--GET JSON FROM GITHUB’ PTIVATE REPOSITORY BY USING:

 - GITHUB\_USERNAME = "viralbalaji";

  - REPO\_NAME = "Product\_Sales";

  - REPO\_BRANCH = "main";

  - GITHUB\_TOKEN = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";

- DIRECTORY= extracted\_data

-Ex. Folder: <https://github.com/viralbalaji/Product_Sales/tree/main/extracted_data/JUNE_2025>

- NOW IN THIS “JUNE” FOLDER ALL JSON FILE WHICH HAVE JSON LINE “ "date": "4/6/2025", ” 6th MONTH IS IN THIS FOLODER.

--JSON FORMAT IS LIKE:

{

"buyerName": "DineshBhai",

"challanNumber": "237",

"vehicle": "(TEMPO Paras)",

"date": "4/6/2025",

"products": [

"300 ml W",

"300 ml BT",

"500 ml BT",

"600 ml BT"

],

"bags": [

"2",

"3",

"3",

"2"

],

"quantities": [

"1000",

"1500",

"1500",

"1000"

],

"rates": [

"2.95",

"2.95",

"3.05",

"4.10"

],

"amounts": [

"2,950",

"4,425",

"4,575",

"4,100"

],

"subtotalText": " SUB TOTAL :",

"subtotalValue": "16,050.0",

"totalBags": "10",

"totalQuantity": "5,000",

"total": "16,050"

}

-- WORKING:  
- USER WILL SELECT AVAILABLE MONTHS BY SELECTING OPTION(JUNE / JULY \*any available\*), FROM GITHUB’s REPOSITORY’s DIRECTORY’s FOLDERS’s NAMES, FOR EFFICIENCY ONLY FETCH FOLDER NAME EXTERNALLY,

NOT BY CHECKING DATE BY THAT FOLDER’s JSON FILES’s JSON LINE, LINE: ("date": "dd/mm/yyyy",) , .

- THEN SHOW AVILABLE BUYER’s NAMES ON THAT MONTH BY FETCHING NAME FROM THAT FOLDER’s JSON FILES’s (\*SCAN ONLY THAT SELECTED FOLDER’s JSON FILES FOR EFFICIENCY\*) .

- ALSO USER CAN FIND THAT SPECIFIC CHALLAN BY, YOU HAVE TO GIVE A FIELD TO SELECT CHALLAN NUMBER, FOR THIS FETCH THAT

FOLDER’s JSON FILES’s (\*SCAN ONLY THAT SELECTED FOLDER’s JSON FILES FOR EFFICIENCY\*) .

-FOR AMOUNT(SHOW IN SUMMARY) FETCH ONLY THAT FOLDER’s JSON FILES’s (\*SCAN ONLY THAT SELECTED FOLDER’s JSON FILES FOR EFFICIENCY\*) .

-- THEN SHOW SUMMARY IN LOGIC LIKE BELOW: (GET ONLY LOGIC NOT ANY OTHER.):-

// Display Summary

function displaySummary(data, selectedBuyer, selectedYear) {

const tableBody = document.querySelector("#buyerTable tbody");

tableBody.innerHTML = "";

if (data.length === 0) {

tableBody.innerHTML = `<tr><td colspan="4">No data found.</td></tr>`;

return;

}

data.sort((a, b) => a.billNumber - b.billNumber);

data.forEach(bill => {

tableBody.innerHTML += `

<tr>

<td>${bill.date}</td>

<td>${bill.buyerName}</td>

<td>₹${bill.billAmount.toFixed(2)}</td>

<td>${bill.billNumber}</td>

</tr>`;

});

updateSummaryForFilteredData(data, selectedBuyer, selectedYear);

}

// Update Summary for Filtered Data

function updateSummaryForFilteredData(filteredBills, selectedBuyer, selectedYear) {

const searchedBuyerSection = document.getElementById("searchedBuyerSection");

const searchedBuyerDetails = document.getElementById("searchedBuyerDetails");

const topBuyerSection = document.getElementById("topBuyer");

const allBuyersDetails = document.getElementById("allBuyersDetails");

const overallSummarySection = document.getElementById("overallSummary");

let buyerSummary = {};

let totalAmount = 0;

let totalChallans = filteredBills.length;

// Hide searched buyer section by default

searchedBuyerSection.style.display = "none";

// If a specific buyer is selected, display their details

if (selectedBuyer) {

const searchedBuyerBills = filteredBills.filter(bill => bill.buyerName === selectedBuyer);

if (searchedBuyerBills.length > 0) {

searchedBuyerSection.style.display = "block";

searchedBuyerDetails.innerHTML = searchedBuyerBills.map(bill =>

`<p>${bill.buyerName} - Bill Number: ${bill.billNumber}, Amount: ₹${bill.billAmount.toFixed(2)}</p>`

).join("");

}

}

// Calculate buyer summary

filteredBills.forEach(bill => {

totalAmount += bill.billAmount;

if (!buyerSummary[bill.buyerName]) {

buyerSummary[bill.buyerName] = { count: 0, totalAmount: 0 };

}

buyerSummary[bill.buyerName].count++;

buyerSummary[bill.buyerName].totalAmount += bill.billAmount;

});

let sortedBuyers = Object.entries(buyerSummary).sort((a, b) => b[1].totalAmount - a[1].totalAmount);

// Top Buyer

const [topBuyer, topBuyerData] = sortedBuyers[0];

topBuyerSection.innerHTML = `

<p><strong>Buyer with Most Amount:</strong> ${topBuyer}</p>

<p><strong>Total Challans by '${topBuyer}':</strong> ${topBuyerData.count}</p>

<p><strong>Average Bill Amount for '${topBuyer}':</strong> ₹${(topBuyerData.totalAmount / topBuyerData.count).toFixed(2)}</p>

<p><strong>Total Amount for '${topBuyer}':</strong> ₹${topBuyerData.totalAmount.toFixed(2)}</p>

`;

// All Buyers

allBuyersDetails.innerHTML = sortedBuyers.map(([buyer, data]) =>

`<p><strong>${buyer}</strong> - Challans: ${data.count}, Total Amount: ₹${data.totalAmount.toFixed(2)}</p>`

).join("");

// Overall Summary

overallSummarySection.innerHTML = `

<p><strong>Total Buyers:</strong> ${sortedBuyers.length}</p>

<p><strong>Total Challans by All Buyers:</strong> ${totalChallans}</p>

<p><strong>Average Amount per Challan:</strong> ₹${(totalAmount / totalChallans).toFixed(2)}</p>

<p><strong>Total Amount of All Challans:</strong> ₹${totalAmount.toFixed(2)}</p>

`;

}

* STYLES FOR IMPLEMENT(USE AS REFERENCE)
* body {
* font-family: Arial, sans-serif;
* margin: 20px;
* }
* h2, h3 {
* color: #333;
* }
* select, button {
* padding: 8px;
* margin: 10px 0;
* font-size: 16px;
* }
* table {
* width: 100%;
* border-collapse: collapse;
* margin-top: 20px;
* }
* table, th, td {
* border: 1px solid #ccc;
* }
* th, td {
* padding: 10px;
* text-align: center;
* }
* th {
* background-color: #f4f4f4;
* }
* .summary-section {
* margin-top: 20px;
* padding: 10px;
* border: 1px solid #ccc;
* border-radius: 5px;
* }
* .buyer-list {
* white-space: pre-line;
* }
* .error-message {
* color: red;
* margin-top: 10px;
* }
* GIVE ME FULL READY TO RUN PROGRAM. (FULLY FRONTEND).

