VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CSE3020 – Data Visualization J Component

Slot - D1 + TD1

NETFLIX DASHBOARD

Development Of Visual Idiom To Understand Usage of Netflix

TEAM MEMBERS

NAME	REGISTRATION NUMBER
PAYAL MAHESHWARI	20BCE2759
NAMAN CHADHA	20BCE0888
K. LOKESH	20BCE2621
VAISHNAVI SELVAKASI	20BCE2756

SUBMITTED TO:

Prof. Margret Anouncia S
Data Visualization Professor
smargretanouncia@vit.ac.in

I. ABSTRACT

Netflix generates a lot of data. One of the ways that we gain useful insights is by visualizing that data in dashboards which allow us to comprehend large amounts of information quickly. Thus to understand and get a deeper insight of how Netflix is being used across the world and to visualize the top countries that use Netflix and also to visualize the country wise statistics ratings we are using Dashboard created using Python and Dash Plotly framework. Data visualization is an efficient means to represent distributions and structures of datasets and reveal hidden patterns in the data. This project introduces the basic concept and pipeline of data visualization, and provides an overview of related data processing techniques for depicting the Netflix database.

II. INTRODUCTION

Today, having the ability to monitor multiple key performance indicators (KPIs) and metrics is crucial to succeeding in business. No one has the time to analyze and interpret large amounts of data manually. By investing in robust data visualization software, your organization can streamline its data collection, analysis, and collaboration with dashboards that help users to quickly digest critical information and make smart business decisions in real time.

A dashboard is a data visualization tool that monitors, analyzes, and displays key performance indicators (KPIs), metrics, and critical data points. Dashboards enable both technical and non-technical users to comprehend and apply business intelligence to make better decisions. Users actively participate in the analytics process by compiling data, visualizing trends or occurrences, and revealing an objective view of performance metrics that can be understood immediately.

Dashboards display data in the form of charts, tables, and gauges. Viewers use these visualizations to assess the organization's health in relation to predefined goals and industry benchmarks.

Using Dashboards leaves us with lots of benefits such as:

1. Visualize multiple Key Performance Indicators(KPIs) at once
2. Make data easier to understand
3. Increase accessibility and collaboration
4. Create reports on the fly

Most organizations utilize a variety of services to track KPIs and metrics, including marketing automation platforms, email marketing platforms, CRM tools, and many more. Monitoring and analyzing the data from each of these tools individually wastes precious time and resources.
Using a data visualization dashboard, users receive a bird’s eye view of the data from each of these platforms in one centralized location, with the ability to quickly understand what it means for the business. The user can then drill down deeper into any aspect of the data, comparing it to established KPIs, which help us to understand what’s working and where there’s room for improvement.

A dashboard does not require you to be a data scientist to use and comprehend it. The average user can quickly scan data visualization dashboards to get a high-level view of key data points without having to sift through spreadsheets, emails, or documents to find answers to critical business questions.

Dashboards make it simple for teams to collaborate, whether they work in the office, remotely, or in the field. Cloud-based dashboard tools are updated in real time and can be accessed via any browser. They aid in keeping everyone on the same page and working toward the same objectives.

Users can also create a link to their dashboards that they can share with stakeholders both inside and outside the organization.

It's crucial to leave the old habit of generating reports at the end of the month, quarter, or year in today's fast-paced global business environment. You can make quick changes by using dashboards that update in real time so that there is no room for mistakes because of outdated data.

Dashboards are easily accessible by every member of a team and its information can be extremely helpful in decision making. There is no need for a person-in-charge to be present to make a decision based on the given data

Due to the above discussed reasons, Dashboards have proved to be a very useful tool because of its versatility and convenience to use. They have become a necessity if visualization of multiple types is required in an intelligible way.

III. MOTIVATION

Users will be able to build dynamic dashboards on their own. Our platform should be self-service so that a centralized team does not become a bottleneck in addressing the various needs of engineers and the services they own.
As a result of the foregoing, it must support a wide range of use cases and usage patterns, so the solution must be highly adaptable and allow for the creation of custom dashboards.
Our goal, as part of the project, is to enable thorough visualization of the netflix dashboard with various qualitative and quantitative data extracted in the form of data sets so that the analysis process will become easier at the back end.

IV. USER REQUIREMENT

Collection of user requirements, their needs and the purpose is the main concern of our project. While having various different visualizations for netflix data such as show type, title, director and many more.

Users need a proper tool to define the vast variety of shows across every country around the globe. Namely, to analyze the popularity of a specific show in each country, the types of genres on which shows are being made, the effect of the show on the population and many other more visualizations when using and accessing data from various sources.

To properly discuss and interpret the same, the project works to meet these requirements of bringing this novel idea and approach for the global population.

V. TOOLS AND PACKAGES USED

To make the application completely interactive and understandable various tools and packages need to be used. Here we would be using-

- PYTHON
- PYCHARM
- DASH
- SEABORN
- PANDAS
- PLOTLY
- HTML
- CSS
- BOOTSTRAP
- FLASK
- SQLALCHEMY
- SQLITE
- GEOPANDAS
- CSV DATASET

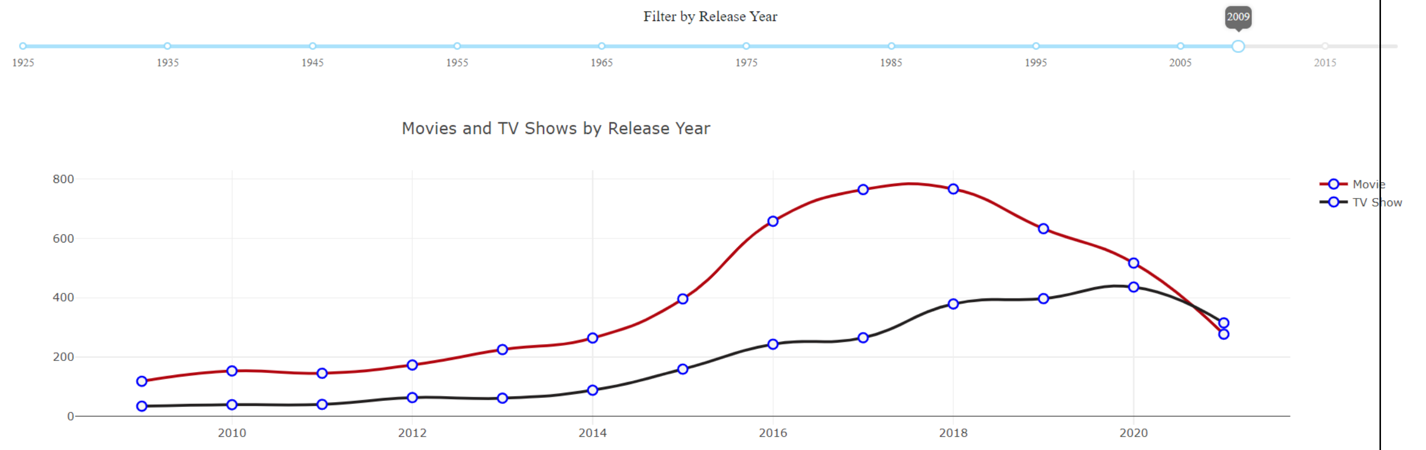
VI. VISUAL IDIOMS & DATA PREFERENCE

It's no secret that data can be very powerful — when you can actually understand what it's telling you, that is. It's not easy to get clear takeaways by looking at a slew of numbers and stats. You need to have the data presented in a logical, easy-to-understand way so you can apply your learnings in an effective way. The human brain processes visual information better than it processes text — so using charts, graphs, and design elements, data visualization can help you explain trends and stats much more easily. Different types of data demand different visualization and analysis methods. While determining how you'll visualize your data, one of the first things you'll want to do is keep the following best practices in mind.

- Choose the best visual for your data and its purpose.
- Ensure your data is easily understandable and viewable.
- Offer necessary context for your audience in and around your visual.
- Keep your visual as simple and straightforward as possible.
- Educate your audience with your visuals.

With these best practices in mind, you may now be wondering how to actually show your data in an effective way.

Line chart



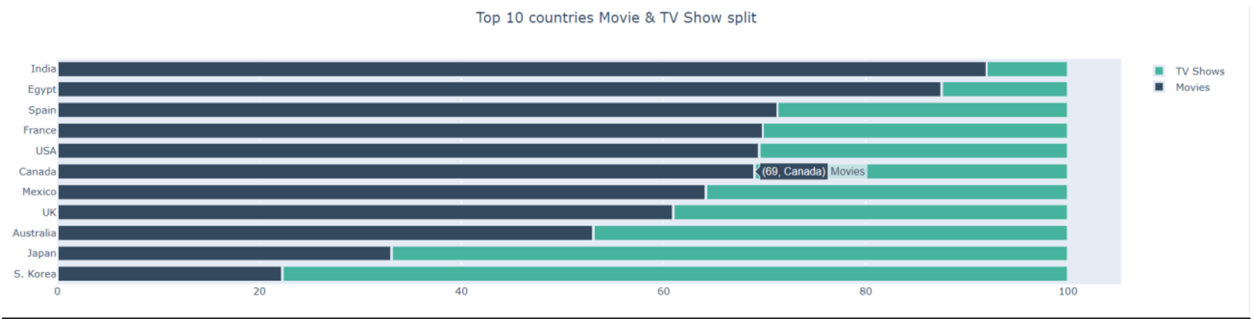
INTERPRETATION:

➢ The following graphs depicts the number of movies and tv shows by release year. With the help of a slider, we can toggle across the different release years.

➢ As we can see the total number of contents(movies and TV- shows) produced, reached its peak during the time interval 2015-2018.

➢ Plotly Express provide sliders, but with implicit animation using the "animate" method. Through this we can observe the change of data displayed or style of a plot.

<u>HORIZONTAL STACKED BAR CHART</u>

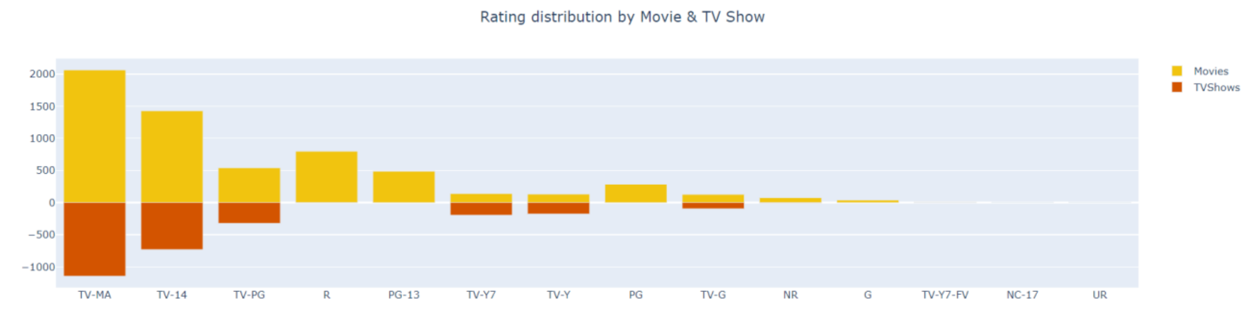


INTERPRETATION:

➢ The above graph is a horizontally stacked bar graph which gives us an idea about the contents produced by top 10 content producing countries. The graph shows the proportion of movies produced with respect to TV shows.

➢ As you can see India leads this list and the proportion of movies produced is comparatively high.

<u>STREAM BAR GRAPH</u>

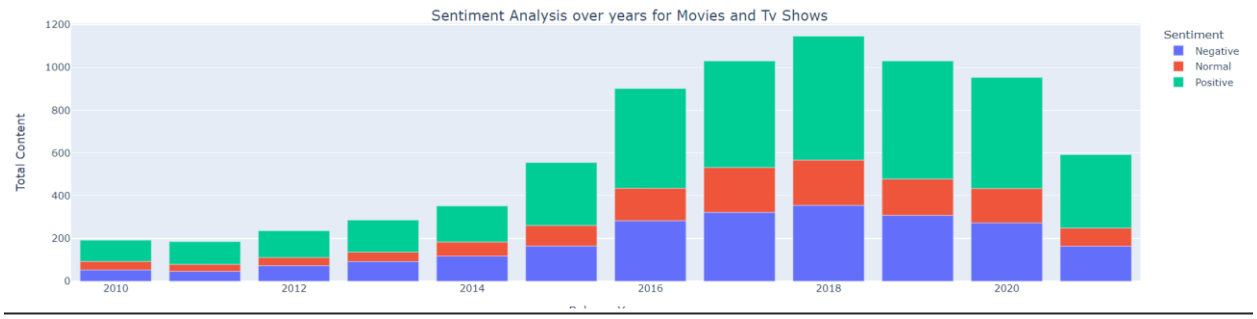


INTERPRETATION:

➢ In the following bar graphs we can see the distribution of movies and Tv shows based on their type of Maturity ratings.

➢ The positive Y axis depicts the no of movies made with reSpect to the different Maturity ratings and the negative Y Axis shows the no of TV shows made with respect to their maturity ratings.

STACKED BAR GRAPH

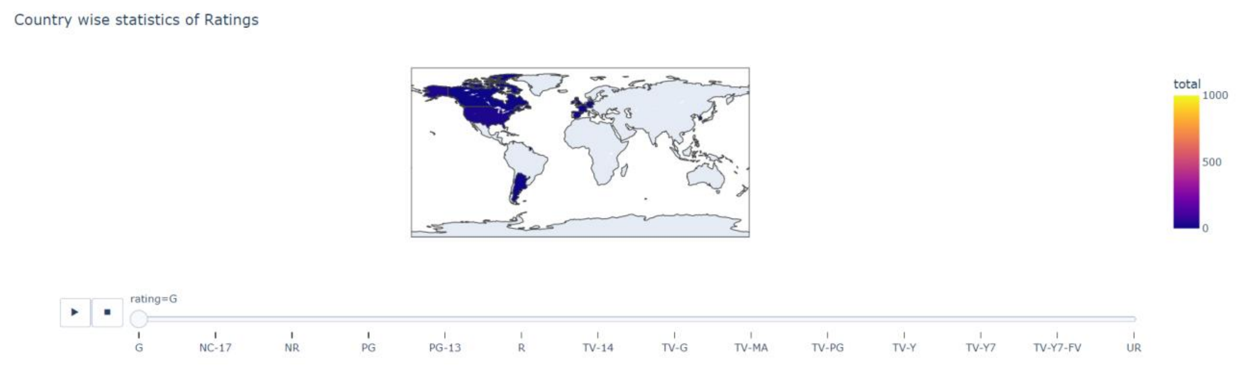


INTERPRETATION:

➢ The following graph is a stacked bar graph. This depicts the annual distribution of movies and TV shows based on the sentimental feedbacks - where blue depicts negative, red depicts Normal and Green depicts Positive .

➢ From the graph we can conclude that over the years, as the number of contents grew the sentimental value of contents deviated toward the positive side

CHOROPLETH MAP



INTERPRETATION:

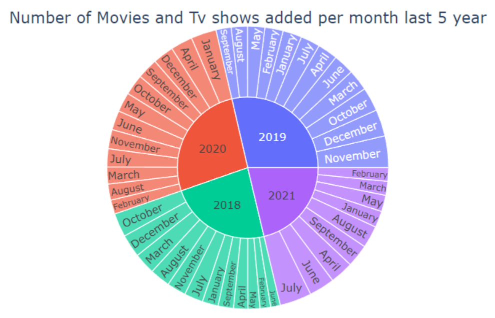
➢ The above geospatial map shows country wise statistics of ratings .It depicts number of contents(tv shows and movies) for a particular rating from all regions. With the help of a slider we can toggle across the different ratings of movies.

