Name: Ayan Das
Date: 03/13/2024
CSC 460000 Homework 1

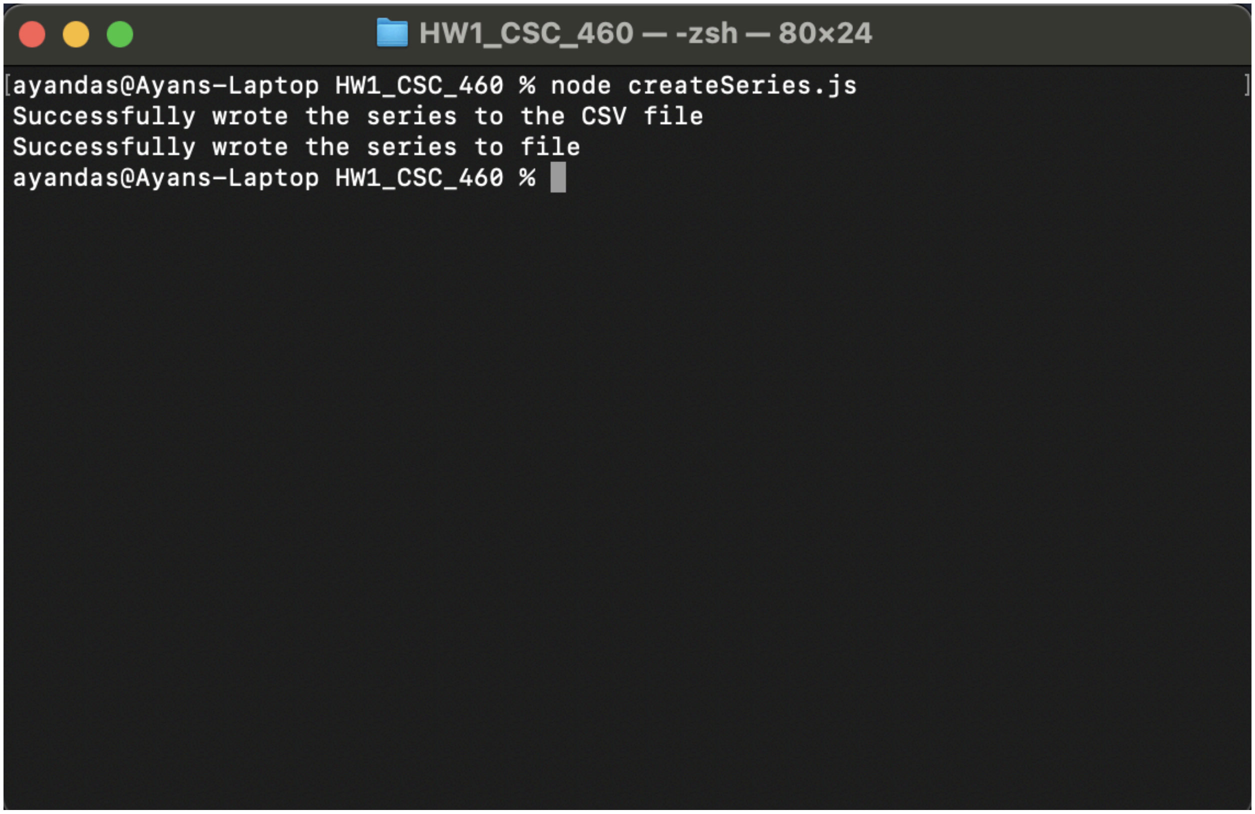
Code for Part 1 a. (Alongside Screenshot of outputs)

- **CreateSeries.js:**

```
const fs = require('fs')

//create a series-like structure using an object
const series = {
'Dune: Part Two' : 9.0,
'Poor Things' : 8.2,
'Dune' : 8.0,
'Avatar : The Last Airbender' : 7.4,
'Oppenheimer' : 8.4,
}

//convert the object to a JSON string
const seriesJSON = JSON.stringify(series, null, 2)

//write a JSON string to a file
fs.writeFile('series.json', seriesJSON, (err) => {
if (err) {
console.error('There was an error writting this file', err); //print out the error
message
} else {
console.log('Successfully wrote the series to file');
}
})

//define the logic for objectToCSV()
function objectToCSV(object) {
const csvRows = []

//get the headers/columns
csvRows.push('Index, Value')

//Loop over the object/dictionary
for (const [key,value] of Object.entries(object)) {
csvRows.push(`${key}, ${value}`) //push the values to their corresponding headers, key
goes to index and value goes to Value column
}

// Form final string CSV
return csvRows.join("\n");
}

//Function to convert an object to a CSV string --> a function that accepts an object
as a parameter
const seriesCSV = objectToCSV(series);

//Write a CSV string to a file
fs.writeFile('series.csv', seriesCSV, (err) => {
if (err) {
console.error('There was an error writting to the CSV file:', err);
} else {
console.log("Successfully wrote the series to the CSV file");
}
});
```

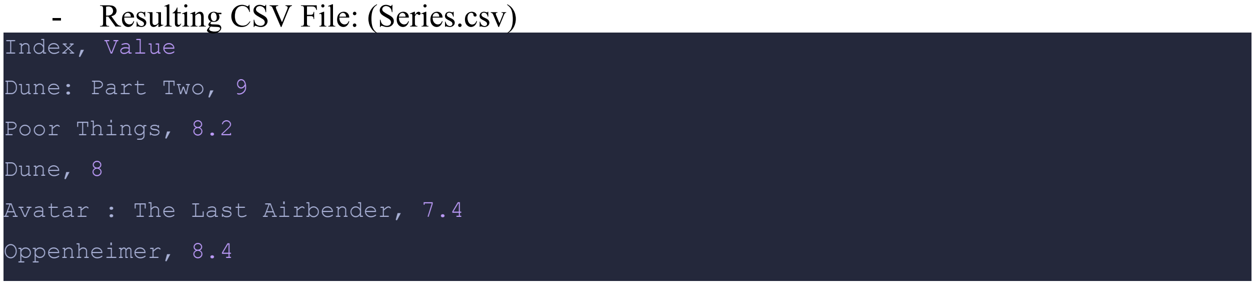
Resulting Output:

- Terminal Output:



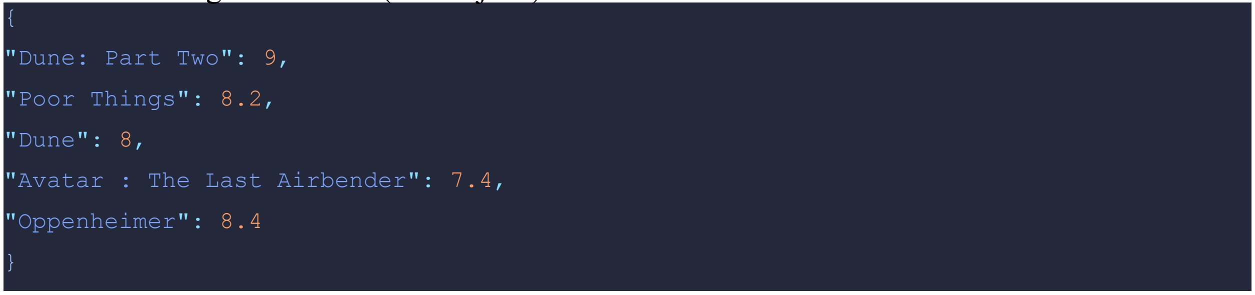
```
[ayandas@Ayans-Laptop HW1_CSC_460 % node createSeries.js                        ]
Successfully wrote the series to the CSV file
Successfully wrote the series to file
ayandas@Ayans-Laptop HW1_CSC_460 % 
```

- Resulting CSV File: (Series.csv)



```
Index, Value
Dune: Part Two, 9
Poor Things, 8.2
Dune, 8
Avatar : The Last Airbender, 7.4
Oppenheimer, 8.4
```

- Resulting JSON File: (Series.json)



```
{
"Dune: Part Two": 9,
"Poor Things": 8.2,
"Dune": 8,
"Avatar : The Last Airbender": 7.4,
"Oppenheimer": 8.4
}
```

- **Code for Part 1b.**

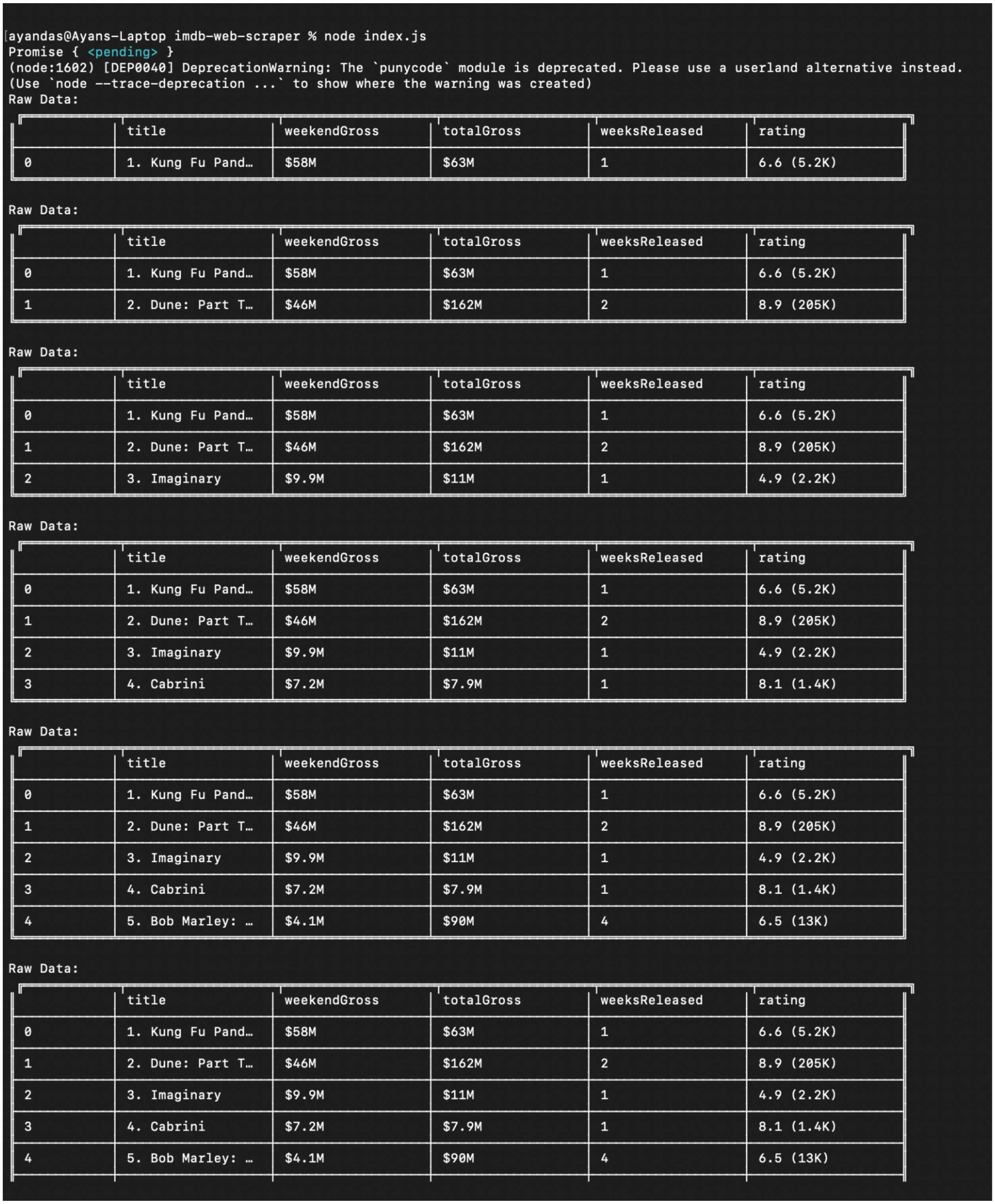
Index.js (A webscraper file to retrieve information about the movies listed on the page):

```
//root file to handle web scraping
/**
* import cheerio and axios
*/

/**
*
* The below code is the data we are interested in the most
*
*/

//import { loadDataAndAnalyze } from './dataAnalysis';


const cheerio = require('cheerio');
const axios = require('axios');
const fs = require('fs');
const { Parser } = require('json2csv');
const dfd = require("danfojs-node");
//const loadDataAndAnalyze = require('./dataAnalysis');

//initialize the data structure
//this will contain the scraped data
const industries = []

//scraping the
//use Axios to connect to target website with the following lines of code
/**
* Downloading target web page
* by performing an HTTP GET request in Axios
*/

async function performScraping() {
const axiosResponse = await axios.request({
method: "GET",
url: "https://www.imdb.com/chart/boxoffice/?ref_=nv_ch_cht",
headers: {
"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/108.0.0.0 Safari/537.36"
}
})

//parsing the html source of the target web page with cheerio
const $ = cheerio.load(axiosResponse.data) //The Cheerio load() method accepts HTML
content in string form

// Selecting the list items containing each title and its metadata
$('ul.ipc-metadata-list li.ipc-metadata-list-summary-item').each((i, element) => {
const title = $(element).find('h3.ipc-title__text').text().trim();
const weekendGross =
$(element).find('ul[data-testid="title-metadata-box-office-data-container"]
li').first().find('span.elpuzG').text().trim();
const totalGross =
$(element).find('ul[data-testid="title-metadata-box-office-data-container"]
li').eq(1).find('span.elpuzG').text().trim();
const weeksReleased =
$(element).find('ul[data-testid="title-metadata-box-office-data-container"]
li').eq(2).find('span.elpuzG').text().trim();
const rating = $(element).find('div[data-testid="ratingGroup--container"]
span.ipc-rating-star--imdb').text().trim();

// Create an object and push it to the industries array
industries.push({ title, weekendGross, totalGross, weeksReleased, rating });
//console.log(industries);

//save to json
// Save to JSON
fs.writeFileSync('movies.json', JSON.stringify(industries, null, 2));

// Save to CSV
const json2csvParser = new Parser();
const csv = json2csvParser.parse(industries);
fs.writeFileSync('industries.csv', csv);

// Reading CSV into DataFrame
let df = new dfd.DataFrame(industries);

// Ensure all numeric data is in a numeric format
// Convert strings to numbers
/* --> TO DO: Fix it, this part of the code causes error
```

