

▼ STC Jawwy

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
"""
Here we install libraries that are not installed by default
Example: pyslsb
Feel free to add any library you are planning to use.
"""

'\nHere we install libraries that are not installed by default \nExample:  pyslsb\nFeel free to add any library
you are planning to use \n'

!pip install pyxlsb

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyxlsb
  Downloading pyxlsb-1.0.10-py2.py3-none-any.whl (23 kB)
Installing collected packages: pyxlsb
Successfully installed pyxlsb-1.0.10

# Import the required libraries
"""
Please feel free to import any required libraries as per your need
"""

import pandas as pd      # provides high-performance, easy to use structures and data analysis tools
import pyxlsb            # Excel extension to read xlsb files (the input file)
import numpy as np       # provides fast mathematical computation on arrays and matrices
```

▼ Jawwy dataset

The dataset consists of meta details about the movies and tv shows as genre. Also details about Users activities, spent duration and if watching in High definition or standard definition. You have to analyse this dataset to find top insights, findings and to solve the four tasks assigned to you.

```
!pip install gspread
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gspread in /usr/local/lib/python3.10/dist-packages (3.4.2)
Requirement already satisfied: requests>=2.2.1 in /usr/local/lib/python3.10/dist-packages (from gspread) (2.27.1)
Requirement already satisfied: google-auth in /usr/local/lib/python3.10/dist-packages (from gspread) (2.17.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.2.1) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.2.1) (2022.9.24)
Requirement already satisfied: charset-normalizer~2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests>=2.2.1) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.2.1) (3.4)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth) (5.2.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth) (0.3.0)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from google-auth->gspread) (1.16.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth->gspread) (4.9)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules) (0.5.0)
```

```
import pandas as pd
```

```
import pandas as pd
```

```
url = "/content/drive/MyDrive/source/stc-TV-Data-Set_T1.xlsx" # Replace with the actual URL of the XLSB file
```

```
# Read the XLSB file into a DataFrame using pandas
dataframet1 = pd.read_excel(url, engine="pyxlsb")
```

```
# dataframet1 = pd.read_excel("/content/stc TV Data Set_T1.xlsx",sheet_name="Final_Dataset")
# Please make a copy of dataset if you are going to work directly and make changes on the dataset
# you can use df=dataframe.copy()
```

```
# check the data shape
dataframet1.shape
```

```
(1048575, 13)
```

```
# display the first 5 rows
dataframet1.head()
```

	Column1	date_	user_id_mapped	program_name	duration_seconds	program_class	season	episode	program_desc
0	1	42882	26138	100 treets	40	MOVIE	0	0	Drama Movie100 Streets
1	3	42876	7946	Moana	17	MOVIE	0	0	Animation MovieMoana (HD)
2	4	42957	7418	The Mermaid Princess	8	MOVIE	0	0	Animation MovieThe Mermaid Princess (HD)
3	5	42942	19307	The Mermaid Princess	76	MOVIE	0	0	Animation MovieThe Mermaid Princess (HD)
4	7	42923	15860	Churchill	87	MOVIE	0	0	Biography MovieChurchill (HD)



```
# Data Preprocessing on the input data
```

```
dataframet1 = dataframet1.drop(columns=['Column1']) # dropping the index column
dataframet1['program_name'] = dataframet1['program_name'].str.strip() # trim spaces in movies names to avoid misspel
dataframet1['date_'] = pd.to_datetime(dataframet1['date_'], unit='d', origin='30/12/1899') # read date column as dat
dataframet1[['duration_seconds', 'season', 'episode', 'series_title', 'hd']] = dataframet1[['duration_seconds', 'season'
dataframet1[['user_id_mapped', 'program_name', 'program_class', 'program_desc', 'program_genre', 'original_name']] = dataf
```

```
<ipython-input-15-e1801d30fd0e>:4: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the defa
dataframet1['date_'] = pd.to_datetime(dataframet1['date_'], unit='d', origin='30/12/1899') # read date colum
```

```
# display the dataset after applying data types
dataframet1.head()
```

	date_	user_id_mapped	program_name	duration_seconds	program_class	season	episode	program_desc	program_g
0	2017-05-27	26138	100 treets	40	MOVIE	0	0	Drama Movie100 Streets	D
1	2017-05-21	7946	Moana	17	MOVIE	0	0	Animation MovieMoana (HD)	Anim
2	2017-08-10	7418	The Mermaid Princess	8	MOVIE	0	0	Animation MovieThe Mermaid Princess (HD)	Anim
3	2017-07-26	19307	The Mermaid Princess	76	MOVIE	0	0	Animation MovieThe Mermaid Princess (HD)	Anim
4	2017-07-07	15860	Churchill	87	MOVIE	0	0	Biography MovieChurchill (HD)	Biogr

```
# describe the numeric values in the dataset
dataframet1.describe()
```

	duration_seconds	season	episode	series_title	hd
count	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06
mean	1.230957e+03	1.342139e+00	6.157952e+00	1.205922e-02	3.862728e-01
std	6.821058e+03	2.104095e+00	1.222015e+01	1.091504e-01	4.868946e-01
min	2.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	5.200000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	1.190000e+02	1.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00
75%	1.328000e+03	1.000000e+00	9.000000e+00	0.000000e+00	1.000000e+00
max	1.461329e+06	2.300000e+01	2.820000e+02	1.000000e+00	1.000000e+00

