

# Crowdfunding ETL – Project 2

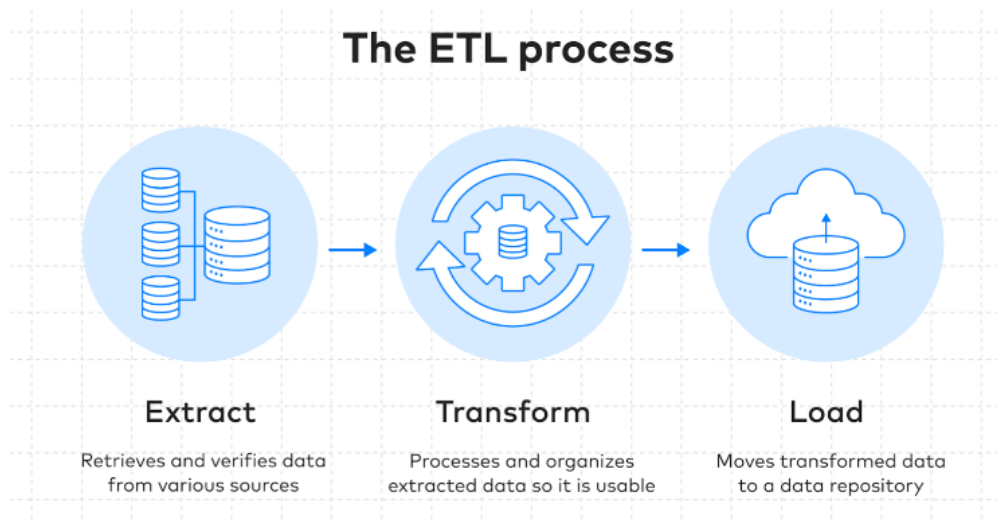
Nicholas Buse

Sam Hoemann

Marty Thompson

## Introduction

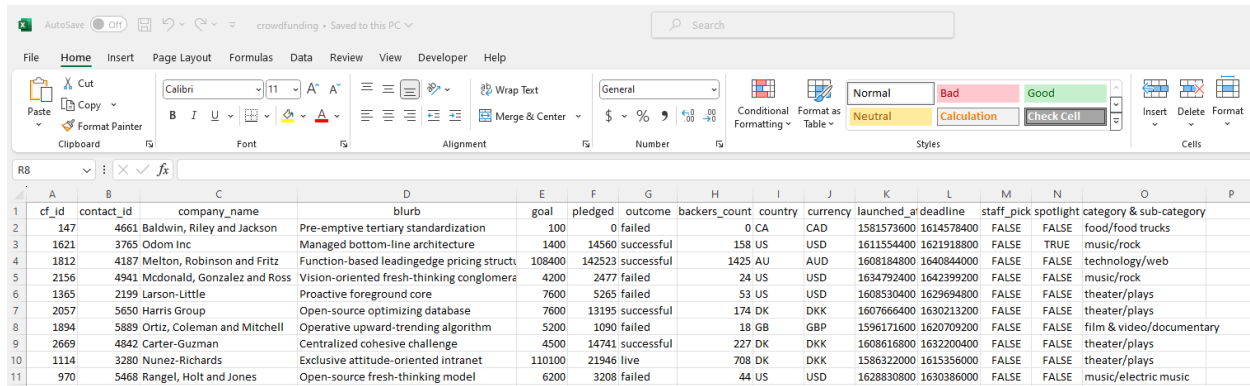
What is ETL? ETL is the process of extracting, transforming, and loading data from multiples sources into a single central data repository. This is the process of taking raw data, cleaning and organizing the information into a usable format, and then loading it into a data repository. We can then access the data to improve business intelligence, generating analytics and reporting to better steer company decisions. This project is a showcase of the ETL process, culminating in an analysis of the data using SQL queries and data visualizations.



## Extraction

In the Extraction phase, we were given raw data in the form of two excel files found in the Resources folder, to simulate retrieval of raw data from multiple sources. Each was saved in a different format for us to extract data into usable files for uploading into our database.

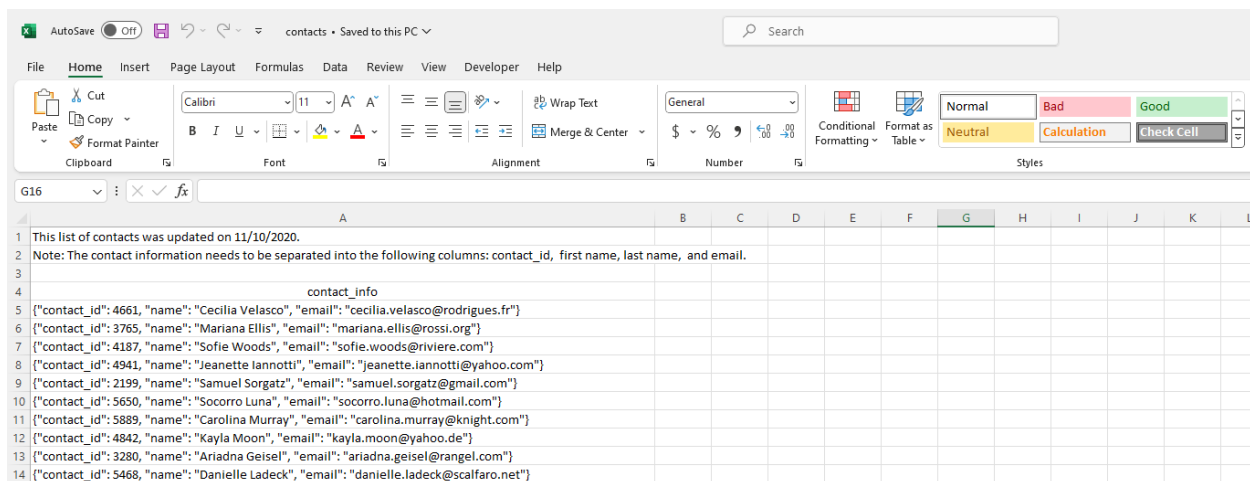
The first, was “Resources/crowdfunding”:



