



CINEMA VS STREAMING

Are good old cinema still relevant today?

Presented by: Aline Hornoff, Camille Evangelista, Edbert Widjaja & Carmen Sin

TOPIC OUTLINE

Introduction



Cinema footing over times (Leaflet)



Number of Exhibitors, screens and seats through the years



Price comparison between cinema admission fare to streaming fee



Number of Box office movie produce



Movie goers' attendance over the years



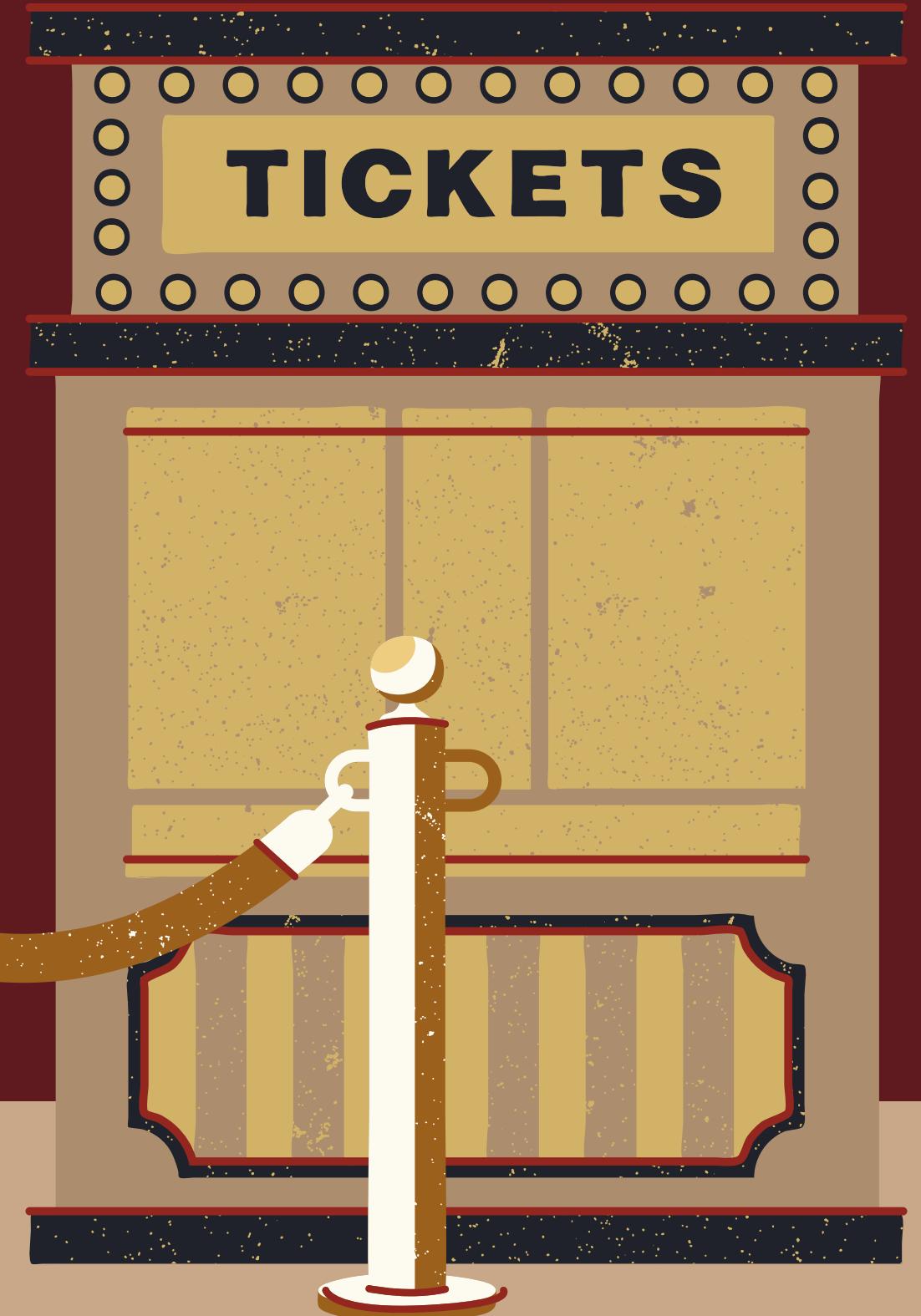
Netflix subscribers numbers



Conclusion



INTRODUCTION



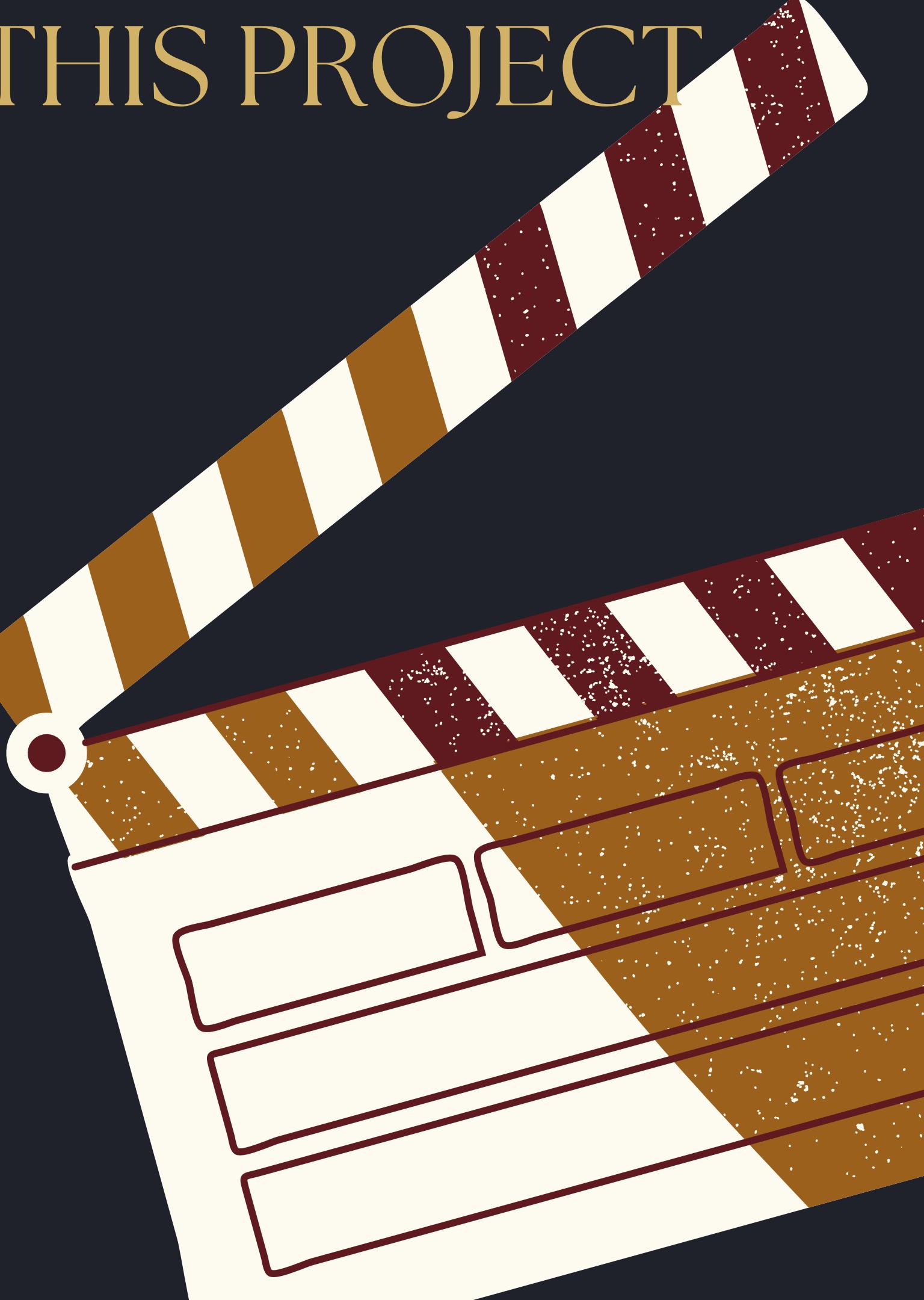
INTRODUCTION

- Back in the day if people wanted to watch the latest movie everyone else was talking about, they go to the cinema with their family or friends. In case they did not make it on the movie schedule, they had to wait for it to be available in their nearest video rental shop.
- Nowadays due to technological advancement we have the option to use streaming services in the comfort of our own homes to watch movies. This option gained more popularity when Covid happened as most people chose to stay home and also restrictions were put in place on public spaces. This made us ask the question, are cinemas still relevant?
- Are cinema still have the demand?
We intend to answer the question from viewing different angles.
- Are we seeing the contraction of cinemas' footprint?
- How often movie goer pay and see the movies in the cinema over time?
- Streaming platforms - how competitive are different streaming offering? (excluding free for view platforms)
- In the case of giant of streaming provider, Netflix. How many subscribers over the years?
- Number of BoxOffice production over the years?



METHODS ADOPTED FOR THIS PROJECT

- (1) Web scraping data for all Australian cinema location, clean up data using ETL and using leaflet to map out all their location with markers.
- (2) Read in data and use d3 to produce interactive line chart to display the changes in screen and seating capacity over time
-
- (3) Web scraping streaming pricing comparison webpage to get data, perform ETL to data and use d3 to map out scatterplot (relationship chart)
- (4) Web scraping webpage to get box office -number produced per year over time / Netflix subscriber numbers and make plotly chart.



