**Data processing workflow**

1. Set Jira ticket status to “Data Mapping in Progress” and assign yourself as the Assignee.
2. Set up the repo.
   1. In your local repo, on the main branch, do a git pull to pull down the newest code updates.
   2. Create a branch for your ticket and switch to it.
   3. If it does not exist, create a new folder in the repo for your processing scripts: /ust/sql/states/XX/UST or /ust/sql/states/XX/Releases, where XX is the state code.
   4. Copy the relevant SQL processing template (ust/sql/templates/UST.sql or ust/sql/templates/releases.sql) and paste it into the folder above. Name the copy of the script XX\_UST.sql or XX\_Releases.sql.
   5. Follow the instructions at the top of the processing template to do a global replace of “XX” in the processing template with the state code and save it.
3. Follow the steps in the SQL processing script.
   1. The scripts are heavily commented. Read the instructions for each step carefully.
   2. Some of the steps instruct you to run a Python script. Some of the Python scripts generate additional SQL scripts that are exported to /ust/sql/states/XX/UST or /ust/sql/states/XX/Releases, which you will then open in DBeaver, possibly edit, and then run.
   3. Do NOT commit changes to the Python scripts to the repo after setting the variables at the top of them.
      1. If you need to modify a script in some way to make it work for a specific state, COPY the script, save it to /ust/sql/states/XX/UST or /ust/sql/states/XX/Releases, and make the changes in the copy. You can commit the copy to the repo.
      2. If you find a bug in the script, let Renae know.
4. When all steps of the SQL processing script have been completed, the data are in the EPA tables, and the review materials have been exported, request a peer review from Renae or Jim for the ticket.
5. Address any issues raised in the peer review and repeat until the reviewer approves the ticket to be sent to OUST.
6. Victoria will open the review with OUST.
7. OUST will make comments in the review spreadsheets and will share their feedback during the bi-weekly Tuesday morning meetings. Do your best to attend any meeting where they are discussing a ticket you worked on.
8. Address any issues raised by the OUST review and repeat until OUST approves the ticket to go to the state for review.
9. Address any issues raised by the state during their review and repeat until the state approves the data for UST Finder.
10. If necessary, ERG will geocode the data. The process for this has not yet been put into place.

**Broad overview of the SQL processing script**

1. Import source data into ERG database
   1. Source data goes into state-specific schemas (xx\_ust and xx\_release, where xx is the lowercase state code)
2. Make an entry for the dataset in the appropriate control table (ust\_control or release\_control).
3. Examine the source data and map the source data elements to the EPA template data elements.
4. Where appropriate, map the data element values to the EPA template lookup table values.
5. Deaggregate any data that has been rolled up into multiple values in a single row in the source data so we end up with a single value per row.
6. If any required ID fields do not exist in the source data, generate those fields and map them.
7. Write views in the state schema that manipulate the source data to the EPA template format.
8. Run the Python script that QA’s the views to ensure they are valid.
9. Insert the data into the EPA tables.
10. Export the review materials:
    1. Control table summary
    2. QAQC spreadsheet
    3. Populated EPA template
11. Upload review materials to EPA Teams site.

**Basic database structure**

* State schemas (xx\_ust and xx\_release)
  + If you create any tables in the state schemas, to generate ID columns, or deaggregate state values into separate rows, or concatenate multiple state columns into a single column, prefix the table name with “erg\_” so it is clear the table was ERG’s creation and not part of the original source data.
  + You can create any views you need in the state schemas. Views do not need to have “erg\_” in the name, but should be named in such a way that it is obvious what their purpose is.
* “public” schema
  + There are comments on most tables in this schema. To see the comment, run the following query:
    - select get\_table\_comment('[TABLE\_NAME]'), where [TABLE\_NAME] is the name of the table.
  + Contol tables – one row per dataset; high-level information about the dataset such as organization\_id, date received, original data source, etc. The primary key on this table is used to uniquelyu relate a state dataset to the mapping and main data tables.
    - ust\_control
    - release\_control
  + Main EPA data tables – these are what are used to populate the review spreadsheet and what will be used by UST Finder.
    - These tables contain many database constraints that enforce the business rules from the EPA template.
    - The views you write in the state schemas should have the same name as the table they are meant to populate, with a “v\_” prefix (i.e., xx\_ust.v\_ust\_facility is used to populate public.ust\_facility).
    - EPA UST tables (required tables marked with an asterisk)
      * ust\_facility \*
        + ust\_tank \*

ust\_compartment \*

ust\_compartment\_substance

ust\_piping

ust\_compartment\_dispenser

ust\_tank\_dispenser

* + - * + ust\_tank\_substance
        + ust\_facility\_dispenser
    - EPA Release tables (required tables marked with an asterisk)
      * ust\_release \*
        + ust\_release\_substance
        + ust\_release\_source
        + ust\_release\_cause
        + ust\_release\_corrective\_action\_strategy
  + Lookup tables
    - Usually have table names that are plural/end in “s” (e.g. “substances”)
    - Relate directly to an element and business rule in the EPA template.
    - When doing the element mapping, insert the ID column name for the epa\_column\_name (e.g. “substance\_id”) into the element mapping table (ust\_element\_mapping or release\_element\_mapping), even if the column in the source data you are mapping to is a description/text column, not an ID column.
  + Mapping tables (and views)
    - UST
      * ust\_element\_mapping
      * ust\_element\_value\_mapping
      * v\_ust\_element\_mapping: combines the two tables above for easy querying
    - Releases
      * release\_element\_mapping
      * release\_element\_value\_mapping
      * v\_release\_element\_mapping: combines the two tables above for easy querying
  + Metadata/informational tables
    - Often used by the Python scripts to generate code.
    - DO NOT UPDATE without permission from Renae.
    - Can be used by the developers to understand the database structure and processes.
    - See table comments for additional information about their use.
    - UST-specific metadata tables:
      * ust\_elements
      * ust\_elements\_tables
      * ust\_element\_table\_sort\_order
      * ust\_element\_allowed\_values
      * ust\_element\_lookup\_tables
      * ust\_template\_data\_tables
      * ust\_template\_lookup\_tables
      * ust\_view\_key\_columns
      * ust\_required\_view\_columns
      * generated\_table\_sort\_order
    - Release-specific metadata tables:
      * release\_elements
      * release\_elements\_tables
      * release\_element\_table\_sort\_order
      * release\_element\_allowed\_values
      * release\_element\_lookup\_tables
      * release\_template\_data\_tables
      * release\_template\_lookup\_tables
      * release\_view\_key\_columns
    - Performance measure tables – stores performance measure data that is extracted from a quarterly publication from EPA; used for informational purposes only in the review materials. Most developers will not need to use these tables for any reason.
      * DO NOT UPDATE these tables. Renae will update them quarterly when they are published.
      * performance\_measure\_ust
      * performance\_measure\_release