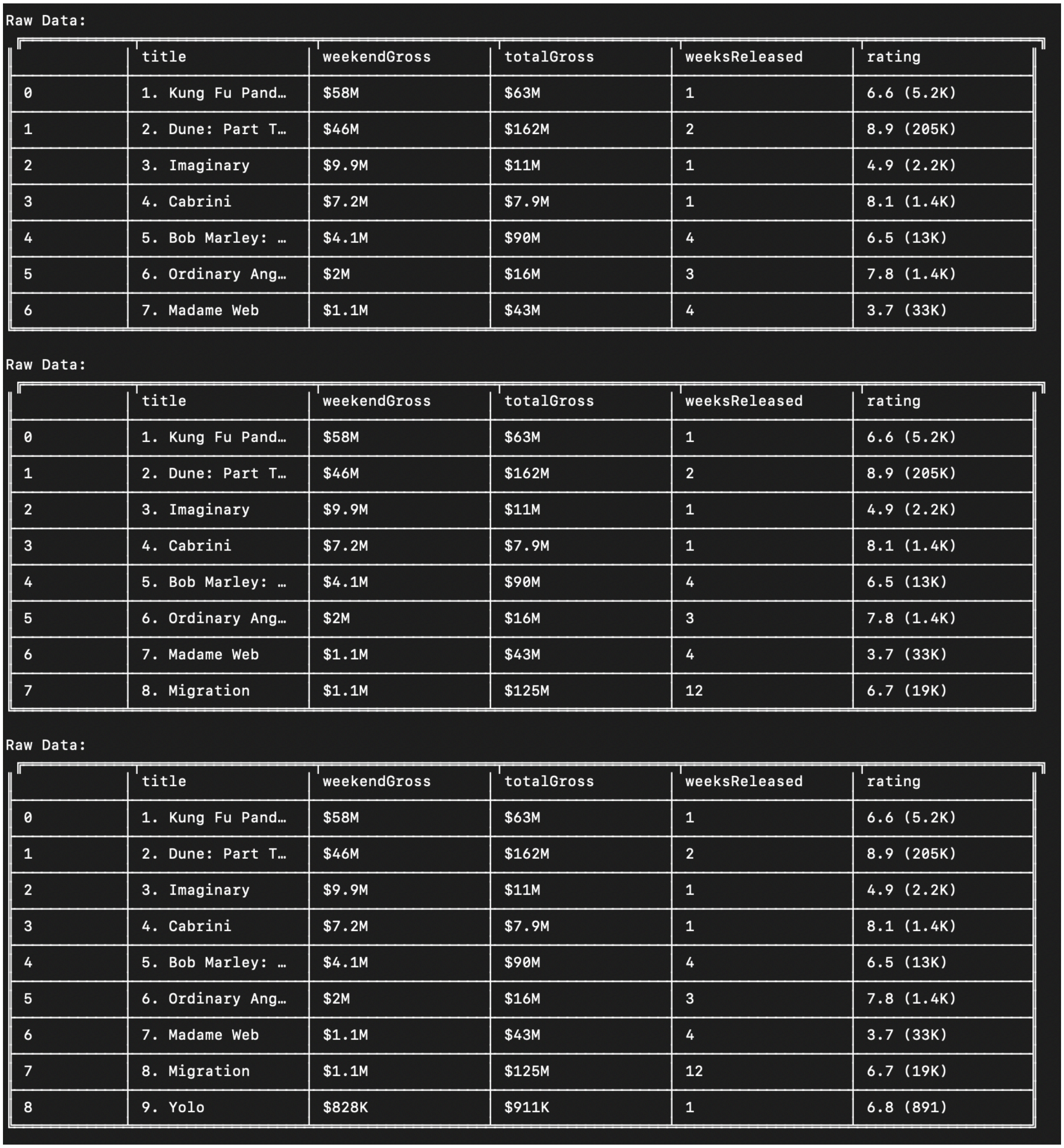
```
df['weekendGross'] = df['weekendGross'].map(value => parseFloat(value.replace(/\$|M/g, '')) * 1e6);
df['totalGross'] = df['totalGross'].map(value => parseFloat(value.replace(/\$|M/g, '')) * 1e6);
df['weeksReleased'] = df['weeksReleased'].map(value => parseInt(value));
df['rating'] = df['rating'].map(value => {
const matches = value.match(/(\d+.\d+|\d+)/);
return matches ? parseFloat(matches[0]) : NaN;
});*/

// Descriptive summary of numerical series
//console.log(df.describe());

// Find all the titles with total gross greater than $40M
//const filteredDf = df.loc({ rows: df['totalGross'].gt(40000000) });
console.log("Raw Data:\n", df.toString());
//console.log("Filtered Data: ", filteredDf.data);

//console.log(axiosResponse);
//console.log("\n The Data is:", axiosResponse.data) //constains the html source code
return df.toString();
})};

//call on the function and save the result
result = performScraping()
console.log(result)
//loadDataAndAnalyze()
//console.log(industries) --> no point printing out the same thing twice

//test to see if file is working, delete afterwards
//console.log("This is a test!");
```

Resulting Output:

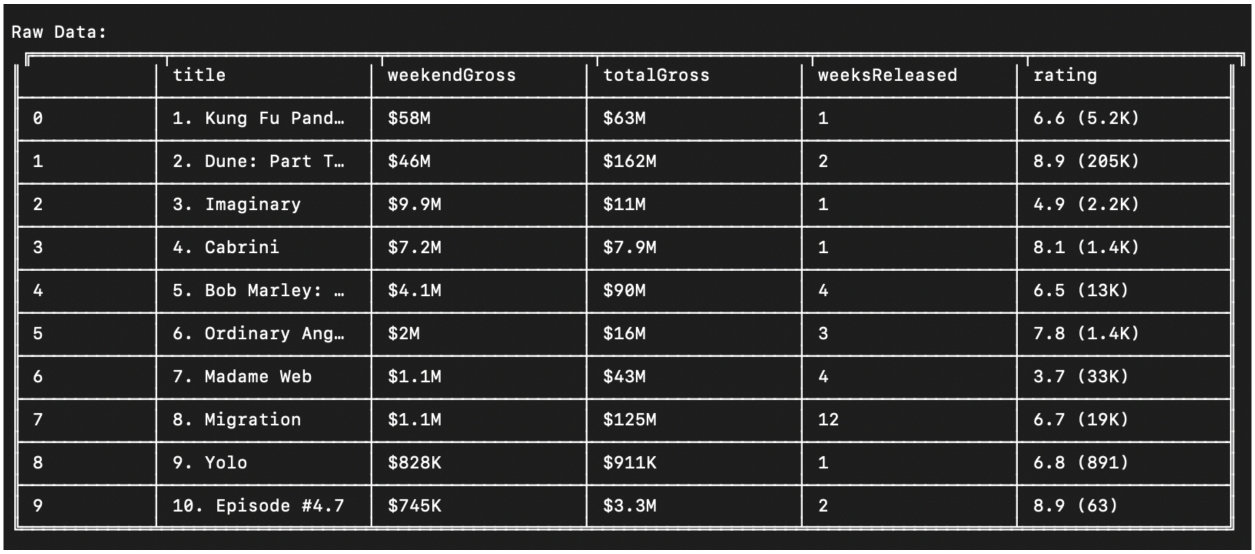
- Terminal Output:



```
[ayandas@Ayans-Laptop imdb-web-scraper % node index.js
Promise { <pending> }
(node:1602) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
Raw Data:
┌────────┬──────────────────┬─────────────┬─────────────┬───────────────┬─────────────┐
│        │ title            │ weekendGross│ totalGross  │ weeksReleased │ rating      │
├────────┼──────────────────┼─────────────┼─────────────┼───────────────┼─────────────┤
│ 0      │ 1. Kung Fu Pand… │ $58M        │ $63M        │ 1             │ 6.6 (5.2K)  │
└────────┴──────────────────┴─────────────┴─────────────┴───────────────┴─────────────┘

Raw Data:
┌────────┬──────────────────┬─────────────┬─────────────┬───────────────┬─────────────┐
│        │ title            │ weekendGross│ totalGross  │ weeksReleased │ rating      │
├────────┼──────────────────┼─────────────┼─────────────┼───────────────┼─────────────┤
│ 0      │ 1. Kung Fu Pand… │ $58M        │ $63M        │ 1             │ 6.6 (5.2K)  │
│ 1      │ 2. Dune: Part T… │ $46M        │ $162M       │ 2             │ 8.9 (205K)  │
└────────┴──────────────────┴─────────────┴─────────────┴───────────────┴─────────────┘

Raw Data:
┌────────┬──────────────────┬─────────────┬─────────────┬───────────────┬─────────────┐
│        │ title            │ weekendGross│ totalGross  │ weeksReleased │ rating      │
├────────┼──────────────────┼─────────────┼─────────────┼───────────────┼─────────────┤
│ 0      │ 1. Kung Fu Pand… │ $58M        │ $63M        │ 1             │ 6.6 (5.2K)  │
│ 1      │ 2. Dune: Part T… │ $46M        │ $162M       │ 2             │ 8.9 (205K)  │
│ 2      │ 3. Imaginary     │ $9.9M       │ $11M        │ 1             │ 4.9 (2.2K)  │
└────────┴──────────────────┴─────────────┴─────────────┴───────────────┴─────────────┘

Raw Data:
┌────────┬──────────────────┬─────────────┬─────────────┬───────────────┬─────────────┐
│        │ title            │ weekendGross│ totalGross  │ weeksReleased │ rating      │
├────────┼──────────────────┼─────────────┼─────────────┼───────────────┼─────────────┤
│ 0      │ 1. Kung Fu Pand… │ $58M        │ $63M        │ 1             │ 6.6 (5.2K)  │
│ 1      │ 2. Dune: Part T… │ $46M        │ $162M       │ 2             │ 8.9 (205K)  │
│ 2      │ 3. Imaginary     │ $9.9M       │ $11M        │ 1             │ 4.9 (2.2K)  │
│ 3      │ 4. Cabrini       │ $7.2M       │ $7.9M       │ 1             │ 8.1 (1.4K)  │
└────────┴──────────────────┴─────────────┴─────────────┴───────────────┴─────────────┘

Raw Data:
┌────────┬──────────────────┬─────────────┬─────────────┬───────────────┬─────────────┐
│        │ title            │ weekendGross│ totalGross  │ weeksReleased │ rating      │
├────────┼──────────────────┼─────────────┼─────────────┼───────────────┼─────────────┤
│ 0      │ 1. Kung Fu Pand… │ $58M        │ $63M        │ 1             │ 6.6 (5.2K)  │
│ 1      │ 2. Dune: Part T… │ $46M        │ $162M       │ 2             │ 8.9 (205K)  │
│ 2      │ 3. Imaginary     │ $9.9M       │ $11M        │ 1             │ 4.9 (2.2K)  │
│ 3      │ 4. Cabrini       │ $7.2M       │ $7.9M       │ 1             │ 8.1 (1.4K)  │
│ 4      │ 5. Bob Marley: … │ $4.1M       │ $90M        │ 4             │ 6.5 (13K)   │
└────────┴──────────────────┴─────────────┴─────────────┴───────────────┴─────────────┘

Raw Data:
┌────────┬──────────────────┬─────────────┬─────────────┬───────────────┬─────────────┐
│        │ title            │ weekendGross│ totalGross  │ weeksReleased │ rating      │
├────────┼──────────────────┼─────────────┼─────────────┼───────────────┼─────────────┤
│ 0      │ 1. Kung Fu Pand… │ $58M        │ $63M        │ 1             │ 6.6 (5.2K)  │
│ 1      │ 2. Dune: Part T… │ $46M        │ $162M       │ 2             │ 8.9 (205K)  │
│ 2      │ 3. Imaginary     │ $9.9M       │ $11M        │ 1             │ 4.9 (2.2K)  │
│ 3      │ 4. Cabrini       │ $7.2M       │ $7.9M       │ 1             │ 8.1 (1.4K)  │
│ 4      │ 5. Bob Marley: … │ $4.1M       │ $90M        │ 4             │ 6.5 (13K)   │
├────────┼──────────────────┼─────────────┼─────────────┼───────────────┼─────────────┤
```



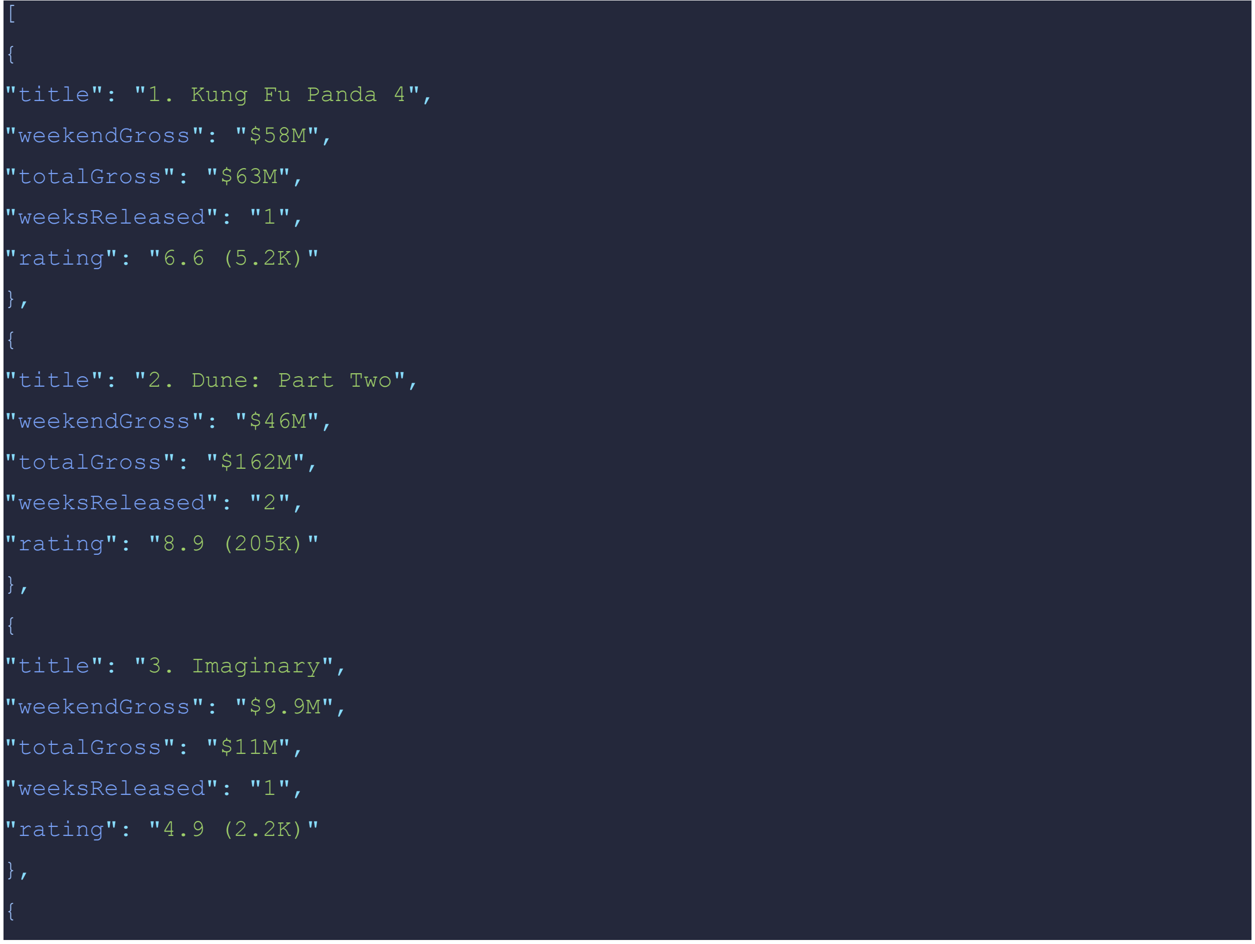
```
Raw Data:
╒════════════╤════════════════╤════════════════╤════════════════╤════════════════╤════════════════╕
│            │ title          │ weekendGross   │ totalGross     │ weeksReleased  │ rating         │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 0          │ 1. Kung Fu Pand… │ $58M         │ $63M           │ 1              │ 6.6 (5.2K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 1          │ 2. Dune: Part T… │ $46M         │ $162M          │ 2              │ 8.9 (205K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 2          │ 3. Imaginary   │ $9.9M          │ $11M           │ 1              │ 4.9 (2.2K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 3          │ 4. Cabrini     │ $7.2M          │ $7.9M          │ 1              │ 8.1 (1.4K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 4          │ 5. Bob Marley: … │ $4.1M        │ $90M           │ 4              │ 6.5 (13K)      │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 5          │ 6. Ordinary Ang… │ $2M          │ $16M           │ 3              │ 7.8 (1.4K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 6          │ 7. Madame Web  │ $1.1M          │ $43M           │ 4              │ 3.7 (33K)      │
╘════════════╧════════════════╧════════════════╧════════════════╧════════════════╧════════════════╛

Raw Data:
╒════════════╤════════════════╤════════════════╤════════════════╤════════════════╤════════════════╕
│            │ title          │ weekendGross   │ totalGross     │ weeksReleased  │ rating         │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 0          │ 1. Kung Fu Pand… │ $58M         │ $63M           │ 1              │ 6.6 (5.2K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 1          │ 2. Dune: Part T… │ $46M         │ $162M          │ 2              │ 8.9 (205K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 2          │ 3. Imaginary   │ $9.9M          │ $11M           │ 1              │ 4.9 (2.2K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 3          │ 4. Cabrini     │ $7.2M          │ $7.9M          │ 1              │ 8.1 (1.4K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 4          │ 5. Bob Marley: … │ $4.1M        │ $90M           │ 4              │ 6.5 (13K)      │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 5          │ 6. Ordinary Ang… │ $2M          │ $16M           │ 3              │ 7.8 (1.4K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 6          │ 7. Madame Web  │ $1.1M          │ $43M           │ 4              │ 3.7 (33K)      │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 7          │ 8. Migration   │ $1.1M          │ $125M          │ 12             │ 6.7 (19K)      │
╘════════════╧════════════════╧════════════════╧════════════════╧════════════════╧════════════════╛

Raw Data:
╒════════════╤════════════════╤════════════════╤════════════════╤════════════════╤════════════════╕
│            │ title          │ weekendGross   │ totalGross     │ weeksReleased  │ rating         │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 0          │ 1. Kung Fu Pand… │ $58M         │ $63M           │ 1              │ 6.6 (5.2K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 1          │ 2. Dune: Part T… │ $46M         │ $162M          │ 2              │ 8.9 (205K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 2          │ 3. Imaginary   │ $9.9M          │ $11M           │ 1              │ 4.9 (2.2K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 3          │ 4. Cabrini     │ $7.2M          │ $7.9M          │ 1              │ 8.1 (1.4K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 4          │ 5. Bob Marley: … │ $4.1M        │ $90M           │ 4              │ 6.5 (13K)      │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 5          │ 6. Ordinary Ang… │ $2M          │ $16M           │ 3              │ 7.8 (1.4K)     │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 6          │ 7. Madame Web  │ $1.1M          │ $43M           │ 4              │ 3.7 (33K)      │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 7          │ 8. Migration   │ $1.1M          │ $125M          │ 12             │ 6.7 (19K)      │
├────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────┤
│ 8          │ 9. Yolo        │ $828K          │ $911K          │ 1              │ 6.8 (891)      │
╘════════════╧════════════════╧════════════════╧════════════════╧════════════════╧════════════════╛
```