METHODS ADOPTED FOR THIS PROJECT

- Python Flask- powered API, HTML/ CSS, JS and MongoDB for the database

- JS library - Cloudflare - Animate.css

<https://cdnjs.cloudflare.com/ajax/libs/animejs/2.0.2/anime.min.js>

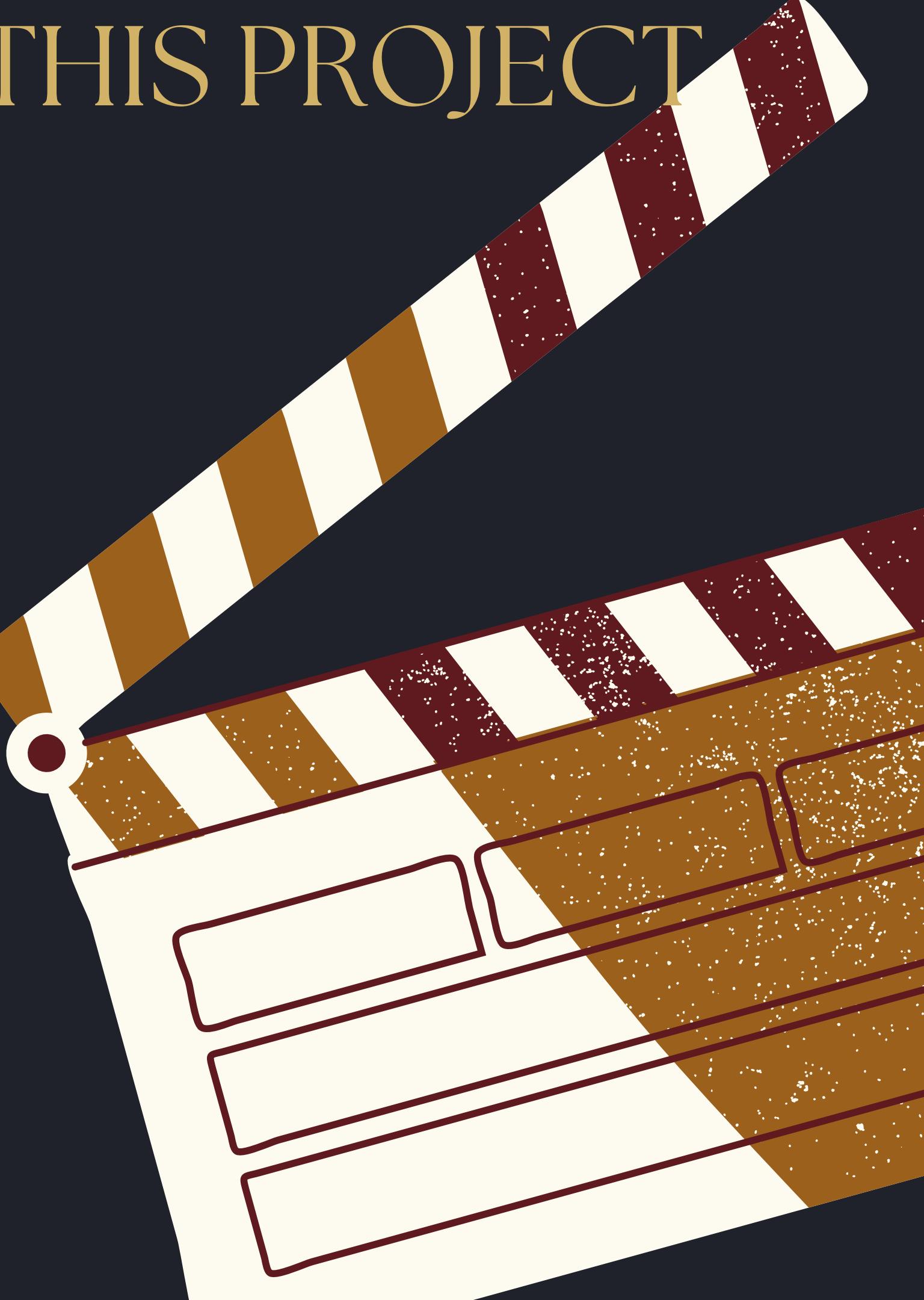
- Interactive views

Control layer for the map

User select text to view line chart

User interact with scatter plot chart to display info

User interact with the bar chart to display info



Folder Struture

```
▽ Cinema-vs-Streaming
  > Images
  > Resources
  > ss
  ▽ static
    > css
    > data
    > img
    > js
  ▽ python
    > __pycache__
    > working
    ⚡ cinema_data.ipynb
    ⚡ CinemaLocations2022.ipynb
    ⚡ db_ipynb
    ⚡ etl_.py
    ⚡ streaming.ipynb
    > Resources
  ▽ templates
    ◊ cinema.html
    ◊ cinemaMap.html
    ◊ index.html
    ◊ streaming.html
    ♦ .gitignore
    ⚡ app.py
    ⓘ README.md
```

Database

MongoDB

```
db_ipynb X
Cinema-vs-Streaming > static > python > db_ipynb > # Declare the collection & Insert
+ Code + Markdown | ▶ Run All ⌘ Clear Outputs of All Cells ⌘ Restart ⌘ Interrupt | Variables ⌘ Outline ...
```

```
[2]
# The default port used by MongoDB is 27017
# https://docs.mongodb.com/manual/reference/default-mongodb-port/
conn = 'mongodb://localhost:27017'
client = pymongo.MongoClient(conn)

# Declare the database
db = client.Cace_db
```

```
[3]
#Grab dataframes from our resources
df_boxoffice = etl_.scrape_boxoffice()
df_cinemadata = etl_.df_Cinemadata()
df_nextflixsubs = etl_.df_Nextflixsubs()
df_streamingfee = etl_.df_Streamingfee()
df_cinema1948 = etl_.df_Cinema1948()
df_cinema2022 = etl_.df_Cinema2022()
```

```
[4]
# Clean up netflixsubs dataframe
df_boxoffice = pd.merge(df_boxoffice, df_nextflixsubs, on="Year")
```

```
[5]
# Declare the collection & Insert
db.boxoffice.insert_many(df_boxoffice.to_dict('records'))
db.cinemadata.insert_many(df_cinemadata.to_dict('records'))
db.nextflixsubs.insert_many(df_nextflixsubs.to_dict('records'))
db.streamingfee.insert_many(df_streamingfee.to_dict('records'))
db.cinema1948.insert_many(df_cinema1948.to_dict('records'))
db.cinema2022.insert_many(df_cinema2022.to_dict('records'))
```

```
... <pymongo.results.InsertManyResult at 0x2368d5d2670>
```

