Benjamin Roseburrough

Sasmita Mohanty

Randall Stevenson

**ETL Project Report**

**Extraction Steps**

* Data for different wine ratings was found on Kaggle and Dataworld websites. The orginal files came in various formats including CSV, JSON and XLXS.
* The Kaggle data was from winemag.com (Wine Magazine’s web site). The CSV file was loaded into GitHub.
* The Dataworld data was originally available in CSV, XLXS and JSON format, but some of the symbols were not UTF-8 characters. This was addressed by reading the XLXS file version.
* A column was added to both extracted pandas indicating the source of the data WM or DW. This was to allow duplicates to be removed and to average the rating and price of any wine in both WM and DW and change the source to indicate it was both.
* The Winemag file did not have vintage as a column. The title of a wine is the winery, the vintage and the designation. The vintage was extracted from the column of string variables under title. The results were stored as integers in a vintage column.
* The Dataworld file had the Vintage column as a mixture of datetime variables and strings. A subroutine converted them into integers for the vintage year and replaced the Vintage column.

**Transformation Steps**

Winemag-130k-v2.csv

* The ‘Winemag-130k-v2.csv’ dataframe was called.
* The first column was initially unnamed but changed to ‘id’.
* Then, the 'id', 'description', 'province', 'region\_1', 'region\_2', 'taster\_name', 'taster\_twitter\_handle', and 'variety' columns were deleted.
  + This was so that only the ‘title’, ‘vintage’, ‘country’, ‘winery’, ‘designation’, ‘points’, ‘price’, and ‘source’ columns remained.
* Next, several lines of code were written to keep only the rows that had values in specified columns.
* Perform groupby with mean aggregation to look at average point and price for each title.
* Perform sort\_values to determine the Top 5 & Bottom 5 popular titles.

Wines.xlsx & Combining

* The ‘Winemag-130k-v2.csv’ data & cleanup code was taken with just a few changes.
  + Instead of columns to drop, columns to keep were specified.
  + The ‘id’ column was kept.
  + A copy of the first dataframe was made.
  + All NaN values were dropped rather than just those from certain columns.
  + A row count was performed.
  + Each step stored the dataframe in a new variable.
* The same steps were then used to cleanup ‘Wines.xlsx’.
* The ‘Wines.xlsx’ dataframe was appended onto the ‘Winemag-130k-v2.csv’ dataframe to make ‘wine\_df\_final’.
  + It was originally proposed to use a ‘join’ statement, but there were issues with finding a unique primary key to join the data on (which is admittedly odd, since the ‘title’ contains a composite of ‘winery’, ‘vintage’, and ‘designation’).
  + Since the data came from separate sites, it was assumed that there wouldn’t be duplicates.
* The ‘groupby’ function was used in conjunction with the ‘mean’ function to return a dataframe with the mean ‘points’ and ‘price’ values by each ‘title’ as ‘wine\_df\_grouped’.
* The ‘sort\_values’ function was run on the resultant dataframe on ‘points’ to show the top 5 & bottom 5 popular wine titles.

**Loading Steps**

* Two dataframes were to be loaded, so a database called ‘wines\_df’ was created in PgAdmin4 with two tables- ‘wines\_table’ (for ‘wine\_df\_final’) and ‘wines\_titles\_table’ (for ‘wine\_df\_grouped’).
* A ‘create table’ statement was created for each table with columns to match the corresponding Pandas dataframes.
* An engine was created, with the table names being confirmed.
* The dataframes were then exported to their corresponding tables in PgAdmin with a ‘to\_sql’ statement.