To run this queries, a Cardano PostgreSQL database instance (cexplorer) should exist in the system with all the appropriate tables and data.

Reminders:

- **Values are in lovelace. (1 ADA = 1,000,000 lovelaces), so we have to divide amounts by 1,000,000 where needed to get ADA values.**
- **Queries have been implemented using cardano-node (version 1.27.0) and cardano-db-sync (version 10.0.1). For other versions, small changes may be needed for the queries as there may be changes in the tables or fields of tables.**

General Queries

Current total (on-chain) supply. It does not include rewards which have not yet been withdrawn and exist in ledger state (Similar to query from https://github.com/input-output-hk/cardano-db-sync/blob/master/doc/interesting-queries.md).

Query:

```
DROP TABLE IF EXISTS current_supply;
SELECT sum (value) / 1000000 AS current_supply INTO current_supply
 FROM tx_out as tx_outer WHERE NOT EXISTS
  ( SELECT tx_out.id FROM tx_out
      INNER JOIN tx_in ON tx_out.tx_id = tx_in.tx_out_id AND tx_out.index = tx_in.tx_out_index
       WHERE tx_outer.id = tx_out.id
  );
```

Total rewards per epoch. Sum rewards of all stake addresses per epoch.

Query:

```
DROP TABLE IF EXISTS total_epoch_rewards;
CREATE TABLE total_epoch_rewards AS
 SELECT epoch_no, SUM(amount)/1000000 AS amount FROM reward
   GROUP BY epoch_no
     ORDER BY epoch_no ASC;
```

Total pools per epoch. From table "epoch_stake" find distinct pool ids per epoch.

Query:

```
DROP TABLE IF EXISTS total_epoch_pools;
CREATE TABLE total_epoch_pools AS
  SELECT COUNT(DISTINCT pool_id) AS amount, epoch_no
   FROM epoch_stake
     GROUP BY epoch_no
       ORDER BY epoch_no ASC;
```

Epoch total stake, total pledge of pools

Create table of epoch pools. Each row of table contains pool id and in which epoch it is active. Reminder: Most of the pools are active in more than one epoch!

Query:

```
DROP TABLE IF EXISTS epoch_pools;
 CREATE TABLE epoch_pools AS
   SELECT DISTINCT pool_id, epoch_no FROM epoch_stake;
```

Create view for pool_update to remove duplicate updates. There are pools which have updated their information more than once in the same epoch. We keep the last update of each pool for every epoch in which they updated their information.

Query:

```
CREATE OR REPLACE VIEW epoch_pool_update AS
 SELECT * FROM pool_update pu
   WHERE NOT EXISTS
     ( SELECT TRUE FROM pool_update pu2
         WHERE pu.hash_id = pu2.hash_id
           AND pu.active_epoch_no = pu2.active_epoch_no
            AND pu.registered_tx_id < pu2.registered_tx_id);
```

Create table with total pool stake of each pool per epoch. We use table "epoch_stake" table and we sum all stakes for each pool for each epoch.

Query:

```
DROP TABLE IF EXISTS epoch_pool_stake;
CREATE TABLE epoch_pool_stake AS
  SELECT pool_id, epoch_no, SUM(amount)/1000000 AS stake
    FROM epoch_stake
     GROUP BY pool_id, epoch_no;
```

Create table with total epoch stake of each pool with registration id.

Query:

```
DROP TABLE IF EXISTS epoch_pool_stake_reg;
CREATE TABLE epoch_pool_stake_reg AS
 SELECT epoch_pool_stake.*, epu.registered_tx_id
   FROM epoch_pool_stake
     INNER JOIN epoch_pool_update epu ON epu.hash_id = pool_id
        WHERE epu.active_epoch_no =
         ( SELECT MAX(epu2.active_epoch_no)
             FROM epoch_pool_update epu2
               WHERE pool_id = epu2.hash_id
                 AND epu2.active_epoch_no <= epoch_no);
```

Create table with pool_id, pool_ticker, epoch_no and total stake. We find ticker of pool using the preexisting table “pool_offline_data” which contains information about the pool.

