A comprehensive list of the changes made in the project files:

1. Download dependencies
2. Download and install openpyxl (library)

# Extraction of data

1. Reading data into Pandas DF
2. Brief description of dataframe using .info()
3. Getting the columns within the dataframe

# Creation of Category and Sub-Category DF

1. Segmentation of column values within dataframe, like module 1 challenge:

* Reference (column segmentation)

<https://pandas.pydata.org/docs/reference/api/pandas.Series.str.split.html#pandas.Series.str.split>

1. getting the unique values within columns and creation of lists

* Reference (unique values)

<https://pandas.pydata.org/docs/reference/api/pandas.api.extensions.ExtensionArray.unique.html#pandas.api.extensions.ExtensionArray.unique>

* Reference (lists)

<https://pandas.pydata.org/docs/reference/api/pandas.api.extensions.ExtensionArray.tolist.html#pandas.api.extensions.ExtensionArray.tolist>

1. Creation of f string to depict category and sub-category (will use a separate cell to ensure depicted values are correct)
   * <https://www.geeksforgeeks.org/formatted-string-literals-f-strings-python/?ref=header_outind>
2. Using LEN function to determine number of distinct variables within category
   * <https://pandas.pydata.org/docs/reference/api/pandas.Series.str.len.html#pandas.Series.str.len>
3. Created a new list to append the word((cat) short for category) to all ids within previous array. Usage of list comprehension.
   * <https://pandas.pydata.org/docs/user_guide/indexing.html>
4. DataFrame creation for category and sub-category
   * <https://www.geeksforgeeks.org/different-ways-to-create-pandas-dataframe/?ref=header_outind>
5. Writing the dataframes into their own csv files respectively
   * <https://www.geeksforgeeks.org/export-pandas-dataframe-to-a-csv-file/?ref=header_outind>

# Create Campaign DataFrame

1. Creation of dataframe copy using .copy() function
2. Renaming the columns within the new(copied) dataframe
   * The changes should be made accordingly with the projected answer (found in file before any changes made
   * <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rename.html#pandas.DataFrame.rename>
3. Conversion of data types from columns goal and pledged
   * <https://www.geeksforgeeks.org/change-data-type-for-one-or-more-columns-in-pandas-dataframe/?ref=header_outind>
4. Checking the data types
   * <https://www.geeksforgeeks.org/get-the-datatypes-of-columns-of-a-pandas-dataframe/?ref=header_outind>
5. Formatting / changing the columns format.
   * The units are set in seconds
   * The units are of Unix timestamp format and should be essentially converted
   * <https://www.geeksforgeeks.org/convert-datetime-string-to-yyyy-mm-dd-hhmmss-format-in-python/?ref=header_outind>
   * Several issues faced when determining timestanmp values for these columns and adjust accordingly. Use forward slash to indicate continuation on next line
6. Merging the two data frames created via category and sub-category
   * <https://www.geeksforgeeks.org/joining-two-pandas-dataframes-using-merge/?ref=header_outind>
7. Dropping the unwanted columns within the new dataframe (can be further discussed but I went through each of the columns and have determined that the following can be dropped: staff\_pick, spotlight, category, sub-category. This is due to the fact that both the category and sub have their own dataframes respectively. This was not the case as the initial solution projects differently. After comparison I found that the columns to drop were staff\_pick, spotlight, category & sub-category, category, sub-category. I dropped these
8. Saved the new (cleaned dataframe) to a csv file and pushed work to git (6th push)

# Create contacts DataFrame

## Pandas

1. Reference the dataframe (contact\_info\_df. This was already pre created in the starter code and will be used for the following section.
   * We have the option to create the dataframe using Pandas or Regex, we will be doing both for the sake of education.
     1. References will be listed as needed
2. The DataFrame seems to be 1x1 with all information bundled in single tabular form
   * We will need to essentially change the dimensions of the said DataFrame and will follow the below:
     1. <https://www.geeksforgeeks.org/how-to-convert-pandas-dataframe-into-json-in-python/?ref=header_outind>
3. There are several key value pairs within the initial data so we can use that to create a for loop and iterate through to create a new structured dataframe.
   * K = key and V=value separated with a delimiter : as shown initially in line 23
4. Note: the contacts info within the excel file starts on line 4
   * Replace initial code :
     1. # Read the data into a Pandas DataFrame. Use the `header=2` parameter when reading in the data.
        1. With:
     2. contact\_info\_df = pd.read\_excel('Resources/contacts.xlsx', header=3)
5. after minor adjustment dictionaries can be created using iteration method.
   * Key functions used json.loads and .items
     1. <https://www.geeksforgeeks.org/json-loads-in-python/?ref=header_outind>
6. Now take the refined data (initially “contact\_info”) and place it into its own dataframe.
   * Creation of dataframe with following:
     1. <https://www.geeksforgeeks.org/different-ways-to-create-pandas-dataframe/?ref=header_outind>
     2. <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html#pandas.DataFrame>
7. Next step to create new dataframe with the following columns:
   * ‘Contact\_id’, ‘name’, ‘email’
   * This can be done by creating separate lists that include the necessary information.
   * The lists can then be used to populate the dictionary
8. After creating the new refined dataframe we can check the datatype using
   * <https://www.geeksforgeeks.org/how-to-check-the-data-type-in-pandas-dataframe/?ref=header_outind>
9. We will also be using .info for more information regarding the datatypes within the refined dataframe
10. Creation of new columns “first\_name” and “last\_name”
    * Minor error in intial code where quotation is used instead of underscore: first”name
    * We will be using the string split method to split the name at the delimiter with reference to the following:
      1. <https://pandas.pydata.org/docs/reference/api/pandas.Series.str.split.html#pandas.Series.str.split>
11. We will then drop the column name since it has been made redundant.
    * After dropping the column, no need to run again otherwise error will ensue
12. Reordering the columns to produce the same output as previewed: contact\_id, first\_name, last\_name, email.
    * <https://www.geeksforgeeks.org/change-the-order-of-a-pandas-dataframe-columns-in-python/?ref=header_outind>
13. Checking the data types one more time before exporting refined data to csv file
    * We will use the .info method and limit it to .head()
      1. Data seems organized and ready to export
      2. Be advised that the DataFrame name was changed to “refined\_contact\_info\_df” which differs from the initial name provided within the starter code “new\_contact\_df”