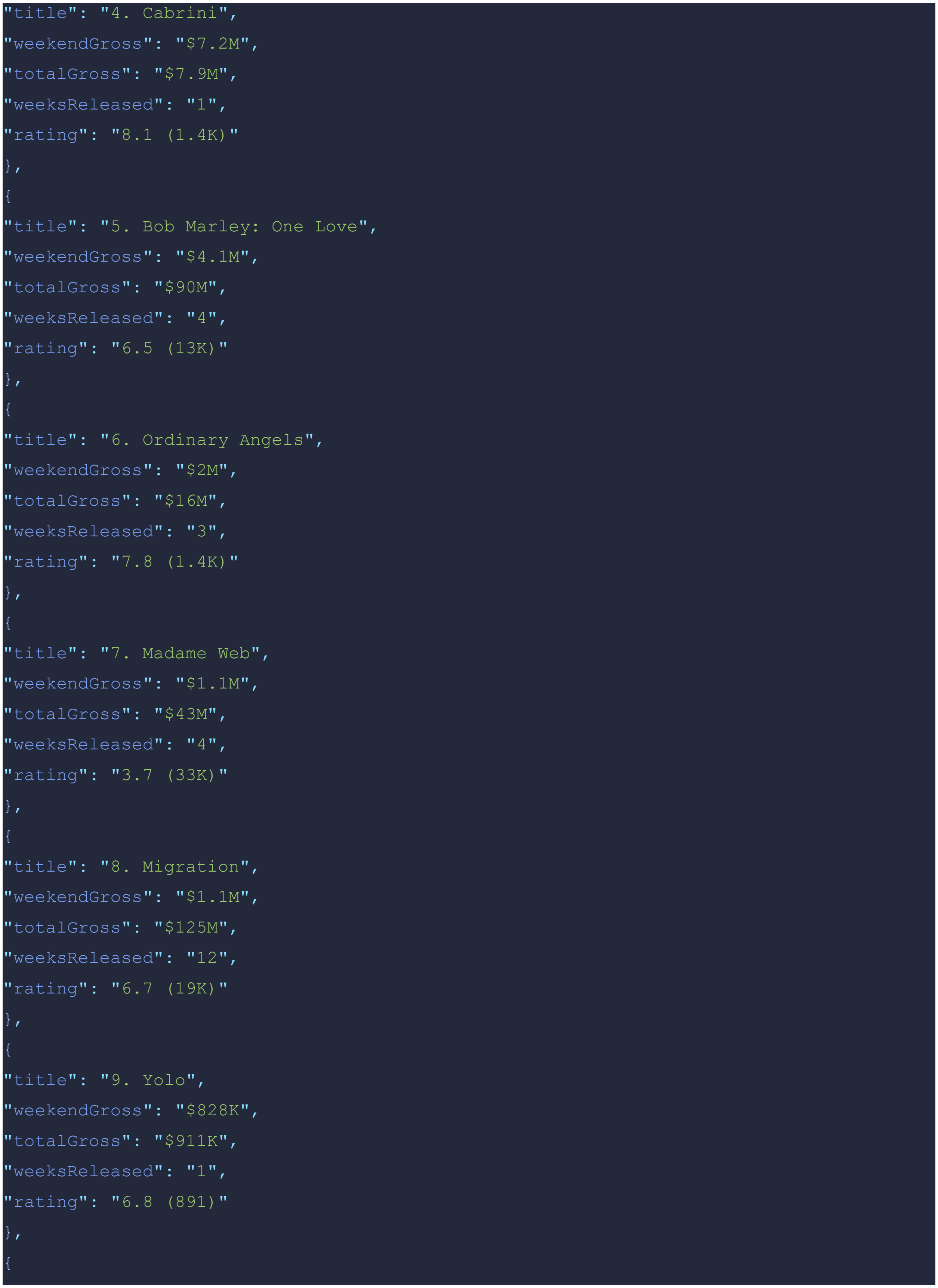
```
Raw Data:
╔════════════╤════════════════╤════════════════╤════════════════╤════════════════╤════════════════╗
║            │ title          │ weekendGross   │ totalGross     │ weeksReleased  │ rating         ║
╟────────────┼────────────────┼────────────────┼────────────────┼────────────────┼────────────────╢
║ 0          │ 1. Kung Fu Pand… │ $58M         │ $63M           │ 1              │ 6.6 (5.2K)     ║
║ 1          │ 2. Dune: Part T… │ $46M         │ $162M          │ 2              │ 8.9 (205K)     ║
║ 2          │ 3. Imaginary   │ $9.9M          │ $11M           │ 1              │ 4.9 (2.2K)     ║
║ 3          │ 4. Cabrini     │ $7.2M          │ $7.9M          │ 1              │ 8.1 (1.4K)     ║
║ 4          │ 5. Bob Marley: … │ $4.1M        │ $90M           │ 4              │ 6.5 (13K)      ║
║ 5          │ 6. Ordinary Ang… │ $2M          │ $16M           │ 3              │ 7.8 (1.4K)     ║
║ 6          │ 7. Madame Web  │ $1.1M          │ $43M           │ 4              │ 3.7 (33K)      ║
║ 7          │ 8. Migration   │ $1.1M          │ $125M          │ 12             │ 6.7 (19K)      ║
║ 8          │ 9. Yolo        │ $828K          │ $911K          │ 1              │ 6.8 (891)      ║
║ 9          │ 10. Episode #4.7 │ $745K        │ $3.3M          │ 2              │ 8.9 (63)       ║
╚════════════╧════════════════╧════════════════╧════════════════╧════════════════╧════════════════╝
```

Note: The output on the terminal shows the process of adding the movie related information each iteration of the loop, since there was a total of 10 movies, 10 iterations were made and the screenshot shows the process of each movie being added one by one.

- Generated file (Movies.json)



```
[
{
"title": "1. Kung Fu Panda 4",
"weekendGross": "$58M",
"totalGross": "$63M",
"weeksReleased": "1",
"rating": "6.6 (5.2K)"
},
{
"title": "2. Dune: Part Two",
"weekendGross": "$46M",
"totalGross": "$162M",
"weeksReleased": "2",
"rating": "8.9 (205K)"
},
{
"title": "3. Imaginary",
"weekendGross": "$9.9M",
"totalGross": "$11M",
"weeksReleased": "1",
"rating": "4.9 (2.2K)"
},
{
```



```
"title": "4. Cabrini",
"weekendGross": "$7.2M",
"totalGross": "$7.9M",
"weeksReleased": "1",
"rating": "8.1 (1.4K)"
},
{
"title": "5. Bob Marley: One Love",
"weekendGross": "$4.1M",
"totalGross": "$90M",
"weeksReleased": "4",
"rating": "6.5 (13K)"
},
{
"title": "6. Ordinary Angels",
"weekendGross": "$2M",
"totalGross": "$16M",
"weeksReleased": "3",
"rating": "7.8 (1.4K)"
},
{
"title": "7. Madame Web",
"weekendGross": "$1.1M",
"totalGross": "$43M",
"weeksReleased": "4",
"rating": "3.7 (33K)"
},
{
"title": "8. Migration",
"weekendGross": "$1.1M",
"totalGross": "$125M",
"weeksReleased": "12",
"rating": "6.7 (19K)"
},
{
"title": "9. Yolo",
"weekendGross": "$828K",
"totalGross": "$911K",
"weeksReleased": "1",
"rating": "6.8 (891)"
},
{
```

```
"title": "10. Episode #4.7",
"weekendGross": "$745K",
"totalGross": "$3.3M",
"weeksReleased": "2",
"rating": "8.9 (63)"
}
]
```

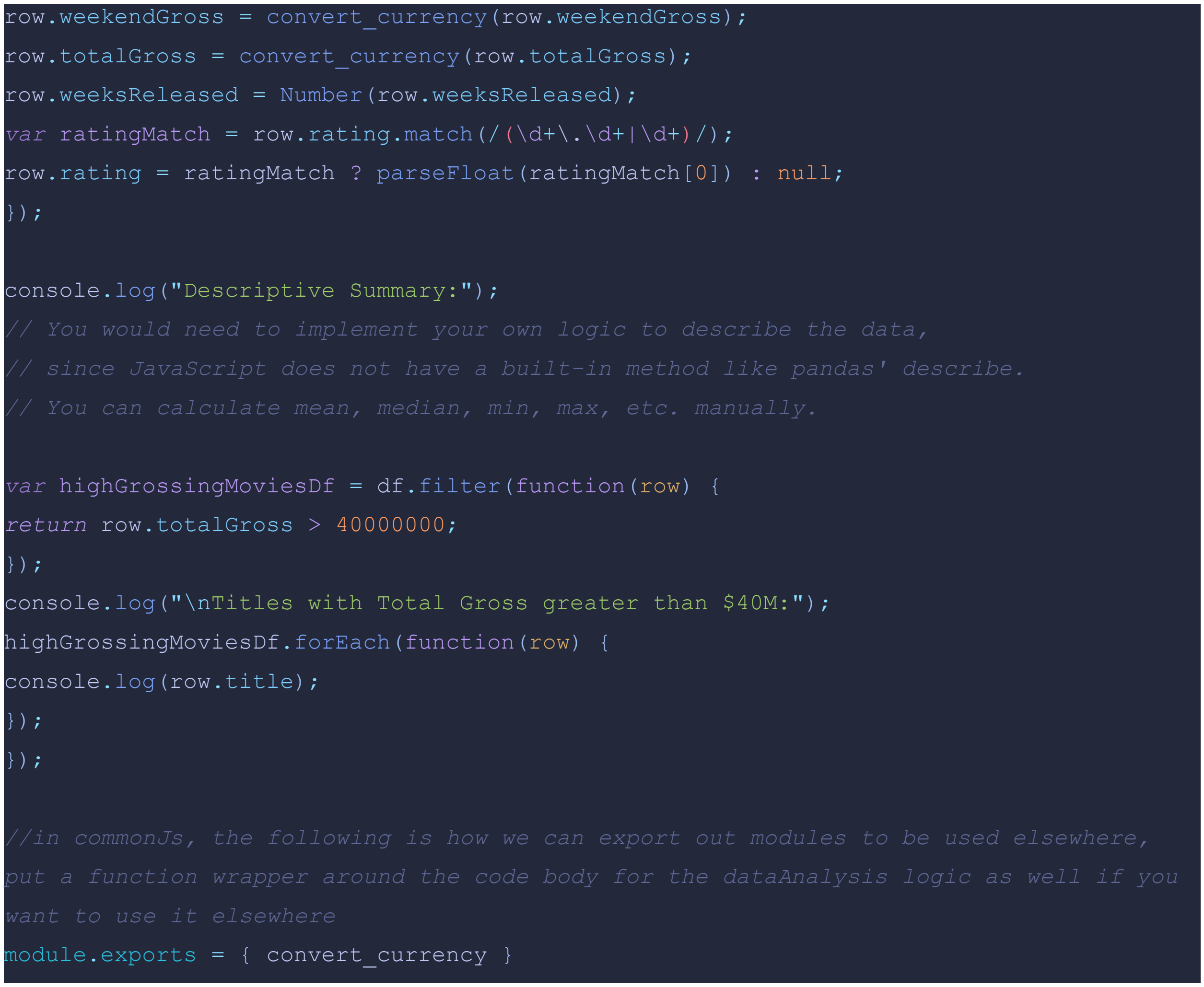
- Generated File (industries.csv) → Same information, but instead saved to a CSV file

```
"title","weekendGross","totalGross","weeksReleased","rating"
"1. Kung Fu Panda 4","$58M","$63M","1","6.6 (5.2K)"
"2. Dune: Part Two","$46M","$162M","2","8.9 (205K)"
"3. Imaginary","$9.9M","$11M","1","4.9 (2.2K)"
"4. Cabrini","$7.2M","$7.9M","1","8.1 (1.4K)"
"5. Bob Marley: One Love","$4.1M","$90M","4","6.5 (13K)"
"6. Ordinary Angels","$2M","$16M","3","7.8 (1.4K)"
"7. Madame Web","$1.1M","$43M","4","3.7 (33K)"
"8. Migration","$1.1M","$125M","12","6.7 (19K)"
"9. Yolo","$828K","$911K","1","6.8 (891)"
"10. Episode #4.7","$745K","$3.3M","2","8.9 (63)"
```

Additional files (done in both python and javascript to do a Descriptive Summary of the numerical series and find all the titles with total gross greater than 40M, using **Mask)**

