ETL Project

Data Analytics and Visualization

Devin Stern

Marie Wothe

Technical Write-up

Death Candy

We elected to join the following datasets together due to an interesting correlation between health and death. Recent research has shown that excessive amounts of sugar will have detrimental effects on a person's health and life expectancy. The combination of these two datasets will help define the actual cause of death and the effects of eating too much sugar, particularly during the Halloween holiday season.

Extract. Dataset #1 *Leading Causes of Death: United States* was downloaded in csv format from Data.Gov’s data catalogue. It was published by the Centers for Disease Control and Prevention by the National Center for Health Statistics. Report Title: Leading Causes of Death: United States, URL: <https://catalog.data.gov/dataset/age-adjusted-death-rates-for-the-top-10-leading-causes-of-death-united-states-2013>.

Extract: Dataset #2: *State-by-State Favorite Candy* was downloaded from a website called CandyStore.com. This csv contained sales from 2007 through 2015 and focused on the three months that lead up to Halloween.

Transform. *Leading Causes of Death: United States.*

1. Import CSV. The first step in transforming this data was to load it into a jupyter notebook and into a pandas dataframe.

#import dependences

import pandas as pd

from sqlalchemy import create\_engine

#Import CSV

death\_file = "death\_causes.csv"

death\_df = pd.read\_csv(death\_file)

1. Filter Years. This dataset contained the top 10 leading causes plus a category for “all Causes” of death by state from 1999 through 2016. We decided to select one year (2012) to load into the database since our other dataset was an aggregate of data by state over years 2007-2015. To select only year = 2012, we used iloc.

#Keep only causes of death from 2012

death\_2012\_df = death\_df.loc[death\_df["Year"] == 2012]

1. Keep Needed Columns. This dataset contained the following columns: *Year, 113 Cause Name, Cause Name, State, Deaths, Age-Adjusted Death Rate*. The dataframe was reformatted to include only needed columns: *Year, Cause Name, State, and Deaths.*

#Take only the year, cause name, state, deaths columns

death\_2012\_df = death\_2012\_df[["Year", "Cause Name", "State", "Deaths"]]

1. Drop ‘All Causes’. We did not need the “All Causes” values, so we dropped all rows containing this value using iloc.

#Drop the "All Causes" cause, and make sure it was dropped

death\_2012\_df = death\_2012\_df[death\_2012\_df['Cause Name']!="All causes"]

1. Rename Columns. The column names for this dataframe needed to be renamed in order to import and play nice with SQL.

#Rename columns to play nice with SQL

death\_2012\_df.columns=["\_Year", "Cause\_Name", "State", "Deaths"]

