4:

What is the average number of input and output transactions per address?

Input avg - 2.77479148939215

Output Avg - 2.4052948903795

What is the average number of transactions per address (including both inputs and outputs)?

#### SQL Queries:

```
SELECT convert(FLOAT, p.outTrans + q.inTrans) / r.addressses AS AvgTrans FROM (
SELECT count(*) outTrans FROM [master].[dbo].[txout] ) p,( SELECT count(*)
inTrans FROM [master].[dbo].[txin] ) q,( SELECT count(*) addresses FROM
[master].[dbo].[addresses] ) r;
```

#### Screenshots:

SQLQuery1.sql - LA...DRJHK\Sindhu (56))\*

```
SELECT convert(FLOAT, p.outTrans) / q.addresses AS inputTransactionAvg
FROM (
    SELECT count(*) outTrans
    FROM [master].[dbo].[txout]
) p
,(
    SELECT count(*) addresses
    FROM [master].[dbo].[addresses]
) q;
```

100 %

Results Messages

	inputTransactionAvg
1	2.77479148939215

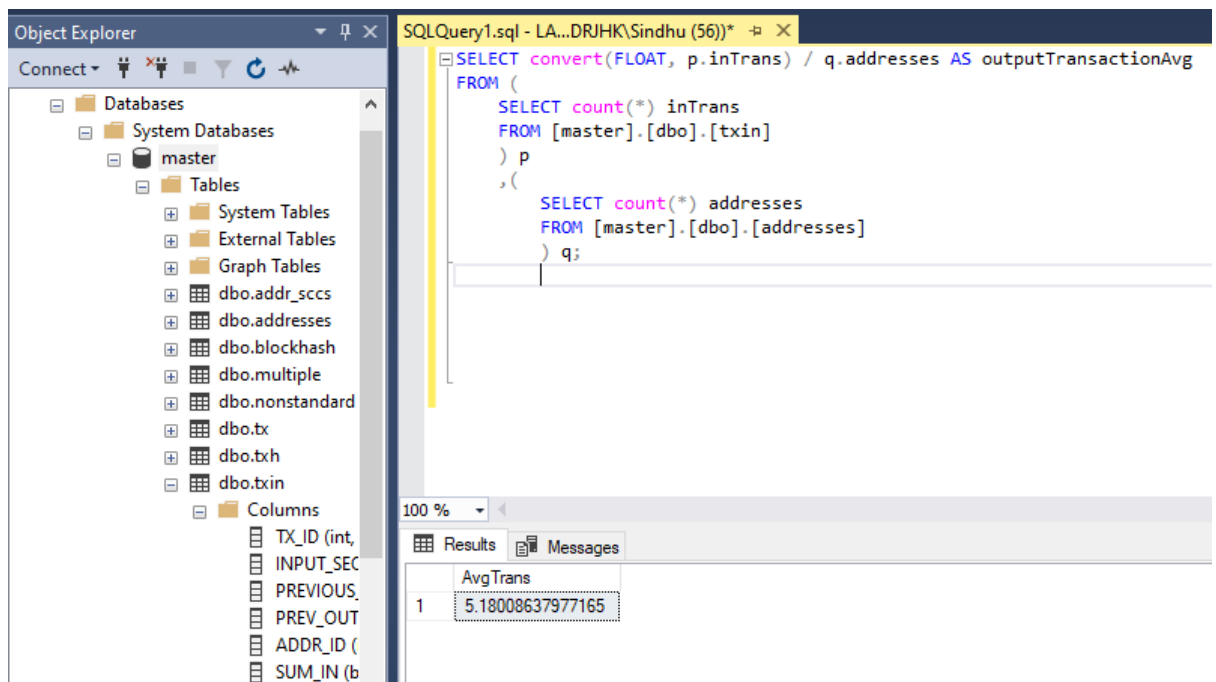
SQLQuery1.sql - LA...DRJHK\Sindhu (56))\*

```
SELECT convert(FLOAT, p.inTrans) / q.addresses AS outputTransactionAvg
FROM (
    SELECT count(*) inTrans
    FROM [master].[dbo].[txin]
) p
,(
    SELECT count(*) addresses
    FROM [master].[dbo].[addresses]
) q;
```

100 %

Results Messages

	outputTransactionAvg
1	2.4052948903795



### ➤ Question 5:

What is the transaction that has the greatest number of inputs? How many inputs exactly? Show the hash of that transaction. If there are multiple transactions that have the same greatest number of inputs, show all of them.