MongoDB Compass - localhost:27017/Cace_db.boxoffice

Connect View Collection Help

localhost:27017

4 DBS 9 COLLECTIONS C

☆ FAVORITE

HOST
localhost:27017

CLUSTER
Standalone

EDITION
MongoDB 5.0.8 Community

{ } My Queries

⌚ Databases

Q Filter your data

▼ Cace_db

- boxoffice
- cinema1948
- cinema2022
- cinemadata
- nextflixsubs
- streamingfee

Cace_db.boxoffice

Documents Aggregations Schema Explain Plan

FILTER { field: 'value' }

ADD DATA VIEW

id: ObjectId('62dc886af0a5a2d2d37f837')
Year: 2022
Total_gross: "\$4,540,382,909"
LY: "-"
Releases: 274
Average: "\$16,570,740"
No1_release: "Top Gun: Maverick"
Total_Subscriber: 221744000
AsiaPacific_Percentages: 0.15
AsiaPacific_Subscribers: 33261600
Cinema_attendance_perc: 0.45
cinema_freq: 3

id: ObjectId('62dc886af0a5a2d2d37f838')
Year: 2021
Total_gross: "\$4,488,914,150"
LY: "+113.4%"
Releases: 438

Flask

```
db_ipynb app.py X
Cinema-vs-Streaming > app.py > ...
1  from flask import Flask, render_template, redirect
2  from flask_pymongo import PyMongo
3  import pymongo
4  from flask import Response
5  from bson import json_util
6  from flask import jsonify
7  import json
8  from bson.json_util import ObjectId
9
10
11
12 # Create an instance of Flask
13 app = Flask(__name__)
14
15 # Use PyMongo to establish Mongo connection
16 mongo = PyMongo(app, uri="mongodb://localhost:27017/cace_db")
17
18 conn = 'mongodb://localhost:27017'
19 client = pymongo.MongoClient(conn)
20
21
22 # Route to render index.html template using data from Mongo
23 @app.route("/")
24 def home():
25
26     # Find one record of data from the mongo database
27     _data = mongo.db.collection.find_one()
28     a = "static/Resources/box_office_scrape.json"
29
30
31     # Return template and data
32     return render_template("index.html", mars=_data, titles = [0])
33
34 @app.route("/streaming")
35 def streaming():
36     # Find one record of data from the mongo database
37     _data = mongo.db.collection.find_one()
38     a = "static/Resources/box_office_scrape.json"
39
40     return render_template("streaming.html")
41
```

db_ipynb app.py index.html etl.py X

Cinema-vs-Streaming > static > python > etl.py > Jupyter > df_Nextlixsubs

Run Cell | Run Below | Debug Cell

```
1 # %%
2 # import dependencies
3 import pandas as pd
4 import requests
5
6 Run Cell | Run Above
7 # %% [markdown]
8 # ### WEB SCRAPE YEARLY BOX OFFICE
9 # Get the number of releases per year and their average box office
10 Run Cell | Run Above | Debug Cell
11 # %%
12 #Scrape usign request to get html of page
13
14 # Scrape for Boxoffice data and return dataframe
15 def scrape_boxoffice():
16     page = requests.get('https://www.boxofficemojo.com/year/?ref_=bo_nb_di_secondarytab')
17     b = page.content
18
19     # Using Pandas to convert html into dataframe
20     df_list = pd.read_html(b)
21     df = df_list[-1]
22
23
24 # Rename column name to be more accessible
25 df = df.rename(columns={"Total Gross": "Total_gross", "%t LY": "LY", "#1 Release": "No1_release"})
26
27 return df
28
29 # Save JSON to resources folder
30 # df.reset_index().to_json("../Resources/box_office_scrape.json", orient='records')
31
32 #Subscriber data to df
33
34
35 def df_Nextlixsubs():
36     df = pd.read_csv("../Resources/Netflix_subscribers_AP_2013-2020.csv")
37     return df
38
39 def df_Cinemadata():
40     df = pd.read_csv("../Resources/cinema_data.csv")
41     return df
```

ETL

Jinja templates

```
✓ Cinema-vs-Streaming          125    |     up the same level as pre-Covid.</p>
  > Images                         126    |   </div>
  > Resources                      127    |   </div>
  > ss                            128    |   </div>
  ✓ static                         129
    > css                           130
    > data                          131    <script src="https://cdnjs.cloudflare.com/ajax/libs/animejs/2.0.2/anime.min.js"></script>
    > img                           132
    > js                            133    <script src="https://d3js.org/d3.v5.min.js"></script>
    > python                        134    <script src="https://cdnjs.cloudflare.com/ajax/libs/d3-tip/0.7.1/d3-tip.min.js"></script>
    > Resources                     135
    > Resources                     136    <script src="https://cdnjs.cloudflare.com/ajax/libs/d3/5.5.0/d3.js"></script>
    ✓ templates                     137    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
      ◉ cinema.html                138
      ◉ cinemasMap.html            139    <script src="{{ url_for('static', filename='js/app.js') }}></script>
      ◉ index.html                  140
      ◉ streaming.html             141
      ◉ .gitignore                  142    </body>
      ◉ app.py                      143
      ◉ app.py                      144
      ◉ app.py                      145
```

js linking to the database

```
    ↴
    ↵ Cinema-vs-Streaming
    > Images
    > Resources
    > ss
    ↵ static
    > css
    > data
    > img
    ↵ js
      JS app.js
      JS ce_chart.js
      JS CinemaMap.js
      JS config.js
      JS cs_chart.js
      JS ew_chart.js
      JS leaflet.extra-markers.min.js
      JS logic.js
      JS logic1.js
    > python
    > Resources
    ↵ templates
```

```
45           .attr("d", d3.line()
46             .x(d => xTimeScale(d.Year))
47             .y(d => yLinearScale(d[selectedgroup])))
48             .attr("stroke", "blue");
49         return valueline;
50     }
51
52
53 // Import data from an external CSV file
54 d3.json("/ce_data").then(function (cinemadata) {
55   console.log(cinemadata);
56   var parseTime = d3.timeParse('XY');
57
58   cinemadata.forEach(function (data) {
59     data.Year = parseTime(data.Year);
60     data.Theatres = +data.Theatres;
61     data.Screens = +data.Screens;
62     data.Seats = +data.Seats
63   });
64
65
66 // Create scaling functions
67 var xTimeScale = d3.scaleTime()
68   .domain(d3.extent(cinemadata, d => d.Year))
```