Query:

```
DROP TABLE IF EXISTS epoch_pool_ticker_stake;
  CREATE TABLE epoch_pool_ticker_stake AS
     SELECT eps.pool_id, ticker_name, epoch_no, stake from epoch_pool_stake_reg eps
        INNER JOIN pool_offline_data pof ON eps.pool_id = pof.pool_id
          INNER JOIN pool_metadata_ref pmr ON pmr.id = pof.pmr_id
            WHERE pmr.registered_tx_id =
              ( SELECT MAX(pmr2.registered_tx_id)
                   FROM pool_metadata_ref pmr2
                     WHERE pmr2.pool_id = eps.pool_id
                         AND pmr2.registered_tx_id <= eps.registered_tx_id);
```

Create table with pools and their ticker per epoch.

Query:

```
DROP TABLE IF EXISTS epoch_tickers CASCADE;
 CREATE TABLE epoch_tickers AS
    SELECT pool_id, ticker_name, epoch_no
       FROM epoch_pool_ticker_stake;
```

Create table with total pool pledge of each pool per epoch [Declared pledge]. We can find declared pledge from the created table “epoch_pool_update”.

Query:

```
DROP TABLE IF EXISTS epoch_pool_pledge;
 CREATE TABLE epoch_pool_pledge AS
    SELECT ep.pool_id, ep.epoch_no, epu.pledge/1000000 as pledge, epu.registered_tx_id
       FROM epoch_pools ep
         INNER JOIN epoch_pool_update epu ON epu.hash_id = ep.pool_id
           WHERE epu.active_epoch_no =
            ( SELECT MAX(epu2.active_epoch_no)
                 FROM epoch_pool_update epu2
                   WHERE ep.pool_id = epu2.hash_id
                      AND epu2.active_epoch_no <= ep.epoch_no )
```

Heuristic grouping of pools using same ticker prefix

Create table with prefix of tickers of 5 characters length excluding “1PCT” group of pools. 1PCT is a special group which many pools with the same 5 letters ticker but of different version which would be grouped in different groups. For example, there are many groups which have ticker 1PCT5 which would be grouped in 1PCT5 group and not in 1PCT. So, we exclude "1PCT" from tickers of 5 characters length.

Query:

```
DROP TABLE IF EXISTS tickers_prefix_5 CASCADE;
  CREATE TABLE tickers_prefix_5 AS
    SELECT pool_id, ticker_name as prefix, epoch_no
       FROM epoch_tickers
         WHERE char_length(ticker_name) = 5
             AND SUBSTRING(ticker_name,1,4) != '1PCT';
```

Create table with prefix of tickers of 4 characters length.

Query:

```
DROP TABLE IF EXISTS tickers_prefix_4 CASCADE;
  CREATE TABLE tickers_prefix_4 AS
     SELECT pool_id, SUBSTRING(ticker_name, 1, 4) AS prefix, epoch_no
        FROM epoch_tickers
           WHERE char_length(ticker_name) = 4 OR
             ( char_length(ticker_name) = 5 AND SUBSTRING(ticker_name, 5, 5) ~ '^[0-9]+$');
```

Create table with prefix of tickers of 3 characters length.

Query:

```
DROP TABLE IF EXISTS tickers_prefix_3 CASCADE;
  CREATE TABLE tickers_prefix_3 AS
     SELECT pool_id, SUBSTRING(ticker_name, 1, 3) AS prefix, epoch_no
        FROM epoch_tickers
           WHERE char_length(ticker_name) = 3 OR
            (char_length(ticker_name) > 3 AND SUBSTRING(ticker_name, 4, 5) ~ '^[0-9]+$');
```

Create table with prefix of tickers of 5 characters for which there are more than 1 pool with that prefix.

Query:

```
DROP TABLE IF EXISTS group_5 CASCADE;
  CREATE TABLE group_5 AS
     SELECT prefix, array_agg(pool_id), epoch_no
        FROM tickers_prefix_5
           GROUP BY prefix, epoch_no HAVING COUNT(pool_id) > 1;
```

Create table with pool ids and prefix of ticker of pools which create groups with ticker prefix of 5 characters.