```
# check if any column has null value in the dataset
dataframet1.isnull().any()
```

date_	False
user_id_mapped	False
program_name	False
duration_seconds	False
program_class	False
season	False
episode	False
program_desc	False
program_genre	False
series_title	False
hd	False
original_name	False
dtype: bool	

▼ Task 1

You are required to work on task one to study and HD flag for available dataset

Task 1 details

At stc TV, we will study user behavior to ensure that all customers' desires are met and the desired content is provided.

So, by looking at the attached Excel file in the supporting resources and using your analytical skills, we want you to do the following:

- Firstly: Determine the appropriate methods for classifying and analyzing viewers according to the Program Class: A movie or a series
- Secondly: Studying the different viewing patterns of users and determining the category that watches stc TV in standard definition (SD) versus the category that watches it in high quality (HD).
- Thirdly: After you have finished analyzing and studying user behavior, share with us your findings by writing code on co-lab

```
# make a copy of the dataframe for working on task 1
dft1=dataframet1.copy()

# Here we try to get the most watched movies (Total Views / Total Users Views / Total watch time)
# For series we concatenated the Session episode to differentiate between episodes
grouped=dft1.copy()
grouped.loc[grouped['program_class'] == 'SERIES/EPISODES', 'program_name'] = grouped['program_name']+ '_SE'+grouped['s
grouped = grouped.groupby(['program_name','program_class'])\
.agg({'user_id_mapped': [('co1', 'nunique'),('co2', 'count')],\
      'duration_seconds': [('co3', 'sum')]}).reset_index()
grouped.columns = ['program_name','program_class','No of Users who Watched', 'No of watches', 'Total watch time in se
grouped['Total watch time in heures']=grouped['Total watch time in seconds']/3600
grouped = grouped.drop(columns=['Total watch time in seconds'])
grouped = grouped.sort_values(by=['Total watch time in heures', 'No of watches','No of Users who Watched'], ascending

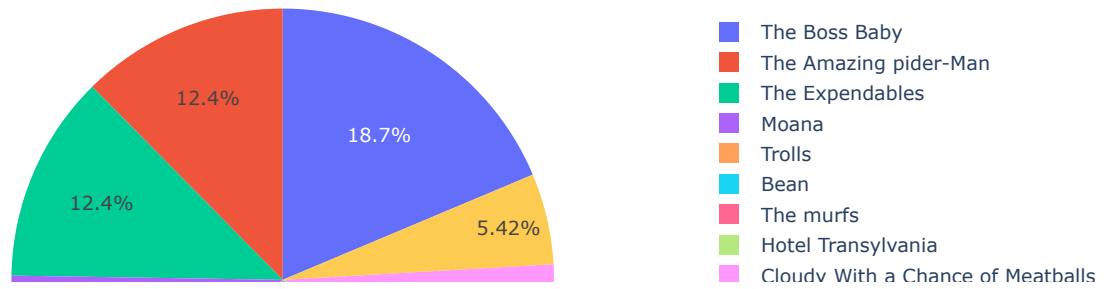
# show the result
grouped.head(35)
```

	program_name	program_class	No of Users who Watched	No of watches	Total watch time in heures
0	The Boss Baby	MOVIE	3389	24047	2961.350833
1	The Amazing pider-Man	MOVIE	1011	2877	1966.119167
2	The Expendables	MOVIE	853	2119	1961.159444
3	Moana	MOVIE	2173	8081	1706.176944
4	Trolls	MOVIE	2613	13793	1601.023056
5	Bean	MOVIE	949	3617	1423.955000
6	The murfs	MOVIE	867	3132	1342.141111
7	Hotel Transylvania	MOVIE	491	1947	1096.533611
8	Cloudy With a Chance of Meatballs	MOVIE	683	2076	948.674722
9	The Man With The Iron Fists	MOVIE	707	2505	859.626389
10	Salt	MOVIE	563	1082	767.392778
11	Unbroken	MOVIE	625	1429	763.078333
12	ParaNorman	MOVIE	614	1746	747.065556
13	Youm Maloosh Lazma	MOVIE	1131	2278	718.109722
14	Ferdinand	MOVIE	1278	6817	714.223056
15	White Chicks	MOVIE	307	916	711.840833
16	Jurassic Park	MOVIE	504	1192	693.394444
17	The November Man	MOVIE	494	1219	679.492222
18	Total Recall	MOVIE	587	1108	661.820000
19	Robin Hood	MOVIE	588	1209	643.935000