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Total Sale Summary</title>

<style>

  body{font-family:Arial;padding:20px;background:#f7f7f7}

  select{padding:8px;margin:10px}

  table{width:100%;border-collapse:collapse;background:#fff;margin-top:20px}

  th,td{border:1px solid #ccc;padding:10px;text-align:left}

  h2{margin-top:40px}

  #summarySections p{margin:4px 0}

</style>

</head>

<body>

<h1>Total Sale Summary</h1>

<label>Select Month:</label>

<select id="monthSelect"></select>

<label>Select Buyer:</label>

<select id="buyerSelect"><option value="">-- All --</option></select>

<label>Select Challan:</label>

<select id="challanSelect"><option value="">-- All --</option></select>

<table id="buyerTable">

  <thead><tr><th>Date</th><th>Buyer</th><th>Amount</th><th>Challan No</th></tr></thead>

  <tbody></tbody>

</table>

<div id="summarySections">

  <h2>Searched Buyer Summary</h2>

  <div id="searchedBuyerSection"><div id="searchedBuyerDetails"></div></div>

  <h2>Top Buyer Summary</h2>

  <div id="topBuyer"></div>

  <h2>All Buyers Summary</h2>

  <div id="allBuyersDetails"></div>

  <h2>Overall Summary</h2>

  <div id="overallSummary"></div>

</div>

<script>

/\* ---------- CONFIG ---------- \*/

const GITHUB\_USERNAME = "viralbalaji";

const REPO\_NAME      = "Product\_Sales";

const REPO\_BRANCH    = "main";

const GITHUB\_TOKEN   = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";   // 🔑 put a valid token here

const DIRECTORY      = "extracted\_data";

/\* ---------- HELPER HEADERS ---------- \*/

const apiHeaders = {

  Authorization: "token " + GITHUB\_TOKEN,

  Accept: "application/vnd.github.v3+json"

};

const rawHeaders = {

  Authorization: "token " + GITHUB\_TOKEN,

  Accept: "application/vnd.github.v3.raw"          // <-- returns raw file text

};

/\* ---------- LIST MONTH FOLDERS ---------- \*/

async function fetchMonths() {

  const url = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${DIRECTORY}?ref=${REPO\_BRANCH}`;

  const res = await fetch(url, {headers: apiHeaders});

  const items = await res.json();

  return items.filter(i => i.type === "dir").map(i => i.name);      // e.g. MAY\_2025, JUNE\_2025 …

}

/\* ---------- GET QUICK METADATA FROM ONE MONTH ---------- \*/

async function fetchMonthMetadata(month) {

  const folderUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${DIRECTORY}/${month}?ref=${REPO\_BRANCH}`;

  const res  = await fetch(folderUrl, {headers: apiHeaders});

  const list = await res.json();

  const jsonFiles = list.filter(f => f.type === "file" && f.name.endsWith(".json"));

  const promises  = jsonFiles.map(async f => {

      const fileUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${f.path}?ref=${REPO\_BRANCH}`;

      const raw     = await fetch(fileUrl, {headers: rawHeaders});

      const text    = await raw.text();

      const lines   = text.split("\n");

      const buyer   = lines[1]?.match(/"buyerName":\s\*"([^"]+)"/)?.[1]    || "";

      const challan = lines[2]?.match(/"challanNumber":\s\*"([^"]+)"/)?.[1]|| "";

      const date    = lines.find(l => l.includes('"date"')) ?.match(/"date":\s\*"([^"]+)"/)?.[1]  || "";

      const totalS  = lines.find(l => l.includes('"total"')&& !l.includes("subtotal"))

                             ?.match(/"total":\s\*"([^"]+)"/)?.[1] || "0";

      const total   = parseFloat(totalS.replace(/,/g,"")) || 0;

      return {buyerName:buyer, challanNumber:challan, date,

              billNumber:challan, billAmount:total};

  });

  return Promise.all(promises);

}

/\* ---------- POPULATE A SELECT ---------- \*/

function fillSelect(select, list) {

  select.innerHTML = '<option value="">-- All --</option>';

  [...new Set(list)].sort().forEach(v=>{

      const o=document.createElement("option"); o.value=v; o.textContent=v; select.appendChild(o);

  });

}

/\* ---------- ENTRY POINT ---------- \*/

async function init() {

  const monthSel   = document.getElementById("monthSelect");

  const buyerSel   = document.getElementById("buyerSelect");

  const challanSel = document.getElementById("challanSelect");

  /\* months \*/

  (await fetchMonths()).forEach(m=>{

      const o=document.createElement("option"); o.value=m; o.textContent=m; monthSel.appendChild(o);

  });

  let current=[];

  monthSel.addEventListener("change", async ()=>{

      const m = monthSel.value;

      if(!m) return;

      current = await fetchMonthMetadata(m);

      fillSelect(buyerSel,   current.map(x=>x.buyerName));

      fillSelect(challanSel, current.map(x=>x.challanNumber));

      displaySummary(current);

  });

  function applyFilters() {

      let data=[...current];

      if(buyerSel.value)   data=data.filter(d=>d.buyerName   === buyerSel.value);

      if(challanSel.value) data=data.filter(d=>d.challanNumber=== challanSel.value);

      displaySummary(data, buyerSel.value);

  }

  buyerSel.addEventListener("change", applyFilters);

  challanSel.addEventListener("change", applyFilters);

}

/\* ---------- SUMMARY TABLE + STATS ---------- \*/

function displaySummary(rows, selectedBuyer="") {

  const tbody=document.querySelector("#buyerTable tbody");

  tbody.innerHTML="";

  if(!rows.length){

    tbody.innerHTML='<tr><td colspan="4">No data found.</td></tr>';

    updateSummaryForFilteredData([], selectedBuyer);

    return;

  }

  rows.sort((a,b)=>a.billNumber.localeCompare(b.billNumber,"en",{numeric:true}));

  rows.forEach(r=>{

      tbody.insertAdjacentHTML("beforeend",

        `<tr><td>${r.date}</td><td>${r.buyerName}</td><td>₹${r.billAmount.toFixed(2)}</td><td>${r.billNumber}</td></tr>`);

  });

  updateSummaryForFilteredData(rows, selectedBuyer);

}

function updateSummaryForFilteredData(bills, selectedBuyer){

  const sbSec=document.getElementById("searchedBuyerSection"),

        sbDet=document.getElementById("searchedBuyerDetails"),

        topB =document.getElementById("topBuyer"),

        allB =document.getElementById("allBuyersDetails"),

        over =document.getElementById("overallSummary");

