Title: Project 2, Team 3
Team: Kimberly Childers, Joshua Hale, and Damarje Brown

Project Description: Project 2 required team members to work together and explore how to extract data utilizing Python and Pandas, transform and clean data, parse string data, place data in dictionaries, use lists to display readable code, and manipulate strings using regular expressions. Team 2 accomplished this by creating Category, Subcategory, Campaign, Contacts, and Crowdfunding data frames.

Analysis:
1) Category data frame was created. Reference category data frame file "category.csv" located in the “Resources (CSV Files)” folder on GitHub. The team created a “category_id” and “category” column from the “crowdfunding.xlsx” data provided.

```
# Read the data into a Pandas DataFrame
crowdfunding_info_df = pd.read_excel('Resources/crowdfunding.xlsx')
crowdfunding_info_df.head()
```

company_name	blurb	goal	pledged	outcome	backers_count	country	currency	launched_at	deadline	staff_pick	spotlight	category & sub-category
Baldwin, Riley and Jackson	Pre-emptive tertiary standardization	100	0	failed	0	CA	CAD	1581573600	1614578400	False	False	food/food trucks
Odom Inc	Managed bottom-line architecture	1400	14560	successful	158	US	USD	1611554400	1621918800	False	True	music/rock
Melton, Robinson and Fritz	Function-based leadingedge pricing structure	108400	142523	successful	1425	AU	AUD	1608184800	1640844000	False	False	technology/web
Mcdonald, Gonzalez and Ross	Vision-oriented fresh-thinking conglomeration	4200	2477	failed	24	US	USD	1634792400	1642399200	False	False	music/rock
Larson-Little	Proactive foreground core	7600	5265	failed	53	US	USD	1608530400	1629694800	False	False	theater/plays

In [15]:
```
# Assign the category and subcategory values to category and subcategory columns.
crowdfunding_info_df[['category', 'subcategory']] = crowdfunding_info_df['category & sub-category'].str.split('/
crowdfunding_info_df.head()
```

Out[15]:

blurb	goal	pledged	outcome	backers_count	country	currency	launched_at	deadline	staff_pick	spotlight	category & sub-category	category	subcategory
emptive tertiary dization	100	0	failed	0	CA	CAD	1581573600	1614578400	False	False	food/food trucks	food	food trucks
lanaged tom-line itecture	1400	14560	successful	158	US	USD	1611554400	1621918800	False	True	music/rock	music	rock
unction-based ingedge pricing tructure	108400	142523	successful	1425	AU	AUD	1608184800	1640844000	False	False	technology/web	technology	web
oriented thinking neration	4200	2477	failed	24	US	USD	1634792400	1642399200	False	False	music/rock	music	rock
roactive eground core	7600	5265	failed	53	US	USD	1608530400	1629694800	False	False	theater/plays	theater	plays

```
In [118]: 1 category_df
```

Out[118]:

	category_id	category
0	cat1	food
1	cat2	music
2	cat3	technology
3	cat4	theater
4	cat5	film & video
5	cat6	publishing
6	cat7	games
7	cat8	photography
8	cat9	journalism

2) Subcategory data frame was created. Reference subcategory data frame file "subcategory.csv." The team created a “subcategory_id” and “subcategory” column.

```
In [22]: 1 subcategory_df
```

Out[22]:

	subcategory_id	subcategory
0	subcat1	food trucks
1	subcat2	rock
2	subcat3	web
3	subcat4	plays
4	subcat5	documentary
5	subcat6	electric music
6	subcat7	drama
7	subcat8	indie rock
8	subcat9	wearables
9	subcat10	nonfiction
10	subcat11	animation
11	subcat12	video games
12	subcat13	shorts
13	subcat14	fiction
14	subcat15	photography books
15	subcat16	radio & podcasts
16	subcat17	metal
17	subcat18	jazz
18	subcat19	translations
19	subcat20	television
20	subcat21	mobile games
21	subcat22	world music
22	subcat23	science fiction
23	subcat24	audio

3) Campaign data frame was created. Reference campaign data frame file "campaign.csv." Columns were renamed from “launched_at” and “deadline” to “launch_date” and “end_date.” Then both UTC times were converted to the datetime format.