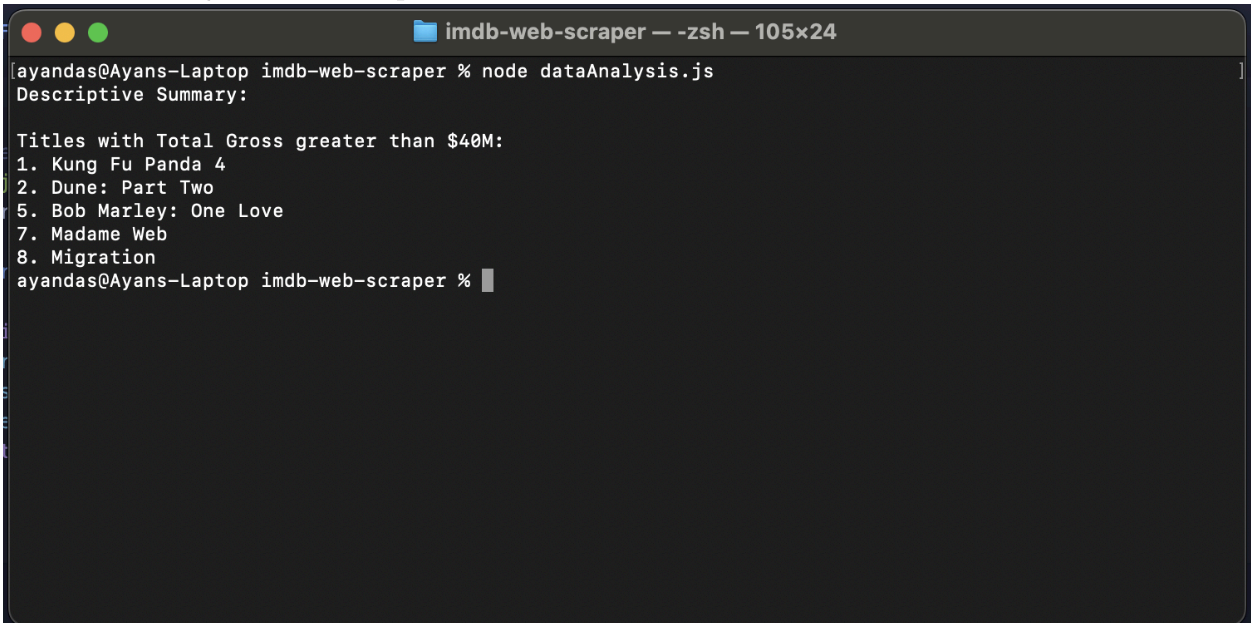
- dataAnalysis.js

```
const fs = require('fs');

function convert_currency(value) {
if (value.includes('M')) {
return parseFloat(value.replace('M', '').replace('$', '')) * 1e6;
} else if (value.includes('K')) {
return parseFloat(value.replace('K', '').replace('$', '')) * 1e3;
} else {
return parseFloat(value.replace('$', ''));
}
}

// Read the JSON file content and then parse it
fs.readFile('movies.json', 'utf8', function(err, data) {
if (err) throw err;
var df = JSON.parse(data);

df.forEach(function(row) {
```



```
row.weekendGross = convert_currency(row.weekendGross);
row.totalGross = convert_currency(row.totalGross);
row.weeksReleased = Number(row.weeksReleased);
var ratingMatch = row.rating.match(/(\d+\.\d+|\d+)/);
row.rating = ratingMatch ? parseFloat(ratingMatch[0]) : null;
});

console.log("Descriptive Summary:");
// You would need to implement your own logic to describe the data,
// since JavaScript does not have a built-in method like pandas' describe.
// You can calculate mean, median, min, max, etc. manually.

var highGrossingMoviesDf = df.filter(function(row) {
return row.totalGross > 40000000;
});
console.log("\nTitles with Total Gross greater than $40M:");
highGrossingMoviesDf.forEach(function(row) {
console.log(row.title);
});
});

//in commonJs, the following is how we can export out modules to be used elsewhere,
put a function wrapper around the code body for the dataAnalysis logic as well if you
want to use it elsewhere
module.exports = { convert_currency }
```

- Resulting Terminal Output:

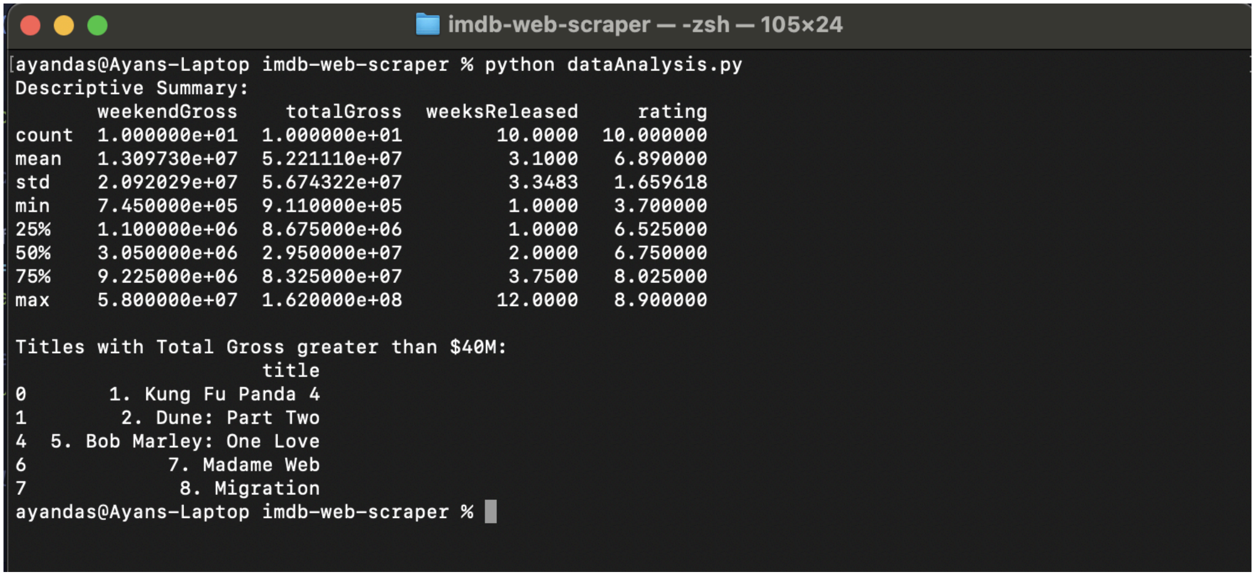


```
[ayandas@Ayans-Laptop imdb-web-scraper % node dataAnalysis.js
Descriptive Summary:

Titles with Total Gross greater than $40M:
1. Kung Fu Panda 4
2. Dune: Part Two
5. Bob Marley: One Love
7. Madame Web
8. Migration
ayandas@Ayans-Laptop imdb-web-scraper % 
```

- dataAnalysis.py

```
import pandas as pd

def convert_currency(value):
if 'M' in value:
return float(value.replace('M', '').replace('$', '')) * 1e6
elif 'K' in value:
return float(value.replace('K', '').replace('$', '')) * 1e3
else:
return float(value.replace('$', ''))

# Load the data from the JSON file
df = pd.read_json('movies.json')

# Clean and convert data to appropriate formats
df['weekendGross'] = df['weekendGross'].apply(convert_currency)
df['totalGross'] = df['totalGross'].apply(convert_currency)
df['weeksReleased'] = df['weeksReleased'].astype(int)
df['rating'] = df['rating'].str.extract('(\d+\.\d+|\d+)').astype(float)

# Perform descriptive summary on the numerical series
print("Descriptive Summary:")
print(df.describe())

# Filter out the titles with total gross greater than $40M
highGrossingMoviesDf = df[df['totalGross'] > 40000000]
print("\nTitles with Total Gross greater than $40M:")
print(highGrossingMoviesDf[['title']])
```

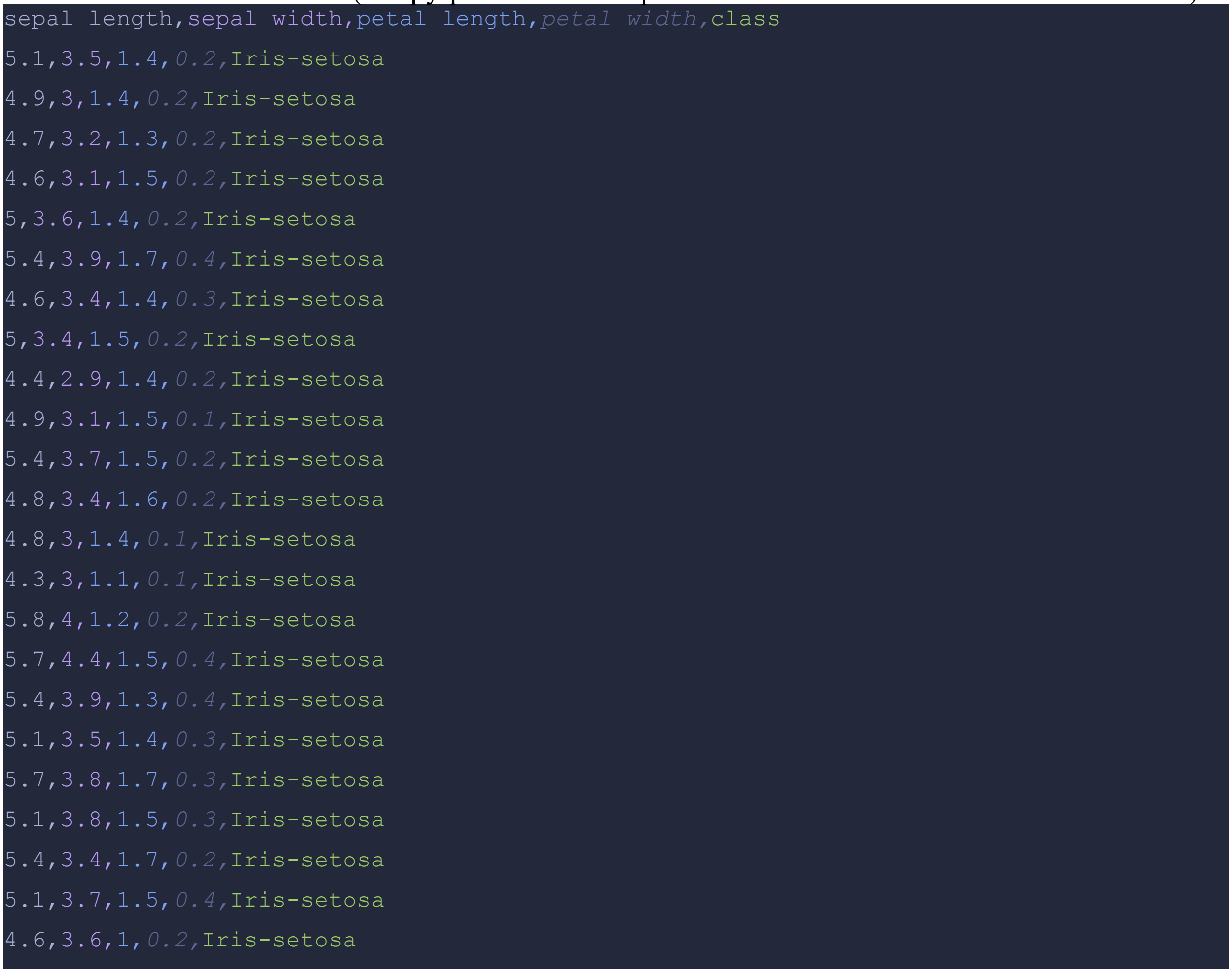
- Resulting Terminal Output:



```
ayandas@Ayans-Laptop imdb-web-scraper % python dataAnalysis.py
Descriptive Summary:
       weekendGross    totalGross  weeksReleased     rating
count  1.000000e+01  1.000000e+01        10.0000  10.000000
mean   1.309730e+07  5.221110e+07         3.1000   6.890000
std    2.092029e+07  5.674322e+07         3.3483   1.659618
min    7.450000e+05  9.110000e+05         1.0000   3.700000
25%    1.100000e+06  8.675000e+06         1.0000   6.525000
50%    3.050000e+06  2.950000e+07         2.0000   6.750000
75%    9.225000e+06  8.325000e+07         3.7500   8.025000
max    5.800000e+07  1.620000e+08        12.0000   8.900000

Titles with Total Gross greater than $40M:
                    title
0       1. Kung Fu Panda 4
1        2. Dune: Part Two
4  5. Bob Marley: One Love
6           7. Madame Web
7            8. Migration
ayandas@Ayans-Laptop imdb-web-scraper %
```

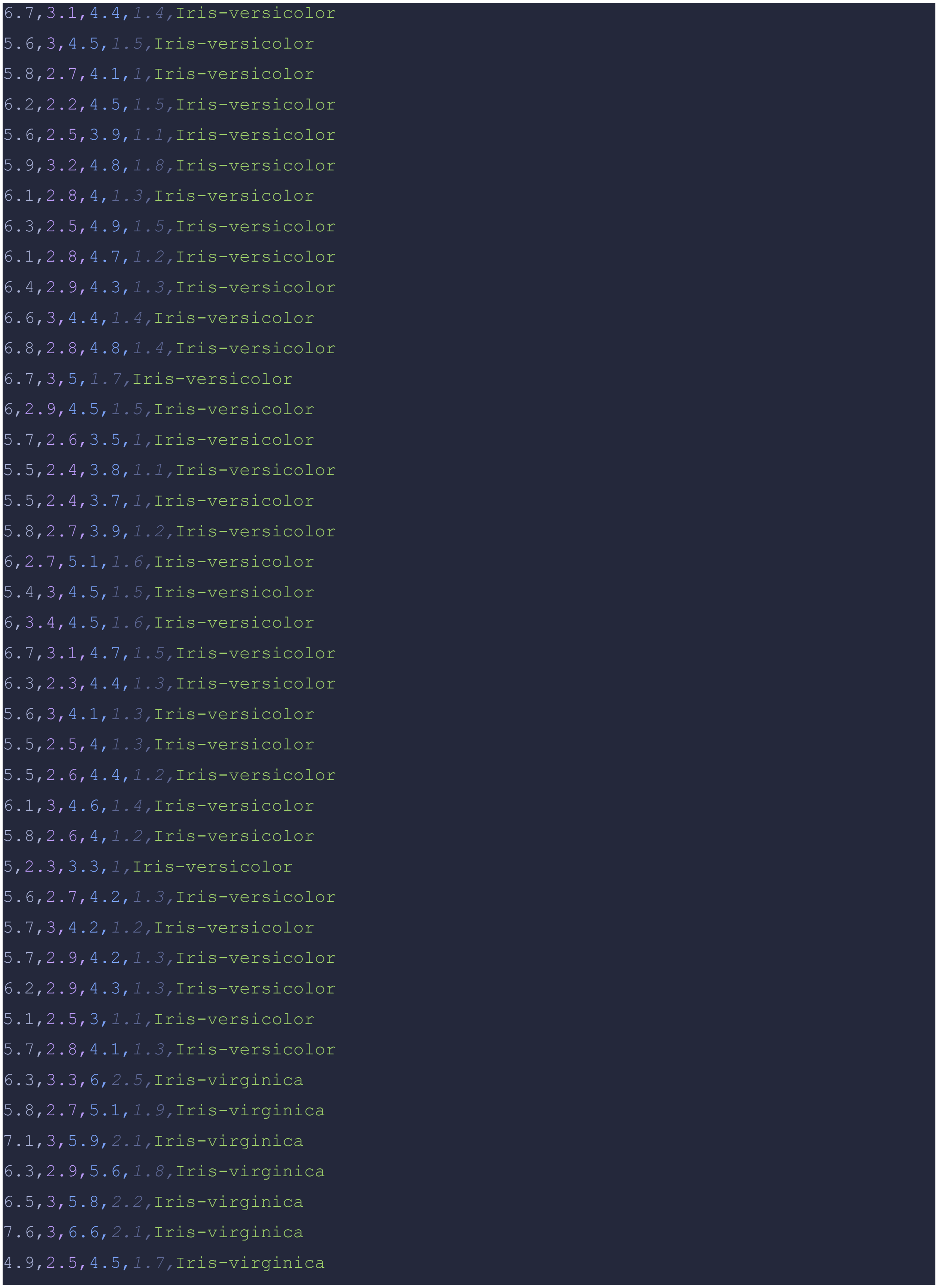
Code For Part 2. (both a and b)

