

[Home](#)[Database](#)[Flask](#)[HTML](#)[Java](#)[Mapping](#)

# Netflix

# Marketing research



Continue to next  
episode



# I TABLE OF CONTENTS

01 INTRODUCTION

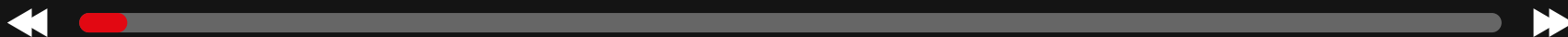
02 DATABASE

03 FLASK

04 JAVA

05 HTML

06 CONCLUSION



[Home](#)

[Database](#)

[Flask](#)

[HTML](#)

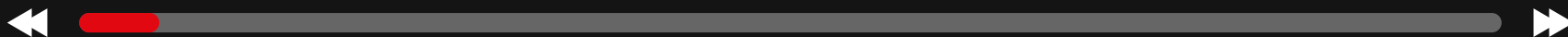
[Java](#)

[Mapping](#)



# 01

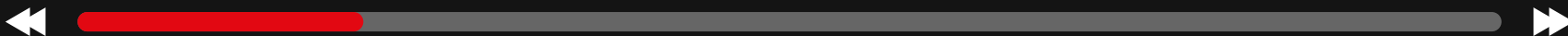
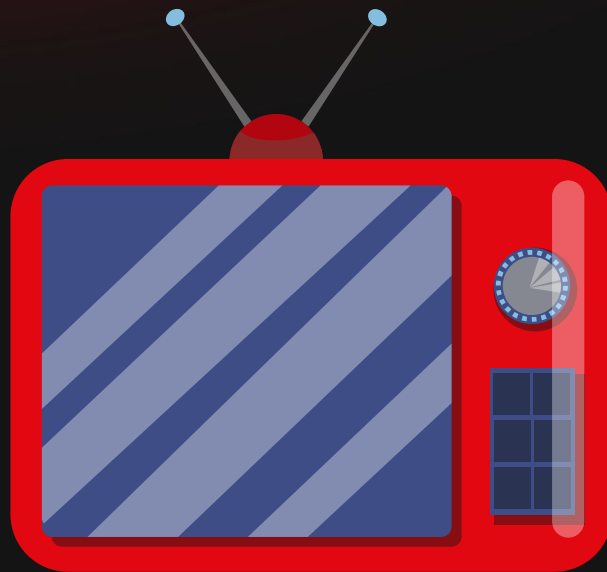
# INTRODUCTION