Type of Transformation needed: *State-by-State Favorite Candy*

1. Import dependencies and load csv

import pandas as pd

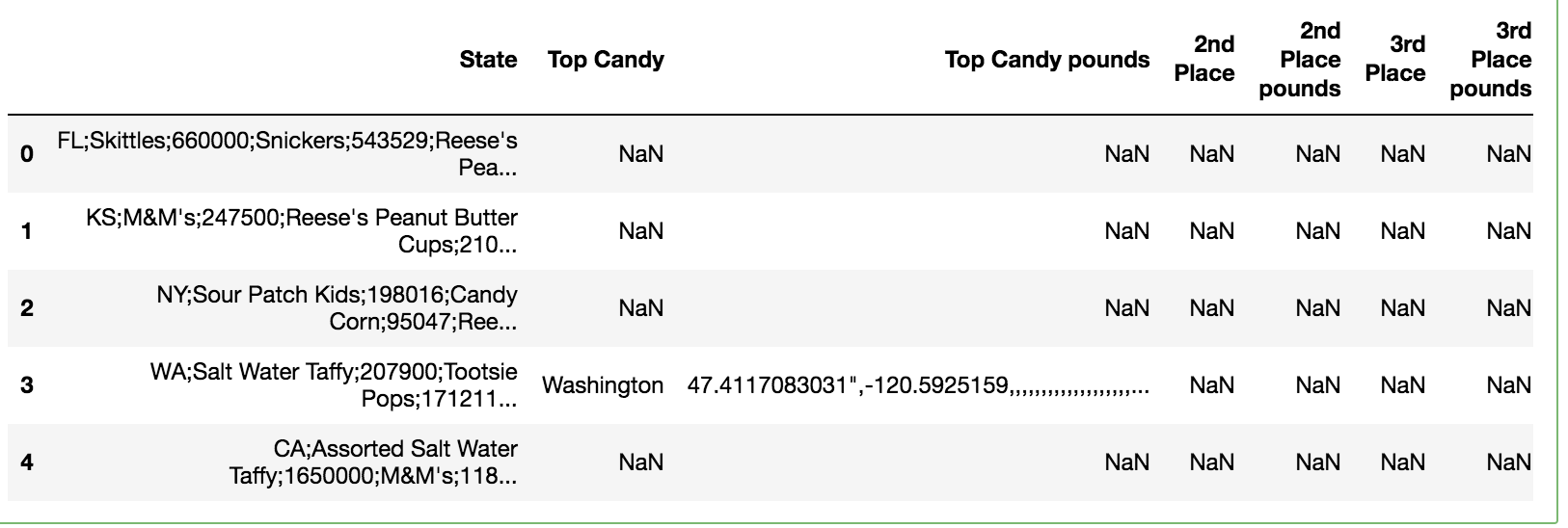
from sqlalchemy import create\_engine

csv\_file = "halloween\_candy.csv"

halloween\_candy\_df = pd.read\_csv(csv\_file, sep=";")

halloween\_candy\_df.head()

1. Because the data was separated by a “;” instead of a “,”, pandas read all the data into the same column (see image) so I had to isolate the single column, split the string and expand the dataframe to make it readable/usable.



#drop the other columns

halloween\_candy\_df = halloween\_candy\_df["State"]

#make a copy to make changes

halloween\_candy\_df.head()

halloween\_practice = halloween\_candy\_df.copy()

#Split the string of the series and make a new column as needed

halloween\_practice = halloween\_practice.str.split(";",

n = -1, expand = True)

1. The image below is the end result after renaming the columns.

#Label columns and drop extra

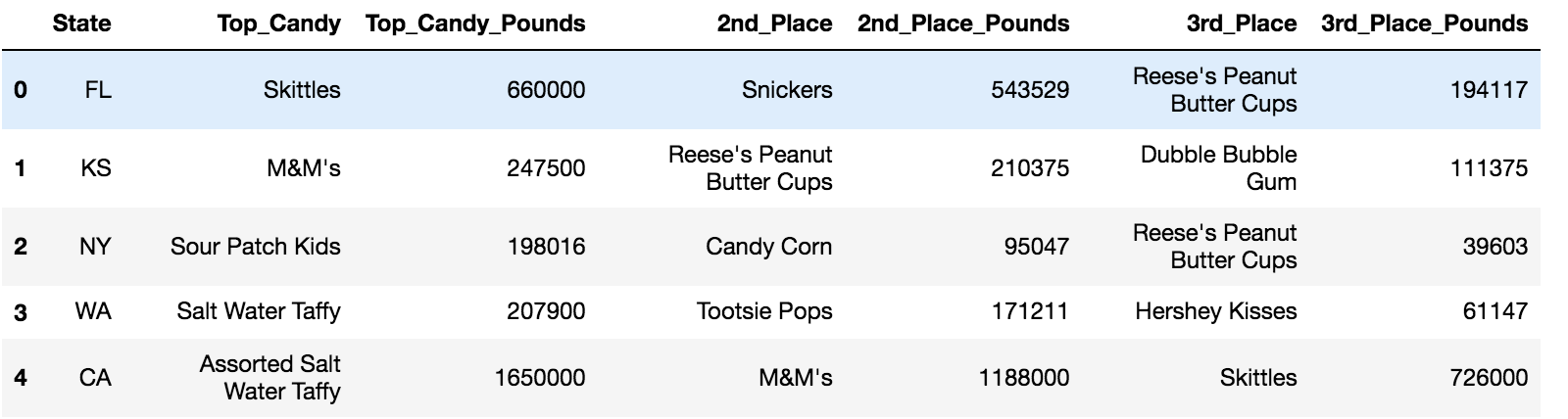
halloween\_practice.columns = ["State", "Top\_Candy",

