Crowdfunding ETL Mini Project Starter Code

Explanation of the Data Processing Logic

This section describes the process of transforming data from two Excel files and preparations for export to CSV, forming the basis for a later database import.

1. Importing and Setting Up:

- This step involves importing the necessary Python libraries: pandas for data manipulation, numpy for numerical operations, textwrap for formatting text, datetime for date handling, json for dealing with JSON objects, re for regular expressions, and os for system interactions (though it's not used for data processing).
- It also sets a pandas display option to ensure column widths are readable.

2. Extracting Data from the Excel File:

- The crowdfunding.xlsx Excel file is loaded into a pandas DataFrame using pd.read_excel(). This prepares the data for manipulation.
- The code then inspects the DataFrame using display(dataframe.head()) to check the raw data, using dataframe.info() to check for data types and number of non-null columns, and print(dataframe.columns) to view the column names for reference.

3. Assigning Category and Subcategory Values:

- The code now checks whether a combined category & sub-category column exists within the imported dataframe.
- If it does exist, it performs the following operations to create two new columns, category and sub-category, using the / as the delimiter, and also replacing NaN values with the text Unknown, thereby cleaning the data.
- It extracts a list of the unique values within the category and subcategory columns to use later for ID mapping.

4. Defining a Formatting Function:

- A utility function called format_list_output is defined, and its purpose is to take a list of values and format them into a wrapped string, so that it can fit within the display window, with a line length that is defined by a width parameter, and indented to be easily read. This is a formatting tool.

5. Using the Formatting Function and Counting Unique Items:

- The code then uses the format_list_output function to format the lists of unique categories and subcategories.
- It counts and prints the total number of unique categories and subcategories by using the len method.

6. Creating Numerical Arrays:

- Two numpy arrays are created, category_ids and subcategory_ids, for the purpose of assigning numerical ids to each category and subcategory. The first array counts from 1-9, while the second counts from 1-24.

7. Creating String-Based IDs:

- Using list comprehensions, the code creates lists of category IDs by concatenating `"cat"` to each number in the `category_ids` array, and then subcategory IDs with `subcat` added to the `subcategory_ids` array.

8. Creating the Category and Subcategory DataFrames:

- Two DataFrames, category_df and subcategory_df are constructed using the categories and subcategories lists, with the id lists to create a dataframe structure.
  - These DataFrames have two columns: "category_id" or "subcategory_id" which are created from the ids, and "category" or "subcategory" which are created from the list of unique values.
- Then, the DataFrames are displayed in the notebook using display().

9. Exporting Category and Subcategory DataFrames:

- These two DataFrames are exported to CSV files, category.csv and subcategory.csv, with a specified path in the Resources folder, and using to_csv, and the index is specifically excluded from the outputted CSV.

10. Setting up the Campaign DataFrame:

- A new DataFrame named campaign_df is created as a copy of the crowdfunding_info_df. The head of the dataframe is shown.
- The code renames the columns: blurb to description, launched_at to launched_date, and deadline to end_date, using rename(). The head of the dataframe is shown again.
  - The goal and pledged columns are cast as float type using the astype() method and the head is displayed once again.
- The launched_date and end_date columns are formatted into the correct datetypes using pd.to_datetime() and are truncated to just show the dates. Then the dataframe head is displayed once more.

11. Mapping the Categories and Subcategories and Merging DataFrames:

- The unique lists of categories and subcategories are mapped to their respective category and subcategory ids, and are then cast as strings before the merge.
- The code merges the campaign_df with the category_df on the category_id column, and merges with subcategory_df on subcategory_id column.
- This creates the structure with the ids in a way that allows the merge operation to run, and ensures the merged DataFrame contains all columns.

12. Dropping Unwanted Columns:

- The code then removes a selection of unwanted columns, named staff_pick, spotlight, category & sub-category, category, and subcategory from the campaign_merged_df and puts this result into a new dataframe named campaign_cleaned, and the tail of this new dataframe is displayed to the console.

13. Exporting the Campaign DataFrame:

- Finally, the campaign_cleaned DataFrame is exported to a CSV file named campaign.csv in the "Resources" directory, while excluding the index.

Explanation of Database Setup:

1. **SQLAlchemy Setup**: Sets up the database connection and creates the tables, including data types and constraints, based on the ERD.
2. **Schema File Output:** Generates the crowdfunding_db_schema.sql file with the table creation commands.
3. **Database and Table Creation:** Establishes the database connection and creates the tables.
4. **Verification Queries:** Queries are added to view the dataframes are created.
5. **CSV Data Import:** Imports the CSV files into their corresponding tables using .to_sql.
6. **Verification**: The last queries verify that the data was uploaded correctly.