```
# we import Visualization libraries
# you can ignore and use any other graphing libraries
import matplotlib.pyplot as plt # a comprehensive library for creating static, animated, and interactive visualizatio
import plotly #a graphing library makes interactive, publication-quality graphs. Examples of how to make line plots,
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

```
# plot top 10 Programs
fig = px.pie(grouped.head(10), values='Total watch time in heures', names='program_name',\
             hover_data=['program_class'],title='top 10 programs in total watch time in heures')
fig.show()
```

top 10 programs in total watch time in heures



```
# Here we try to study the customer experience against Program class
grouped=dft1.copy()
grouped = grouped.groupby('program_class')\
.agg({'user_id_mapped': [('co1', 'nunique'),('co2', 'count')],\
     'duration_seconds': [('co3', 'sum')] }).reset_index()
grouped.columns = ['program_class','No of Users who Watched', 'No of watches', 'Total watch time in seconds']
grouped['Total watch time in heures']=grouped['Total watch time in seconds']/3600
grouped = grouped.drop(columns=['Total watch time in seconds'])
grouped = grouped.sort_values(by=['Total watch time in heures', 'No of watches','No of Users who Watched'], ascending
```

```
# show the result
grouped.head()
```

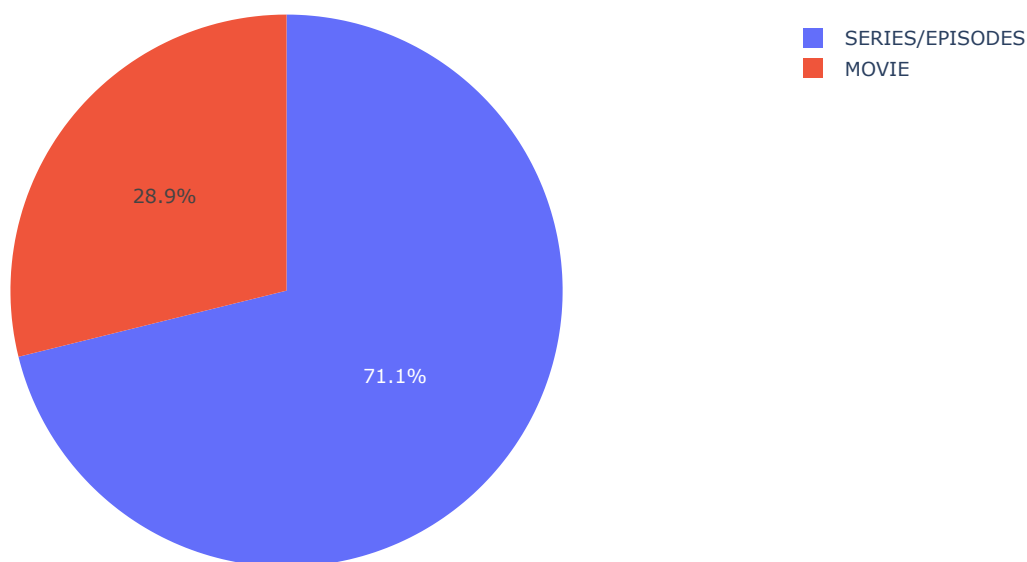
	program_class	No of Users who Watched	No of watches	Total watch time in heures
0	SERIES/EPISODES	3901	560174	255097.787500
1	MOVIE	11355	488401	103444.145556



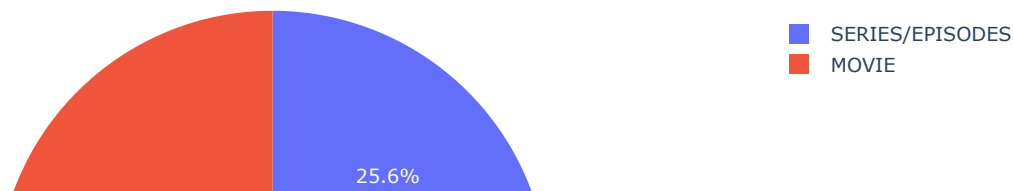
```
# plot the total watch time against total number of users and report your findings
fig = px.pie(grouped, values='Total watch time in heures', names='program_class',\
             hover_data=['program_class'],title='Total duration spent by program_class')
fig2 = px.pie(grouped, values='No of Users who Watched', names='program_class',\
              hover_data=['program_class'],title='Total Users watching by program_class')
```

```
fig.update_traces(sort=False)
fig2.update_traces(sort=False)
fig.show()
fig2.show()
```