In [28]:
```
# Format the launched_date and end_date columns to datetime format
from datetime import datetime as dt
campaign_df["launch_date"] = [dt.utcfromtimestamp(int(x)) for x in campaign_df["launch_date"]]

```

In [29]:
```
campaign_df["end_date"] = [dt.utcfromtimestamp(int(x)) for x in campaign_df["end_date"]]
```

In [30]:
```
campaign_df.head()
```

Out[30]:

e	description	goal	pledged	outcome	backers_count	country	currency	launch_date	end_date	staff_pick	spotlight	category & sub-category	category	sub
y n	Pre-emptive tertiary standardization	100.0	0.0	failed	0	CA	CAD	2020-02-13 06:00:00	2021-03-01 06:00:00	False	False	food/food trucks	food	fo
c	Managed bottom-line architecture	1400.0	14560.0	successful	158	US	USD	2021-01-25 06:00:00	2021-05-25 05:00:00	False	True	music/rock	music	
i, d z	Function-based leadingedge pricing structure	108400.0	142523.0	successful	1425	AU	AUD	2020-12-17 06:00:00	2021-12-30 06:00:00	False	False	technology/web	technology	
l, d s	Vision-oriented fresh-thinking conglomeration	4200.0	2477.0	failed	24	US	USD	2021-10-21 05:00:00	2022-01-17 06:00:00	False	False	music/rock	music	
e	Proactive foreground core	7600.0	5265.0	failed	53	US	USD	2020-12-21 06:00:00	2021-08-23 05:00:00	False	False	theater/plays	theater	

4) Contact data frame was created. Reference contact data frame file "contacts.csv." The team chose to use option one which used Python dictionaries as opposed to regular expressions. Each row of the provided data was converted into a dictionary and pandas was used to make a data frame listing the contact_id, name, and e-mail. The image below shows the names column split into first and last name.

