STC Jawwy

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

```
Requirement already satisfied: pyxlsb in /usr/local/lib/python3.10/dist-packages (1.0.10)
```

```
!pip install Openpyxl
```

```
Requirement already satisfied: Openpyxl in /usr/local/lib/python3.10/dist-packages (3.1.2)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.10/dist-packages (from Openpyxl) (1.1.0)
```

```
# Import the required libraries
"""
Please feel free to import any required libraries as per your needs
"""
import pandas as pd     # provides high-performance, easy to use structures and data analysis tools
import pyxlsb           # Excel extention to read xlsb files (the input file)
import numpy as np      # provides fast mathematical computation on arrays and matrices
from datetime import date
```

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.

```
dataframe = pd.read_excel("/content/sample_data/stc TV Data Set_T1.xlsx")
# 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
dataframe.shape
```

```
(1048575, 13)
```

```
#df=dataframe.copy()
```

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

	Column1	date_	user_id_maped	program_name	duration_seconds	program_class	seas
0	1	2017-05-27	26138	100 treets	40	MOVIE	
1	3	2017-05-21	7946	Moana	17	MOVIE	
2	4	2017-08-10	7418	The Mermaid Princess	8	MOVIE	
3	5	2017-07-26	19307	The Mermaid Princess	76	MOVIE	
4	7	2017-	15860	Churchill	87	MOVIE	

```
# Data Preprocessing on the input data
dataframe = dataframe.drop([1,1])         # dropping the index column
```

```
dataframe['program_name'] = dataframe['program_name'].str.strip()  # trim spaces in movies names to avoid misspellings in input data
#dataframe['date_'] = pd.to_datetime(dataframe['date_'], unit='D', origin='1899-12-30')  # read date column as date data type
dataframe[['duration_seconds', 'season','episode','series_title','hd']] = dataframe[['duration_seconds', 'season','episode','series_
dataframe[['user_id_maped', 'program_name','program_class','program_desc','program_genre','original_name']] = dataframe[['user_id_ma
```

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

	Column1	date_	user_id_maped	program_name	duration_seconds	program_class	seas
0	1	2017-05-27	26138	100 treets	40	MOVIE	
2	4	2017-08-10	7418	The Mermaid Princess	8	MOVIE	
3	5	2017-07-26	19307	The Mermaid Princess	76	MOVIE	
4	7	2017-07-07	15860	Churchill	87	MOVIE	
5	8	2017-08-19	20775	Beavis And Butt-Head Do America	3	MOVIE	

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

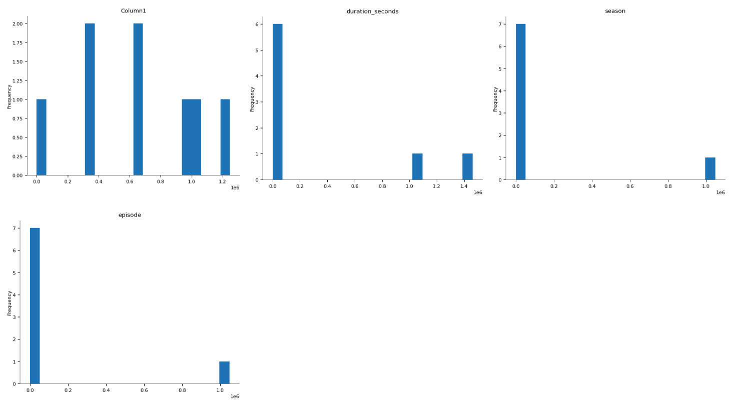
1 to 8 of 8 entries Filter

index	Column1	duration_seconds	season	episode	s
count	1048574.0	1048574.0	1048574.0	1048574.0	
mean	628173.5990669233	1230.9583701293375	1.342139896659654	6.157958331982292	0.0120
std	359703.70509784657	6821.061038505865	2.1040959772571464	12.22015904257431	0.1091
min	1.0	2.0	0.0	0.0	
25%	318067.5	52.0	0.0	0.0	
50%	630355.5	119.0	1.0	1.0	
75%	939822.75	1328.0	1.0	9.0	
max	1247852.0	1461329.0	23.0	282.0	

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

Distributions



2-d distributions

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

```
Column1             False
date_               False
```

```
user_id_maped      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

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

