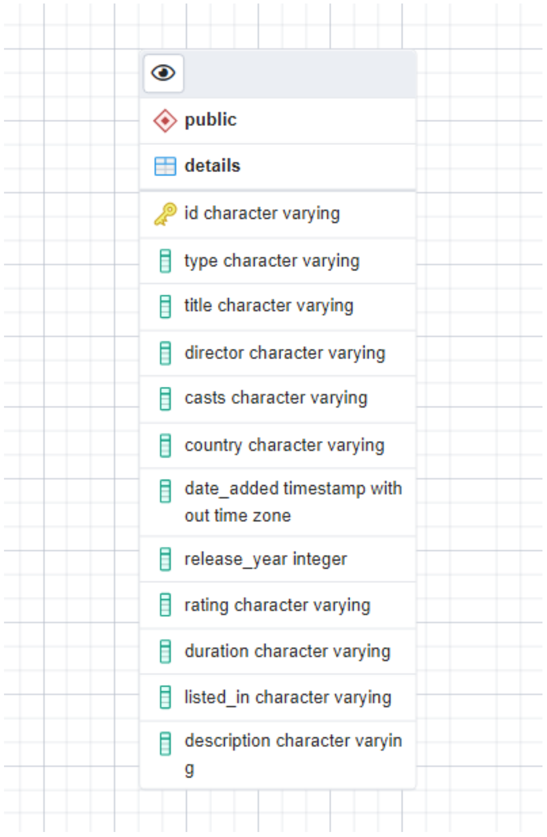
DATABASE HOMEWORK 3 REPORT

李嘉玲 – 109550186

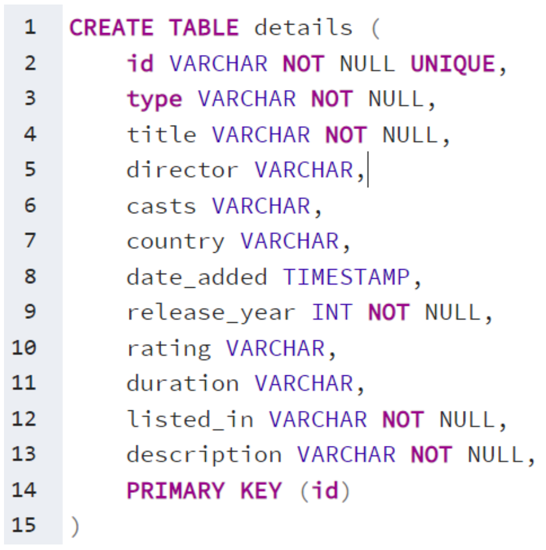
For this final project, I am making a Netflix Recommendation System based on factors like title, directors, casts or actors, genres, and descriptions of the movies and TV shows. The motivation for doing this is because I am personally a Netflix user too. Eating alone and watching Netflix is one of the habits that I have had for quite a long time. However, everytime I would spend most of the time finding the right movies to watch and left with little time watching it. I believe that most people have this problem of contemplation too. People opened Netflix and saw too many movies and TV shows in front of their eyes that they don’t even know what to watch. They don’t want to just randomly picked a movie and unfortunately, it happened to be a bad one. Thus, I am motivated to make this application as it will give you recommendations from your picked or inputted movie and it will give you recommendations based on some of the factors your picked movie have.

For this application, you only need to run the python and the Tkinter GUI will be shown to you. You only need to input your picked movie and it will give you top ten similar movies or shows recommendation. The data source that I picked is from Kaggle (Netflix Movies and TV Shows). As the data was already in csv file, I make a new database from AWS module as well as in the pgadmin. Eventhough it is only one table, but it seems like all the columns or attributes are already independent or normalized. Therefore, I did not separate them into tables or normalize it again. The performance trade-off might happen that the performance increased but storage decreased. Then, I create a table based on the columns in the table and import the data from the csv file into table like how we did in the previous homeworks. As the data is taken directly from the csv file, we would need to update the data manually by modifying it from time to time.

The database schema below is from the visualization tool in DBMS, which is from the generate tool from pgadmin database. Some of the constraints that are unable to be seen from the database schema can be seen are the not null values, which are the id, type, title, released year, genres, and descriptions. It can also be seen in the screenshot below that I took when creating the table.

