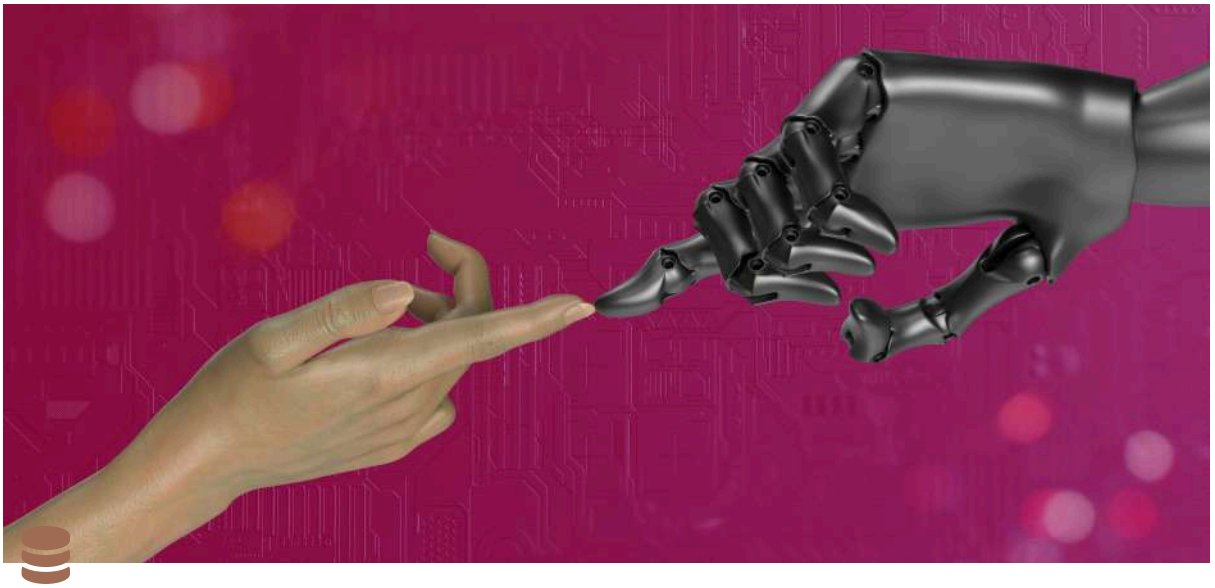


🌾Agmarknet Data Automation

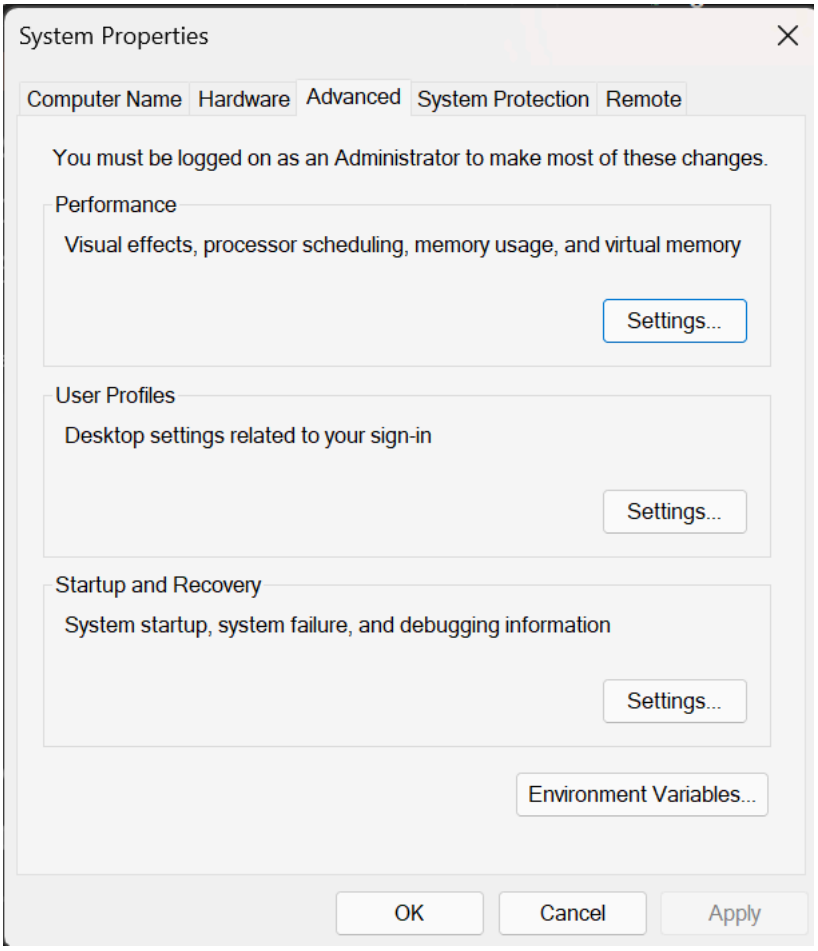
Description	This project automates downloading agricultural market data (arrival & price) from the **Agmarknet** website using Python and **Playwright**. It extracts Excel reports for multiple commodities and date ranges automatically — useful for data analysis in Power BI, Excel, or Python.
Files & media	commodities.csv dates.csv download_agmark_excel.py

