Running the Transform and Additional Documentation

Link to the tool: https://github.com/PEDSnet/pcornetcdm_to_pedsnetcdm/tree/sql_etl

The PCORnet to PEDSnet transform currently is programmed to transform data from **PCORnet v6.0 CDM to PEDSnet v4.5 CDM (a variation of OMOP v5.4 CDM)**.

This is a CLI tool that can be ran in the command line via a python virtual environment. Essentially, the tool consists of a python wrapper that runs a list of SQL queries. These SQL queries reference generic "SITE" tables in the target database. During running, the "SITE" values are replaced by actual data mart site names. The tool targets data living in PCORnet CDM modeled tables (schema must be in format "site_pcornet"), creates a new schema and tables based on the PEDSnet DDL ("site_pedsnet"), transforms data from the PCORnet tables, and then inserts the newly transformed data into the PEDSnet tables.

Initial setup requirements:

- Proper access credentials for a Postgres server instance.
- Proper database permissions within that server instance. Will need permissions to create schemas, create tables, edit tables, and insert into tables.
- A PCORnet data modeled schema in that server instance with the format "site_pcornet." e.g. "chop_pcornet"
- Populate the User Roles/Groups names as referenced in the **alter_tbl_owner.sql** and **privileges.sql** files that live in **pcornetcdm_to_pedsnetcdm >> sql_etl >> scripts >> ddl_scripts. These role names should be changed to fit the roles in your own postgres permissioning system. They can also simply be commented out.**

Setting up the tool's package requirements within a Python Virtual Environment:

- Open up the command line and clone the repository via "git clone https://github.com/PEDSnet/pcornetcdm_to_pedsnetcdm/"
- Verify that you have python and pip installed and are running the most recent version (note: tool was built and tested in python 3.10.1).
  - You can use "python --version" command to check the version
  - You can use "python -m ensurepip --upgrade" to upgrade
- Verify that you have Postgres installed using "which psql" command
  - if not installed use command "brew install postgres" or "yum install postgres"
- Create a new python virtual environment by running the command "python3 -m venv venv". This will create a new virtual environment to run in your directory. It will live in a folder we called "venv".
- Activate the virtual environment by entering the command "source venv/bin/activate"
- Install all required packages by entering the command "pip install -r requirements.txt"
- run the command "pip install --editable ."

Running the tool:

- Use "cd sql_etl" to move into the sql_etl directory
- Start the tool with the following command: "python3 loading_pedsnet/main.py"
- You will be prompted to enter required postgres credentials including
  - Username
  - Password
  - server name
  - database name within that server
  - schema name (target PCORnet schema to transform)
- This will create a database.ini file in your directory that the program will reference
- Once credentials are entered, you will be prompted with the following options:

  - **Pipeline** - runs **DDL** command followed by **ETL** command. see each command details below.
  - **ETL** - take data from target PCORnet query, transforms data and inserts it into tables within a PEDSnet schema with the same SITE name
  - **DDL** - generates a PEDSnet schema via PEDSnet data models page http://data-models-sqlalchemy.research.chop.edu/pedsnet/
    - Also generates mapping table beforehand
  - **load_mapping_table** - Will generate schema "pcornet_maps" with the table "pedsnet_pcornet_valueset_map" populated with mapping values if not yet created.
  - **test_script** - run a particular .sql transform file that lives in **scripts >> etl_scripts**. Particularly used for testing. To change the .sql file to be ran, change the value of the global variable "test_script_file" at the top of the file l**oading_pedsnet >> process.py**

If starting from scratch, you will want to run the pipeline option. Simply type "pipeline" into the command line and hit enter.

How "pipeline" works:

- Pipeline calls the pipeline_full() function in l**oading_pedsnet >> process.py.** This function subsequently calls ddl_only() followed by etl_only() functions.
- **ddl_only()**

    1. Opens Postgres connection
    2. Verifies whether a corresponding SITE_pedsnet schema exists, if not, it will create one.
    3. Verifies whether pcornet_maps.pedsnet_pcornet_valueset_map table exists, if not, it will create and populate it
    4. Runs the dll() function from l**oading_pedsnet >> query.py** to grab the postgres SQL code for a specified version from http://data-models-sqlalchemy.research.chop.edu/pedsnet/.
    5. Executes that SQL code to generate all necessary tables, indices, and constraints for the pedsnet DDL within the SITE_pedsnet schema
    6. Adds the "site" column to each table in the SITE_pedsnet schema
    7. Set permissions and table owner in the SITE_pedsnet schema
    8. Closes Postgres connection
- **etl_only()**
    1. Opens postgres connection
    2. Verifies whether the cdm_staging schema currently exists. If not, it creates that schema. (note, there are supplementary tables generated in this schema later on that are needed for mappings).
    3. Calls get_etl_ready() function (from l**oading_pedsnet >> query.py**) which creates a new directory **scripts >> etl_scripts_temp**. It then runs through all of the files in **scripts >> etl_scripts,** copies all of the files into **scripts >> etl_scripts_temp**, and replaces all instances of the phrase "SITE" with the actual data mart site name.
    4. For each .sql file in **scripts >> etl_scripts_temp**, a postgres command line command (via **bash script >> etl_bash.sh**) is called to run the .sql file against the target database. These .sql files pull data from the PCORnet tables, reformat them, and inserts them into the Pedsnet tables. Each .sql file either populates a pedsnet table or performs supplementary set up to later populate a pedsnet table in a subsequent .sql file.
        a. Note that for any supplementary tables created in the cdm_staging schema, if the tables already exist, you will receive an "error" statement but the script will continue running. This is okay and should be ignored.
    5. Sets permissions for any newly created or edited tables
    6. Closes postgres connection