Query:

```
DROP TABLE IF EXISTS group_5_ids CASCADE;
  CREATE TABLE group_5_ids AS
     SELECT pool_id, g.prefix, g.epoch_no
        FROM group_5 g, tickers_prefix_5 p
           WHERE g.prefix = p.prefix AND g.epoch_no = p.epoch_no;
```

Create table with prefix of tickers of 4 characters for which there are more than 1 pool with that prefix.

Query:

```
DROP TABLE IF EXISTS group_4 CASCADE;
   CREATE TABLE group_4 AS
     SELECT prefix, array_agg(pool_id) AS pools, epoch_no
        FROM tickers_prefix_4
          WHERE pool_id NOT IN
            ( SELECT pool_id
                 FROM group_5_ids
                   WHERE epoch_no = tickers_prefix_4.epoch_no)
                      GROUP BY prefix, epoch_no HAVING COUNT(pool_id) > 1;
```

Create table with pool ids and prefix of ticker of pools which create groups with ticker prefix of 4 characters.

```
DROP TABLE IF EXISTS group_4_ids CASCADE;
  CREATE TABLE group_4_ids AS
     SELECT pool_id, g.prefix, g.epoch_no
        FROM group_4 g, tickers_prefix_4 p
          WHERE g.prefix = p.prefix
             AND g.epoch_no = p.epoch_no
                 AND pool_id NOT IN
                  ( SELECT pool_id
                       FROM group_5_ids
                         WHERE epoch_no = p.epoch_no);
```

Create table with prefix of tickers of 3 characters for which there are more than 1 pool with that prefix.

Query:

```
DROP TABLE IF EXISTS group_3 CASCADE;
  CREATE TABLE group_3 AS
     SELECT prefix, array_agg(pool_id) AS pools, epoch_no
        FROM tickers_prefix_3
          WHERE pool_id NOT IN
            ( SELECT pool_id
                 FROM group_4_ids
                   WHERE epoch_no = tickers_prefix_3.epoch_no
            ) AND pool_id NOT IN
            ( SELECT pool_id
                 FROM group_5_ids
                   WHERE epoch_no = tickers_prefix_3.epoch_no
             )
                GROUP BY prefix, epoch_no HAVING COUNT(pool_id) > 1;"
```

# Create table with pool ids and prefix of ticker of pools which create groups with ticker prefix of 3 characters.

Query:

```
DROP TABLE IF EXISTS group_3_ids CASCADE;
CREATE TABLE group_3_ids AS
   SELECT pool_id, g.prefix, g.epoch_no
     FROM group_3 g, tickers_prefix_3 p
        WHERE g.prefix = p.prefix  AND g.epoch_no = p.epoch_no
          AND pool_id NOT IN
             ( SELECT pool_id
                 FROM group_4_ids
                    WHERE epoch_no = p.epoch_no)
           AND pool_id NOT IN
              ( SELECT pool_id
                   FROM group_5_ids
                     WHERE epoch_no = p.epoch_no);
```

Create table with IOG pool ids.

Query:

```
DROP TABLE IF EXISTS iog_pools;
CREATE TABLE iog_pools AS
   SELECT DISTINCT pool_id
      FROM (
         ( SELECT pool_id FROM group_3_ids WHERE prefix = 'IOG')
         UNION ALL
         ( SELECT pool_id FROM group_4_ids WHERE prefix = 'IOGP')
      ) temp;
```

Pool stake per epoch [no IOG pools included].

Query:

```
DROP TABLE IF EXISTS epoch_pool_stake_no_iog;
CREATE TABLE epoch_pool_stake_no_iog AS
   SELECT pool_id, epoch_no, SUM(amount)/1000000 AS stake
      FROM epoch_stake
        WHERE pool_id NOT IN (SELECT * FROM iog_pools)
            GROUP BY pool_id, epoch_no;
```

Create table with pool pledge per epoch [no IOG pools and declared pledge less than 2 billions (there is a pool in epoch 225-226 with declared pledge of around 9 trillion which is impossible as there are only around 32 billion ADA in circulation right now)].