▼ Tool Installation

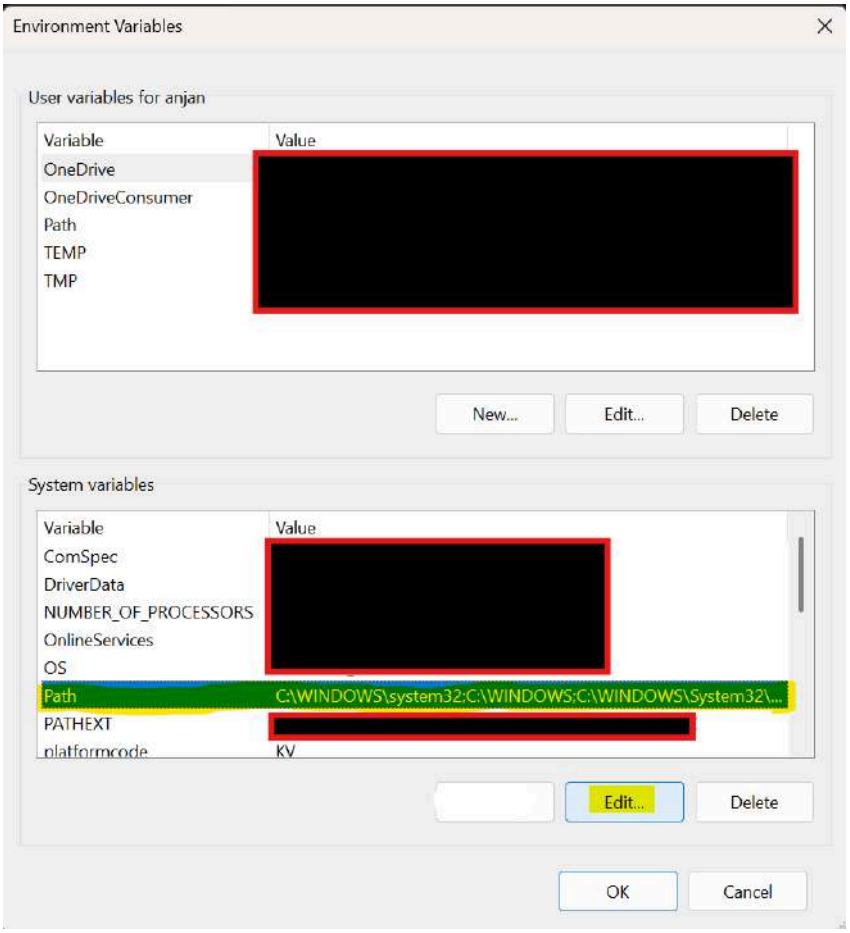
1. **Install Python**
   - Download Python from https://www.python.org/downloads/.
   - During installation:
     - ✅ Check **"Add Python to PATH"**.
   - Verify installation:

```
python --version
```