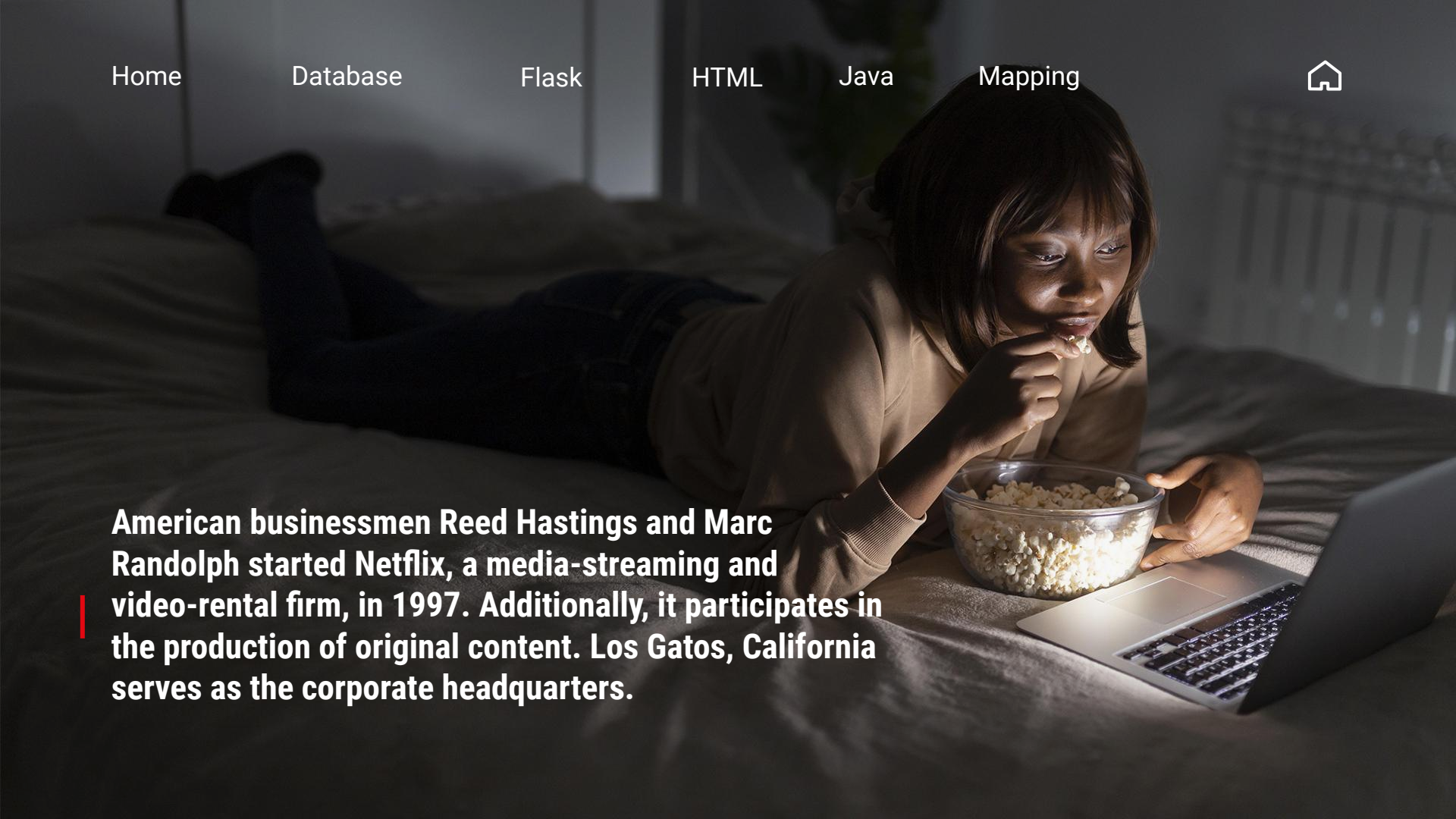




# WHAT IS NETFLIX?

With more than 70 million subscribers in more than 190 countries watching more than 125 million hours of TV shows and movies every day, including original series, documentaries, and feature films, Netflix is the most popular Internet television network in the world. On almost any screen that is linked to the Internet, members can watch as much as they want, whenever and wherever. Without interruptions or obligations, members can play, pause, and resume watching at any time.



A woman with dark hair is lying on her stomach on a bed, watching a laptop. She is eating popcorn from a clear bowl. The room is dimly lit, with the light from the laptop screen illuminating her face and the popcorn. The background shows a bed with white linens and a headboard.[Home](#)[Database](#)[Flask](#)[HTML](#)[Java](#)[Mapping](#)

**American businessmen Reed Hastings and Marc Randolph started Netflix, a media-streaming and video-rental firm, in 1997. Additionally, it participates in the production of original content. Los Gatos, California serves as the corporate headquarters.**



# | When did they go live?

On January 6, 2016, Netflix debuted its service worldwide in Las Vegas, expanding its Internet TV network to more than 130 new nations. Since launching its streaming service in 2007, Netflix has reached 60 different nations by first entering Canada, then Latin America, Europe, Australia, New Zealand, and Japan.



[Home](#)

[Database](#)

[Flask](#)

[HTML](#)

[Java](#)

[Mapping](#)