  /\* searched buyer \*/

  sbSec.style.display="none";

  if(selectedBuyer){

      const list=bills.filter(b=>b.buyerName===selectedBuyer);

      if(list.length){

          sbSec.style.display="block";

          sbDet.innerHTML=list.map(b=>`<p>${b.buyerName} – Challan ${b.billNumber} – ₹${b.billAmount.toFixed(2)}</p>`).join("");

      }

  }

  /\* buyer aggregates \*/

  const agg={}, totalCh=bills.length;

  let totalAmt=0;

  bills.forEach(b=>{

      totalAmt+=b.billAmount;

      if(!agg[b.buyerName]) agg[b.buyerName]={cnt:0,sum:0};

      agg[b.buyerName].cnt++;

      agg[b.buyerName].sum+=b.billAmount;

  });

  const sorted=Object.entries(agg).sort((a,b)=>b[1].sum-a[1].sum);

  const [topName,topData]=sorted[0]||["-",{cnt:0,sum:0}];

  topB.innerHTML=`

     <p><strong>Top Buyer:</strong> ${topName}</p>

     <p><strong>Challans:</strong> ${topData.cnt}</p>

     <p><strong>Average Bill:</strong> ₹${(topData.sum/topData.cnt||0).toFixed(2)}</p>

     <p><strong>Total:</strong> ₹${topData.sum.toFixed(2)}</p>`;

  allB.innerHTML=sorted.map(([n,d])=>`<p><strong>${n}</strong> – Challans ${d.cnt}, Total ₹${d.sum.toFixed(2)}</p>`).join("");

  over.innerHTML=`

    <p><strong>Total Buyers:</strong> ${sorted.length}</p>

    <p><strong>Total Challans:</strong> ${totalCh}</p>

    <p><strong>Avg per Challan:</strong> ₹${(totalAmt/totalCh||0).toFixed(2)}</p>

    <p><strong>Total Amount:</strong> ₹${totalAmt.toFixed(2)}</p>`;

}

/\* ---------- GO! ---------- \*/

init();

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Total Sale Summary</title>

<style>

  body{font-family:Arial;padding:20px;background:#f7f7f7}

  select{padding:8px;margin:10px}

  table{width:100%;border-collapse:collapse;background:#fff;margin-top:20px}

  th,td{border:1px solid #ccc;padding:10px;text-align:left}

  h2{margin-top:40px}

  #summarySections p{margin:4px 0}

</style>

</head>

<body>

<h1>Total Sale Summary</h1>

<label>Select Month:</label>

<select id="monthSelect"></select>

<label>Select Buyer:</label>

<select id="buyerSelect"><option value="">-- All --</option></select>

<label>Select Challan:</label>

<select id="challanSelect"><option value="">-- All --</option></select>

<table id="buyerTable">

  <thead><tr><th>Date</th><th>Buyer</th><th>Amount</th><th>Challan No</th></tr></thead>

  <tbody></tbody>

</table>

<div id="summarySections">

  <h2>Searched Buyer Summary</h2>

  <div id="searchedBuyerSection"><div id="searchedBuyerDetails"></div></div>

  <h2>Top Buyer Summary</h2>

  <div id="topBuyer"></div>

  <h2>All Buyers Summary</h2>

  <div id="allBuyersDetails"></div>

  <h2>Overall Summary</h2>

  <div id="overallSummary"></div>

</div>

<script>

/\* ---------- CONFIG ---------- \*/

const GITHUB\_USERNAME = "viralbalaji";

const REPO\_NAME      = "Product\_Sales";

const REPO\_BRANCH    = "main";

const GITHUB\_TOKEN   = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";   // 🔑 put a valid token here

const DIRECTORY      = "extracted\_data";

/\* ---------- HELPER HEADERS ---------- \*/

const apiHeaders = {

  Authorization: "token " + GITHUB\_TOKEN,

  Accept: "application/vnd.github.v3+json"

};

const rawHeaders = {

  Authorization: "token " + GITHUB\_TOKEN,

  Accept: "application/vnd.github.v3.raw"          // <-- returns raw file text

};

/\* ---------- LIST MONTH FOLDERS ---------- \*/

async function fetchMonths() {

  const url = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${DIRECTORY}?ref=${REPO\_BRANCH}`;

  const res = await fetch(url, {headers: apiHeaders});

  const items = await res.json();

  return items.filter(i => i.type === "dir").map(i => i.name);      // e.g. MAY\_2025, JUNE\_2025 …

}

/\* ---------- GET QUICK METADATA FROM ONE MONTH ---------- \*/

async function fetchMonthMetadata(month) {

  const folderUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${DIRECTORY}/${month}?ref=${REPO\_BRANCH}`;

  const res  = await fetch(folderUrl, {headers: apiHeaders});

  const list = await res.json();

  const jsonFiles = list.filter(f => f.type === "file" && f.name.endsWith(".json"));

  const promises  = jsonFiles.map(async f => {

      const fileUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${f.path}?ref=${REPO\_BRANCH}`;

      const raw     = await fetch(fileUrl, {headers: rawHeaders});

      const text    = await raw.text();

      const lines   = text.split("\n");

      const buyer   = lines[1]?.match(/"buyerName":\s\*"([^"]+)"/)?.[1]    || "";

      const challan = lines[2]?.match(/"challanNumber":\s\*"([^"]+)"/)?.[1]|| "";

      const date    = lines.find(l => l.includes('"date"')) ?.match(/"date":\s\*"([^"]+)"/)?.[1]  || "";

      const totalS  = lines.find(l => l.includes('"total"')&& !l.includes("subtotal"))

                             ?.match(/"total":\s\*"([^"]+)"/)?.[1] || "0";

      const total   = parseFloat(totalS.replace(/,/g,"")) || 0;

      return {buyerName:buyer, challanNumber:challan, date,

              billNumber:challan, billAmount:total};

  });

  return Promise.all(promises);

}

/\* ---------- POPULATE A SELECT ---------- \*/

function fillSelect(select, list) {

  select.innerHTML = '<option value="">-- All --</option>';

  [...new Set(list)].sort().forEach(v=>{

      const o=document.createElement("option"); o.value=v; o.textContent=v; select.appendChild(o);

  });

}

/\* ---------- ENTRY POINT ---------- \*/

async function init() {

  const monthSel   = document.getElementById("monthSelect");

  const buyerSel   = document.getElementById("buyerSelect");

  const challanSel = document.getElementById("challanSelect");

  /\* months \*/

  (await fetchMonths()).forEach(m=>{

      const o=document.createElement("option"); o.value=m; o.textContent=m; monthSel.appendChild(o);

  });

  let current=[];

  monthSel.addEventListener("change", async ()=>{

      const m = monthSel.value;

      if(!m) return;

      current = await fetchMonthMetadata(m);

      fillSelect(buyerSel,   current.map(x=>x.buyerName));

      fillSelect(challanSel, current.map(x=>x.challanNumber));

      displaySummary(current);