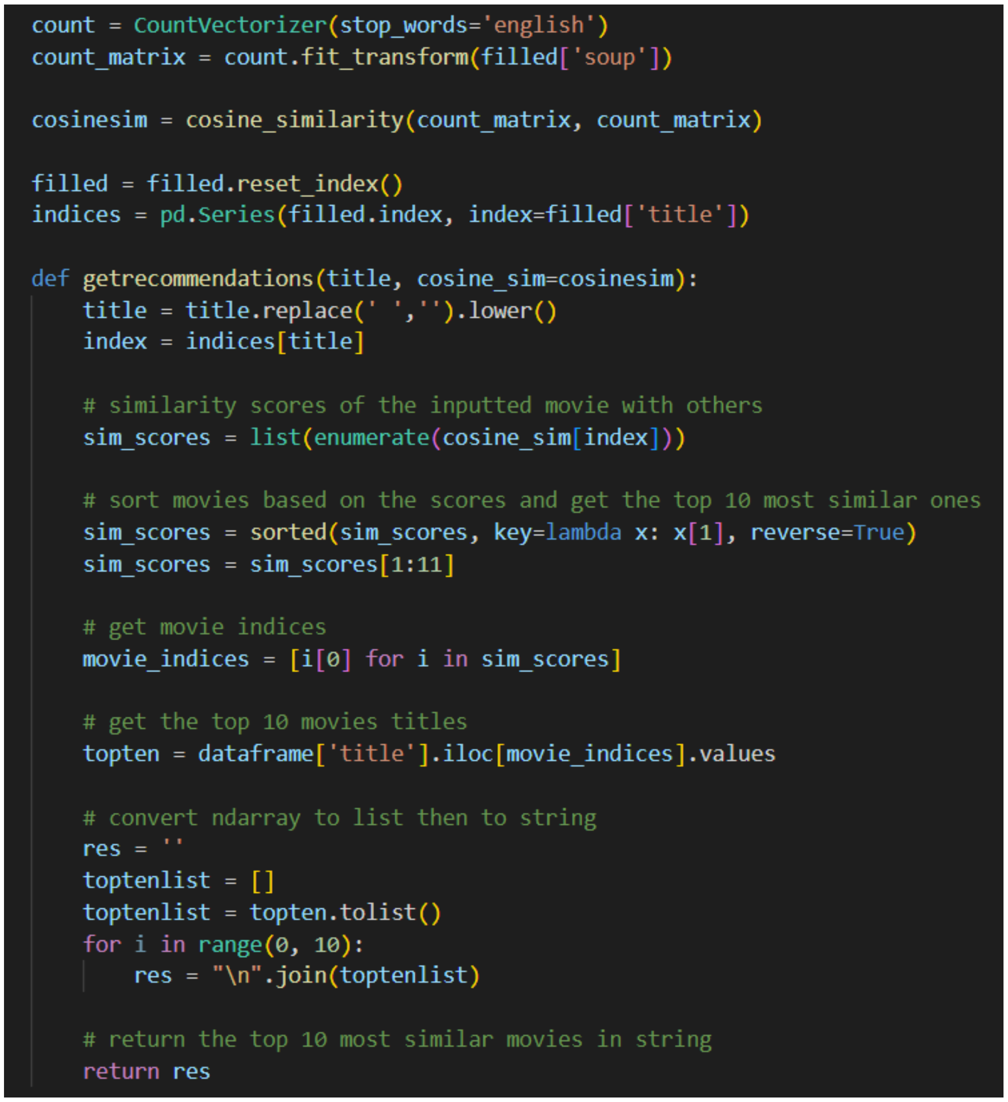


public
details
id character varying
type character varying
title character varying
director character varying
casts character varying
country character varying
date_added timestamp without time zone
release_year integer
rating character varying
duration character varying
listed_in character varying
description character varying



```
CREATE TABLE details (
    id VARCHAR NOT NULL UNIQUE,
    type VARCHAR NOT NULL,
    title VARCHAR NOT NULL,
    director VARCHAR,
    casts VARCHAR,
    country VARCHAR,
    date_added TIMESTAMP,
    release_year INT NOT NULL,
    rating VARCHAR,
    duration VARCHAR,
    listed_in VARCHAR NOT NULL,
    description VARCHAR NOT NULL,
    PRIMARY KEY (id)
)
```

