PACE-RI DATABASE INFORMATION GUIDE

2019

**7**

Contents

[**Overview** 2](#_Toc33091407)

[Database Files 2](#_Toc33091408)

[Required Files Overview 2](#_Toc33091409)

[SQL Table Information Overview 2](#_Toc33091410)

[Connections to the Databases 2](#_Toc33091411)

[Pipeline for Creation and Updates 3](#_Toc33091412)

[Location 3](#_Toc33091413)

[Setup for PACE-RI Dashboard 3](#_Toc33091414)

[**Workflow** 4](#_Toc33091415)

[Download Files 4](#_Toc33091416)

[Run Database Pipeline File 4](#_Toc33091417)

[To Create the Databases 4](#_Toc33091418)

[To Update the Databases 5](#_Toc33091419)

[Pipeline Details 5](#_Toc33091420)

[Log Clean Up 6](#_Toc33091421)

[Updating An Individual Table 6](#_Toc33091422)

[**Required Files** 6](#_Toc33091423)

[Cognify Files 6](#_Toc33091424)

[Prime Suite 8](#_Toc33091425)

[Grids Outside EHRs 9](#_Toc33091426)

[Additional Files Needed 9](#_Toc33091427)

[**General Data Transformation Information** 10](#_Toc33091428)

[For All Data Report/Tables 10](#_Toc33091429)

[Dataset Specific Transformations 10](#_Toc33091430)

[**Considerations** 11](#_Toc33091431)

[**Troubleshooting** 11](#_Toc33091432)

[**Future Work** 12](#_Toc33091433)

**Styled Words: Databases, *Python Packages,* Table Names,** *Column Names,* **Folder Names,** *File Names*

# **Overview**

There are two databases created for and utilized by the internal dashboard. These databases can also be used for various reports and analyses. They are the default databases used by the ***paceutils*** package.

**For updating the database – the workflow sections outlines the steps in a quick and easy format.**

## Database Files

**PaceDashboard:** a database that combines Cognify, PrimeSuite, and manually collected reports in one place after light data cleaning/wrangling.

**agg:** an aggregate database that compiles data from the PaceDashboard database tables into per month or per quarter values.

## Required Files Overview

All required files are indicated in the [Required Files](#_Required_Files) section of this document as well as the corresponding spreadsheet [*datafiles\_for\_database*](datafiles_for_database.xlsx). This spreadsheet includes column names, descriptions, and information on the transformation or cleaning of each column.

## SQL Table Information Overview

All tables have a primary key which may consist of one column or multiple columns.

*The SQL PRIMARY KEY is a column (or set of columns) in a table which must contain a unique value that is used to identify each row of a table.*

Most tables in the database have a foreign key column (*member\_id*) that is linked to the **ppts** table’s *member\_id* column (primary key of the **ppts** table). This ensures that all member IDs in the reports have been captured in the ppts table.

*A FOREIGN KEY is a key used to link two tables together. A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table. The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.*

The **referrals** table (ppts who never enroll do not have a member ID*)* and the **month\_census** table (it is a unique aggregate table) do not have a foreign key column.

There are 6 views in the **PaceDashboard** database, they each pertain to a different slice of the **inpatient** table (**acute**, **psych**, **nursing\_home**, **custodia**l, **skilled**, and **respite**).

*SQL Views: A view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.*

Column names and descriptions can be found in the [*agg\_database*](file:///C:\Users\hcarmichael\AppData\Local\Temp\7zO0ABF9F47\agg_database.xlsx) and [*dashboard\_database*](file:///C:\Users\hcarmichael\AppData\Local\Temp\7zO0ABF9F47\dashboard_database.xlsx) spreadsheets.

## Connections to the Databases

Connections to the database are created in the python code using the ***Sqlite3*** package. The same package is used in the ***paceutils*** module. This allows for both reading and writing to the database.

## Pipeline for Creation and Updates

The ***Luigi*** package is used to manage the creating and updating pipelines. This package ensures that a process function related to a data report will not run until the related file is copied into the **data\_raw** folder. It then ensures the related *“to SQL”* function will not run until the processed data file is saved in the **data\_processed** folder.