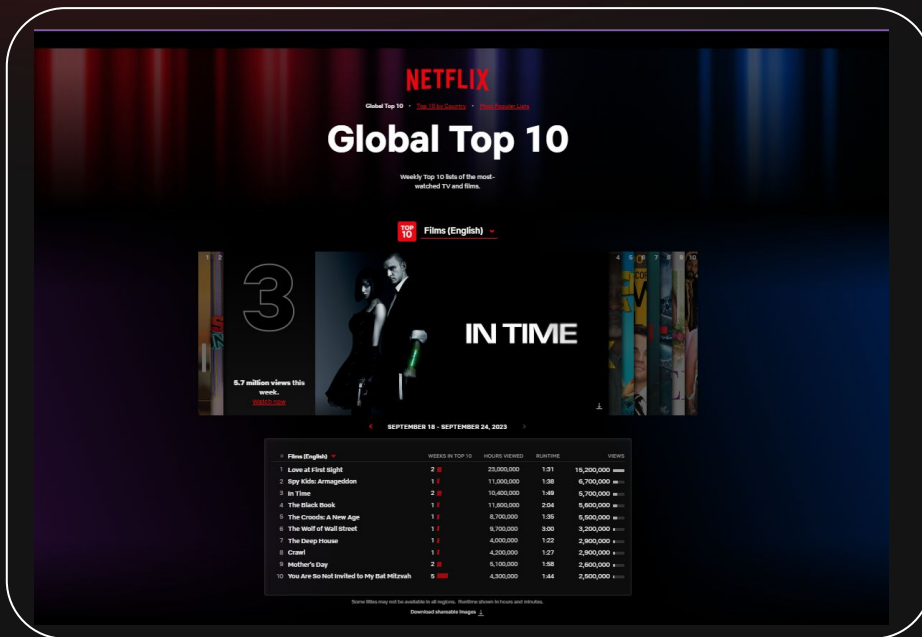
# DATABASES 02



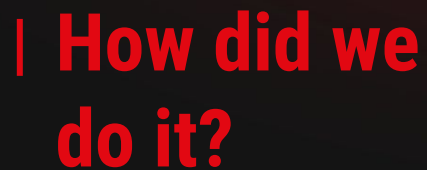


# NETFLIX TUDUM

The official Netflix companion website, Tudum, provides information to viewers about the shows and movies they enjoy. Exclusive interviews, behind-the-scenes material, additional movies, and more are all available on Tudum. Additionally, if you sign in using your Netflix credentials, you will have a more tailored experience based on the Netflix series and films that you have recently rated or viewed.







[Home](#)

[Database](#)

[Flask](#)

[HTML](#)

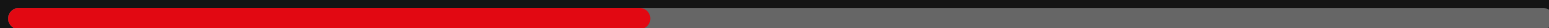
[Java](#)

[Mapping](#)



# 03

# FLASK





# | Flask



## What is Flask?

A Python web application framework called Flask makes it simple to create web applications. It was created by Armin Ronacher, who served as the team leader of Pocco. Its core is compact and simple to extend; it's a microframework without an object relationship manager or similar capabilities. It does have a lot of great features, including a template engine and url routing. It is a web app framework for WSGI.



## What is a Web Framework

Web application developers can design apps without having to be concerned about low-level aspects like protocol, thread management, and other issues thanks to a web framework, also known as a web application framework.





## WSGI

For the creation of Python web applications, the Web Server Gateway Interface (WSGI) has been the de facto standard. The WSGI specification describes a standard interface for web servers and online applications.



## jinja2

A well-liked Python template engine is jinja2. A web template system renders a dynamic web page by fusing a template with a particular data source.



## Werkzeug

Requests, response objects, and utility functions are all implemented by the WSGI toolkit known as Werkzeug. On top of it can now be created a web frame. Werkzeug serves as one of the foundations of the Flask framework.



## Microframework

Many people refer to Flask as a microframework. It is intended to keep the application's core simple and scalable. Instead of using an abstraction layer to enable databases, Flask allows extensions to the application to add these features.



[Home](#)

[Database](#)

[Flask](#)

[HTML](#)

[Java](#)

[Mapping](#)



**LET'S TAKE A TRIP TO THE CODE**



# I How did we do it?

```

1  import numpy as np
2  import sqlalchemy
3  from sqlalchemy.ext.automap import automap_base
4  from sqlalchemy.orm import Session
5  from sqlalchemy import create_engine, func
6  from flask import Flask, jsonify
7  from datetime import datetime
8  from flask_cors import CORS
9  import pandas as pd
10
11 # python -m http.server [PORT]
12
13 #####
14 # Database Setup
15 #####
16 engine = create_engine("sqlite:///Netflix_DB.sqlite")
17
18 # reflect an existing database into a new model
19 Base = automap_base()
20 # reflect the table
21 Base.prepare(autoload_with=engine)
22
23 # Save reference to the table
24 global1 = Base.classes.global1
25 canada = Base.classes.canada
26 united_kingdom = Base.classes.united_kingdom
27 australia = Base.classes.australia
28 egypt = Base.classes.egypt
29 countries = Base.classes.countries
30
31 #####
32 # Flask Setup
33 #####
34 app = Flask(__name__)
35
36 Config(app, resources={"api/": {"origins": "http://127.0.0.1:8085"}})
37
38 #####
39 # Flask Routes
40 #####
41
42 @app.route("/")
43 def welcome():
44     """List all available api routes."""
45     return (
46         f"Available Routes: {chr(10)}<br/>"
47         f"/api/v1.0/global1<br/>"
48         f"/api/v1.0/global2<br/>"
49         f"/api/v1.0/canada<br/>"
50         f"/api/v1.0/united_kingdom<br/>"
51         f"/api/v1.0/australia<br/>"
52         f"/api/v1.0/egypt<br/>"
53         f"/api/v1.0/countries"
54     )