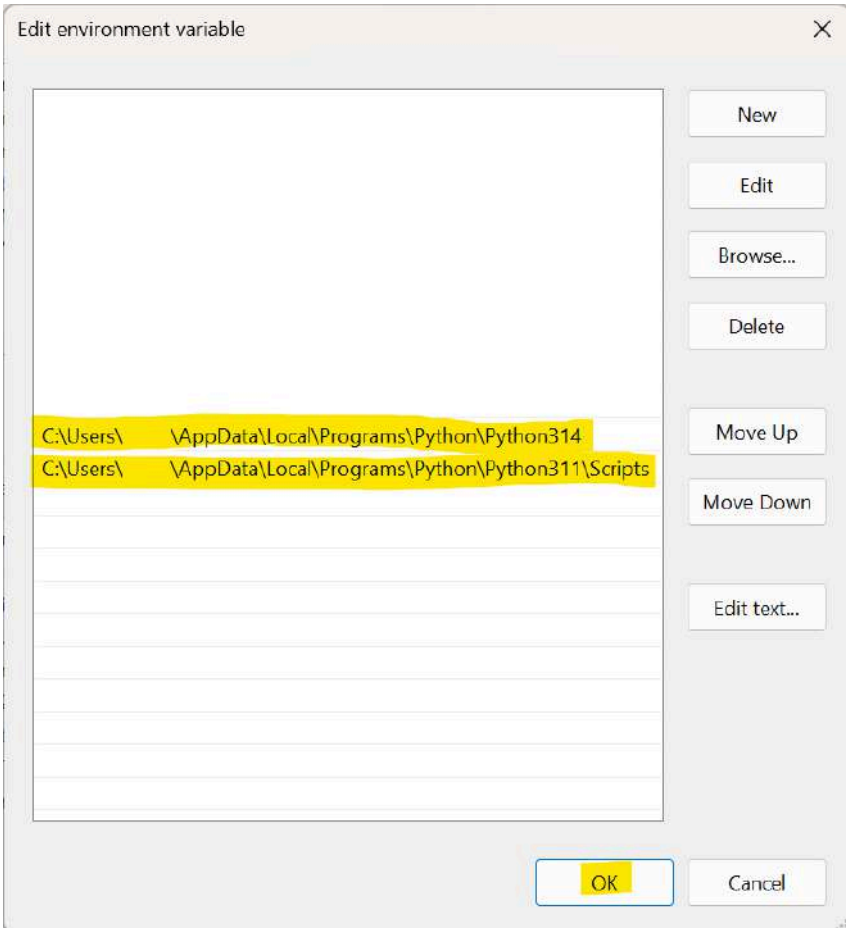
- If you wont get Python version when you check in bash(PowerShell/cmd) then follow below steps to add PATH manually
  1. Press **Win + S**, type **"Environment Variables"** → **Edit system environment variables** → **Environment Variables**.



2. Under **User variables**, select **Path** → **Edit** → **New**.



3. Paste the Python folder path you copied (the one with python.exe).C:\Users\YourName\AppData\Local\Programs\Python\Python311
4. Also, add the **Scripts folder** inside it (needed for pip), e.g.: C:\Users\YourName\AppData\Local\Programs\Python\Python311\Scripts



5. Click **OK** → Close all windows.

2. **Install Visual Studio Code (VS Code)**
   - Download from https://code.visualstudio.com/.
   - Open VS Code after installation.
   - Install **Python Extension**:
     - Go to Extensions (Ctrl+Shift+X)
     - Search for **Python**
     - Click **Install**