- dataset-iris.csv file: (I copy pasted the link provided and saved it under this file's name)



```
sepal length,sepal width,petal length,petal width,class
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3,1.4,0.1,Iris-setosa
4.3,3,1.1,0.1,Iris-setosa
5.8,4,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1,0.2,Iris-setosa
```

```
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5,3,1.6,0.2,Iris-setosa
5,3.4,1.6,0.4,Iris-setosa
5.2,3.5,1.5,0.2,Iris-setosa
5.2,3.4,1.4,0.2,Iris-setosa
4.7,3.2,1.6,0.2,Iris-setosa
4.8,3.1,1.6,0.2,Iris-setosa
5.4,3.4,1.5,0.4,Iris-setosa
5.2,4.1,1.5,0.1,Iris-setosa
5.5,4.2,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5,3.2,1.2,0.2,Iris-setosa
5.5,3.5,1.3,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
4.4,3,1.3,0.2,Iris-setosa
5.1,3.4,1.5,0.2,Iris-setosa
5,3.5,1.3,0.3,Iris-setosa
4.5,2.3,1.3,0.3,Iris-setosa
4.4,3.2,1.3,0.2,Iris-setosa
5,3.5,1.6,0.6,Iris-setosa
5.1,3.8,1.9,0.4,Iris-setosa
4.8,3,1.4,0.3,Iris-setosa
5.1,3.8,1.6,0.2,Iris-setosa
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5,3.3,1.4,0.2,Iris-setosa
7,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
5.7,2.8,4.5,1.3,Iris-versicolor
6.3,3.3,4.7,1.6,Iris-versicolor
4.9,2.4,3.3,1,Iris-versicolor
6.6,2.9,4.6,1.3,Iris-versicolor
5.2,2.7,3.9,1.4,Iris-versicolor
5,2,3.5,1,Iris-versicolor
5.9,3,4.2,1.5,Iris-versicolor
6,2.2,4,1,Iris-versicolor
6.1,2.9,4.7,1.4,Iris-versicolor
5.6,2.9,3.6,1.3,Iris-versicolor
```



```
6.7,3.1,4.4,1.4,Iris-versicolor
5.6,3,4.5,1.5,Iris-versicolor
5.8,2.7,4.1,1,Iris-versicolor
6.2,2.2,4.5,1.5,Iris-versicolor
5.6,2.5,3.9,1.1,Iris-versicolor
5.9,3.2,4.8,1.8,Iris-versicolor
6.1,2.8,4,1.3,Iris-versicolor
6.3,2.5,4.9,1.5,Iris-versicolor
6.1,2.8,4.7,1.2,Iris-versicolor
6.4,2.9,4.3,1.3,Iris-versicolor
6.6,3,4.4,1.4,Iris-versicolor
6.8,2.8,4.8,1.4,Iris-versicolor
6.7,3,5,1.7,Iris-versicolor
6,2.9,4.5,1.5,Iris-versicolor
5.7,2.6,3.5,1,Iris-versicolor
5.5,2.4,3.8,1.1,Iris-versicolor
5.5,2.4,3.7,1,Iris-versicolor
5.8,2.7,3.9,1.2,Iris-versicolor
6,2.7,5.1,1.6,Iris-versicolor
5.4,3,4.5,1.5,Iris-versicolor
6,3.4,4.5,1.6,Iris-versicolor
6.7,3.1,4.7,1.5,Iris-versicolor
6.3,2.3,4.4,1.3,Iris-versicolor
5.6,3,4.1,1.3,Iris-versicolor
5.5,2.5,4,1.3,Iris-versicolor
5.5,2.6,4.4,1.2,Iris-versicolor
6.1,3,4.6,1.4,Iris-versicolor
5.8,2.6,4,1.2,Iris-versicolor
5,2.3,3.3,1,Iris-versicolor
5.6,2.7,4.2,1.3,Iris-versicolor
5.7,3,4.2,1.2,Iris-versicolor
5.7,2.9,4.2,1.3,Iris-versicolor
6.2,2.9,4.3,1.3,Iris-versicolor
5.1,2.5,3,1.1,Iris-versicolor
5.7,2.8,4.1,1.3,Iris-versicolor
6.3,3.3,6,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3,5.9,2.1,Iris-virginica
6.3,2.9,5.6,1.8,Iris-virginica
6.5,3,5.8,2.2,Iris-virginica
7.6,3,6.6,2.1,Iris-virginica
4.9,2.5,4.5,1.7,Iris-virginica
```

```
7.3,2.9,6.3,1.8,Iris-virginica
6.7,2.5,5.8,1.8,Iris-virginica
7.2,3.6,6.1,2.5,Iris-virginica
6.5,3.2,5.1,2,Iris-virginica
6.4,2.7,5.3,1.9,Iris-virginica
6.8,3,5.5,2.1,Iris-virginica
5.7,2.5,5,2,Iris-virginica
5.8,2.8,5.1,2.4,Iris-virginica
6.4,3.2,5.3,2.3,Iris-virginica
6.5,3,5.5,1.8,Iris-virginica
7.7,3.8,6.7,2.2,Iris-virginica
7.7,2.6,6.9,2.3,Iris-virginica
6,2.2,5,1.5,Iris-virginica
6.9,3.2,5.7,2.3,Iris-virginica
5.6,2.8,4.9,2,Iris-virginica
7.7,2.8,6.7,2,Iris-virginica
6.3,2.7,4.9,1.8,Iris-virginica
6.7,3.3,5.7,2.1,Iris-virginica
7.2,3.2,6,1.8,Iris-virginica
6.2,2.8,4.8,1.8,Iris-virginica
6.1,3,4.9,1.8,Iris-virginica
6.4,2.8,5.6,2.1,Iris-virginica
7.2,3,5.8,1.6,Iris-virginica
7.4,2.8,6.1,1.9,Iris-virginica
7.9,3.8,6.4,2,Iris-virginica
6.4,2.8,5.6,2.2,Iris-virginica
6.3,2.8,5.1,1.5,Iris-virginica
6.1,2.6,5.6,1.4,Iris-virginica
7.7,3,6.1,2.3,Iris-virginica
6.3,3.4,5.6,2.4,Iris-virginica
6.4,3.1,5.5,1.8,Iris-virginica
6,3,4.8,1.8,Iris-virginica
6.9,3.1,5.4,2.1,Iris-virginica
6.7,3.1,5.6,2.4,Iris-virginica
6.9,3.1,5.1,2.3,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
6.8,3.2,5.9,2.3,Iris-virginica
6.7,3.3,5.7,2.5,Iris-virginica
6.7,3,5.2,2.3,Iris-virginica
6.3,2.5,5,1.9,Iris-virginica
6.5,3,5.2,2,Iris-virginica
6.2,3.4,5.4,2.3,Iris-virginica
```

```
5.9,3,5.1,1.8,Iris-virginica
```

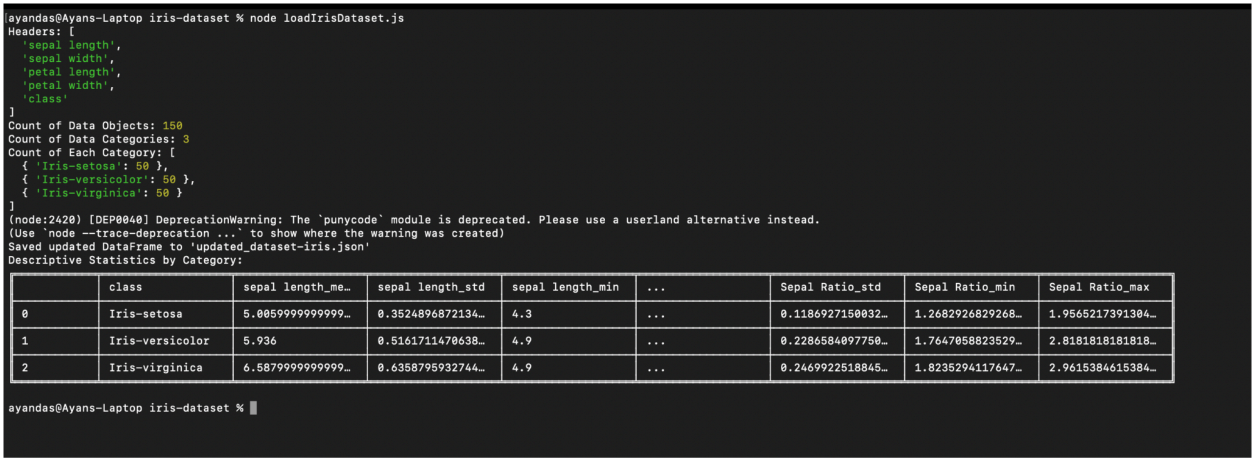
- loadIrisDataset.js

```
const fs = require('fs');
const Papa = require('papaparse');
const dfd = require('danfojs-node');
// Read the CSV file
const fileContent = fs.readFileSync('dataset-iris.csv', 'utf8');

// Parse the CSV content
Papa.parse(fileContent, {
header: true,
complete: (results) => {
const data = results.data;

// Headers
const headers = results.meta.fields;

// Count of Data Object (Number of Rows)
const countOfDataObjects = data.length;

// Count of Data Categories (Unique Classes)
const uniqueClasses = [...new Set(data.map(item => item.class))];
const countOfDataCategories = uniqueClasses.length;

// Count of Each Category
const countOfEachCategory = uniqueClasses.map((cls) => ({
[cls]: data.filter((item) => item.class === cls).length
}));

// Series Data Types
// This is more complex in JavaScript because all CSV fields are read as strings.
// You would need to write a function to determine types manually, usually in JS we
keep it as is.

console.log("Headers:", headers);
console.log("Count of Data Objects:", countOfDataObjects);
console.log("Count of Data Categories:", countOfDataCategories);
console.log("Count of Each Category:", countOfEachCategory);
// Data types not shown due to the manual nature of determination
}
```

```
});

dfd.readCSV('dataset-iris.csv').then(df => {
// Add new columns
df.addColumn('Petal Ratio', df['petal length'].div(df['petal width']), { inplace: true
});
df.addColumn('Sepal Ratio', df['sepal length'].div(df['sepal width']), { inplace: true
});

// Convert DataFrame to a JSON object
const jsonObj = dfd.toJSON(df);

// Stringify the JSON object
const jsonStr = JSON.stringify(jsonObj);

// Write the stringified JSON to a file
fs.writeFileSync('updated_dataset-iris.json', jsonStr);
console.log("Saved updated DataFrame to 'updated_dataset-iris.json'");

// Group by the 'class' column and aggregate
let groupedDf = df.groupby(['class']).agg({
'sepal length': ['mean', 'std', 'min', 'max'],
'sepal width': ['mean', 'std', 'min', 'max'],
'petal length': ['mean', 'std', 'min', 'max'],
'petal width': ['mean', 'std', 'min', 'max'],
'Petal Ratio': ['mean', 'std', 'min', 'max'],
'Sepal Ratio': ['mean', 'std', 'min', 'max'],
});

console.log("Descriptive Statistics by Category:");
groupedDf.print();

// Optionally, save the aggregated DataFrame to a new CSV file
dfd.toCSV(groupedDf, {filePath: "descriptive_stats_by_category.csv"})
/*
.then(() => {
console.log("Saved descriptive statistics by category to
'descriptive_stats_by_category.csv'");
}).catch(err => {
console.error("Error saving CSV: ", err);
}); */
}).catch(err => {
```

```
console.error("Error processing data: ", err);
});
```

- Resulting Terminal Output:



```
ayandas@Ayans-Laptop iris-dataset % node loadIrisDataset.js
Headers: [
  'sepal length',
  'sepal width',
  'petal length',
  'petal width',
  'class'
]
Count of Data Objects: 150
Count of Data Categories: 3
Count of Each Category: [
  { 'Iris-setosa': 50 },
  { 'Iris-versicolor': 50 },
  { 'Iris-virginica': 50 }
]
(node:2420) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
Saved updated DataFrame to 'updated_dataset-iris.json'
Descriptive Statistics by Category:
┌─────────┬─────────────────┬───────────────────┬───────────────────┬───────────────────┬─────────┬───────────────────┬───────────────────┬───────────────────┐
│         │ class           │ sepal length_me…  │ sepal length_std  │ sepal length_min  │ ...     │ Sepal Ratio_std   │ Sepal Ratio_min   │ Sepal Ratio_max   │
├─────────┼─────────────────┼───────────────────┼───────────────────┼───────────────────┼─────────┼───────────────────┼───────────────────┼───────────────────┤
│ 0       │ Iris-setosa     │ 5.0059999999999…  │ 0.3524896872134…  │ 4.3               │ ...     │ 0.1186927150032…  │ 1.2682926829268…  │ 1.9565217391304…  │
├─────────┼─────────────────┼───────────────────┼───────────────────┼───────────────────┼─────────┼───────────────────┼───────────────────┼───────────────────┤
│ 1       │ Iris-versicolor │ 5.936             │ 0.5161711470638…  │ 4.9               │ ...     │ 0.2286584097750…  │ 1.7647058823529…  │ 2.8181818181818…  │
├─────────┼─────────────────┼───────────────────┼───────────────────┼───────────────────┼─────────┼───────────────────┼───────────────────┼───────────────────┤
│ 2       │ Iris-virginica  │ 6.5879999999999…  │ 0.6358795932744…  │ 4.9               │ ...     │ 0.2469922518845…  │ 1.8235294117647…  │ 2.9615384615384…  │
└─────────┴─────────────────┴───────────────────┴───────────────────┴───────────────────┴─────────┴───────────────────┴───────────────────┴───────────────────┘

ayandas@Ayans-Laptop iris-dataset % 
```

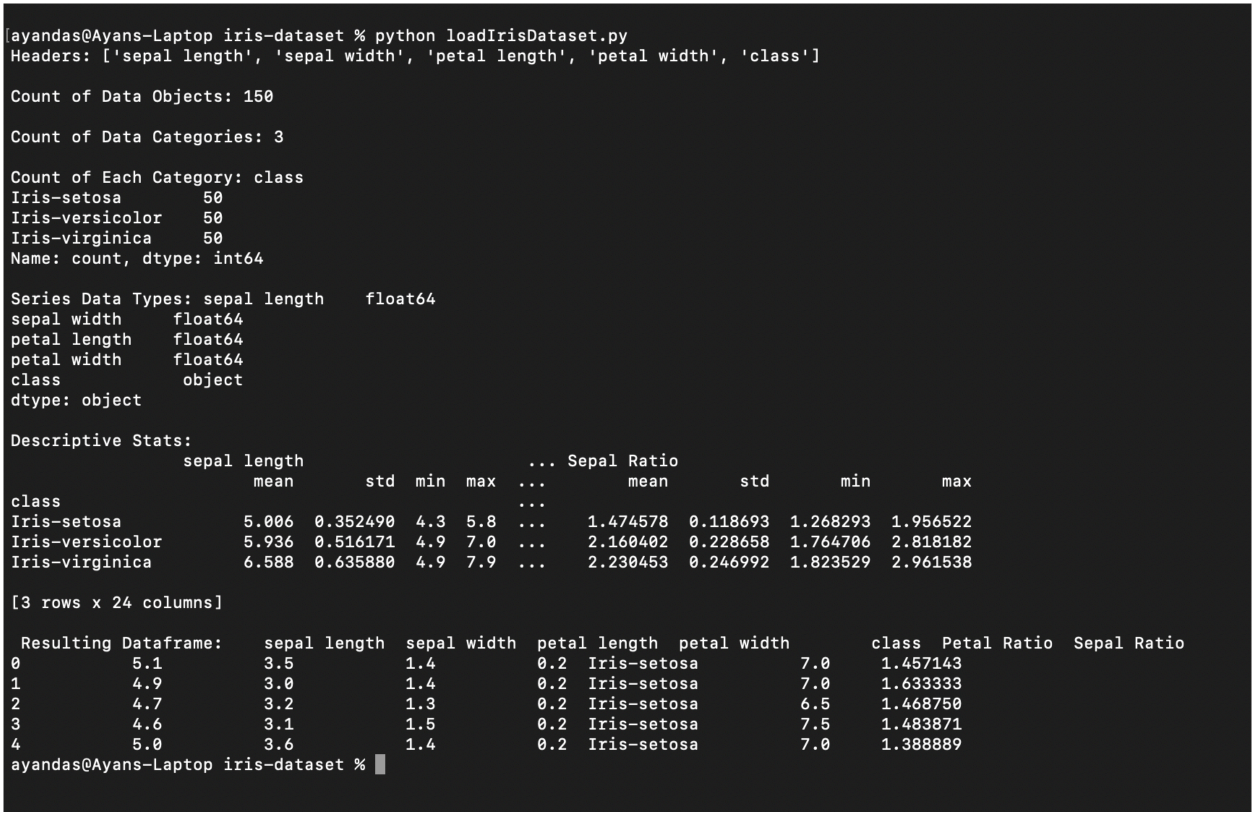
- Resulting generated file (descriptive_stats_by_category.csv) → This is how the .csv file appears in VS code

```
class,sepal length_mean,sepal length_std,sepal length_min,sepal length_max,sepal width_mean,sepal width_std,sepal width_min,sepal width_max,petal length_mean,petal length_std,petal length_min,petal length_max,petal width_mean,petal width_std,petal width_min,petal width_max,Petal Ratio_mean,Petal Ratio_std,Petal Ratio_min,Petal Ratio_max,Sepal Ratio_mean,Sepal Ratio_std,Sepal Ratio_min,Sepal Ratio_max
Iris-setosa,5.005999999999999,0.3524896872134512,4.3,5.8,3.4180000000000006,0.38102439795469095,2.3,4.4,1.464,0.1735111594364455,1,1.9,0.24399999999999999,0.10720950308167837,0.1,0.6,7.077999999999999,3.1237794714001565,2.666666666666667,15,1.4745783620263324,0.11869271500322688,1.2682926829268295,1.956521739130435
Iris-versicolor,5.936,0.5161711470638635,4.9,7,2.7700000000000005,0.3137983233784114,2,3.4,4.26,0.46991097723995806,3,5.1,1.3259999999999998,0.197752680004544,1,1.8,3.2428369326751683,0.3124564643692557,2.6666666666666665,4.1,2.160402191687831,0.2286584097750541,1.7647058823529411,2.8181818181818181
Iris-virginica,6.587999999999998,0.635879593274432,4.9,7.9,2.9739999999999998,0.3224966381726376,2.2,3.8,5.552,0.5518946956639835,4.5,6.9,2.026,0.27465005563666733,1.4,2.5,2.7806623384004467,0.4073670302541582,2.125,4,2.230452738894224,0.24699225188455234,1.823529411764706,2.9615384615384617
```

