# API Documentation

We did not create our own functions. However, we used a lot of built-in functions. Listed below are the built-in functions we utilized for our analysis:

**Libraries**

```
pandas
numpy
seaborn
matplotlib.pyplot
statsmodels.api
sk.learn.preprocessing
   - LabelEncoder
statsmodels.formula.api
   - ols
pandas.tseries.offsets
   - MonthEnd
```

**Dataframe Setup & Information**

`pd.read_csv()` = reads the file and turn them into a pandas dataframe
`df.head()` = prints the top 5 rows of the dataframe
`df.info()` = returns the number of cells of the dataframe in each columns, as well as its data type
`df.columns` = returns the column labels of the dataframe
`df.copy()` = creates a copy of the dataframe
`reset_index()` = resets the index of the dataframe

**Data Type Manipulation**

`pd.to_numeric()` = converts all data in a column into numeric values
`variable_name.astype()` = changes the data type of a variable of your choice

**Dataframe Manipulation**

`df.dropna()` = drops a row or column with missing values
`df.drop()` = drops a row or column of our choice
`df.join()` = combines two dataframes together
`pd.merge()` = combines two or more dataframes together

**Dataframe Filtering**

`df[column_name].replace(original_data, new_data)` = replaces a data from a specified column with a new data
`df.groupby(column_name)[response_variable]` = returns the value of the response variable, when they are grouped by the column of choice
`df[column_name].isin(list_of_values)` = filters the dataframe to only include the values specified, which should be inside the column of your choice

`variable_name.sort_values()` = returns your data sorted based on column values, in either ascending or descending order
`variable_name.sort_index()` = returns your data sorted based on index values, in either ascending or descending order

**Counts & Missing Values**
`df[column_name].value_counts()` = returns the count of all of the values in the column of your choice
`df.isna().sum()` = returns the count of missing values in each of the dataframe's columns
`df[column_name].isnull()` = returns the rows with missing values

**Time as a Data Value**
`pd.to_datetime()` = creates a new column with a datetime data type, based on pre-specified dataset columns of year, month, and date
`pd.Timestamp()` = specifies a timestamp as a variable
`np.timedelta64()` = creates a duration of a time object, to aid in time-related numerical analysis

**Pivot Table**
`pd.pivot_table()` = creates a pivot table of your choice

**Descriptive Statistics**
`mean()` = returns the mean of the variable of choice
`variable_name.describe()` = returns the count, mean, standard deviation, minimum, maximum, 25th percentile, 50th percentile, and 75th percentile values of your variable of choice
`df.describe()` = returns the count, mean, standard deviation, minimum, maximum, 25th percentile, 50th percentile, and 75th percentile values of your dataframe
`variable_name.round()` = rounds up the numerical data from your variable of choice
`np.log1p(column_name).quantile()` = returns the quantile of your choice of the column you specified

**Graph Creation**
`variable_name.plot()` = creates a graph of your variable of choice
`plt.scatter()` = creates a scatterplot of the two variables of your choice
`plt.subplots()` = creates multiple plots all in one figure
`axes[].set_title()` = sets the title of your subplots
`axes[].set_xlabel()` = sets the x-axis label of your subplots
`axes[].set_ylabel()` = sets the y-axis label of your subplots
`sns.set()` = set a specific seaborn style
`sns.scatterplot()` = creates a scatterplot using seaborn
`sns.barplot()` = creates a barplot using seaborn
`sns.boxplot()` = creates a boxplot using seaborn

`sns.histplot()` = creates a histogram using seaborn
`sns.heatmap()` = creates a heatmap using seaborn
`plt.figure()` = specifies the dimension/size of your graph
`plt.title()` = specifies the title of your graph
`plt.xlabel()` = specifies the x-axis label of your graph
`plt.ylabel()` = specifies the y-axis label of your graph
`plt.xtics()` = specifies the degree rotation of your x-axis bar label
`plt.grid()` = adds grid lines to a plot
`plt.show()` = shows the plot you created

**Correlation Analysis**
`df.corr()` = creates a correlation matrix of all of the columns inside your dataframe
`df.corr()[column_name]` = calculates the correlation of the column of your choice with the rest of the columns

**Categorical Labels Encoding**
`LabelEncoder()` = calls for a function to help convert categorical labels into numerical labels
`label_encoder.fit_transform()` = converts categorical labels into numerical labels
`pd.get_dummies()` = converts categorical labels into numerical labels, using one-hot encoding technique (creates new columns for all of the categorical labels and represents the variable data through binary values [0 = no, 1 = yes])

**Linear and Logistic Regression Analysis**
`np.log1p()` = calculates the ln (natural log) of your variable of choice
`ols().fit()` = creates an ordinary least squares regression model, and fits the model to the dataset
`sm.stats.anova_lm()` = creates an anova table of your model