CSS file - styling

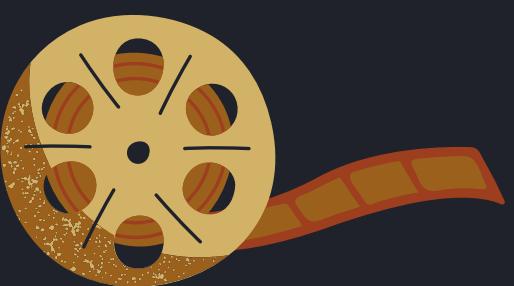
```
# style.css  X  
Cinema-vs-Streaming > static > css > # style.css > axisText  
1  body{  
2    background-color: black;  
3  }  
4  
5  .container{  
6    background-color: white;  
7    margin-top: 50px;  
8  }  
9  
10 p {  
11   color: rgb(78, 93, 119);  
12   font-family: sans-serif;  
13   font-size: 20px;  
14   text-align: justify;  
15   margin-left: 30px;  
16 }  
17 .btn-secondary{  
18   background-color: #a0353a;  
19   color: white;  
20 }  
21 .card-title{  
22   color: rgb(78, 93, 119);  
23   font-family: sans-serif;  
24   font-size: 20px;  
25   font-weight: bold;  
26   text-align: justify;  
27 }  
28 .card-text{  
29   color: rgb(78, 93, 119);  
30   font-family: sans-serif;  
31   font-size: 20px;  
32   text-align: justify;  
33 }
```

CINEMA FOOTING OVER TIMES (LEAFLET)