Transaction ID: 7553001

Number of inputs: 1312

Hash value: 9621b3c67f9bddd3de65fafc488087b8f2b40b638e3a06209a904c66c0b32982

### SQL Query:

```
SELECT
    a.TX_ID,
    b.N_INPUTS,
    a.HASH_VALUE
FROM
    txh a, (
        SELECT
            TX_ID,
            N_INPUTS
        FROM
            tx
        WHERE
            N_INPUTS = (
                SELECT
                    max(N_INPUTS)
                FROM
                    tx
            )
    ) b
WHERE
    a.TX_ID = b.TX_ID;
```

### Screenshots:



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure for 'master' and 'dbo'. The 'dbo' folder is expanded, showing tables like 'addr\_sccs', 'addresses', 'blockhash', 'multiple', 'nonstandard', 'tx', 'txh', and 'txin'. The 'Columns' folder is also expanded, showing columns like 'TX\_ID (int)', 'INPUT\_SEC', 'PREVIOUS\_', 'PREV\_OUT', 'ADDR\_ID (', and 'SUM\_IN (b'. The main window displays a SQL query in the 'SQLQuery1.sql' file. The query is as follows:

```
SELECT
    a.TX_ID,
    b.N_INPUTS,
    a.HASH_VALUE
FROM
    txh a, (
        SELECT
            TX_ID,
            N_INPUTS
        FROM
            tx
        WHERE
            N_INPUTS = (
                SELECT
                    max(N_INPUTS)
                FROM
                    tx
            )
    ) b
WHERE
    a.TX_ID = b.TX_ID;
```

The query results are displayed in the 'Results' tab, showing a single row with the following values:

	TX_ID	N_INPUTS	HASH_VALUE
1	7553001	1312	9621b3c67f9bddd3de65fadc488087b8f2b40b638e3a0620...

➤ **Question 6:**

**What is the average transaction value?**

**12315588064.0354**

**SQL Query:**

```
SELECT
    convert(FLOAT, p.countSum) / q.transaction
FROM
    (
        SELECT
            sum([SUM_OUT]) AS countSum
        FROM
            txout
    ) p, (
        SELECT
            count(*) AS transaction
        FROM
            tx
    ) q;
```

**Screenshots:**

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure, including the 'master' database and various tables like 'dbo.tx' and 'dbo.txh'. The main window displays a SQL query in the 'SQLQuery1.sql' file:

```
SELECT
    convert(FLOAT, p.countSum) / q.transaction
FROM
    (
        SELECT
            sum([SUM_OUT]) AS countSum
        FROM
            txout
    ) p, (
        SELECT
            count(*) AS transaction
        FROM
            tx
    ) q;
```

The Results window shows a single row with the value 12315588064.0354.

(No column name)
1 12315588064.0354

### ➤ Question 7:

How many coinbase transactions are there in the dataset?  
212576

SQL Query:

```
SELECT
    count(*) as coinbaseTrans
FROM
    tx
WHERE
    n_inputs = 0
```

Screenshots:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure, including the 'master' database and various tables like 'dbo.tx' and 'dbo.txh'. The main window displays a SQL query in the 'SQLQuery1.sql' file:

```
SELECT
    count(*) as coinbaseTrans
FROM
    tx
WHERE
    n_inputs = 0
```

The Results window shows a single row with the value 212576.

coinbaseTrans
1 212576

➤ **Question 8:**

What is the average number of transactions per block?

txCount	10000055
blockCount	212576
avgTransPerBlk	47.0422578277887

**SQL Query:**

```
SELECT
    txCount,
    blockCount, (CONVERT(FLOAT, txCount) / blockCount) AS avgTransPerBlk
FROM
    (
        SELECT
            count(*) AS blockCount
        FROM
            blockhash
    ) p, (
        SELECT
            count(*) AS txCount
        FROM
            tx
    ) q;
```

**Screenshots:**

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure, including the 'master' database and the 'dbo' schema. The 'tx' table is highlighted. On the right, the SQL Query window shows the following query:

```
SELECT
    txCount,
    blockCount, (CONVERT(FLOAT, txCount) / blockCount) AS avgTransPerBlk
FROM
    (
        SELECT
            count(*) AS blockCount
        FROM
            blockhash
    ) p, (
        SELECT
            count(*) AS txCount
        FROM
            tx
    ) q;
```

Below the query, the Results tab shows the output of the query:

	txCount	blockCount	avgTransPerBlk
1	10000055	212576	47.0422578277887

V. PART-2 Solutions

➤ **Question 1**

How many users are there in the dataset?