Total duration spent by program_class



Total Users watching by program_class



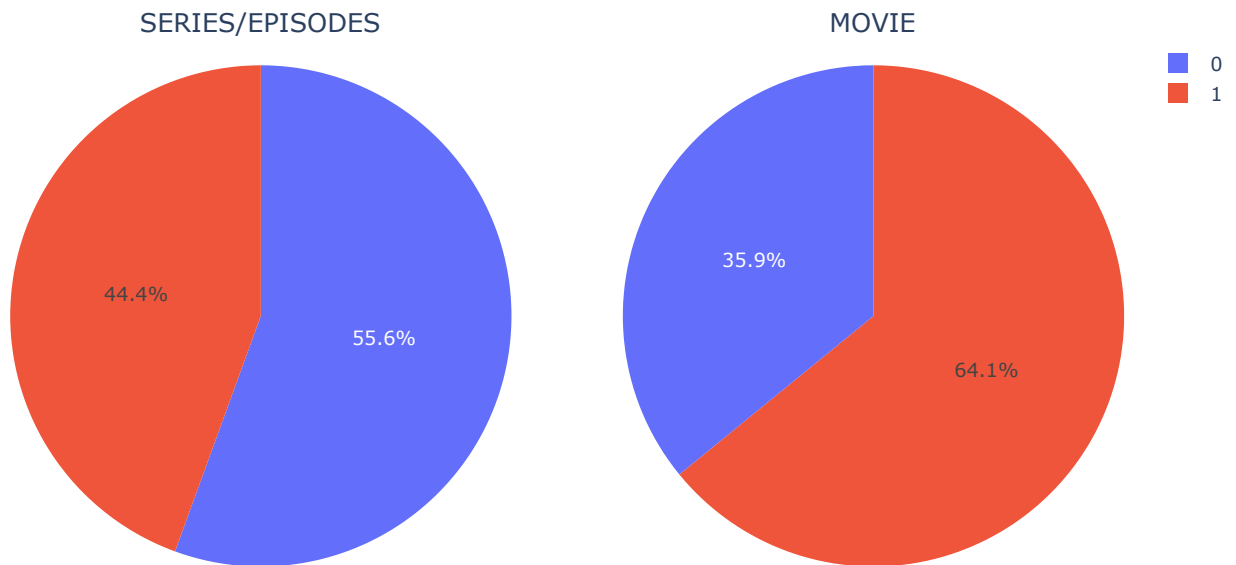
```
# This is task 1
# study the relation and user's behaviour Against HD flag, feel free to include any comparison
grouped=dft1.copy()
grouped = grouped.groupby(['hd','program_class'])\
.agg({'user_id_mapped': [ ('co1', 'nunique'), ('co2', 'count') ],\
'duration_seconds': [ ('co3', 'sum') ] }).reset_index()
grouped.columns = ['hd','program_class','No of Users who Watched', 'No of watches', 'Total watch time in seconds']
grouped['Total watch time in heures']=grouped['Total watch time in seconds']/3600
grouped = grouped.drop(columns=['Total watch time in seconds'])
grouped = grouped.sort_values(by=['Total watch time in heures', 'No of watches','No of Users who Watched'], ascending

# show the result
grouped.head()
```

	hd	program_class	No of Users who Watched	No of watches	Total watch time in heures
0	0	SERIES/EPISODES	3282	486884	229776.593333
1	1	MOVIE	10880	331746	64856.366111
2	0	MOVIE	6093	156655	38587.779444
3	1	SERIES/EPISODES	2625	73290	25321.194167

```
# plot the results
fig = make_subplots(1, 2, specs=[[{'type':'domain'}], {'type':'domain'}],
                    subplot_titles=['SERIES/EPISODES', 'MOVIE'])
fig.add_trace(go.Pie(labels=grouped[grouped['program_class'] == 'SERIES/EPISODES']['hd'],\
                      values=grouped[grouped['program_class'] == 'SERIES/EPISODES']['No of Users who Watched'], name='
# add the 2nd graph
fig.add_trace(go.Pie(labels=grouped[grouped['program_class'] == 'MOVIE']['hd'],\
                      values=grouped[grouped['program_class'] == 'MOVIE']['No of Users who Watched'], name='MOVIE',dir
fig.update_layout(title_text='No of Users who Watched Vs Program quality flag')
fig.show()
```

No of Users who Watched Vs Program quality flag



▼ Task 2

In this task, you have a new goal!

See the attached excel file in this task and help us using the available data on the following:

- First: Build an easy and simple model that enables officials to make decisions and make them able to predict the number of expected views of customers during the next two months, and determine possible peak times
- Second: Write code on the co-lab about the results that you got from building the model

▼ Jawwy dataset

The dataset includes total watching hours for customers per day.

You are required to work on predicting the forecast for the watching hours.


```
import pandas as pd


# Read the XLSB file into a DataFrame using pandas
dataframet2 = pd.read_excel("/content/drive/MyDrive/source/stc TV Data Set_T2.xlsb",index_col=0)

# dataframet2 = pd.read_excel("/content/stc TV Data Set_T2.xlsb", index_col=0)
# Please make a copy of dataset if you are going to work directly and make changes on the dataset
# you can use df=dataframe.copy()


# check the data shape
dataframet2.shape

(86, 2)

# display the first 5 rows
dataframet2.head()
```

	date_	Total_watch_time_in_houres	
0	2018-01-01	1123.551944	
1	2018-01-02	1000.129722	
2	2018-01-03	881.924444	
3	2018-01-04	782.669444	
4	2018-01-05	1051.939444	

```
# describe the numeric values in the dataset
dataframet2.describe()
```

	Total_watch_time_in_houres	
count	86.000000	
mean	780.817926	
std	122.992002	
min	562.124722	
25%	707.709653	
50%	763.181389	
75%	840.985278	
max	1123.551944	

```
# check if any column has null value in the dataset
dataframet2.isnull().any()
```


```
date_          False
Total_watch_time_in_houres  False
dtype: bool
```

```
# we import Visualization libraries
# you can ignore and use any other graphing libraries
import matplotlib.pyplot as plt # a comprehensive library for creating static, animated, and interactive visualizations
import plotly # a graphing library makes interactive, publication-quality graphs. Examples of how to make line plots,
import plotly.express as px
```