"Top\_Candy\_Pounds", "2nd\_Place", "2nd\_Place\_Pounds",

"3rd\_Place", "3rd\_Place\_Pounds"] #, "Extra Crap"]

halloween\_practice = halloween\_practice.drop("Extra Crap", axis =1)

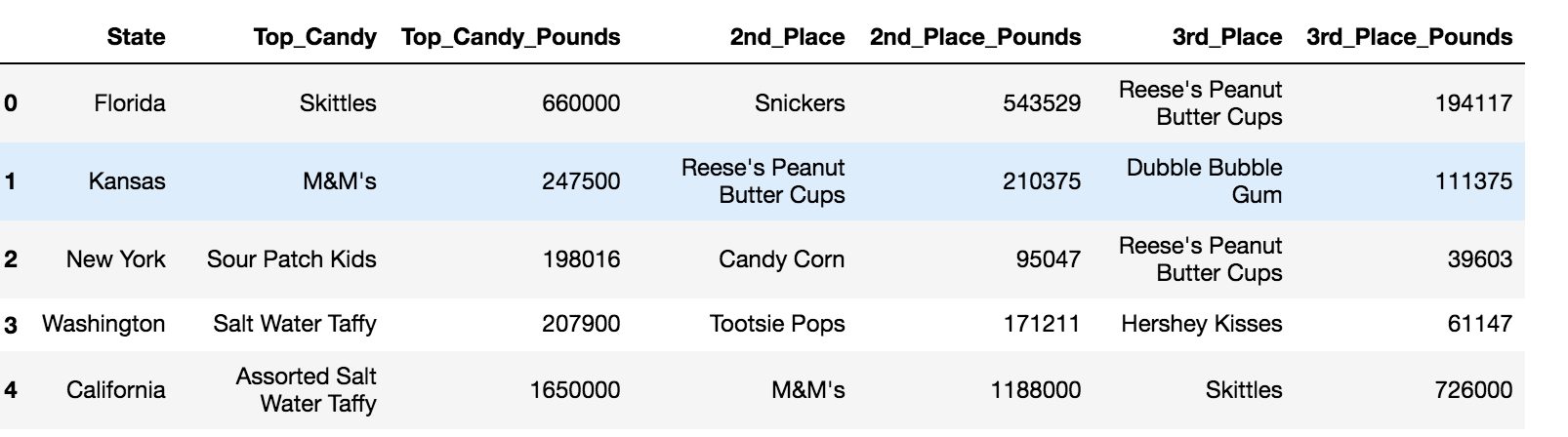
halloween\_practice.head()



1. The next issue was how to make the two datasets joinable. Because we were joining by state and this dataset had the two-letter state codes as opposed to the full name of each state, we changed the state code to the full name. There was already a dictionary of each state name and country code on the internet; we simply had to write the line to map over the values and replace them with the state’s full name. The result is the image.

#map over the values

halloween\_df["State"] = halloween\_df["State"].map(us\_state\_abbrev)



Load.

1. Create Database and Tables. Before loading the transformed data into a database, we needed to create the framework for the relational database and the table to hold the data. This was done using MySQL Workbench and this is done thusly:

CREATE DATABASE Death\_Candy;

USE Death\_Candy;

CREATE TABLE death state (

\_Year INT,

Cause\_Name TEXT,

State TEXT,

Deaths INT

);

CREATE TABLE halloween (

State TEXT,

Top\_Candy TEXT,

Top\_Candy\_Pounds int,

2nd\_Place TEXT,

2nd\_Place\_Pounds int,

3rd\_Place TEXT,

3rd\_Place\_Pounds int

);

1. Create Database Connection. In jupyter notebook, create a database connection to database *Death\_Candy* in MySQL Workbench.

#Connect to database

connection\_string = "PASSWORDREDACTED@127.0.0.1/Death\_Candy"

engine = create\_engine(f'mysql+pymysql://{connection\_string}')

1. Populate tables. Finally, we connected to the target database and appended it with the new transformed data.

death\_2012\_df.to\_sql(name='death\_state', con=engine, if\_exists='append', index=False)

halloween\_df.to\_sql(name='halloween', con=engine,

if\_exists='replace',index=False)

1. Check data has loaded. This creates a relational database (we can join the two datasets on state) and the tables in MySQL look like this:

