Source Code
Notable Court Cases

Below is the Google Apps Script (GAS) source code for Prairie HR Solutions “Notable Court Cases” UI Tool. This source code is intended to be used with a dedicated Google Sheet, and the code is bound to that sheet. The assumption is that individuals who downloaded this source code already have a functional understanding of GAS and HTML. Organizations should always consult with their IT functions for proper usage and security procedures.

How it works-

1. User loads the webapp into their browser, which then populates the first parent dropdown selection.
2. A user makes a selection, the selection is sent to the server side (webapp ui) to”
    a. Identify the appropriate tab labeled the same as the selections.
    b. Pull the data for the secondary selection dropdown (child), and then it returns back to the webapp to populate.
    c. At the same time, the server side is sending the whole data set for the appropriate tab to the client side. This will populate all the data by default, the second dropdown acts somewhat as a filter for that data.
    d. Once a user selects an option from the secondary or child dropdown, the table updates with only the items that match that selection.

Key notes to remember -

1. You will need your own Google Sheet in which to “bind” the GAS.
2. You will need to go through the proper authorizations for publishing.
3. There is logic built in for links to not display that column, but allow it to be “linkable” should there be content in the “link” column.

Code GS File

```
 */


function doGet(){

  return HtmlService.createHtmlOutputFromFile("Web Page File Name You
Choose").setTitle("Prairie HR Solutions Landmark Case
Tool").setXFrameOptionsMode(HtmlService.XFrameOptionsMode.ALLOWALL);
}

// Function to get all sheet names
function getSheetNames() {
  const sheets = SpreadsheetApp.getActiveSpreadsheet().getSheets();
  return sheets.map(sheet => sheet.getName());
}

// Function to get unique "Key Issue" values from a specific sheet
function getKeyIssues(sheetName) {
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(sheetName);
  if (!sheet) return [];
  const data = sheet.getDataRange().getValues();
  const columnIndex = data[0].indexOf('Key Issue'); // Assuming the header contains "Key Issue"
  if (columnIndex === -1) return [];
  return [...new Set(data.slice(1).map(row => row[columnIndex]))]; // Unique values, excluding
header
}

// Function to get filtered data by "Key Issue" or all data if "All Key Issues" is selected
function getFilteredData(sheetName, keyIssue) {
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(sheetName);
  if (!sheet) return [];
  const data = sheet.getDataRange().getValues();
  const columnIndex = data[0].indexOf('Key Issue'); // Assuming the header contains "Key Issue"
  if (keyIssue === "All Key Issues" || columnIndex === -1) {
    return data; // Return all data if "All Key Issues" is selected or column is missing
  }
  return [data[0], ...data.slice(1).filter(row => row[columnIndex] === keyIssue)];
}
```

Web Page File

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Prairie HR Solutions, LLC Landmark Employment Cases</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+A
LEwIH" crossorigin="anonymous">
  </head>
  </head>
  <body>
    <style>
th {
  background: #551E19 !important;
  color: white !important;
  text-align: center;
}
.border-phr {
  border-style: solid;
  border-color: #551E19;
}
.table-hover thead tr:hover, .table-hover tbody tr:hover td {
    background-color: #C56E33;
    color: #fff;
}
h1, h2, h3 {
  text-align: center;
  color: #551E19;
}
table tr.clickable-row {
  cursor: pointer;
}
  .form-control:focus, .form-select:focus {
  border-color: #551E19;
  box-shadow: inset 0 1px 1px rgba(85, 30, 25, 0.075), 0 0 8px rgba(197, 110, 51, 0.6);
}
.container {
  border: solid;
  border-color: #551E19;
}
.iframe-container {
```

```
  overflow: hidden;
  padding-top: 56.25%; /* 16:9 */
  position: relative;
}

.iframe-container iframe {
  position: absolute;
  top: 0;
  left: 0;
  border: 0;
  width: 100%;
  height: 100%;
}
    </style>
  </head>
  <body>

    <div class="container">
    <h1>Prairie HR Solutions</h1>
    <h2>Notable Employment Case Reference</h2>
    <h3 class="mb-3">Filter Data by Region and Key Issue</h3>
    <p>Prairie HR Solutions consistently reads cases concerning employment matters, to identify employment trends, and assist in the risk analysis process concerning HR situations. We have created a webapp through Google Apps Script as a reference guide. We believe that knowledge should be shared, and will update this free reference as we encounter situations and references. Prairie HR Solutions is not a law firm and nothing that we provide as a reading resource is not to ever be construed as legal advice or interpretation. Always consult an attorney before making decisions with legal impact. <br><br>Artificial Intelligence (AI) platforms were and are utilized to locate specific cases and provide high level information. All data provided is publically available and unaltered from the respective websource. AI can make mistakes, and any information in this database should be verified and consulted by an attorney.<br><br>To use this reference, please select the appropriate state or region, then the appropriate key issue. By not selecting a key issue, all available references will be populated. This database will continuously be updated and added to, and we hope it can assist in navigating the People Management space. The intent of this webapp is to provide a free resource, however, if you feel your organization can benefit from the solution the webapp provides with your organization's data, please reach out to us.</p>
  <hr>
    <label for="sheetSelector">Select State or Region:</label>
    <select class="form-select mb-3" id="sheetSelector" onchange="loadKeyIssues()"></select>

