Process:

1. Using Python scripts, export assessment roll shapefiles to SQLite database to run queries against *(test.sqlite – name to be changed)*
   1. Add included extra .dbf tables as table in SQLite
   2. Join ‘ROL\_B72\_TAB’ and ‘ROL\_B61\_TAB’ tables to main table for address names and property value data
   3. Index ‘start\_address’, ‘end\_address’, ‘street\_name’, ‘apartment’, and ‘municipal\_code’ columns (vital for running scripts later on -> speed up from hour(s) to minutes)
2. Create .xls consisting only of sales from 2009 and have a sale price > 0. *(crop\_xls.py -> official\_mls\_sales\_2009.xls)* [17,324 addresses]
3. Create base map to correlate MLS sales to the municipality code used by tax assessment database
   1. Find the best map possible for the tax assessment coverage area *(montreal\_municipalites.shp -* <http://geocommons.com/overlays/236782>)
      1. Plot assessment roll shapefile *(Rol\_unite\_p.shp)*
      2. Create polygons for regions not existing in the map already *(assessment\_roll\_regions.shp)*
         1. Export data for points outlying from map boundaries *(outlying\_points.shp)*
         2. Perform Convex Hull by Attribute (municipal\_code) analysis using QGIS *(convex\_hulls.shp)*
         3. Manually clip *convex\_hulls.shp* to best fit actual geographic bounds
   2. Sample 5% of ~1,300,000 points in assessment roll shapefile to make municipal code assignment (in later steps) more manageable using QGIS *(sample\_points.shp)*
      1. Spatial join of ‘region name’ from base map layer to sample points *(correlated\_sample\_points\_municipalities.shp)*
   3. Using Python script, determine what the majority of the sample points in each base map region agrees upon as the ‘municipal\_code’ attribute *(sample\_name\_majority.py)*
      1. Add this code to the base map layer as an attribute (‘CODE\_INT’) *(assessment\_roll\_regions\_correlated.shp)*
4. Geocode MLS sales using Google API *(geocoded\_xls.sqlite)*
   1. Query: ‘start\_address’ + ‘street\_name’ + ‘, ’ + ‘mls\_municipality’\*
      1. \*this is different from the one used by the tax roll and does not match 1:1
   2. Criteria (failed searches are saved in *failed\_lookup.sqlite*):
      1. Fails if result does not contain exact street number as search
      2. Fails if Google could not find address
      3. Fails if data for some reason could not be inserted into the geocoded results database
5. Find only the MLS sales that fall within the geographic region covered by the assessment roll data *(sales\_within\_roll\_regions.shp) [13,393 addresses] <- this should be trimmed to remove regions on base map not covered by assessment roll data for cleaner workflow (would be 12,666 addresses); illegitimate entries still currently removed during failed matching SQL query*
   1. Spatial join of assessment roll municipal code from base map *(sales\_with\_muni\_code.shp -> .csv using QGIS -> .xls using Excel )*
6. Match *sales\_with\_muni\_code.xls* to *test.sqlite* – the official assessment roll data from step 1 *(matched\_mls\_assessments.sqlite)*
   1. Create lookup tables for French articles and generic odonyms used by assessment roll database *(LookupTables.py)*
      1. Strip any of these found in the MLS street name for the SQL query
      2. For example:
         1. MLS input data: “Ave. de l'Hôtel de Ville”
         2. Parsing script: “Ave” -> 08; “de l’” -> J; “Hotel de Ville”
         3. SQL query: “Hotel de Ville” -> ‘street\_name’
   2. Criteria:
      1. Start address
      2. Street name
      3. Municipal code
      4. Apartment/suite no. if available
   3. Tables:
      1. All\_data: contains all suitable geocoded MLS sales [12666 entries]
      2. Matched\_sales: Only MLS sales that were successfully matched to assessment roll database [7885 entries]
7. Export ‘matched\_sales’ table to .csv