cf_id	contact_id	company_name	blurb	goal	pledged	outcome	backers_count	country	currency	launched_at	deadline	staff_pick	spotlight	category & sub-category
147	4661	Baldwin, Riley and Jackson	Pre-emptive tertiary standardization	100	0	failed	0	CA	CAD	1581573600	1614578400	FALSE	FALSE	food/food trucks
1621	3765	Odom Inc	Managed bottom-line architecture	1400	14560	successful	158	US	USD	1611554400	1621918800	FALSE	TRUE	music/rock
1812	4187	Melton, Robinson and Fritz	Function-based leadingedge pricing structu	108400	142523	successful	1425	AU	AUD	1608184800	1640844000	FALSE	FALSE	technology/web
2156	4941	Mcdonald, Gonzalez and Ross	Vision-oriented fresh-thinking conglomerate	4200	2477	failed	24	US	USD	1634792400	1642399200	FALSE	FALSE	music/rock
1365	2199	Larson-Little	Proactive foreground core	7600	5265	failed	53	US	USD	1608530400	1629694800	FALSE	FALSE	theater/plays
2057	5650	Harris Group	Open-source optimizing database	7600	13195	successful	174	DK	DKK	1607666400	1630213200	FALSE	FALSE	theater/plays
1894	5889	Ortiz, Coleman and Mitchell	Operative upward-trending algorithm	5200	1090	failed	18	GB	GBP	1596171600	1620709200	FALSE	FALSE	film & video/documentary
2669	4842	Carter-Guzman	Centralized cohesive challenge	4500	14741	successful	227	DK	DKK	1608616800	1632200400	FALSE	FALSE	theater/plays
1114	3280	Nunez-Richards	Exclusive attitude-oriented intranet	110100	21946	live	708	DK	DKK	1586322000	1615356000	FALSE	FALSE	theater/plays
970	5468	Rangel, Holt and Jones	Open-source fresh-thinking model	6200	3208	failed	44	US	USD	1628830800	1630386000	FALSE	FALSE	music/electric music

This excel format had information clearly separated by columns. We were to use this file to separate information needed to generate our category, subcategory, and campaign tables. Our contacts table would require the second resource file, saved in JSON format.

The second, was “Resources/contacts”:

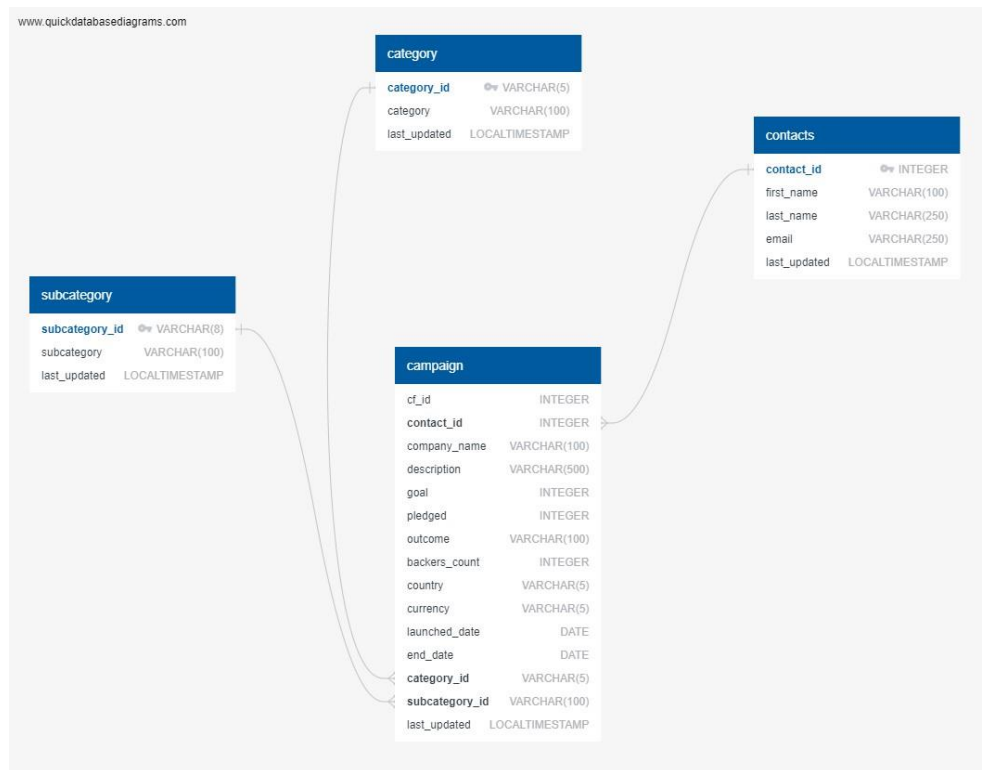


A	B	C	D	E	F	G	H	I	J	K	L
This list of contacts was updated on 11/10/2020.											
Note: The contact information needs to be separated into the following columns: contact_id, first name, last name, and email.											
contact_info											
{ "contact_id": 4661, "name": "Cecilia Velasco", "email": "cecilia.velasco@rodrigues.fr" }											
{ "contact_id": 3765, "name": "Mariana Ellis", "email": "mariana.ellis@rossi.org" }											
{ "contact_id": 4187, "name": "Sofie Woods", "email": "sofie.woods@riviere.com" }											
{ "contact_id": 4941, "name": "Jeanette Iannotti", "email": "jeanette.iannotti@yahoo.com" }											
{ "contact_id": 2199, "name": "Samuel Sorgatz", "email": "samuel.sorgatz@gmail.com" }											
{ "contact_id": 5650, "name": "Socorro Luna", "email": "socorro.luna@hotmail.com" }											
{ "contact_id": 5889, "name": "Carolina Murray", "email": "carolina.murray@knight.com" }											
{ "contact_id": 4842, "name": "Kayla Moon", "email": "kayla.moon@yahoo.de" }											
{ "contact_id": 3280, "name": "Ariadna Geisel", "email": "ariadna.geisel@rangel.com" }											
{ "contact_id": 5468, "name": "Danielle Ladeck", "email": "danielle.ladeck@scalfaro.net" }											

## Transformation

After receiving our extracted data, we then transitioned into the Transformation phase. To begin this phase, we designed a table schema for our database using QuickDBD (quickdatabasediagram.com). As part of this creation, we generated an Entity Relationship Diagram to isolate necessary information for each required table and a SQL Source File for the scripting necessary to create tables within our database software, Postgres.