Query:

```
DROP TABLE IF EXISTS epoch_pool_pledge_no_iog;
  CREATE TABLE epoch_pool_pledge_no_iog AS
     SELECT pool_id, epoch_no, pledge
        FROM epoch_pool_pledge
          WHERE pool_id NOT IN (SELECT * FROM iog_pools) AND pledge < 1000000000;
```

Total stake per group of pools per epoch

Create table with total stake of group of pools with ticker (pools with same prefix of ticker with 3 characters length in same group - heuristic) per epoch.

Query:

```
DROP TABLE IF EXISTS epoch_group_3_stake CASCADE;
  CREATE TABLE epoch_group_3_stake AS
     SELECT prefix, SUM(stake) AS total_stake, ep.epoch_no, count(*) AS pools_count
       FROM group_3_ids g
         INNER JOIN epoch_pool_stake_no_iog ep ON g.pool_id = ep.pool_id
            AND g.epoch_no = ep.epoch_no
              GROUP BY g.prefix, ep.epoch_no;
```

Create table with total stake of group of pools with ticker (pools with same prefix of ticker with 4 characters length in same group - heuristic) per epoch.

Query:

```
DROP TABLE IF EXISTS epoch_group_4_stake CASCADE;
CREATE TABLE epoch_group_4_stake AS
   SELECT prefix, SUM(stake) AS total_stake, ep.epoch_no, count(*) AS pools_count
      FROM group_4_ids g
        INNER JOIN epoch_pool_stake_no_iog ep ON g.pool_id = ep.pool_id
          AND g.epoch_no = ep.epoch_no
            GROUP BY g.prefix, ep.epoch_no;
```

Create table with total stake of group of pools with ticker (pools with same prefix of ticker with 5 characters length in same group - heuristic) per epoch.

Query:

```
DROP TABLE IF EXISTS epoch_group_5_stake CASCADE;
CREATE TABLE epoch_group_5_stake AS
   SELECT prefix, SUM(stake) AS total_stake, ep.epoch_no, count(*) AS pools_count
      FROM group_5_ids g
        INNER JOIN epoch_pool_stake_no_iog ep ON g.pool_id = ep.pool_id
          AND g.epoch_no = ep.epoch_no
            GROUP BY g.prefix, ep.epoch_no;
```

Total pledge per group of pools per epoch

Create table with total pledge of group of pools with ticker (pools with same prefix of ticker with 3 characters length in same group - heuristic) per epoch.

Query:

```
DROP TABLE IF EXISTS epoch_group_3_pledge CASCADE;
CREATE TABLE epoch_group_3_pledge AS
   SELECT prefix, SUM(pledge) AS total_pledge, ep.epoch_no, count(*) AS pools_count
```

```
    FROM group_3_ids g
      INNER JOIN epoch_pool_pledge_no_iog ep ON g.pool_id = ep.pool_id
        AND g.epoch_no = ep.epoch_no
          GROUP BY g.prefix, ep.epoch_no;
```

Create table with total pledge of group of pools with ticker (pools with same prefix of ticker with 4 characters length in same group - heuristic) per epoch.

Query:

```
DROP TABLE IF EXISTS epoch_group_4_pledge CASCADE;
CREATE TABLE epoch_group_4_pledge AS
  SELECT prefix, SUM(pledge) AS total_pledge, ep.epoch_no, count(*) AS pools_count
    FROM group_4_ids g
      INNER JOIN epoch_pool_pledge_no_iog ep ON g.pool_id = ep.pool_id
        AND g.epoch_no = ep.epoch_no
          GROUP BY g.prefix, ep.epoch_no;
```

Create table with total pledge of group of pools with ticker (pools with same prefix of ticker with 5 characters length in same group - heuristic) per epoch.

