1. Have one member of your group create a new repository, named Crowdfunding\_ETL, for this project. Add your partner as a collaborator. **Do not add this project to an existing repository**.
2. Clone the new repository to your computer.
3. Have one person rename the ETL\_Mini\_Project\_starter\_code.ipynb file with the first name initial and last name of each member of the group, for example, ETL\_Mini\_Project\_NRomanoff\_JSmith.ipynb. Then, add this Jupyter notebook file and the Resources folder containing the crowdfunding.xlsx and the contacts.xlsx files to your repository.
4. Push the changes to GitHub.
5. Have your partner pull the changes, so both of you have the same notebook available on your computer.
6. As you work through the project deliverables, you may find it helpful to break up the work across other notebooks that you each work on individually. However, once complete, please combine all the subsections back into the final ETL\_Mini\_Project notebook.

### **Instructions**

The instructions for this mini project are divided into the following subsections:

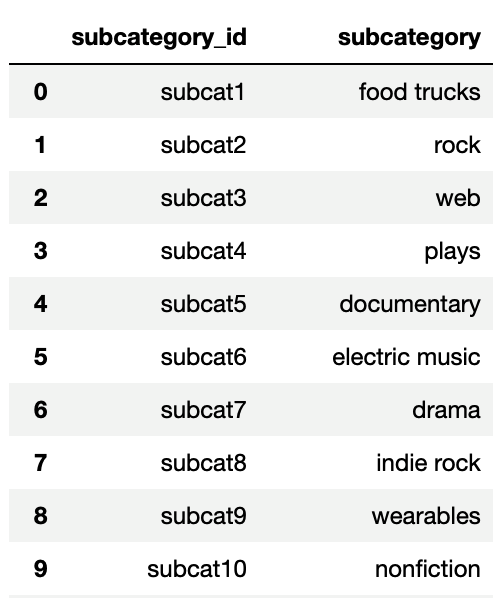
* Create the Category and Subcategory DataFrames
* Create the Campaign DataFrame
* Create the Contacts DataFrame
* Create the Crowdfunding Database

#### **Create the Category and Subcategory DataFrames**

1. Extract and transform the crowdfunding.xlsx Excel data to create a category DataFrame that has the following columns:
   * A "category\_id" column that has entries going sequentially from "cat1" to "cat*n*", where *n* is the number of unique categories
   * A "category" column that contains only the category titles
   * The following image shows this category DataFrame:



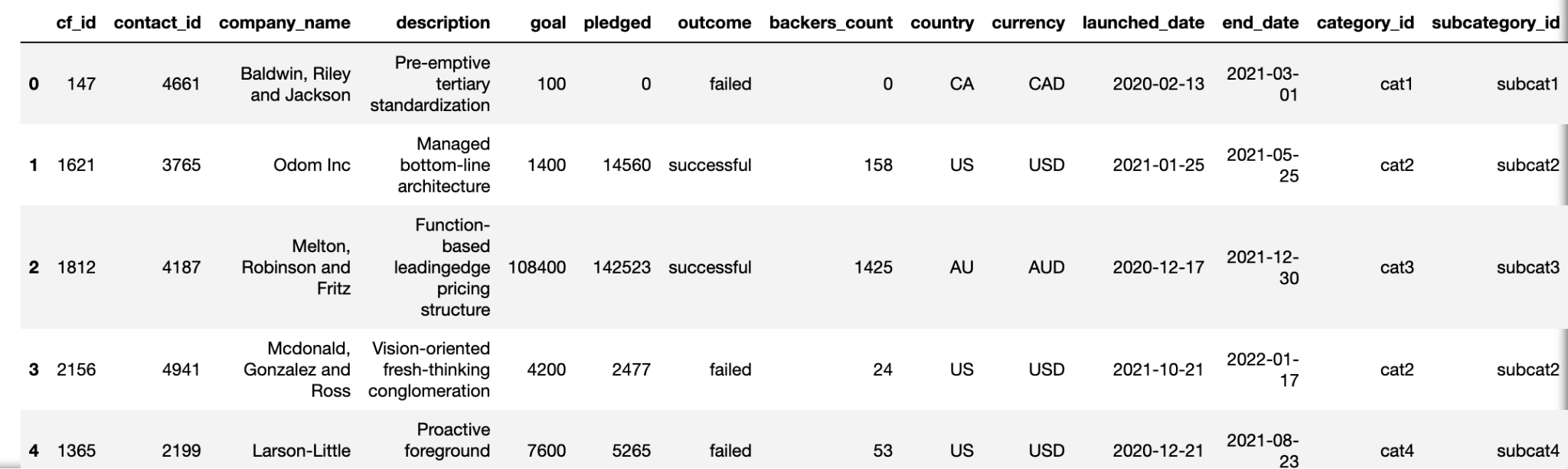
1. Export the category DataFrame as category.csv and save it to your GitHub repository.
2. Extract and transform the crowdfunding.xlsx Excel data to create a subcategory DataFrame that has the following columns:
   * A "subcategory\_id" column that has entries going sequentially from "subcat1" to "subcat*n*", where *n* is the number of unique subcategories
   * A "subcategory" column that contains only the subcategory titles
   * The following image shows this subcategory DataFrame:



4.Export the subcategory DataFrame as subcategory.csv and save it to your GitHub repository.