## Entity Relationship Diagram, “QUICKDBD/QuickDBD-Project 2.png”



## Table Creation Scripting, “QUICKDBD/ QuickDBD-Project 2.sql”

```
1 -- Exported from QuickDBD: https://www.quickdatabasediagrams.com/
2 -- Link to schema: https://app.quickdatabasediagrams.com/#/d/Rxx2KH
3 -- NOTE! If you have used non-SQL datatypes in your design, you will have to change these here.
4
5
6 CREATE TABLE "category" (
7     "category_id" VARCHAR(5) NOT NULL,
8     "category" VARCHAR(100) NOT NULL,
9     "last_updated" TIMESTAMPTZ DEFAULT LOCALTIMESTAMP NOT NULL,
10    CONSTRAINT "pk_category" PRIMARY KEY (
11        "category_id"
12    )
13);
14
15 CREATE TABLE "subcategory" (
16     "subcategory_id" VARCHAR(8) NOT NULL,
17     "subcategory" VARCHAR(100) NOT NULL,
18     "last_updated" TIMESTAMPTZ DEFAULT LOCALTIMESTAMP NOT NULL,
19    CONSTRAINT "pk_subcategory" PRIMARY KEY (
20        "subcategory_id"
21    )
22);
23
24 CREATE TABLE "contacts" (
25     "contact_id" INTEGER NOT NULL,
26     "first_name" VARCHAR(100) NOT NULL,
27     "last_name" VARCHAR(250) NOT NULL,
28     "email" VARCHAR(250) NOT NULL,
29     "last_updated" TIMESTAMPTZ DEFAULT LOCALTIMESTAMP NOT NULL,
30    CONSTRAINT "pk_contacts" PRIMARY KEY (
31        "contact_id"
32    )
33);
34
35 CREATE TABLE "campaign" (
36     "cf_id" INTEGER NOT NULL,
37     "contact_id" INTEGER NOT NULL,
38     "company_name" VARCHAR(100) NOT NULL,
```

The next step in our Transformation phase was data cleaning and separation. To do this, we used Jupiter Notebook. All of our transformation scripting was done in “ELT\_Mini\_Project\_Starter\_Code.ipynb”. We used this notebook to transform the raw data into four CSV (comma-separated values) files that we would later use to load data into our Postgres tables.

For the category and subcategory tables, we split the “category & sub-category” column into two columns, created an ID column, created DataFrames for each table, and exported the DataFrames as CSVs.

Split Columns:

```
# Assign the category and subcategory values to category and subcategory columns.
cdf[["category", "subcategory"]] = cdf['category & sub-category'].str.split("/", expand=True)
cdf.head()
```

Created an ID column:

```
# Create numpy arrays from 1-9 for the categories and 1-24 for the subcategories.
category_ids = np.arange(1, 10)
subcategory_ids = np.arange(1, 25)

print(category_ids)
print(subcategory_ids)
```

```
[1 2 3 4 5 6 7 8 9]
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]
```

[+ Code](#)

```
# Use a list comprehension to add "cat" to each category_id.
cat_ids = ['cat' + str(num) for num in category_ids]
# Use a list comprehension to add "subcat" to each subcategory_id.
scat_ids = ['subcat' + str(num) for num in subcategory_ids]

print(cat_ids)
print(scat_ids)
```

```
['cat1', 'cat2', 'cat3', 'cat4', 'cat5', 'cat6', 'cat7', 'cat8', 'cat9']
['subcat1', 'subcat2', 'subcat3', 'subcat4', 'subcat5', 'subcat6', 'subcat7', 'subcat8', 'subcat9', 'subcat10',
```

◀

Created DataFrames for each table and saved them as CSVs:

```
▷ # Create a category DataFrame with the category_id array as the category_id and categories list as the category name.
category_df = {
    "category_id": cat_ids,
    "category": categories
}
category_df = pd.DataFrame(category_df)

# Create a category DataFrame with the subcategory_id array as the subcategory_id and subcategories list as the subcategory name.
subcategory_df = {
    "subcategory_id": scat_ids,
    "subcategory": subcategories
}
subcategory_df = pd.DataFrame(subcategory_df)

[10]

# Export categories_df and subcategories_df as CSV files.
category_df.to_csv("CSV/category.csv", index=False)

subcategory_df.to_csv("CSV/subcategory.csv", index=False)

[13]
```

For the campaign table, we renamed columns to more readable titles, changed datatypes, merged with the category and subcategory tables to bring in category\_id and subcategory\_id, dropped unwanted columns, and exported to a CSV.

Renaming columns:

```
# Rename the blurb, launched_at, and deadline columns.
campaign_df = campaign_df.rename(columns={'blurb': 'description', 'launched_at': 'launched_date', 'deadline': 'end_date'})
campaign_df.head()

1
```

Changed Datatypes:

```
# Convert the goal and pledged columns to a `float` data type.
campaign_df[['goal', 'pledged']] = campaign_df[['goal', 'pledged']].astype('float')
campaign_df.head()

[17]
...

▷ # Format the launched_date and end_date columns to datetime format
from datetime import datetime as dt

campaign_df['launched_date'] = pd.to_datetime(campaign_df['launched_date'], unit='s')
campaign_df['end_date'] = pd.to_datetime(campaign_df['end_date'], unit='s')

campaign_df

[19]
...
```

Merged category and subcategory dataframes into campaign dataframe:

```
▷ # Merge the campaign_df with the category_df on the "category" column and
  # the subcategory_df on the "subcategory" column.

  campaign_merged_df = campaign_df.merge(category_df, on='category', how='inner')
  campaign_merged_df = campaign_merged_df.merge(subcategory_df, on='subcategory', how='inner')
  campaign_merged_df.tail(10)
```

[20]

...

Dropped unwanted columns and Exported to a CSV:

```
# Drop unwanted columns
campaign_cleaned = campaign_merged_df.drop(['staff_pick', 'spotlight', 'category & sub-category', 'category', 'subcategory'], axis=1)
campaign_cleaned.head()
```

[21]

...

```
# Export the DataFrame as a CSV file.
campaign_cleaned.to_csv("CSV/campaign.csv", index=False)
```

[22]

For the contacts table, appended each row to a dictionary, created a DataFrame, split the name column into first and last name, reordered columns, and exported to a CSV.

