**ETL Project Report**

## Team Members:

* Sasmita Mohanty
* Benjamin Roseburrough
* Randall Stevenson

# Extraction Steps

* Data for different wine ratings was found on Kaggle and Data world websites. The original files came in various formats including CSV, JSON and XLSX.
* The CSV and JSON files from Kaggle had almost the same data, so it was decided to use only the CSV file with a large dataset.
* The Kaggle data was from winemag.com (Wine Magazine’s web site). It was in CSV format, which was converted to a panda’s data frame by pd.read\_csv().
* The Data world data was available in XLSX formats, which was converted to panda’s data frame by pd.read\_excel().
* A column was added to both extracted panda data frames indicating the source of the data WM (wine magazine) or DW (data world). 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 CSV 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 XLSX file had the Vintage column as a mixture of datetime variables and strings. So, the vintage year was extracted from each record and converted into an integer for the vintage year and replaced the Vintage column.

# Transformation Steps

## Data Transformations

### winemag-130k-v2.csv

* The ‘Winemag-130k-v2.csv’ was loaded into a panda’s data frame.
* The index column was initially unnamed but changed to ‘id’, which duplicated the pandas index.
* 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, records with NaN values were dropped by .dropna() after making a copy of the data frame.

### Wines.xlsx

* The ‘Wines.xlsx file was converted to a panda’s data frame by pd.read\_excel().
* Only ['Vintage', 'Country', 'Designation', 'Points', 'Price', 'Title', 'Variety', 'Winery', 'source' ] were kept in the data frame. The column names in Wines.xlsx were capitalized.
* A copy of the data frame was made and Nans were dropped using .dropna().
* Index was reset using reset\_index().
* All the columns were renamed to ['id', 'vintage', 'country', 'designation', 'points', 'price', 'title', 'variety', 'winery', 'source'], to match the column names for the data frame from the csv file.
* The columns were rearranged so that both data frames have same order of columns.
* The ‘price’ column had $ in it and datatype was object. So, the $ sign was striped and ‘price’ column was converted to float using .astype(‘float’).

## Combining the data frames

* The ‘Wines.xlsx’ data frame was appended onto the ‘winemag-130k-v2.csv’ data frame 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’ by ‘title’ function was used in conjunction with the ‘mean’ function to return a data frame to see ‘points’ and ‘price’ columns and it was sorted descending by ‘point’ using .sort\_values() as ‘wine\_df\_sorted’.
* This ‘wine\_df\_sorted’ data was run using .head() and .tail() function to show the top 5 & bottom 5 rated wine titles respectively.

# Loading Steps

* It was decided to load the data frames to the SQL based database, PostgreSQL, because of the nature of the data frames. These data frames were large tables with almost similar kind of data in them and a pre-defined schema could be done before loading them.
* A database called ‘wines\_db’ was created in PgAdmin4 with two tables- ‘wines\_table’ (for ‘wine\_df\_final’) and ‘wines\_titles\_table’ (for ‘wine\_df\_sorted’).
* A ‘create table’ statement was created for each table with columns to match the corresponding Pandas data frames.
* An engine was created, to connect to PostgreSQL from pandas.
* The data frames were then loaded to their corresponding tables in PgAdmin4 with ‘to\_sql’ statements.