- loadIrisDataset.py

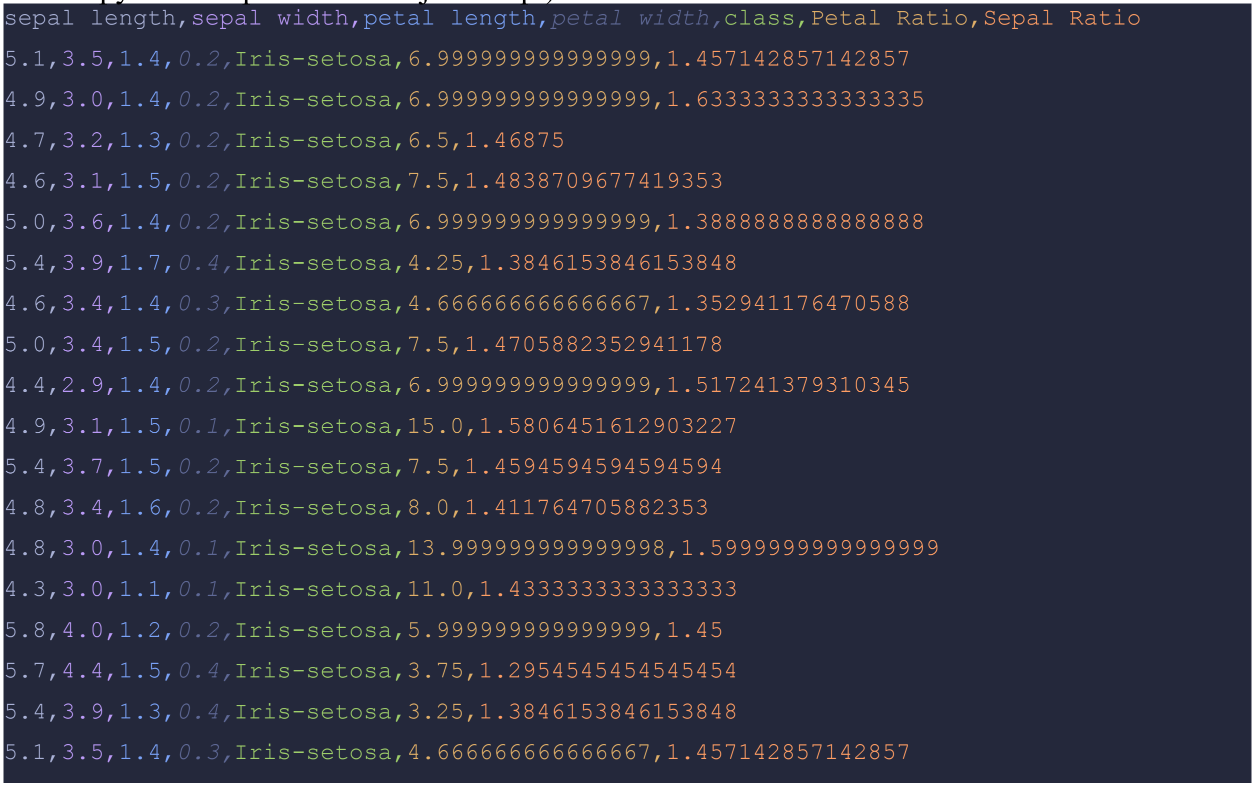
```python
import pandas as pd

# Load the data into a DataFrame
df = pd.read_csv('dataset-iris.csv') # set the path to the iris dataset csv file

# Headers
headers = df.columns.tolist()

# Count of Data Object (Number of Rows)
count_of_data_objects = df.shape[0]

# Count of Data Categories (Unique Classes)
count_of_data_categories = df['class'].nunique()

# Count of Each Category
count_of_each_category = df['class'].value_counts()

# Series Data Types
series_data_types = df.dtypes

# Add new columns
df['Petal Ratio'] = df['petal length'] / df['petal width']
df['Sepal Ratio'] = df['sepal length'] / df['sepal width']

# Save the new DataFrame into a CSV file
df.to_csv('updated_dataset-iris.csv', index=False)

# Get descriptive statistics
descriptive_stats = df.groupby('class').agg(['mean', 'std', 'min', 'max'])

# Printing the information
print("Headers:", headers) # lists out the column in the dataset
print("\nCount of Data Objects:", count_of_data_objects)
print("\nCount of Data Categories:", count_of_data_categories)
print("\nCount of Each Category:", count_of_each_category)
print("\nSeries Data Types:", series_data_types)
# Print the information
print("\nDescriptive Stats:\n", descriptive_stats)
print("\n Resulting Dataframe:", df.head())
```

- Resulting Terminal Output:

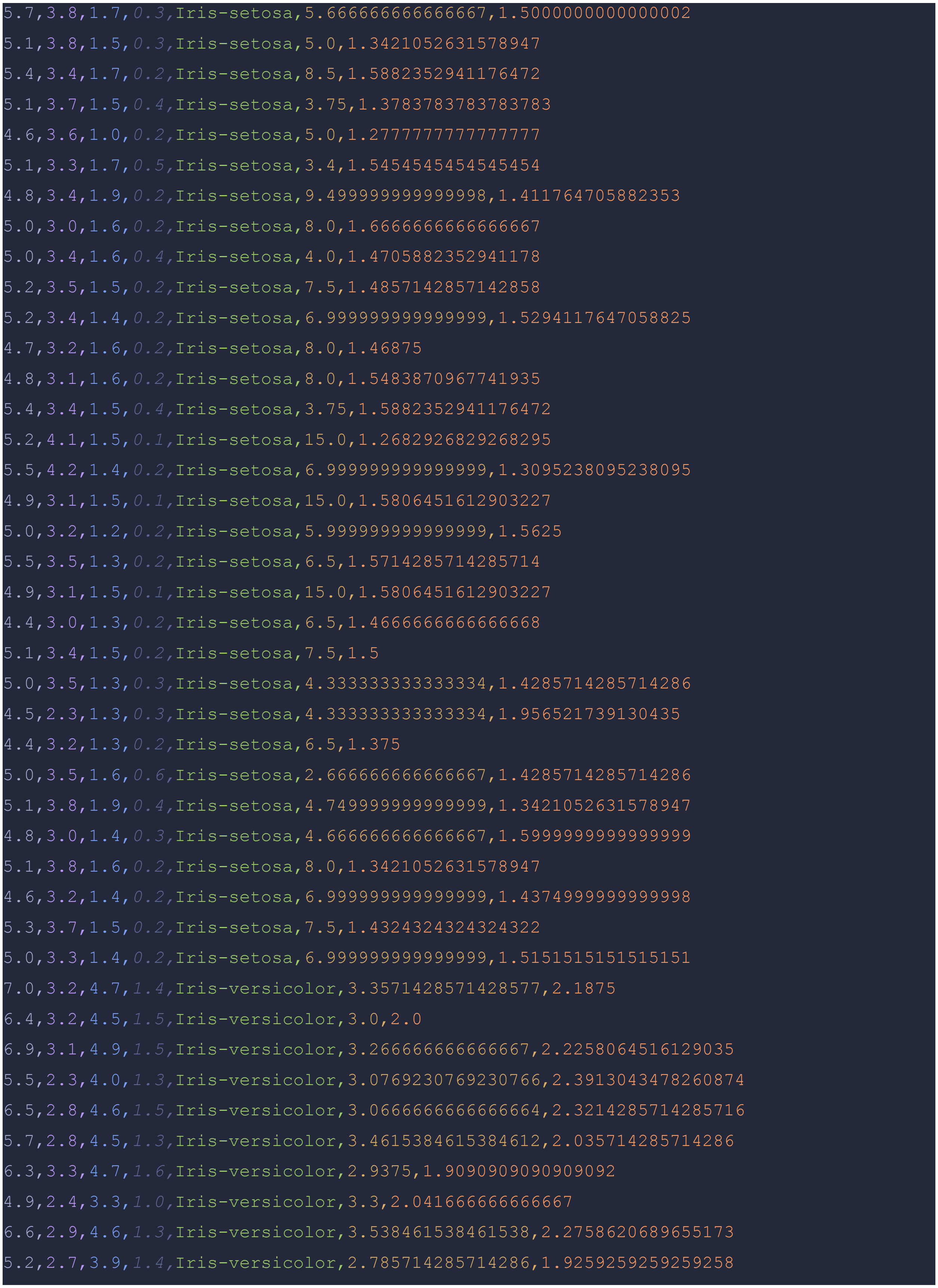


```
[ayandas@Ayans-Laptop iris-dataset % python loadIrisDataset.py
Headers: ['sepal length', 'sepal width', 'petal length', 'petal width', 'class']

Count of Data Objects: 150

Count of Data Categories: 3

Count of Each Category: class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64

Series Data Types: sepal length    float64
sepal width     float64
petal length    float64
petal width     float64
class            object
dtype: object

Descriptive Stats:
                 sepal length                      ... Sepal Ratio
                        mean       std  min  max  ...        mean       std       min       max
class                                             ...
Iris-setosa            5.006  0.352490  4.3  5.8  ...    1.474578  0.118693  1.268293  1.956522
Iris-versicolor        5.936  0.516171  4.9  7.0  ...    2.160402  0.228658  1.764706  2.818182
Iris-virginica         6.588  0.635880  4.9  7.9  ...    2.230453  0.246992  1.823529  2.961538

[3 rows x 24 columns]

 Resulting Dataframe:    sepal length  sepal width  petal length  petal width        class  Petal Ratio  Sepal Ratio
0           5.1          3.5           1.4          0.2  Iris-setosa          7.0     1.457143
1           4.9          3.0           1.4          0.2  Iris-setosa          7.0     1.633333
2           4.7          3.2           1.3          0.2  Iris-setosa          6.5     1.468750
3           4.6          3.1           1.5          0.2  Iris-setosa          7.5     1.483871
4           5.0          3.6           1.4          0.2  Iris-setosa          7.0     1.388889
ayandas@Ayans-Laptop iris-dataset % 
```

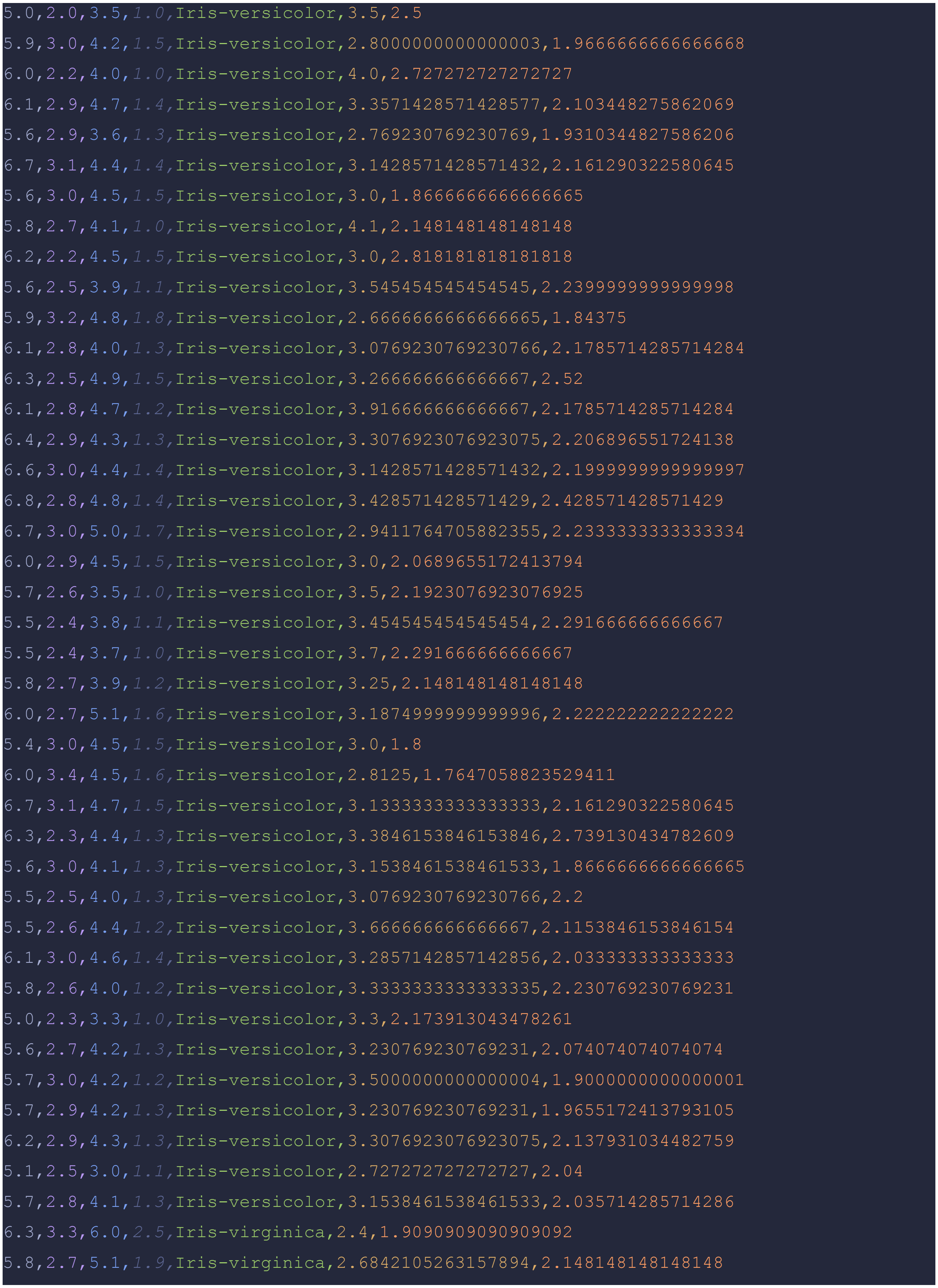
- Resulting generated csv file (The same as the previous generated file, but based on the python script rather than javascript)