Query:

```
DROP TABLE IF EXISTS epoch_group_5_pledge CASCADE;
CREATE TABLE epoch_group_5_pledge AS
  SELECT prefix, SUM(pledge) AS total_pledge, ep.epoch_no, count(*) AS pools_count
    FROM group_5_ids g
      INNER JOIN epoch_pool_pledge_no_iog ep ON g.pool_id = ep.pool_id
        AND g.epoch_no = ep.epoch_no
          GROUP BY g.prefix, ep.epoch_no;
```

Leverage per group of pools per epoch

Create view with leverage of group of pools with ticker (pools with same prefix of ticker with 3 characters length in same group - heuristic) per epoch.

Query:

```
CREATE OR REPLACE VIEW epoch_leverage_group_3 AS
  SELECT gs.prefix, gs.epoch_no,
    COALESCE( gs.total_stake / NULLIF( gp.total_pledge, 0), gs.total_stake) AS leverage
    FROM epoch_group_3_stake gs
      INNER JOIN epoch_group_3_pledge gp ON gs.prefix = gp.prefix
        AND gs.epoch_no = gp.epoch_no;
```

Create view with leverage of group of pools with ticker (pools with same prefix of ticker with 4 characters length in same group - heuristic) per epoch.

Query:

```
CREATE OR REPLACE VIEW epoch_leverage_group_4 AS
  SELECT gs.prefix, gs.epoch_no,
    COALESCE( gs.total_stake / NULLIF( gp.total_pledge, 0), gs.total_stake) AS leverage
    FROM epoch_group_4_stake gs
      INNER JOIN epoch_group_4_pledge gp ON gs.prefix = gp.prefix
       AND gs.epoch_no = gp.epoch_no;
```

Create view with leverage of group of pools with ticker (pools with same prefix of ticker with 5 characters length in same group - heuristic) per epoch.

Query:

```
CREATE OR REPLACE VIEW epoch_leverage_group_5 AS
  SELECT gs.prefix, gs.epoch_no,
   COALESCE( gs.total_stake / NULLIF( gp.total_pledge, 0), gs.total_stake) AS leverage
     FROM epoch_group_5_stake gs
       INNER JOIN epoch_group_5_pledge gp ON gs.prefix = gp.prefix
         AND gs.epoch_no = gp.epoch_no;
```

Leverage of single pools per epoch

Create table with leverage of single pools (pools without ticker and pools with ticker which are not in group) per epoch.

Query:

```
DROP TABLE IF EXISTS epoch_leverage_no_group;
CREATE TABLE epoch_leverage_no_group AS
  SELECT eps.pool_id, eps.epoch_no,
   COALESCE( eps.stake / NULLIF( epp.pledge, 0), eps.stake) AS leverage
   FROM epoch_pool_stake_no_iog eps
     INNER JOIN epoch_pool_pledge_no_iog epp ON eps.pool_id = epp.pool_id
       AND eps.epoch_no = epp.epoch_no
         WHERE NOT EXISTS
          ( SELECT TRUE
              FROM group_3_ids g
                WHERE epoch_no = eps.epoch_no AND pool_id = eps.pool_id )
           AND NOT EXISTS
           ( SELECT TRUE
               FROM group_4_ids g
                 WHERE epoch_no = eps.epoch_no AND pool_id = eps.pool_id )
           AND NOT EXISTS
           ( SELECT TRUE
               FROM group_5_ids g
                 WHERE epoch_no = eps.epoch_no AND pool_id = eps.pool_id);
```

Total Stake & Total Pledge per epoch

Total stake per epoch [No IOG pools included].

Query:

```
DROP TABLE IF EXISTS total_epoch_stake;
CREATE TABLE total_epoch_stake AS
  SELECT epoch_no, SUM(stake) AS total_stake
    FROM epoch_pool_stake_no_iog
      GROUP BY epoch_no ORDER BY epoch_no ASC;
```

Total pledge per epoch [No IOG pools included].

