As mentioned before, the datasets were strategically picked so that transforming and merging the datasets went as smoothly as possible. All datasets were in csv format utilizing the same columns. With that not being a factor at all, our biggest task was figuring out “how” we wanted to transform our dataset for the best possible analysis. We automatically wanted to get rid of any extra columns. Next, we wanted to drop any NaN in our datasets. We localized both the tv shows and movies to see which category was the largest. Not only were movies the largest dataset, but also had the least number of NaNs. Figuring out how to merge the data was probably the hardest task we had to figure out. We knew that creating an “ID” would be problematic for us when it came to putting the data in PostgresSQL because it would differentiate which tv streaming service is associated to which data. Instead, we created a new column and used that column solely to name which tv streaming service is associated to the movies for each dataset. From there, were merged all four datasets into one. We used the nine following steps to transform our four datasets:

1. Used “***pd.read\_csv”*** to read the csv file
2. Used “***drop”*** to dropped columns “show\_id”, “type”, “director”, “date\_added”, “cast”, “rating” and “duration.”
   * We originally wanted the “cast” column because we thought the cast of a movies plays a big role in viewers decision to watch a movie. However, Postgres SQL was not letting that column load into the database. Even after renaming the column, changing the type to “text”/“null”/“not null”, we still received an error. As a result, we dropped the column.
3. Used ***“dropna”*** to drop any NaN within the country column.
4. Used ***“loc”*** to only show movies in rather than both tv show and movies
5. Used ***“rename”*** to change “listed\_in” 🡪 “genre”
6. ***Added*** “Streaming Services” as a new column in with the name of the tv streaming service as the data
7. Used ***“to\_csv”*** to change the transformed/finalized data into a new csv file
8. Used ***“merge”*** to merge all four datasets together into one
9. Used ***“to\_csv”*** to the merge dataset into a csv file

Adding the dataset in PostgresSQL was simple. We simply created a table and used the “export/import” to bring the csv file into PostgresSQL. Because this project was fairly simple. We wanted to share areas of improvement. The biggest area of improvement is deep diving more into how to extract the data to answer essentially, “which is tv streaming service is the best for your buck?” If given the time, we thought these following questions would help viewers accurately pick a certain tv streaming services based on their needs/interests:

1. Which tv streaming service has the most movie variety in movie genres?
2. Which tv streaming service has the most variety in ethnicity-based movies?
3. Which tv streaming service is more children friendly?
4. Which tv streaming service has the most recent movies (>2000)?
5. Which tv streaming service has the most “old school” movies (<2000)?

Utilizing these questions requires further transformation of our data. Possibly creating several data frames for views to analysis, and even utilizing SQL alchemy as an option to filter and sort our data. With this full-thought out analysis, we believe the improvements would change drastically how individuals pick which tv streaming service they should subscribe to.