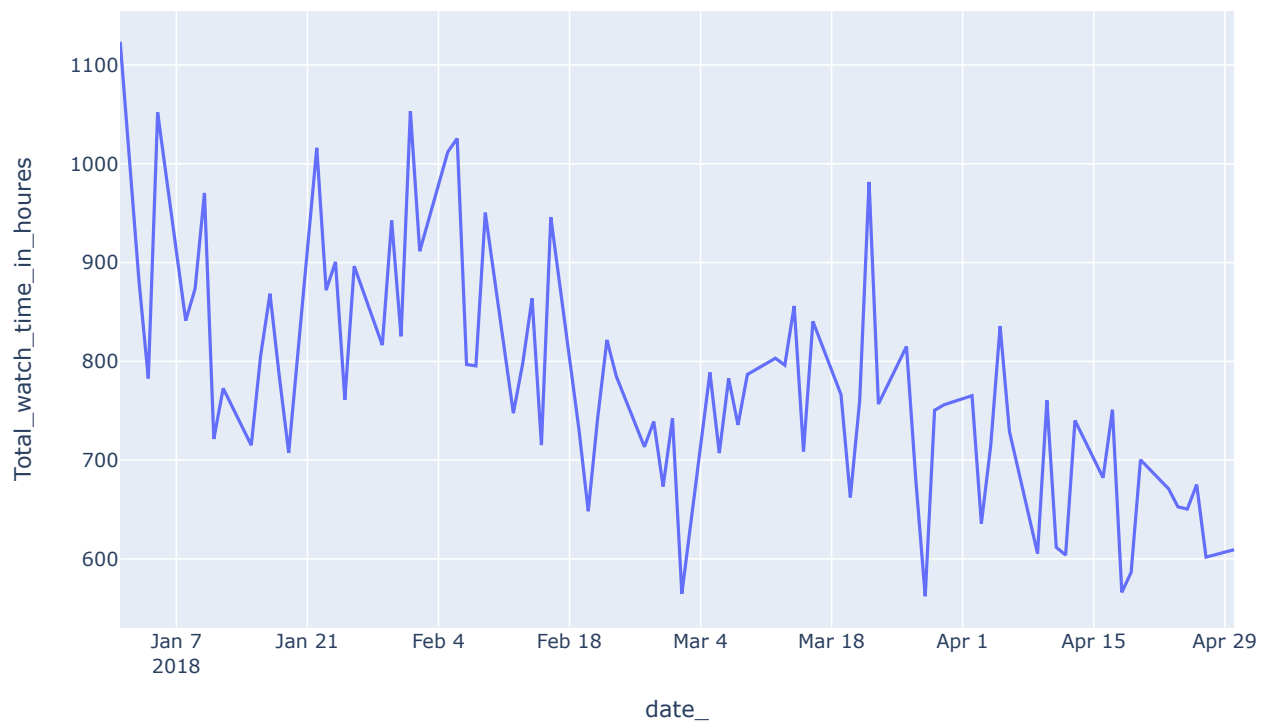
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Setting the date as index
dataframet2.set_index('date_', inplace=True)

# Display the dataframe after setting the date as index
dataframet2.head()
```

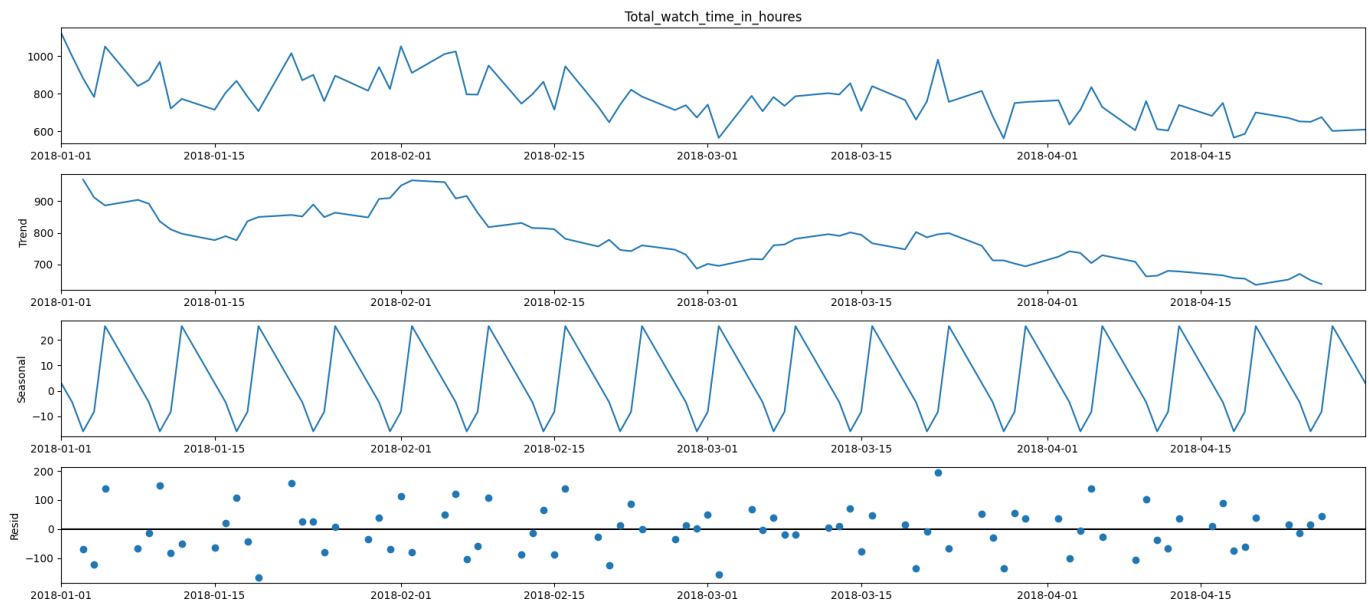
Total_watch_time_in_houres 	
date_	
2018-01-01	1123.551944
2018-01-02	1000.129722
2018-01-03	881.924444
2018-01-04	782.669444
2018-01-05	1051.939444

