**ETL Project – Final Report**

**Summary:**

In my current position, we have several sources of data from a handful of different places. This data historically has been merged and managed manually on spreadsheets. A method that is overly prone to error and subject to size limitations. Each week I get a list of employees that are set to start from Human Resources (HR). This data includes job titles, manager, salary, and station information). I then must set up a file to upload to our timekeeping system that includes what charge codes each employee will need. Charge codes are allocated by job title, and there are two places to get them. The first is attained by exporting what is currently assigned to each person by job title in our timekeeping system, and the second is an export from a SharePoint list that we manage and update with budget changes and adds. We mostly have 15-20 employees deploying in a week. However, during the start of the austral summer, it can be 80 to 100. The volume is a time killer since excel does not really allow lookups by grouping or auto add people to a list of active employees for tracking. I used python and PGAdmin to automatically create the file I need to upload and to add the employees and charge codes to two database tables that can be queried and maintained.

**Extract**

I extracted data from three sources:

1. Export.CSV: This is an export of the charge codes assigned to each person that is currently active in our system. I need this file to ensure each new person’s charge code list matches all the others with the same job title.
2. Sharpoint.CSV: This is an export of which job titles were originally budgeted for which charge codes. I need this file for two reasons. One, if a new employee has a job title that is new, there will not be corresponding codes in the export.csv. Two, if a budget owner has added a charge code for a resource that is tracked on this list.
3. CurrentWeek.csv: This is the spreadsheet from HR that I get each week showing who will be starting and who needs set up. This comes from a program database, that I do not have access to. Again, names, actual salaries and any other personal information has been replaced with dummy data.

**Transform**

I transformed each file in the following way:

1. Export.CSV and Sharpoint.CSV:

Goal is to combine the two files and have one concise list of charge codes by job code (PLC) to eventually merge with the current week file.

* 1. Dropped unnecessary columns
  2. Combined the two lists
  3. Dropped duplicates
  4. Found a charge code that is not supposed to be charged to and deleted it from the list

1. CurrentWeek.CSV:

Goal is to pull the create a column that assigns the correct job code to each employee so that can be merged with the above table. Each job code is comprised of four letters. The first denotes what station the employee is working at, the middle two are referred to as the “skill code”, and the last letter tells us what company they work for. In this case, the final letter is always “P” for PAE.

* 1. Created a station dictionary and cast it into a data frame.
  2. Pulled the two-letter skill code out of the billet column and added a new column with this information.
  3. Looped through the “Station/Season” column using a matcher function to essentially find any text matching station names in created dictionary and add what it finds to another column in the data frame.
  4. Created a column that pulls in the station letter based on the station data that was just populated.
  5. Created a final column called “PLC” to concatenate the station letter, skill code, and letter “p”, and then dropped unnecessary columns. Now I have a column that I can merge with my export.

1. Putting it all together:

Goal is to create a data frame that I can export to a CSV for upload into our timekeeping system.

* 1. Performed an inner join on PLC columns to populate all charge codes that need to be uploaded based on an employee’s job code.
  2. Dropped all columns that are not needed for the upload file.
  3. Added two new columns that the timekeeping system requires and populated them with necessary values.
  4. Renamed the columns to the names required by the timekeeping system
  5. Re-set the index
  6. This data frame can now be exported at any time and uploaded easily into the timekeeping system.

**Load**

Now I can update a running list of charge codes by job title code and add the new employees to an “employees” table. I chose to use PQAdmin and create a relational database, because this particular data requires more structure. To do this I created a path to my database and connected to my database engine. I checked that the tables I needed were in the database. Then appended my new tables to each corresponding table. Lastly, read the tables to ensure the data was indeed pushed to each corresponding table.

**Reflection:**

Overall, this is amazing and will save me so much time. It will also help to ensure consistent charging across job titles and reduce the number of requests I receive to add charge codes to new people that were missed due to the manual process we were utilizing before. The hardest part was getting the station to populate into a new column, because the station names are all different lengths and often written or abbreviated in different ways in the station/season column. I really enjoyed getting to use my newly acquired skills to automate something I do weekly and seeing how using different coding languages can ensure accuracy and efficiency.