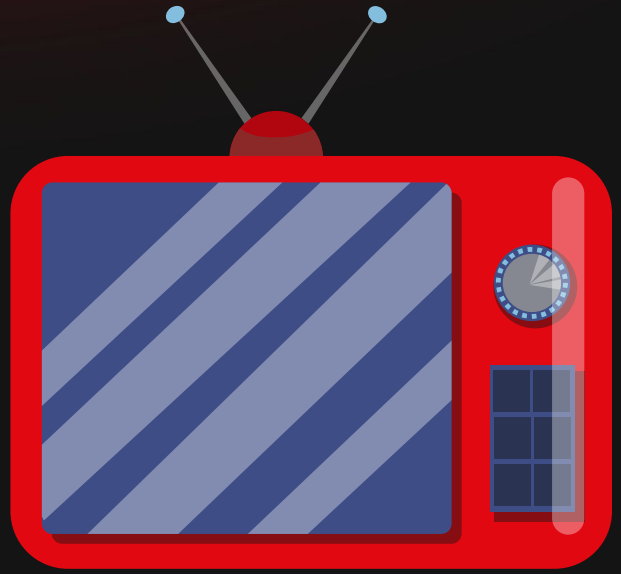
Netflix Marketing research

Continue to next episode

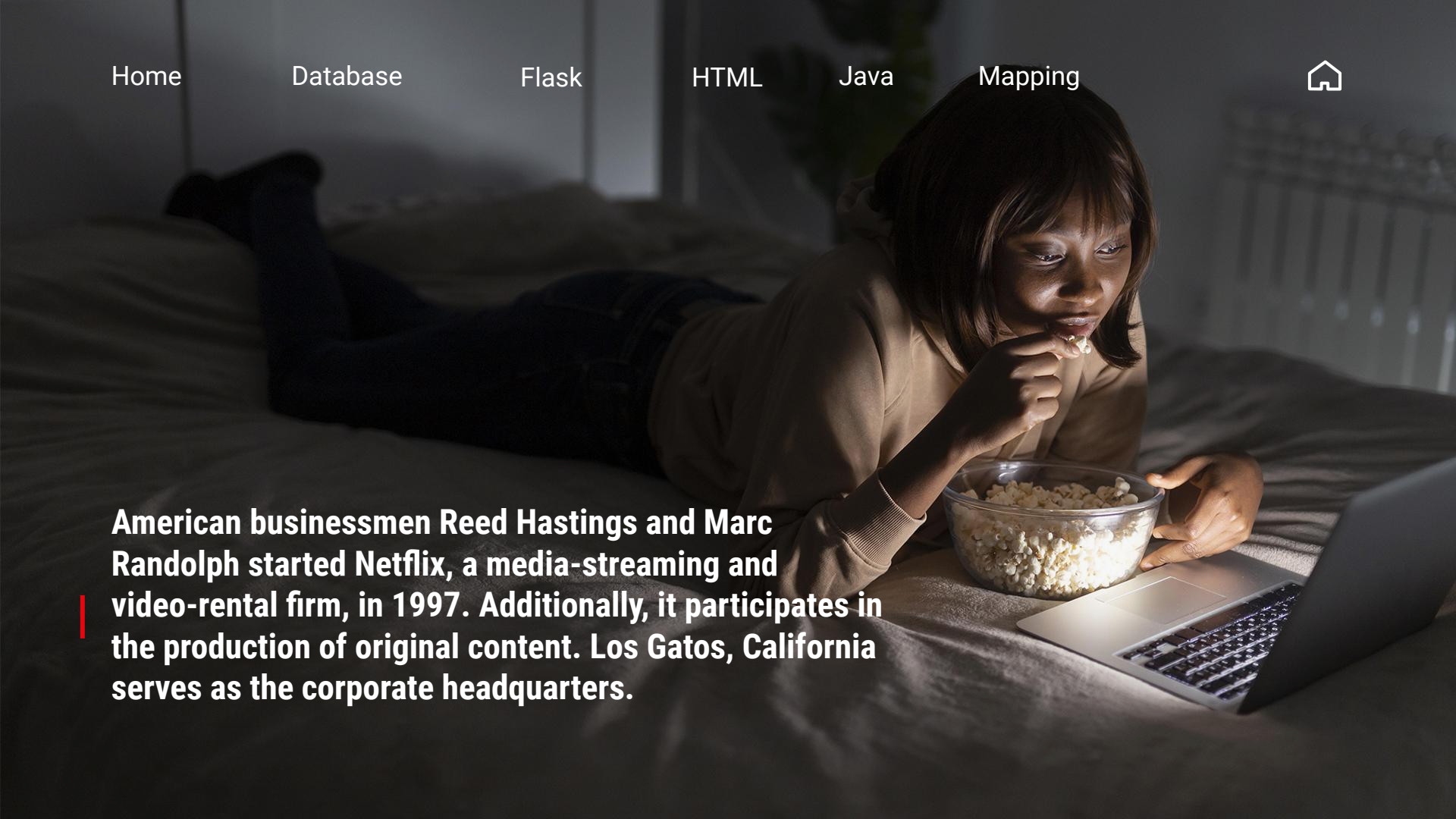
TABLE OF CONTENTS

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.