```
# show the dataframe
fig = px.line(dataframet2, y="Total_watch_time_in_houres")
fig.show()
```



```
# using the previous dataset build a prediction model to predict the expected watch time for the next two months
y = dataframet2["Total_watch_time_in_houres"]
import statsmodels.api as sm
from pylab import rcParams
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(y, model='additive')
```

```
fig = decomposition.plot()
plt.show()
```



```
import itertools
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]
print('Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))

Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)

for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                             order=param,
                                             seasonal_order=param_seasonal,
                                             enforce_stationarity=True,
                                             enforce_invertibility=False)

            results = mod.fit(dis=0)
            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

```

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:946.3835810154476
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:911.3145856366127
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:1025.7966752044695
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:1027.6654471304373
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:920.3449042845947
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:910.7381314171264

mod = sm.tsa.statespace.SARIMAX(y,
                                order=(0, 1, 1),
                                seasonal_order=(0, 1, 1, 12),
                                enforce_stationarity=True,
                                enforce_invertibility=False)
results = mod.fit(dis=0)
print(results.summary().tables[1])

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:
No frequency information was provided, so inferred frequency B will be used.

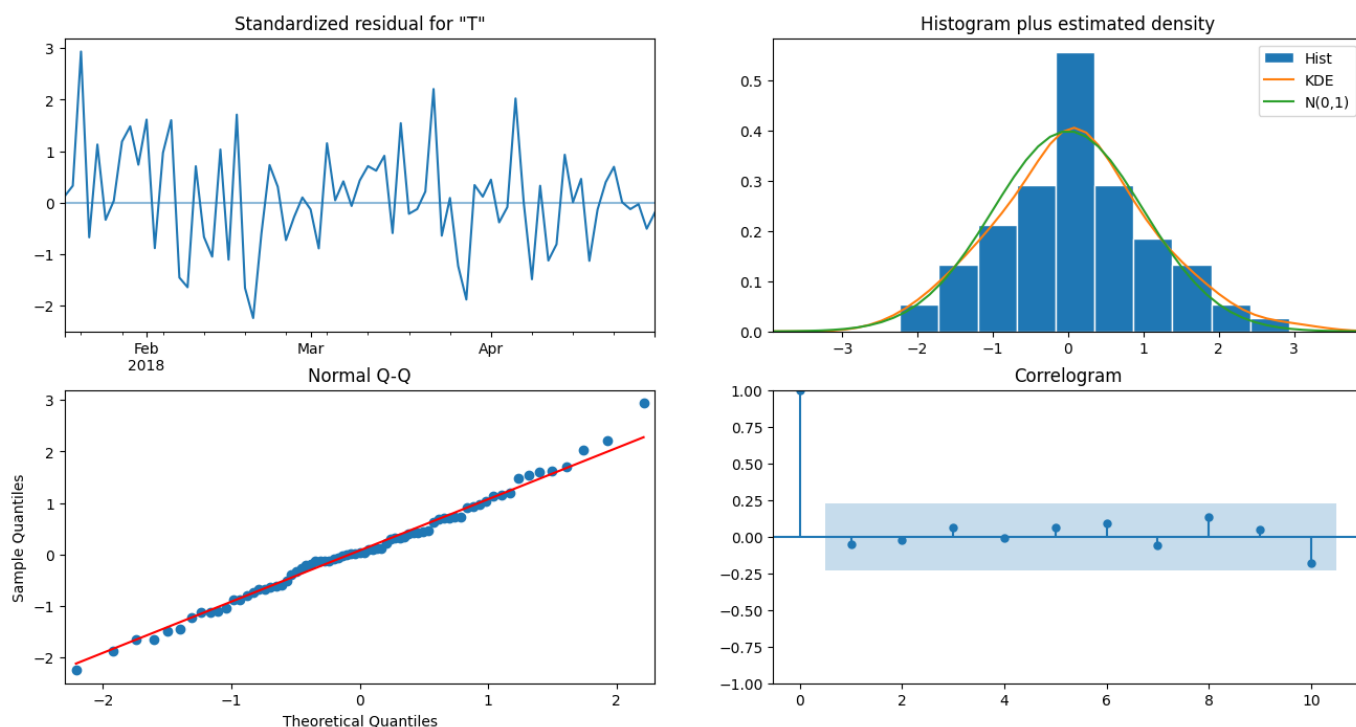
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:

```

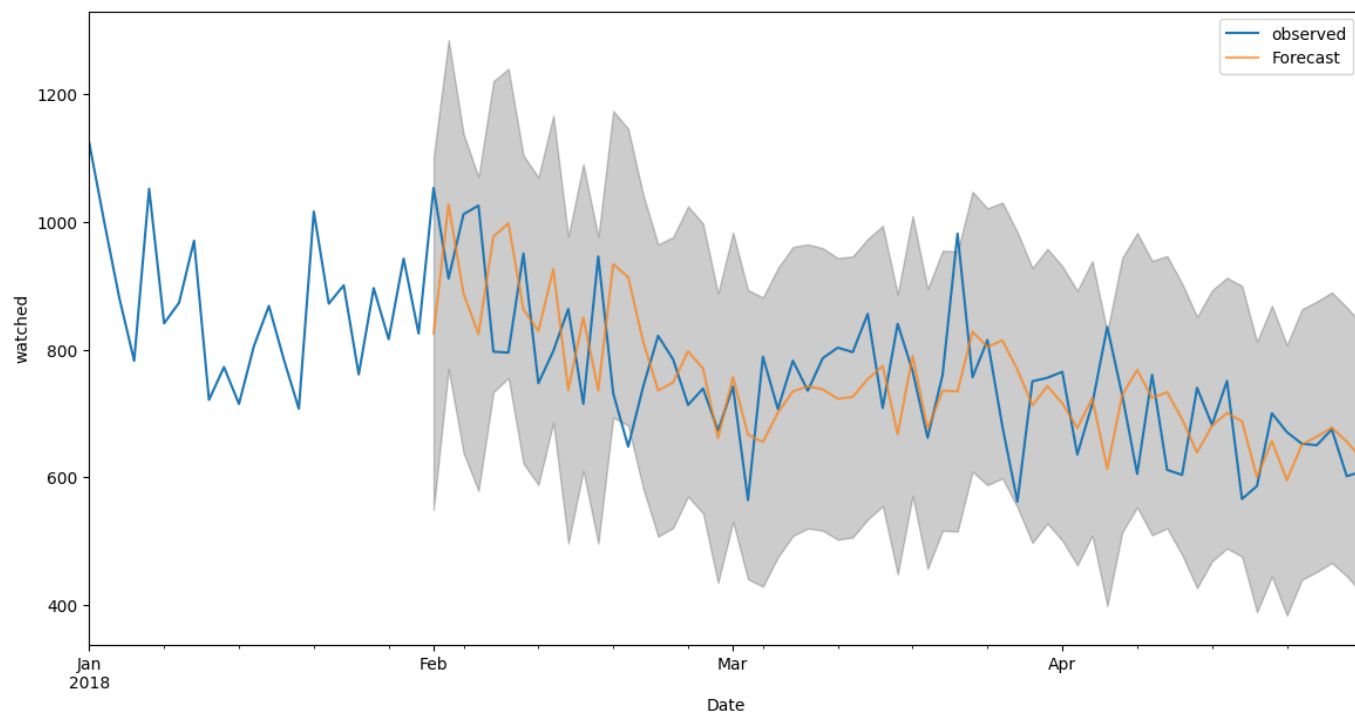
No frequency information was provided, so inferred frequency B will be used.

	coef	std err	z	P> z	[0.025	0.975]
ma.L1	-0.7956	0.087	-9.121	0.000	-0.967	-0.625
ma.S.L12	-1.0000	0.158	-6.339	0.000	-1.309	-0.691
sigma2	1.002e+04	1.57e-05	6.36e+08	0.000	1e+04	1e+04

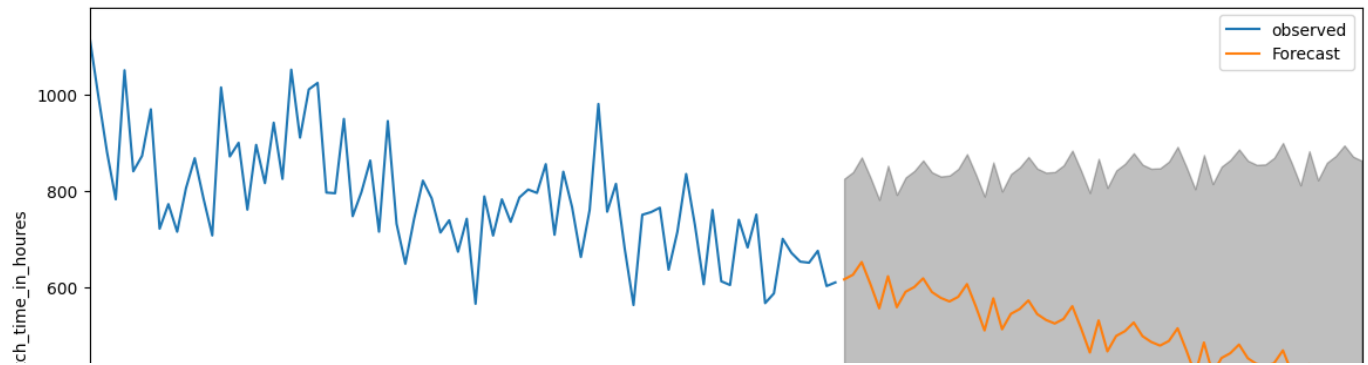
```
results.plot_diagnostics(figsize=(16, 8))
plt.show()
```



```
# show the Model out come agianst the actual data
pred = results.get_prediction(start=pd.to_datetime('2018-02-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = dataframe2['Total_watch_time_in_houres'].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='Forecast', alpha=.7, figsize=(14, 7))
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date')
ax.set_ylabel('watched')
plt.legend()
plt.show()
```



```
# show forecasts
pred_uc = results.get_forecast(steps=60)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='observed', figsize=(14, 7))
pred_uc.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_ci.index,
               pred_ci.iloc[:, 0],
               pred_ci.iloc[:, 1], color='k', alpha=.25)
ax.set_xlabel('Date')
ax.set_ylabel('Total_watch_time_in_houres')
plt.legend()
plt.show()
```