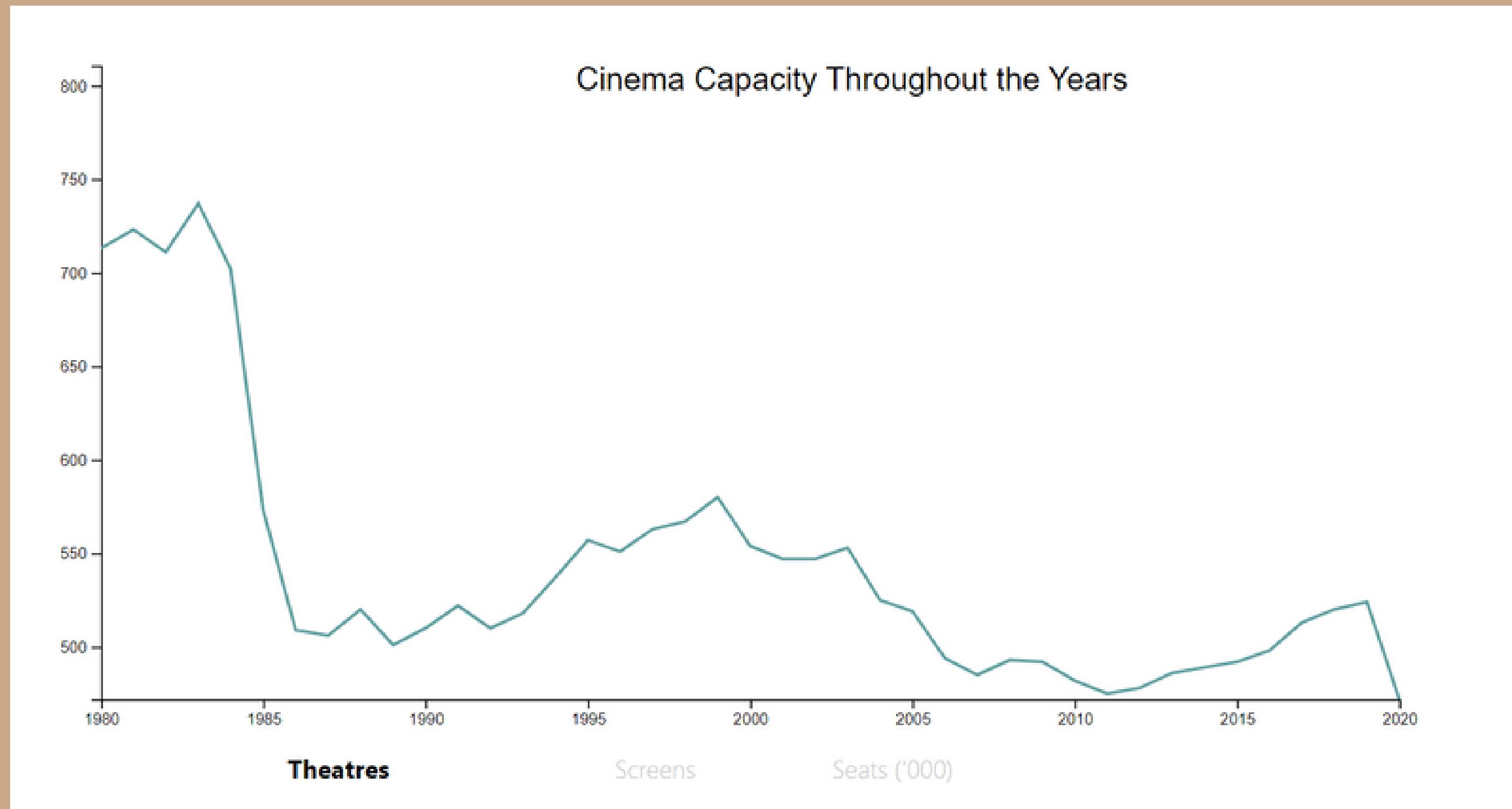


Stay Tune ... This is coming soon

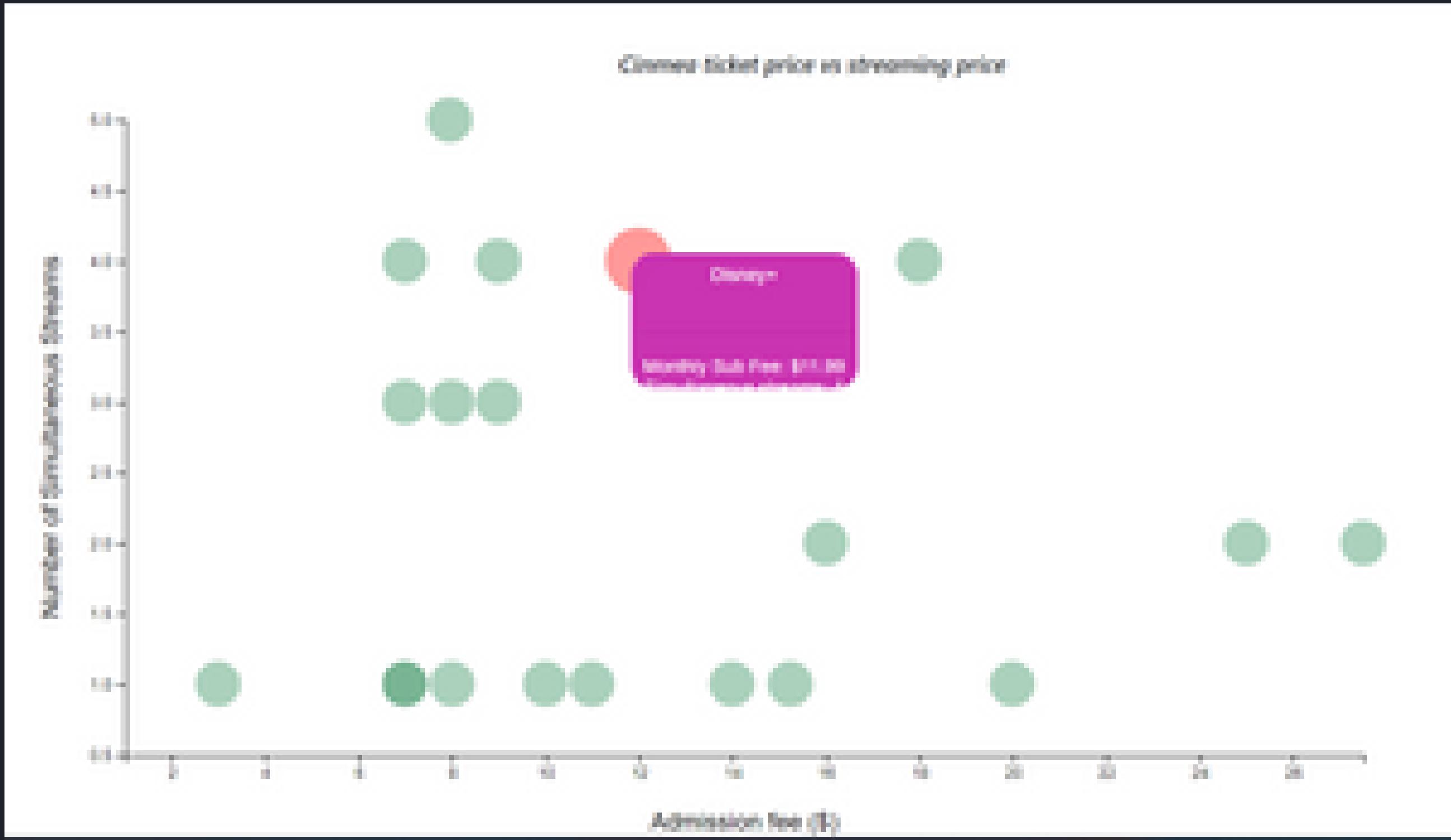
**G E O
M A P P I N G**



CINEMA SCREENS / SEATING CAPACITY

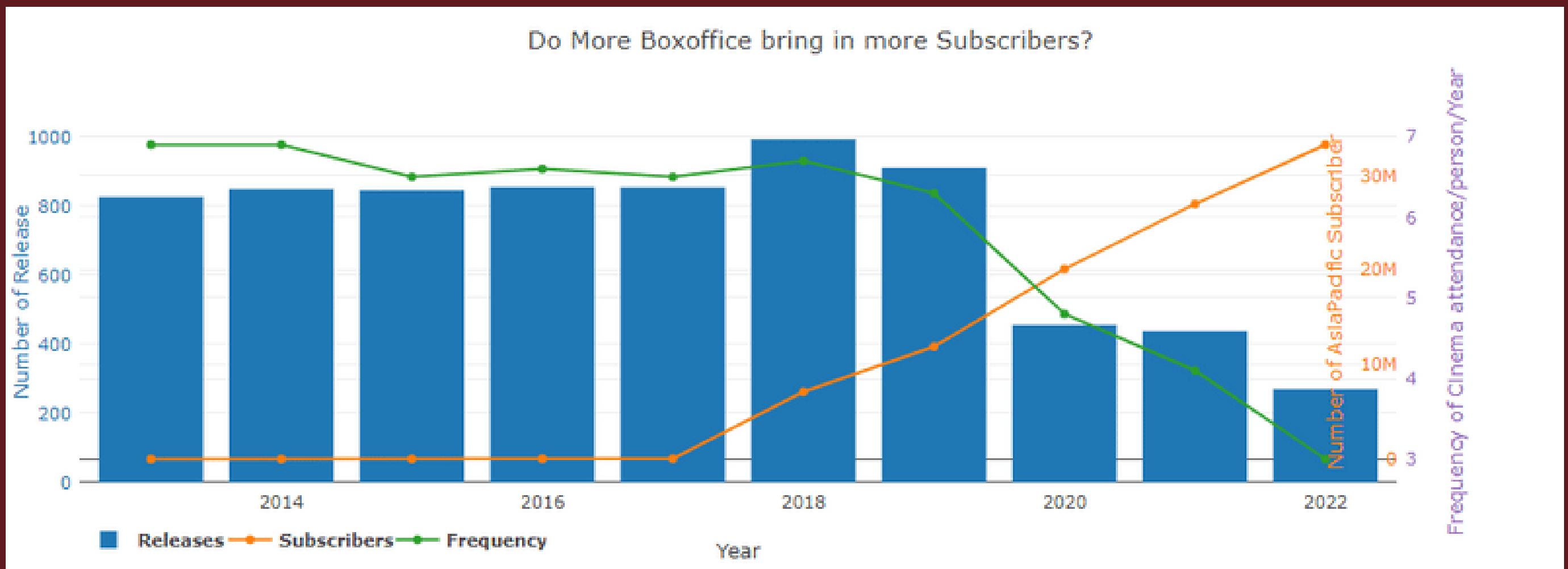


CINEMA VS STREAMING IN PRICING





BOX OFFICE RELEASE / SUBSCRIBERS NUMBERS CINEMA ATTENDENCE FREQ



Subscriber Number

Netflix subscriber number has up 550% from 34m 2013 to 221m in 2022 (as of March)