## Location

The databases are currently saved in **V:\Databases**. This is where the **db\_mgmt** scripts connect to and where the ***paceutils*** package reads the data from. The internal PACE dashboard copies the files from the **V:\** **Drive** and saves them locally on the server.

## Setup for PACE-RI Dashboard

**This only has to be done the first time you download the files or install python. If you are already setup this can be skipped.**

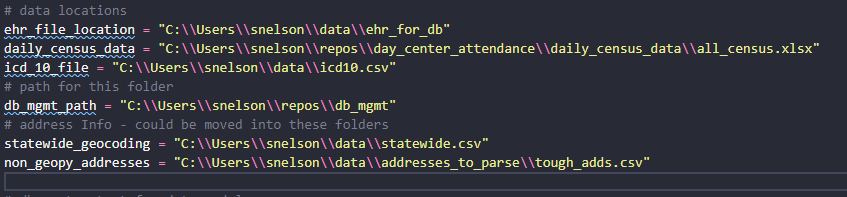
Install python 3.7.5 or greater (Anaconda recommended for personal computers – minconda for servers)

Download all pace\_dash, db\_mgmt, and pace\_utils files

***For the Database***

Create a folder somewhere on your computer name **ehr\_for\_db** **– this is where you will save all downloaded files used to update the database.**

In the **db\_mgmt/code** folder open the filepaths.py file and update the file paths for all data location variables.



Using the anaconda command line prompt navigate to the **db\_mgmt** folder and run ***conda env create -f environment.yml.***

* Activate the environment by running the ***conda active db\_mgmt\_env*** command
* Next run***pip install -r requirements.txt*** and then ***pip install -e <path to pace\_utils folder>***

You are now ready to the follow the instructions in this document for updating the database.

**For the Dashboard**

For local computers – navigate to the **pace\_dash** folder and run ***conda env create -f environment.yml***

* Activate the environment by running the ***conda active pace\_dash\_env*** command
* Once in the environment run;
  + Run ***pip install -r requirements.txt*** in the command line
  + Run ***pip install -e <path to pace\_utils folder>***
  + Run ***python run\_flask.py –debug***
* In your browser (Chrome/Firefox) navigate to http://localhost:8050/ and check that the site is working

This is a local version of the dashboard that can be used to test changes. Any updated files that need to be sent to the server just need to be copied to the V:\Dashboard\pace\_dash folder.

For the server (this is already set up – but in case you need to change servers).

* Run ***pip install -r requirements.txt*** in the command line
* Run ***pip install -e <path to pace\_utils folder>***
* Run python run\_cherry.py to start the dashboard

# **Workflow**

Download the correct files with the correct names to *the folder indicated in the file path you have specified with the* **ehr\_file\_location***variable in the file\_paths.py file in the* **db\_mgmt\code** *folder.*

Open Anaconda Prompt and use the cd command to navigate to the db\_mgmt folder.

Once in the folder type the command ***conda active db\_mgmt\_env*** to activate the environment where the database packages have been installed.

Next run the script for creating (*create\_database.py*) for or updating (*update\_database.py*) the database.



If this runs successfully the function will print the following in the command prompt window: *This progress looks :) because there were no failed tasks or missing dependencies.*

Next you want to run *clean\_logs.py* will to delete the data files and logs created during the process.



***Note:*** *Before creating the database, check the file\_paths.py file to ensure all filepath variables are assigned correctly.*

If you receive an error(“:(“) – the most common cause is that you did not put ***all*** the requirered files in the **ehr\_for\_db** folder.