Appended rows to a dictionary:

```
▷ # Iterate through the contact_info_df and convert each row to a dictionary.
  import json
  dict_values = []

  for index, row in contact_info_df.iterrows():
      contact_dict = json.loads(row['contact_info'])
      dict_values.append(contact_dict)

  # Print out the list of values for each row.
  print(dict_values)
```

[24]

```
... [{ 'contact_id': 4661, 'name': 'Cecilia Velasco', 'email': 'cecilia.velasco@rodrigues.fr'}, {'c
```

◀ ■

Created a DataFrame:

```
> # Create a contact_info DataFrame and add each list of values, i.e., each row
# to the 'contact_id', 'name', 'email' columns.

contact_info = pd.DataFrame(dict_values)
contact_info.head()
```

5]

\*

Manipulated columns:

```
# Create a "first_name" and "last_name" column with the first and last names from the "name" column.
contact_info[["first_name", "last_name"]] = contact_info['name'].str.split(" ", expand=True)
```

27]

```
# Drop the contact_name column
contact_info = contact_info.drop(['name'], axis=1)
contact_info.head()
```

28]

..

```
# Reorder the columns
contacts_df_clean = contact_info.reindex(columns=['contact_id', 'first_name', 'last_name', 'email'])
contacts_df_clean.head()
```

29]

..

Exported to a CSV:

```
# Export the DataFrame as a CSV file.
contacts_df_clean.to_csv("CSV/contacts.csv", encoding='utf8', index=False)
```

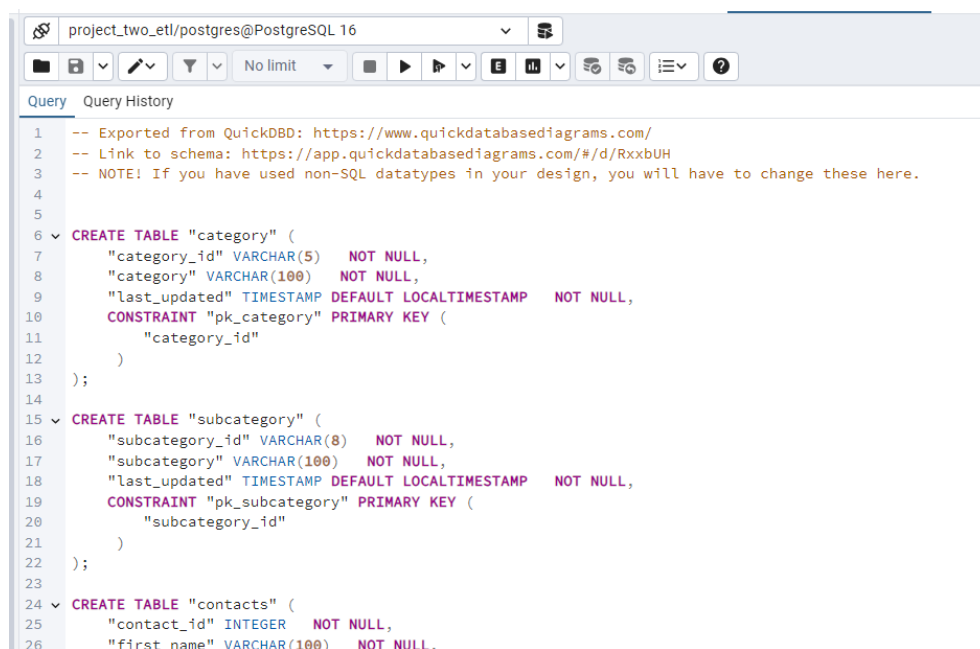
]

Created CSVs stored in CSV folder: "CSV/category.csv" "CSV/subcategory.csv"  
"CSV/campaign.csv" and "CSV/contacts.csv"

## Load

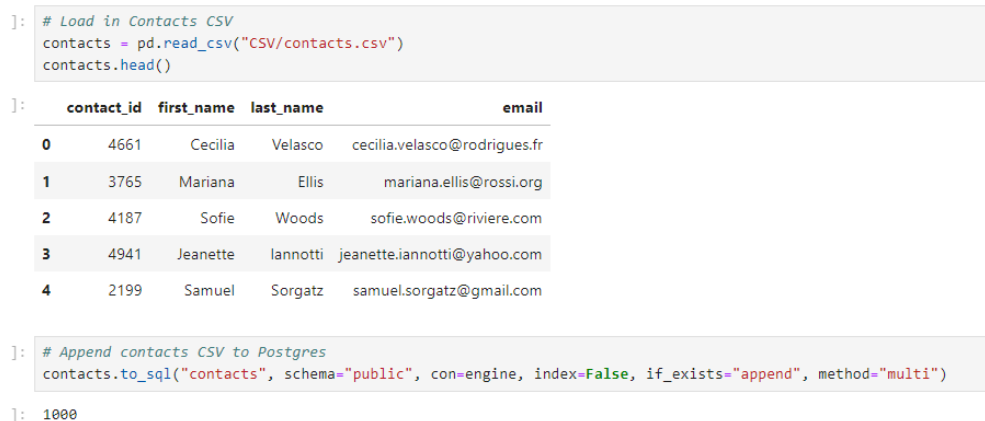
The final phase of the ETL process was the Load phase. The first step in our Load phase was to create a database, labelled `project_two_etl`, for the project within Postgres and use the script created by QUICKDBD to create our four tables within that database. After the database and tables were generated, we created a second Jupyter Notebook, “ETL\_Load CSVs.ipynb”, to house all of the script necessary to upload the CSVs to the Postgres tables.

### Creating tables within Postgres



```
1 -- Exported from QuickDBD: https://www.quickdatabasediagrams.com/
2 -- Link to schema: https://app.quickdatabasediagrams.com/#/d/RxxbUH
3 -- NOTE! If you have used non-SQL datatypes in your design, you will have to change these here.
4
5
6 CREATE TABLE "category" (
7     "category_id" VARCHAR(5) NOT NULL,
8     "category" VARCHAR(100) NOT NULL,
9     "last_updated" TIMESTAMP DEFAULT LOCALTIMESTAMP NOT NULL,
10    CONSTRAINT "pk_category" PRIMARY KEY (
11        "category_id"
12    )
13 );
14
15 CREATE TABLE "subcategory" (
16     "subcategory_id" VARCHAR(8) NOT NULL,
17     "subcategory" VARCHAR(100) NOT NULL,
18     "last_updated" TIMESTAMP DEFAULT LOCALTIMESTAMP NOT NULL,
19    CONSTRAINT "pk_subcategory" PRIMARY KEY (
20        "subcategory_id"
21    )
22 );
23
24 CREATE TABLE "contacts" (
25     "contact_id" INTEGER NOT NULL,
26     "first_name" VARCHAR(100) NOT NULL,
```

### Reading CSV and Appending CSV to SQL database



```
] # Load in Contacts CSV
contacts = pd.read_csv("CSV/contacts.csv")
contacts.head()
```

	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

```
] # Append contacts CSV to Postgres
contacts.to_sql("contacts", schema="public", con=engine, index=False, if_exists="append", method="multi")

]: 1000
```



## Data Analysis

As a ship cannot steer without a rudder, companies cannot function effectively without data analytics. Our ETL pipeline supplied us with information about crowdfunding projects. Stopping there would be a disservice to the information we worked so hard to gather. Our group decided to review this information based on a West vs. East approach, using the United States (US) and Canada (CA) as the West and identifying the East as China (CH) and Australia (AU).