```

```

54
55 @app.route("/api/v1.0/global1")
56 def global1_func():
57     # Create our session (link) from Python to the DB
58     session = Session(engine)
59
60     """Return a list of global1 data"""
61     # Query all passengers
62     results = session.query(global1.No, global1.movie, global1.week_at_Top10, global1.hours_seen, global1.duration, global1.views).all()
63
64     session.close()
65
66     # Create a dictionary from the row data and append to a list of all_passengers
67     all_data = []
68     for No, movie, week_at_Top10, hours_seen, duration, views in results:
69         data_dict = {}
70         data_dict["No"] = No
71         data_dict["movie"] = movie
72         data_dict["week_at_Top10"] = week_at_Top10
73         data_dict["hours_seen"] = round(hours_seen, 2)
74         data_dict["duration"] = str(duration)
75         data_dict["views"] = round(views, 2)
76         all_data.append(data_dict)
77
78     return jsonify(all_data)
79
80 @app.route("/api/v1.0/canada")
81 def canada_func():
82     # Create our session (link) from Python to the DB
83     session = Session(engine)
84     """Return a list of the selected country data"""
85     # Query all passengers
86     results = session.query(canada.No, canada.movie, canada.week_at_Top10).all()
87
88     session.close()
89
90     # Create a dictionary from the row data and append to a list of all_passengers
91     all_data = []
92     for No, movie, week_at_Top10 in results:
93         data_dict = {}
94         data_dict["No"] = No
95         data_dict["movie"] = movie
96         data_dict["week_at_Top10"] = week_at_Top10
97         all_data.append(data_dict)
98
99     return jsonify(all_data)
100
101 @app.route("/api/v1.0/united_kingdom")
102 def united_func():
103     # Create our session (link) from Python to the DB
104     session = Session(engine)
105     """Return a list of the selected country data"""
106     # Query all passengers
107     results = session.query(united_kingdom.No, united_kingdom.movie, united_kingdom.week_at_Top10).all()

```







# How did we do it?

```

110 session.close()
111
112 # Create a dictionary from the row data and append to a list of all_passengers
113 all_data = []
114 for No, movie, week_at_Top10 in results:
115     data_dict = {}
116     data_dict['No'] = No
117     data_dict['movie'] = movie
118     data_dict['week_at_Top10'] = week_at_Top10
119     all_data.append(data_dict)
120
121 return jsonify(all_data)
122
123 @app.route("/api/v1.0/egypt")
124 def egypt_func():
125     # Create our session (link) from Python to the DB
126     session = Session(engine)
127     """Return a list of the selected country data"""
128     # Query all passengers
129     results = session.query(egypt.No, egypt.movie, egypt.week_at_Top10).all()
130
131     session.close()
132
133     # Create a dictionary from the row data and append to a list of all_passengers
134     all_data = []
135     for No, movie, week_at_Top10 in results:
136         data_dict = {}
137         data_dict['No'] = No
138         data_dict['movie'] = movie
139         data_dict['week_at_Top10'] = week_at_Top10
140         all_data.append(data_dict)
141
142     return jsonify(all_data)
143
144
145 @app.route("/api/v1.0/australia")
146 def australia_func():
147     # Create our session (link) from Python to the DB
148     session = Session(engine)
149     """Return a list of the selected country data"""
150     # Query all passengers
151     results = session.query(australia.No, australia.movie, australia.week_at_Top10).all()
152
153     session.close()
154
155     # Create a dictionary from the row data and append to a list of all_passengers
156     all_data = []
157     for No, movie, week_at_Top10 in results:
158         data_dict = {}
159         data_dict['No'] = No
160         data_dict['movie'] = movie
161         data_dict['week_at_Top10'] = week_at_Top10
162         all_data.append(data_dict)