In [41]:
```
# 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)


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

Out[41]:

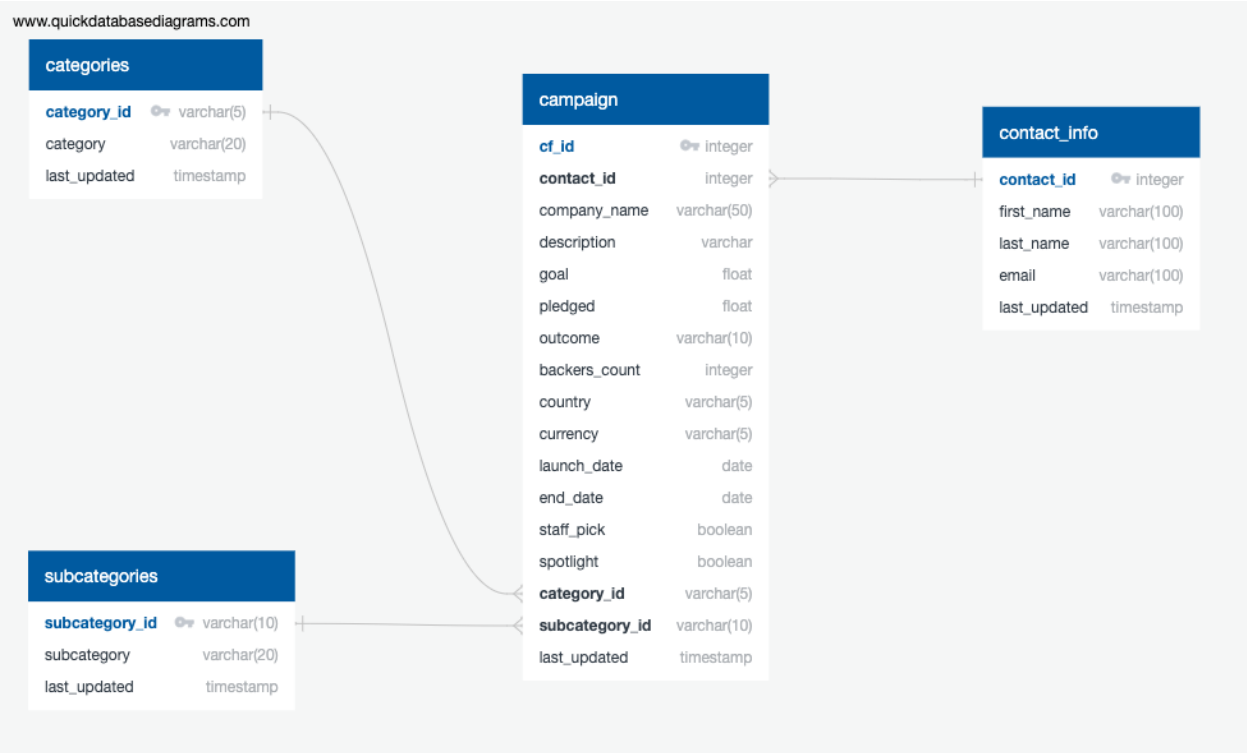
	contact_id	email	first_name	last_name
0	4661	cecilia.velasco@rodrigues.fr	Cecilia	Velasco
1	3765	mariana.ellis@rossi.org	Mariana	Ellis
2	4187	sofie.woods@riviere.com	Sofie	Woods
3	4941	jeanette.iannotti@yahoo.com	Jeanette	Iannotti
4	2199	samuel.sorgatz@gmail.com	Samuel	Sorgatz

In [42]:
```
# Reorder the columns
contact_info = contact_info[['contact_id', 'first_name', 'last_name', 'email']]
contact_info.head()
```

Out[42]:

	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

5) Crowdfunding database was created. Reference crowdfunding database file "crowdfunding_db." QuickDBD was used to an ERD of tables.



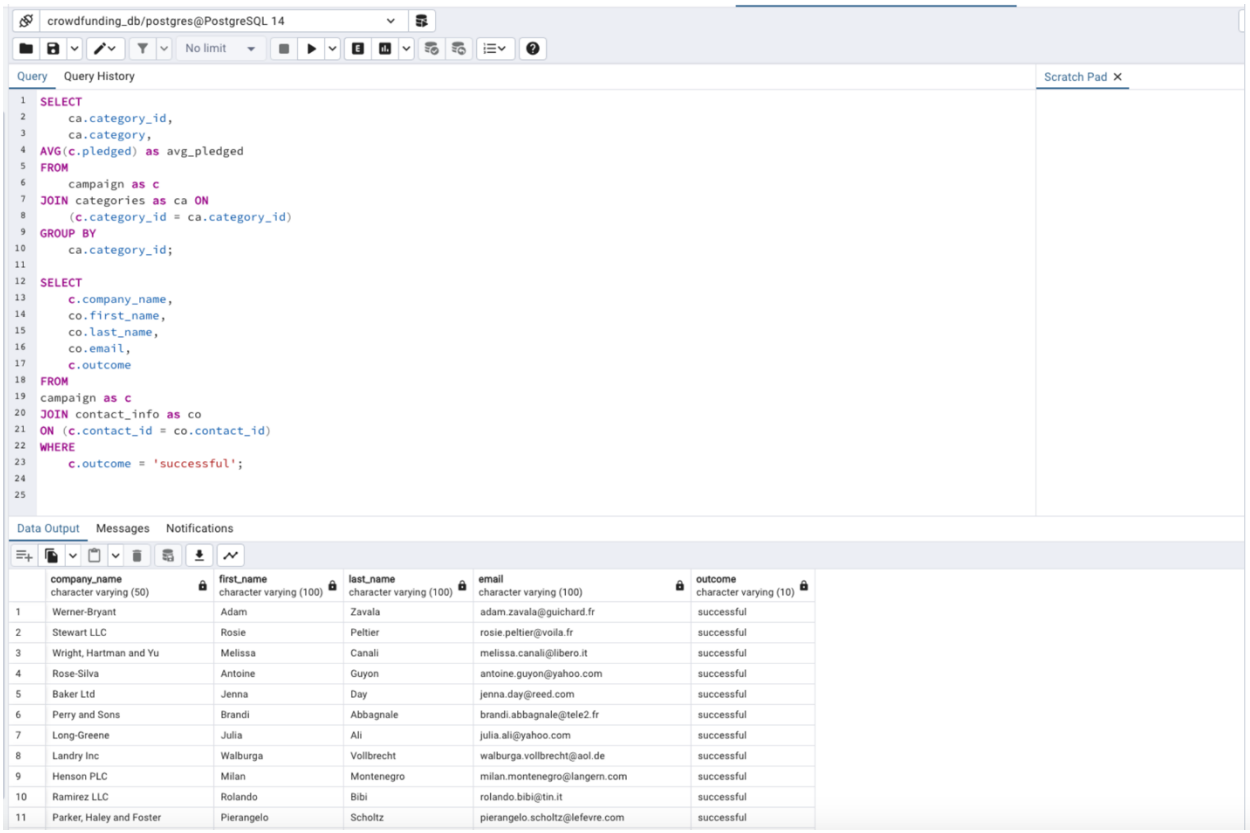
The team finally made a Postgres file and verified that all tables were created. All those files can be found on the GitHub folder titled “Screen shots of verifying data in pgAdmin.” The file contains screenshot and downloads of tables from pgAdmin4.

Future Implications:

Team 2 could utilize this data for future analysis to discover any trends such as creating another database to group successful campaigns by their category to observe categories with higher percentages. Automating scripts to pull campaign and contact data continuously would depict any trends over time on successful companies and assist with forecasting results. Finally, team 2 could improve the coding and make it more robust by accounting for decimal numbers. Data inputs currently characterized as integers should be changed to a float because monetary pledges will not always be whole number.

Queries:

Please reference folder in GitHub titled “Queries.” The team performed a JOIN and AGGREGATE query followed by another JOIN.



	company_name character varying (50)	first_name character varying (100)	last_name character varying (100)	email character varying (100)	outcome character varying (10)
1	Werner-Bryant	Adam	Zavala	adam.zavala@guichard.fr	successful
2	Stewart LLC	Rosie	Peltier	rosie.peltier@voila.fr	successful
3	Wright, Hartman and Yu	Melissa	Canali	melissa.canali@libero.it	successful
4	Rose-Silva	Antoine	Guyon	antoine.guyon@yahoo.com	successful
5	Baker Ltd	Jenna	Day	jenna.day@reed.com	successful
6	Perry and Sons	Brandi	Abbagnale	brandi.abbagnale@tele2.fr	successful
7	Long-Greene	Julia	Ali	julia.ali@yahoo.com	successful
8	Landry Inc	Walburga	Vollbrecht	walburga.vollbrecht@aol.de	successful
9	Henson PLC	Milan	Montenegro	milan.montenegro@langern.com	successful
10	Ramirez LLC	Rolando	Bibi	rolando.bibi@tin.it	successful
11	Parker, Haley and Foster	Pierangelo	Scholtz	pierangelo.scholtz@lefevre.com	successful

Bonus:

Reference the “Code” folder in GitHub for the code titled “Read_Write_Pistgre_Demo_Childers”

In [53]:

```
query = """
        SELECT
            *
        FROM
            campaign
        ORDER BY
            goal DESC
        LIMIT 10;
        """

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

Out[53]:

	cf_id	contact_id	company_name	description	goal	pledged	outcome	backers_count	country	currency	launch_da
0	1905	2329	Romero-Hoffman	Open-source zero administration complexity	199200.0	184750.0	failed	2253	CA	CAD	2021-06
1	553	4593	Hensley Ltd	Versatile cohesive open system	199000.0	142823.0	failed	3483	US	USD	2021-07
2	2489	2272	Miller Ltd	Open-architected mobile emulation	198600.0	97037.0	failed	1198	US	USD	2020-11
3	2311	1673	Farrell and Sons	Synergized 4thgeneration conglomeration	198500.0	123040.0	failed	1482	AU	AUD	2020-08
4	2837	3768	Kelly-Colon	Stand-alone grid-enabled leverage	197900.0	110689.0	failed	4428	AU	AUD	2021-09

In [54]:

```
plt.figure(figsize=(15,8))
plt.barh(df.company_name, df.goal, color="blue")
plt.title("Top 10 Company's Goal Amount", fontweight="bold")
plt.xlabel("Goal")
plt.xlim(195000, 200000)
plt.grid(axis="x", color="lightgrey", linestyle="--", alpha=0.10)

plt.show()
```