Query:

```
DROP TABLE IF EXISTS total_epoch_pledge;
CREATE TABLE total_epoch_pledge AS
  SELECT sum(pledge) AS total_pledge, epoch_no
    FROM epoch_pool_pledge_no_iog
     GROUP BY epoch_no ORDER BY epoch_no ASC;
```

Live Pools, Pool Operators, Delegators, Leverage

Create table with live pools by keeping only pools which have not been retired in an epoch before current epoch.

Query:

```
DROP TABLE IF EXISTS live_pool CASCADE;
CREATE TABLE live_pool AS
  SELECT * FROM pool_update
    WHERE registered_tx_id IN
      ( SELECT max(registered_tx_id)
          FROM pool_update
            GROUP BY hash_id
       ) AND NOT EXISTS
        ( SELECT *
            FROM pool_retire
              WHERE pool_retire.hash_id = pool_update.hash_id
                AND pool_retire.retiring_epoch <=
                  ( SELECT MAX(epoch_no) FROM block) );
```

Create table with the latest data of pools using data from the preexisting table "pool_offline_data".

Query:

```
DROP TABLE IF EXISTS latest_pool_data CASCADE;
CREATE TABLE latest_pool_data AS
  SELECT pof_outer.pool_id, pof_outer.ticker_name, pof_outer.metadata
    FROM live_pool lp
      INNER JOIN pool_offline_data pof_outer ON lp.hash_id = pof_outer.pool_id
        INNER JOIN pool_metadata_ref pmr ON pmr.id = pof_outer.pmr_id
          WHERE pmr.registered_tx_id =
            ( SELECT MAX(pmr2.registered_tx_id)
                FROM pool_metadata_ref pmr2
                  WHERE pmr2.pool_id = lp.hash_id
```

```
                    AND pmr2.registered_tx_id <= lp.registered_tx_id);
```

Create table with all the live delegations. We find the latest delegation for each stake address which has not been deregistered.

Query.

```
DROP TABLE IF EXISTS live_delegation CASCADE;
  CREATE TABLE live_delegation AS
     SELECT d_outer.*
        FROM delegation d_outer
          INNER JOIN live_pool ON live_pool.hash_id = d_outer.pool_hash_id
            WHERE NOT EXISTS
              ( SELECT TRUE
                  FROM delegation d
                     WHERE d_outer.addr_id = d.addr_id  AND d_outer.id < d.id
              ) AND NOT EXISTS
                ( SELECT TRUE
                     FROM stake_deregistration sd
                       WHERE d_outer.addr_id = sd.addr_id AND d_outer.tx_id < sd.tx_id );
```

Create table with live stake of each stake address (Only UtxOs).

Query:

```
DROP TABLE IF EXISTS live_stake CASCADE;
CREATE TABLE live_stake AS
   SELECT ld.addr_id as addr_id, ld.pool_hash_id as pool_id,
      COALESCE(
          (  ( SELECT COALESCE( SUM(tx_outer.value), 0)
                 FROM tx_out as tx_outer
                    WHERE tx_outer.stake_address_id = ld.addr_id
                      AND NOT EXISTS
                        ( SELECT tx_out.id
                             FROM tx_out
                                INNER JOIN tx_in  ON tx_out.tx_id = tx_in.tx_out_id
                                AND tx_out.index = tx_in.tx_out_index
                                   WHERE tx_outer.id = tx_out.id
                        )
                          GROUP BY tx_outer.stake_address_id )
              ) / 1000000, 0)  AS stake
          FROM live_delegation ld;
```

Create view with all live pool owners (pool operators). We find pool owners using preexisting table "pool_owner" which contains all pool owners in combination with the new table "live_pool" which contains all current live pools.

Query:

```
CREATE OR REPLACE VIEW live_pool_owner AS
   SELECT pool_owner.*
      FROM pool_owner
```

```
        INNER JOIN live_pool ON live_pool.registered_tx_id = pool_owner.registered_tx_id
          INNER JOIN live_delegation ON live_delegation.addr_id = pool_owner.addr_id;
```

Create table with total stake of live pools. Using stake of delegators per pool.