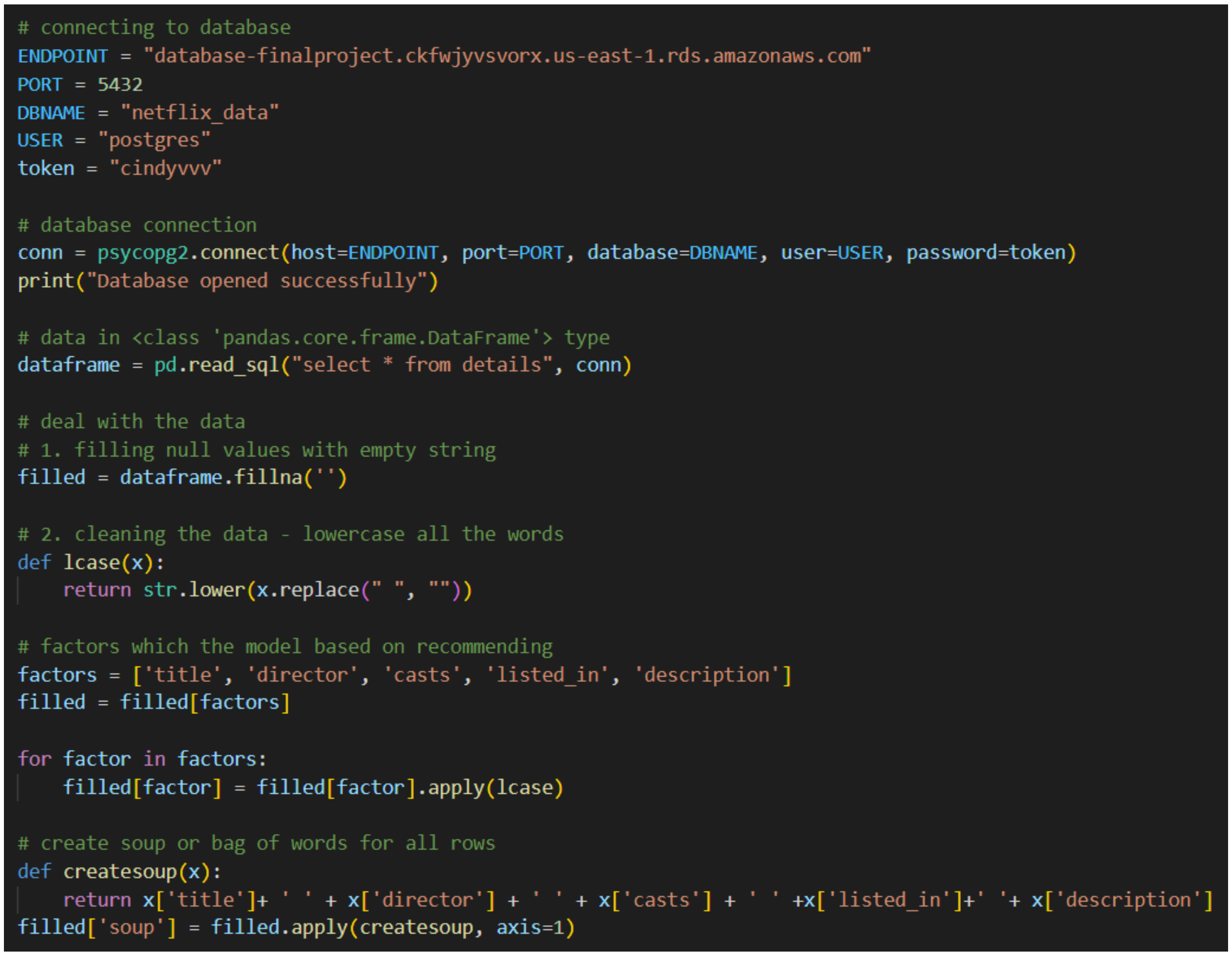
For the application's functions, I made a Netflix Recommendation System by using some of the attributes from the schema. I used the cosine similarity to calculate the similarity score between the inputted movie and other movies in the database. The fact that it is independent of magnitude makes it easier and faster to calculate compared to the others. The cosine similarity work by calculating the similarity scores, then we sort them and get the top ten most similar movies or shows. Before that, we also use sklearn functions like Count Vectorizer to help getting the recommendations too. Below is the function for the recommendation system.



```python
count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(filled['soup'])

cosinesim = cosine_similarity(count_matrix, count_matrix)

filled = filled.reset_index()
indices = pd.Series(filled.index, index=filled['title'])

def getrecommendations(title, cosine_sim=cosinesim):
    title = title.replace(' ','').lower()
    index = indices[title]

    # similarity scores of the inputted movie with others
    sim_scores = list(enumerate(cosine_sim[index]))

    # sort movies based on the scores and get the top 10 most similar ones
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:11]

    # get movie indices
    movie_indices = [i[0] for i in sim_scores]

    # get the top 10 movies titles
    topten = dataframe['title'].iloc[movie_indices].values

    # convert ndarray to list then to string
    res = ''
    toptenlist = []
    toptenlist = topten.tolist()
    for i in range(0, 10):
        res = "\n".join(toptenlist)

    # return the top 10 most similar movies in string
    return res
```