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?

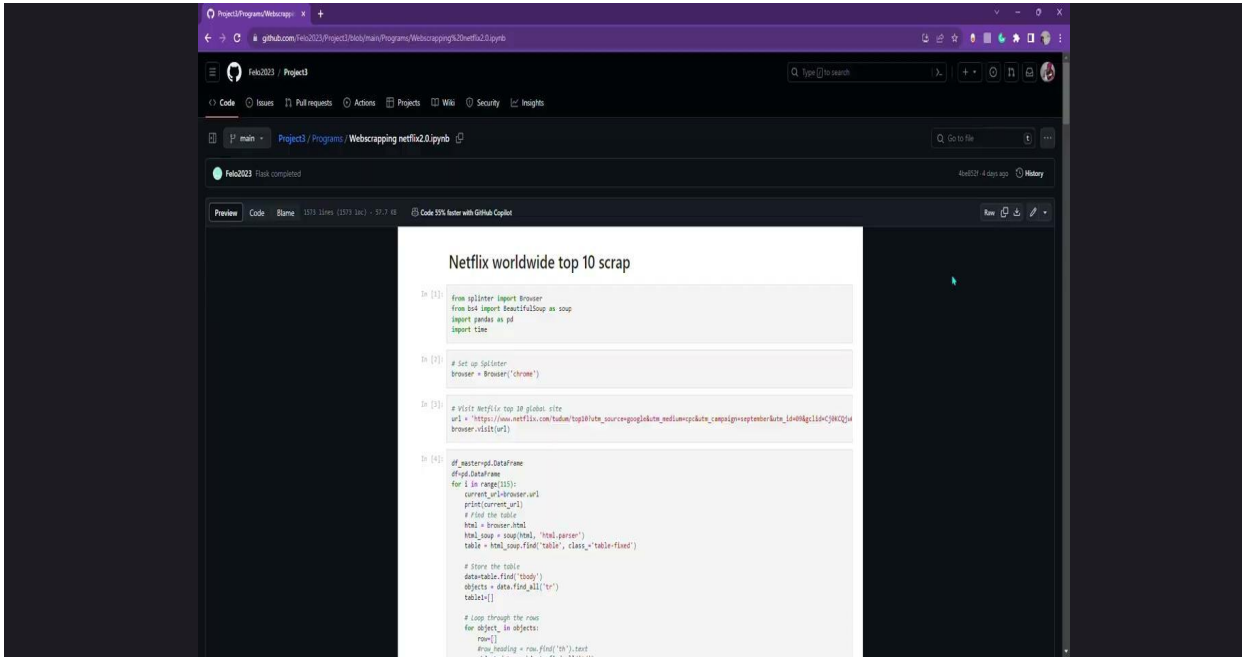
On January 6, 2016, Netflix debuted its service worldwide in Las Vegas, expanding its Internet TV network to more than 130 new nations. Since launching its streaming service in 2007, Netflix has reached 60 different nations by first entering Canada, then Latin America, Europe, Australia, New Zealand,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

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.

How did we do it?



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 Poocco. 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.