Query:

```
DROP TABLE IF EXISTS live_pool_stake CASCADE;
CREATE TABLE live_pool_stake AS
   SELECT pool_id, COALESCE( SUM(stake), 0) AS total_stake
     FROM live_stake GROUP BY pool_id;
```

Create table with total pledge of live pools. Using stakes of live pool owners per pool.

Query:

```
DROP TABLE IF EXISTS live_pool_pledge CASCADE;
CREATE TABLE live_pool_pledge AS
   SELECT pool_id, COALESCE( SUM(stake), 0) AS total_pledge
      FROM live_stake
        INNER JOIN live_pool_owner  ON live_stake.addr_id = live_pool_owner.addr_id
         GROUP BY pool_id;
```

LIVE Total stake per group of pools

Create table with total stake of group of pools with ticker (pools with same prefix of ticker with 3 characters length in same group - heuristic). [LIVE POOLS]

Query:

```
DROP TABLE IF EXISTS live_group_3_stake cascade;
CREATE TABLE live_group_3_stake AS
   SELECT prefix, SUM(total_stake) AS total_stake, count(*) AS pools_count
      FROM group_3_ids g
        INNER JOIN live_pool_stake lps ON g.pool_id = lps.pool_id
          WHERE g.epoch_no =
             ( SELECT MAX(epoch_no) FROM epoch_stake)
                GROUP BY g.prefix;
```

Create table with total stake of group of pools with ticker (pools with same prefix of ticker with 4 characters length in same group - heuristic). [LIVE POOLS]

Query:

```
DROP TABLE IF EXISTS live_group_4_stake CASCADE;
   CREATE TABLE live_group_4_stake AS
      SELECT prefix, SUM(total_stake) AS total_stake, count(*) AS pools_count
         FROM group_4_ids g
           INNER JOIN live_pool_stake lps ON g.pool_id = lps.pool_id
             WHERE g.epoch_no =
                ( SELECT MAX(epoch_no) FROM epoch_stake)
                   GROUP BY g.prefix;
```

Create table with total stake of group of pools with ticker (pools with same prefix of ticker with 5 characters length in same group - heuristic). [LIVE POOLS]

```
DROP TABLE IF EXISTS live_group_5_stake CASCADE;
CREATE TABLE live_group_5_stake AS
   SELECT prefix, SUM(total_stake) AS total_stake, count(*) AS pools_count
     FROM group_5_ids g
        INNER JOIN live_pool_stake lps ON g.pool_id = lps.pool_id
          WHERE g.epoch_no =
             ( SELECT MAX(epoch_no) FROM epoch_stake )
                 GROUP BY g.prefix;
```

LIVE Total pledge per group of pools

Create table with total pledge of group of pools with ticker (pools with same prefix of ticker with 3 characters length in same group - heuristic) . [LIVE POOLS]

Query:

```
DROP TABLE IF EXISTS live_group_3_pledge CASCADE;
CREATE TABLE live_group_3_pledge AS
  SELECT prefix, SUM(total_pledge) AS total_pledge, count(*) AS pools_count
    FROM group_3_ids g
       INNER JOIN live_pool_pledge lpp ON g.pool_id = lpp.pool_id
         WHERE g.epoch_no =
            ( SELECT MAX(epoch_no) FROM epoch_stake )
               GROUP BY g.prefix;
```

Create table with total pledge of group of pools with ticker (pools with same prefix of ticker with 4 characters length in same group - heuristic). [LIVE POOLS]

Query:

```
DROP TABLE IF EXISTS live_group_4_pledge CASCADE;
CREATE TABLE live_group_4_pledge AS
   SELECT prefix, SUM(total_pledge) AS total_pledge, count(*) AS pools_count
      FROM group_4_ids g
         INNER JOIN live_pool_pledge lpp ON g.pool_id = lpp.pool_id
           WHERE g.epoch_no =
               ( SELECT MAX(epoch_no) FROM epoch_stake )
                   GROUP BY g.prefix;
```

Create table with total pledge of group of pools with ticker (pools with same prefix of ticker with 5 characters length in same group - heuristic). [LIVE POOLS]