Directory and File Formatting within sql_etl

- **bash_script** - directory for all .sh file that run command line commands
    - **combine_csv.sh - currently not implemented**
    - **etl_bash.sh -** runs PostgreSQL commands using database.ini info and a .sql file as input
    - **test_etl_script.sh** - runs PostgreSQL commands using database.ini info and a .sql file as input with AUTOCOMMIT = off. Used for ETL file testing
- **data >> concept_map.txt** - tab delimited text file used to populate the pedsnet-pcornet mapping table. Contains mappings between OMOP concept vocabulary and PCORnet controlled vocabulary
- **loading_pedsnet -** directory for all python driver / supplementary function files
    - **main.py -** driver file. Runs CLICK CLI package and calls functions from process.py
    - **process.py -** contains all primary python functions used in program called from main.py. Calls query.py functions for sql interaction
    - **query.py** - python functions that use or interact with .sql files
- **scripts >> ddl_scripts -** all sql used in creating, editing, or deleting the PEDSnet DDL
    - **alter_tbl_owner.sql** - sql to update table owner for a particular table
    - **create_table.sql -** sql to create mapping table pcornet_maps.pedsnet_pcornet_valueset_map
    - **privileges.sql** - sql to update privileges For PEDSnet tables
    - **site_col.sql -** sql to add "site" column to all tables in PEDSnet schema
    - **trunk_fk_idx.sql - currently not implemented.** Truncates tables, removes foreign keys and indices
- **scripts >> etl_scripts** - all sql that transforms data from PCORnet schema and inserts into PEDSNet schema
    - **Supplementary table creation files -** generates supplementary tables in cdm_staging schema
        - a_ethnicity_xwalk.sql
        - a_gender_xwalk
        - a_p2o_admitting_source_xwalk.sql
        - a_p2o_death_term_xwalk.sql
        - a_p2o_discharge_status_xwalk.sql
        - a_p2o_facility_type_xwalk.sql
        - a_p2o_medadmin_term_xwalk.sql
        - a_p2o_term_xwalk.sql
        - a_p2o_vital_term_xwalk.sql
        - a_race_xwalk.sql
        - a_visit_xwalk.sql
    - **a_location.sql** - populates the **pedsnet.location** table using **pcornet.encounter** and **pcornet.lds_address_history** tables
    - **b_care_site.sql** - populates **pedsnet.care_site** table from the **pcornet.encounter** table
    - **b_provider.sql** - populates the **pedsnet.provider** table using **pcornet.provider** table
    - **c_before_c_person.sql / c_before_person_vacuum.sql / c_person.sql** - calculates each unique patient's primary provider, and then populates the **pedsnet.person** table via the **pcornet.demographic** table
    - **d_death.sql** - populates **pedsnet.death** table via the **pcornet.death** table
    - **d_hash_token.sql** - populates **pedsnet.hash_token** table via the **pcornet.hash_token** table
    - **d_visit_occurrence.sql** - populates the **pedsnet.visit_occurrence** table using the **pcornet.encounter** table
    - **d_visit_payer.sql** - populates the **pedsnet.visit_occurrence** table using the **pcornet.encounter** table
    - **e_condition_occurrence.sql** - populate the **pedsnet.condition_occurrence** table using the **pcornet.condition** and **pcornet. diagnosis** tables
    - **e_observation.sql** - populates the **pedsnet_observation** table via
        - discharge status from **pcornet.encounter** table
        - DRG from **pcornet.encounter** table
        - smoking/tobacco use from **pcornet.vital** table
    - **e_procedure_occurrence.sql** - populates **pedsnet.procedure_occurrence** table using the **pcornet.procedures** table

    - **f_immunization.sql** - populates **pedsnet.immunization** table using the **pcornet.immunization** table
    - **f_location_history.sql** - populates the **pedsnet.location_history** tables from the **pcornet.lds_address_history** table
    - **g_drug_exposure.sql** - populates the **pedsnet.drug_exposure** table from the **pcornet.dispensing**, **pcornet.prescribing**, and **pcornet .med_admin** tables
    - **g_measurement.sql** - populates the **pedsnet.measurement** tables via
        - height, weight, systolic, diastolic, and BMI data from **pcornet.vitals** table
        - lab data from **pcornet.lab_results_cm** table
        - LOINC / SNOMED clinical data from **pcornet.obs_clin** table
- **scripts >> reset_table_scripts - NOT USED**
- **config.py -** python file that pulls and sets database config information from the database.ini file
- **database.ini -** file created and edited after entering credential information. Config file used for database access