➢ For example , we have total 2458 contents from USA under TV-Y7 rating . Similarly we can see that for NR rating we have total 1679 contents from USA.

➢ So we interpret from the map that India is the second largest producer of TV-14,TV- Y,TV-Y7-FV and UR rated contents and USA is the largest producer of contents in all categories.

➢ Plotly Express provide sliders, but with implicit animation using the "animate" method. Through this we can observe the change of data displayed or style of a plot.

SUNBURST MAP



INTERPRETATION:

➢ The above pie chart shows the amount of entertainment content like TV shows and movies released in the past three years that is from January 2018 to January 2020. This chart gives a brief idea of how the movies and tv shows were released in the past 49 months.

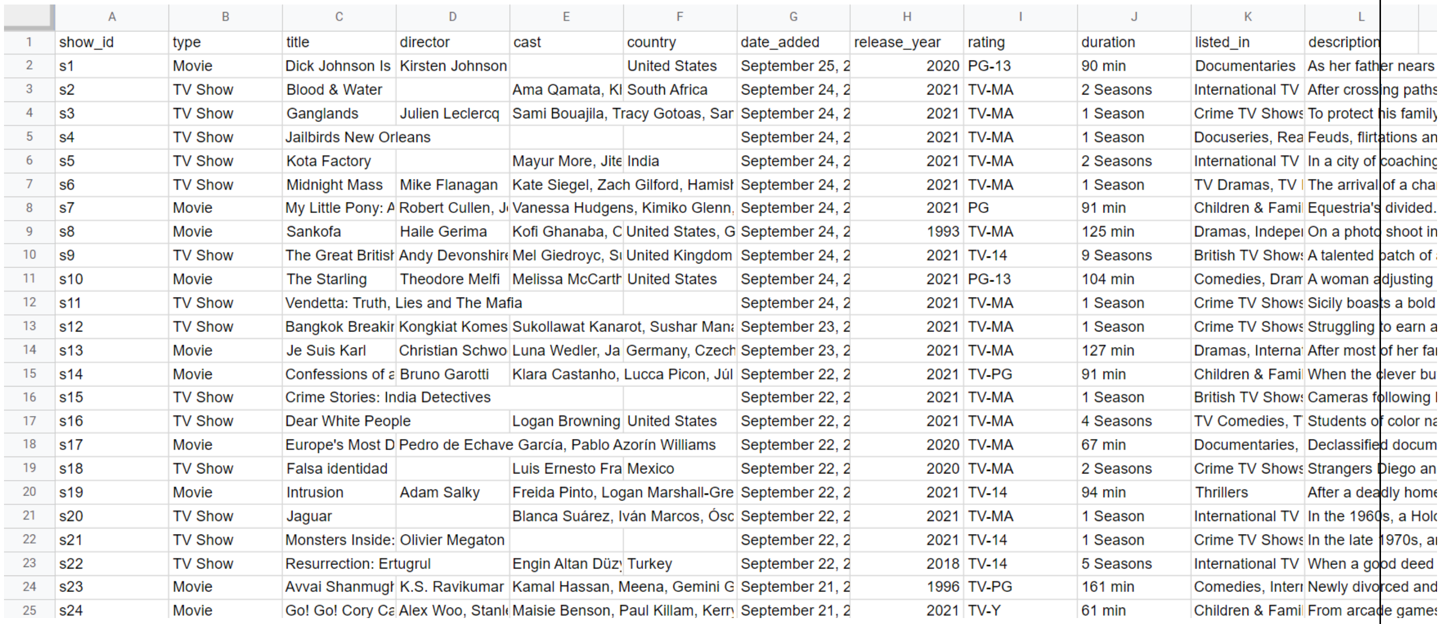
➢ So as we can interpret from the graph, the total amount of content produced in 2019 is more in number when compared to 2018.

➢ Also from the pie chart, it is evident that the maximum amount of content was produced in November 2019. So we can interpret that the people are accepting this kind of entertainment and people started enjoying the content hence the demand is increasing.

VII. DATASET

The searching and fetching of the dataset was the most annoying task, but thanks to kaggle the dataset for the Netflix movies and TVshows could be easily found on: https://www.kaggle.com/datasets/shivamb/netflix-shows/version/3

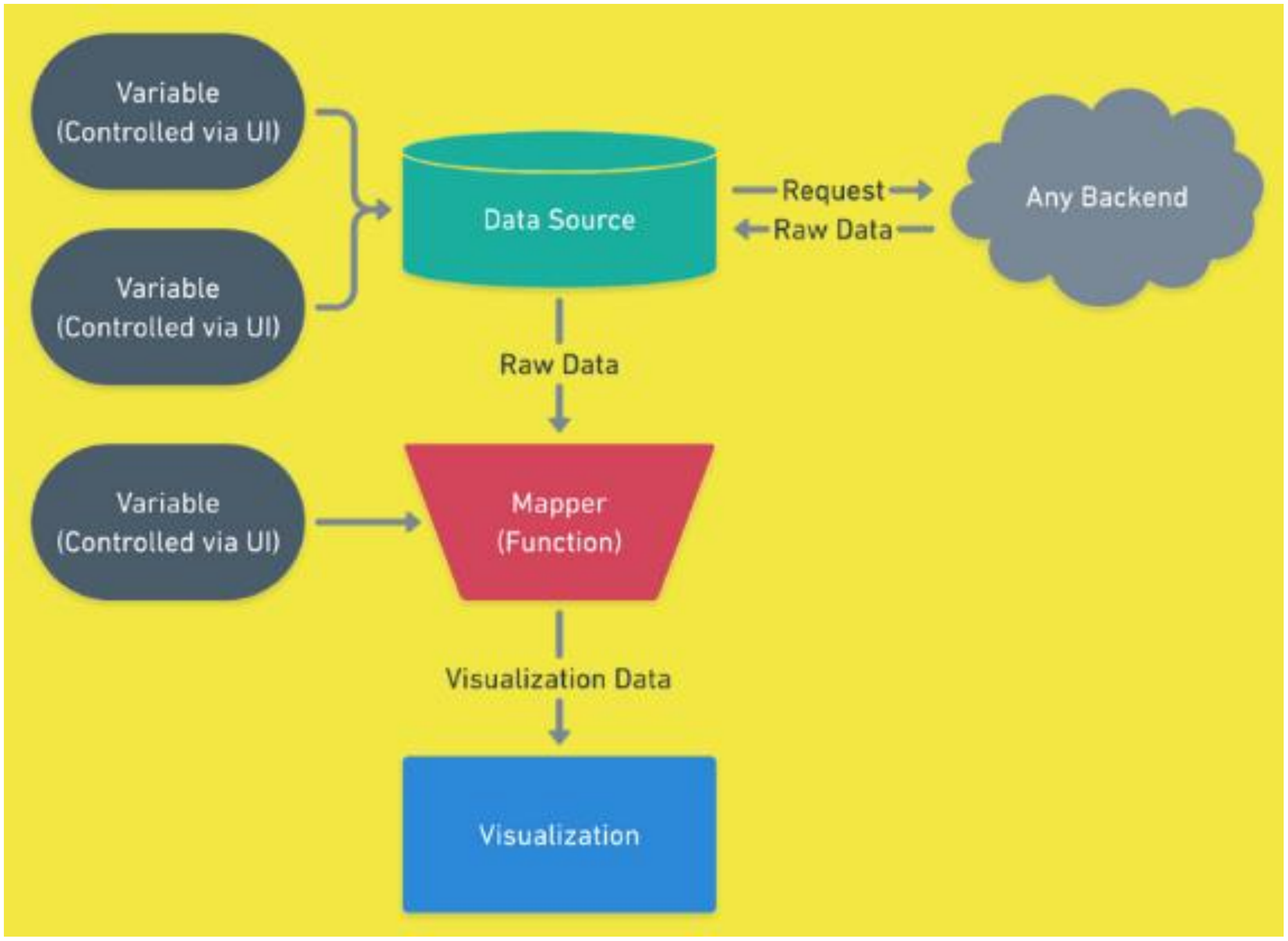
Netflix Sample Dataset:



	A	B	C	D	E	F	G	H	I	J	K	L
1	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
2	s1	Movie	Dick Johnson Is	Kirsten Johnson		United States	September 25, 2	2020	PG-13	90 min	Documentaries	As her father nears
3	s2	TV Show	Blood & Water		Ama Qamata, Kl	South Africa	September 24, 2	2021	TV-MA	2 Seasons	International TV	After crossing paths
4	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Sar		September 24, 2	2021	TV-MA	1 Season	Crime TV Shows	To protect his family
5	s4	TV Show	Jailbirds New Orleans				September 24, 2	2021	TV-MA	1 Season	Docuseries, Rea	Feuds, flirtations an
6	s5	TV Show	Kota Factory		Mayur More, Jite	India	September 24, 2	2021	TV-MA	2 Seasons	International TV	In a city of coaching
7	s6	TV Show	Midnight Mass	Mike Flanagan	Kate Siegel, Zach Gilford, Hamish		September 24, 2	2021	TV-MA	1 Season	TV Dramas, TV	The arrival of a cha
8	s7	Movie	My Little Pony: A	Robert Cullen, J	Vanessa Hudgens, Kimiko Glenn,		September 24, 2	2021	PG	91 min	Children & Famil	Equestria's divided.
9	s8	Movie	Sankofa	Haile Gerima	Kofi Ghanaba, C	United States, G	September 24, 2	1993	TV-MA	125 min	Dramas, Indepe	On a photo shoot in
10	s9	TV Show	The Great British	Andy Devonshire	Mel Giedroyc, Su	United Kingdom	September 24, 2	2021	TV-14	9 Seasons	British TV Shows	A talented batch of
11	s10	Movie	The Starling	Theodore Melfi	Melissa McCarth	United States	September 24, 2	2021	PG-13	104 min	Comedies, Dram	A woman adjusting
12	s11	TV Show	Vendetta: Truth, Lies and The Mafia				September 24, 2	2021	TV-MA	1 Season	Crime TV Shows	Sicily boasts a bold
13	s12	TV Show	Bangkok Breakir	Kongkiat Komes	Sukollawat Kanarot, Sushar Mana		September 23, 2	2021	TV-MA	1 Season	Crime TV Shows	Struggling to earn a
14	s13	Movie	Je Suis Karl	Christian Schwo	Luna Wedler, Ja	Germany, Czech	September 23, 2	2021	TV-MA	127 min	Dramas, Interna	After most of her fa
15	s14	Movie	Confessions of a	Bruno Garotti	Klara Castanho, Lucca Picon, Júl		September 22, 2	2021	TV-PG	91 min	Children & Famil	When the clever bu
16	s15	TV Show	Crime Stories: India Detectives				September 22, 2	2021	TV-MA	1 Season	British TV Shows	Cameras following
17	s16	TV Show	Dear White People		Logan Browning	United States	September 22, 2	2021	TV-MA	4 Seasons	TV Comedies, T	Students of color na
18	s17	Movie	Europe's Most D	Pedro de Echave García, Pablo Azorín Williams			September 22, 2	2020	TV-MA	67 min	Documentaries,	Declassified docum
19	s18	TV Show	Falsa identidad		Luis Ernesto Fra	Mexico	September 22, 2	2020	TV-MA	2 Seasons	Crime TV Shows	Strangers Diego an
20	s19	Movie	Intrusion	Adam Salky	Freida Pinto, Logan Marshall-Gre		September 22, 2	2021	TV-14	94 min	Thrillers	After a deadly home
21	s20	TV Show	Jaguar		Blanca Suárez, Iván Marcos, Ósc		September 22, 2	2021	TV-MA	1 Season	International TV	In the 1960s, a Hol
22	s21	TV Show	Monsters Inside:	Olivier Megaton			September 22, 2	2021	TV-14	1 Season	Crime TV Shows	In the late 1970s, a
23	s22	TV Show	Resurrection: Ertugrul		Engin Altan Düzy	Turkey	September 22, 2	2018	TV-14	5 Seasons	International TV	When a good deed
24	s23	Movie	Avvai Shanmugh	K.S. Ravikumar	Kamal Hassan, Meena, Gemini G		September 21, 2	1996	TV-PG	161 min	Comedies, Inter	Newly divorced and
25	s24	Movie	Go! Go! Cory Ca	Alex Woo, Stanl	Maisie Benson, Paul Killam, Kerry		September 21, 2	2021	TV-Y	61 min	Children & Famil	From arcade games

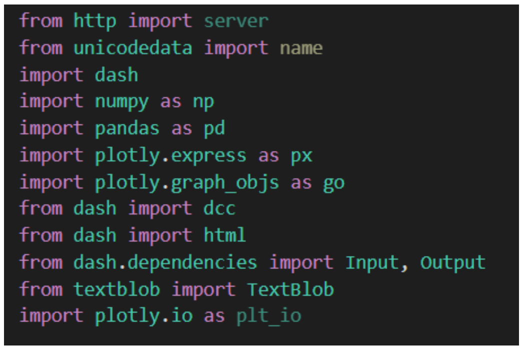
VIII. IMPLEMENTATION

The whole implementation was done on python while the platform for both the sections were different, one was completely developed in Pycharm while the other on Visual Studio Code.



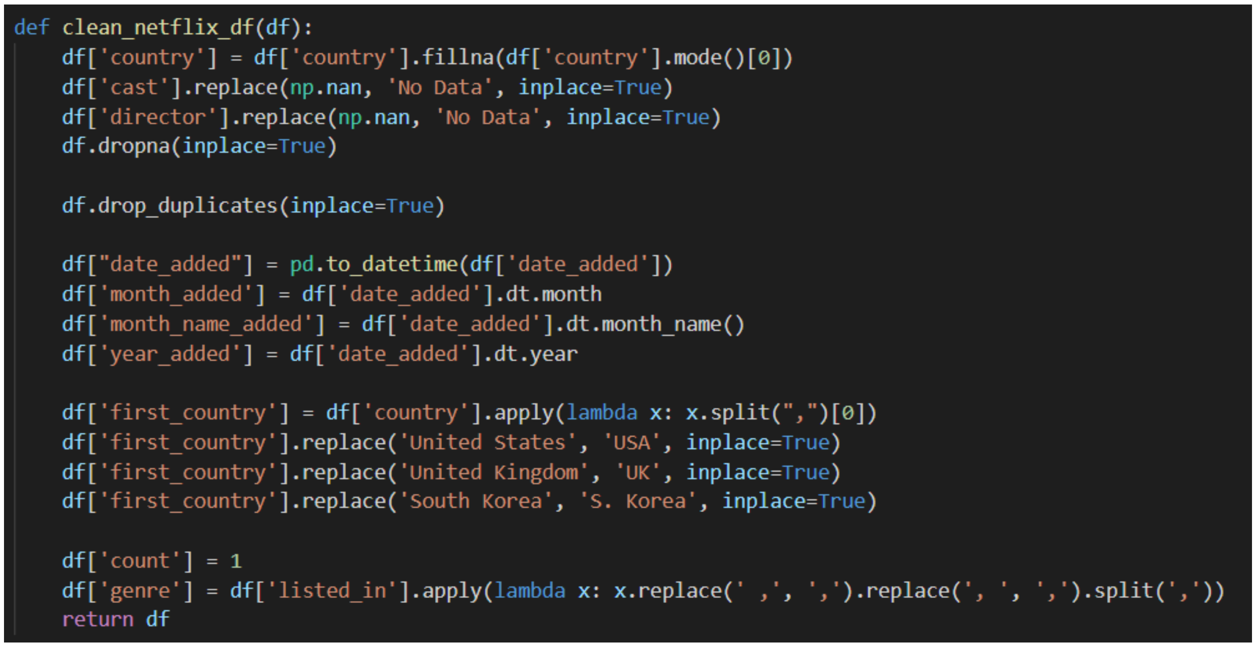
<u>Netflix Dataset Dashboard:</u>

For creating a dashboard we used a python framework dash plotly which provides an interactive interface for creating a dashboard and visualizing the raw data with ease. For this we implemented various modules and libraries such as numpy, pandas, plotly and textblob.



```
from http import server
from unicodedata import name
import dash
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objs as go
from dash import dcc
from dash import html
from dash.dependencies import Input, Output
from textblob import TextBlob
import plotly.io as plt_io
```

First we created a function to clean the data such as converting to lowercase, removing stopwords , removing duplicates etc.



```python
def clean_netflix_df(df):
    df['country'] = df['country'].fillna(df['country'].mode()[0])
    df['cast'].replace(np.nan, 'No Data', inplace=True)
    df['director'].replace(np.nan, 'No Data', inplace=True)
    df.dropna(inplace=True)

    df.drop_duplicates(inplace=True)

    df["date_added"] = pd.to_datetime(df['date_added'])
    df['month_added'] = df['date_added'].dt.month
    df['month_name_added'] = df['date_added'].dt.month_name()
    df['year_added'] = df['date_added'].dt.year

    df['first_country'] = df['country'].apply(lambda x: x.split(",")[0])
    df['first_country'].replace('United States', 'USA', inplace=True)
    df['first_country'].replace('United Kingdom', 'UK', inplace=True)
    df['first_country'].replace('South Korea', 'S. Korea', inplace=True)

    df['count'] = 1
    df['genre'] = df['listed_in'].apply(lambda x: x.replace(' ,', ',').replace(', ', ',').split(','))
    return df
```

Then we created different functions for creating different graphs for visualizing the pre processed data and these include line charts, horizontal stacked bar charts, sunburst graphs and choropleth graphs.

