Crowdfunding ETL SQL DB Setup

Explanation:

1. **target_dir Variable:** The path to the directory is defined by a variable so it can easily be changed, and can be modified to run on local and remote databases as well.
2. **os.path.join():** The script now uses os.path.join(target_dir, 'filename.ext') to correctly create the full file paths for both the SQL and JSON files. This will make the script more platform independent as well as easy to use.
3. **Directory Creation:** Uses os.makedirs(target_dir, exist_ok=True) to make sure that the output directory will always exist.
4. **Import paths:** Updates the import paths so they make use of the os.path.join function.
5. **Database Connection String** The database connection string is now saved into the variable db_string, this allows the database location to be easily changed to any database (local or remote) as needed.
6. **Database Creation Check:** The script now explicitly checks for the existence of the crowdfunding_db database. If it does not exist, it creates the database before setting up the tables.
7. **Postgres Schema Export:**
    - After creating the tables, the code now opens crowdfunding_db_schema.sql in write mode ('w').
    - It iterates through the table metadata using metadata.sorted_tables to respect foreign key order.
    - For each table it generates a CREATE TABLE statement using sqlalchemy.schema.CreateTable(table).compile(engine) to generate the SQL create table command, and then writes the command into the schema file.
    - A message indicating that the schema has been saved is printed to the console.

3. **infer_mongodb_schema(df) Function:**
    - Takes a Pandas DataFrame as input.
    - Iterates over each column in the DataFrame.
    - Inspects the data type of the column using .dtype.
        - Assigns int, float, or string as the column type based on the Pandas dtype.
    - Stores the inferred column types in a dictionary
    - Returns the schema dictionary.

4. **generate_mongodb_schemas(category_df, subcategory_df, contacts_df, campaign_cleaned) Function:**
    - Takes the four dataframes as an input
    - Runs the function to generate the schemas for each dataframe
    - Returns a dictionary of the form {"<dataframe name>_schema": <schema generated>}
5. **Main execution block:**
    - Assumes that you have already loaded your CSV files into Pandas DataFrames named category_df, subcategory_df, contacts_df, campaign_df as done in the previous notebook.
    - Calls generate_mongodb_schemas to get the schemas.
    - Opens a file named crowdfunding_db_schema.json in write mode ('w').
    - Uses json.dump() to write the schema dictionary to the file in a formatted (indented) JSON structure.
    - Prints a message that the file has been saved.

Key Features:

- **Error Handling:** Included try-except blocks to handle potential issues during database creation/connection and table operations.
- **Clearer Table Definitions:** Made the SQLAlchemy table definitions clearer, specifying primary keys, data types, and foreign key relationships
- **Proper Table Creation Order:** Drops tables first, then creates them ensuring no issues with foreign key references.
- **Column Mapping:** Implemented the conversion of the categorical columns using the dictionary mappings.
- **Data Verification:** The script now includes print statements to verify that the data has been imported and manipulated correctly, by printing the top 5 and bottom 5 entries as well as the data type, of each table.
- **Schema Creation:** Included the SQL file for creating the schema, and instructions for applying it.
- **Schema Saved:** The schema is now saved to a file named crowdfunding_db_schema.sql as requested.
- **Type Inference:** The code attempts to infer a type that MongoDB might use (like string, int, float, date).
- **Date Inference:** Checks if string is a date and sets type to date if so.

- **JSON Output:** The schema is written to a JSON file (crowdfunding_db_schema.json), making it easy to share and use with MongoDB-related tools or documentation.
- **Simplified Type Handling:** Makes use of startswith for better and more reliable type handling.
- **json Import:** Added import json to include the json library functionality.
- **Schema Generation Placement:** The MongoDB schema generation is now right after the database tables are created using SQLAlchemy, and prior to any data imports.
- **Dataframe Manipulation:** The Dataframe manipulation is now included as needed by the MongoDB schema generation.

ERD Sketch (Conceptual)

Here's a conceptual sketch of the ERD based on the CSV files:

- **category Table:**
    - category_id (Primary Key)
    - category
- **subcategory Table:**
    - subcategory_id (Primary Key)
    - subcategory
- **contacts Table:**
    - contact_id (Primary Key)
    - first_name
    - last_name
    - email
- **campaign Table:**
    - cf_id (Primary Key)
    - contact_id (Foreign Key to contacts)
    - company_name
    - description
    - goal
    - pledged
    - outcome

- backers_count
- country
- currency
- launched_date
- end_date
- category_id (Foreign Key to category)
- subcategory_id (Foreign Key to subcategory)