## Download Files

Download or copy files as outlined in the [Required Files](#_Required_Files) sections. ***Note:*** *If you are creating the database for the first time or recreating the database, run all of the reports with a start date of the first of the first month of the program’s operation.*

## Run Database Pipeline File

### To Create the Databases

Run *create\_database.py*. This can be done using the command line and navigating to the **db\_mgmt** folder and running *python code\create\_database.py*



If this runs successfully the function will print the following in the command prompt window: *This progress looks :) because there were no failed tasks or missing dependencies*

### To Update the Databases

Run *update\_database.py****.*** This can be run using the command line and navigating in to the **db\_mgmt** folder and running *python code\ update\_database.py*



If this runs successfully the function will print the following in the command prompt window: *This progress looks :) because there were no failed tasks or missing dependencies.*

## Pipeline Details

Both *create\_database* and *update\_database* use ***Luigi*** to create a pipeline that runs a number of dependent functions.  
The pipeline begins by copying and renaming (also converting xlsx/xls to csv) files from the indicated **ehr\_file\_location** folder to the **data\_raw** folder.

Next process functions are run for the various reports. Each of these reads a file from the **data\_raw** folder, performs transformations (light data cleaning, renaming of columns, more details can be found in the transformation sections of this document or the related excel spreadsheet), and saves the processed data as a csv file in the **data\_processed** folder.

The pipeline will then begin to run the *“to sql”* related functions. Some tables in the database are updated everytime the pipeline file is run and some tables are completely deleted and replaced with each update.

For updated tables, a **temp** table with the new data is created. Any rows in the **temp** table that have a primary key and already exist in the table are updated. Any rows in the **temp** table with a new primary key, that is a primary key not already in the existing SQL table, are inserted in to the table after the first query that updates existing rows is complete. The **temp** table is then dropped(deleted) from the database.

***Tables That Are Updated:*** **addresses, admission\_claims\_to\_sql, alfs, appts, centers, center\_days, demographics, dx, enrollment, influ, meds, monthly\_census, payments, pneumo, ppts, referrals, teams, wounds**.

***Tables That Are Dropped and Recreated:***: **Auths, Burns, claims\_detail, er\_only** (only last 3 months dropped), **falls, infections, inpatient** (only last 3 months dropped), **med\_errors**.

After all of the tables in the dashboard database file are created/updated, the aggregate tables are created/updated using the data in the **PaceDashboard** database.

After the tables have been created/updated, the pipeline will create a zip folder that includes all of the raw data and the processed data files. The folder is named **todays\_date\_update.zip**. Each month the **data\_archive** folder is automatically cleaned by the pipeline – all folders from the previous month are deleted except for the most recent one.

## Log Clean Up

For the pipeline to know if has updated a table without error, it creates an empty text file in the logs folder.

After the pipeline is complete, the *clean\_logs.py* file can be run in the command line using *python code\clean\_logs.py*. This will delete log files created during the pipeline run. This will only work if the pipeline ran successfully.



## Updating An Individual Table

If you only need to update one table use the *update\_table.py* file. Ensure the correct files needed are in the **ehr\_file\_location** folder, then run the *update\_table.py* file. To do so, navigate to the **db\_mgmt** folder in the command line and run*python code\update\_table.py –table\_name=‘name of table’****.*** *All tables require the enrollment details and quick list files if they include ppts who are not in the database yet.*

# **Required Files**

Download files from sources indicated below to the folder indicated in the file path you specify with the **ehr\_file\_location** variable in the *file\_paths.py* file in the **db\_mgmt\code** folder.

Any parameters for running the report in Cognify/PrimeSuite are indicated below the report. If a parameter is not indicated, it does not need to be changed.

The **>** symbol indicates where in the system the report is located. Some familiarity with Cognify and PrimeSuite is assumed.

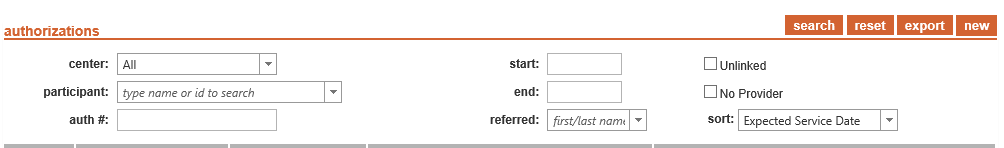
## Cognify Files

***Note:*** *You will need to use the save as button to make sure they save to the correct folder. A couple files will need their names change upon saving as well because the same report is being saved with different parameters.*

**Encounter**

**>Authorizations**

Export all authorizations



**Reporting**

**>Acctg/Claims**

***Claims Admissions and Discharges***

Start Date: 3 months prior to today’s date.