Then we created a dash app which is integrated with the flak app at the backend which is the web page of our project and then we used the layout method of dash framework to structure the different graphs and add different functionalities to the dashboard which is finally commenced through the callback method of dash app.

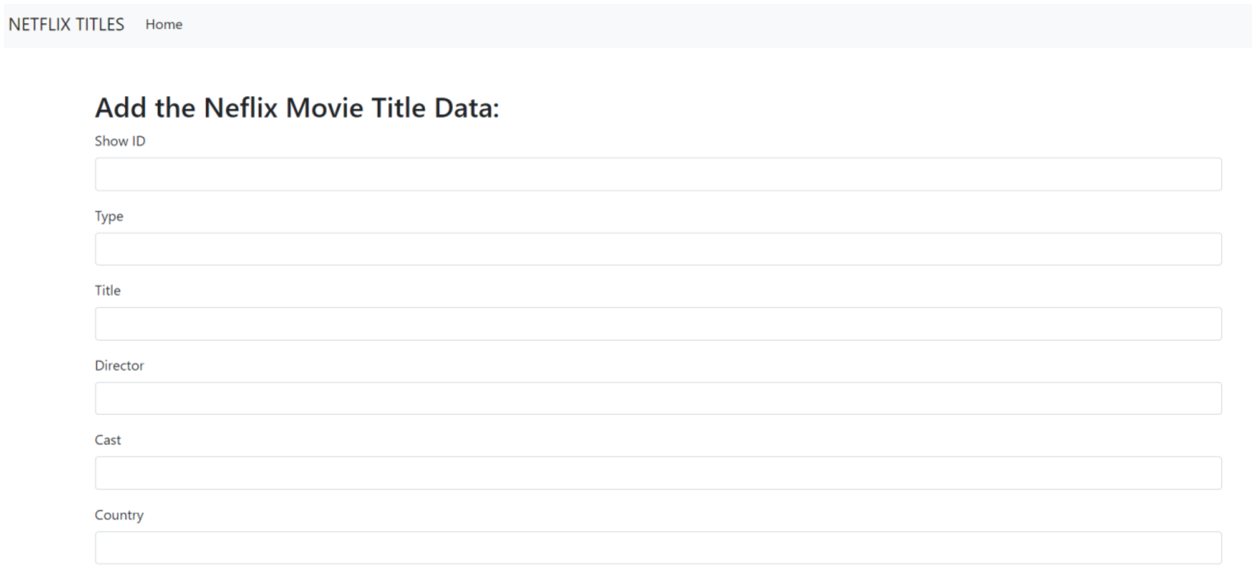
Personalized Dashboard

In this everything is similar to the previous dashboard except that we make our own dataset through our website which is stored in the database in the backend and then thaat database file is converted into CSV file which is then passed through the same code of dashboard creation and the updated personalized dashboard is displayed which is integrated with the website through flask at a different end point.

```
app1 = dash.Dash(server= flask_app, title="Netflix Dashboard",url_base_pathname='/personal/')
conn = sql.connect('naman1.db', isolation_level=None, detect_types=sql.PARSE_COLNAMES)
db_df = pd.read_sql_query("SELECT * FROM netflix", conn)
db_df.to_csv('netflix_dataset.csv', index=False)
netflix_df = pd.read_csv("D:/flask1/netflix_dataset.csv")
```

Website Frontend

In this part we used HTML, CSS and BOOTSTRAP to create the frontend part of our website which the user will visit and there they can enter and create their own dataset by filling the forms and clicking the submit button and the data will be stored in the database and the dashboard will be automatically updated and also the entered data will be displayed below.



Add the Neflix Movie Title Data:

Show ID

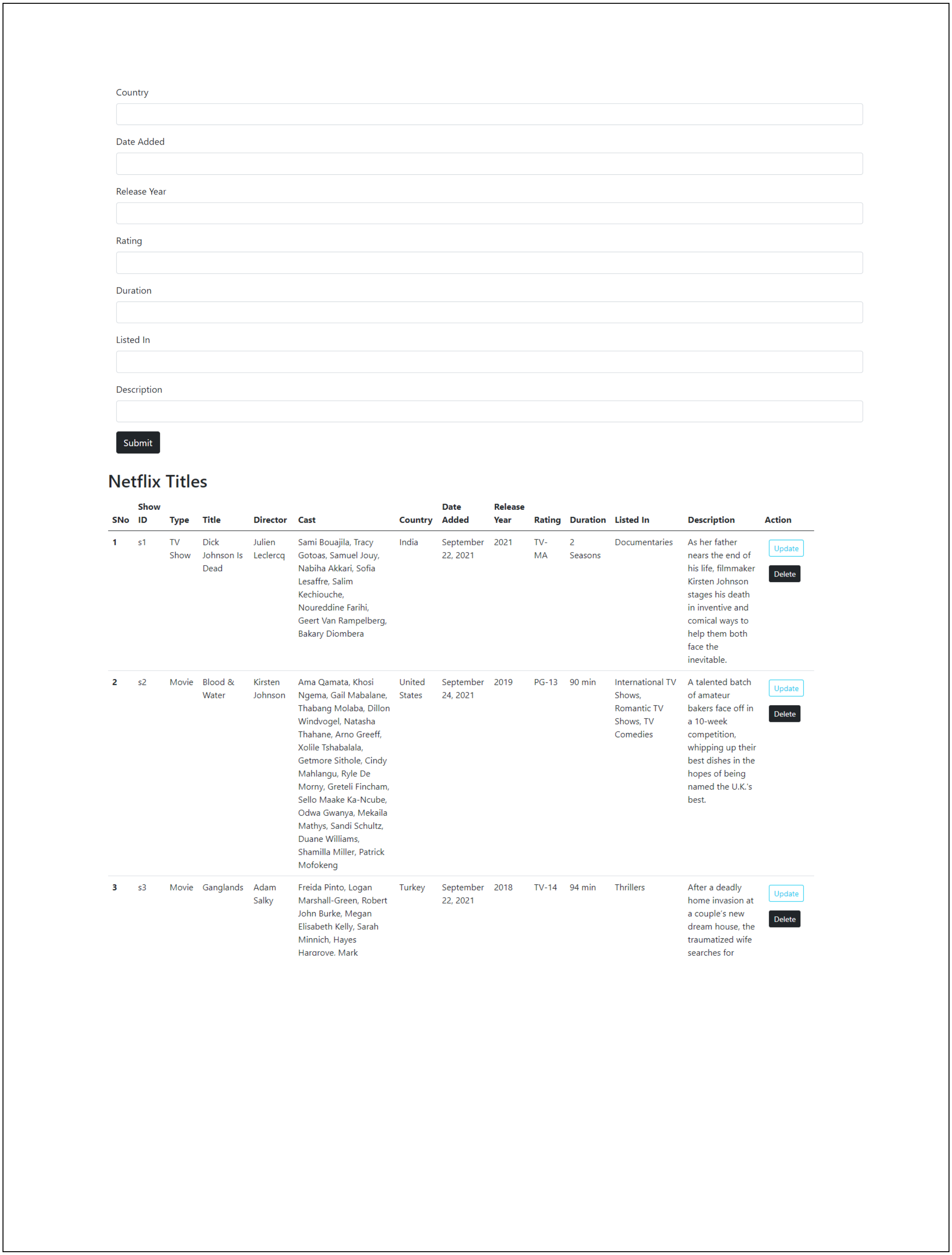
Type

Title

Director

Cast

Country



Country

Date Added

Release Year

Rating

Duration

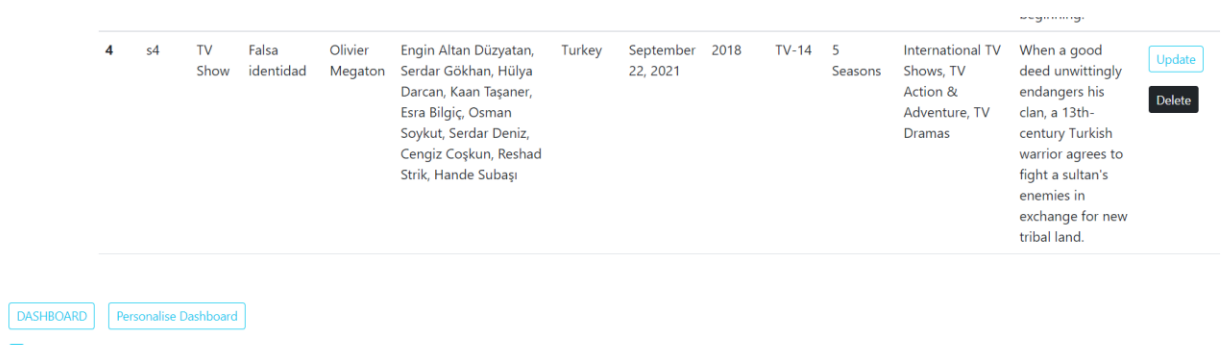
Listed In

Description

Submit

Netflix Titles

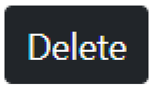
SNo	Show ID	Type	Title	Director	Cast	Country	Date Added	Release Year	Rating	Duration	Listed In	Description	Action
1	s1	TV Show	Dick Johnson Is Dead	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabiha Akkari, Sofia Lesaffre, Salim Kechiouche, Noureddine Farihi, Geert Van Rampelberg, Bakary Diombera	India	September 22, 2021	2021	TV-MA	2 Seasons	Documentaries	As her father nears the end of his life, filmmaker Kirsten Johnson stages his death in inventive and comical ways to help them both face the inevitable.	Update Delete
2	s2	Movie	Blood & Water	Kirsten Johnson	Ama Qamata, Khosi Ngema, Gail Mabalane, Thabang Molaba, Dillon Windvogel, Natasha Thahane, Arno Greeff, Xolile Tshabalala, Getmore Sithole, Cindy Mahlangu, Ryle De Morny, Greteli Fincham, Sello Maake Ka-Ncube, Odwa Gwanya, Mekaila Mathys, Sandi Schultz, Duane Williams, Shamilla Miller, Patrick Mofokeng	United States	September 24, 2021	2019	PG-13	90 min	International TV Shows, Romantic TV Shows, TV Comedies	A talented batch of amateur bakers face off in a 10-week competition, whipping up their best dishes in the hopes of being named the U.K.'s best.	Update Delete
3	s3	Movie	Ganglands	Adam Salky	Freida Pinto, Logan Marshall-Green, Robert John Burke, Megan Elisabeth Kelly, Sarah Minnich, Hayes Hargrove, Mark	Turkey	September 22, 2021	2018	TV-14	94 min	Thrillers	After a deadly home invasion at a couple's new dream house, the traumatized wife searches for	Update Delete



4	s4	TV Show	Falsa identidad	Olivier Megaton	Engin Altan Düzyatan, Serdar Gökhan, Hülya Darcan, Kaan Taşaner, Esra Bilgiç, Osman Soykut, Serdar Deniz, Cengiz Coşkun, Reshad Strik, Hande Subaşı	Turkey	September 22, 2021	2018	TV-14	5 Seasons	International TV Shows, TV Action & Adventure, TV Dramas	When a good deed unwittingly endangers his clan, a 13th-century Turkish warrior agrees to fight a sultan's enemies in exchange for new tribal land.	Update Delete

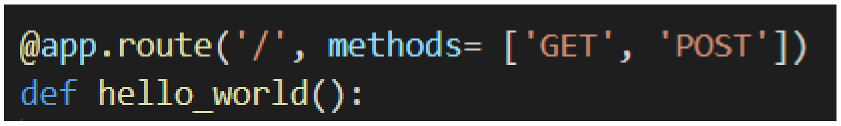
And also two buttons are also provided at the end of the page using which the user can visit both the Netflix dataset dashboard and the personalized dashboard which he created using his own data that he entered in the fields provided in the website.

The users are also provided with the update and delete option using which they can alter the data they entered to create their personalized dashboard.