Werkzeug

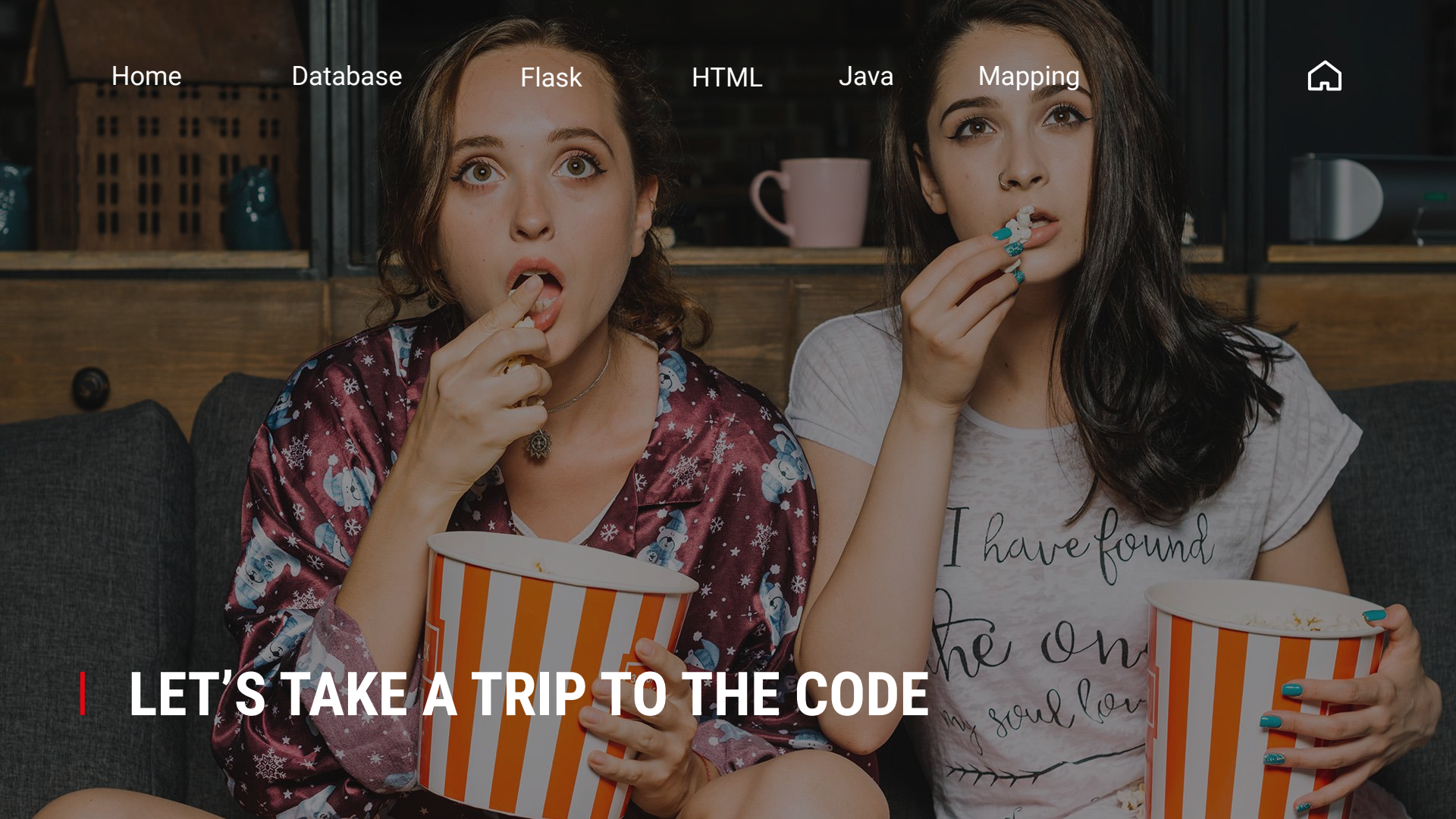
Requests, response objects, and utility functions are all implemented by the WSGI toolkit known as Werkzeug. On top of it can now be created a web frame. Werkzeg serves as one of the foundations of the Flask framework.

jinja2

A well-liked Python template engine is inja2. A web template system renders a dynamic web page by fusing a template with a particular data source.

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.