Task 3

200

Task details

After studying the data, use machine learning algorithms for the following:

- First: build a model to provide recommendations to the user based on the views of other users who share the same preferences, to make the viewing experience on stc TV richer and richer.
- Second: Show top 5 recommendations for people who watched Moana.
- Third: Once you've finished your hard work, share a sample of the results generated from the code-recommended system on your co-lab.

Jawwy dataset

The dataset consists of details about each customer and the movies and/or tv shows watched in addition to the genre.

You are required to work on task three to build a recommendation engine for our platform to Recommend movies to users that they might be interested in

```
url = "/content/drive/MyDrive/source/stc TV Data Set_T3.xlsx" # Replace with the actual URL of the XLSB file
```

```
# Read the XLSB file into a DataFrame using pandas
dataframet3 = pd.read_excel(url, index_col=0)
```

```
# check the data shape
dataframet3.shape
```

```
(1048575, 5)
```

```
# display the first 5 rows
dataframet3.head()
```

user_id_mapped	program_name	rating	date_	program_genre
----------------	--------------	--------	-------	---------------

```
# describe the numeric values in the dataset
dataframet3.describe()
```

	user_id_mapped	rating
count	1.048575e+06	1.048575e+06
mean	1.709266e+04	2.497283e+00
std	1.003513e+04	1.119837e+00
min	1.000000e+00	1.000000e+00
25%	8.253000e+03	1.000000e+00
50%	1.714900e+04	2.000000e+00
75%	2.566500e+04	3.000000e+00
max	3.428000e+04	4.000000e+00

```
# check if any column has null value in the dataset
dataframet3.isnull().any()
```

user_id_mapped	False
program_name	False
rating	False
date_	False
program_genre	False
dtype:	bool

```
# we import Visualization libraries
# you can ignore and use any other graphing libraries
import matplotlib.pyplot as plt # a comprehensive library for creating static, animated, and interactive visualizations
import plotly # a graphing library makes interactive, publication-quality graphs. Examples of how to make line plots,
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

```
# Pair-wise table for user rating for all movies
movie_features_df=dataframet3.pivot_table(index='program_name',columns='user_id_mapped',values='rating').fillna(0)
movie_features_df.head()
```

user_id_mapped	1	5	9	11	15	17	20	26	28	30	...	34259	34261	34263	34265	34267	34269	34271
program_name																		
#FollowFriday	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10 Days in a Madhouse	0.0	0.0	0.0	0.0	1.5	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
100 treets	0.0	0.0	0.0	1.0	2.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
101 Dalmatians	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
102																		

```
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
movie_features_df_matrix = csr_matrix(movie_features_df.values)
```



```
model_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
model_knn.fit(movie_features_df_matrix)
```

```
NearestNeighbors
NearestNeighbors(algorithm='brute', metric='cosine')
```

```
movie_features_df_selection= movie_features_df.reset_index()
movie_features_df_selection[['program_name']]
dataframet3[['program_name']].head(10)
```

	program_name	
0	100 treet	
1	Moana	
2	The Mermaid Princess	
3	The Mermaid Princess	
4	Churchill	
5	Beavis And Butt-Head Do America	
6	The Mermaid Princess	
7	Coco	
8	Kidnap	
9	The Accountant	

```
# show the recommendations (top 5)
```

```
program_name='Moana'
recomendations=5
```

```
query_index = np.random.choice(movie_features_df.shape[0])
distances, indices = model_knn.kneighbors(movie_features_df.
iloc[movie_features_df_selection.index[movie_features_df_selection['program_name'] == program_na
n_neighbors = recommendations+1)
```

```
for i in range(0, len(distances.flatten())):
    if i == 0:
        print('Recommendations for {0}:\n'.format(movie_features_df.index[movie_features_df_selection.index[movie_fea
    else:
        print('{0}: {1}, with distance of {2}:".format(i, movie_features_df.index[indices.flatten()][i]], distances.fl
```

```
Recommendations for Moana:
```

```
1: Trolls, with distance of 0.42764217010640215:
2: Surf's Up : WaveMania, with distance of 0.4705763355181768:
3: The Mermaid Princess, with distance of 0.5066377099343184:
4: The Boss Baby, with distance of 0.551442834662541:
5: The Jetsons & WWE: Robo-WrestleMania!, with distance of 0.5610577907608365:
```

✓ 0 sn. tamamlanma zamanı: 12:10

● ✕