```

```

145 @app.route("/api/v1.0/australia")
146 def australia_func():
147     # Create our session (link) from Python to the DB
148     session = Session(engine)
149     """Return a list of the selected country data"""
150     # Query all passengers
151     results = session.query(australia.No, australia.movie, australia.week_at_Top10).all()
152
153     session.close()
154
155     # Create a dictionary from the row data and append to a list of all_passengers
156     all_data = []
157     for No, movie, week_at_Top10 in results:
158         data_dict = {}
159         data_dict['No'] = No
160         data_dict['movie'] = movie
161         data_dict['week_at_Top10'] = week_at_Top10
162         all_data.append(data_dict)
163
164     return jsonify(all_data)
165
166
167 @app.route("/api/v1.0/global")
168 def global_func():
169     # Create our session (link) from Python to the DB
170     session = Session(engine)
171
172     """Return a list of global data"""
173     # Query all passengers
174     results = session.query(global.No, global.movie, global.week_at_Top10, global.hours_seen, global.duration, global.views).all()
175
176     session.close()
177
178     # Create a dictionary from the row data and append to a list of all_passengers
179     all_data = []
180     df = pd.DataFrame(results)
181     # Group by "Movie Name" and Find the maximum "Week at Top 10 Score" for each movie
182     max_week_scores = df.groupby('movie')['week_at_Top10'].max().reset_index()
183     max_week_scores = max_week_scores.sort_values('week_at_Top10', ascending=False)
184     # Merge the original DataFrame with the maximum week scores to get the desired row
185     i=1
186     for index, row in max_week_scores.iterrows():
187         result_dict = {}
188         result_dict = row.to_dict()
189         result_dict['No'] = i
190         all_data.append(result_dict)
191         i+=1
192     return jsonify(all_data)
193
194 if __name__ == '__main__':
195     app.run(debug=True)

```

[Home](#)

[Database](#)

[Flask](#)

[HTML](#)

[Java](#)

[Mapping](#)



# 04 **J**A**V**A







# | What is Java?

Web developers frequently employ the lightweight programming language JavaScript to include dynamic interactions in web pages, applications, servers, and even video games.

- It functions in perfect harmony with HTML and CSS, completing CSS's role of formatting HTML components while also enabling user interaction, which CSS by itself is unable to do.
- JavaScript is a useful language to master because of how frequently it is used in game, mobile app, and online development.



[Home](#)

[Database](#)

[Flask](#)

[HTML](#)

[Java](#)

[Mapping](#)



Home

Database

Flask

HTML

Java

Mapping



# How did we do it?

```

1 document.addEventListener("DOMContentLoaded", function () {
2   const globalTab = document.getElementById("global-tab");
3   const canadaTab = document.getElementById("canada-tab");
4   const egyptTab = document.getElementById("egypt-tab");
5   const australiaTab = document.getElementById("australia-tab"); // New tab
6   const movieTable = document.getElementById("movie-data");
7   const nextButton = document.getElementById("next-button");
8   const previousButton = document.getElementById("previous-button");
9   const barChartDiv = document.getElementById("bar-chart");
10  const movieTable2 = document.getElementById("movie-data2");
11  const nextButton2 = document.getElementById("next-button2");
12  const previousButton2 = document.getElementById("previous-button2");
13
14  let currentPage = 0; // Current page for pagination
15  const pageSize = 10; // Number of rows per page
16  let currentTabId = "global"; // Initialize with the default tab
17  let currentTabId2 = "global2"; // Initialize with the default tab
18
19  globalTab.addEventListener("click", () => loadMovieData("global"));
20  canadaTab.addEventListener("click", () => loadMovieData("canada"));
21  egyptTab.addEventListener("click", () => loadMovieData("egypt"));
22  australiaTab.addEventListener("click", () => loadMovieData("australia")); // New tab
23  nextButton.addEventListener("click", () => nextPage());
24  previousButton.addEventListener("click", () => previousPage());
25  nextButton2.addEventListener("click", () => nextPage2());
26  previousButton2.addEventListener("click", () => previousPage2());
27
28  function loadMovieData(apiEndpoint) {
29    currentTabId = apiEndpoint; // Update the current tab
30    // Make a request to your API endpoint using fetch or jQuery.ajax
31    fetch(`http://127.0.0.1:5000/api/v1.0/${apiEndpoint}`)
32      .then((response) => response.json())
33      .then((data) => {
34        // Clear previous data
35        movieTable.innerHTML = "";
36
37        // Calculate the starting and ending index for the current page
38        const startIndex = currentPage * pageSize;
39        const endIndex = startIndex + pageSize;
40
41        // Get data for the current page
42        const currentPageData = data.slice(startIndex, endIndex);
43
44        // Populate the table with the retrieved data for the current page
45        for (let i = 0; i < currentPageData.length; i++) {
46          const movie = currentPageData[i];
47          const row = document.createElement("tr");
48          row.innerHTML = `
49            <td>${movie.No}</td>
50            <td>${movie.title}</td>
51            <td>${movie.week_at_top10}</td>

