NETFLIX ETL

Paso 1: Configurar el Proyecto

1. **Inicializar un nuevo proyecto Node.js:**

```
mkdir netflix-etl
cd netflix-etl
npm init -y
```

2. **Instalar dependencias necesarias:**

```
npm install express mongodb csv-parser ejs
```

Paso 2: Crear el Script ETL en Node.js

Vamos a crear un script en Node.js para extraer, transformar y cargar los datos en MongoDB.

`etl.js`

```
const fs = require('fs');
const csv = require('csv-parser');
const { MongoClient } = require('mongodb');
const path = require('path');

// Ruta del archivo CSV
const filePath = path.join(__dirname, 'netflix_titles.csv');

// Conexión a MongoDB
const url = 'mongodb://localhost:27017';
const client = new MongoClient(url);
const dbName = 'netflix';
let db;

async function runETL() {
  await client.connect();
  db = client.db(dbName);
  const collection = db.collection('titles');

  // Limpiar la colección antes de cargar nuevos datos
  await collection.deleteMany({});

  const insertPromises = [];

  // Leer y transformar datos del CSV
  fs.createReadStream(filePath)
    .pipe(csv())
    .on('data', (row) => {
      // Transformar la duración a minutos y calcular la antigüedad del contenido
      if (row.duration.includes('Season')) {
        row.duration = parseInt(row.duration.split(' ')[0]) * 60; // Aproximación
de 60 minutos por temporada
```

```
      } else {
        row.duration = parseInt(row.duration.split(' ')[0]);
      }
      row.date_added = new Date(row.date_added);
      row.content_age = new Date().getFullYear() - parseInt(row.release_year);

      // Agregar la operación de inserción a la lista de promesas
      insertPromises.push(collection.insertOne(row));
    })
    .on('end', async () => {
      // Esperar a que todas las inserciones se completen
      try {
        await Promise.all(insertPromises);
        console.log('Datos cargados en MongoDB exitosamente.');
      } catch (error) {
        console.error('Error durante la inserción:', error);
      } finally {
        await client.close();
      }
    });
}

runETL().catch(console.error);
```

Paso 3: Crear el Servidor Express

Vamos a crear un servidor Express que servirá una vista para mostrar los datos cargados.

`index.js`

```
const express = require('express');
const { MongoClient } = require('mongodb');
const path = require('path');
const app = express();
const port = 3000;

// Configurar EJS como motor de plantillas
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

// Conexión a MongoDB
const url = 'mongodb://localhost:27017';
const client = new MongoClient(url);
const dbName = 'netflix';
let db;

async function connectDB() {
  await client.connect();
  db = client.db(dbName);
}

connectDB().catch(console.error);

// Ruta para la vista principal
app.get('/', async (req, res) => {
  const collection = db.collection('titles');
```

```
  const data = await collection.find().toArray();
  res.render('index', { data });
});

app.listen(port, () => {
  console.log(`Servidor corriendo en http://localhost:${port}`);
});
```

Paso 4: Crear la Vista

Vamos a crear una vista para mostrar los datos en una tabla.

`views/index.ejs`

```
<!DOCTYPE html>
<html>
<head>
  <title>Netflix Titles</title>
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
</head>
<body>
  <div class="container">
    <h1 class="mt-5">Netflix Titles</h1>
    <table class="table table-bordered mt-3">
      <thead>
        <tr>
          <th>Title</th>
          <th>Type</th>
          <th>Director</th>
          <th>Cast</th>
          <th>Country</th>
          <th>Date Added</th>
          <th>Release Year</th>
          <th>Rating</th>
          <th>Duration</th>
          <th>Listed In</th>
          <th>Description</th>
        </tr>
      </thead>
      <tbody>
        <% data.forEach(function(item) { %>
          <tr>
            <td><%= item.title %></td>
            <td><%= item.type %></td>
            <td><%= item.director %></td>
            <td><%= item.cast %></td>
            <td><%= item.country %></td>
            <td><%= item.date_added.toISOString().split('T')[0] %></td>
            <td><%= item.release_year %></td>
            <td><%= item.rating %></td>
            <td><%= item.duration %> minutes</td>
            <td><%= item.listed_in %></td>
            <td><%= item.description %></td>
```

```
            </tr>
          <% }); %>
        </tbody>
      </table>
    </div>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
</body>
</html>
```

Paso 5: Ejecutar el Script ETL y el Servidor

1. **Ejecutar el script ETL:**

   ```
   node etl.js
   ```

2. **Ejecutar el servidor Express:**

   ```
   node index.js
   ```

Verificar

Abre tu navegador y ve a `http://localhost:3000` para ver los datos de Netflix en una tabla.