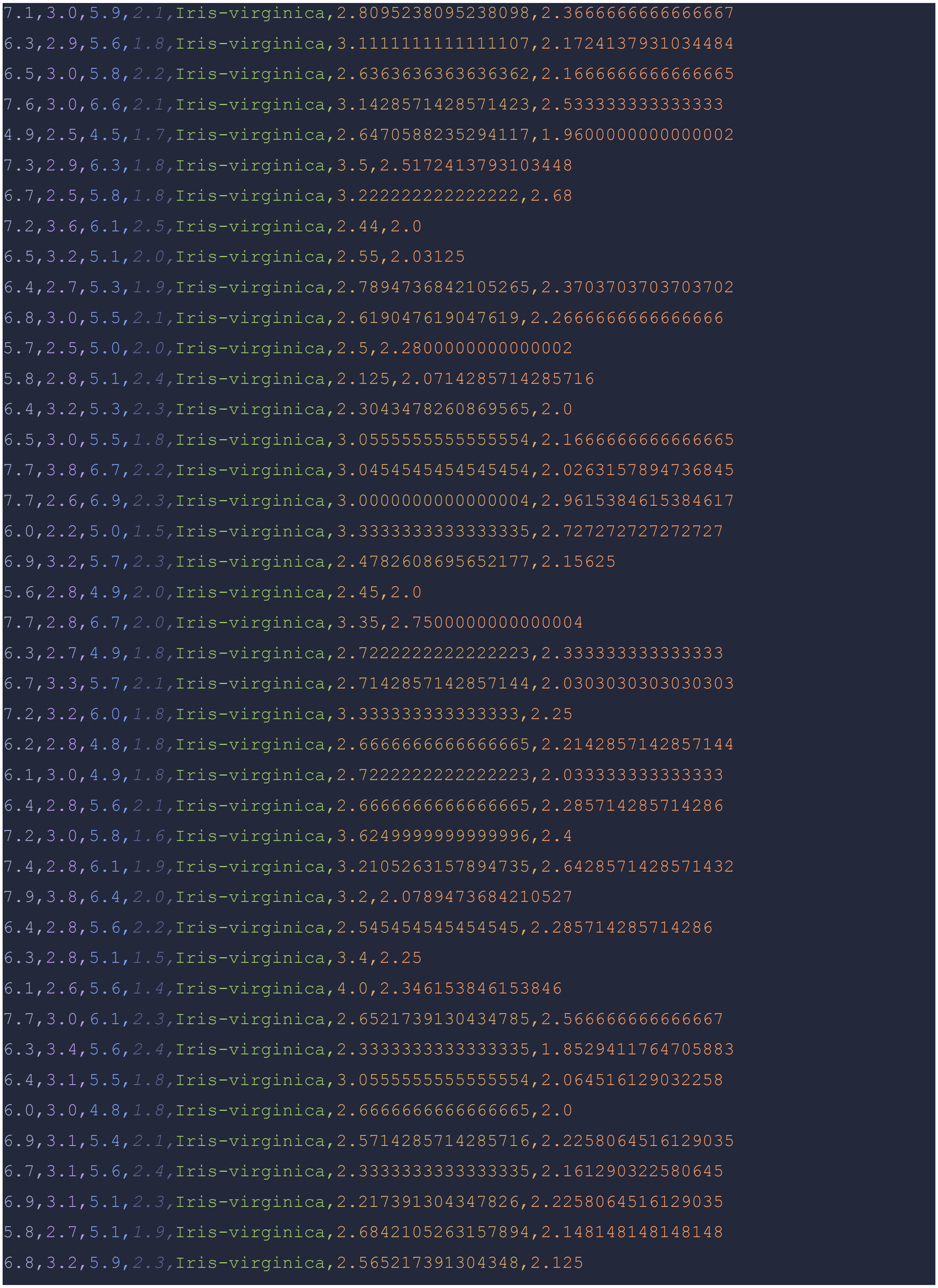
```
sepal length,sepal width,petal length,petal width,class,Petal Ratio,Sepal Ratio
5.1,3.5,1.4,0.2,Iris-setosa,6.999999999999999,1.457142857142857
4.9,3.0,1.4,0.2,Iris-setosa,6.999999999999999,1.6333333333333335
4.7,3.2,1.3,0.2,Iris-setosa,6.5,1.46875
4.6,3.1,1.5,0.2,Iris-setosa,7.5,1.4838709677419353
5.0,3.6,1.4,0.2,Iris-setosa,6.999999999999999,1.3888888888888888
5.4,3.9,1.7,0.4,Iris-setosa,4.25,1.3846153846153848
4.6,3.4,1.4,0.3,Iris-setosa,4.666666666666667,1.352941176470588
5.0,3.4,1.5,0.2,Iris-setosa,7.5,1.4705882352941178
4.4,2.9,1.4,0.2,Iris-setosa,6.999999999999999,1.517241379310345
4.9,3.1,1.5,0.1,Iris-setosa,15.0,1.5806451612903227
5.4,3.7,1.5,0.2,Iris-setosa,7.5,1.4594594594594594
4.8,3.4,1.6,0.2,Iris-setosa,8.0,1.411764705882353
4.8,3.0,1.4,0.1,Iris-setosa,13.999999999999998,1.5999999999999999
4.3,3.0,1.1,0.1,Iris-setosa,11.0,1.4333333333333333
5.8,4.0,1.2,0.2,Iris-setosa,5.999999999999999,1.45
5.7,4.4,1.5,0.4,Iris-setosa,3.75,1.2954545454545454
5.4,3.9,1.3,0.4,Iris-setosa,3.25,1.3846153846153848
5.1,3.5,1.4,0.3,Iris-setosa,4.666666666666667,1.457142857142857
```



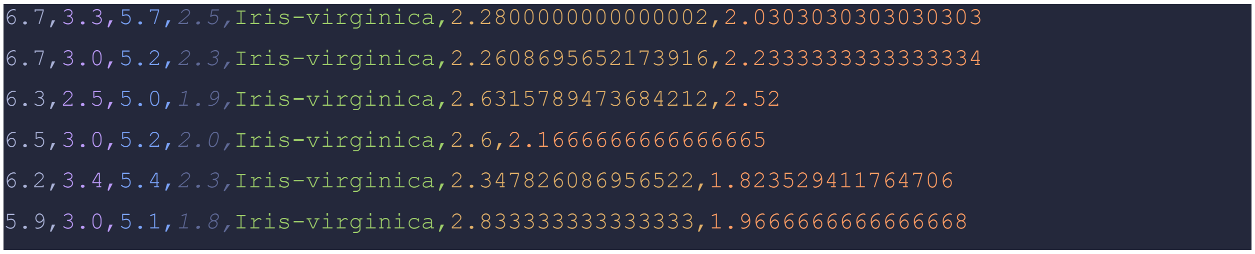
```
5.7,3.8,1.7,0.3,Iris-setosa,5.666666666666667,1.5000000000000002
5.1,3.8,1.5,0.3,Iris-setosa,5.0,1.3421052631578947
5.4,3.4,1.7,0.2,Iris-setosa,8.5,1.5882352941176472
5.1,3.7,1.5,0.4,Iris-setosa,3.75,1.3783783783783783
4.6,3.6,1.0,0.2,Iris-setosa,5.0,1.2777777777777777
5.1,3.3,1.7,0.5,Iris-setosa,3.4,1.5454545454545454
4.8,3.4,1.9,0.2,Iris-setosa,9.499999999999998,1.411764705882353
5.0,3.0,1.6,0.2,Iris-setosa,8.0,1.6666666666666667
5.0,3.4,1.6,0.4,Iris-setosa,4.0,1.4705882352941178
5.2,3.5,1.5,0.2,Iris-setosa,7.5,1.4857142857142858
5.2,3.4,1.4,0.2,Iris-setosa,6.999999999999999,1.5294117647058825
4.7,3.2,1.6,0.2,Iris-setosa,8.0,1.46875
4.8,3.1,1.6,0.2,Iris-setosa,8.0,1.5483870967741935
5.4,3.4,1.5,0.4,Iris-setosa,3.75,1.5882352941176472
5.2,4.1,1.5,0.1,Iris-setosa,15.0,1.2682926829268295
5.5,4.2,1.4,0.2,Iris-setosa,6.999999999999999,1.3095238095238095
4.9,3.1,1.5,0.1,Iris-setosa,15.0,1.5806451612903227
5.0,3.2,1.2,0.2,Iris-setosa,5.999999999999999,1.5625
5.5,3.5,1.3,0.2,Iris-setosa,6.5,1.5714285714285714
4.9,3.1,1.5,0.1,Iris-setosa,15.0,1.5806451612903227
4.4,3.0,1.3,0.2,Iris-setosa,6.5,1.4666666666666668
5.1,3.4,1.5,0.2,Iris-setosa,7.5,1.5
5.0,3.5,1.3,0.3,Iris-setosa,4.333333333333334,1.4285714285714286
4.5,2.3,1.3,0.3,Iris-setosa,4.333333333333334,1.956521739130435
4.4,3.2,1.3,0.2,Iris-setosa,6.5,1.375
5.0,3.5,1.6,0.6,Iris-setosa,2.666666666666667,1.4285714285714286
5.1,3.8,1.9,0.4,Iris-setosa,4.749999999999999,1.3421052631578947
4.8,3.0,1.4,0.3,Iris-setosa,4.666666666666667,1.5999999999999999
5.1,3.8,1.6,0.2,Iris-setosa,8.0,1.3421052631578947
4.6,3.2,1.4,0.2,Iris-setosa,6.999999999999999,1.4374999999999998
5.3,3.7,1.5,0.2,Iris-setosa,7.5,1.4324324324324322
5.0,3.3,1.4,0.2,Iris-setosa,6.999999999999999,1.5151515151515151
7.0,3.2,4.7,1.4,Iris-versicolor,3.3571428571428577,2.1875
6.4,3.2,4.5,1.5,Iris-versicolor,3.0,2.0
6.9,3.1,4.9,1.5,Iris-versicolor,3.266666666666667,2.225806451612903 5
5.5,2.3,4.0,1.3,Iris-versicolor,3.0769230769230766,2.3913043478260874
6.5,2.8,4.6,1.5,Iris-versicolor,3.0666666666666664,2.3214285714285716
5.7,2.8,4.5,1.3,Iris-versicolor,3.4615384615384612,2.035714285714286
6.3,3.3,4.7,1.6,Iris-versicolor,2.9375,1.9090909090909092
4.9,2.4,3.3,1.0,Iris-versicolor,3.3,2.0416666666666667
6.6,2.9,4.6,1.3,Iris-versicolor,3.538461538461538,2.2758620689655173
5.2,2.7,3.9,1.4,Iris-versicolor,2.785714285714286,1.9259259259259258
```



```
5.0,2.0,3.5,1.0,Iris-versicolor,3.5,2.5
5.9,3.0,4.2,1.5,Iris-versicolor,2.8000000000000003,1.9666666666666668
6.0,2.2,4.0,1.0,Iris-versicolor,4.0,2.727272727272727
6.1,2.9,4.7,1.4,Iris-versicolor,3.3571428571428577,2.103448275862069
5.6,2.9,3.6,1.3,Iris-versicolor,2.769230769230769,1.9310344827586206
6.7,3.1,4.4,1.4,Iris-versicolor,3.1428571428571432,2.161290322580645
5.6,3.0,4.5,1.5,Iris-versicolor,3.0,1.8666666666666665
5.8,2.7,4.1,1.0,Iris-versicolor,4.1,2.148148148148148
6.2,2.2,4.5,1.5,Iris-versicolor,3.0,2.818181818181818
5.6,2.5,3.9,1.1,Iris-versicolor,3.545454545454545,2.2399999999999998
5.9,3.2,4.8,1.8,Iris-versicolor,2.6666666666666665,1.84375
6.1,2.8,4.0,1.3,Iris-versicolor,3.0769230769230766,2.1785714285714284
6.3,2.5,4.9,1.5,Iris-versicolor,3.266666666666667,2.52
6.1,2.8,4.7,1.2,Iris-versicolor,3.916666666666667,2.1785714285714284
6.4,2.9,4.3,1.3,Iris-versicolor,3.3076923076923075,2.206896551724138
6.6,3.0,4.4,1.4,Iris-versicolor,3.1428571428571432,2.1999999999999997
6.8,2.8,4.8,1.4,Iris-versicolor,3.428571428571429,2.428571428571429
6.7,3.0,5.0,1.7,Iris-versicolor,2.9411764705882355,2.2333333333333334
6.0,2.9,4.5,1.5,Iris-versicolor,3.0,2.0689655172413794
5.7,2.6,3.5,1.0,Iris-versicolor,3.5,2.1923076923076925
5.5,2.4,3.8,1.1,Iris-versicolor,3.454545454545454,2.291666666666667
5.5,2.4,3.7,1.0,Iris-versicolor,3.7,2.291666666666667
5.8,2.7,3.9,1.2,Iris-versicolor,3.25,2.148148148148148
6.0,2.7,5.1,1.6,Iris-versicolor,3.1874999999999996,2.222222222222222
5.4,3.0,4.5,1.5,Iris-versicolor,3.0,1.8
6.0,3.4,4.5,1.6,Iris-versicolor,2.8125,1.7647058823529411
6.7,3.1,4.7,1.5,Iris-versicolor,3.1333333333333333,2.161290322580645
6.3,2.3,4.4,1.3,Iris-versicolor,3.3846153846153846,2.739130434782609
5.6,3.0,4.1,1.3,Iris-versicolor,3.1538461538461533,1.8666666666666665
5.5,2.5,4.0,1.3,Iris-versicolor,3.0769230769230766,2.2
5.5,2.6,4.4,1.2,Iris-versicolor,3.666666666666667,2.1153846153846154
6.1,3.0,4.6,1.4,Iris-versicolor,3.2857142857142856,2.033333333333333
5.8,2.6,4.0,1.2,Iris-versicolor,3.3333333333333335,2.230769230769231
5.0,2.3,3.3,1.0,Iris-versicolor,3.3,2.173913043478261
5.6,2.7,4.2,1.3,Iris-versicolor,3.230769230769231,2.074074074074074
5.7,3.0,4.2,1.2,Iris-versicolor,3.5000000000000004,1.9000000000000001
5.7,2.9,4.2,1.3,Iris-versicolor,3.230769230769231,1.9655172413793105
6.2,2.9,4.3,1.3,Iris-versicolor,3.3076923076923075,2.137931034482759
5.1,2.5,3.0,1.1,Iris-versicolor,2.727272727272727,2.04
5.7,2.8,4.1,1.3,Iris-versicolor,3.1538461538461533,2.035714285714286
6.3,3.3,6.0,2.5,Iris-virginica,2.4,1.9090909090909092
5.8,2.7,5.1,1.9,Iris-virginica,2.6842105263157894,2.148148148148148
```



```
7.1,3.0,5.9,2.1,Iris-virginica,2.8095238095238098,2.3666666666666667
6.3,2.9,5.6,1.8,Iris-virginica,3.1111111111111107,2.1724137931034484
6.5,3.0,5.8,2.2,Iris-virginica,2.6363636363636362,2.1666666666666665
7.6,3.0,6.6,2.1,Iris-virginica,3.1428571428571423,2.533333333333333
4.9,2.5,4.5,1.7,Iris-virginica,2.6470588235294117,1.9600000000000002
7.3,2.9,6.3,1.8,Iris-virginica,3.5,2.5172413793103448
6.7,2.5,5.8,1.8,Iris-virginica,3.2222222222222222,2.68
7.2,3.6,6.1,2.5,Iris-virginica,2.44,2.0
6.5,3.2,5.1,2.0,Iris-virginica,2.55,2.03125
6.4,2.7,5.3,1.9,Iris-virginica,2.7894736842105265,2.3703703703703702
6.8,3.0,5.5,2.1,Iris-virginica,2.619047619047619,2.2666666666666666
5.7,2.5,5.0,2.0,Iris-virginica,2.5,2.2800000000000002
5.8,2.8,5.1,2.4,Iris-virginica,2.125,2.0714285714285716
6.4,3.2,5.3,2.3,Iris-virginica,2.3043478260869565,2.0
6.5,3.0,5.5,1.8,Iris-virginica,3.0555555555555554,2.1666666666666665
7.7,3.8,6.7,2.2,Iris-virginica,3.0454545454545454,2.0263157894736845
7.7,2.6,6.9,2.3,Iris-virginica,3.0000000000000004,2.9615384615384617
6.0,2.2,5.0,1.5,Iris-virginica,3.3333333333333335,2.727272727272727
6.9,3.2,5.7,2.3,Iris-virginica,2.4782608695652177,2.15625
5.6,2.8,4.9,2.0,Iris-virginica,2.45,2.0
7.7,2.8,6.7,2.0,Iris-virginica,3.35,2.7500000000000004
6.3,2.7,4.9,1.8,Iris-virginica,2.7222222222222223,2.333333333333333
6.7,3.3,5.7,2.1,Iris-virginica,2.7142857142857144,2.0303030303030303
7.2,3.2,6.0,1.8,Iris-virginica,3.333333333333333,2.25
6.2,2.8,4.8,1.8,Iris-virginica,2.6666666666666665,2.2142857142857144
6.1,3.0,4.9,1.8,Iris-virginica,2.7222222222222223,2.033333333333333
6.4,2.8,5.6,2.1,Iris-virginica,2.6666666666666665,2.285714285714286
7.2,3.0,5.8,1.6,Iris-virginica,3.6249999999999996,2.4
7.4,2.8,6.1,1.9,Iris-virginica,3.2105263157894735,2.6428571428571432
7.9,3.8,6.4,2.0,Iris-virginica,3.2,2.0789473684210527
6.4,2.8,5.6,2.2,Iris-virginica,2.545454545454545,2.285714285714286
6.3,2.8,5.1,1.5,Iris-virginica,3.4,2.25
6.1,2.6,5.6,1.4,Iris-virginica,4.0,2.346153846153846
7.7,3.0,6.1,2.3,Iris-virginica,2.6521739130434785,2.566666666666667
6.3,3.4,5.6,2.4,Iris-virginica,2.3333333333333335,1.8529411764705883
6.4,3.1,5.5,1.8,Iris-virginica,3.0555555555555554,2.064516129032258
6.0,3.0,4.8,1.8,Iris-virginica,2.6666666666666665,2.0
6.9,3.1,5.4,2.1,Iris-virginica,2.5714285714285716,2.2258064516129035
6.7,3.1,5.6,2.4,Iris-virginica,2.3333333333333335,2.161290322580645
6.9,3.1,5.1,2.3,Iris-virginica,2.217391304347826,2.2258064516129035
5.8,2.7,5.1,1.9,Iris-virginica,2.6842105263157894,2.148148148148148
6.8,3.2,5.9,2.3,Iris-virginica,2.565217391304348,2.125
```



```
6.7,3.3,5.7,2.5,Iris-virginica,2.2800000000000002,2.0303030303030303
6.7,3.0,5.2,2.3,Iris-virginica,2.260869565217391,2.2333333333333334
6.3,2.5,5.0,1.9,Iris-virginica,2.6315789473684212,2.52
6.5,3.0,5.2,2.0,Iris-virginica,2.6,2.1666666666666665
6.2,3.4,5.4,2.3,Iris-virginica,2.347826086956522,1.823529411764706
5.9,3.0,5.1,1.8,Iris-virginica,2.833333333333333,1.9666666666666668
```

3. Built in plotting code

Note: Although initially I wanted to do the data visualization in javascript, to save time, I resorted to using python, although preferably I would have used javascript had I had more time, will do so for future homeworks/projects.