LET'S TAKE A TRIP TO THE CODE

How did we do it?

```python
import numpy as np
import sqlalchemy
from sqlalchemy.ext.automap import automap_base
from sqlalchemy.orm import Session
from sqlalchemy import create_engine, func
from flask import Flask, jsonify
from datetime import datetime
from flask_cors import CORS
import pandas as pd

# python -m http.server [PORT]

#################################################
# Database Setup
#################################################
engine = create_engine("sqlite:///Netflix_DB.sqlite")

# reflect an existing database into a new model
Base = automap_base()
# reflect the tables
Base.prepare(autoload_with=engine)

# Save reference to the table
global1 = Base.classes.global1
canada = Base.classes.canada
united_kingdom=Base.classes.united_kingdom
australia=Base.classes.australia
egypt=Base.classes.egypt
countries=Base.classes.countries

#################################################
# Flask Setup
#################################################
app = Flask(__name__)
CORS(app, resources={r"/api/*": {"origins": "http://127.0.0.1:8005"}})
#################################################
# Flask Routes
#################################################

@app.route("/")
def welcome():
    """List all available api routes."""
    return (
        f"Available Routes:!!!!!!<br/>"
        f"/api/v1.0/global1<br/>"
        f"/api/v1.0/global2<br/>"
        f"/api/v1.0/canada<br/>"
        f"/api/v1.0/united_kingdom<br/>"
        f"/api/v1.0/australia<br/>"
        f"/api/v1.0/egypt<br/>"
        f"/api/v1.0/countries"
    )
```

```python
@app.route("/api/v1.0/global1")
def global1_func():
    # Create our session (link) from Python to the DB
    session = Session(engine)

    """Return a list of global1 data"""
    # Query all passengers
    results = session.query(global1.No, global1.movie, global1.week_at_Top10,global1.hours_seen,global1.duration,global1.views).all()

    session.close()

    # Create a dictionary from the row data and append to a list of all_passengers
    all_data = []
    for No, movie, week_at_Top10, hours_seen, duration,views in results:
        data_dict = {}
        data_dict["No"] = No
        data_dict["movie"] = movie
        data_dict["week_at_Top10"] = week_at_Top10
        data_dict["hours_seen"] = round(hours_seen,2)
        data_dict["duration"] = str(duration)
        data_dict["views"] = round(views,2)
        all_data.append(data_dict)

    return jsonify(all_data)

@app.route("/api/v1.0/canada")
def canada_func():
    # Create our session (link) from Python to the DB
    session = Session(engine)
    """Return a list of the selected country data"""
    # Query all passengers
    results = session.query(canada.No, canada.movie, canada.week_at_Top10).all()

    session.close()

    # Create a dictionary from the row data and append to a list of all_passengers
    all_data = []
    for No, movie, week_at_Top10 in results:
        data_dict = {}
        data_dict["No"] = No
        data_dict["movie"] = movie
        data_dict["week_at_Top10"] = week_at_Top10
        all_data.append(data_dict)

    return jsonify(all_data)


@app.route("/api/v1.0/united_kingdom")
def united_func():
    # Create our session (link) from Python to the DB
    session = Session(engine)
    """Return a list of the selected country data"""
    # Query all passengers
    results = session.query(united_kingdom.No, united_kingdom.movie, united_kingdom.week_at_Top10).all()
```

How did we do it?

```python
    session.close()

    # Create a dictionary from the row data and append to a list of all_passengers
    all_data = []
    for No, movie, week_at_Top10 in results:
        data_dict = {}
        data_dict["No"] = No
        data_dict["movie"] = movie
        data_dict["week_at_Top10"] = week_at_Top10
        all_data.append(data_dict)

    return jsonify(all_data)

@app.route("/api/v1.0/egypt")
def egypt_func():
    # Create our session (link) from Python to the DB
    session = Session(engine)
    """Return a list of the selected country data"""
    # Query all passengers
    results = session.query(egypt.No, egypt.movie, egypt.week_at_Top10).all()

    session.close()

    # Create a dictionary from the row data and append to a list of all_passengers
    all_data = []
    for No, movie, week_at_Top10 in results:
        data_dict = {}
        data_dict["No"] = No
        data_dict["movie"] = movie
        data_dict["week_at_Top10"] = week_at_Top10
        all_data.append(data_dict)

    return jsonify(all_data)


@app.route("/api/v1.0/australia")
def australia_func():
    # Create our session (link) from Python to the DB
    session = Session(engine)
    """Return a list of the selected country data"""
    # Query all passengers
    results = session.query(australia.No, australia.movie, australia.week_at_Top10).all()

    session.close()

    # Create a dictionary from the row data and append to a list of all_passengers
    all_data = []
    for No, movie, week_at_Top10 in results:
        data_dict = {}
        data_dict["No"] = No
        data_dict["movie"] = movie
        data_dict["week_at_Top10"] = week_at_Top10
        all_data.append(data_dict)
```

```python
@app.route("/api/v1.0/australia")
def australia_func():
    # Create our session (link) from Python to the DB
    session = Session(engine)
    """Return a list of the selected country data"""
    # Query all passengers
    results = session.query(australia.No, australia.movie, australia.week_at_Top10).all()

    session.close()

    # Create a dictionary from the row data and append to a list of all_passengers
    all_data = []
    for No, movie, week_at_Top10 in results:
        data_dict = {}
        data_dict["No"] = No
        data_dict["movie"] = movie
        data_dict["week_at_Top10"] = week_at_Top10
        all_data.append(data_dict)

    return jsonify(all_data)

@app.route("/api/v1.0/global2")
def global2_func():
    # Create our session (link) from Python to the DB
    session = Session(engine)

    """Return a list of global1 data"""
    # Query all passengers
    results = session.query(global1.No, global1.movie, global1.week_at_Top10,global1.hours_seen,global1.duration,global1.views).all()

    session.close()

    # Create a dictionary from the row data and append to a list of all_passengers
    all_data = []
    df = pd.DataFrame(results)
    # Group by 'Movie Name' and find the maximum 'Week at Top 10 Score' for each movie
    max_week_scores = df.groupby('movie')['week_at_Top10'].max().reset_index()
    max_week_scores=max_week_scores.sort_values('week_at_Top10',ascending=False)
    # Merge the original DataFrame with the maximum week scores to get the desired rows
    i=1
    for index, row in max_week_scores.iterrows():
            result_dict={}
            result_dict = row.to_dict()
            result_dict['No']=i
            all_data.append(result_dict)
            i=i+1
    return jsonify(all_data)

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

04 JAVA

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.

How did we do it?