#### **Create the Campaign DataFrame**

1. Extract and transform the crowdfunding.xlsx Excel data to create a campaign DataFrame has the following columns:
   * The "cf\_id" column
   * The "contact\_id" column
   * The "company\_name" column
   * The "blurb" column, renamed to "description"
   * The "goal" column, converted to the float data type
   * The "pledged" column, converted to the float data type
   * The "outcome" column
   * The "backers\_count" column
   * The "country" column
   * The "currency" column
   * The "launched\_at" column, renamed to "launch\_date" and with the UTC times converted to the datetime format
   * The "deadline" column, renamed to "end\_date" and with the UTC times converted to the datetime format
   * The "category\_id" column, with unique identification numbers matching those in the "category\_id" column of the category DataFrame
   * The "subcategory\_id" column, with the unique identification numbers matching those in the "subcategory\_id" column of the subcategory DataFrame
   * The following image shows this campaign DataFrame:



#### **Create the Crowdfunding Database**

1. Inspect the four CSV files, and then sketch an ERD of the tables by using [QuickDBD Links to an external site.](http://www.quickdatabasediagrams.com/)
2. Use the information from the ERD to create a table schema for each CSV file.  
   **Note:** Remember to specify the data types, primary keys, foreign keys, and other constraints.
3. Save the database schema as a Postgres file named crowdfunding\_db\_schema.sql, and save it to your GitHub repository.
4. Create a new Postgres database, named crowdfunding\_db.
5. Using the database schema, create the tables in the correct order to handle the foreign keys.
6. Verify the table creation by running a SELECT statement for each table.
7. Import each CSV file into its corresponding SQL table.
8. Verify that each table has the correct data by running a SELECT statement for each.

#### **Hints**

* 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.
* To get the unique category and subcategory values from the "category" and "subcategory" columns, create a NumPy array where the array length equals the number of unique categories and unique subcategories from each column. For information about how to do so, see [numpy.arange Links to an external site.](https://numpy.org/doc/stable/reference/generated/numpy.arange.html)
* in the NumPy documentation.
* To create the category and subcategory identification numbers, use a list comprehension to add the "cat" string or the "subcat" string to each number in the category or the subcategory array, respectively.
* For more information about creating a new Pandas DataFrame, see the [pandas.DataFrame Links to an external site.](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html)
* in the Pandas documentation.
* To convert the "goal" and "pledged" columns to the float data type, use the astype() method.
* To convert the "launch\_date" and "end\_date" UTC times to the datetime format, see the Transform\_Grocery\_Orders\_Solved.ipynb activity solution.
* For more information about how to add the "category\_id" and "subcategory\_id" unique identification numbers to the campaign DataFrame, see the [pandas.DataFrame.merge Links to an external site.](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.merge.html)
* in the Pandas documentation.

### **Requirements**

#### **A Category DataFrame is Created (15 points)**

* The DataFrame contains a "category\_id" column that has entries going sequentially from "cat1" to "cat*n*", where *n* is the number of unique categories (5 points)
* The DataFrame has a "category" column that contains only the category titles (5 points)
* The category DataFrame is exported as category.csv (5 points)

#### **A Subcategory DataFrame is Created (15 points)**

* The DataFrame contains a "subcategory\_id" column that has entries going sequentially from "subcat1" to "subcat*n*", where *n* is the number of unique subcategories (5 points)
* The DataFrame contains a "subcategory" column that contains only the subcategory titles (5 points)
* The subcategory DataFrame is exported as subcategory.csv (5 points)

#### **A Campaign DataFrame is Created (30 points)**

* The DataFrame has the following columns: (25 points)
  + A "cf\_id" column
  + A "contact\_id" column
  + A "company\_name" column
  + A "description" column
  + A "goal" column that is a float data type
  + A "pledged" column that is a float data type
  + An "outcome" column
  + A "backers\_count" column
  + A "country" column
  + A "currency" column
  + A "launch\_date" with the time formatted as "YYYY-MM-DD"
  + An "end\_date" with the time formatted as "YYYY-MM-DD"
  + A "category\_id" column that contains the unique identification numbers matching those in the "category\_id" column of the category DataFrame
  + A "subcategory\_id" column that contains the unique identification numbers matching those in the "subcategory\_id" column of the subcategory DataFrame
* The campaign DataFrame is exported as campaign.csv (5 points)

#### **A Contacts DataFrame is Created (15 points)**

* The DataFrame has the following columns: (10 points)
  + A "contact\_id" column
  + A "first\_name" column
  + A "last\_name" column
  + An "email" column
* The contacts DataFrame is exported as contacts.csv (5 points)

#### **A Crowdfunding Database is Created (25 points)**

* A database schema labeled, crowdfunding\_db\_schema.sql is created (5 points)
* A crowdfunding\_db is created using the crowdfunding\_db\_schema.sql file (5 points)
* The database has the appropriate primary and foreign keys and relationships (5 points)
* Each CSV file is imported into the appropriate table without errors (5 points)
* The data from each table is displayed using a SELECT \* statement (5 points)