Part a:
Reason for Choosing Boxplot:

- The boxplot is chosen for visualizing the distribution of the petal and sepal ratio for each category (Iris species) because it is an effective way to display the data's central tendency, variability, and the presence of outliers within each category. Boxplots summarize the distribution of a continuous variable for different groups, making them ideal for comparing these distributions across several categories. They show the median, quartiles, and outliers, providing a clear visual summary of the data's spread and central tendency.

Summary of the Plot:

- The plot displays the distribution of petal ratios and sepal ratios across different Iris species (categories). Each boxplot represents one species and shows the median (the central line in the box), the interquartile range (the box's length), and any potential outliers (points outside the whiskers). This visualization helps in understanding how the petal and sepal ratios vary between species, indicating significant differences in these ratios among the Iris species. For example, one species might have a noticeably higher petal ratio than the others, highlighted by its boxplot's position and spread. Similarly, the sepal ratio distribution could reveal another aspect of the morphological differences between species.

Part b:
Summary of the Plot

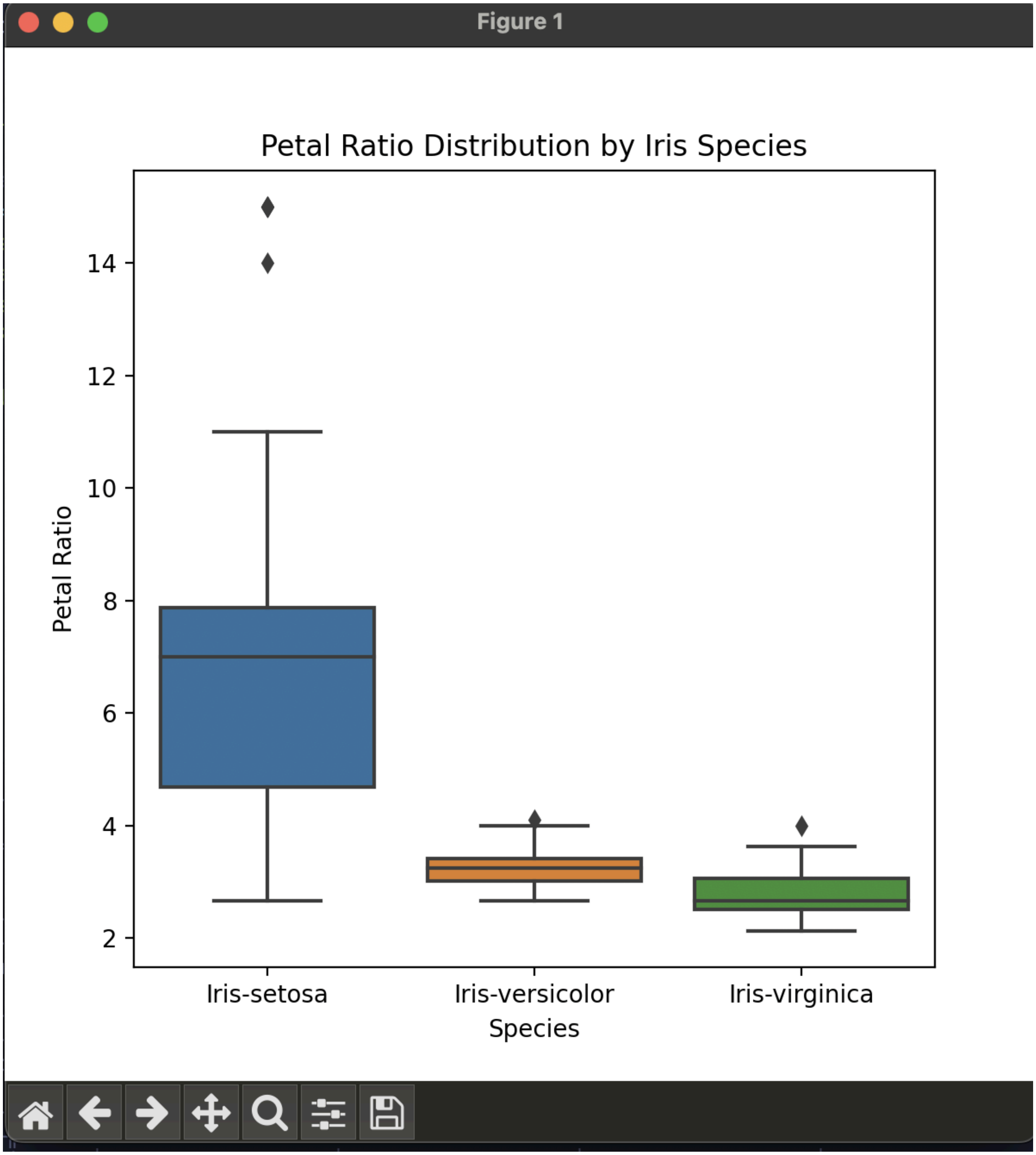
- The scatter plot generated using `sns.pairplot` with the `hue='class'` argument visualizes the relationships between the original features (sepal length, sepal width, petal length, and petal width) for each Iris species. By grouping points by color based on their species, this plot provides insights into the correlations between these features within each category and across different categories. It allows for the observation of patterns such as the linear or non-linear relationships between features, clusters of species based on the features, and the separation between species in terms of feature values. For instance, the scatter plot could show that one species tends to have larger petal lengths but narrower sepal widths, while another displays the opposite trend. This visualization is invaluable for exploratory data analysis, offering a comprehensive overview of how these original features interrelate and contribute to species differentiation.

Code and Resulting Diagrams:

- **Petal_ratio_distribution.py**

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load data
df = pd.read_csv('updated_dataset-iris.csv')

# Plotting Petal Ratio Distribution
plt.figure(figsize=(6, 6))
sns.boxplot(x='class', y='Petal Ratio', data=df)
plt.title('Petal Ratio Distribution by Iris Species')
plt.xlabel('Species')
plt.ylabel('Petal Ratio')

plt.savefig('visualization/petal_ratio_distribution.png')
plt.show()
```

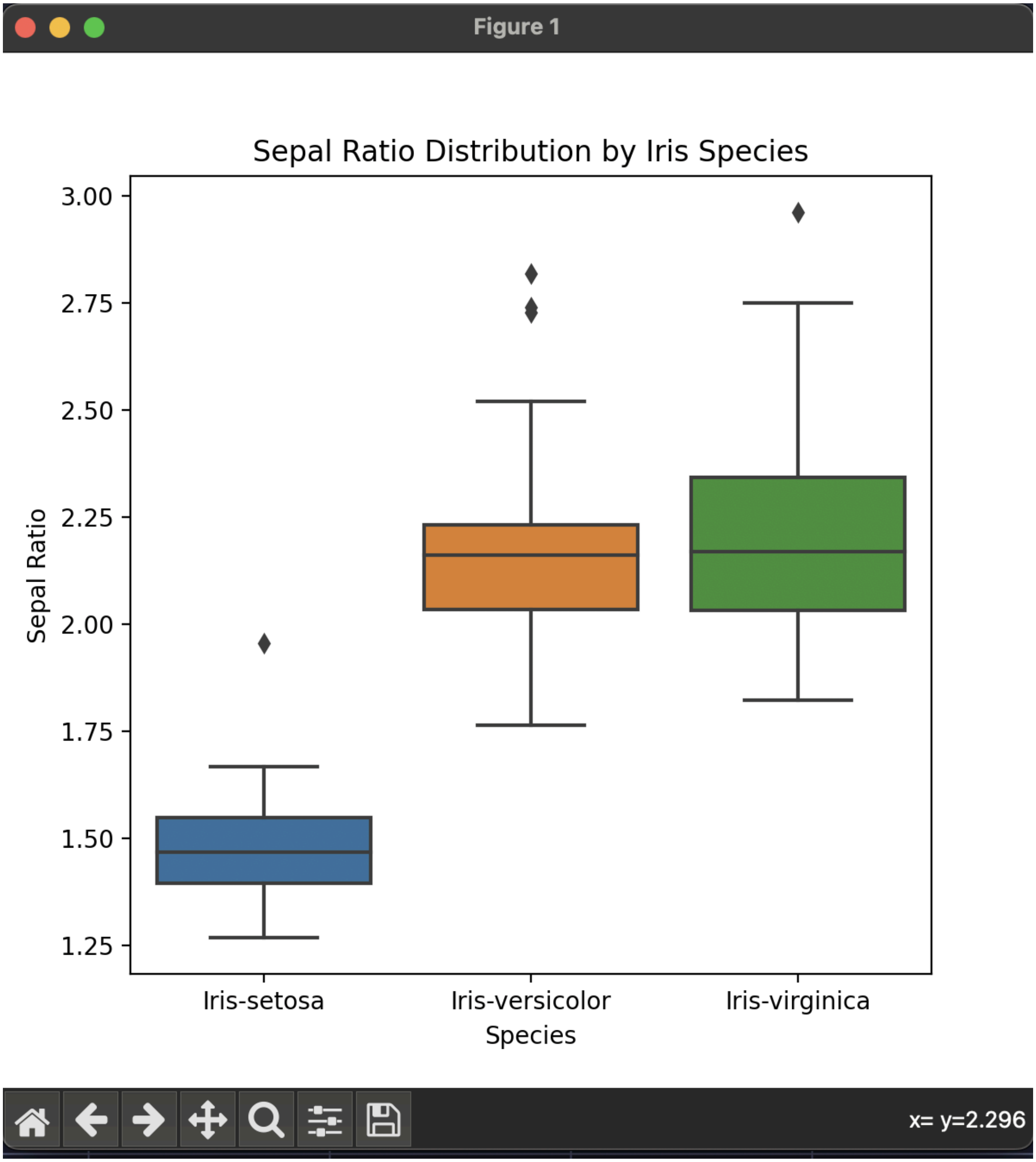
Resulting Output:



- **sepalRatioDistribution.py**

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load data
df = pd.read_csv('updated_dataset-iris.csv')

# Plotting Sepal Ratio Distribution
plt.figure(figsize=(6, 6))
sns.boxplot(x='class', y='Sepal Ratio', data=df)
plt.title('Sepal Ratio Distribution by Iris Species')
plt.xlabel('Species')
plt.ylabel('Sepal Ratio')

plt.savefig('visualization/sepal_ratio_distribution.png')
plt.show()
```

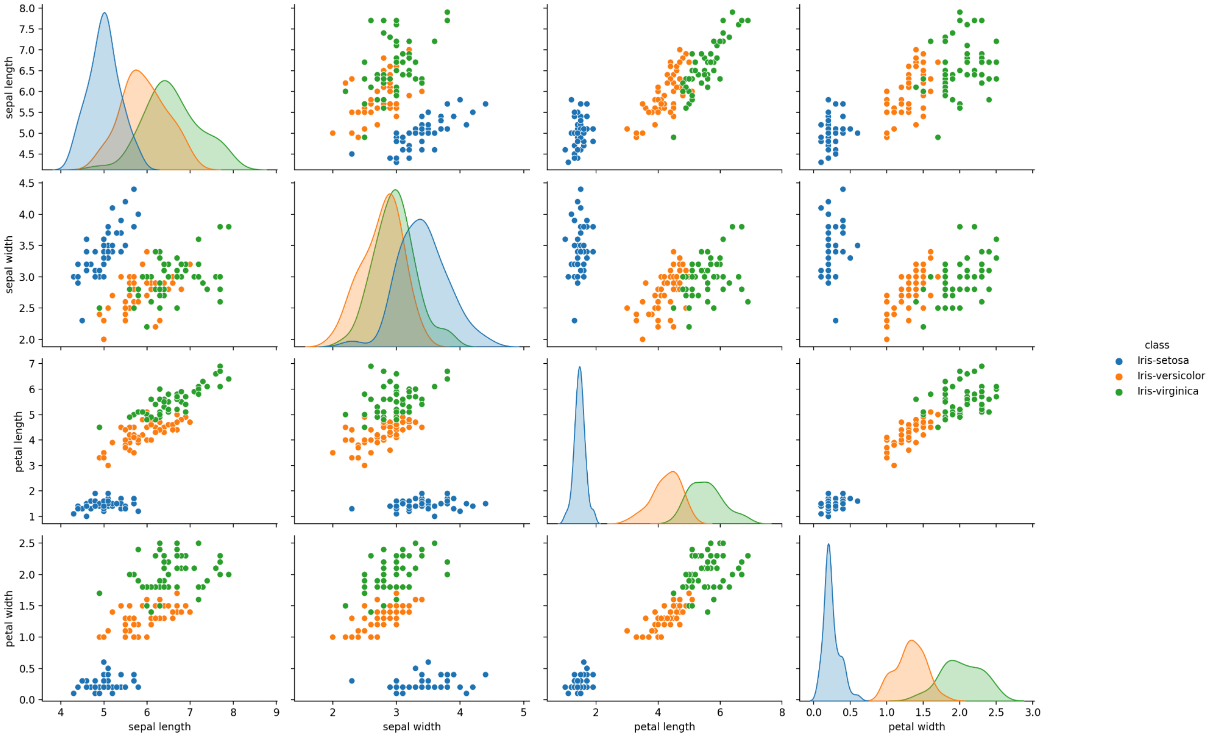
- Resulting Output:



- **scatter_plot_original_features.py**

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load data
df = pd.read_csv('updated_dataset-iris.csv')

# Scatter Plot of Original Features
sns.pairplot(df, hue='class', vars=['sepal length', 'sepal width', 'petal length',
'petal width'])
plt.suptitle('Scatter Plot of Original Features by Iris Species', y=1.02)

plt.savefig('visualization/scatter_plot_original_features.png')
plt.show()
```

- Resulting Output:



Note 2: The generated graphs can also be found within the visualization directory.