```java
document.addEventListener("DOMContentLoaded", function () {
    const globalTab = document.getElementById("global-tab");
    const canadaTab = document.getElementById("canada-tab");
    const egyptTab = document.getElementById("egypt-tab");
    const ukTab = document.getElementById("uk-tab"); // New tab
    const australiaTab = document.getElementById("australia-tab"); // New tab
    const movieTable = document.getElementById("movie-data");
    const nextButton = document.getElementById("next-button");
    const previousButton = document.getElementById("previous-button");
    const barChartDiv = document.getElementById("bar-chart");
    const movieTable2 = document.getElementById("movie-data2");
    const nextButton2 = document.getElementById("next-button2");
    const previousButton2 = document.getElementById("previous-button2");


    let currentPage = 0; // Current page for pagination
    const pageSize = 10; // Number of rows per page
    let currentTabId = "global1"; // Initialize with the default tab
    let currentTabId2 = "global2"; // Initialize with the default tab

    globalTab.addEventListener("click", () => loadMovieData("global1"));
    canadaTab.addEventListener("click", () => loadMovieData("canada"));
    egyptTab.addEventListener("click", () => loadMovieData("egypt"));
    ukTab.addEventListener("click", () => loadMovieData("united_kingdom")); // New tab
    australiaTab.addEventListener("click", () => loadMovieData("australia")); // New tab
    nextButton.addEventlistener("click", () => nextPage());
    previousButton.addEventlistener("click", () => previousPage());
    nextButton2.addEventlistener("click", () => nextPage2());
    previousButton2.addEventlistener("click", () => previousPage2());


    function loadMovieData(apiEndpoint) {
        currentTabId = apiEndpoint; // Update the current tab
        // Make a request to your API endpoint using fetch or jQuery.ajax
        fetch(`http://127.0.0.1:5000/api/v1.0/${apiEndpoint}`)
            .then((response) => response.json())
            .then((data) => {
                // Clear previous data
                movieTable.innerHTML = "";

                // Calculate the starting and ending index for the current page
                const startIndex = currentPage * pageSize;
                const endIndex = startIndex + pageSize;

                // Get data for the current page
                const currentPageData = data.slice(startIndex, endIndex);

                // Populate the table with the retrieved data for the current page
                for (let i = 0; i < currentPageData.length; i++) {
                    const movie = currentPageData[i];
                    const row = document.createElement("tr");
                    row.innerHTML = `
                        <td>${movie.No}</td>
                        <td>${movie.movie}</td>
                        <td>${movie.week_at_Top10}</td>
```

```java
                        <td>${movie.hours_seen}</td>
                        <td>${movie.duration}</td>
                        <td>${movie.views}</td>
                    `;
                    movieTable.appendChild(row);
                }

                // Update the bar chart with the current page data
                updateBarChart(currentPageData);
            })
            .catch((error) => {
                console.error("Error fetching data:", error);
            });
    }

    function nextPage() {
        currentPage++;
        // Load data for the current tab with pagination
        loadMovieData(currentTabId);
    }

    function previousPage() {
        if (currentPage > 0) {
            currentPage--;
            // Load data for the current tab with pagination
            loadMovieData(currentTabId);
        }
    }

    function updateBarChart(data) {
        const weeksAtTop10 = data.map((movie) => movie.week_at_Top10);
        const movieNames = data.map((movie) => movie.movie);

        const chartData = [{
            x: movieNames,
            y: weeksAtTop10,
            type: 'bar',
        }];

        const layout = {
            title: 'Weeks at Top 10',
            xaxis: { title: 'Movie' },
            yaxis: { title: 'Weeks' },
        };

        // Use Plotly.react to update the chart
        Plotly.react(barChartDiv, chartData, layout);
    }

    // Load data for the default tab (e.g., 'global1')
    loadMovieData("global1");

//

function loadtop10top10(apiEndpoint2) {
    currentTabId2 = apiEndpoint2; // Update the current tab
    // Make a request to your API endpoint using fetch or jQuery.ajax
```