3. **Playwright Installation & Setup Guide**

- Open the terminal in the directory where you want to download or run your project files. If using **VS Code**, open the **Integrated Terminal** (`Ctrl + ~`).
- Run the following command to ensure your package manager (`pip`) is up to date:

```
python -m pip install --upgrade pip
```

  ✅ **Purpose:** Keeps your Python environment updated and avoids installation errors.
- Run this command to install all necessary dependencies:

```
python -m pip install playwright pandas openpyxl python-dateutil
```

  ✅ **Includes:**
  - **playwright** → for browser automation and web scraping
  - **pandas** → for data handling and analysis
  - **openpyxl** → for reading/writing Excel files
  - **python-dateutil** → for flexible date and time handling
- Playwright needs browser engines to run. Install them using:

```
python -m playwright install
```

  ✅ **This installs:**
  - Chromium (Google Chrome)
  - Firefox
  - WebKit (Safari)
- To confirm Playwright is working, run:

```
python -c "from playwright.sync_api import sync_playwright; print('Playwright works!')"
```

  If successful, you'll see the message:

```
Playwright works!
```

- Other Required Libraries

```
pip install pandas
```

  (Optional but recommended for reading/writing CSVs.)

▼ Preparing Input files

Prepare the Input CSVs

1. 📄 commodities.csv

   Create a CSV file with commodity names and their corresponding IDs (from Agmarknet dropdown).

CommodityName	CommodityCode
Apple	17
Banana	19
Tomato	78
Onion	23
Potato	24

   Extracting Commodity Names and Codes from Agmarknet(However I have given the list)

Objective is to obtain the **commodity name–ID mapping** (e.g., *Apple → 17*) from the dropdown menu of the Agmarknet search page, which is required to automate data downloads using Playwright.

Use the Agmarknet "Search" page (manual check)

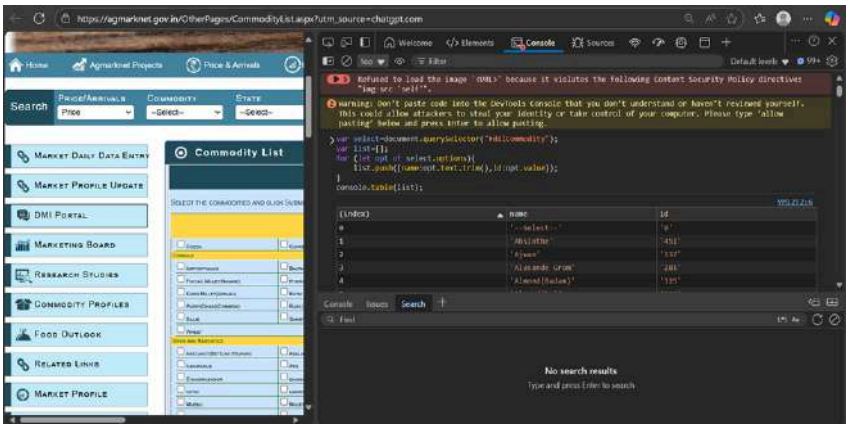
1. Go to 👉 https://agmarknet.gov.in/SearchCmmMkt.aspx
2. Right-click → **Inspect** → open **Console** tab.
3. Paste this JavaScript snippet and hit **Enter**:

```
var select = document.querySelector("#ddlCommodity");
 var list = [];
 for (let opt of select.options) {
    list.push({ name: opt.text.trim(), id: opt.value });
 }
 console.table(list);
```

4. ✅ You'll get a full table of all commodities with their IDs — exactly what's used in URLs like:

```
Tx_Commodity=17  ⇒ Apple
 Tx_Commodity=73  ⇒ Water Melon
```

5. Select all(ctrl+a) copy and paste in excel



2. 📄 dates.csv

Define all your from–to date periods. Keep date ranges short (1–3 months).

