

Project 2

Due Wednesday by 11:59pm **Points** 100
Submitting a text entry box or a website url

For the ETL mini project, you will work with a partner to practice building an ETL pipeline using Python, Pandas, and either Python dictionary methods or regular expressions to extract and transform the data. After you transform the data, you'll create four CSV files and use the CSV file data to create an ERD and a table schema. Finally, you'll upload the CSV file data into a Postgres database.

Since this is a one-week project, make sure that you have done at least half of your project before the third day of class to stay on track.

Although you and your partner will divide the work, it's essential to collaborate and communicate while working on different parts of the project. Be sure to check in with your partner regularly and offer support.

Files

Download the starter code and files to help you get started:

[Project 2 ETL files](#) 

Before You Begin

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 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:

| | category_id | category |
|---|-------------|--------------|
| 0 | cat1 | food |
| 1 | cat2 | music |
| 2 | cat3 | technology |
| 3 | cat4 | theater |
| 4 | cat5 | film & video |
| 5 | cat6 | publishing |
| 6 | cat7 | games |
| 7 | cat8 | photography |
| 8 | cat9 | journalism |

2. Export the category DataFrame as `category.csv` and save it to your GitHub repository.

3. 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:

| | subcategory_id | subcategory |
|---|----------------|----------------|
| 0 | subcat1 | food trucks |
| 1 | subcat2 | rock |
| 2 | subcat3 | web |
| 3 | subcat4 | plays |
| 4 | subcat5 | documentary |
| 5 | subcat6 | electric music |
| 6 | subcat7 | drama |
| 7 | subcat8 | indie rock |
| 8 | subcat9 | wearables |
| 9 | subcat10 | nonfiction |

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:

| | cf_id | contact_id | company_name | description | goal | pledged | outcome | backers_count | country | currency | launched_date | end_date | category_id | subcategory_id |
|---|-------|------------|-----------------------------|---|--------|---------|------------|---------------|---------|----------|---------------|------------|-------------|----------------|
| 0 | 147 | 4661 | Baldwin, Riley and Jackson | Pre-emptive tertiary standardization | 100 | 0 | failed | 0 | CA | CAD | 2020-02-13 | 2021-03-01 | cat1 | subcat1 |
| 1 | 1621 | 3765 | Odom Inc | Managed bottom-line architecture | 1400 | 14560 | successful | 158 | US | USD | 2021-01-25 | 2021-05-25 | cat2 | subcat1 |
| 2 | 1812 | 4187 | Melton, Robinson and Fritz | Function-based leadingedge pricing structure | 108400 | 142523 | successful | 1425 | AU | AUD | 2020-12-17 | 2021-12-30 | cat3 | subcat1 |
| 3 | 2156 | 4941 | McDonald, Gonzalez and Ross | Vision-oriented fresh-thinking conglomeration | 4200 | 2477 | failed | 24 | US | USD | 2021-10-21 | 2022-01-17 | cat2 | subcat1 |
| 4 | 1365 | 2199 | Larson-Little | Proactive foreground | 7600 | 5265 | failed | 53 | US | USD | 2020-12-21 | 2021-08-23 | cat4 | subcat1 |


2. Export the campaign DataFrame as `campaign.csv` and save it to your GitHub repository.

Create the Contacts DataFrame

1. Choose one of the following two options for extracting and transforming the data from the `contacts.xlsx` Excel data:
 - **Option 1:** Use Python dictionary methods.
 - **Option 2:** Use regular expressions.
2. If you chose Option 1, complete the following steps:
 - Import the `contacts.xlsx` file into a DataFrame.
 - Iterate through the DataFrame, converting each row to a dictionary.
 - Iterate through each dictionary, doing the following:
 - Extract the dictionary values from the keys by using a Python list comprehension.
 - Add the values for each row to a new list.
 - Create a new DataFrame that contains the extracted data.
 - Split each "name" column value into a first and last name, and place each in a new column.
 - Clean and export the DataFrame as `contacts.csv` and save it to your GitHub repository.
3. If you chose Option 2, complete the following steps:
 - Import the `contacts.xlsx` file into a DataFrame.
 - Extract the "contact_id", "name", and "email" columns by using regular expressions.
 - Create a new DataFrame with the extracted data.
 - Convert the "contact_id" column to the integer type.
 - Split each "name" column value into a first and a last name, and place each in a new column.
 - Clean and then export the DataFrame as `contacts.csv` and save it to your GitHub repository.
4. Check that your final DataFrame resembles the one in the following image:

| | contact_id | first_name | last_name | email |
|---|------------|------------|-----------|------------------------------|
| 0 | 4661 | Cecilia | Velasco | cecilia.velasco@rodrigues.fr |
| 1 | 3765 | Mariana | Ellis | mariana.ellis@rossi.org |
| 2 | 4187 | Sofie | Woods | sofie.woods@riviere.com |
| 3 | 4941 | Jeanette | Iannotti | jeanette.iannotti@yahoo.com |
| 4 | 2199 | Samuel | Sorgatz | samuel.sorgatz@gmail.com |
| 5 | 5650 | Socorro | Luna | socorro.luna@hotmail.com |
| 6 | 5889 | Carolina | Murray | carolina.murray@knight.com |
| 7 | 4842 | Kayla | Moon | kayla.moon@yahoo.de |
| 8 | 3280 | Ariadna | Geisel | ariadna.geisel@rangel.com |
| 9 | 5468 | Danielle | Ladeck | danielle.ladeck@scalfaro.net |

Create the Crowdfunding Database

1. Inspect the four CSV files, and then sketch an ERD of the tables by using [QuickDBD](#) .
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](#) in the NumPy documentation.
- To create the category and subcategory identification numbers, use a list comprehension to add the "cat" string to 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](#) 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](#) in the Pandas documentation.

Support and Resources

Your instructional team will provide support during classes and office hours. You will also have access to learning assistants and tutors to help you with topics as needed. Make sure to take advantage of these resources as you collaborate with your partner on this project.

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)

This project will be evaluated against the requirements and assigned a grade according to the following table:

| Grade | Points |
|---------|--------|
| A (+/-) | 90+ |

| Grade | Points |
|---------|--------|
| B (+/-) | 80–89 |
| C (+/-) | 70–79 |
| D (+/-) | 60–69 |
| F (+/-) | < 60 |

Submission

You are required to submit the URL of your GitHub repository for grading.

NOTE

Projects are requirements for graduation. While you are allowed to miss up to two Challenge assignments and still earn your certificate, projects cannot be skipped.