```java
    fetch(`http://127.0.0.1:5000/api/v1.0/${apiEndpoint2}`)
        .then((response) => response.json())
        .then((data) => {
            // Clear previous data
            movieTable2.innerHTML = "";

            // Calculate the starting and ending index for the current page
            const startIndex2 = currentPage * pageSize;
            const endIndex2 = startIndex2 + pageSize;

            // Get data for the current page
            const currentPageData = data.slice(startIndex2, endIndex2);

            // Populate the table with the retrieved data for the current page
            for (let i = 0; i < currentPageData.length; i++) {
                const movie = currentPageData[i];
                const row = document.createElement("tr");
                row.innerHTML = `
                    <td>${movie.No}</td>
                    <td>${movie.movie}</td>
                    <td>${movie.week_at_Top10}</td>

                `;
                movieTable2.appendChild(row);
            }

        })
        .catch((error) => {
            console.error("Error fetching data:", error);
        });
}

function nextPage2() {
    currentPage++;
    // Load data for the current tab with pagination
    loadtop10top10("global2");
}

function previousPage2() {
    if (currentPage > 0) {
        currentPage--;
        // Load data for the current tab with pagination
        loadtop10top10("global2");
    }
}
loadtop10top10("global2");

});
```

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.

LET’S TAKE A TRIP TO THE CODE

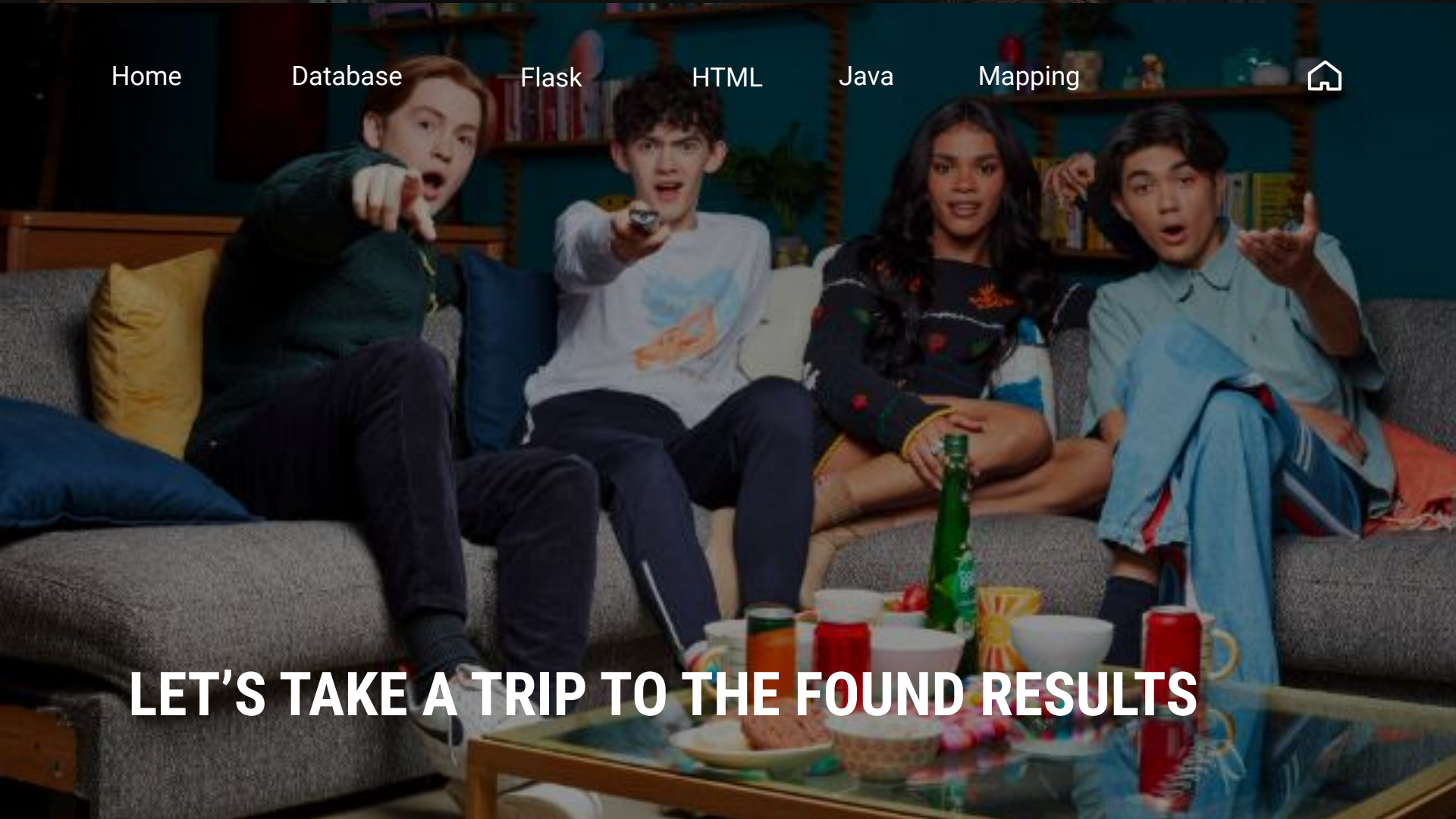
How did we do it?

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Top 10 Movie Dashboard NETFLIX</title>
    <link rel="stylesheet" href="styles2.css">
</head>
<body>
    <div class="scroll-container">
    <div class="header">
    <h1>Top 10 Movies NETFLIX</h1>
   <div class="tabs">
    <button id="global-tab">Global</button>
    <button id="canada-tab">Canada</button>
    <button id="egypt-tab">Egypt</button>
    <button id="uk-tab">United Kingdom</button> <!-- New tab -->
