**Distributed Tweet Location Project**

Goals:

1. Load tweet files into separate SQLite databases in 1-day batches
   1. Database will be named **<batch\_date>.db**
   2. Include user\_id, tweet\_text, tweet\_location, tweet\_date, fips as table columns
   3. This table will be called **tweets**
2. Get the modal location for each 1-day batch of users
3. Check for which users the modal location changes

Steps:

1. Import tweets to database with **import\_tweets\_to\_db.py (0 for Raw, 1 for mp)**
   1. Chunk tweet file by day
   2. Loop through day chunks
      1. Loop through files in chunk
      2. Loop through tweets in file
         1. Insert tweets into database
2. Save unique users to tweets database with **get\_users\_per\_db.py**
3. Loop through unique users in each database with **get\_user\_fips\_per\_db.py (0 for Raw, 1 for mp) mp is a lot faster (NOT SURE YET ABOUT MP)**
   1. Save user\_id and fips in table called users\_fips
4. Loop through records in each users\_fips table and create a new database **aggregated\_data.db**
   1. Insert unique user\_id + fips records into new table with **join\_users\_fips.py**
5. Get potential movers with **get\_potential\_movers.py (-1 FOR ALL USERS)**
   1. Loop through all users in **aggregated\_data.db** table users\_fips
   2. Fetch COUNT(user\_id)
      1. If count > 1 this is a potential mover
6. Get actual movers with **get\_actual\_movers.py (-1 FOR ALL USERS)**
   1. Get unique potential movers
   2. Chunk potential movers into batches of 10000
   3. Loop through user chunks
      1. Loop through each user in each chunk
      2. Check if user has moved and if so get all moves
      3. If user is mover, insert user\_id, fips1, date 1, fips 2, date 2 into **actual\_movers.db**
7. Combine (fips1, date1) and (fips2, date2) into two columns with **get\_actual\_mover\_chronology.py**
   1. Save this to **user\_chronology.db**
8. Get a chronological map of each user’s fips with **get\_fine\_grained\_user\_chronology.py**
   1. For each user
      1. Loop through fips and sort by date
      2. Choose the earliest fips as home
         1. Compare with future fips pairwise but make sure home is always earlier than what it’s being compared with
         2. If fips are far apart enough (100 mi) change home to this fips
         3. Add (user\_id, new\_fips\_home, date) to records
9. Get tweets for users that qualified as movers with **get\_tweets\_for\_movers.py**
   1. Get all tweet db files
   2. Loop through each tweet db file
      1. Create movers tweets db with the same name as tweet db file
      2. Loop through user\_chunks and process user chunk
         1. Loop through each user in user\_chunks
         2. Get all the tweets for each user
         3. Insert all these tweets into table tweets
      3. Close current tweet db file
      4. Commit insertions into mover tweets db
      5. Close mover tweets db
10. Get tweets that are about weather with **get\_weather\_tweets.py**
    1. Get all mover tweets db files
    2. Loop through each db file
       1. Create new database called **weather\_info/date-of-mover-tweets-db-file.db** and a table called **tweets** with all an additional column called **weather**
       2. Open current mover tweets db
          1. Select all rows from table **tweets**
          2. Start a transaction
          3. Infinitely loop while getting TWEET\_BATCH\_SIZE number of tweets during each iteration
             1. If there are tweets to grab from cursor, process this tweet batch by calculating weather tweet\_text is talking about weather or not and append 0 or 1 to each row
             2. Insert this batch of tweets into **tweets** into **weather\_info/date-of-mover-tweets-db-file.db**
          4. Get out of loop when there are no more tweets to grab
          5. Commit insertions
11. Get tweet statistics with **get\_tweet\_statistics.py**
    1. Connect to database **tweet\_statistics.db** and create **statistics** table
    2. Loop through weather\_info database files
       1. Connect to current weather\_info database
       2. Get unique user ids and put them into chunks of 1000
       3. Begin transaction on tweet\_statistics.db
       4. Loop through user chunks
          1. Loop through each user
          2. Call **get\_tweet\_distribution(weather\_info\_db, user\_id)** and add results to empty set called **stats**
             1. Get total tweets, date, and fips for current user
             2. Get number of weather tweets for current user
             3. Return user\_id, date, ntotal\_tweets, nweather\_tweets, fips
          3. Insert current **stats** set into **statistics** table using an executemany
       5. Commit transaction
       6. Close current weather\_info database
    3. Close **tweet\_statistics.db**