Distinct user IDs – 4315768

Distinct AddrIDs -

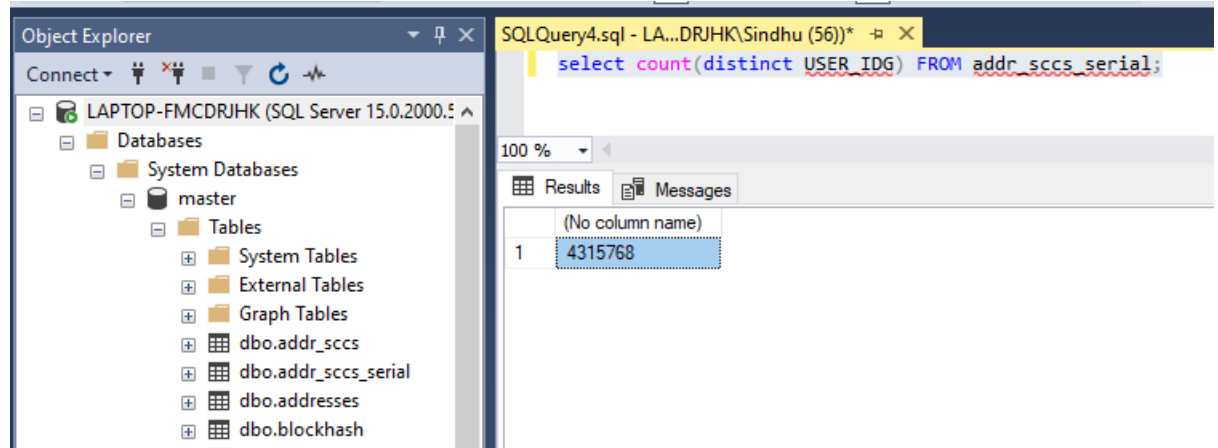
**SQL Query:**

```

select
    count(distinct USER_IDG)
FROM
    addr_sccs_serial;

```

#### Screenshots:



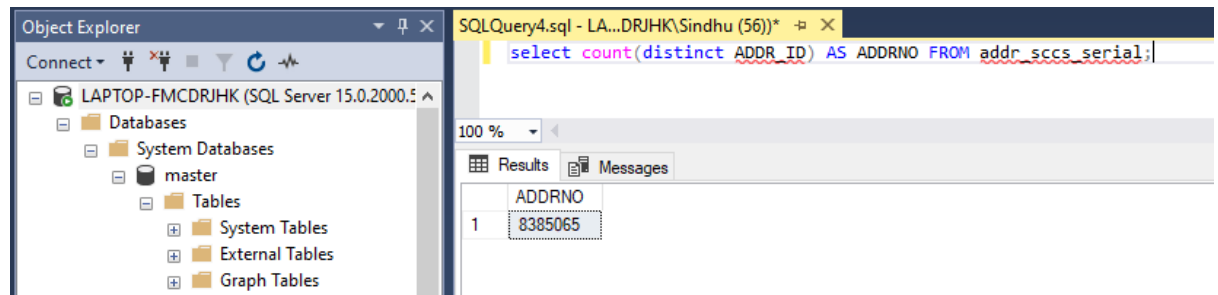
#### SQL Query:

```

select
    count(distinct ADDR_ID)
FROM
    addr_sccs_serial;

```

#### Screenshot:



#### ➤ Question 2.1:

What is the Bitcoin user that is holding the greatest amount of bitcoins?

12461805

How much is that exactly?

2645440

#### SQL Query:

```

select top 10 USER_IDG , ADDR_ID , R.GrtValPerUser
from addrSccsSerial P,
(select top 1 userInfo, sum(greatestBalCount) as GrtValPerUser
from (select blc.Address, blc.greatestBalCount, S.USER_IDG as userInfo

```

```

from(SELECT ADDR_ID AS Address,SUM([SUM_OUT]) AS greatestBalCount
FROM txout T WHERE TX_ID NOT IN (
SELECT PREVIOUS_TX_ID
FROM txin WHERE prev_output_seq = T.output_seq
AND PREVIOUS_TX_ID = T.TX_ID
) GROUP BY ADDR_ID) blc, addrScssSerial S
where blc.Address = S.ADDR_ID) Q
group by userInfo order by GrtValPerUser desc) R
where P.USER_IDG = R.userInfo