***Payment Register***

Start Date: date of last database update.

***Claim Detail Export***

Start Date: 01/01/2005 (or start date of program’s data)

Thru Date: today’s date

**>Intake/Enroll**

***Enrollment and Disenrollment Details***

End Date: today’s date

***Referral Details***

Call Start Date: 01/01/2005 (or start date of program’s data) – *must run from start of time because otherwise updates to enrollment date or cancel date are not captured.*

***Transfers Between Centers***

Start Date: date of last database update.

**>EMR**

***Dropped Hcc's***

***Note:*** *Report must be run twice, once to capture current dx and once to capture non-current dx. The screen shots below show the parameter for current problems vs non.*

*Current Problems*



*Not Current Problems (add \_nc to this file when saving to the folder)*



**>Encounter**

***Inpatient***

Start Date: 01/01/2005 (or start date of program’s data)

***ER Only***

***Note:*** *This report must be run twice, once for ER only visits and once for ER visits that result in an inpatient admission.*

*ER Only Visits*

Start Date: 01/01/2005 (or start date of program’s data)

Service Tracking: DP33

*ER Resulting in Inpatient (add \_IP to this file when saving to the folder)*

Start Date: 01/01/2005 (or start date of program’s data)

Service Tracking: IP02

**>Participant**

***Quick List***

Participant Status: enrolled

***Admission Changes***

Start Date: date of last database update.

End Date: today’s date.

***Center Days***

As Of: Don’t change.

**>Quality**

***Incident Tracking Survey***

***Note:*** *This will need to be run with the same date parameters, but just changing the type to cover all incidents except General. The list of incident types we are reporting on are Falls, Med Errors, Burns, and Infection.*

Start Date (occurred): 01/01/2005 (or start date of program’s data)

End Date(occurred): today’s date

## Prime Suite

**Reporting > Report Selection > Custom Reports**

**>Chart > Medications**

***meds (save as*** *meds****)***

Runs with a predefined filter of the last 2 weeks

**>Schedule>Patient Appointment**

***appointments (save as*** *appts****)***

Match Value: date of last database update.

**>System>Patient Demographics**

***db\_demo (save as*** *demographics****)***

Runs with a filter for enrolled ppts only

***addresses (save as*** *addresses****)***

Runs with a filter for enrolled ppts only

**Reporting>Flex Reports>Chart**

***Influenza contra (save as*** *influ\_contra****)***

***Pneumococcal contra (save as*** *pneumo\_contra****)***

***Pneumococcal (save as*** *pneumo****)***

Runs with a filter of date administered after 01/01/1900

***Influenza (save as*** *influ****)***

Runs with a filter of date administered after 01/01/2017

## Grids Outside EHRs

***wound\_grid***

Before copying to the **ehr\_for\_db** folder, fill in any missing Member IDs. (file pw: Wounds2018)

Copy file to the **ehr\_for\_db** folder.

Open file and delete any non-dates in the data discovered or date healed columns (sometimes these columns have nonhealing in them)

Save first tab as a csv named wound\_grid

*Note: Needs to be copied from* **V:\Nursing** *and the first tab should be saved as a .csv file with file name wound\_grid.*

***all\_census***

*Update with census counts sent by day center managers.*

***Note:*** *ensure this file is in the folder indicated in the file path you specify with the* **daily\_census\_data***variable in the file\_paths.py file in the* **db\_mgmt\code** *folder.*

## Additional Files Needed

**Both files only need to be put into place once – can be copied from the current data analyst.**

***OpenAddress Geocode*** *File – only needs to be downloaded once – RI, MA, and CT are provided with these files.*

Download for your area from <http://results.openaddresses.io/>

*Note: ensure this file is in the folder indicated in the file path you specify with the* **statewide\_geocoding***variable in the file\_paths.py file in the* **db\_mgmt\code** *folder.*

***ICD10 File***

The included list of ICD10 codes in a csv.