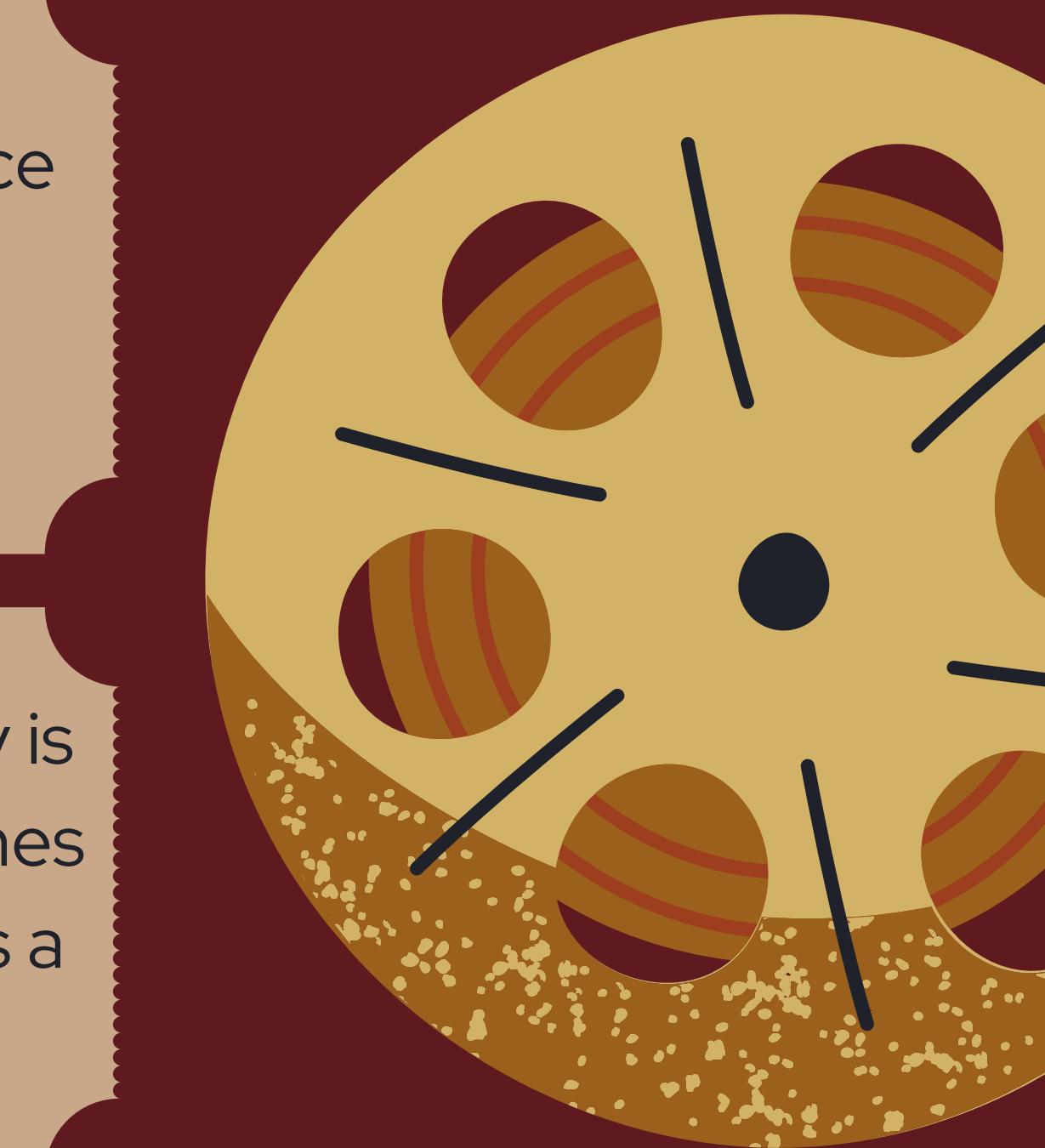
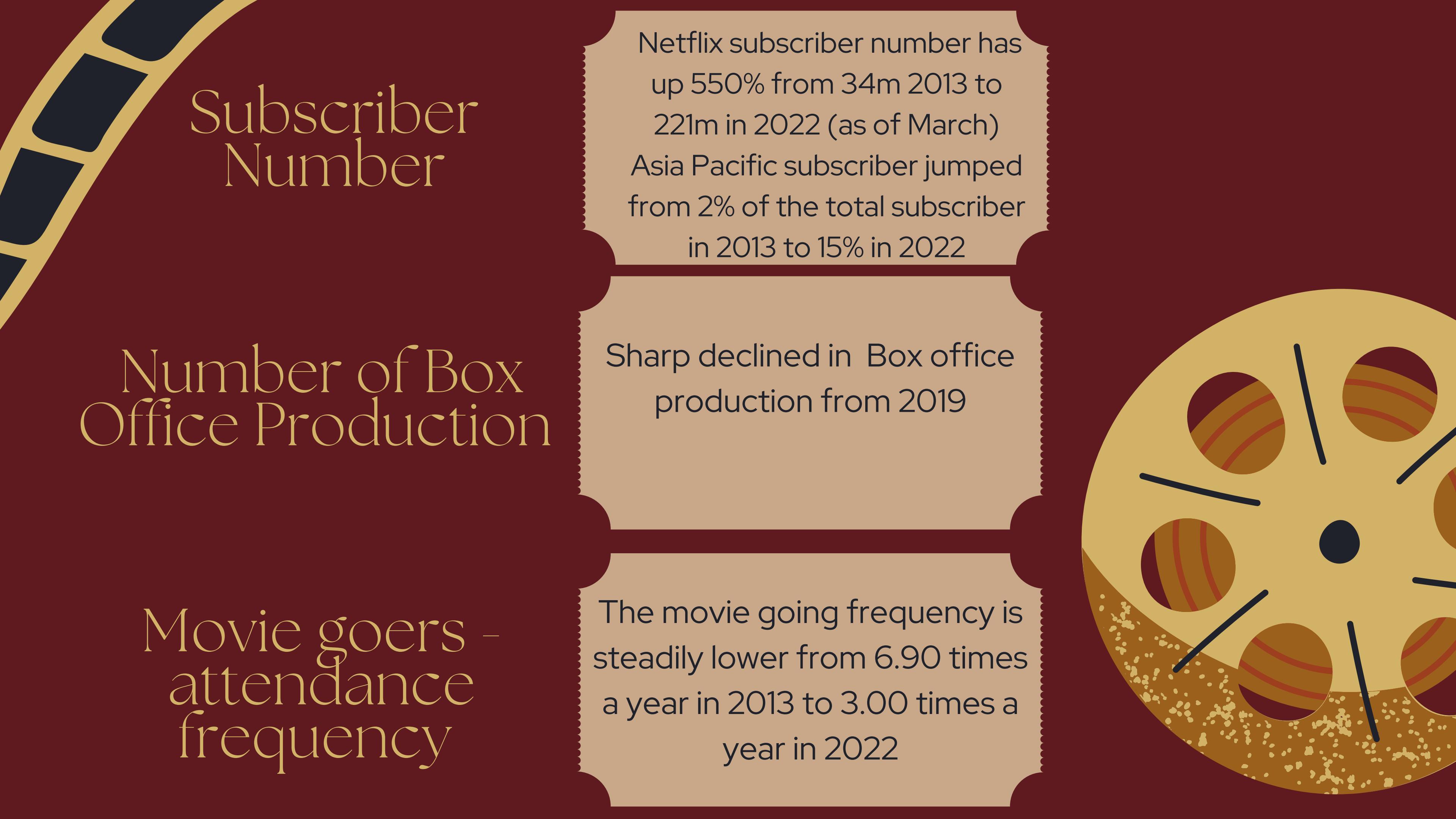
Asia Pacific subscriber jumped from 2% of the total subscriber in 2013 to 15% in 2022

Number of Box Office Production

Sharp declined in Box office production from 2019

Movie goers - attendance frequency

The movie going frequency is steadily lower from 6.90 times a year in 2013 to 3.00 times a year in 2022



CINEMA

- Number of movie-goers remain relatively similar even though the streaming numbers are growing
- Data shows that movie-goers continue to see movies in cinema, the frequency was less due to COVID related restriction

STREAMING PROVIDER

- Market share will be fragmented as more new providers available
- choices will drive the streaming fee downward
- More free and on-demand streaming provide alternative entertainment





A stage set featuring red and gold striped curtains on either side. In the lower-left foreground, a searchlight is directed towards the center stage. The background is a light beige color with a pattern of hanging red crescent moons and stars.

THANK YOU FOR LISTENING

Don't ask any questions!