We separated our analysis into three questions:

1. When grouped by category, how does the overall count of successes vs. failures differ for each of our four selected countries?
2. How do our selected countries compare when looking at set goals, pledged finances, and number of backers?
3. How does the length of a campaign compare to the success/failure of our selected countries?

To connect our Jupyter Notebook to our Postgres database. We set up an engine to communicate with our database and then inspected the engine to ensure communication.

```
: # Assign SQL variables
SQL_USERNAME = "postgres"
SQL_PASSWORD = "postgres" # change this
SQL_IP = "localhost"
PORT = 5432
DATABASE = "project_two_etl" # change this

: # Set up Engine
connection_string = f"postgresql+psycopg2://{SQL_USERNAME}:{SQL_PASSWORD}@{SQL_IP}:{PORT}/{DATABASE}"
engine = create_engine(connection_string)

: # Create the inspector and connect it to the engine
inspector = inspect(engine)

# Collect the names of tables within the database
tables = inspector.get_table_names()

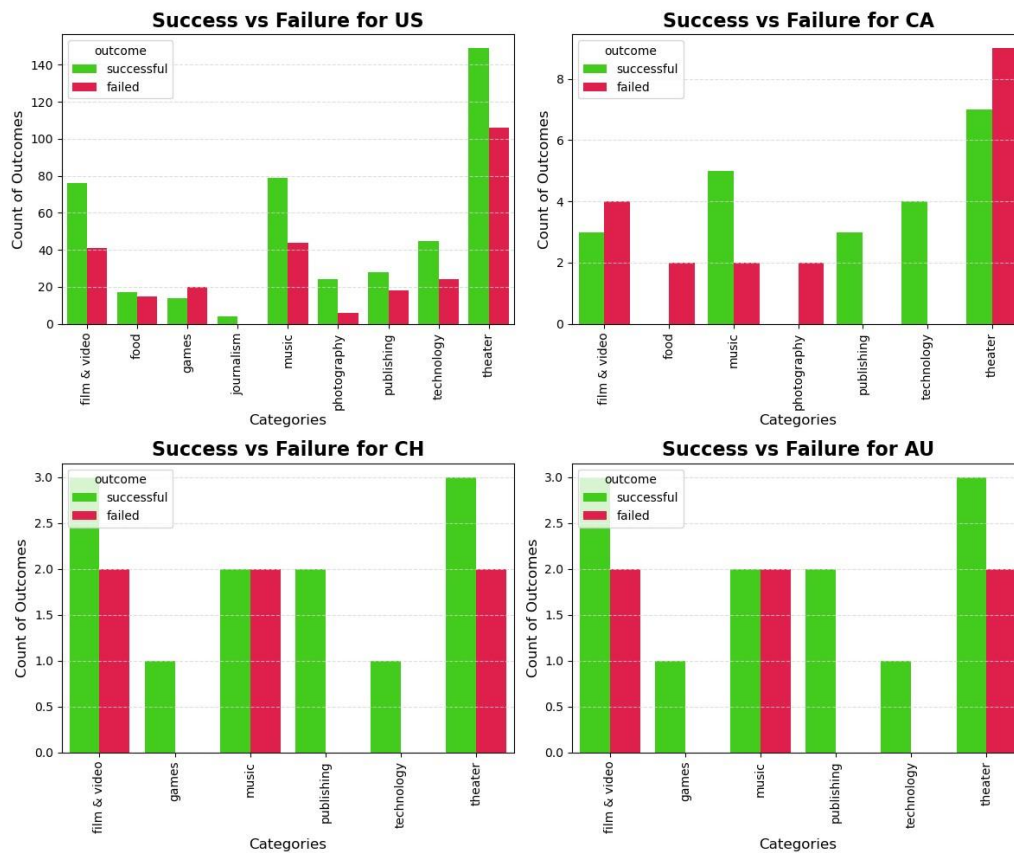
# Use the inspector to print the column names within each table and its data types
for table in tables:
    print(table)
    columns = inspector.get_columns(table)
    for column in columns:
        print(column["name"], column["type"])

    print()

contacts
contact_id INTEGER
first_name VARCHAR(100)
last_name VARCHAR(250)
email VARCHAR(250)
last_updated TIMESTAMP

campaign
cf_id INTEGER
contact_id INTEGER
```

## Question #1 - Four countries, grouped by category, success vs. failure comparison



This dataset had a strong representation of US crowdfunding projects and a smaller number of overall projects for each of our other selected countries. We are able to see some interesting trends however. Three categories, film & video, music, and theater, had a strong representation in all four countries, showing a popularity for the arts in crowdfunding projects seeking support. It was also interesting to see four categories in China had more failures than successes, a trend that wasn't reflected in the other three countries, leading us to believe projects in the west had a greater chance of success than in the east.

Individual query and charting work done in "Ind Nick Work/ETL\_Data\_Viz". All required charting saved in "Charts" folder.