FromDate	ToDate
01-Jan-2018	31-Mar-2018
01-Apr-2018	30-Jun-2018
01-Jul-2018	30-Sep-2018

Note: Every time you make changes in csv files please save it and close before running the script

▼ Python Automation Script

Create a new file in VS Code and Save the Automation Script (download_agmark_excel.py)

Paste this final working code 👇

```
from playwright.sync_api import sync_playwright
import pandas as pd
import os
import time

# ========== CONFIG ================
BASE_URL = "https://agmarknet.gov.in/SearchCmmMkt.aspx"
DOWNLOAD_DIR = "downloads_agmark"
FAILED_CSV = "failed_downloads.csv"

os.makedirs(DOWNLOAD_DIR, exist_ok=True)

# ======== READ INPUT FILES =========
commodities = pd.read_csv("commodities.csv")   # CommodityName,CommodityCode
dates = pd.read_csv("dates.csv")               # FromDate,ToDate

failed_rows = []

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    context = browser.new_context(accept_downloads=True)
    page = context.new_page()

    for _, c_row in commodities.iterrows():
        commodity_name = c_row["CommodityName"]
        commodity_code = str(c_row["CommodityCode"])

        for _, d_row in dates.iterrows():
            from_date = d_row["FromDate"]
            to_date = d_row["ToDate"]

            print(f"\n🌐 Processing {commodity_name} | {from_date} → {to_date}")

            # Build URL
            url = (
                f"{BASE_URL}?Tx_Commodity={commodity_code}"
                f"&Tx_State=0&Tx_District=0&Tx_Market=0"
                f"&DateFrom={from_date}&DateTo={to_date}"
                f"&Fr_Date={from_date}&To_Date={to_date}"
                f"&Tx_Trend=2"
                f"&Tx_CommodityHead={commodity_name}"
                f"&Tx_StateHead=--Select--&Tx_DistrictHead=--Select--&Tx_MarketHead=--Select--"
            )

            try:
                page.goto(url, timeout=120000)

                # Wait until Excel button appears
                page.wait_for_selector("#cphBody_ButtonExcel", timeout=60000)

                # Download Excel
                print("📥 Clicking 'Export To Excel'...")
                with page.expect_download() as download_info:
                    page.click("#cphBody_ButtonExcel")

                download = download_info.value
                filename = f"{commodity_name}_{from_date}_to_{to_date}.xls"
```

```
            save_path = os.path.join(DOWNLOAD_DIR, filename)
            download.save_as(save_path)

            print(f"✅ Downloaded: {filename}")

            # Small delay between downloads
            time.sleep(3)

        except Exception as e:
            print(f"❌ Failed: {commodity_name} ({from_date} → {to_date}) | {e}")
            failed_rows.append({
                "CommodityName": commodity_name,
                "CommodityCode": commodity_code,
                "FromDate": from_date,
                "ToDate": to_date,
                "Error": str(e)
            })
            continue

    # Save failed downloads
    if failed_rows:
        pd.DataFrame(failed_rows).to_csv(FAILED_CSV, index=False)
        print(f"\n⚠️ Some downloads failed. Logged in {FAILED_CSV}")

    browser.close()

print("\n🎯 All done.")
```

✅ **To Run the Script:**

1. Open your terminal in the folder where your script is saved.
2. Make sure you have:
   - `commodities.csv` → with columns: `CommodityName, CommodityCode`
   - `dates.csv` → with columns: `FromDate, ToDate`
3. Run the command:

```
download_agmark_excel.py
```

4. Downloads will be saved in the folder **`downloads_agmark`**.
5. Any failed downloads will be listed in **`failed_downloads.csv`**.