```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure, including tables like 'dbo.addresses', 'dbo.addrScss', and 'dbo.addrScssSerial'. The right pane shows a SQL query window with the following query:

```

select top 10 USER_IDG , ADDR_ID , R.GrtValPerUser
from addrScssSerial P,
(select top 1 userInfo, sum(greatestBalCount) as GrtValPerUser
from (select blc.Address, blc.greatestBalCount, S.USER_IDG as userInfo
from(SELECT ADDR_ID AS Address,SUM([SUM_OUT]) AS greatestBalCount
FROM txout T WHERE TX_ID NOT IN (
SELECT PREVIOUS_TX_ID
FROM txin WHERE prev_output_seq = T.output_seq
AND PREVIOUS_TX_ID = T.TX_ID
) GROUP BY ADDR_ID) blc, addrScssSerial S
where blc.Address = S.ADDR_ID) Q
group by userInfo order by GrtValPerUser desc) R
where P.USER_IDG = R.userInfo

```

Below the query, the Results tab shows the following data:

	USER_IDG	ADDR_ID	GrtValPerUser
1	12461805	62528	213940806047898
2	12461805	62530	213940806047898
3	12461805	62533	213940806047898
4	12461805	62534	213940806047898
5	12461805	62535	213940806047898
6	12461805	62537	213940806047898
7	12461805	62538	213940806047898
8	12461805	62539	213940806047898
9	12461805	62541	213940806047898
10	12461805	62542	213940806047898

### SQL Query:

```

select count(addrId)
from serial ss,
(select top 1 userInfo, sum(highestBalanceCount) as HighestBalPerUser
from (select blc.Address, blc.highestBalanceCount, srl.userid as userInfo
from(SELECT addrID AS Address,SUM([SUM]) AS highestBalanceCount
FROM txout tt WHERE txID NOT IN (
SELECT prev_txID
FROM txin WHERE prev_output_seq = tt.output_seq
AND prev_txID = tt.txID
) GROUP BY addrID) blc, serial srl
where blc.Address = srl.addrId) k1
group by userInfo order by HighestBalPerUser desc) bg
where ss.userid = bg.userInfo

```

### Screenshot:

```

select count(addrID)
from serial p,
(select top 1 info, sum(highbal) as highbalUser
from (select h.addr, h.highbal, sl.userid as info
from (SELECT addrID AS addr, SUM([SUM]) AS highbal
FROM txout tt WHERE txID NOT IN (
SELECT prev_txID
FROM txin WHERE prev_output_seq = tt.output_seq
AND prev_txID = tt.txID
) GROUP BY addrID) h, serial sl
where h.addr = sl.addrID) k1
group by info order by highbalUser desc) a
where p.userid = a.info

```

Results Messages

(No column name)

2645440

➤ **Question 2.2:**

What is the average balance per address?

**SQL Query:**

```

SELECT convert(FLOAT, t1.total) / t2.username
FROM ( SELECT SUM(T) AS total
FROM (SELECT SUM([SUM_OUT]) T
FROM txout e
WHERE TX_ID NOT IN (
SELECT PREVIOUS_TX_ID FROM txin
WHERE prev_output_seq = e.output_seq
AND PREVIOUS_TX_ID = e.TX_ID)) t1) t1
,(SELECT count(distinct USER_IDG) as username
from addrSccsSerial) t2;

```

**Screenshots:**

➤ **Question 2.3:**

What is the average number of input and output transactions per address?

What is the average number of transactions per address

**5.39111625091988**

**4.67322478872822**

**10.0643410396481**

**SQL Query:**

```

SELECT convert(FLOAT,P.outputTx) / Q.userInfo AS Input_Avg
FROM (SELECT count(*) AS outputTx FROM txout
) P,(SELECT count(distinct USER_IDG) as userInfo from addrSccsSerial) Q;

```

**Screenshots:**

Object Explorer

Connect

System Databases

master

Tables

System Tables

External Tables

Graph Tables

dbo.addresses

dbo.addrSccs

dbo.addrSccsSerial

Columns

ADDR\_ID (int, null)

USER\_IDG (int, nul)

Keys

Constraints

Triggers

Indexes

Statistics

dbo.blockhash

SQLQuery5.sql - LA...DRJHK\Sindhu (53))\*

SQLQuery4.sql - LA...DRJHK\Sindhu (56))\*

```
SELECT convert(FLOAT,P.outputTx) / Q.userInfo AS Input_Avg
FROM (SELECT count(*) AS outputTx FROM txout
) P,(SELECT count(distinct USER_IDG) as userInfo from addrSccsSerial) Q;
```

