In this group project we worked on taking two excel files and creating a pipeline off of those files to load them into both PgAdmin and Jupyter Notebook so that we could query off those excel files and do analysis on the given data. We started with creating our ERD diagrams so that we could create the tables necessary in PgAdmin so we could later load the data into those tables. Then we took the data itself and cleaned it to our liking by adding and deleting columns, changing data types, etc.. After that we then loaded the cleaned up data into the previously created database and tables in PgAdmin. Finally, we decided on six questions we wanted to answer with the data we were given.

1. Which category pledged the most
2. Which subcategory pledged the most
3. Who pledged the most money individually
4. Who pledged the most in the Plays subcategory
5. Did each category meet/exceed/miss their pledged goal
6. Which subcategory pledged the most for the music category

Once we had our questions defined we made a query for each one so that we could create a dataframe from the database and then create a visualization on them.

We started with creating our ERD diagram so that we could create each table and set their corresponding column names rather than creating them all in PgAdmin and typing them all out. This helped save a lot of time except for the fact where we later realized that when we created our tables, we uppercased them which caused us some troubles when trying to query in PgAdmin. After creating our ERD diagrams we exported the schema into PgAdmin to finally create the database and corresponding tables in the database. Unfortunately we had to make some changes to the ERD after we had already exported, so this caused us to delete all of the tables we had created in PgAdmin a few times and recreate them. After we finally had what we wanted for each table and column we added a line in each table for the last updated column that our ERD export didn't give us which was "default localtimestamp".

In the transformation part of the analysis, we worked on the formation of data and cleaned our dataset to show the specific information needed for this project. We started by reading in the crowdfunding excel workbook. We then split category & subcategory columns into their own respective columns. From the new columns we extracted the unique values using the .unique() function. Leading us to create new data frames by establishing new columns that used a list comprehension to create an ID column. We matched the unique values from the category and subcategory columns with these ID's. We then saved the new data frames to CSV files.

We copied the crowdfunding excel workbook and named it "campaign_df". We renamed some of the columns and changed the data types to make the data frame more readable and reflective for this dataset. We then merged our prior two data frames with our campaign data frame. We dropped the unwanted rows to include the information we wanted. With the cleaned data frame, we save this off to another CSV file.

The final portion of our transformation was to read and clean up the contacts excel spreadsheet. We started first by splitting the name column into their own separate first and last name columns. Next, we reordered the columns to show contact_id, first_name, last_name, and email. Once again save the cleaned data frame to a CSV file.

To load the data, we connected to a PostgreSQL database using SQLAlchemy and used an inspector to gather and print information about the tables and columns within the database. We then read data from contacts, campaigns, categories, and subcategories csv files located in the Crowdfunding_ETL/Resources folder into pandas dataframes.

In the process of trying to load our data we learned that PgAdmin is case sensitive. Before our data could be uploaded we needed to ensure our tables names and csv files were all lowercase. After verifying the data, we loaded each dataframe into the corresponding table to the PostGres database using the “to_sql” method.This allowed us to append the data since the tables already existed. Finally, we ensured the proper closure of the SQLAlchemy connection to maintain database integrity and resource management.

We created six queries using SQLAlchemy to answer the following questions (using matplotlib and seaborn to create visualizations for for #1, 2, 5 & 6) :

1. Which category pledged the most?: We plotted the results using a horizontal bar chart, showing that the 'theater' category received the highest total amount of pledges.
2. Which subcategory pledged the most?: For the second question we used a horizontal bar chart to show that 'plays' received the most pledges among all subcategories.
3. Who pledged the most and what are their emails for future callbacks?: The results returned the top 50 individuals who pledged the most, along with their contact information for future engagement or crowdfunding campaigns.
4. Who pledged the most in the 'plays' subcategory and what are their emails for future callbacks?: This query returned the individuals who pledged the most in the 'plays' subcategory, highlighting key contributors for targeted communication.
5. Did the categories meet or exceed their pledge goals?: The total goals and total pledges were compared for each category to determine if they met or exceeded their goals. We

created a bar chart to show that the campaigns for theater, music, technology, and journalism each exceeded their goals.

6. Which subcategory pledged more within the Music category?: We identified the subcategories within the 'music' category that received the most pledges. Two visualizations were created: a donut chart and a pie chart, highlighting that 'rock' received the most pledges within the music category.