  });

  function applyFilters() {

      let data=[...current];

      if(buyerSel.value)   data=data.filter(d=>d.buyerName   === buyerSel.value);

      if(challanSel.value) data=data.filter(d=>d.challanNumber=== challanSel.value);

      displaySummary(data, buyerSel.value);

  }

  buyerSel.addEventListener("change", applyFilters);

  challanSel.addEventListener("change", applyFilters);

}

/\* ---------- SUMMARY TABLE + STATS ---------- \*/

function displaySummary(rows, selectedBuyer="") {

  const tbody=document.querySelector("#buyerTable tbody");

  tbody.innerHTML="";

  if(!rows.length){

    tbody.innerHTML='<tr><td colspan="4">No data found.</td></tr>';

    updateSummaryForFilteredData([], selectedBuyer);

    return;

  }

  rows.sort((a,b)=>a.billNumber.localeCompare(b.billNumber,"en",{numeric:true}));

  rows.forEach(r=>{

      tbody.insertAdjacentHTML("beforeend",

        `<tr><td>${r.date}</td><td>${r.buyerName}</td><td>₹${r.billAmount.toFixed(2)}</td><td>${r.billNumber}</td></tr>`);

  });

  updateSummaryForFilteredData(rows, selectedBuyer);

}

function updateSummaryForFilteredData(bills, selectedBuyer){

  const sbSec=document.getElementById("searchedBuyerSection"),

        sbDet=document.getElementById("searchedBuyerDetails"),

        topB =document.getElementById("topBuyer"),

        allB =document.getElementById("allBuyersDetails"),

        over =document.getElementById("overallSummary");

  /\* searched buyer \*/

  sbSec.style.display="none";

  if(selectedBuyer){

      const list=bills.filter(b=>b.buyerName===selectedBuyer);

      if(list.length){

          sbSec.style.display="block";

          sbDet.innerHTML=list.map(b=>`<p>${b.buyerName} – Challan ${b.billNumber} – ₹${b.billAmount.toFixed(2)}</p>`).join("");

      }

  }

  /\* buyer aggregates \*/

  const agg={}, totalCh=bills.length;

  let totalAmt=0;

  bills.forEach(b=>{

      totalAmt+=b.billAmount;

      if(!agg[b.buyerName]) agg[b.buyerName]={cnt:0,sum:0};

      agg[b.buyerName].cnt++;

      agg[b.buyerName].sum+=b.billAmount;

  });

  const sorted=Object.entries(agg).sort((a,b)=>b[1].sum-a[1].sum);

  const [topName,topData]=sorted[0]||["-",{cnt:0,sum:0}];

  topB.innerHTML=`

     <p><strong>Top Buyer:</strong> ${topName}</p>

     <p><strong>Challans:</strong> ${topData.cnt}</p>

     <p><strong>Average Bill:</strong> ₹${(topData.sum/topData.cnt||0).toFixed(2)}</p>

     <p><strong>Total:</strong> ₹${topData.sum.toFixed(2)}</p>`;

  allB.innerHTML=sorted.map(([n,d])=>`<p><strong>${n}</strong> – Challans ${d.cnt}, Total ₹${d.sum.toFixed(2)}</p>`).join("");

  over.innerHTML=`

    <p><strong>Total Buyers:</strong> ${sorted.length}</p>

    <p><strong>Total Challans:</strong> ${totalCh}</p>

    <p><strong>Avg per Challan:</strong> ₹${(totalAmt/totalCh||0).toFixed(2)}</p>

    <p><strong>Total Amount:</strong> ₹${totalAmt.toFixed(2)}</p>`;

}

/\* ---------- GO! ---------- \*/

init();

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Product Sales Analysis</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 20px;

        }

        h2, h3 {

            color: #333;

        }

        select, button {

            padding: 8px;

            margin: 10px 0;

            font-size: 16px;

        }

        table {

            width: 100%;

            border-collapse: collapse;

            margin-top: 20px;

        }

        table, th, td {

            border: 1px solid #ccc;

        }

        th, td {

            padding: 10px;

            text-align: center;

        }

        th {

            background-color: #f4f4f4;

        }

        .summary-section {

            margin-top: 20px;

            padding: 10px;

            border: 1px solid #ccc;

            border-radius: 5px;

        }

        .buyer-list {

            white-space: pre-line;

        }

        .error-message {

            color: red;

            margin-top: 10px;

        }

        .loading {

            display: inline-block;

            width: 20px;

            height: 20px;

            border: 3px solid rgba(0,0,0,.3);

            border-radius: 50%;

            border-top-color: #000;

            animation: spin 1s ease-in-out infinite;

            margin-left: 10px;

        }

        @keyframes spin {

            to { transform: rotate(360deg); }

        }

        .section {

            margin-bottom: 20px;

            padding: 15px;

            background-color: #f9f9f9;

            border-radius: 5px;

        }

        .filter-container {

            display: flex;

            gap: 20px;

            flex-wrap: wrap;

            margin-bottom: 15px;

        }

        .filter-group {

            display: flex;

            flex-direction: column;

            min-width: 200px;

        }

        .loading-container {

            padding: 10px;

            text-align: center;

        }

    </style>

</head>

<body>

    <h1>Product Sales Analysis</h1>

    <div class="section">

        <h2>Filters</h2>

        <div class="filter-container">

            <div class="filter-group">

                <label for="monthSelect">Select Month</label>

                <select id="monthSelect" onchange="monthChanged()">

                    <option value="">-- Select Month --</option>

                </select>

            </div>

            <div class="filter-group">

                <label for="buyerSelect">Select Buyer</label>

                <select id="buyerSelect" onchange="buyerChanged()">

                    <option value="">-- All Buyers --</option>

                </select>

            </div>

            <div class="filter-group">

                <label for="challanSelect">Select Challan</label>

                <select id="challanSelect" onchange="challanChanged()">

                    <option value="">-- All Challans --</option>

                </select>

            </div>

        </div>

        <div class="loading-container">

            <span id="loadingIndicator" class="loading" style="display: none;"></span>

            <span id="loadingText" style="display: none;">Loading data...</span>

        </div>

    </div>

    <div id="errorMessage" class="error-message"></div>

    <div id="summaryTable" style="display: none;">

        <h2>Transaction Summary</h2>

        <table id="buyerTable">

            <thead>

                <tr>

                    <th>Date</th>

                    <th>Buyer Name</th>

                    <th>Amount</th>

                    <th>Challan Number</th>

                </tr>

            </thead>

            <tbody></tbody>

        </table>

    </div>

    <div id="summaryDetails" style="display: none;">

        <div class="summary-section" id="searchedBuyerSection">

            <h3>Searched Buyer Details</h3>

            <div id="searchedBuyerDetails"></div>

        </div>

        <div class="summary-section">

            <h3>Top Buyer</h3>

            <div id="topBuyer"></div>

        </div>

        <div class="summary-section">

            <h3>All Buyers Summary</h3>

            <div id="allBuyersDetails"></div>

        </div>

        <div class="summary-section">

            <h3>Overall Summary</h3>

            <div id="overallSummary"></div>

        </div>

    </div>

    <script>

        // Configuration

        const GITHUB\_USERNAME = "viralbalaji";

        const REPO\_NAME = "Product\_Sales";

        const REPO\_BRANCH = "main";

        const GITHUB\_TOKEN = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";

        const DIRECTORY = "extracted\_data";

        let allMonths = [];

        let allData = [];

        let currentMonth = '';

        let isLoading = false;

        // Initialize the application

        window.onload = async function() {

            await loadAvailableMonths();

        };

        // Show loading indicator

        function showLoading(show) {

            document.getElementById('loadingIndicator').style.display = show ? 'inline-block' : 'none';

            document.getElementById('loadingText').style.display = show ? 'inline' : 'none';

            isLoading = show;

        }

        // Load available months from GitHub repository

        async function loadAvailableMonths() {

            showLoading(true);

            document.getElementById('errorMessage').textContent = '';

            try {

                const url = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${DIRECTORY}`;

                const response = await fetch(url, {

                    headers: {

                        'Authorization': `token ${GITHUB\_TOKEN}`,

                        'Accept': 'application/vnd.github.v3+json'

                    }

                });

                if (!response.ok) {

                    throw new Error(`GitHub API error: ${response.status}`);

                }

                const data = await response.json();

                allMonths = data.filter(item => item.type === 'dir')

                                .map(item => item.name)

                                .sort((a, b) => {

                                    // Sort by month and year (assuming format like JUNE\_2025)

                                    const [monthA, yearA] = a.split('\_');

                                    const [monthB, yearB] = b.split('\_');

                                    const monthOrder = ["JANUARY", "FEBRUARY", "MARCH", "APRIL", "MAY", "JUNE",

                                                         "JULY", "AUGUST", "SEPTEMBER", "OCTOBER", "NOVEMBER", "DECEMBER"];

                                    const monthIndexA = monthOrder.indexOf(monthA);

                                    const monthIndexB = monthOrder.indexOf(monthB);

                                    if (yearA !== yearB) return yearB - yearA;

                                    return monthIndexB - monthIndexA;

                                });

                const monthSelect = document.getElementById('monthSelect');

                monthSelect.innerHTML = '<option value="">-- Select Month --</option>';

                allMonths.forEach(month => {

                    monthSelect.innerHTML += `<option value="${month}">${month.replace('\_', ' ')}</option>`;

                });

                showLoading(false);

            } catch (error) {

                showLoading(false);

                document.getElementById('errorMessage').textContent = `Error loading months: ${error.message}`;

                console.error('Error loading months:', error);

            }

        }

        // Handle month selection change

        async function monthChanged() {

            const selectedMonth = document.getElementById('monthSelect').value;

            if (!selectedMonth) {

                document.getElementById('buyerSelect').innerHTML = '<option value="">-- All Buyers --</option>';

                document.getElementById('challanSelect').innerHTML = '<option value="">-- All Challans --</option>';

                document.getElementById('summaryTable').style.display = 'none';

                document.getElementById('summaryDetails').style.display = 'none';

                return;

            }

            currentMonth = selectedMonth;

            await loadBuyersAndChallans();

            updateSummary();

        }

        // Handle buyer selection change

        function buyerChanged() {

            const selectedBuyer = document.getElementById('buyerSelect').value;

            if (selectedBuyer) {

                // Clear challan selection if buyer is selected

                document.getElementById('challanSelect').value = '';

            }

            updateSummary();

        }

        // Handle challan selection change

        function challanChanged() {

            const selectedChallan = document.getElementById('challanSelect').value;

            if (selectedChallan) {

                // Clear buyer selection if challan is selected

                document.getElementById('buyerSelect').value = '';

            }

            updateSummary();

        }

        // Load buyers and challans for selected month

        async function loadBuyersAndChallans() {

            if (!currentMonth || isLoading) return;

            showLoading(true);

            document.getElementById('errorMessage').textContent = '';

            try {

                // Get list of JSON files in the selected month folder

                const url = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${DIRECTORY}/${currentMonth}`;

                const response = await fetch(url, {

                    headers: {

                        'Authorization': `token ${GITHUB\_TOKEN}`,

                        'Accept': 'application/vnd.github.v3+json'

                    }

                });

                if (!response.ok) {

                    throw new Error(`GitHub API error: ${response.status}`);

                }

                const files = await response.json();

                const jsonFiles = files.filter(file => file.name.endsWith('.json'));

                // Process each JSON file

                allData = [];

                const buyers = new Set();

                const challans = new Set();

                for (const file of jsonFiles) {

                    try {

                        const fileUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${DIRECTORY}/${currentMonth}/${file.name}`;

                        const fileResponse = await fetch(fileUrl, {

                            headers: {

                                'Authorization': `token ${GITHUB\_TOKEN}`,

                                'Accept': 'application/vnd.github.v3.raw'

                            }

                        });

                        if (!fileResponse.ok) {

                            console.error(`Error loading file ${file.name}: ${fileResponse.status}`);

                            continue;

                        }

                        const jsonData = await fileResponse.json();

                        // Store the data

                        allData.push({

                            date: jsonData.date,

                            buyerName: jsonData.buyerName,

                            billAmount: parseFloat(jsonData.total.replace(/,/g, '')),

                            billNumber: jsonData.challanNumber,

                            vehicle: jsonData.vehicle,

                            products: jsonData.products,

                            quantities: jsonData.quantities,

                            rates: jsonData.rates,

                            amounts: jsonData.amounts

                        });

                        // Collect unique buyers and challans

                        buyers.add(jsonData.buyerName);

                        challans.add(jsonData.challanNumber);

                    } catch (error) {

                        console.error(`Error processing file ${file.name}:`, error);

                    }

                }

                // Update buyer dropdown

                const buyerSelect = document.getElementById('buyerSelect');

                buyerSelect.innerHTML = '<option value="">-- All Buyers --</option>';

                Array.from(buyers).sort().forEach(buyer => {

                    buyerSelect.innerHTML += `<option value="${buyer}">${buyer}</option>`;

                });

                // Update challan dropdown

                const challanSelect = document.getElementById('challanSelect');

                challanSelect.innerHTML = '<option value="">-- All Challans --</option>';

                Array.from(challans).sort((a, b) => a - b).forEach(challan => {

                    challanSelect.innerHTML += `<option value="${challan}">${challan}</option>`;

                });

                showLoading(false);

            } catch (error) {

                showLoading(false);

                document.getElementById('errorMessage').textContent = `Error loading data: ${error.message}`;

                console.error('Error loading data:', error);

            }

        }

        // Update summary based on current filters

        function updateSummary() {

            if (!currentMonth || allData.length === 0) {

                document.getElementById('summaryTable').style.display = 'none';

                document.getElementById('summaryDetails').style.display = 'none';

                return;

            }

            const selectedBuyer = document.getElementById('buyerSelect').value;

            const selectedChallan = document.getElementById('challanSelect').value;

            // Filter data based on selections

            let filteredData = [...allData];

            if (selectedBuyer) {

                filteredData = filteredData.filter(item => item.buyerName === selectedBuyer);

            }

            if (selectedChallan) {

                filteredData = filteredData.filter(item => item.billNumber === selectedChallan);

            }

            // Display the summary

            displaySummary(filteredData, selectedBuyer || selectedChallan);

            // Show the summary sections

            document.getElementById('summaryTable').style.display = 'block';

            document.getElementById('summaryDetails').style.display = 'block';

        }

        // Display Summary

        function displaySummary(data, selectedFilter) {

            const tableBody = document.querySelector("#buyerTable tbody");

            tableBody.innerHTML = "";

            if (data.length === 0) {

                tableBody.innerHTML = `<tr><td colspan="4">No data found.</td></tr>`;

                return;

            }

            data.sort((a, b) => a.billNumber - b.billNumber);

            data.forEach(bill => {

                tableBody.innerHTML += `

                    <tr>

                        <td>${bill.date}</td>

                        <td>${bill.buyerName}</td>

                        <td>₹${bill.billAmount.toFixed(2)}</td>

                        <td>${bill.billNumber}</td>

                    </tr>`;

            });

            updateSummaryForFilteredData(data, selectedFilter);

        }

        // Update Summary for Filtered Data

        function updateSummaryForFilteredData(filteredBills, selectedFilter) {

            const searchedBuyerSection = document.getElementById("searchedBuyerSection");

            const searchedBuyerDetails = document.getElementById("searchedBuyerDetails");

            const topBuyerSection = document.getElementById("topBuyer");

            const allBuyersDetails = document.getElementById("allBuyersDetails");

            const overallSummarySection = document.getElementById("overallSummary");

            let buyerSummary = {};

            let totalAmount = 0;

            let totalChallans = filteredBills.length;

            // Hide searched buyer section by default

            searchedBuyerSection.style.display = "none";

            // If a specific filter is selected, display details

            if (selectedFilter) {

                const searchedBills = filteredBills.filter(bill =>

                    bill.buyerName === selectedFilter || bill.billNumber === selectedFilter

                );

                if (searchedBills.length > 0) {

                    searchedBuyerSection.style.display = "block";

                    searchedBuyerDetails.innerHTML = searchedBills.map(bill =>

                        `<p>${bill.date} - ${bill.buyerName} - Challan: ${bill.billNumber}, Amount: ₹${bill.billAmount.toFixed(2)}</p>`

                    ).join("");

                }

            }

            // Calculate buyer summary

            filteredBills.forEach(bill => {

                totalAmount += bill.billAmount;

                if (!buyerSummary[bill.buyerName]) {

                    buyerSummary[bill.buyerName] = { count: 0, totalAmount: 0 };

                }

                buyerSummary[bill.buyerName].count++;

                buyerSummary[bill.buyerName].totalAmount += bill.billAmount;

            });

            let sortedBuyers = Object.entries(buyerSummary).sort((a, b) => b[1].totalAmount - a[1].totalAmount);

            // Top Buyer

            if (sortedBuyers.length > 0) {

                const [topBuyer, topBuyerData] = sortedBuyers[0];

                topBuyerSection.innerHTML = `

                    <p><strong>Buyer with Most Amount:</strong> ${topBuyer}</p>

                    <p><strong>Total Challans by '${topBuyer}':</strong> ${topBuyerData.count}</p>

                    <p><strong>Average Bill Amount for '${topBuyer}':</strong> ₹${(topBuyerData.totalAmount / topBuyerData.count).toFixed(2)}</p>

                    <p><strong>Total Amount for '${topBuyer}':</strong> ₹${topBuyerData.totalAmount.toFixed(2)}</p>

                `;

            } else {

                topBuyerSection.innerHTML = `<p>No data available</p>`;

            }

            // All Buyers

            allBuyersDetails.innerHTML = sortedBuyers.map(([buyer, data]) =>

                `<p><strong>${buyer}</strong> - Challans: ${data.count}, Total Amount: ₹${data.totalAmount.toFixed(2)}</p>`

            ).join("");

            // Overall Summary

            overallSummarySection.innerHTML = `

                <p><strong>Total Buyers:</strong> ${sortedBuyers.length}</p>

                <p><strong>Total Challans by All Buyers:</strong> ${totalChallans}</p>

                <p><strong>Average Amount per Challan:</strong> ₹${totalChallans > 0 ? (totalAmount / totalChallans).toFixed(2) : '0.00'}</p>

                <p><strong>Total Amount of All Challans:</strong> ₹${totalAmount.toFixed(2)}</p>

            `;

        }

    </script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Product Sales Analysis</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 20px;

        }

        h2, h3 {

            color: #333;

        }

        select, button {

            padding: 8px;

            margin: 10px 0;

            font-size: 16px;

        }

        table {

            width: 100%;

            border-collapse: collapse;

            margin-top: 20px;

        }

        table, th, td {

            border: 1px solid #ccc;

        }

        th, td {

            padding: 10px;

            text-align: center;

        }

        th {

            background-color: #f4f4f4;

        }

        .summary-section {

            margin-top: 20px;

            padding: 10px;

            border: 1px solid #ccc;

            border-radius: 5px;

        }

        .buyer-list {

            white-space: pre-line;

        }

        .error-message {

            color: red;

            margin-top: 10px;

        }

        .loading {

            display: inline-block;

            width: 20px;

            height: 20px;

            border: 3px solid rgba(0,0,0,.3);

            border-radius: 50%;

            border-top-color: #000;

            animation: spin 1s ease-in-out infinite;

            margin-left: 10px;

        }

        @keyframes spin {

            to { transform: rotate(360deg); }

        }

        .section {

            margin-bottom: 20px;

            padding: 15px;

            background-color: #f9f9f9;

            border-radius: 5px;

        }

        .filter-container {

            display: flex;

            gap: 20px;

            flex-wrap: wrap;

            margin-bottom: 15px;

        }

        .filter-group {

            display: flex;

            flex-direction: column;

            min-width: 200px;

        }

        .loading-container {

            padding: 10px;

            text-align: center;