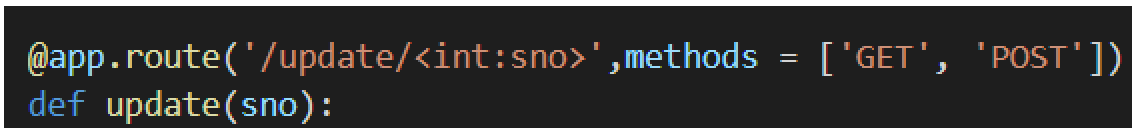


Website Backend

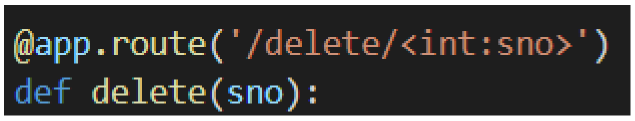
For the backend of the website we used flask with which we created various end points of our website such as the index page and the update page and both the dashboard pages and then integrated all of them to serve as a single website.



```
@app.route('/', methods= ['GET', 'POST'])
def hello_world():
```

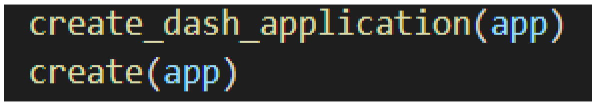


```
@app.route('/update/<int:sno>',methods = ['GET', 'POST'])
def update(sno):
```



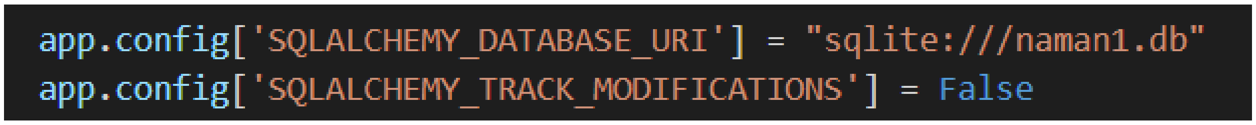
```
@app.route('/delete/<int:sno>')
def delete(sno):
```

Dashboard Integration code:

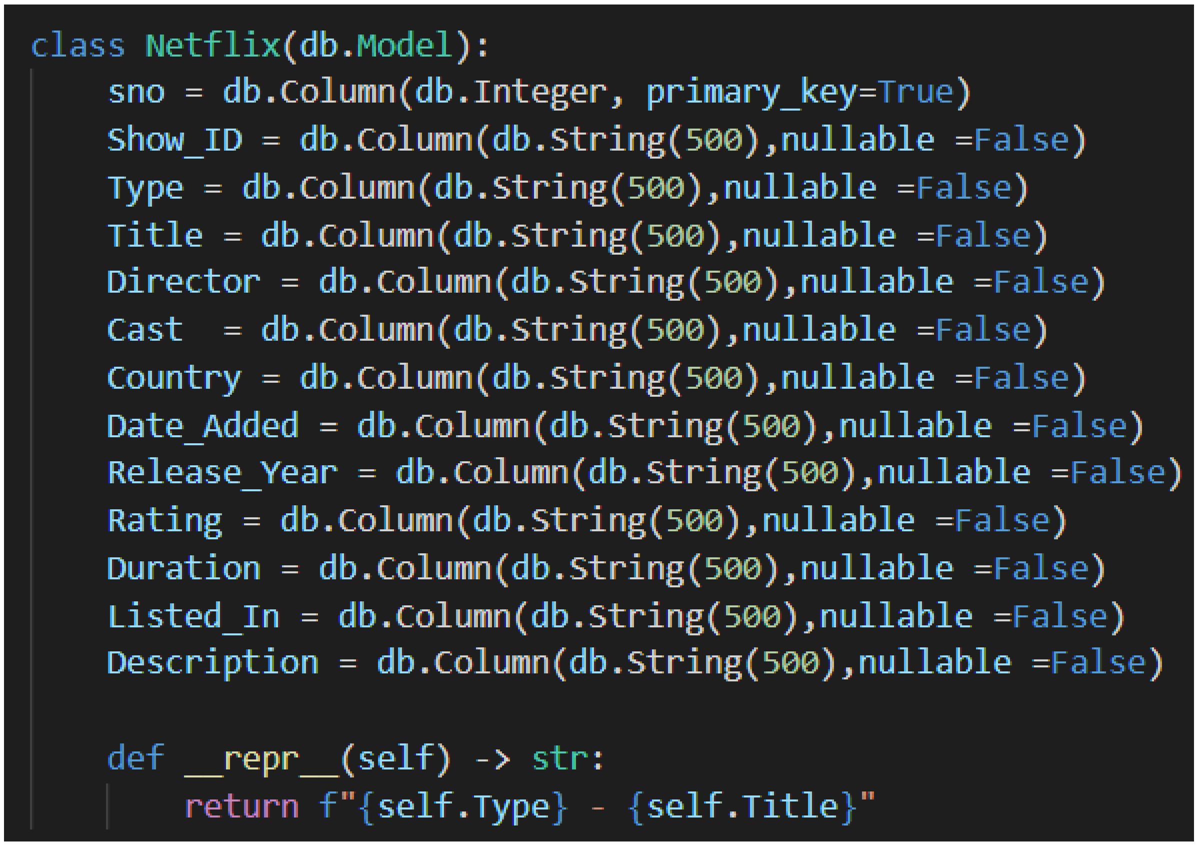


```
create_dash_application(app)
create(app)
```

We also used Sqlite database to store the data given by the user to create the dashboard.



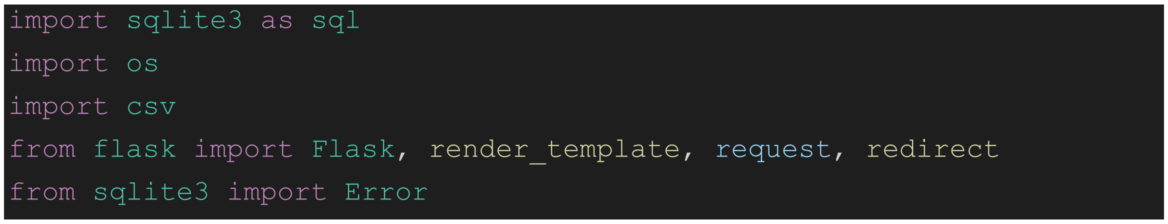
```
app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:///naman1.db"
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```



```
class Netflix(db.Model):
    sno = db.Column(db.Integer, primary_key=True)
    Show_ID = db.Column(db.String(500),nullable =False)
    Type = db.Column(db.String(500),nullable =False)
    Title = db.Column(db.String(500),nullable =False)
    Director = db.Column(db.String(500),nullable =False)
    Cast  = db.Column(db.String(500),nullable =False)
    Country = db.Column(db.String(500),nullable =False)
    Date_Added = db.Column(db.String(500),nullable =False)
    Release_Year = db.Column(db.String(500),nullable =False)
    Rating = db.Column(db.String(500),nullable =False)
    Duration = db.Column(db.String(500),nullable =False)
    Listed_In = db.Column(db.String(500),nullable =False)
    Description = db.Column(db.String(500),nullable =False)

    def __repr__(self) -> str:
        return f"{self.Type} - {self.Title}"
```

All Finalized Programs Of Our Website:

1. dashPersonal_applicaation.py



```
import sqlite3 as sql
import os
import csv
from flask import Flask, render_template, request, redirect
from sqlite3 import Error
```

```python
from turtle import goto
import dash
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objs as go
from dash import dcc
from dash import html
from dash.dependencies import Input, Output
from textblob import TextBlob
import plotly.io as plt_io



def create(flask_app):

    app1 = dash.Dash(server= flask_app, title="Netflix
Dashboard",url_base_pathname='/personal/')
    conn = sql.connect('naman1.db', isolation_level=None,
detect_types=sql.PARSE_COLNAMES)
    db_df = pd.read_sql_query("SELECT * FROM netflix", conn)
    db_df.to_csv('netflix_dataset.csv', index=False)
    netflix_df = pd.read_csv("D:/flask1/netflix_dataset.csv")
    def clean_netflix_df(df):
        df['Country'] = df['Country'].fillna(df['Country'].mode()[0])
        df['Cast'].replace(np.nan, 'No Data', inplace=True)
        df['Director'].replace(np.nan, 'No Data', inplace=True)
        df.dropna(inplace=True)

        df.drop_duplicates(inplace=True)

        df["Date_Added"] = pd.to_datetime(df['Date_Added'])
        df['month_added'] = df['Date_Added'].dt.month
        df['month_name_added'] = df['Date_Added'].dt.month_name()
        df['year_added'] = df['Date_Added'].dt.year

        df['first_Country'] = df['Country'].apply(lambda x:
x.split(",")[0])
        df['first_Country'].replace('United States', 'USA',
inplace=True)
```

```
        df['first_Country'].replace('United      Kingdom',      'UK',
inplace=True)
        df['first_Country'].replace('South    Korea',    'S.    Korea',
inplace=True)

        netflix_df['count'] = 1
        df['genre'] = df['Listed_In'].apply(lambda x: x.replace(' ,',
',').replace(', ', ',').split(','))
        return df
    netflix_df = clean_netflix_df(netflix_df)

    def fig_bar_horiz():
        country_order                                                    =
netflix_df['first_Country'].value_counts()[:11].index
        data_q2q3                =                netflix_df[['Type',
'first_Country']].groupby('first_Country')['Type'].value_counts().un
stack().loc[
            country_order]
        data_q2q3['sum'] = data_q2q3.sum(axis=1)
        data_q2q3_ratio          =          (data_q2q3.T           /
data_q2q3['sum']).T[['Movie',   'TV   Show']].sort_values(by='Movie',
ascending=False)[
                        ::-1]
        fig_bar = go.Figure()
        fig_bar.add_trace(go.Bar(
            y=data_q2q3_ratio.index,
            x=round(data_q2q3_ratio['Movie'] * 100, 2),
            name='Movies',
            orientation='h',

            marker=dict(
                color='#34495E',
                line=dict( width=3),
            )
        ))
        fig_bar.add_trace(go.Bar(
            y=data_q2q3_ratio.index,
            x=round(data_q2q3_ratio['TV Show'] * 100, 2),
            name='TV Shows',
            orientation='h',
            marker=dict(
                color='#45B39D',
                line=dict(width=3)
            )
        ))

        fig_bar.update_layout(barmode='stack',
                            title={
                                'text': "Top 10 countries Movie & TV
Show split",
                                'y': 0.9,
                                'x': 0.5,
                                'xanchor': 'center',
                                'yanchor': 'top'}
                            )
        return fig_bar

    def fig_bar_stacked():
        order =
pd.DataFrame(netflix_df.groupby('Rating')['count'].sum().sort_values
(ascending=False).reset_index())
        rating_order = list(order['Rating'])
        mf =
netflix_df.groupby('Type')['Rating'].value_counts().unstack().sort_i
ndex().fillna(0).astype(int)[rating_order]

        movie = mf.loc['Movie']
        tv = - mf.loc['TV Show']

        fig_stacked = go.Figure()
        fig_stacked.add_trace(go.Bar(x=movie.index, y=movie,
name='Movies',marker_color='#F1C40F'))
        fig_stacked.add_trace(go.Bar(x=tv.index, y=tv,
name='TVShows',marker_color='#D35400'))
        fig_stacked.update_layout(barmode='relative',
                                title={
                                        'text': 'Rating distribution
by Movie & TV Show',
                                        'y': 0.9,
                                        'x': 0.5,
                                        'xanchor': 'center',
                                        'yanchor': 'top'}
                                )
        return fig_stacked

    def fig_stack_without_flying():
        dfx = netflix_df[['Release_Year', 'Description']]
        dfx = dfx.rename(columns={'Release_Year': 'Release Year'})
        for index, row in dfx.iterrows():
            z = row['Description']
            testimonial = TextBlob(z)
            p = testimonial.sentiment.polarity
            if p == 0:
                sent = 'Normal'
            elif p > 0:
                sent = 'Positive'
            else:
                sent = 'Negative'
            dfx.loc[[index, 1], 'Sentiment'] = sent

        dfx          =          dfx.groupby(['Release          Year',
'Sentiment']).size().reset_index(name='Total Content')

        dfx = dfx[dfx['Release Year'] >= 2010]
        fig_stacked_without_fly   =   px.bar(dfx,   x="Release   Year",
y="Total Content",color='Sentiment')
        fig_stacked_without_fly.update_layout(title={
            'text': 'Sentiment Analysis over years for Movies and Tv
Shows',
            'y': 0.9,
            'x': 0.5,
            'xanchor': 'center',
            'yanchor': 'top'})
        return fig_stacked_without_fly

    def fig_pie_purst():
        fig_purst  =  px.sunburst(netflix_df[netflix_df['year_added']
>= 2018], path=['year_added', 'month_name_added'],
                                values='count',
color_continuous_scale='armyrose')
```

```
        fig_purst.update_layout(title={
            'text': 'Number of Movies and Tv shows added per month
last 5 year',
            'y': 0.9,
            'x': 0.5,
            'xanchor': 'center',
            'yanchor': 'top'})
        return fig_purst

    def world_map():
        df_country_year                                            =
netflix_df.groupby(by=['Country','Type','Rating']).count().reset_ind
ex()

df_country_year['total']=df_country_year.groupby(by=['Country'])['Ti
tle'].cumsum()


fig_world_map=px.choropleth(df_country_year.sort_values(by='Rating')
,   locations='Country',   title='Country   wise   statistics   of
Ratings',color='total',        locationmode='country        names',
animation_frame='Rating', range_color=[0,1000], )
        return fig_world_map

    app1.layout = html.Div(children=[

        html.H1(id='header',    children=[html.Div("Netflix    Title
Analysis", id='header-text')],
                style={'textAlign':  'center',  'color':  '#b20710'},
className="mb-3"),
        html.Br(),
        html.Br(),
        html.Br(),
        html.Div(children=[
            html.Div(children=[
                html.Div([
                    dcc.Graph(id='bar_fig',
figure=fig_bar_horiz())], id='FigBarGraphDiv')
            ], className="col-md-6"),
        ], className="row"),
        html.Div(children=[
            html.Div(children=[
                html.Div([
                    dcc.Graph(id='stack_fig',
figure=fig_bar_stacked())], id='StackedGraphDiv')

            ], className="col-md-6"),
            html.Div(children=[
                html.Div([
                    dcc.Graph(id='SentBarGraphDiv',
figure=fig_stack_without_flying()), ])
            ], className="col-md-6"),

            ], className="col-md-6"),
            html.Div(children=[
                html.Div(children=[
                    html.Div([
                    dcc.Graph(id='world_fig', figure=world_map())],
id='World_Fig')
            ], className="col-md-6"),

            ], className="row"),

        html.Div(children=[
            html.Div(children=[
                html.Div([
                    dcc.Graph(id='purst_fig',
figure=fig_pie_purst())],
                    id='PurstGraphDiv')




            ], className="col-md-6"),
            html.Div(children=[
                html.Label('Movie Statistics Calculator',
id='calculator'),
                html.Div([
                    dcc.Dropdown(id='dropDown', options=[{'label': x,
'value': x} for x in netflix_df['first_Country'].unique()],
                                value='Egypt'),
                    html.Br(),
                    html.Br(),
                    html.Table([
                        html.Tbody([
                            html.Tr([
                                html.Td("No. of Movies till date"),

                                html.Td([
                                    html.Div(
                                        id="val1"

                                    )

                                ])
                            ]),
                            html.Tr([
                                html.Td("No. of TV Shows till date"),

                                html.Td([
                                    html.Div(
                                        id="val2"

                                    )

                                ])
                            ]),
                            html.Tr([
                                html.Td("Top Actor"),

                                html.Td([
                                    html.Div(
                                        id="val3"

                                    )

                                ])
                            ]),
                            html.Tr([
                                html.Td("Top Director"),
                                html.Td([
                                    html.Div(
                                        id="val4"

                                    )

                                ])
                            ])

                        ])
                    ], className="table table-striped")
                ])
            ], className="col-md-6"),

        ], className="row"),

    ], className="container-fluid",)


    @app1.callback(
        [Output('val1',  'children'),  Output('val2',  'children'),
Output('val3', 'children'), Output('val4', 'children')],
        Input('dropDown', 'value')
    )
    def updateTable(dropDown):
        dfx = netflix_df[['Type', 'Country']]
        dfMovie = dfx[dfx['Type'] == 'Movie']
        dfTV = dfx[dfx['Type'] == 'TV Show']
        dfM1 = dfMovie['Country'].str.split(',', expand=True).stack()
        dfTV1 = dfTV['Country'].str.split(',', expand=True).stack()
        dfM1 = dfM1.to_frame()
        dfTV1 = dfTV1.to_frame()
        dfM1.columns = ['Country']
        dfTV1.columns = ['Country']
        dfM2                                                                =
dfM1.groupby(['Country']).size().reset_index(name='counts')
        dfTV2                                                               =
dfTV1.groupby(['Country']).size().reset_index(name='counts')
        dfM2['Country'] = dfM2['Country'].str.strip()
        dfTV2['Country'] = dfTV2['Country'].str.strip()
        val11 = dfM2[dfM2['Country'] == dropDown]
        val22 = dfTV2[dfTV2['Country'] == dropDown]
        val11 = val11.reset_index()
        val22 = val22.reset_index()

        if val11.empty:
            val1 = 0
        else:
            val1 = val11.loc[0]['counts']

        if val22.empty:
            val2 = 0
        else:
            val2 = val22.loc[0]['counts']

