**Extract**

We extracted 2 video game data sets; “vgsales” and “ersb\_rating” from Kaggle as a CSV. To be able to code our extraction, we imported pandas as pd. For our sales CSV we created the following code:    
           sales\_csv = "Resources/vgsales.csv"  
           sales\_df = pd.read\_csv(sales\_csv)and to pull our rating CSV we created the following:  
           rating\_csv = "Resources/Video\_games\_esrb\_rating.csv"  
rating = pd.read\_csv(rating\_csv)In order to connect to the databases, we needed to initiate a connection engine using a pgAdmin tables to create a SQL database.  We start off by creating a path from sqlalchemy imported create engine.   
   
To create the connection, we used postgres as a local host;  
 rds\_connection\_string = "postgres:postgres@localhost:5432/videogame\_db"And to extract an engine we used the following code:  
engine = create\_engine(f'postgresql://{rds\_connection\_string}')After effectively establishing the steps from above, we were successfully able to extract CSV and PgAdmin table into a data frame.

**Transform**

We took the two csv files that we extracted from Kaggle and read them into the jupyter notebook using pandas. After looking at what the dataframes looked like, we were able to decide which columns we wanted to keep for our database. For the ratings table, we only kept the Title and ESRB\_Rating columns and changed the Title column name to Name to match the sales table. From there, we merged this cleaned table with the sales table.

For the merge, we did an inner join on the Name columns of both tables. This made it so that it only kept the games they had in common. After viewing the merged table, we realized a lot of the games had duplicates from the sales table due to them being available on multiple platforms (meaning each platform had different sales totals). We decided to keep only the first instance of each game, because the Global\_Sales column was in descending order. This left us with each game’s platform that had the most sales.

The final step in our transforming process was to get rid of other unnecessary columns. The columns we kept were Rank, Name, Genre, Platform, Global\_Sales, and ESRB\_Rating. In order to properly load this table into our database, we had to make each column name lowercase due to pgAdmin being case-sensitive. We also made sure to reset the index so we could use the index as our primary key in the database.

**Load**

For the load part of the project, we start with creating a connection to the database. In doing this we created a connection string (rds\_connection\_string = "postgres:postgres@localhost:5432/videogame\_db"), and engine (engine = create\_engine(f'postgresql://{rds\_connection\_string}'), which allows us to connect to local database we created using pgAdmin (videogame\_db). The next step in the process was to utilize pandas to load the csv converted DataFrame into the database using the to\_sql command ([merge\_df.to](http://merge_df.to/" \t "_blank)\_sql(name='sales', con=engine, if\_exists='append', index=True)). In other words we are taking the merge\_df and putting it into the sales table we created earlier. In order to make sure everything went smoothly we checked in jupyter notebook for a successful connection with pd.read\_sql\_query. We are querying the sales table to confirm the data has been added and it showed in the output of the notebook.