SQL query used to retrieve data from our database, changing WHERE clause for each country:

```
# Query for 'US'
query = """
SELECT
    cat.category,
    cam.outcome,
    COUNT(cam.outcome) AS count,
    cam.country
FROM campaign AS cam
JOIN category AS cat ON (cam.category_id = cat.category_id)
WHERE cam.outcome IN ('successful', 'failed')
    AND cam.country = 'US'
GROUP BY cat.category, cam.outcome, cam.country
ORDER BY cat.category, cam.outcome DESC;
"""

question1us = pd.read_sql(text(query), con=engine)
question1us.head(10)
```

Python code to generate four charts as subplots:

```
# Create a figure and axes with 2 rows and 2 columns
fig, axs = plt.subplots(2, 2, figsize=(12, 10))

# Plot for question1us
sns.barplot(data=question1us, x='category', y='count', hue='outcome', palette='prism', ax=axs[0, 0])
axs[0, 0].set_xlabel('Categories', fontsize=12)
axs[0, 0].set_ylabel('Count of Outcomes', fontsize=12)
axs[0, 0].set_title('Success vs Failure for US', fontsize=16, fontweight='bold')
axs[0, 0].tick_params(axis='x', rotation=90)
axs[0, 0].grid(color='lightgrey', axis='y', linestyle='--', alpha=0.75)

# Plot for question1ca
sns.barplot(data=question1ca, x='category', y='count', hue='outcome', palette='prism', ax=axs[0, 1])
axs[0, 1].set_xlabel('Categories', fontsize=12)
axs[0, 1].set_ylabel('Count of Outcomes', fontsize=12)
axs[0, 1].set_title('Success vs Failure for CA', fontsize=16, fontweight='bold')
axs[0, 1].tick_params(axis='x', rotation=90)
axs[0, 1].grid(color='lightgrey', axis='y', linestyle='--', alpha=0.75)

# Plot for question1ch
sns.barplot(data=question1ch, x='category', y='count', hue='outcome', palette='prism', ax=axs[1, 0])
axs[1, 0].set_xlabel('Categories', fontsize=12)
axs[1, 0].set_ylabel('Count of Outcomes', fontsize=12)
axs[1, 0].set_title('Success vs Failure for CH', fontsize=16, fontweight='bold')
axs[1, 0].tick_params(axis='x', rotation=90)
axs[1, 0].grid(color='lightgrey', axis='y', linestyle='--', alpha=0.75)

# Plot for question1au
sns.barplot(data=question1au, x='category', y='count', hue='outcome', palette='prism', ax=axs[1, 1])
axs[1, 1].set_xlabel('Categories', fontsize=12)
axs[1, 1].set_ylabel('Count of Outcomes', fontsize=12)
axs[1, 1].set_title('Success vs Failure for AU', fontsize=16, fontweight='bold')
axs[1, 1].tick_params(axis='x', rotation=90)
axs[1, 1].grid(color='lightgrey', axis='y', linestyle='--', alpha=0.75)

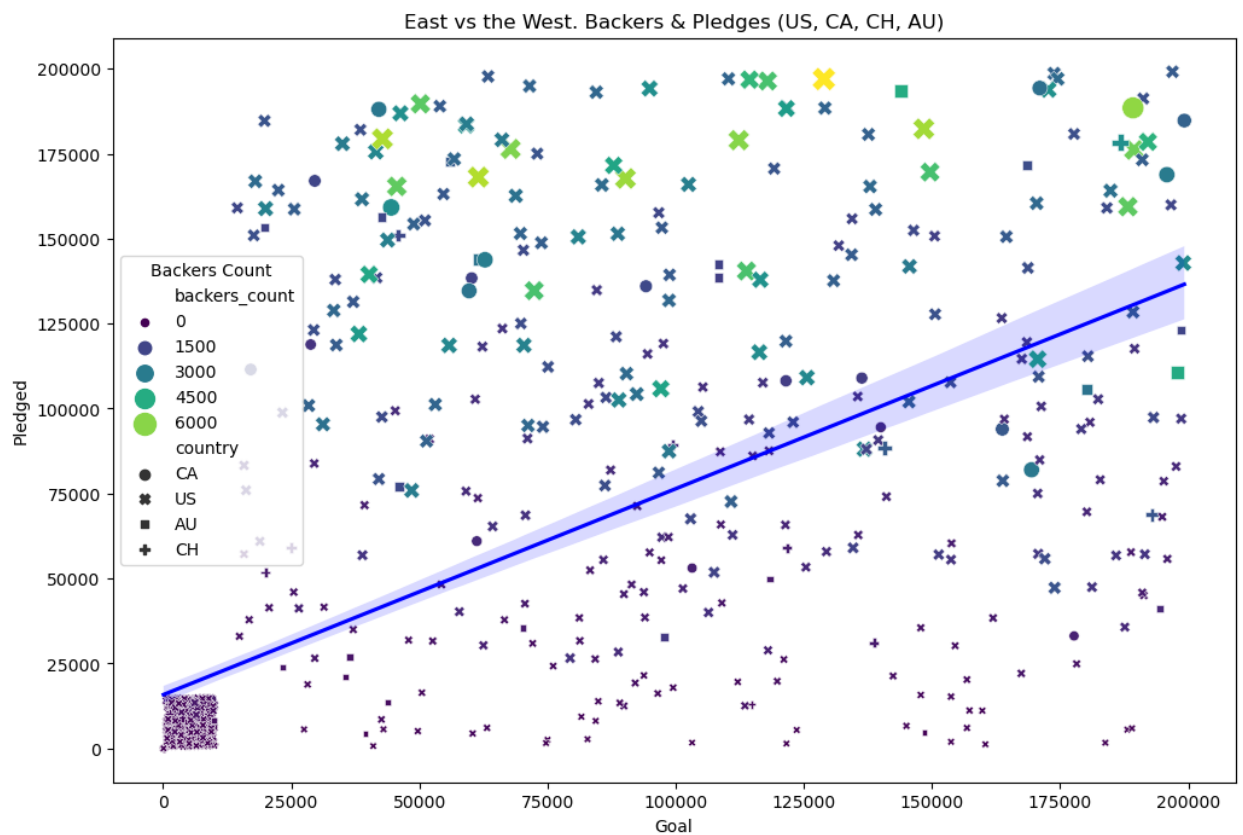
plt.tight_layout()

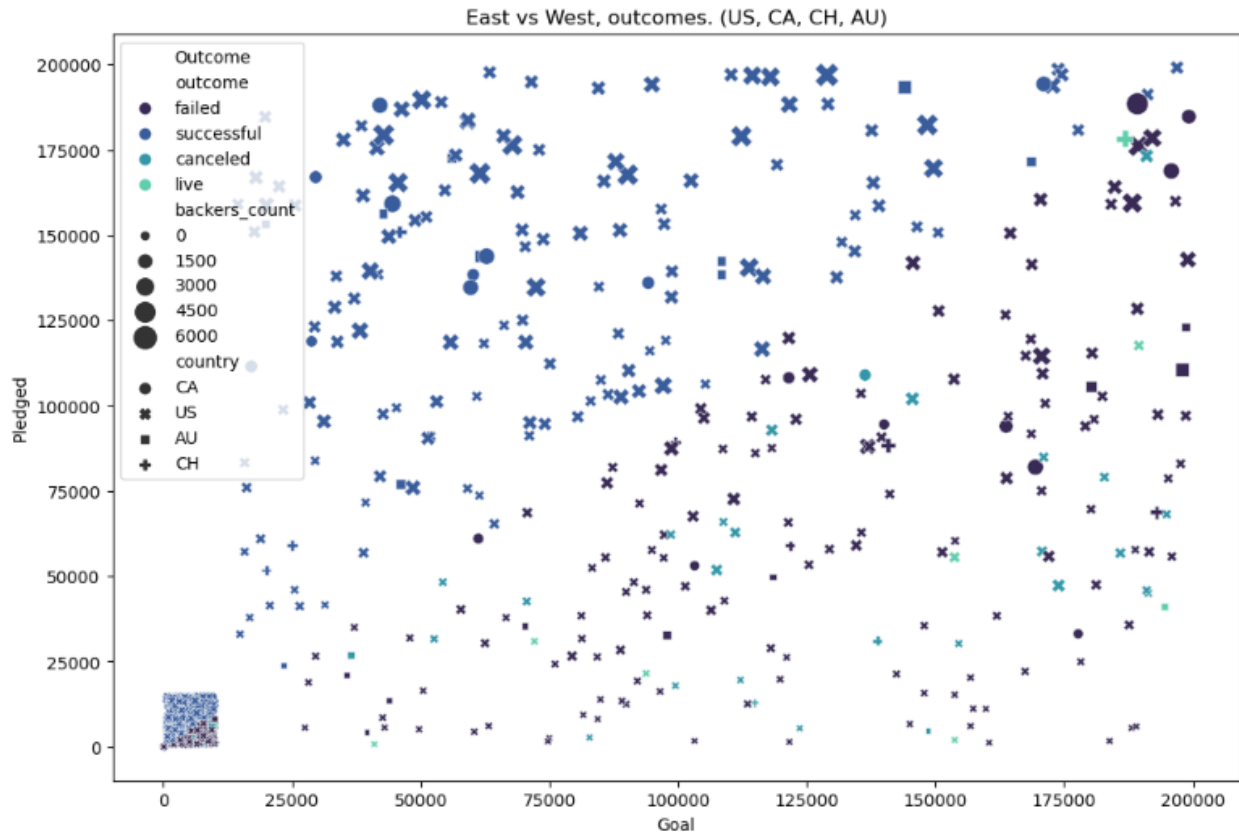
# save chart in Charts folder
plt.savefig('../Charts/success_vs_failure_combined.png', bbox_inches='tight')

# show
plt.show()
```