    <button id="australia-tab">Australia</button> <!-- New tab -->
</div>
</div>
<div class="main-container">
    <table id="movie-table">
        <thead>
            <tr>
                <th>No</th>
                <th>Movie</th>
                <th>Weeks at Top 10</th>
                <th>Hours Seen</th>
                <th>Duration</th>
                <th>Views</th>
            </tr>
        </thead>
        <tbody id="movie-data">
            <!-- Data will be populated here -->
        </tbody>
    </table>
    <button id="previous-button">Next week</button>
    <button id="next-button">Previous week</button>
    <div id="bar-chart">
        <!--h2 class="chart-title">Top 10</h2-->
        <div id="plotly-chart"></div>
    </div>
    <table id="movie-table2">
        <thead>
            <tr>
                <th>No</th>
                <th>Movie</th>
                <th>Weeks at Top 10</th>
                <th>Hours Seen</th>
                <th>Duration</th>
                <th>Views</th>
            </tr>
        </thead>
        <tbody id="movie-data">
            <!-- Data will be populated here -->
        </tbody>
    </table>
    <button id="previous-button">Next week</button>
    <button id="next-button">Previous week</button>
    <div id="bar-chart">
        <!--h2 class="chart-title">Top 10</h2-->
        <div id="plotly-chart"></div>
    </div>
    <table id="movie-table2">
        <thead>
            <tr>
                <th>No</th>
                <th>Movie</th>
                <th>Weeks at Top 10</th>
            </tr>
        </thead>
        <tbody id="movie-data2">
            <!-- Data will be populated here -->
        </tbody>
    </table>
    <button id="previous-button2">Next week2</button>
    <button id="next-button2">Previous week2</button>
</div>

</div>
    <script src="https://d3js.org/d3.v5.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.7.1/leaflet.js"></script>
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
    <script src="./Javascript/Readflaskdata6.0.js"></script> <!-- Tu archivo de JavaScript personalizado -->
</body>
</html>
```