```

```

52            <td>${movie.hours_seen}</td>
53            <td>${movie.duration}</td>
54            <td>${movie.scores}</td>
55          `;
56          movieTable.appendChild(row);
57        }
58
59        // Update the bar chart with the current page data
60        updateBarChart(currentPageData);
61      });
62    .catch((error) => {
63      console.error("Error fetching data", error);
64    });
65  }
66
67  function nextPage() {
68    currentPage++;
69    // Load data for the current tab with pagination
70    loadMovieData(currentTabId);
71  }
72
73  function previousPage() {
74    if (currentPage > 0) {
75      currentPage--;
76      // Load data for the current tab with pagination
77      loadMovieData(currentTabId);
78    }
79  }
80
81  function updateBarChart(data) {
82    const weekAtTop10 = data.map((movie) => movie.week_at_top10);
83    const movieNames = data.map((movie) => movie.movie);
84
85    const chartData = [
86      { movieNames: movieNames,
87        y: weekAtTop10,
88        type: 'bar' },
89    ];
90
91    const layout = {
92      title: 'Weeks at Top 10',
93      xaxis: { title: 'Movie' },
94      yaxis: { title: 'Weeks' },
95    };
96
97    // Use Plotly.js to update the chart
98    Plotly.react(chartDiv, chartData, layout);
99  }
100
101  // Load data for the default tab (e.g., "global")
102  loadMovieData("global");
103
104  //
105
106  function loadTop10(apiEndpoint) {
107    currentTabId2 = apiEndpoint; // Update the current tab
108    // Make a request to your API endpoint using fetch or jQuery.ajax

```

```

109    fetch(`http://127.0.0.1:5000/api/v1.0/${apiEndpoint2}`)
110      .then((response) => response.json())
111      .then((data) => {
112        // Clear previous data
113        movieTable2.innerHTML = "";
114
115        // Calculate the starting and ending index for the current page
116        const startIndex2 = currentPage2 * pageSize;
117        const endIndex2 = startIndex2 + pageSize;
118
119        // Get data for the current page
120        const currentPage2Data = data.slice(startIndex2, endIndex2);
121
122        // Populate the table with the retrieved data for the current page
123        for (let i = 0; i < currentPage2Data.length; i++) {
124          const movie = currentPage2Data[i];
125          const row = document.createElement("tr");
126          row.innerHTML = `
127            <td>${movie.No}</td>
128            <td>${movie.movie}</td>
129            <td>${movie.week_at_top10}</td>
130
131            `;
132          movieTable2.appendChild(row);
133        }
134      });
135    .catch((error) => {
136      console.error("Error fetching data2", error);
137    });
138  }
139
140  function nextPage2() {
141    currentPage2++;
142    // Load data for the current tab with pagination
143    loadTop10(apiEndpoint2);
144  }
145
146  function previousPage2() {
147    if (currentPage2 > 0) {
148      currentPage2--;
149      // Load data for the current tab with pagination
150      loadTop10(apiEndpoint2);
151    }
152  }
153
154  // Load data for the default tab (e.g., "global2")
155  loadTop10("global2");
156
157  //
158
159  //
160
161  //

```



[Home](#)

[Database](#)

[Flask](#)

[HTML](#)

[Java](#)

[Mapping](#)



# HTML 05

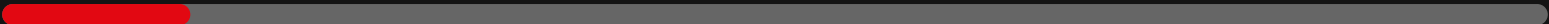


# | What is HTML



## HyperText Markup Language

It is a common markup language used to create web pages. Using HTML components, such as tags and attributes, it enables the development and structuring of sections, paragraphs, and links.





# | HTML has a lot of use cases, namely:



## Web development

To control how text, hyperlinks, and media files are displayed by browsers, developers employ HTML code.



