**Objective**

This week is a data migration exercise.

## Step 1 - Start the feature branch

Create a new branch 'assignment9'. Change to the branch and push the branch to GitHub. All assignment9 work will be done in the assignment 9 branch.

**-- Feature branches are just what we do - 0pts**

## Step 2 - Generate Dump Scripts

**Team management concerns...**

**This step needs to be done early!** The person you are paired with is depending on your data to complete their own assignment. While the project is not technically "group work" it is still group/team work. If you have not resolved the data export with your peer by Wednesday night, you need to get ahold of the instructor so that something can be done on Thursday. "Resolved" in this case may not mean that the export is working but that the export format is well defined and you have confidence that the exporter will be working in time to write your import script.

A communication strategy to help management help you is to CC your manager when trying to coordinate with other teams. That lets your manager know what is going on so that they can support your efforts. This assignment provides a context to practice this communication strategy. Please CC me on email to setup a meeting time to discuss the export/import and CC me on email that captures your understanding coming out of the meeting. I'm not going to your meetings but I do need a datapoint regarding when you met and that you have figured out a way forward before the end of the week.

**Export Implementation**

Add a directory to your repository named 'export'. The export directory should hold a script named 'export_data.sh' and any other scripts needed to export data from your database to one or more data files. It will be run from $REPO/export with

```
bash export_data.sh <dbname> <output dir>
```

- The first argument <dbname> will be the name of the database to export the data from. The database server will be running on localhost (127.0.0.1 or /tmp) at port 5432.
- The second argument <output dir> with be the path for the directory where the data files should be written.

You should also add a README.txt file to the export directory that documents the files in the export directory, how to run the export_data.sh script, and the format of your export files.

The data you export should be sufficient to reconstruct the history of an asset. Where an asset is and when/where it has been moved.

- assets - minimally asset tag and description
- facilities - minimally fcode and common name
- asset at information - minimally asset tag, fcode, and arrival/departure/disposal times
- asset transit - minimally asset tag, load/unload times
- users - optionally you may want to include who recorded the load/unload for transit, which would likely require exporting the list of users.

You should talk with the person who will be trying to import your data about what a the data dump should look like to support their import. I like/suggest CSV files but in discussing the problem with your peer, you may identify a better file format.

After completing this step, your repository should look something like:

```
$REPO/
    README.txt
    preflight.sh
    sql/
        README.txt
        create_tables.sql
        ...
    src/
        app.py
        config.py
        templates/
            ...
        static/
            ...
    export/
        export_data.sh
        ...
```

Test that your code generates the expected output files in the correct format. Commit and push your code.

**-- Data Export - 10 pts (5pts resolved by midweek, 5pts for generating export files)**

## Step 3 - Swap Data

In this step you will generate some data using your peer's application and export that data.

Clone your peer's repository and install their application. Use their application to make some users, facilities, and assets. Execute some asset transits using their solution. Keep a mental note of the data you are creating.

Use their export_data.sh script to dump the data you entered into data files.

**-- Data Generation - 0 pts (This isn't that different from a code review)**

## Step 4 - Import Implementation

Now that you've generated some data, time to see if that data can be imported into your own system.

Use your own preflight script ot install your application. Use your application to make some users, facilities, and assets. The records you create should partially overlap with the records you created in the last step (e.g. some of the users, asset tags, and facility codes should be the same). You should also include entries that are different (users, asset tags, and facilities that were not used in the previous step).

Add a directory to your repository named 'import'. The export directory should hold a script named 'import_data.sh' and any other scripts needed to import data from your peer's data files. It will be run from $REPO/import with

```
bash import_data.sh <dbname> <input dir>
```

- The first argument <dbname> will be the name of the database to import the data into. The database server will be running on localhost (127.0.0.1 or /tmp) at port 5432.
- The second argument <input dir> with be the path for the directory where the data files should be read from.

You should also add a README.txt file to the import directory that documents the files in the import directory and how to run the import_data.sh script.

Your import scripts should merge your peer's data into your database. Overlapping records should not be duplicated (e.g. if your database already has an asset tag AX3427 and the import data refers to an asset tag AX3427, your existing AX3427 asset record should be used rather than inserting a second AX3427 asset record.). After your import_data.sh script is run, using a web browser, a user should see both the records created using your application and the records imported from the data files.

After completing this step, your repository should look something like:

```
$REPO/
    README.txt
    preflight.sh
    sql/
        README.txt
        create_tables.sql
        ...
    src/
        app.py
        config.py
        templates/
            ...
        static/
            ...
    export/
        export_data.sh
        ...
    import/
        import_data.sh
        ...
```

Test your code. Commit and push your code.

**-- Data Import - 10 pts (5pts assets list existing and imported assets at the correct locations on the asset report date, 5pts transit report includes existing and imported assets in transit on the transit report date)**

## Step 5 -- Merge the feature branch

With the iteration complete, merge the work back into the master branch. Specific git commands can be found in earlier assignments.