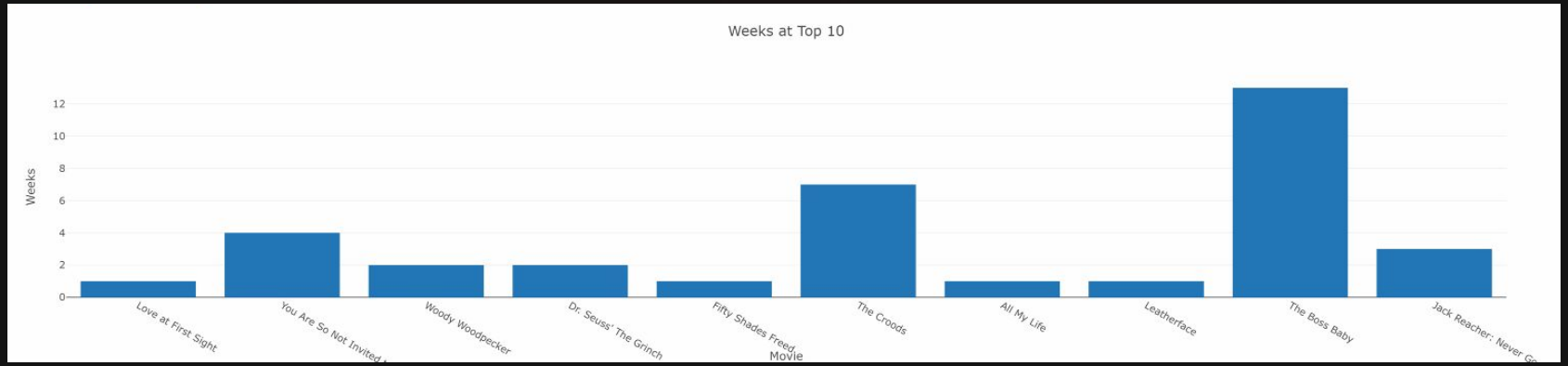
06

Conclusion



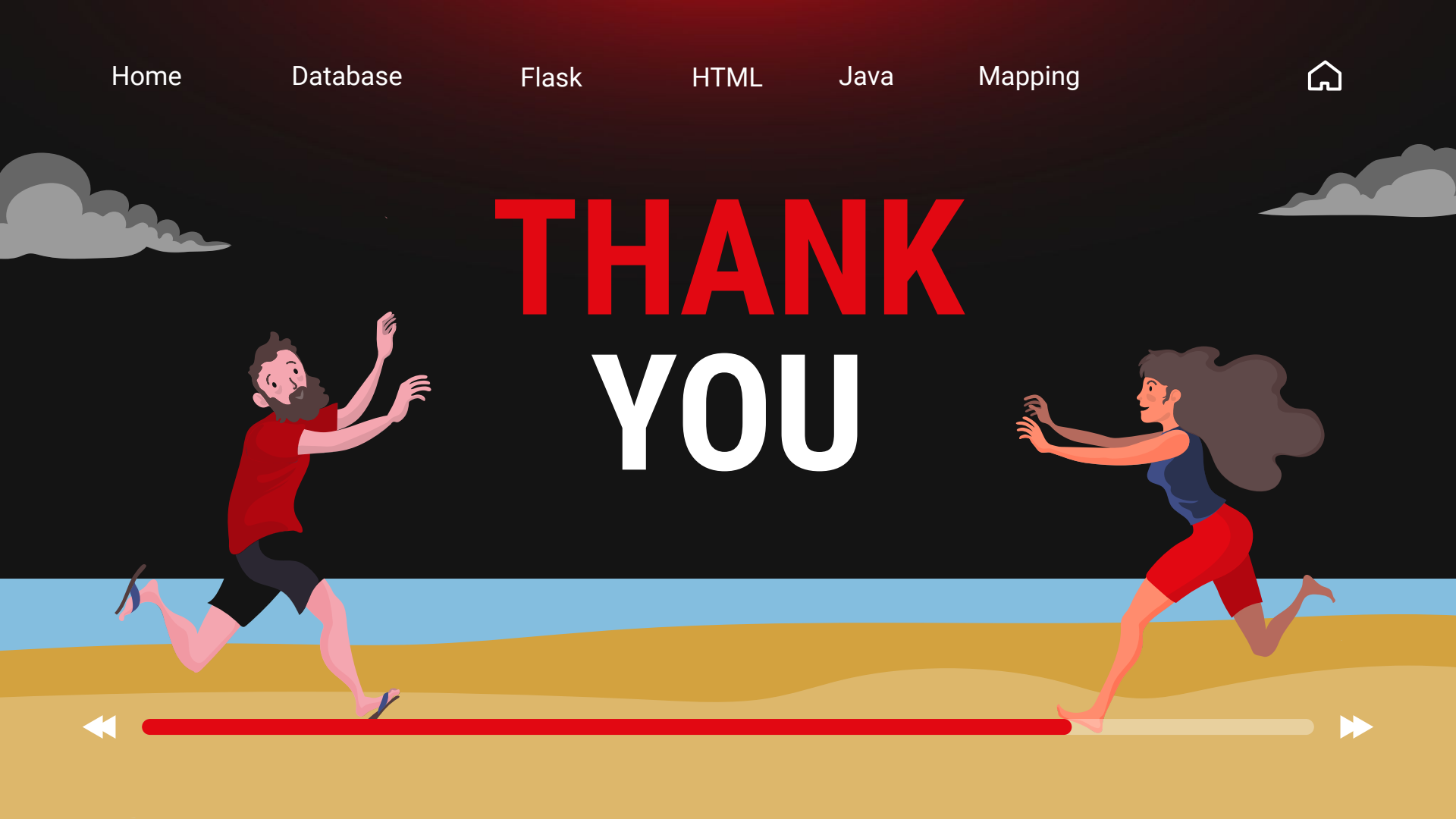
LET’S TAKE A TRIP TO THE FOUND RESULTS

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


Weeks at Top 10
Weeks
0
2
4
6
8
10
12
Love at First Sight
You Are So Not Invited
Woody Woodpecker
Dr. Seuss' The Grinch
Fifty Shades Freed
The Croods
All My Life
Leatherface
The Boss Baby
Jack Reacher: Never Go
Movie

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

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/