```python
# 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=df.copy()
grouped.loc[grouped['program_class'] == 'SERIES/EPISODES', 'program_name'] = grouped['program_name']+'_SE'+grouped['season'].astype(
grouped = grouped.groupby(['program_name','program_class'])\
.agg({'user_id_maped': [('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 seconds']
grouped['Total watch time in houres']=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 houres', 'No of watches','No of Users who Watched'], ascending=False).reset_i

```

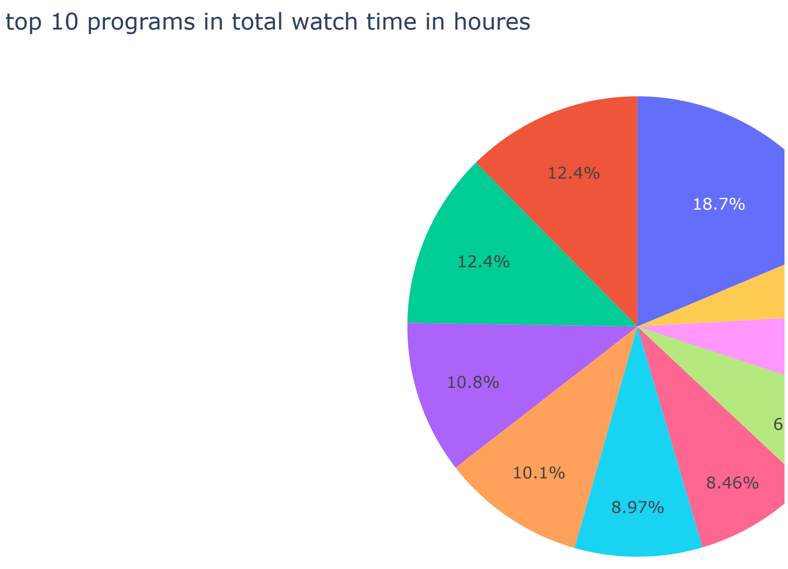
```python
# show the result
grouped.head(35)
```

	program_name	program_class	No of Users who Watched	No of watches	Total watch time in houres
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	8080	1706.172222
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
20	Public Enemies	MOVIE	368	716	634.035000
21	Daddy Day Camp	MOVIE	263	647	625.338333
22	Oblivion	MOVIE	790	1678	609.391111
23	Blitz	MOVIE	562	1200	570.521944
24	War for the Planet of the Apes	MOVIE	879	2028	567.597778
25	Inside Man	MOVIE	532	1567	560.386111
26	Bad Boys	MOVIE	438	871	559.277500
27	Easy A	MOVIE	513	990	557.068611
28	Battleship	MOVIE	634	1324	552.857222
29	Baywatch	MOVIE	2062	7436	548.995556
30	Police tory	MOVIE	409	737	520.077222

Next steps: Generate code with grouped | View recommended plots

```
# 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, scatter 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 houres', names='program_name',\
            hover_data=['program_class'],title='top 10 programs in total watch time in houres')
fig.show()
```



```
# Here we try to study the customer experience against Program class
grouped=df.copy()
grouped = grouped.groupby('program_class')\
.agg({'user_id_maped': [('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 houres']=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 houres', 'No of watches','No of Users who Watched'], ascending=False).reset_i

```

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

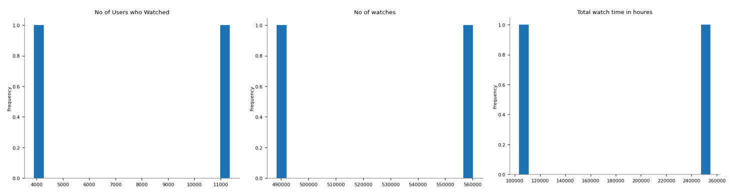
1 to 2 of 2 entries Filter

index	program_class	No of Users who Watched	No of watches	Total watch time in houres
0	SERIES/EPISODES	3901	560174	255097.7875
1	MOVIE	11355	488400	103444.14083333334

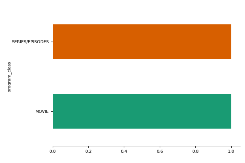
Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

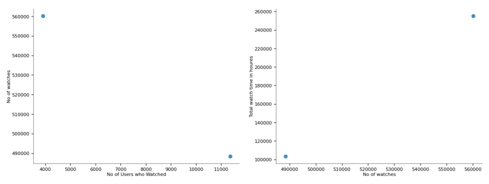
Distributions



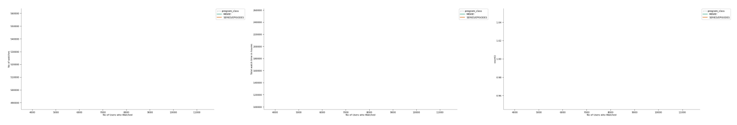
Categorical distributions



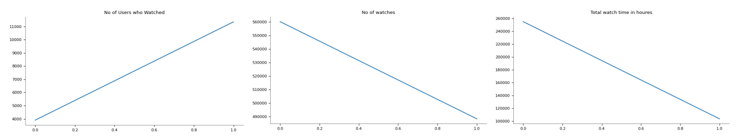
2-d distributions



Time series



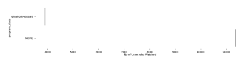
Values



Faceted distributions

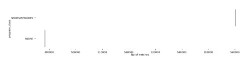
```
<string>:5: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14
```



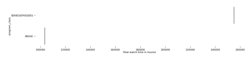
```
<string>:5: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14
```



```
<string>:5: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14
```

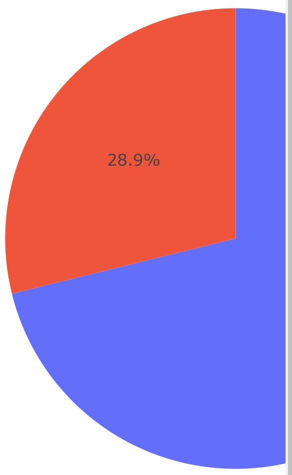


Next steps: Generate code with grouped | View recommended plots

```
# plot the total watch time against total number of users and report your findings
fig = px.pie(grouped, values='Total watch time in houres', 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