## Internet navigation

Since HTML is widely used to embed hyperlinks, users may navigate and insert links between relevant pages and websites with ease.



## Web documentation

Similar to Microsoft Word, HTML allows for document organization and formatting.





## | What is CSS?

CSS, or Cascading Style Sheets, is a markup language that is used to style elements in markup languages like HTML. It divides the website's visual design from its content. Since HTML serves as a site's very foundation and CSS handles all of the aesthetics for a whole website, the two are closely related.





[Home](#)

[Database](#)

[Flask](#)

[HTML](#)

[Java](#)

[Mapping](#)

A group of people sitting on a couch in a dimly lit living room, watching a television. A man in the foreground is holding a remote control. There are beer bottles and a bowl of popcorn on a table in front of them. The text 'LET'S TAKE A TRIP TO THE CODE' is overlaid at the bottom.

**LET'S TAKE A TRIP TO THE CODE**





# How did we do it?

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Top 10 Movie Dashboard NETFLIX</title>
7   <link rel="stylesheet" href="styles2.css">
8 </head>
9 <body>
10   <div class="scroll-container">
11     <div class="header">
12       <h1>Top 10 Movies NETFLIX</h1>
13     <div class="tabs">
14       <button id="global-tab">Global</button>
15       <button id="canada-tab">Canada</button>
16       <button id="egypt-tab">Egypt</button>
17       <button id="uk-tab">United Kingdom</button> <!-- New tab -->
18       <button id="australia-tab">Australia</button> <!-- New tab -->
19     </div>
20   </div>
21   <div class="main-container">
22     <table id="movie-table">
23       <thead>
24         <tr>
25           <th>No</th>
26           <th>Movie</th>
27           <th>Weeks at Top 10</th>
28           <th>Hours Seen</th>
29           <th>Duration</th>
30           <th>Views</th>
31         </tr>
32       </thead>
33       <tbody id="movie-data">
34         <!-- Data will be populated here -->
35       </tbody>
36     </table>
37     <button id="previous-button">Next week</button>
38     <button id="next-button">Previous week</button>
39     <div id="bar-chart">
40       <!-- h2 class="chart-title">Top 10</h2 -->
41       <div id="plotly-chart"></div>
42     </div>
43     <table id="movie-table2">
44       <thead>

```