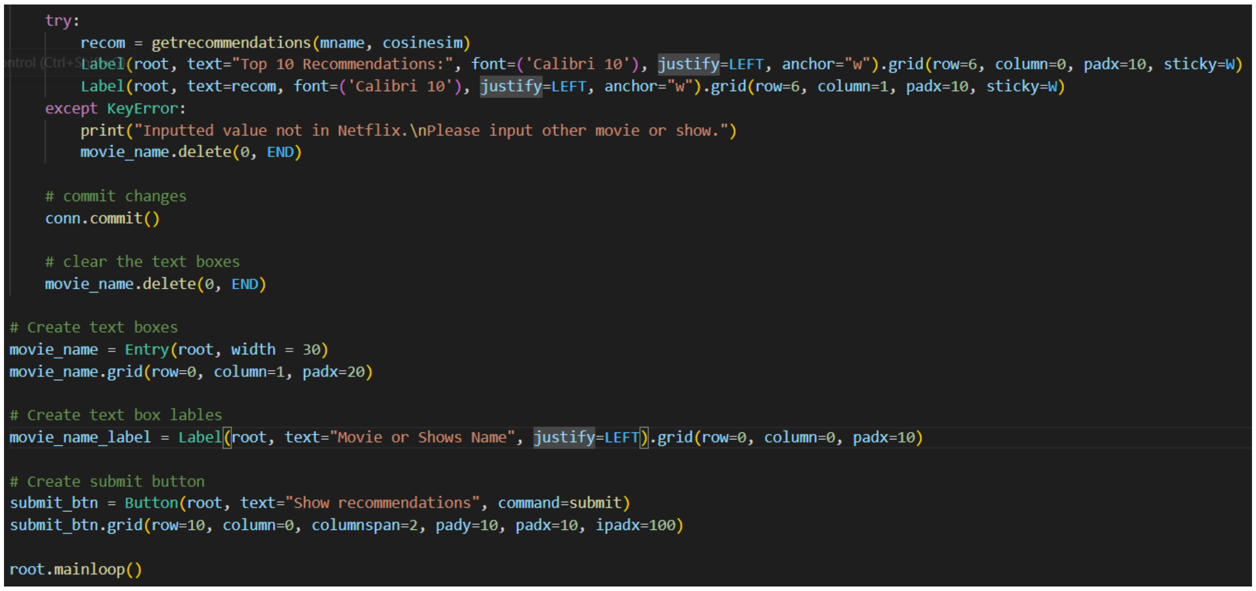
While the screenshot below shows the integration between the database and application as well as some processing with the data, like fill the null values and lowercase the words. I started by installing the library psycopg2 to connect the AWS database systems to this python application file. I also used panda’s library read_sql to query the database and get the data in panda dataframe type.

As the application is dependent on the database, it can be concluded that the application really needs a database. Data is the foundation of this whole project, and every input and output are related to the database. Beside recommendation system, I am sure that there are still lot of ways to work and make use of this database too.



```python
# connecting to database
ENDPOINT = "database-finalproject.ckfwjyvsvorx.us-east-1.rds.amazonaws.com"
PORT = 5432
DBNAME = "netflix_data"
USER = "postgres"
token = "cindyvvv"

# database connection
conn = psycopg2.connect(host=ENDPOINT, port=PORT, database=DBNAME, user=USER, password=token)
print("Database opened successfully")

# data in <class 'pandas.core.frame.DataFrame'> type
dataframe = pd.read_sql("select * from details", conn)

# deal with the data
# 1. filling null values with empty string
filled = dataframe.fillna('')

# 2. cleaning the data - lowercase all the words
def lcase(x):
    return str.lower(x.replace(" ", ""))

# factors which the model based on recommending
factors = ['title', 'director', 'casts', 'listed_in', 'description']
filled = filled[factors]

for factor in factors:
    filled[factor] = filled[factor].apply(lcase)

# create soup or bag of words for all rows
def createsoup(x):
    return x['title']+ ' ' + x['director'] + ' ' + x['casts'] + ' ' +x['listed_in']+' '+ x['description']
filled['soup'] = filled.apply(createsoup, axis=1)
```

