Crowdfunding ETL Write Up Group #1

Understanding the ETL Mini Project

In our recent project, we teamed up to tackle an ETL (Extract, Transform, Load) challenge using Python and Pandas. The main goal was to process data from Excel files (`crowdfunding.xlsx` and `contacts.xlsx`), transform it into structured CSV files, design a database schema, and then populate a PostgreSQL database named `crowdfunding_db`.

Getting Started

To kick things off, we set up a dedicated GitHub repository named `Crowdfunding_ETL`. This repository served as our central workspace, where we stored all project files and collaborated on the project tasks. By adding each other as collaborators, we ensured seamless teamwork and efficient communication throughout the project.

Breakdown of Tasks and Timeline

The project timeline spanned approximately one week, with a goal to complete at least half of the tasks by the third day. This approach helped us maintain a steady pace and allowed ample time for refining our work, troubleshooting any issues, and finalizing all deliverables.

Data Transformation Process

Our first major task was to extract and transform data from `crowdfunding.xlsx` to create two critical datasets: `category.csv` and `subcategory.csv`. These CSV files were structured to include sequential IDs (`cat1`, `cat2`, etc., and `subcat1`, `subcat2`, etc.) and their respective category and subcategory titles.

Following that, we processed the same Excel file to generate the `campaign.csv` file. This dataset contained comprehensive details about each crowdfunding campaign, such as IDs, company names, funding goals, pledged amounts, outcomes, backers count, geographical details, currencies, launch dates, end dates, and crucially, linked to the `category.csv` and `subcategory.csv` via unique identification numbers.

For handling `contacts.xlsx`, we opted to use Python's dictionary methods for data extraction and transformation. This approach allowed us to effectively split each contact's name into first and last names, alongside their email addresses, resulting in the creation of the `contacts.csv` file.

Designing and Implementing the Database Schema

With our CSV files ready, we moved on to the database design phase. Using QuickDBD, we visually mapped out an Entity-Relationship Diagram (ERD) that illustrated the relationships between our tables (`categories`, `subcategories`, `campaigns`, and `contacts`). This step was crucial for ensuring the integrity and efficiency of our database structure.

Next, we translated the ERD into a PostgreSQL schema script (`crowdfunding_db_schema.sql`). This script meticulously defined each table, specified their columns, established primary and foreign keys, and set up any necessary constraints to maintain data consistency and relational integrity across the database.

Deploying and Verifying the Database

With our schema script in hand, we proceeded to create the `crowdfunding_db` in PostgreSQL. Following the order defined in our schema script, we systematically created each table to ensure that all dependencies were properly managed.

To validate our database setup, we ran `SELECT *` queries on each table. This final step confirmed that data from our CSV files had been successfully imported into the respective database tables without any discrepancies.

Leveraging Support and Resources

Throughout the project, we made the most of available resources such as classroom sessions, office hours, and guidance from teaching assistants and tutors. These resources played a pivotal role in clarifying concepts, troubleshooting technical challenges, and ultimately ensuring the successful completion of our project objectives.

Deliverables and Evaluation

Our final deliverables included the meticulously prepared CSV files (`category.csv`, `subcategory.csv`, `campaign.csv`, `contacts.csv`) and the comprehensive `crowdfunding_db_schema.sql` script, all securely uploaded to our GitHub repository. The project was evaluated based on our adherence to requirements, accuracy in data transformation, completeness of the database schema, and the successful implementation of the database with verified data insertion and retrieval capabilities.

By navigating through these tasks collaboratively and utilizing the resources effectively, we not only enhanced our proficiency in data handling with Python and Pandas but also gained practical experience in database management using PostgreSQL—a valuable skill set for our future endeavors in data engineering and analytics.