    <label for="keyIssueSelector">Select Key Issue:</label>
    <select class="form-select mb-3" id="keyIssueSelector"
onchange="loadFilteredData()"></select>
```

```
    <table class="table table-striped table-hover table-bordered border-phr"
id="table-container"></table>
    </div><!--container-->

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIe
Hz" crossorigin="anonymous"></script>
    <script>
 // Populate the first dropdown with sheet names
      google.script.run.withSuccessHandler(populateSheetDropdown).getSheetNames();

      function populateSheetDropdown(sheetNames) {
        const sheetDropdown = document.getElementById('sheetSelector');
        sheetDropdown.innerHTML = '<option value="">--Select a Region--</option>';
        sheetNames.forEach(name => {
          const option = document.createElement('option');
          option.value = name;
          option.textContent = name;
          sheetDropdown.appendChild(option);
        });
      }

      // Populate the second dropdown with unique key issues from the selected sheet
      function loadKeyIssues() {
        const selectedSheet = document.getElementById('sheetSelector').value;
        if (selectedSheet) {
          google.script.run.withSuccessHandler(keyIssues => {
            populateKeyIssueDropdown(keyIssues);
            loadFilteredData(); // Automatically load all data for the selected sheet
          }).getKeyIssues(selectedSheet);
        } else {
          document.getElementById('keyIssueSelector').innerHTML = '<option value="">--Select a
Key Issue--</option>';
          document.getElementById('table-container').innerHTML = '';
        }
      }

      function populateKeyIssueDropdown(keyIssues) {
        const keyIssueDropdown = document.getElementById('keyIssueSelector');
        keyIssueDropdown.innerHTML = '<option value="All Key Issues">--All Key
Issues--</option>'; // Default option
        keyIssues.forEach(issue => {
          const option = document.createElement('option');
```

```
        option.value = issue;
        option.textContent = issue;
        keyIssueDropdown.appendChild(option);
      });
      keyIssueDropdown.value = "All Key Issues"; // Automatically select "All Key Issues"
    }

    // Load filtered data based on the selected key issue
    function loadFilteredData() {
      const selectedSheet = document.getElementById('sheetSelector').value;
      const selectedKeyIssue = document.getElementById('keyIssueSelector').value || "All Key
Issues";
      if (selectedSheet && selectedKeyIssue) {
        google.script.run.withSuccessHandler(displayTable).getFilteredData(selectedSheet,
selectedKeyIssue);
      } else {
        document.getElementById('table-container').innerHTML = '';
      }
    }

    // Display the data as an HTML table
function displayTable(data) {
  const container = document.getElementById('table-container');
  if (data.length === 0) {
    container.innerHTML = '<p>No data available for the selected criteria.</p>';
    return;
  }

  let tableHTML = "<table>";

  // Add headers (excluding "Link" column)
  tableHTML += "<tr>";
  data[0].forEach(header => {
    if (header !== "Link") { // Exclude the "Link" column from headers
      tableHTML += `<th>${header}</th>`;
    }
  });
  tableHTML += "</tr>";

  // Identify the "Link" column index
  const linkColumnIndex = data[0].indexOf("Link");

  // Add rows (excluding "Link" column)
  for (let i = 1; i < data.length; i++) {
```

```
    const link = linkColumnIndex !== -1 ? data[i][linkColumnIndex] : null;

    if (link) {
      // Make the row clickable by adding 'clickable-row' class
      tableHTML += `<tr class="clickable-row" onclick="window.open('${link}', '_blank')">`;
    } else {
      tableHTML += `<tr>`;
    }

    // Add table cells (excluding "Link" column)
    data[i].forEach((cell, index) => {
      if (index !== linkColumnIndex) { // Exclude the "Link" column from rows
        tableHTML += `<td>${cell}</td>`;
      }
    });

    tableHTML += "</tr>";
  }

  tableHTML += "</table>";
  container.innerHTML = tableHTML;
}
    </script>
  </body>
</html>
```