After integrating the data, the database, and the application, last is the GUI or the interface for the application. In this final project, I choose to use Tkinter as the GUI. Despite having no experience using Tkinter before, I still choose to give it a try and make a simple interface with only input and output. Below is part of the Tkinter interface used in this homework. There are also some lines of code for the output and try-except for the KeyError value that will appears when the inputted value is not in the database. Therefore, to prevent this kind of error appearing, I used try-except and instead output of trying again will appear in the terminal.

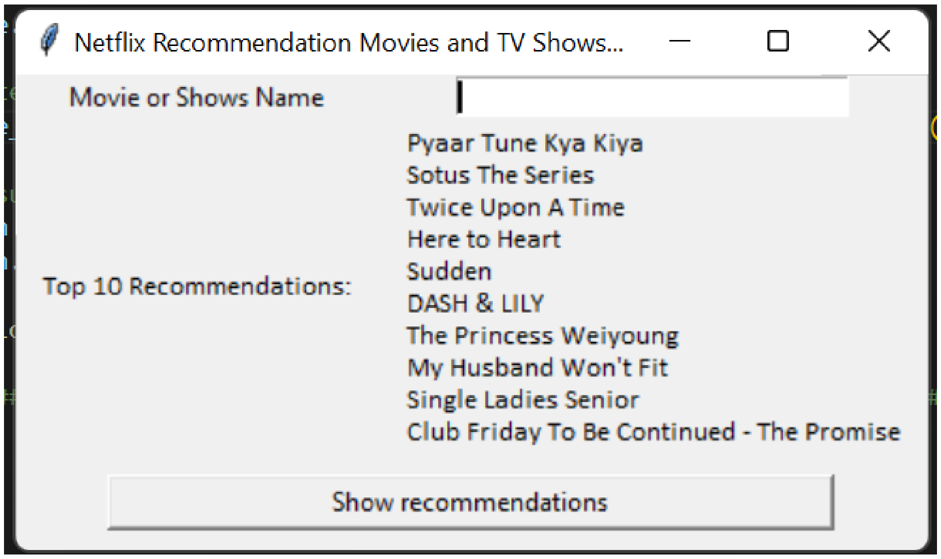


```
    try:
        recom = getrecommendations(mname, cosinesim)
        Label(root, text="Top 10 Recommendations:", font=('Calibri 10'), justify=LEFT, anchor="w").grid(row=6, column=0, padx=10, sticky=W)
        Label(root, text=recom, font=('Calibri 10'), justify=LEFT, anchor="w").grid(row=6, column=1, padx=10, sticky=W)
    except KeyError:
        print("Inputted value not in Netflix.\nPlease input other movie or show.")
        movie_name.delete(0, END)

    # commit changes
    conn.commit()

    # clear the text boxes
    movie_name.delete(0, END)

# Create text boxes
movie_name = Entry(root, width = 30)
movie_name.grid(row=0, column=1, padx=20)

# Create text box lables
movie_name_label = Label(root, text="Movie or Shows Name", justify=LEFT).grid(row=0, column=0, padx=10)

# Create submit button
submit_btn = Button(root, text="Show recommendations", command=submit)
submit_btn.grid(row=10, column=0, columnspan=2, pady=10, padx=10, ipadx=100)

root.mainloop()
```

Above are all the details of this whole project, which starts from how I get the data, where the data from, to the integration between the data and the database. There is also explanation about the integration between the database and the application as well as the application interface. It has been a long journey from how we first joined this class to this final project, and it feels amazing to see how far we have gone. After finishing this final project, I felt like I have learned a lot through this final project and through this class.

Results:

Input = Bridgerton

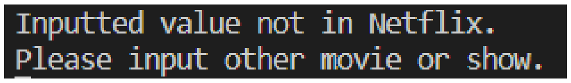


Top 10 Recommendations:

```
Pyaar Tune Kya Kiya
Sotus The Series
Twice Upon A Time
Here to Heart
Sudden
DASH & LILY
The Princess Weiyoung
My Husband Won't Fit
Single Ladies Senior
Club Friday To Be Continued - The Promise
```

Input = abcde (not in Netflix database)

Output shown in terminal:



```
Inputted value not in Netflix.
Please input other movie or show.
```