Query:

```
DROP TABLE IF EXISTS live_group_5_pledge CASCADE;
CREATE TABLE live_group_5_pledge AS
   SELECT prefix, SUM(total_pledge) AS total_pledge, count(*) AS pools_count
      FROM group_5_ids g
        INNER JOIN live_pool_pledge lpp ON g.pool_id = lpp.pool_id
        WHERE g.epoch_no =
          ( SELECT MAX(epoch_no) FROM epoch_stake )
            GROUP BY g.prefix;
```

LIVE Leverage per group of pools per epoch

Create view with leverage of group of pools with ticker (pools with same prefix of ticker with 3 characters length in same group – heuristic). [LIVE POOLS]

Query:

```
CREATE OR REPLACE VIEW live_leverage_group_3 AS
  SELECT gs.prefix,
     COALESCE( gs.total_stake / NULLIF( gp.total_pledge, 0), gs.total_stake) AS leverage
      FROM live_group_3_stake gs
        INNER JOIN live_group_3_pledge gp ON gs.prefix = gp.prefix;
```

Create view with leverage of group of pools with ticker (pools with same prefix of ticker with 4 characters length in same group – heuristic). [LIVE POOLS]

Query:

```
CREATE OR REPLACE VIEW live_leverage_group_4 AS
   SELECT gs.prefix,
      COALESCE( gs.total_stake / NULLIF( gp.total_pledge, 0), gs.total_stake) AS leverage
        FROM live_group_4_stake gs
          INNER JOIN live_group_4_pledge gp ON gs.prefix = gp.prefix;
```

Create view with leverage of group of pools with ticker (pools with same prefix of ticker with 5 characters length in same group – heuristic). [LIVE POOLS]

Query:

```
CREATE OR REPLACE VIEW live_leverage_group_5 AS
   SELECT gs.prefix,
     COALESCE( gs.total_stake / NULLIF( gp.total_pledge, 0), gs.total_stake) AS leverage
       FROM live_group_5_stake gs
         INNER JOIN live_group_5_pledge gp ON gs.prefix = gp.prefix;
```

LIVE Leverage of single pools

Create table with leverage of single pools (pools without ticker and pools with single ticker). [LIVE POOLS]

Query:

```
DROP TABLE IF EXISTS live_leverage_no_group;
CREATE TABLE live_leverage_no_group AS
  SELECT lps.pool_id,
     COALESCE( lps.total_stake / NULLIF( lpp.total_pledge, 0), lps.total_stake) AS leverage
        FROM live_pool_stake lps
```

```
INNER JOIN live_pool_pledge lpp ON lps.pool_id = lpp.pool_id
  WHERE lpp.pool_id NOT IN
    ( SELECT g.pool_id
        FROM  group_3_ids g
          WHERE g.epoch_no = (SELECT MAX(epoch_no) FROM epoch_stake)
    ) AND lpp.pool_id NOT IN
      ( SELECT g.pool_id
          FROM group_4_ids g
            WHERE g.epoch_no = ( SELECT MAX(epoch_no) FROM epoch_stake)
      ) AND lpp.pool_id NOT IN
          ( SELECT g.pool_id
              FROM group_5_ids g
                WHERE g.epoch_no = (SELECT MAX(epoch_no) FROM epoch_stake)
           );
```

Addresses (Targets of Transaction Outputs which have not yet been used as an Input in another Transaction) and their balance [Only UTXOs]

Create table with addresses and their balance in descending order using their balance.

Query:

```
DROP TABLE IF EXISTS richest_address;
CREATE TABLE richest_address AS
   SELECT tx_outer.address, SUM(tx_outer.value)/1000000 AS balance
      FROM tx_out as tx_outer
        WHERE NOT EXISTS
            ( SELECT tx_out.id
                 FROM tx_out
                    INNER JOIN tx_in ON tx_out.tx_id = tx_in.tx_out_id
                      AND tx_out.index = tx_in.tx_out_index
                        WHERE tx_outer.id = tx_out.id )
                           GROUP BY ADDRESS
                              ORDER BY balance DESC;
```