```

24       <tr>
25         <th>No</th>
26         <th>Movie</th>
27         <th>Weeks at Top 10</th>
28         <th>Hours Seen</th>
29         <th>Duration</th>
30         <th>Views</th>
31       </tr>
32     </thead>
33     <tbody id="movie-data">
34       <!-- Data will be populated here -->
35     </tbody>
36   </table>
37   <button id="previous-button">Next week</button>
38   <button id="next-button">Previous week</button>
39   <div id="bar-chart">
40     <!-- h2 class="chart-title">Top 10</h2 -->
41     <div id="plotly-chart"></div>
42   </div>
43   <table id="movie-table2">
44     <thead>
45       <tr>
46         <th>No</th>
47         <th>Movie</th>
48         <th>Weeks at Top 10</th>
49       </tr>
50     </thead>
51     <tbody id="movie-data2">
52       <!-- Data will be populated here -->
53     </tbody>
54   </table>
55   <button id="previous-button2">Next week2</button>
56   <button id="next-button2">Previous week2</button>
57 </div>
58 </div>
59 </div>
60 <script src="https://d3js.org/d3.v5.min.js"></script>
61 <script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.7.1/leaflet.js"></script>
62 <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
63 <script src="/javascript/headFlaskdata6.0.js"></script> <!-- Tu archivo de JavaScript personalizado -->
64 </body>
65 </html>

```



[Home](#)[Database](#)[Flask](#)[HTML](#)[Java](#)[Mapping](#)

# 06

## Conclusion



[Home](#)

[Database](#)

[Flask](#)

[HTML](#)

[Java](#)

[Mapping](#)

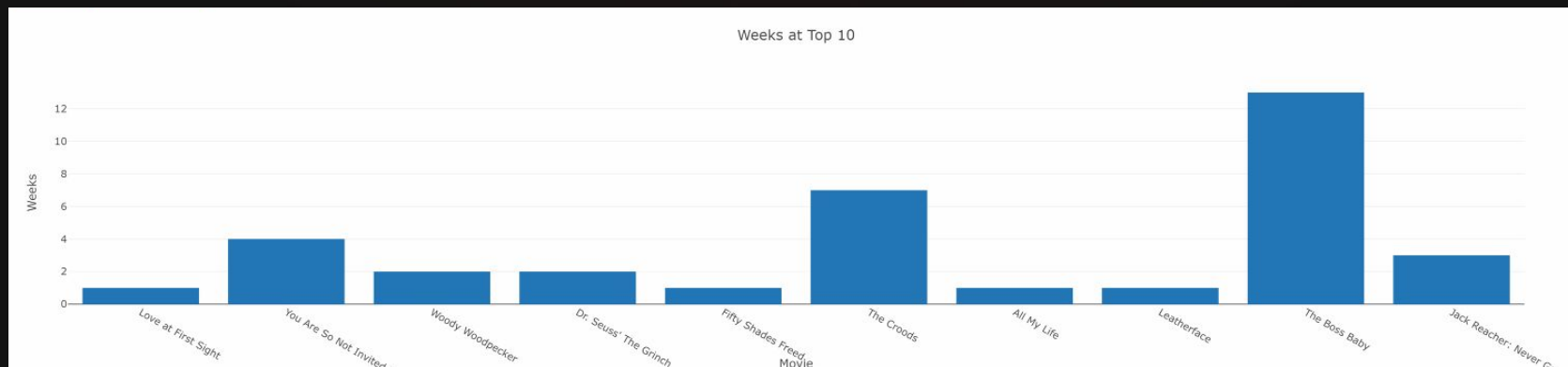
A photograph of four young adults sitting on a grey couch in a living room. From left to right: a man in a green sweater pointing forward with an open mouth, a man in a white t-shirt holding a remote control, a woman in a dark patterned sweater looking forward, and a man in a light blue shirt gesturing with his hand. In front of them is a glass coffee table with various snacks, drinks, and a green bottle. The background shows a bookshelf with books and decorative items.

**LET'S TAKE A TRIP TO THE FOUND RESULTS**

[Home](#)[Database](#)[Flask](#)[HTML](#)[Java](#)[Mapping](#)

No	Movie	Weeks at Top 10	Hours Seen	Duration	Views
1	Love at First Sight	1	21400000.00	01:31:00	14100000.00
2	You Are So Not Invited to My Bat Mitzvah	4	7400000.00	01:44:00	4300000.00
3	Woody Woodpecker	2	5300000.00	01:31:00	3500000.00
4	Dr. Seuss' The Grinch	2	4900000.00	01:26:00	3400000.00
5	Fifty Shades Freed	1	5900000.00	01:45:00	3400000.00
6	The Croods	7	5200000.00	01:38:00	3200000.00
7	All My Life	1	4600000.00	01:32:00	3000000.00
8	Leatherface	1	4300000.00	01:28:00	2900000.00
9	The Boss Baby	13	4600000.00	01:38:00	2800000.00
10	Jack Reacher: Never Go Back	3	5200000.00	01:58:00	2600000.00



[Home](#)[Database](#)[Flask](#)[HTML](#)[Java](#)[Mapping](#)

[Home](#)[Database](#)[Flask](#)[HTML](#)[Java](#)[Mapping](#)

No	Movie	Weeks at Top 10
1	Red Notice	14
2	The Boss Baby	13
3	Despicable Me 2	11
4	Sing	9
5	Sing 2	9
6	The Adam Project	8
7	Don't Look Up	8
8	Sonic the Hedgehog	8
9	Glass Onion: A Knives Out Mystery	7
10	Back to the Outback	7



[Home](#)

[Database](#)

[Flask](#)

[HTML](#)

[Java](#)

[Mapping](#)



# THANK YOU





# BIBLIOGRAPHY

S., A. (2023, August 25). What is HTML? hypertext markup language basics explained. Hostinger Tutorials. <https://www.hostinger.com/tutorials/what-is-html>

S., A. (2023, August 25). What Is CSS and How Does It Work? <https://www.hostinger.com/tutorials/what-is-css>

S., A. (2023, August 25). What Is Java: The Beginner's Guide to the Java Programming Language. <https://blog.hubspot.com/website/what-is-java>

Kariuki, (2023, August 25). How and When Did Netflix Start? A Brief History of the Company. <https://www.makeuseof.com/how-when-netflix-start-brief-company-history/>