CommodityName	CommodityCode	FromDate	ToDate	Error
Pomegranate	190	6-Sep-18	6-Oct-18	Timeout 30000ms exceeded while waiting for "download"========== logs ==========waiting for eve "download"==========

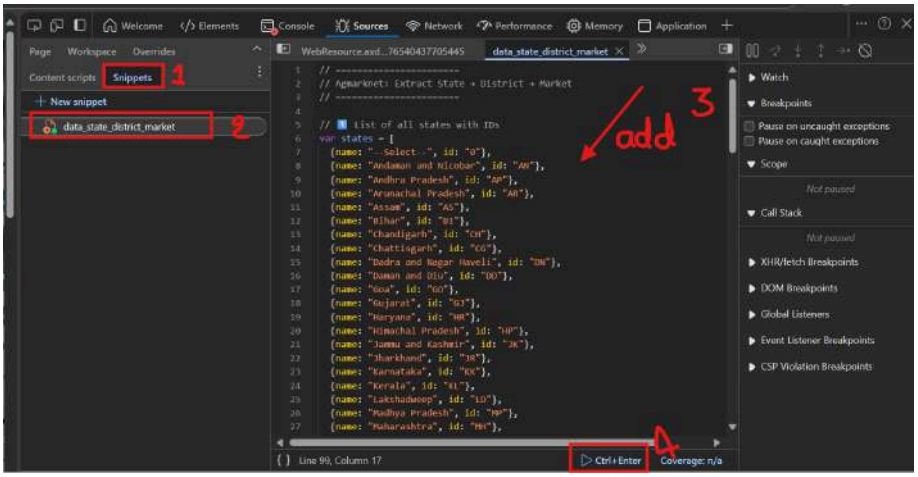
▼ Get all State_District_Market & their ID

- Go to Agriculture Marketing and press Ctrl+Shift+I to inspect.
- First get state_list and Id by typing following in console

```
var select = document.querySelector("#ddlStates");
var list = [];
for (let opt of select.options) {
    list.push({ name: opt.text.trim(), id: opt.value });
```

```
}
console.table(list);
```

- Open DevTools → **Sources tab** → **Snippets**.
- Click **New Snippet**, give it a name.
- Paste your full script into the snippet editor.
- Right-click → **Run**, or press `Ctrl + Enter`.



- Here's the **final code** to paste in a **Chrome/Edge DevTools Snippet**:

```
// ========================
// Agmarknet: Extract State → District → Market (Test 4 States)
// ========================

// List of states with IDs
var states = [
  {name: "Andaman and Nicobar", id: "AN"},
  {name: "Andhra Pradesh", id: "AP"},
  {name: "Arunachal Pradesh", id: "AR"},
  {name: "Assam", id: "AS"},
  {name: "Bihar", id: "BI"},
  {name: "Chandigarh", id: "CH"},
  {name: "Chattisgarh", id: "CG"},
  {name: "Dadra and Nagar Haveli", id: "DN"},
  {name: "Daman and Diu", id: "DD"},
  {name: "Goa", id: "GO"},
  {name: "Gujarat", id: "GJ"},
  {name: "Haryana", id: "HR"},
  {name: "Himachal Pradesh", id: "HP"},
  {name: "Jammu and Kashmir", id: "JK"},
  {name: "Jharkhand", id: "JR"},
  {name: "Karnataka", id: "KK"},
  {name: "Kerala", id: "KL"},
  {name: "Lakshadweep", id: "LD"},
  {name: "Madhya Pradesh", id: "MP"},
  {name: "Maharashtra", id: "MH"},
  {name: "Manipur", id: "MN"},
  {name: "Meghalaya", id: "MG"},
  {name: "Mizoram", id: "MZ"},
  {name: "Nagaland", id: "NG"},
  {name: "NCT of Delhi", id: "DL"},
```

```
  {name: "Odisha", id: "OR"},
  {name: "Pondicherry", id: "PC"},
  {name: "Punjab", id: "PB"},
  {name: "Rajasthan", id: "RJ"},
  {name: "Sikkim", id: "SK"},
  {name: "Tamil Nadu", id: "TN"},
  {name: "Telangana", id: "TL"},
  {name: "Tripura", id: "TR"},
  {name: "Uttar Pradesh", id: "UP"},
  {name: "Uttrakhand", id: "UC"},
  {name: "West Bengal", id: "WB"}
];

// Array to store results
var allData = [];

// Helper function to wait
function wait(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}

// Main async function
async function getAllMarkets() {
  for (let s of states) {
    try {
      console.log(`⏳ Processing state: ${s.name}`);

      // Select state
      var stateDropdown = document.querySelector("#ddlState");
      stateDropdown.value = s.id;
      if (typeof stateDropdown.onchange === "function") stateDropdown.onchange();

      await wait(8000); // wait for districts to load

      // Get districts
      var districts = document.querySelectorAll("#ddlDistrict option");
      for (let d of districts) {
        if (d.value === "0") continue; // skip "--Select--"

        try {
          var districtDropdown = document.querySelector("#ddlDistrict");
          districtDropdown.value = d.value;
          if (typeof districtDropdown.onchange === "function") districtDropdown.onchange();

          await wait(6000); // wait for markets to load

          // Get markets
          var markets = document.querySelectorAll("#ddlMarket option");
          for (let m of markets) {
            if (m.value === "0") continue; // skip "--Select--"
            allData.push({
              state: s.name,
              stateId: s.id,
              district: d.text.trim(),
              districtId: d.value,
              market: m.text.trim(),
              marketId: m.value
            });
```

```
        }
      } catch (err) {
        console.error(`❌ Error processing district ${d.text} in ${s.name}:`, err);
        continue;
      }
    }

    console.log(`✅ Completed: ${s.name}`);
  } catch (err) {
    console.error(`❌ Error with state ${s.name}:`, err);
    continue;
  }
}

console.table(allData);
console.log("✅ Finished processing test states!");

// Optional: copy to clipboard as CSV
if (allData.length > 0) {
  const header = Object.keys(allData[0]).join(",");
  const rows = allData.map(obj => Object.values(obj).join(",")).join("\n");
  const csv = header + "\n" + rows;
  copy(csv);
  console.log("CSV copied to clipboard. You can paste it into Excel.");
}
}

// Run the script
getAllMarkets();
```