        }

    </style>

</head>

<body>

    <h1>Product Sales Analysis</h1>

    <div class="section">

        <h2>Filters</h2>

        <div class="filter-container">

            <div class="filter-group">

                <label for="monthSelect">Select Month</label>

                <select id="monthSelect" onchange="monthChanged()">

                    <option value="">-- Select Month --</option>

                </select>

            </div>

            <div class="filter-group">

                <label for="buyerSelect">Select Buyer</label>

                <select id="buyerSelect" onchange="buyerChanged()">

                    <option value="">-- All Buyers --</option>

                </select>

            </div>

            <div class="filter-group">

                <label for="challanSelect">Select Challan</label>

                <select id="challanSelect" onchange="challanChanged()">

                    <option value="">-- All Challans --</option>

                </select>

            </div>

        </div>

        <div class="loading-container">

            <span id="loadingIndicator" class="loading" style="display: none;"></span>

            <span id="loadingText" style="display: none;">Loading data...</span>

        </div>

    </div>

    <div id="errorMessage" class="error-message"></div>

    <div id="summaryTable" style="display: none;">

        <h2>Transaction Summary</h2>

        <table id="buyerTable">

            <thead>

                <tr>

                    <th>Date</th>

                    <th>Buyer Name</th>

                    <th>Amount</th>

                    <th>Challan Number</th>

                </tr>

            </thead>

            <tbody></tbody>

        </table>

    </div>

    <div id="summaryDetails" style="display: none;">

        <div class="summary-section" id="searchedBuyerSection">

            <h3>Searched Buyer Details</h3>

            <div id="searchedBuyerDetails"></div>

        </div>

        <div class="summary-section">

            <h3>Top Buyer</h3>

            <div id="topBuyer"></div>

        </div>

        <div class="summary-section">

            <h3>All Buyers Summary</h3>

            <div id="allBuyersDetails"></div>

        </div>

        <div class="summary-section">

            <h3>Overall Summary</h3>

            <div id="overallSummary"></div>

        </div>

    </div>

    <script>

        // Configuration

        const GITHUB\_USERNAME = "viralbalaji";

        const REPO\_NAME = "Product\_Sales";

        const REPO\_BRANCH = "main";

        const GITHUB\_TOKEN = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";

        const DIRECTORY = "extracted\_data";

        let allMonths = [];

        let allData = [];

        let currentMonth = '';

        let isLoading = false;

        // Initialize the application

        window.onload = async function() {

            await loadAvailableMonths();

        };

        // Show loading indicator

        function showLoading(show) {

            document.getElementById('loadingIndicator').style.display = show ? 'inline-block' : 'none';

            document.getElementById('loadingText').style.display = show ? 'inline' : 'none';

            isLoading = show;

        }

        // Load available months from GitHub repository

        async function loadAvailableMonths() {

            showLoading(true);

            document.getElementById('errorMessage').textContent = '';

            try {

                const url = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${DIRECTORY}`;

                const response = await fetch(url, {

                    headers: {

                        'Authorization': `token ${GITHUB\_TOKEN}`,

                        'Accept': 'application/vnd.github.v3+json'

                    }

                });

                if (!response.ok) {

                    throw new Error(`GitHub API error: ${response.status}`);

                }

                const data = await response.json();

                allMonths = data.filter(item => item.type === 'dir')

                                .map(item => item.name)

                                .sort((a, b) => {

                                    // Sort by month and year (assuming format like JUNE\_2025)

                                    const [monthA, yearA] = a.split('\_');

                                    const [monthB, yearB] = b.split('\_');

                                    const monthOrder = ["JANUARY", "FEBRUARY", "MARCH", "APRIL", "MAY", "JUNE",

                                                         "JULY", "AUGUST", "SEPTEMBER", "OCTOBER", "NOVEMBER", "DECEMBER"];

                                    const monthIndexA = monthOrder.indexOf(monthA);

                                    const monthIndexB = monthOrder.indexOf(monthB);

                                    if (yearA !== yearB) return yearB - yearA;

                                    return monthIndexB - monthIndexA;

                                });

                const monthSelect = document.getElementById('monthSelect');

                monthSelect.innerHTML = '<option value="">-- Select Month --</option>';

                allMonths.forEach(month => {

                    monthSelect.innerHTML += `<option value="${month}">${month.replace('\_', ' ')}</option>`;

                });

                showLoading(false);

            } catch (error) {

                showLoading(false);

                document.getElementById('errorMessage').textContent = `Error loading months: ${error.message}`;

                console.error('Error loading months:', error);

            }

        }

        // Handle month selection change

        async function monthChanged() {

            const selectedMonth = document.getElementById('monthSelect').value;

            if (!selectedMonth) {

                document.getElementById('buyerSelect').innerHTML = '<option value="">-- All Buyers --</option>';

                document.getElementById('challanSelect').innerHTML = '<option value="">-- All Challans --</option>';

                document.getElementById('summaryTable').style.display = 'none';

                document.getElementById('summaryDetails').style.display = 'none';

                return;

            }

            currentMonth = selectedMonth;

            await loadBuyersAndChallans();

            updateSummary();

        }

        // Handle buyer selection change

        function buyerChanged() {

            const selectedBuyer = document.getElementById('buyerSelect').value;

            if (selectedBuyer) {

                // Clear challan selection if buyer is selected

                document.getElementById('challanSelect').value = '';

            }

            updateSummary();

        }

        // Handle challan selection change

        function challanChanged() {

            const selectedChallan = document.getElementById('challanSelect').value;

            if (selectedChallan) {

                // Clear buyer selection if challan is selected

                document.getElementById('buyerSelect').value = '';

            }

            updateSummary();

        }

        // Load buyers and challans for selected month

        async function loadBuyersAndChallans() {

            if (!currentMonth || isLoading) return;

            showLoading(true);

            document.getElementById('errorMessage').textContent = '';

            try {

                // Get list of JSON files in the selected month folder

                const url = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${DIRECTORY}/${currentMonth}`;

                const response = await fetch(url, {

                    headers: {

                        'Authorization': `token ${GITHUB\_TOKEN}`,

                        'Accept': 'application/vnd.github.v3+json'

                    }

                });

                if (!response.ok) {

                    throw new Error(`GitHub API error: ${response.status}`);

                }

                const files = await response.json();

                const jsonFiles = files.filter(file => file.name.endsWith('.json'));

                // Process each JSON file

                allData = [];

                const buyers = new Set();

                const challans = new Set();

                for (const file of jsonFiles) {

                    try {

                        const fileUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${DIRECTORY}/${currentMonth}/${file.name}`;

                        const fileResponse = await fetch(fileUrl, {

                            headers: {

                                'Authorization': `token ${GITHUB\_TOKEN}`,

                                'Accept': 'application/vnd.github.v3.raw'

                            }

                        });

                        if (!fileResponse.ok) {

                            console.error(`Error loading file ${file.name}: ${fileResponse.status}`);

                            continue;

                        }

                        const jsonData = await fileResponse.json();

                        // Store the data

                        allData.push({

                            date: jsonData.date,

                            buyerName: jsonData.buyerName,

                            billAmount: parseFloat(jsonData.total.replace(/,/g, '')),

                            billNumber: jsonData.challanNumber,

                            vehicle: jsonData.vehicle,

                            products: jsonData.products,

                            quantities: jsonData.quantities,

                            rates: jsonData.rates,

                            amounts: jsonData.amounts

                        });

                        // Collect unique buyers and challans

                        buyers.add(jsonData.buyerName);

                        challans.add(jsonData.challanNumber);

                    } catch (error) {

                        console.error(`Error processing file ${file.name}:`, error);