Extensiones y Mejoras

- **Añadir Paginación:** Implementa paginación para manejar grandes volúmenes de datos.
- **Visualización:** Añadir gráficos y otros elementos visuales para analizar los datos.
- **Filtros de Búsqueda:** Añadir filtros para buscar y filtrar títulos por diferentes criterios (por ejemplo, por año de lanzamiento, género, etc.).

Dependencias

1. **Express:**

   - **Propósito:** `express` es un framework web para Node.js que facilita la creación de aplicaciones web y APIs. Es ligero y flexible, proporcionando un robusto conjunto de características para aplicaciones web y móviles.
   - **Uso:** Se utiliza para crear el servidor web, definir rutas, manejar peticiones HTTP, etc.

   ```
   npm install express
   ```

   Ejemplo de uso:

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(port, () => {
  console.log(`Servidor corriendo en http://localhost:${port}`);
});
```

2. **MongoDB:**

   - **Propósito:** `mongodb` es el driver oficial de MongoDB para Node.js. Proporciona una interfaz para interactuar con bases de datos MongoDB desde una aplicación Node.js.
   - **Uso:** Se utiliza para conectar con MongoDB, realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y manejar datos en la base de datos.

```
npm install mongodb
```

Ejemplo de uso:

```
const { MongoClient } = require('mongodb');
const url = 'mongodb://localhost:27017';
const client = new MongoClient(url);

async function connectDB() {
  await client.connect();
  const db = client.db('mydatabase');
  console.log('Conectado a MongoDB');
}

connectDB().catch(console.error);
```

3. **csv-parser:**

   - **Propósito:** `csv-parser` es una librería para analizar (parsear) archivos CSV en Node.js de manera eficiente y sencilla.
   - **Uso:** Se utiliza para leer y transformar datos desde archivos CSV, facilitando su procesamiento y manipulación.

```
npm install csv-parser
```

Ejemplo de uso:

```
const fs = require('fs');
const csv = require('csv-parser');

fs.createReadStream('file.csv')
  .pipe(csv())
  .on('data', (row) => {
    console.log(row);
```

```
  })
  .on('end', () => {
    console.log('CSV file successfully processed');
  });
```

4. **EJS:**

   - **Propósito:** `ejs` (Embedded JavaScript Templating) es un motor de plantillas que permite generar HTML con JavaScript simple.
   - **Uso:** Se utiliza para renderizar vistas dinámicas en el servidor, permitiendo la inclusión de datos de manera dinámica en las plantillas HTML.

```
npm install ejs
```

Ejemplo de uso:

```
app.set('view engine', 'ejs');

app.get('/', (req, res) => {
  res.render('index', { title: 'Mi Título', message: 'Hola Mundo' });
});
```

`fs` (File System)

- **Propósito:** `fs` es un módulo nativo de Node.js que proporciona una API para interactuar con el sistema de archivos. Permite leer, escribir, actualizar y eliminar archivos y directorios.
- **Uso:** Se utiliza en este proyecto para leer el archivo CSV de Netflix.

Ejemplo de uso:

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

// Leer un archivo
fs.readFile('path/to/file', 'utf8', (err, data) => {
  if (err) {
    console.error(err);
    return;
  }
  console.log(data);
});

// Escribir en un archivo
fs.writeFile('path/to/file', 'Hello, world!', (err) => {
  if (err) {
    console.error(err);
    return;
  }
  console.log('File has been written');
});
```

En el contexto de este proyecto, `fs` se utiliza junto con `csv-parser` para leer y procesar el archivo CSV de Netflix, transformando y cargando los

datos en MongoDB.