## Question #2 - Four countries, comparing set goals, pledged finances, and number of backers

The provided charts indicate a strong positive correlation between the funding goals and the pledged amounts, especially for successful projects, highlighting the importance of setting achievable and realistic goals. Higher backer counts generally lead to higher pledged amounts, emphasizing the need to engage and attract more backers for a successful campaign. The United States stands out as the dominant contributor, with significantly higher percentages of both backers and pledged amounts compared to other countries. Overall, while successful projects show consistent trends with high pledges and backer counts, failed and canceled projects exhibit more variability and inconsistency, underscoring the challenges in meeting high funding goals without sufficient support.

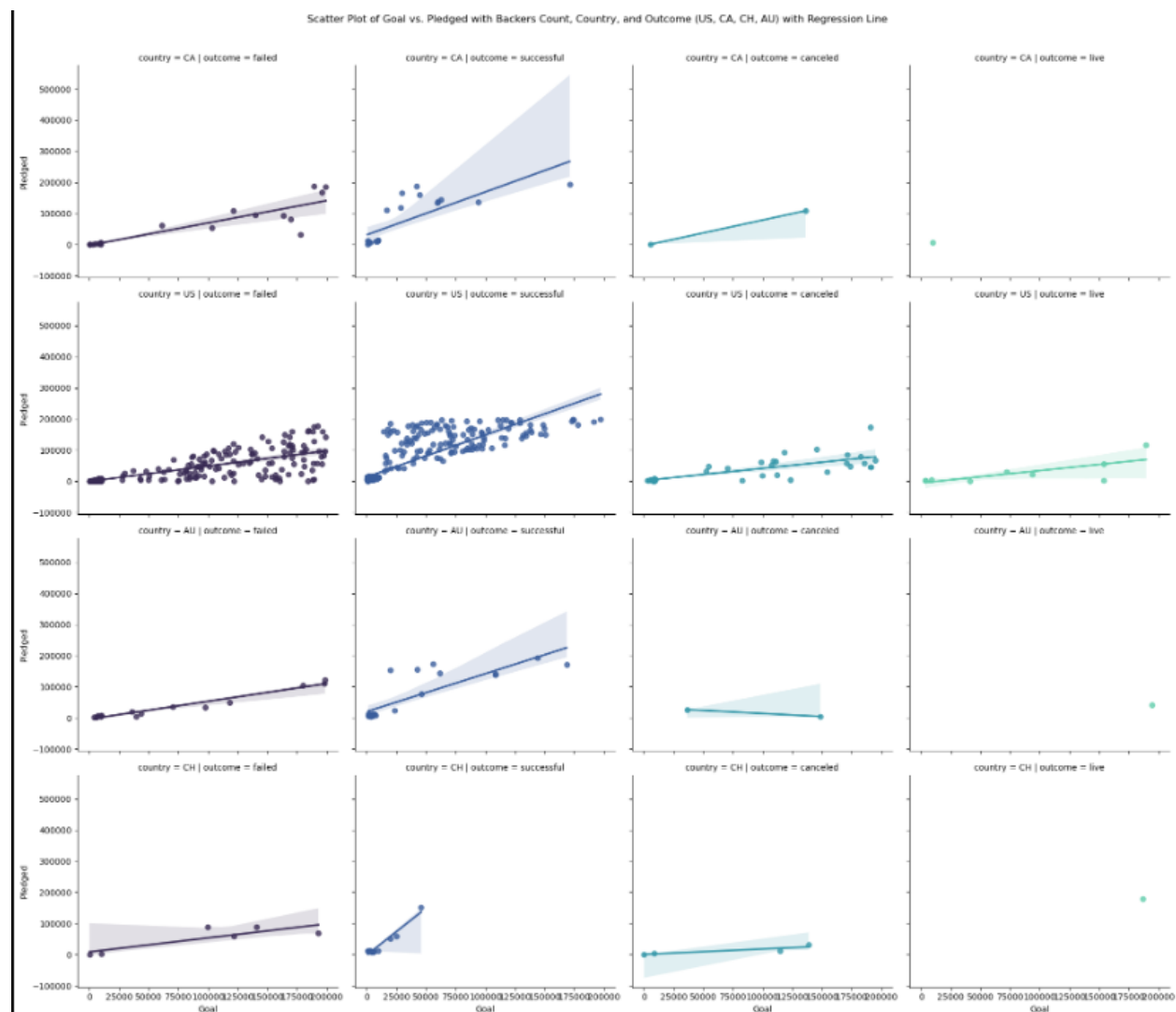




The scatter plots reveal that successful projects tend to have tighter distributions and higher concentrations of data points along the regression lines, suggesting a more predictable and stable relationship between goals and pledged amounts. In contrast, failed and canceled projects display a broader spread, indicating less predictability and greater challenges in reaching their funding targets. This variability underscores the importance of strategic planning and effective marketing to ensure sufficient backer engagement and support.

Additionally, the comparative analysis between different countries highlights significant disparities in contributions. The US, being a major player, shows a wide range of goals and pledged amounts, with larger backer counts correlating with higher pledges. Other countries like Canada, Australia, and China also contribute but to a lesser extent. These insights suggest that