        # Top Actor
        dfA = netflix_df[['Cast', 'Country']]
        dfA1 = dfA[dfA['Country'].str.contains(dropDown, case=False)]
        dfA2 = dfA1['Cast'].str.split(',', expand=True).stack()
        dfA2 = dfA2.to_frame()
        dfA2.columns = ['Cast']
        dfA3                                                         =
dfA2.groupby(['Cast']).size().reset_index(name='counts')
        dfA3 = dfA3[dfA3['Cast'] != 'No Cast Specified']
        dfA3 = dfA3.sort_values(by='counts', ascending=False)
        if dfA3.empty:
            val3 = "Actor data from this country is not available"
        else:
            val3 = dfA3.iloc[0]['Cast']
        # Top Director
        dfD = netflix_df[['Director', 'Country']]
        dfD1 = dfD[dfD['Country'].str.contains(dropDown, case=False)]
        dfD2 = dfD1['Director'].str.split(',', expand=True).stack()
        dfD2 = dfD2.to_frame()
        dfD2.columns = ['Director']
        dfD3                                                         =
dfD2.groupby(['Director']).size().reset_index(name='counts')
        dfD3 = dfD3[dfD3['Director'] != 'No Director Specified']
        dfD3 = dfD3.sort_values(by='counts', ascending=False)
        if dfD3.empty:
            val4 = "Director data from this country is not available"
        else:
            val4 = dfD3.iloc[0]['Director']
        return val1, val2, val3, val4

    @app1.callback(Output('line_chart', 'figure'),
                [Input('slider_year', 'value')])
    def update_graph(slider_year):
        type_movie      =      netflix_df[(netflix_df['Type']      ==
'Movie')][['Type', 'Release_Year']]
        type_movie['type1'] = type_movie['Type']
        type_movie_1      =      type_movie.groupby(['Release_Year',
'type1'])['Type'].count().reset_index()
        filter_movie = type_movie_1[(type_movie_1['Release_Year'] >=
slider_year)]

        type_tvshow   =   netflix_df[(netflix_df['type']   ==   'TV
Show')][['Type', 'Release_Year']]
        type_tvshow['type2'] = type_tvshow['Type']
        type_tvshow_1     =     type_tvshow.groupby(['Release_Year',
'type2'])['Type'].count().reset_index()
        filter_tvshow = type_tvshow_1[(type_tvshow_1['Release_Year']
>= slider_year)]

        return {
            'data': [go.Scatter(
                x=filter_movie['Release_Year'],
                y=filter_movie['Type'],
                mode='markers+lines',
                name='Movie',
                line=dict(shape="spline",   smoothing=1.3,   width=3,
color='#b20710'),
                marker=dict(size=10,                symbol='circle',
color='#f5f5f1',
                            line=dict(color='blue', width=2)
                            ),

                hoverinfo='text',
                hovertext=
                '<b>Release          Year</b>:          '         +
filter_movie['Release_Year'].astype(str) + '<br>' +
                '<b>Type</b>: ' + filter_movie['type1'].astype(str) +
'<br>' +
                '<b>Count</b>:   '   +   [f'{x:,.0f}'   for   x   in
filter_movie['Type']] + '<br>'
            ),

                go.Scatter(
                    x=filter_tvshow['Release_Year'],
                    y=filter_tvshow['Type'],
                    mode='markers+lines',
                    name='TV Show',
                    line=dict(shape="spline",          smoothing=1.3,
width=3,color='#221f1f' ),
                    marker=dict(size=10,
symbol='circle',color='#f5f5f1',
                                line=dict(color='blue',width=2)
                                ),

                    hoverinfo='text',
                    hovertext=
                    '<b>Release         Year</b>:          '         +
filter_tvshow['Release_Year'].astype(str) + '<br>' +
                    '<b>Type</b>:                '                +
filter_tvshow['type2'].astype(str) + '<br>' +
                    '<b>Count</b>:   '   +   [f'{x:,.0f}'   for   x   in
filter_tvshow['Type']] + '<br>'

                )],
            'layout': go.Layout(
                title={
                    'text': 'Movies and TV Shows by Release Year',
                    'xanchor': 'right',
                    'yanchor': 'top'},
                width=1200
            ),

        }
    return app1
```

2. dash_application.py

```
from http import server
from unicodedata import name
import dash
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objs as go
from dash import dcc
from dash import html
from dash.dependencies import Input, Output
from textblob import TextBlob
import plotly.io as plt_io



def clean_netflix_df(df):
    df['country'] = df['country'].fillna(df['country'].mode()[0])
    df['cast'].replace(np.nan, 'No Data', inplace=True)
    df['director'].replace(np.nan, 'No Data', inplace=True)
    df.dropna(inplace=True)

    df.drop_duplicates(inplace=True)

    df["date_added"] = pd.to_datetime(df['date_added'])
    df['month_added'] = df['date_added'].dt.month
    df['month_name_added'] = df['date_added'].dt.month_name()
    df['year_added'] = df['date_added'].dt.year

    df['first_country']      =      df['country'].apply(lambda      x:
x.split(",")[0])
    df['first_country'].replace('United States', 'USA', inplace=True)
    df['first_country'].replace('United Kingdom', 'UK', inplace=True)
    df['first_country'].replace('South     Korea',     'S.     Korea',
inplace=True)

    df['count'] = 1
    df['genre'] = df['listed_in'].apply(lambda x: x.replace(' ,',
',').replace(', ', ',').split(','))
    return df

def fig_bar_horiz(netflix_df):
    country_order =
netflix_df['first_country'].value_counts()[:11].index
    data_q2q3 = netflix_df[['type',
'first_country']].groupby('first_country')['type'].value_counts().un
stack().loc[
        country_order]
    data_q2q3['sum'] = data_q2q3.sum(axis=1)
    data_q2q3_ratio = (data_q2q3.T / data_q2q3['sum']).T[['Movie',
'TV Show']].sort_values(by='Movie', ascending=False)[
                      ::-1]
    fig_bar = go.Figure()
    fig_bar.add_trace(go.Bar(
        y=data_q2q3_ratio.index,
        x=round(data_q2q3_ratio['Movie'] * 100, 2),
        name='Movies',
        orientation='h',

        marker=dict(
            color='#34495E',
            line=dict( width=3),
        )
    ))
    fig_bar.add_trace(go.Bar(
        y=data_q2q3_ratio.index,
        x=round(data_q2q3_ratio['TV Show'] * 100, 2),
        name='TV Shows',
        orientation='h',
        marker=dict(
            color='#45B39D',
            line=dict(width=3)
        )
    ))

    fig_bar.update_layout(barmode='stack',
                          title={
                              'text': "Top 10 countries Movie & TV
Show split",
                              'y': 0.9,
                              'x': 0.5,
                              'xanchor': 'center',
                              'yanchor': 'top'}
                          )
    return fig_bar

def fig_bar_stacked(netflix_df):
    order                                                          =
pd.DataFrame(netflix_df.groupby('rating')['count'].sum().sort_values
(ascending=False).reset_index())
    rating_order = list(order['rating'])
    mf                                                             =
netflix_df.groupby('type')['rating'].value_counts().unstack().sort_i
ndex().fillna(0).astype(int)[rating_order]

    movie = mf.loc['Movie']
    tv = - mf.loc['TV Show']

    fig_stacked = go.Figure()
    fig_stacked.add_trace(go.Bar(x=movie.index,             y=movie,
name='Movies',marker_color='#F1C40F'))
    fig_stacked.add_trace(go.Bar(x=tv.index,                   y=tv,
name='TVShows',marker_color='#D35400'))
    fig_stacked.update_layout(barmode='relative',
                              title={
                                      'text': 'Rating distribution by
Movie & TV Show',
                                       'y': 0.9,
                                       'x': 0.5,
                                       'xanchor': 'center',
                                       'yanchor': 'top'}
                             )
    return fig_stacked

def fig_stack_without_flying(netflix_df):
    dfx = netflix_df[['release_year', 'description']]
    dfx = dfx.rename(columns={'release_year': 'Release Year'})
    for index, row in dfx.iterrows():
        z = row['description']
        testimonial = TextBlob(z)
        p = testimonial.sentiment.polarity
        if p == 0:
            sent = 'Normal'
        elif p > 0:
            sent = 'Positive'
        else:
            sent = 'Negative'
        dfx.loc[[index, 2], 'Sentiment'] = sent

    dfx            =           dfx.groupby(['Release           Year',
'Sentiment']).size().reset_index(name='Total Content')

    dfx = dfx[dfx['Release Year'] >= 2010]
    fig_stacked_without_fly = px.bar(dfx, x="Release Year", y="Total
Content",color='Sentiment')
    fig_stacked_without_fly.update_layout(title={
        'text':  'Sentiment  Analysis  over  years  for  Movies  and  Tv
Shows',
        'y': 0.9,
        'x': 0.5,
        'xanchor': 'center',
        'yanchor': 'top'})
    return fig_stacked_without_fly

def fig_pie_purst(netflix_df):
    fig_purst  =  px.sunburst(netflix_df[netflix_df['year_added']  >=
2018], path=['year_added', 'month_name_added'],
                             values='count',
color_continuous_scale='armyrose')
    fig_purst.update_layout(title={
        'text': 'Number of Movies and Tv shows added per month last 5
year',
        'y': 0.9,
        'x': 0.5,
        'xanchor': 'center',
        'yanchor': 'top'})
    return fig_purst
```

```
def world_map(netflix_df):
    df_country_year =
netflix_df.groupby(by=['country','type','rating']).count().reset_ind
ex()

df_country_year['total']=df_country_year.groupby(by=['country'])['ti
tle'].cumsum()


fig_world_map=px.choropleth(df_country_year.sort_values(by='rating')
, locations='country', title='Country wise statistics of
Ratings',color='total', locationmode='country names',
animation_frame='rating', range_color=[0,1000],)
    return fig_world_map

def create_dash_application(flask_app):
    app1 = dash.Dash(server=flask_app,name="Netflix Dashboard",
url_base_pathname="/dash/")
    netflix_df = pd.read_csv("D:/flask1/netflix_titles (1).csv")
    netflix_df = clean_netflix_df(netflix_df)
    app1.layout = html.Div(children=[

        html.H1(id='header', children=[html.Div("Netflix Title
Analysis", id='header-text')],
                style={'textAlign': 'center', 'color': '#b20710'},
className="mb-3"),
        html.Br(),
        html.Br(),
        html.Br(),
        html.Div(children=[
            html.Div(children=[
                html.P('Filter by Release Year',
className='fix_label', style={'text-align': 'center', 'color':
'#221f1f'}),
                dcc.Slider(id='slider_year',
                        included=True,
                        updatemode='drag',
                        tooltip={'always_visible': True},
                        min=1925,
                        max=2020,
                        step=1,
                        value=2009,
                        marks={str(yr): str(yr) for yr in range(1925,
2020, 10)}, className="row"
                        ),
                html.Div([
                    html.Div([
                        dcc.Graph(id='line_chart',
config={'displayModeBar':                'hover'}),                ],
className="row",style={'width': '100%'}),
                ]),
            ], className="col-md-6"),
            html.Div(children=[
                html.Div([
                    dcc.Graph(id='bar_fig',
figure=fig_bar_horiz(netflix_df))], id='FigBarGraphDiv')
            ], className="col-md-6"),

        ],className="row"),
        html.Div(children=[
            html.Div(children=[
                html.Div([
                    dcc.Graph(id='stack_fig',
figure=fig_bar_stacked(netflix_df))], id='StackedGraphDiv')

            ], className="col-md-6"),
            html.Div(children=[
                html.Div([
                    dcc.Graph(id='SentBarGraphDiv',
figure=fig_stack_without_flying(netflix_df)), ])
            ], className="col-md-6"),

        ], className="col-md-6"),
        html.Div(children=[
            html.Div(children=[
                html.Div([
                dcc.Graph(id='world_fig',
figure=world_map(netflix_df))], id='World_Fig')
        ], className="col-md-6"),
        ], className="row"),

    html.Div(children=[
        html.Div(children=[
            html.Div([
                dcc.Graph(id='purst_fig',
figure=fig_pie_purst(netflix_df))],
                id='PurstGraphDiv')



        ], className="col-md-6"),
        html.Div(children=[
            html.Label('Movie Statistics Calculator',
id='calculator'),
            html.Div([
                dcc.Dropdown(id='dropDown', options=[{'label': x,
'value': x} for x in netflix_df['first_country'].unique()],
                             value='Egypt'),
                html.Br(),
                html.Br(),
                html.Table([
                    html.Tbody([
                        html.Tr([
                            html.Td("No. of Movies till date"),

                            html.Td([
                                html.Div(
                                    id="val1"

                                )

                            ])
                        ]),
                        html.Tr([
                            html.Td("No. of TV Shows till date"),

                            html.Td([
                                html.Div(
                                    id="val2"

                                )

                            ])
                        ]),
                        html.Tr([
                            html.Td("Top Actor"),

                            html.Td([
                                html.Div(
                                    id="val3"

                                )

                            ])
                        ]),
                        html.Tr([
                            html.Td("Top Director"),

                            html.Td([
                                html.Div(
                                    id="val4"

                                )

                            ])
                        ])

                    ])
                ], className="table table-striped")
            ])
        ], className="col-md-6"),

    ], className="row"),

], className="container-fluid",)


    @app1.callback(
    [Output('val1', 'children'), Output('val2', 'children'),
Output('val3', 'children'), Output('val4', 'children')],
    Input('dropDown', 'value'))
    def updateTable(dropDown):
        dfx = netflix_df[['type', 'country']]
        dfMovie = dfx[dfx['type'] == 'Movie']
        dfTV = dfx[dfx['type'] == 'TV Show']
        dfM1 = dfMovie['country'].str.split(',', expand=True).stack()
        dfTV1 = dfTV['country'].str.split(',', expand=True).stack()
        dfM1 = dfM1.to_frame()
        dfTV1 = dfTV1.to_frame()
        dfM1.columns = ['country']
        dfTV1.columns = ['country']
        dfM2                                                              =
dfM1.groupby(['country']).size().reset_index(name='counts')
        dfTV2                                                             =
dfTV1.groupby(['country']).size().reset_index(name='counts')
        dfM2['country'] = dfM2['country'].str.strip()
        dfTV2['country'] = dfTV2['country'].str.strip()
        val11 = dfM2[dfM2['country'] == dropDown]
        val22 = dfTV2[dfTV2['country'] == dropDown]
        val11 = val11.reset_index()
        val22 = val22.reset_index()

        if val11.empty:
            val1 = 0
        else:
            val1 = val11.loc[0]['counts']

        if val22.empty:
            val2 = 0
        else:
            val2 = val22.loc[0]['counts']