## Regex

1. Using Regex to create contacts DataFrame
   * Has not been used before so the following research will be done using the references found within this section of document
   * <https://pandas.pydata.org/docs/reference/api/pandas.api.types.is_re.html>
   * <https://www.geeksforgeeks.org/python-regex/?ref=header_outind>
2. Created a copy of the original contact\_info\_df which included only one column
3. In order to use Regex we have to import the dependency
4. The process of data extraction via regex can be found in the following :
   * <https://www.geeksforgeeks.org/extracting-email-addresses-using-regular-expressions-python/?ref=header_outind>
5. Regex patterns and tutorial can be found at:
   * <https://www.geeksforgeeks.org/write-regular-expressions/?ref=header_outind>
     1. Depiction of the regex expression as follows:
        1. Reference to the column to use
        2. \s\* matches zero or more spaces after referenced column
        3. (\d{4}) used to capture exactly 4 digits
6. Checking data types for confirmation using .info() DataFrame method
7. Conversion of data types within regex can be done with the following:
   * <https://www.geeksforgeeks.org/replace-values-in-pandas-dataframe-using-regex/?ref=header_outind>
   * To change the data type we reference the copied dataframe that was created in previous steps and further reference the column we would like to change data type for. Followed by = and same dataframe and column with .astype(with reference to the data type we want to convert to)
   * We will be using int 64 because it is numeric
   * Double check the change reflected with .info()
8. Extraction of name of contact using the regex method and adding it to a new column. Follows:
   * <https://pandas.pydata.org/docs/reference/api/pandas.Series.str.extract.html>
   * <https://www.geeksforgeeks.org/python-extract-substring-using-regex/?ref=header_outind>
   * .str.extract(r'([^nameil"\s][A-Za-z]+\s+[A-Za-z]+)')
     1. .str.extract : pandas method used for extraction of specific portion of string
     2. ([^nameil"\s] : matches any single letter not listed inside the brackets
     3. [A-Za-z]: for the first word matches one or more letters from a to z both capital and lower-case
     4. \s+ : matches one or more spaces between the words
     5. [A-Za-z]: matches the second word
   * Further reading included in rubric: To split each "category & sub-category" column value into "category" and "subcategory" column values, use df[["new\_column1","new\_column2"]] = df["column"].str.split(). Make sure to pass the correct parameters to the split() function.
9. We will continue to utilize the string extraction method shown above but for the email address.
   * For this case the regex pattern can be depicted through:
   * "email":: Matches the exact text "email": in the JSON-like string.
   * \s\*: Matches any optional spaces after "email":
   * "([^"]+)":
     1. Matches and captures the email address, which is enclosed in double quotes (").
     2. [^"]+: Matches one or more characters that are not a double quote (").
10. Prompted to create a new DataFrame with the following and essentially drop the contact\_info column as it has now become redundant
    * We will utilize the .copy () method in pandas to create a new DataFrame with all the refined data
      1. There is no prompt for the new dataframe name as of now so we will call it “refined\_contacts\_info\_df”
         1. Referenced to the contacts\_info\_df\_copy
11. After refinement of data we can now segment further into first\_name and last\_name columns
    * Doing so using the regex method would be done with splitting the string at the .
      1. Referencing the last data frame we can create the two new columns using [[“first\_name” , “last\_name”]]
      2. String split : ' ', n=1, expand=True
         1. <https://www.geeksforgeeks.org/javascript-string-split-example-with-regex/?ref=header_outind>
12. Dropping the name column as it has now become redundant
    * Reference refined DataFrame “contacts\_info\_final” and drop column “name”
13. Reordering the columns as intended
    * Reference refined\_contacts\_info\_db
    * Creation of new variable holding the correct sequence with regard to the example solution
14. Checking the data types once more for confirmation and cross reference with rubric
    * As indented everything seems to be correct
15. Slight difference in final name of CSV in comparison to proposed solution name
    * rubric : contacts\_df\_clean
    * work: refined\_contacts\_info\_df

# END

# SQL

Now that the refined dataframe(s) have been created we can import them to SQL and create an operating database

1. creation of database with the following tables: contacts, category, subcategory, and campaign
   1. defining the parameters of each table while maintaining the structured schema.
      1. Defining the variable attributes and primary keys
2. Once the structure has been created we can populate the data into the tables by effectively importing our csv files to their respective tables
   1. Importing the files by hovering mouse over table and “import/export” data
      1. The csv files should be imported in an order that is recognized
         1. For our case we will import in the following order:
            1. Category
            2. Contacts
            3. Subcategory
            4. Campaign
3. Check the tables generated and the data within them and cross-reference the csv files.
   1. Can be done with:
      1. SELECT \* FROM (reference table name);
         1. End with semicolon to close query

# END