100 %

Results Messages

	Input_Avg
1	5.39111625091988

Object Explorer

Connect

System Databases

master

Tables

System Tables

External Tables

Graph Tables

dbo.addresses

dbo.addrSccs

dbo.addrSccsSerial

Columns

ADDR\_ID (int, null)

USER\_IDG (int, nul)

Keys

Constraints

Triggers

Indexes

Statistics

dbo.blockhash

SQLQuery5.sql - LA...DRJHK\Sindhu (53))\*

SQLQuery4.sql - LA...DRJHK\Sindhu (56))\*

```
SELECT convert(float, P.inputTx) / Q.userInfo AS Output_Avg
FROM ( SELECT count(*) AS inputTx FROM txin) P
,(SELECT count(distinct USER_IDG) as userInfo
from addrSccsSerial) Q;
```

100 %

Results Messages

	Output_Avg
1	4.67322478872822

Object Explorer

Connect

System Databases

master

Tables

System Tables

External Tables

Graph Tables

dbo.addresses

dbo.addrSccs

dbo.addrSccsSerial

Columns

ADDR\_ID (int, null)

USER\_IDG (int, nul)

Keys

Constraints

Triggers

Indexes

Statistics

dbo.blockhash

SQLQuery5.sql - LA...DRJHK\Sindhu (53))\*

SQLQuery4.sql - LA...DRJHK\Sindhu (56))\*

```
SELECT convert(float,P.outputTx + Q.inputTx) / R. userInfo AS Trans_Avg
FROM (SELECT count(*) outputTx FROM txout) P
,( SELECT count(*) inputTx FROM txin) Q
,(SELECT count(distinct USER_IDG) as userInfo from addrSccsSerial) R;
```

100 %

Results Messages

	Trans_Avg
1	10.0643410396481

- **Question 3:**  
Give the hash of the transaction sending the greatest number of bitcoins

to the user who is holding the greatest balance.

c246c27e7bacc667d27ace253abf2bba82aa1e5fcd1d73e1b85863f6b890e1bf

#### SQL Query:

```
with P as (select userInfo, sum(greatestBal) as balncCount
from ( select blc.Address, blc.greatestBal, sU.USER_IDG as userInfo
from ( SELECT ADDR_ID AS Address, SUM([SUM_OUT]) AS greatestBal
FROM txout ttout WHERE TX_ID NOT IN (SELECT PREVIOUS_TX_ID FROM txin
WHERE prev_output_seq = ttout.output_seq AND PREVIOUS_TX_ID = ttout.TX_ID)
GROUP BY ADDR_ID
) blc, addrScCsSerial sU
where blc.Address = sU.ADDR_ID) S
group by userInfo)
select top 1 HASH_VALUE from
(select * from
(select top 1 userInfo from P order by balncCount desc) c join addrScCsSerial
op on c.userInfo = op.USER_IDG)R
join txin Q on Q.ADDR_ID = R.ADDR_ID join txh T on Q.TX_ID = T.TX_ID
order by SUM_IN desc ;
```

#### Screenshots:

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure, including tables like 'dbo.addresses', 'dbo.addrScCs', and 'dbo.addrScCsSerial'. The main pane shows a SQL query window with the following query:

```
with P as (select userInfo, sum(greatestBal) as balncCount
from ( select blc.Address, blc.greatestBal, sU.USER_IDG as userInfo
from ( SELECT ADDR_ID AS Address, SUM([SUM_OUT]) AS greatestBal
FROM txout ttout WHERE TX_ID NOT IN (SELECT PREVIOUS_TX_ID FROM txin
WHERE prev_output_seq = ttout.output_seq AND PREVIOUS_TX_ID = ttout.TX_ID)
GROUP BY ADDR_ID
) blc, addrScCsSerial sU
where blc.Address = sU.ADDR_ID) S
group by userInfo)
select top 1 HASH_VALUE from
(select * from
(select top 1 userInfo from P order by balncCount desc) c join addrScCsSerial
op on c.userInfo = op.USER_IDG)R
join txin Q on Q.ADDR_ID = R.ADDR_ID join txh T on Q.TX_ID = T.TX_ID
order by SUM_IN desc ;
```

The bottom pane shows the results of the query, displaying a single row with the HASH\_VALUE:

HASH_VALUE
c246c27e7bacc667d27ace253abf2bba82aa1e5fcd1d73e1b85863f6b890e1bf