    # Top Actor
        dfA = netflix_df[['cast', 'country']]
        dfA1 = dfA[dfA['country'].str.contains(dropDown, case=False)]
        dfA2 = dfA1['cast'].str.split(',', expand=True).stack()
        dfA2 = dfA2.to_frame()
        dfA2.columns = ['Cast']
        dfA3 =
dfA2.groupby(['Cast']).size().reset_index(name='counts')
        dfA3 = dfA3[dfA3['Cast'] != 'No Cast Specified']
        dfA3 = dfA3.sort_values(by='counts', ascending=False)
        if dfA3.empty:
            val3 = "Actor data from this country is not available"
        else:
            val3 = dfA3.iloc[0]['Cast']
        # Top Director
        dfD = netflix_df[['director', 'country']]
        dfD1 = dfD[dfD['country'].str.contains(dropDown, case=False)]
        dfD2 = dfD1['director'].str.split(',', expand=True).stack()
        dfD2 = dfD2.to_frame()
        dfD2.columns = ['Director']
        dfD3 =
dfD2.groupby(['Director']).size().reset_index(name='counts')
        dfD3 = dfD3[dfD3['Director'] != 'No Director Specified']
        dfD3 = dfD3.sort_values(by='counts', ascending=False)
        if dfD3.empty:
            val4 = "Director data from this country is not available"
        else:
            val4 = dfD3.iloc[0]['Director']
        return val1, val2, val3, val4

    @app1.callback(Output('line_chart', 'figure'),
              [Input('slider_year', 'value')])
    def update_graph(slider_year):
        type_movie = netflix_df[(netflix_df['type'] ==
'Movie')][['type', 'release_year']]
        type_movie['type1'] = type_movie['type']
        type_movie_1 = type_movie.groupby(['release_year',
'type1'])['type'].count().reset_index()
        filter_movie = type_movie_1[(type_movie_1['release_year'] >=
slider_year)]

        type_tvshow = netflix_df[(netflix_df['type'] == 'TV
Show')][['type', 'release_year']]
        type_tvshow['type2'] = type_tvshow['type']
        type_tvshow_1 = type_tvshow.groupby(['release_year',
'type2'])['type'].count().reset_index()
```

```
        filter_tvshow = type_tvshow_1[(type_tvshow_1['release_year']
>= slider_year)]

        return {
            'data': [go.Scatter(
                x=filter_movie['release_year'],
                y=filter_movie['type'],
                mode='markers+lines',
                name='Movie',
                line=dict(shape="spline", smoothing=1.3, width=3,
color='#b20710'),
                marker=dict(size=10, symbol='circle',
color='#f5f5f1',line=dict(color='blue', width=2)),
                hoverinfo='text',
                hovertext='<b>Release Year</b>: ' +
filter_movie['release_year'].astype(str) + '<br>' +'<b>Type</b>: ' +
filter_movie['type1'].astype(str) + '<br>' +'<b>Count</b>: ' +
[f'{x:,.0f}' for x in filter_movie['type']] + '<br>'),

                go.Scatter(
                    x=filter_tvshow['release_year'],
                    y=filter_tvshow['type'],
                    mode='markers+lines',
                    name='TV Show',
                    line=dict(shape="spline", smoothing=1.3,
width=3,color='#221f1f' ),
                    marker=dict(size=10,
symbol='circle',color='#f5f5f1',line=dict(color='blue',width=2)),
                    hoverinfo='text',
                    hovertext='<b>Release Year</b>: ' +
filter_tvshow['release_year'].astype(str) + '<br>' +'<b>Type</b>: ' +
filter_tvshow['type2'].astype(str) + '<br>' +'<b>Count</b>: ' +
[f'{x:,.0f}' for x in filter_tvshow['type']] + '<br>')],
            'layout': go.Layout(
                title={
                    'text': 'Movies and TV Shows by Release Year',
                    'xanchor': 'right',
                    'yanchor': 'top'},
                width=1200),
        }
    return app1
```

3. app.py

```
from flask import Flask, render_template, request, redirect
import os
import csv
from sqlite3 import Error
import pandas as pd
from dash_application import create_dash_application
from dashPersonal_application import create
import sqlite3 as sql
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)

app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:///naman1.db"
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
create_dash_application(app)
create(app)

class Netflix(db.Model):
    sno = db.Column(db.Integer, primary_key=True)
    Show_ID = db.Column(db.String(500),nullable =False)
    Type = db.Column(db.String(500),nullable =False)
    Title = db.Column(db.String(500),nullable =False)
    Director = db.Column(db.String(500),nullable =False)
    Cast  = db.Column(db.String(500),nullable =False)
    Country = db.Column(db.String(500),nullable =False)
    Date_Added = db.Column(db.String(500),nullable =False)
    Release_Year = db.Column(db.String(500),nullable =False)
    Rating = db.Column(db.String(500),nullable =False)
    Duration = db.Column(db.String(500),nullable =False)
    Listed_In = db.Column(db.String(500),nullable =False)
    Description = db.Column(db.String(500),nullable =False)

    def __repr__(self) -> str:
        return f"{self.Type} - {self.Title}"

@app.route('/', methods= ['GET', 'POST'])
def hello_world():
    if request.method == 'POST':
        a = request.form['Show_ID']
        b = request.form['Type']
        c = request.form['Title']
        d = request.form['Director']
        e = request.form['Cast']
        f = request.form['Country']
        g = request.form['Date_Added']
        h = request.form['Release_Year']
        i = request.form['Rating']
        j = request.form['Duration']
        k = request.form['Listed_In']
        l = request.form['Description']
        nam = Netflix(Show_ID = a, Type=b,Title = c, Director= d,
Cast= e, Country=f,Date_Added = g, Release_Year=h,Rating=i,Duration=
j,Listed_In = k,Description=l)
        db.session.add(nam)
        db.session.commit()


    allnet = Netflix.query.all()
    return render_template('index.html',allnet = allnet)


@app.route('/update/<int:sno>',methods = ['GET', 'POST'])
def update(sno):
    if request.method == 'POST':
        a = request.form['Show_ID']
        b = request.form['Type']
        c = request.form['Title']
        d = request.form['Director']
        e = request.form['Cast']
        f = request.form['Country']
        g = request.form['Date_Added']
        h = request.form['Release_Year']
        i = request.form['Rating']
        j = request.form['Duration']
        k = request.form['Listed_In']
        l = request.form['Description']
        naman = Netflix.query.filter_by(sno=sno).first()
        naman.Show_ID = a
        naman.Type = b
        naman.Title_ID = c
        naman.Director = d
        naman.Cast = e
        naman.Country = f
        naman.Date_Added = g
        naman.Release_Year = h
        naman.Rating = i
        naman.Duration = j

        naman.Listed_In = k
        naman.Description = l
        db.session.add(naman)
        db.session.commit()
        return redirect("/")

    naman = Netflix.query.filter_by(sno=sno).first()
    return render_template('update.html',naman = naman)

@app.route('/delete/<int:sno>')
def delete(sno):
    naman = Netflix.query.filter_by(sno=sno).first()
    db.session.delete(naman)
    db.session.commit()
    return redirect("/")

if __name__ =='__main__':
    app.run(debug=True, port=8000)
```

4. base.html

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstra
p.min.css" rel="stylesheet"
        integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

    <title>{% block title %} {% endblock title %} Netflix
Titles!</title>
</head>

<body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">
            <a class="navbar-brand" href="/">NETFLIX TITLES</a>
            <button class="navbar-toggler" type="button" data-bs-
toggle="collapse"
                data-bs-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false"
                aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse"
id="navbarSupportedContent">
                <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                    <li class="nav-item">
                        <a class="nav-link active" aria-
current="page" href="/">Home</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
    {% block body %}

    {% endblock body %}

</body>

</html>
```

5. index.html

```
{% extends 'base.html' %}
{% block title %} Home {% endblock title %}
{% block body %}


    <div class="container my-5">
        <h2>Add the Neflix Movie Title Data:</h2>
        <form action="/" method="POST">
            <div class="mb-3">
                <label      for="Show_ID"      class="form-label">Show
ID</label>
                <input        type="text"        class="form-control"
name="Show_ID" id="Show_ID" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Type" class="form-label">Type</label>
                <input  type="text"  class="form-control"  name="Type"
id="Type" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Title" class="form-label">Title</label>
                <input type="text"  class="form-control"  name="Title"
id="Title" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label          for="Director"          class="form-
label">Director</label>
                <input        type="text"        class="form-control"
name="Director" id="Director">
```

```
            </div>
            <div class="mb-3">
                <label for="Cast" class="form-label">Cast</label>
                <input type="text" class="form-control" name="Cast"
id="Cast" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Country" class="form-
label">Country</label>
                <input type="text" class="form-control"
name="Country" id="Country" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Date_Added" class="form-label">Date
Added</label>
                <input type="text" class="form-control"
name="Date_Added" id="Date_Added" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Release_Year" class="form-label">Release
Year</label>
                <input type="text" class="form-control"
name="Release_Year" id="Release_Year"
                    aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Rating" class="form-label">Rating</label>
                <input type="text" class="form-control" name="Rating"
id="Rating" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Duration" class="form-
label">Duration</label>
                <input type="text" class="form-control"
name="Duration" id="Duration" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Listed_In" class="form-label">Listed
In</label>
```

```
                <input type="text" class="form-control"
name="Listed_In" id="Listed_In" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Description" class="form-
label">Description</label>
                <input type="text" class="form-control"
name="Description" id="Description"
                    aria-describedby="emailHelp">
            </div>

            <button type="submit" class="btn btn-
dark">Submit</button>
        </form>
    </div>
    <div class="container my-5">
        <h2>Netflix Titles</h2>
            {% if allnet|length == 0 %}
                <div class="alert alert-dark" role="alert">
                    No Todos found. Add your first todo now!
                </div>
            {% else %}
            <table class="table">
                <thead>
                    <tr>
                        <th scope="col">SNo</th>
                        <th scope="col">Show ID</th>
                        <th scope="col">Type</th>
                        <th scope="col">Title</th>
                        <th scope="col">Director</th>
                        <th scope="col">Cast</th>
                        <th scope="col">Country</th>
                        <th scope="col">Date Added</th>
                        <th scope="col">Release Year</th>
                        <th scope="col">Rating</th>
                        <th scope="col">Duration</th>
                        <th scope="col">Listed In</th>
                        <th scope="col">Description</th>
                        <th scope="col">Action</th>
                    </tr>
                </thead>
                <tbody>
                    {% for nam in allnet %}
                    <tr>
                        <th scope="row">{{loop.index}}</th>
                        <td>{{nam.Show_ID}}</td>
                        <td>{{nam.Type}}</td>
                        <td>{{nam.Title}}</td>
                        <td>{{nam.Director}}</td>
                        <td>{{nam.Cast}}</td>
                        <td>{{nam.Country}}</td>
                        <td>{{nam.Date_Added}}</td>
                        <td>{{nam.Release_Year}}</td>
                        <td>{{nam.Rating}}</td>
                        <td>{{nam.Duration}}</td>
                        <td>{{nam.Listed_In}}</td>
                        <td>{{nam.Description}}</td>
                        <td><a             href="/update/{{nam.sno}}"
type="button"
                                class="btn  btn-outline-info  btn-sm
mx-2 my-2">Update</button>
                                <a         href="/delete/{{nam.sno}}"
type="button"
                                    class="btn  btn-dark  btn-sm  mx-2
my-2">Delete</button>
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
            {% endif %}

        </div>
        <div>
            <a href="/dash" style="margin-left: 100px;" type="button"
class="btn btn-outline-info btn-sm mx-2">DASHBOARD</button>
            <a     href="/personal"     style="margin-left:     100px;"
type="button"    class="btn    btn-outline-info    btn-sm    mx-2    my-
2">Personalise Dashboard</button>
        </div>
```

```
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.
bundle.min.js"
        integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
        crossorigin="anonymous"></script>

 {% endblock body %}
```

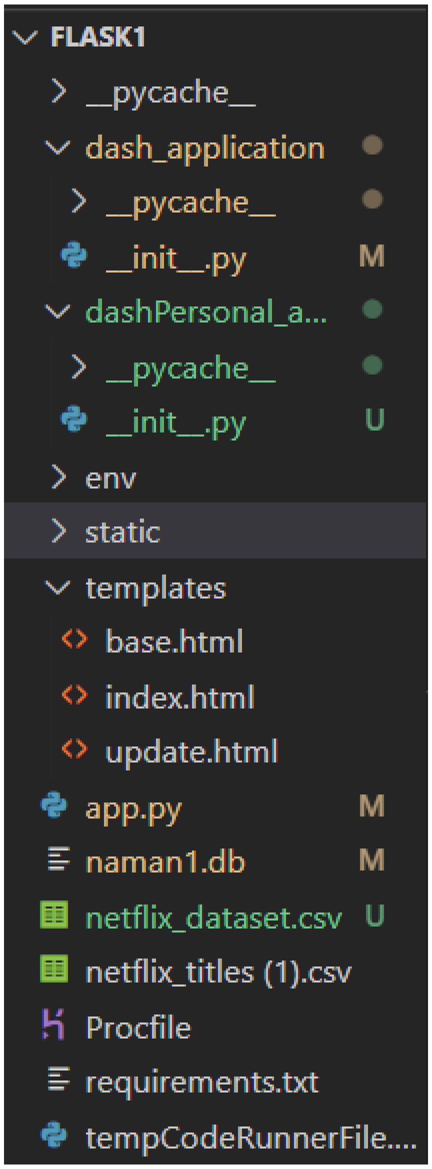
6. update.html

```
      {% extends 'base.html' %}
{% block title %} Home{% endblock title %}
{% block body %}

    <div class="container my-3">
        <h2>Update Netflix Title</h2>
        <form action="/update/{{naman.sno}}" method="POST">
            <div class="mb-3">
              <label for="Show_ID" class="form-label">Show ID</label>
                <input  type="text"  class="form-control"  name="Show_ID"
id="Show_ID" value="{{naman.Show_ID}}" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Type" class="form-label">Type</label>
                <input   type="text"   class="form-control"   name="Type"
id="Type" value="{{naman.Type}}" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Title" class="form-label">Title</label>
                <input   type="text"   class="form-control"   name="Title"
id="Title" value="{{naman.Title}}" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Director" class="form-label">Director</label>
                <input          type="text"          class="form-control"
name="Director"value= "{{naman.Director}}" id="Director">
            </div>
```

```html
            <div class="mb-3">
                <label for="Cast" class="form-label">Cast</label>
                <input   type="text"   class="form-control"   name="Cast"
id="Cast" value="{{naman.Cast}}" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Country" class="form-label">Country</label>
                <input  type="text"  class="form-control"  name="Country"
id="Country" value="{{naman.Country}}" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label      for="Date_Added"      class="form-label">Date
Added</label>
                <input  type="text"  class="form-control"  name="Date_Added"
id="Date_Added" value="{{naman.Date_Added}}" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label    for="Release_Year"    class="form-label">Release
Year</label>
                <input type="text" class="form-control" name="Release_Year"
id="Release_Year"           value="{{naman.Release_Year}}"           aria-
describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Rating" class="form-label">Rating</label>
                <input   type="text"   class="form-control"   name="Rating"
id="Rating" value="{{naman.Rating}}" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Duration" class="form-label">Duration</label>
                <input  type="text"  class="form-control"  name="Duration"
id="Duration" value="{{naman.Duration}}" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for="Listed_In" class="form-label">Listed In</label>
                <input  type="text"  class="form-control"  name="Listed_In"
id="Listed_In" value="{{naman.Listed_In}}" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
```

```
                <label              for="Description"             class="form-
label">Description</label>
                <input type="text" class="form-control" name="Description"
id="Description" value="{{naman.Description}}"
                    aria-describedby="emailHelp">
            </div>

            <button type="submit" class="btn btn-dark">Update</button>
          </form>
    </div>

    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script             src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta2/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
b5kHyXgcpbZJO/tY9Ul7kGkf1S0CWuKcCD38l8YkeH8z8QjE0GmW1gYU5S9FOnJ0"
        crossorigin="anonymous"></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.6.0/dist/umd/popper.min
.js"                                                    integrity="sha384-
KsvD1yqQ1/1+IA7gi3P0tyJcT3vR+NdBTt13hSJ2lnve8agRGXTTyNaBYmCR/Nwi"
crossorigin="anonymous"></script>
    <script             src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta2/dist/js/bootstrap.min.js"                         integrity="sha384-
nsg8ua9HAw1y0W1btsyWgBklPnCUAFLuTMS2G72MMONqmOymq585AcH49TLBQObG"
crossorigin="anonymous"></script>
    -->

{% endblock body %}
```