*Note: ensure this file is in the folder indicated in the file path you specify with the* **icd\_10\_file***variable in the file\_paths.py file in the* **db\_mgmt\code** *folder.*

# **General Data Transformation Information**

## For All Data Report/Tables

* Any member id/participant id column is renamed to member\_id.
* Column names are all converted to lowercase.
* All spaces in column names are replaced with an underscore.
* Camel Case (CamelCase) column names are broken up with an underscore (camel\_case).
* Some column names must be renamed explicitly in the code.
* All dates are formatted as YYYY-MM-DD.

## Dataset Specific Transformations

**Addresses**

Unit information is removed from the address column and merged with the address\_2 column to create a unit column. Catches all Apt, Flr, Fl, Box, Bldg, Unit, and # related information.

Addresses have their latitude and longitude geocoded using either the OpenAddress data file or the Nominatim geolocator via the geopy package.

Any addresses that cannot be parsed by these two methods are added to the file indicated in the file path specified by the **non\_geopy\_addresses** in the *file\_paths.py* file. These can be manually run through an online geocoder and then added to the OpenAddress file locally. Next time the addresses are updated, they will be geocoded.

**Inpatient/Claims\_Detail/Admission\_Claims**

Some common facility names are replaced in utilization/claims files to make merging the tables easier. A list can be found in any of the related process\_ files.

The Admission Changes file is used to track only ALF admissions. The discharge type column is broken into discharge type and discharge facility.

Admission within 6 months of enrollment and day of the week information is added to utilization data.

**Authorizations**

Service and Authorization codes are replaced with the description of the code. Prevents needing to look them up later, which often occurs.

**Demographics**

The gender column in demographics is converted into a binary column (1: female/0: male).

**Enrollment**

The Medicaid and Medicare columns in the enrollment file have their numbers replaced with 1 to indicate an id was there and 0 to indicate it was not. This lessens the amount of PHI in the database.

**Incidents (Falls/Med\_Errors/Burns/Infections/Wounds)**

All incidents datasets have Yes/No columns converted to 1/0 columns (1: yes/ 0: no)

**Ppts/Quick List**

Quick List is only used to get team information for the ppts table

**Vaccinations (Influ/Pneumo)**

Dose Status for vaccinations information is mapped to integer values. (1: administered, 0: not administered, 99: contra).

# **Considerations**

The update table function is quite slow and there could be an improvement in just recreating the database weekly. Unfortunately, some reports crash Cognify/PrimeSuite when run for too long of a period – so this would not be possible for all tables.

SQLite may be outgrown soon, and data will need to be moved to a different database. This may require help from IT in setting up a database server and will need the python code to be updated to use another connection engine.

DBeaver is a useful opensource program for connecting and exploring these databases.

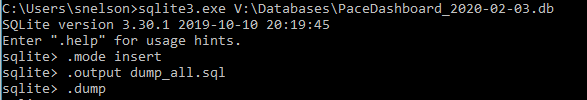
# **Troubleshooting**

If you receive a ☹ from the update script, start by checking that you have all the necessary files needed in the **ehr\_for\_db**. Try running the script again, sometimes it doesn’t complete due to skipping steps, so it just needs to be run again. It will skip any processes that finished in the previous run.

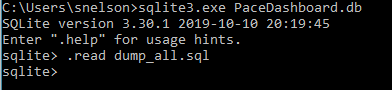
Those are the two most common errors; the script should handle any other ones on its own.

In the case of a ***SQLiteException: The database disk image is malformed database disk image is malformed*** error, there are some additional steps to take to copy and restart the database.

In the command line run sqlite3.exe V:\Dashboards\PaceDashboard.db, then run the following lines;



This will create a SQL file with all the data. Next uses Ctrl C to exit the sqlite3 command. Then in the command line run sqlite3.exe PaceDashboard.db, then run the following;



This creates a fixed version of the database in your users folder. Delete the version in the V:\Drive and copy this new version into the folder.

# **Future Work**

Add type annotations and unit testing to codebase.

Day Center attendance needs to be updated to use PrimeSuite appointment data instead of data collected by the day center managers.

Need to update file paths to use OS module.