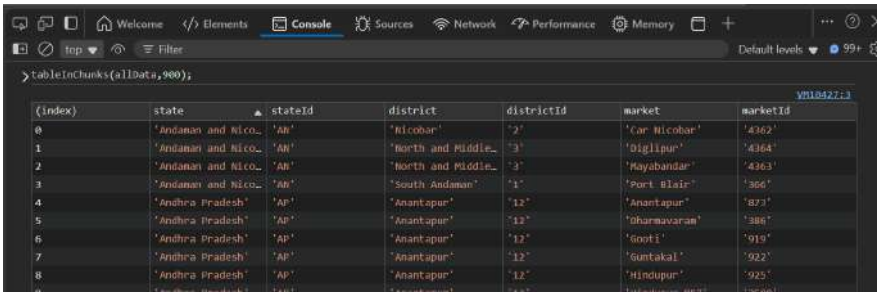
- Once Complted,to display large datasets in smaller, manageable tables inside the browser console — because the console only shows a limited number of rows per table (typically 1000).

  Type this in console 👇

```
function tableInChunks(array, chunkSize = 900) {
  for (let i = 0; i < array.length; i += chunkSize) {
    console.table(array.slice(i, i + chunkSize));
  }
}
```

✅ Next run 👇

```
tableInChunks(allData, 900);
```



- Then copy table by selecting Ctrl+A and paste from all tables in Excel

▼ Benefits for Stakeholders

Farmers

- Access to real-time market prices helps farmers decide when and where to sell their produce for maximum profit.
- Historical data analysis enables better planning for future crop cycles.
- Reduced dependency on intermediaries for market information.

Traders and Merchants

- Quick access to price trends across multiple markets facilitates better trading decisions.
- Automated alerts for significant price movements or market opportunities.
- Comprehensive data for inventory management and procurement planning.

Policy Makers and Researchers

- Aggregated data provides insights into agricultural market dynamics.
- Historical trends support evidence-based policy formulation.
- Market analysis helps identify areas requiring intervention or support.