understanding and leveraging regional trends and backer behavior can be crucial for tailoring campaigns to maximize success. Overall, the data underscores the need for realistic goal-setting, robust backer engagement strategies, and awareness of regional dynamics to enhance the likelihood of successful crowdfunding campaigns.

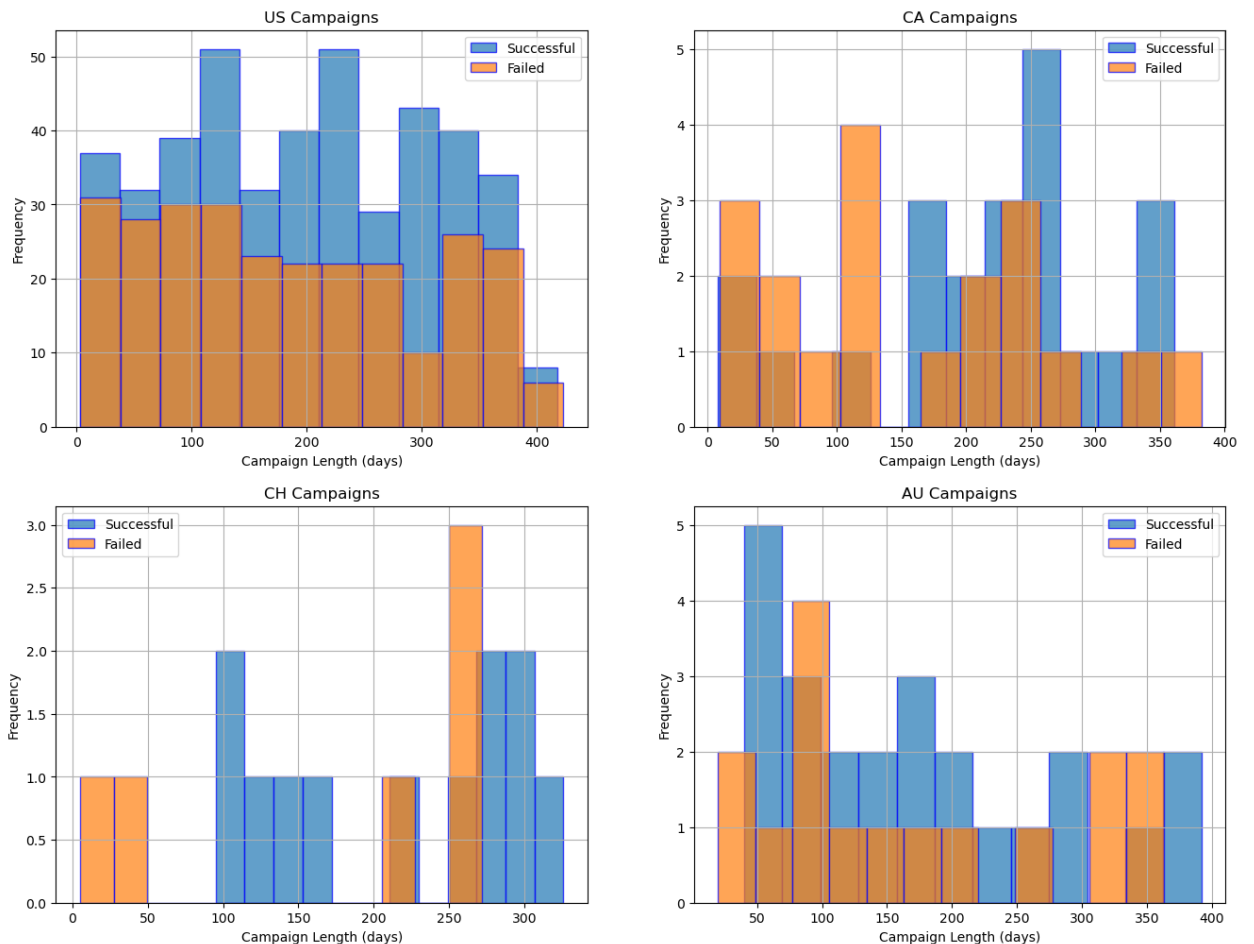
The third chart reinforces the positive correlation between funding goals and pledged amounts across various countries and outcomes. Successful projects consistently show strong positive trends with higher pledges, especially in the US and Canada, indicating that well-planned campaigns significantly impact funding success. In contrast, failed and canceled projects display more variability, highlighting the difficulties in reaching funding targets without adequate support. The data emphasizes the importance of setting realistic goals, effectively engaging backers, and considering regional trends to tailor campaigns for maximum success.



Individual query and charting work done in "Ind Sam Work/Data\_Viz\_Sam". All required charting saved in "Charts" folder.

### Question #3 – Compare the length of campaigns to outcome (Success and Failures)

Visualization of the data does not show a solid correlation between the length of the campaign and its success or failure.



When looked by individual countries the US has successful campaigns across most days, but the least successful in the shortest and the longest days. Canada has the most success between approximately 225 and 250 days, China between 275 and 300, and Australia around 50 days. All countries had success with shorter and longer campaigns.

The length of the campaign is only one factor, other factors should be explored such as the type of play, the time of year in the specific country, the goal, etc.