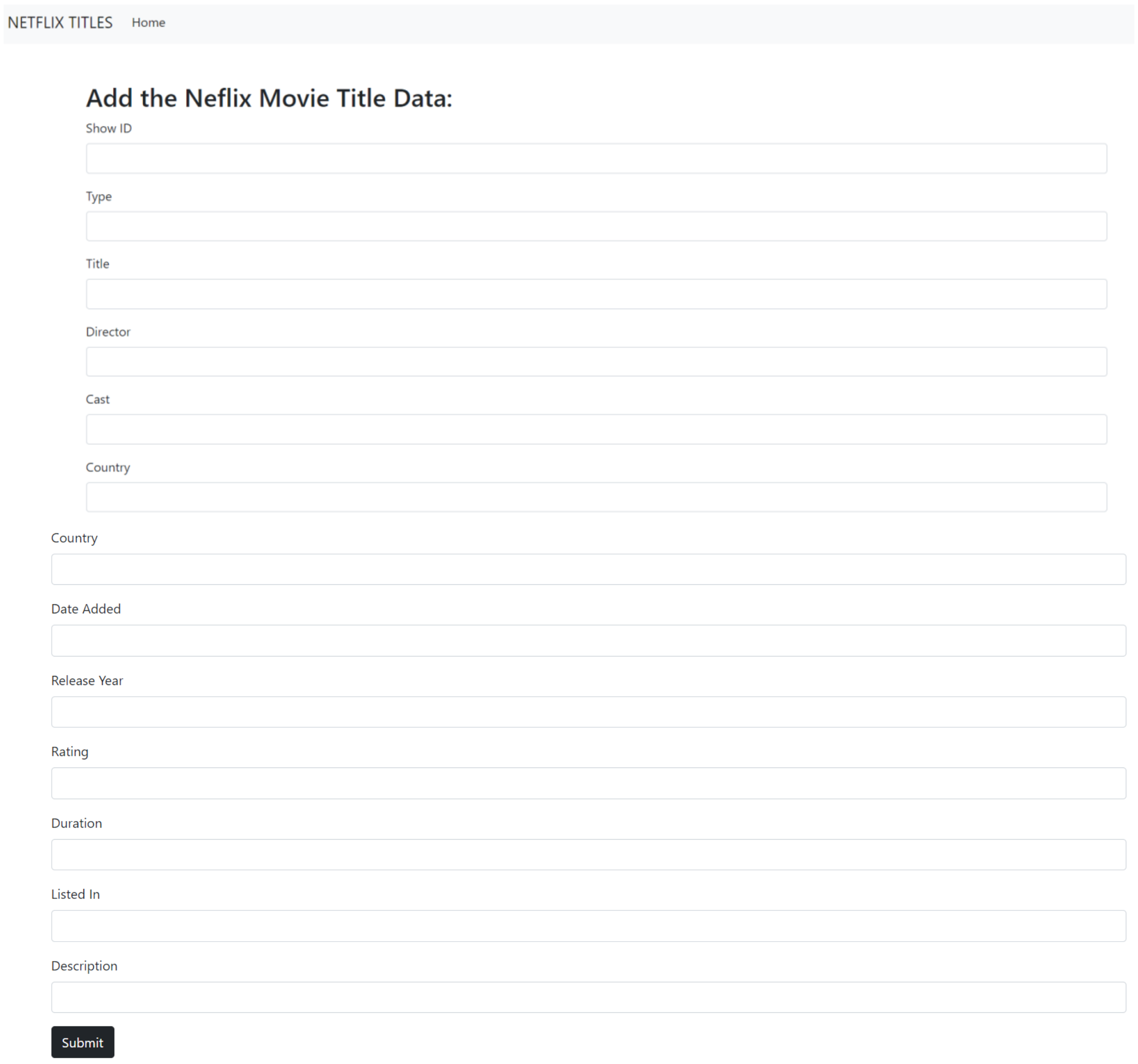
7. File Directory



```
FLASK1
  > __pycache__
  v dash_application        ●
    > __pycache__           ●
      __init__.py           M
  v dashPersonal_a...       ●
    > __pycache__           ●
      __init__.py           U
  > env
  > static
  v templates
      base.html
      index.html
      update.html
    app.py                  M
    naman1.db               M
    netflix_dataset.csv     U
    netflix_titles (1).csv
    Procfile
    requirements.txt
    tempCodeRunnerFile....
```

IX. RESULT

Website:


NETFLIX TITLES Home

Add the Neflix Movie Title Data:

Show ID

Type

Title

Director

Cast

Country

Country

Date Added

Release Year

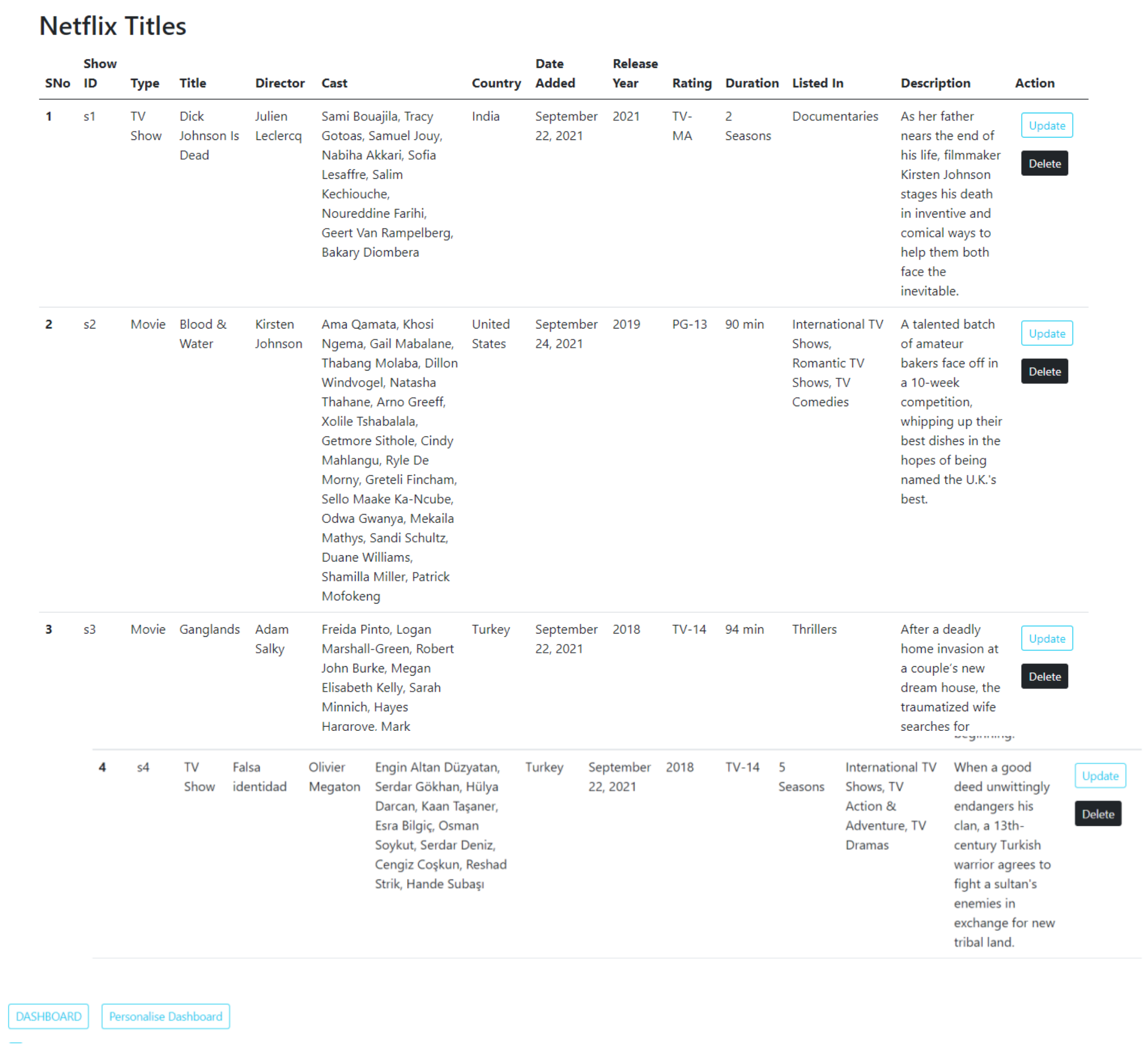
Rating

Duration

Listed In

Description

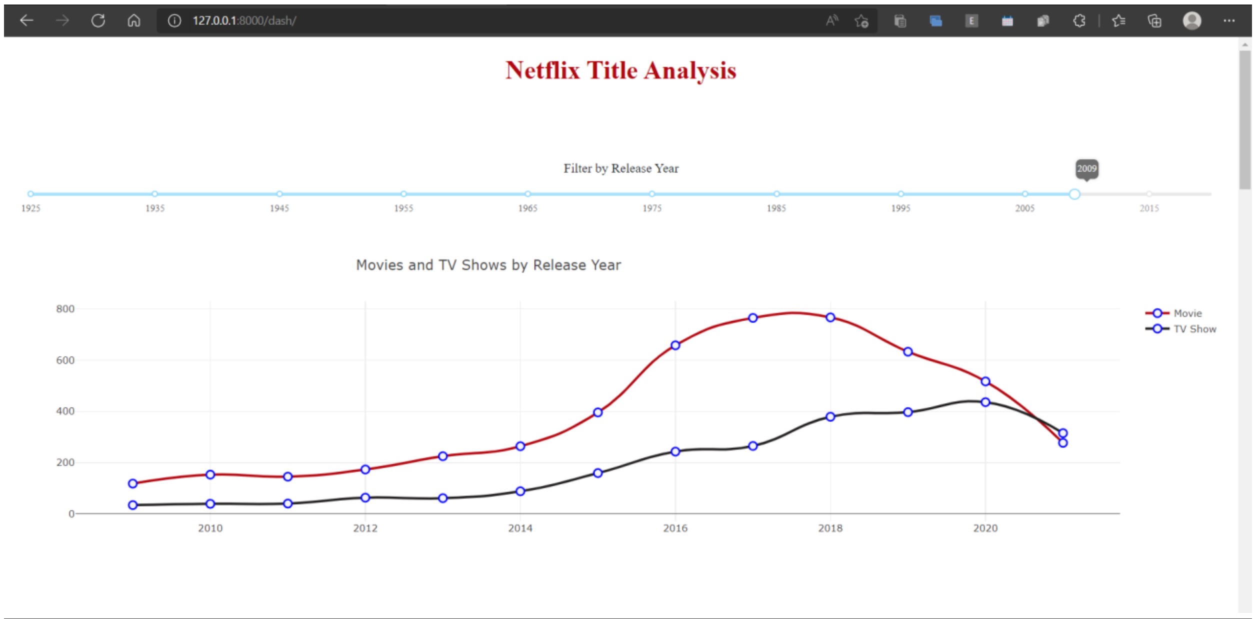
Submit

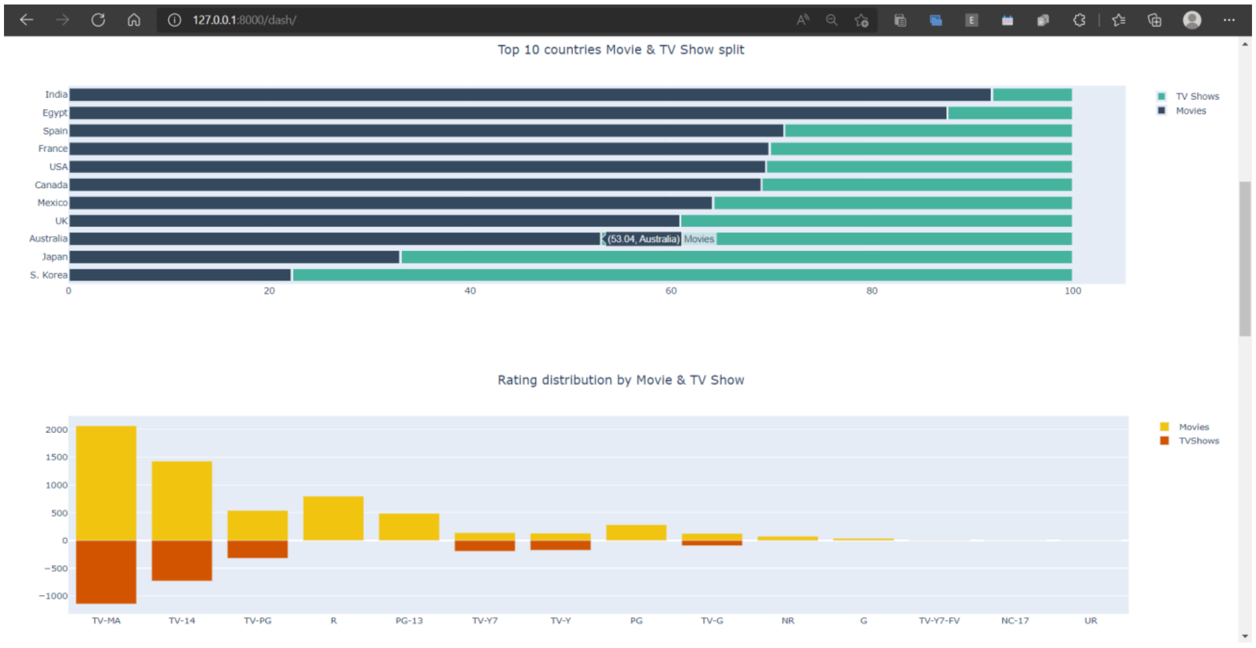


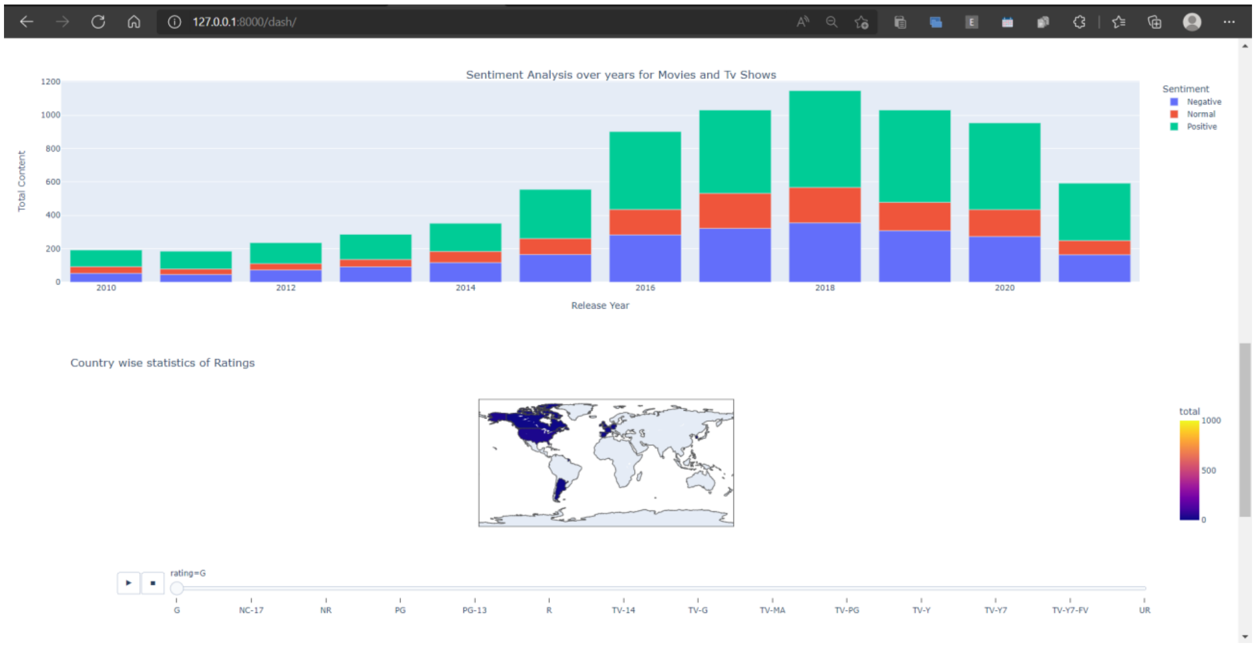
Netflix Titles

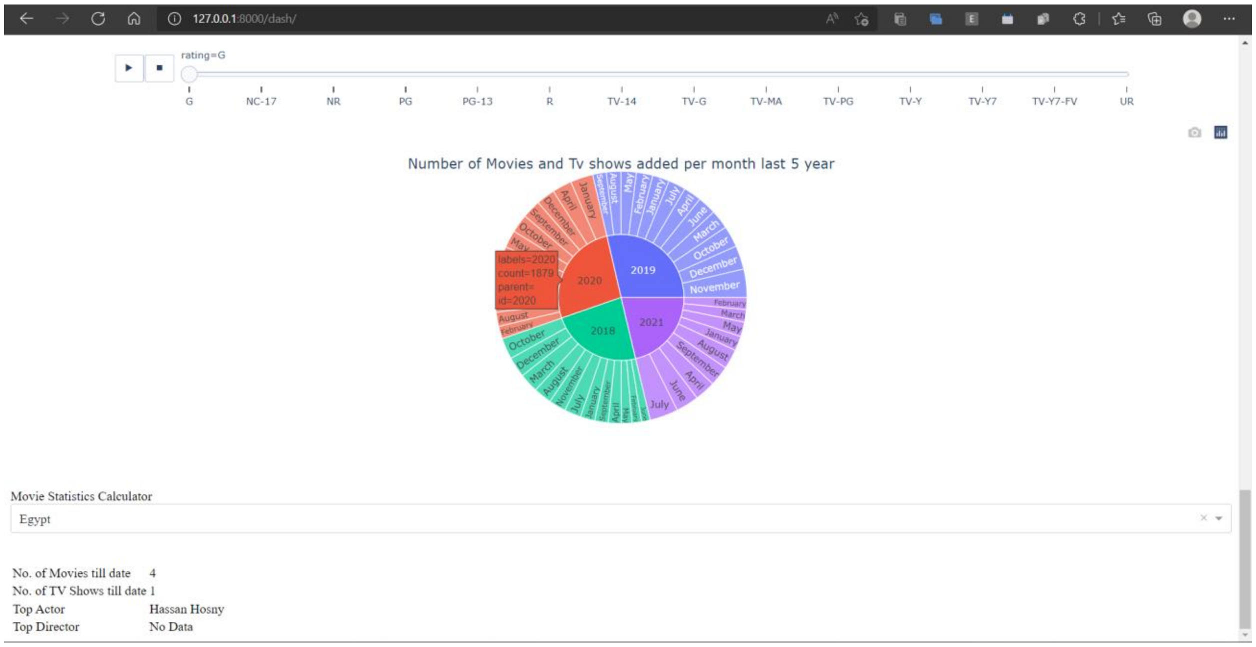
SNo	Show ID	Type	Title	Director	Cast	Country	Date Added	Release Year	Rating	Duration	Listed In	Description	Action
1	s1	TV Show	Dick Johnson Is Dead	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabiha Akkari, Sofia Lesaffre, Salim Kechiouche, Noureddine Farihi, Geert Van Rampelberg, Bakary Diombera	India	September 22, 2021	2021	TV-MA	2 Seasons	Documentaries	As her father nears the end of his life, filmmaker Kirsten Johnson stages his death in inventive and comical ways to help them both face the inevitable.	Update Delete
2	s2	Movie	Blood & Water	Kirsten Johnson	Ama Qamata, Khosi Ngema, Gail Mabalane, Thabang Molaba, Dillon Windvogel, Natasha Thahane, Arno Greeff, Xolile Tshabalala, Getmore Sithole, Cindy Mahlangu, Ryle De Morny, Greteli Fincham, Sello Maake Ka-Ncube, Odwa Gwanya, Mekaila Mathys, Sandi Schultz, Duane Williams, Shamilla Miller, Patrick Mofokeng	United States	September 24, 2021	2019	PG-13	90 min	International TV Shows, Romantic TV Shows, TV Comedies	A talented batch of amateur bakers face off in a 10-week competition, whipping up their best dishes in the hopes of being named the U.K.'s best.	Update Delete
3	s3	Movie	Ganglands	Adam Salky	Freida Pinto, Logan Marshall-Green, Robert John Burke, Megan Elisabeth Kelly, Sarah Minnich, Hayes Harorove. Mark	Turkey	September 22, 2021	2018	TV-14	94 min	Thrillers	After a deadly home invasion at a couple's new dream house, the traumatized wife searches for	Update Delete
4	s4	TV Show	Falsa identidad	Olivier Megaton	Engin Altan Düzyatan, Serdar Gökhan, Hülya Darcan, Kaan Taşaner, Esra Bilgiç, Osman Soykut, Serdar Deniz, Cengiz Coşkun, Reshad Strik, Hande Subaşı	Turkey	September 22, 2021	2018	TV-14	5 Seasons	International TV Shows, TV Action & Adventure, TV Dramas	When a good deed unwittingly endangers his clan, a 13th-century Turkish warrior agrees to fight a sultan's enemies in exchange for new tribal land.	Update Delete

Dashboard Output:







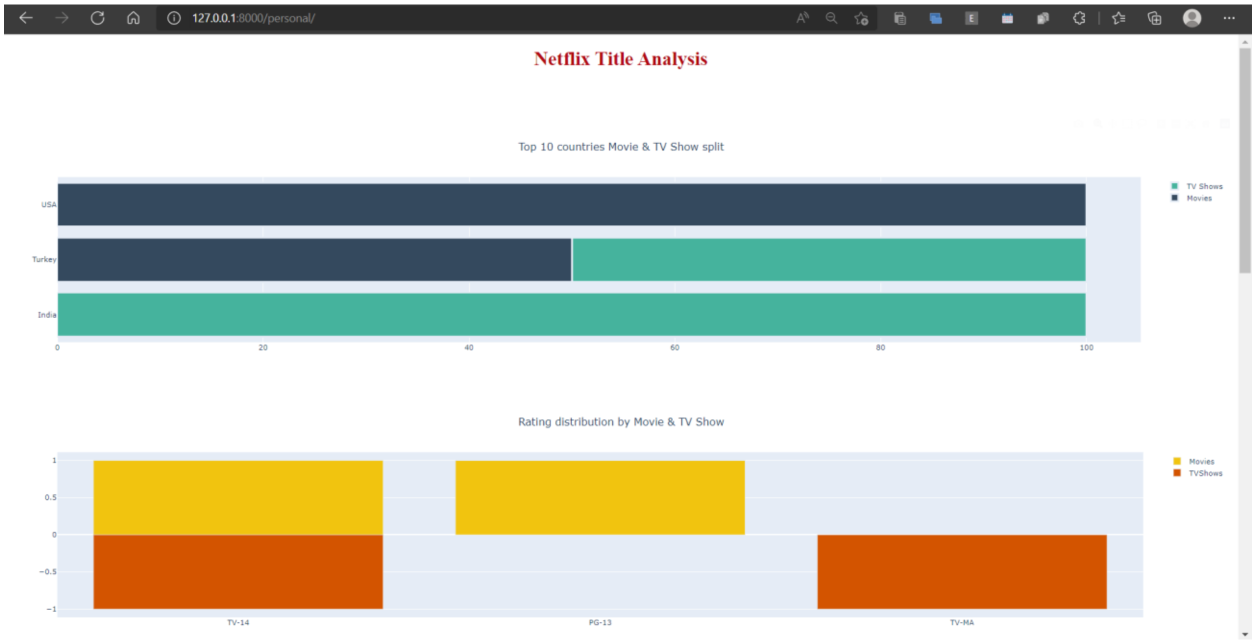


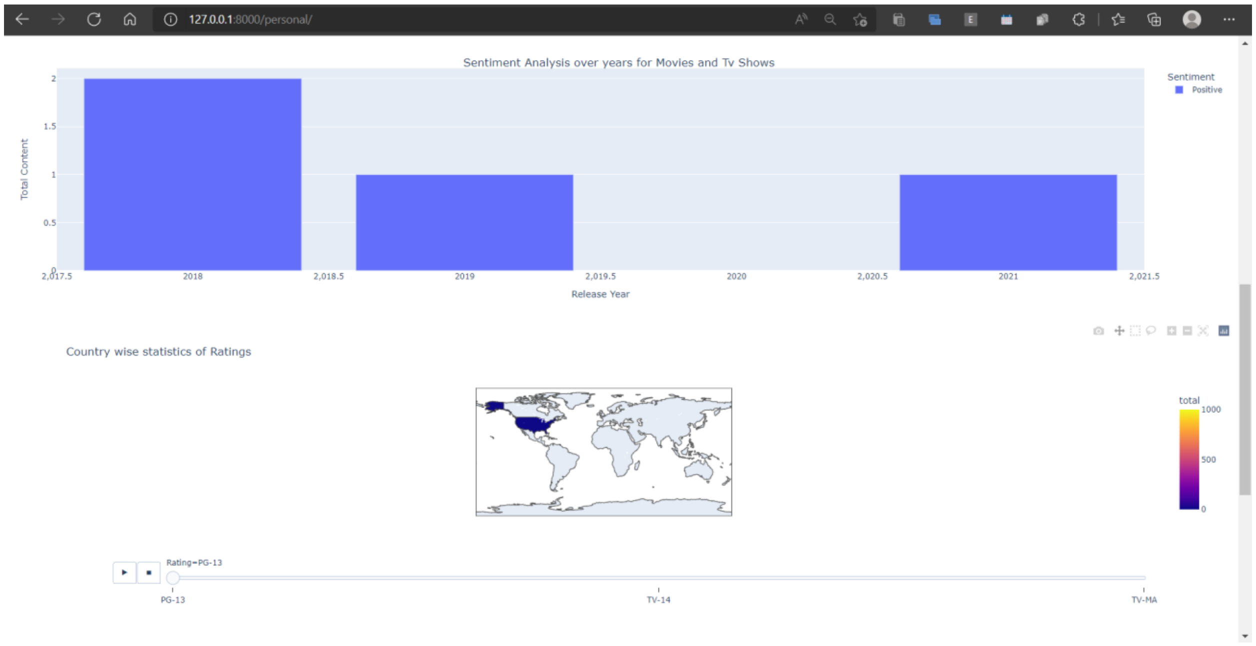
Movie Statistics Calculator

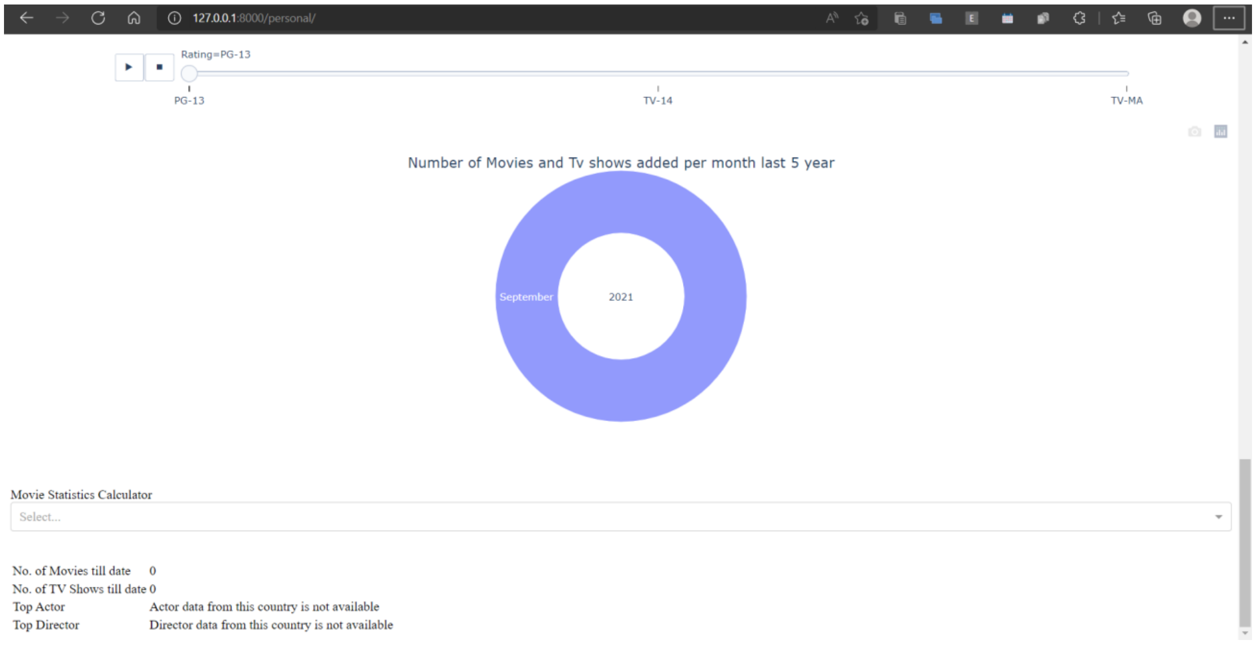
Egypt

No. of Movies till date	4
No. of TV Shows till date	1
Top Actor	Hassan Hosny
Top Director	No Data

Personalized Dashboard Output:






127.0.0.1:8000/personal/
Rating=PG-13
PG-13
TV-14
TV-MA
Number of Movies and Tv shows added per month last 5 year
September
2021
Movie Statistics Calculator
Select...

No. of Movies till date 0
No. of TV Shows till date 0
Top Actor Actor data from this country is not available
Top Director Director data from this country is not available

X. CONCLUSION

Since raw data cannot directly be visualized and is not interpretable by the user thus we use the dashboard feature to overcome this problem.

Our ultimate goal with Dashboard is to meet all the challenges for our myriad users such as:

- Data Transparency
- Access To Data
- Interactivity

A lot of companies use the dashboard as a medium to understand their end users and their needs.

REFERENCES

- https://www.kaggle.com/datasets/shivamb/netflix-shows/version/3
- Welcome to Flask — Flask Documentation (2.1.x) (palletsprojects.com)
- SQLite Documentation
- Dash Documentation & User Guide | Plotly