                    }

                }

                // Update buyer dropdown

                const buyerSelect = document.getElementById('buyerSelect');

                buyerSelect.innerHTML = '<option value="">-- All Buyers --</option>';

                Array.from(buyers).sort().forEach(buyer => {

                    buyerSelect.innerHTML += `<option value="${buyer}">${buyer}</option>`;

                });

                // Update challan dropdown

                const challanSelect = document.getElementById('challanSelect');

                challanSelect.innerHTML = '<option value="">-- All Challans --</option>';

                Array.from(challans).sort((a, b) => a - b).forEach(challan => {

                    challanSelect.innerHTML += `<option value="${challan}">${challan}</option>`;

                });

                showLoading(false);

            } catch (error) {

                showLoading(false);

                document.getElementById('errorMessage').textContent = `Error loading data: ${error.message}`;

                console.error('Error loading data:', error);

            }

        }

        // Update summary based on current filters

        function updateSummary() {

            if (!currentMonth || allData.length === 0) {

                document.getElementById('summaryTable').style.display = 'none';

                document.getElementById('summaryDetails').style.display = 'none';

                return;

            }

            const selectedBuyer = document.getElementById('buyerSelect').value;

            const selectedChallan = document.getElementById('challanSelect').value;

            // Filter data based on selections

            let filteredData = [...allData];

            if (selectedBuyer) {

                filteredData = filteredData.filter(item => item.buyerName === selectedBuyer);

            }

            if (selectedChallan) {

                filteredData = filteredData.filter(item => item.billNumber === selectedChallan);

            }

            // Display the summary

            displaySummary(filteredData, selectedBuyer || selectedChallan);

            // Show the summary sections

            document.getElementById('summaryTable').style.display = 'block';

            document.getElementById('summaryDetails').style.display = 'block';

        }

        // Display Summary

        function displaySummary(data, selectedFilter) {

            const tableBody = document.querySelector("#buyerTable tbody");

            tableBody.innerHTML = "";

            if (data.length === 0) {

                tableBody.innerHTML = `<tr><td colspan="4">No data found.</td></tr>`;

                return;

            }

            data.sort((a, b) => a.billNumber - b.billNumber);

            data.forEach(bill => {

                tableBody.innerHTML += `

                    <tr>

                        <td>${bill.date}</td>

                        <td>${bill.buyerName}</td>

                        <td>₹${bill.billAmount.toFixed(2)}</td>

                        <td>${bill.billNumber}</td>

                    </tr>`;

            });

            updateSummaryForFilteredData(data, selectedFilter);

        }

        // Update Summary for Filtered Data

      function updateSummaryForFilteredData(filteredBills, selectedFilter) {

    const searchedBuyerSection = document.getElementById("searchedBuyerSection");

    const searchedBuyerDetails = document.getElementById("searchedBuyerDetails");

    const topBuyerSection = document.getElementById("topBuyer");

    const allBuyersDetails = document.getElementById("allBuyersDetails");

    const overallSummarySection = document.getElementById("overallSummary");

    const selectedBuyer = document.getElementById("buyerSelect").value;

    const selectedChallan = document.getElementById("challanSelect").value;

    // Clear all sections

    searchedBuyerDetails.innerHTML = "";

    topBuyerSection.innerHTML = "";

    allBuyersDetails.innerHTML = "";

    overallSummarySection.innerHTML = "";

    // Hide all by default

    searchedBuyerSection.style.display = "none";

    topBuyerSection.parentElement.style.display = "none";

    allBuyersDetails.parentElement.style.display = "none";

    overallSummarySection.parentElement.style.display = "none";

    // Show searched buyer/challan details if either selected

    if (selectedBuyer || selectedChallan) {

        searchedBuyerSection.style.display = "block";

        searchedBuyerDetails.innerHTML = filteredBills.map(bill =>

            `<p>${bill.date} - ${bill.buyerName} - Challan: ${bill.billNumber}, Amount: ₹${bill.billAmount.toFixed(2)}</p>`

        ).join("");

    }

    // Only show Overall Summary if buyer is selected (but not challan)

    if (selectedBuyer && !selectedChallan) {

        let totalAmount = 0;

        let totalChallans = filteredBills.length;

        filteredBills.forEach(bill => {

            totalAmount += bill.billAmount;

        });

        overallSummarySection.innerHTML = `

            <p><strong>Total Challans by '${selectedBuyer}':</strong> ${totalChallans}</p>

            <p><strong>Average Bill Amount:</strong> ₹${(totalAmount / totalChallans).toFixed(2)}</p>

            <p><strong>Total Amount:</strong> ₹${totalAmount.toFixed(2)}</p>

        `;

        overallSummarySection.parentElement.style.display = "block";

        return;

    }

    // If Challan is selected, show only searched details and exit

    if (selectedChallan) return;

    // If no filter: Show all summaries

    let buyerSummary = {};

    let totalAmount = 0;

    let totalChallans = filteredBills.length;

    filteredBills.forEach(bill => {

        totalAmount += bill.billAmount;

        if (!buyerSummary[bill.buyerName]) {

            buyerSummary[bill.buyerName] = { count: 0, totalAmount: 0 };

        }

        buyerSummary[bill.buyerName].count++;

        buyerSummary[bill.buyerName].totalAmount += bill.billAmount;

    });

    const sortedBuyers = Object.entries(buyerSummary).sort((a, b) => b[1].totalAmount - a[1].totalAmount);

    // Top Buyer

    if (sortedBuyers.length > 0) {

        const [topBuyer, topBuyerData] = sortedBuyers[0];

        topBuyerSection.innerHTML = `

            <p><strong>Buyer with Most Amount:</strong> ${topBuyer}</p>

            <p><strong>Total Challans:</strong> ${topBuyerData.count}</p>

            <p><strong>Average Bill Amount:</strong> ₹${(topBuyerData.totalAmount / topBuyerData.count).toFixed(2)}</p>

            <p><strong>Total Amount:</strong> ₹${topBuyerData.totalAmount.toFixed(2)}</p>

        `;

        topBuyerSection.parentElement.style.display = "block";

    }

    // All Buyers Summary

    allBuyersDetails.innerHTML = sortedBuyers.map(([buyer, data]) =>

        `<p><strong>${buyer}</strong> - Challans: ${data.count}, Total Amount: ₹${data.totalAmount.toFixed(2)}</p>`

    ).join("");

    allBuyersDetails.parentElement.style.display = "block";

    // Overall Summary

    overallSummarySection.innerHTML = `

        <p><strong>Total Buyers:</strong> ${sortedBuyers.length}</p>

        <p><strong>Total Challans:</strong> ${totalChallans}</p>

        <p><strong>Average Amount per Challan:</strong> ₹${(totalAmount / totalChallans).toFixed(2)}</p>

        <p><strong>Total Amount of All Challans:</strong> ₹${totalAmount.toFixed(2)}</p>

    `;

    overallSummarySection.parentElement.style.display = "block";

}

    </script>

</body>

</html>