Trend Discovery Agent – Full Workflow


```
{
  "name": "Market Trend Discovery Agent",
  "nodes": [
    {
      "parameters": {
        "options": {}
      },
      "type": "@n8n/n8n-nodes-langchain.chatTrigger",
      "typeVersion": 1.4,
      "position": [
        0,
        -400
      ],
      "id": "ca240351-45df-4844-86b9-b8b838fdac7a",
      "name": "When chat message received",
      "webhookId": "8368411a-2133-49d5-aaec-1c3ab5176d15"
    },
    {
      "parameters": {
        "options": {}
      },
      "type": "@n8n/n8n-nodes-langchain.lmChatGoogleGemini",
      "typeVersion": 1,
      "position": [
        240,
        -176
      ],
      "id": "699974ec-4e34-465c-ad89-968649e6291f",
      "name": "Google Gemini Chat Model",
```

```
      "credentials": {
        "googlePalmApi": {
          "id": "RjFd103QS7sBuu6k",
          "name": "Google Gemini(PaLM) Api account"
        }
      }
    },
    {
      "parameters": {},
      "type": "@n8n/n8n-nodes-langchain.memoryBufferWindow",
      "typeVersion": 1.3,
      "position": [
        368,
        -176
      ],
      "id": "aa986326-023d-42bf-b935-fdc49ce28882",
      "name": "Simple Memory"
    },
    {
      "parameters": {
        "options": {
          "systemMessage": "You are a user intent classification
specialist.\n\nYour task: Determine if the user's input relates to
rugs trends.\n\nClassification Rules:\n\nOutput {\"trend\": \"yes\"}
if the user:\n• Mentions product trends, trending topics, or market
trends\n• Shares URLs from e-commerce platforms (Amazon, Walmart)
suggesting trending items\n• Asks about what's popular, trending, or
gaining traction\nOutput {\"trend\": \"no\"} if the user:\n• Asks
general questions unrelated to trends\n• Shares content without trend
indicators\n\nResponse Format:\n\n•Return ONLY valid JSON with no
additional text or explanation\n•Base classification strictly on the
provided input—do not infer beyond what's explicitly stated\n•Analyze
both direct mentions and contextual signals (URLs, keywords,
phrasing)"
        }
      },
```

```
      "type": "@n8n/n8n-nodes-langchain.agent",
      "typeVersion": 3,
      "position": [
        224,
        -400
      ],
      "id": "ecdbcff6-b4a1-4a09-9eed-600219f661fb",
      "name": "Find User Intent"
    },
    {
      "parameters": {
        "jsCode": "// Input example:\n// [\n//   {\n//     \"output\": \"```json\\n{\\n  \\\"trend\\\": \\\"no\\\",\\n  \\\"answer\\\": \\\"Hello! How can I help you today?\\\"\\n}\\n```\"\n//   },\n// {\n//     \"output\": \"```json\\n{\\n  \\\"trend\\\": \\\"yes\\\"\\n}\\n```\"\n//   }\n// ]\n\nreturn items.map(item => {\n let raw = item.json.output;\n\n  // Remove markdown formatting like ```json ... ```\n  raw = raw.replace(/```json|```/g, \"\").trim();\n\n let parsed;\n  try {\n    parsed = JSON.parse(raw);\n  } catch (e) {\n parsed = { trend: \"no\", answer: \"Parsing error\" };\n  }\n\n return {\n    json: parsed\n  };\n});"
      },
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [
        576,
        -400
      ],
      "id": "9e7c70f8-6985-4860-a909-93c68a0f11fb",
      "name": "Code in JavaScript"
    },
    {
      "parameters": {
        "conditions": {
          "options": {
```

```
        "caseSensitive": true,
        "leftValue": "",
        "typeValidation": "strict",
        "version": 2
      },
      "conditions": [
        {
          "id": "d2eac434-0edb-4847-90e9-a3fb8cc50c85",
          "leftValue": "={{$json[\"trend\"]}}",
          "rightValue": "yes",
          "operator": {
            "type": "string",
            "operation": "equals",
            "name": "filter.operator.equals"
          }
        }
      ],
      "combinator": "and"
    },
    "options": {}
  },
  "type": "n8n-nodes-base.if",
  "typeVersion": 2.2,
  "position": [
    800,
    -400
  ],
  "id": "19565e8e-b088-4071-a9a1-1bc9fd2793d3",
  "name": "If"
},
{
  "parameters": {
```

```
        "promptType": "define",
        "text": "={{ $('When chat message
received').item.json.chatInput }}\n",
        "batching": {}
      },
      "type": "@n8n/n8n-nodes-langchain.chainLlm",
      "typeVersion": 1.7,
      "position": [
        1024,
        112
      ],
      "id": "680ef556-cf94-441a-90b7-ebc667cabe88",
      "name": "Basic LLM Chain"
    },
    {
      "parameters": {
        "options": {}
      },
      "type": "@n8n/n8n-nodes-langchain.lmChatGoogleGemini",
      "typeVersion": 1,
      "position": [
        1104,
        336
      ],
      "id": "44493681-1a12-4733-8d3e-df762b1c2772",
      "name": "Google Gemini Chat Model1",
      "credentials": {
        "googlePalmApi": {
          "id": "RjFd103QS7sBuu6k",
          "name": "Google Gemini(PaLM) Api account"
        }
      }
```

```
    },
    {
      "parameters": {
        "url": "https://locustlanerugs.com/blogs/the-lane.atom",
        "options": {}
      },
      "type": "n8n-nodes-base.rssFeedRead",
      "typeVersion": 1.2,
      "position": [
        1088,
        -688
      ],
      "id": "8dac45a0-f8b7-4fe4-aad6-02908452a948",
      "name": "RSS-Locust"
    },
    {
      "parameters": {
        "url": "https://www.greendecore.co.uk/blogs/home-and-garden-
decor.atom",
        "options": {}
      },
      "type": "n8n-nodes-base.rssFeedRead",
      "typeVersion": 1.2,
      "position": [
        1088,
        -880
      ],
      "id": "f5f884f7-2d18-4885-a5aa-ece0d4456ca4",
      "name": "RSS-Garden Decor"
    },
    {
      "parameters": {
```

```
      "url": "https://www.obeetee.in/blogs/know-your-carpet.atom",
      "options": {}
    },
    "type": "n8n-nodes-base.rssFeedRead",
    "typeVersion": 1.2,
    "position": [
      1088,
      -496
    ],
    "id": "31d0d44a-63fa-490e-a1df-65ba624983f7",
    "name": "RSS-Obeetee"
  },
  {
    "parameters": {
      "url": "https://blog.sisalcarpet.com/feed/",
      "options": {}
    },
    "type": "n8n-nodes-base.rssFeedRead",
    "typeVersion": 1.2,
    "position": [
      1088,
      -192
    ],
    "id": "2785062f-3473-40b0-abe9-10c8da3fe9c9",
    "name": "RSS-Sisal"
  },
  {
    "parameters": {
      "numberInputs": 4
    },
    "type": "n8n-nodes-base.merge",
    "typeVersion": 3.2,
```

```
      "position": [
        1376,
        -624
      ],
      "id": "6e652f48-a318-45f9-8ea8-b400bcac2909",
      "name": "Merge - RSS Feeds"
    },
    {
      "parameters": {
        "fieldToSplitOut": "title, link, pubDate, content,
contentSnippet, isoDate",
        "options": {}
      },
      "type": "n8n-nodes-base.splitOut",
      "typeVersion": 1,
      "position": [
        1600,
        -592
      ],
      "id": "01ef20ce-d2f9-4e9a-8172-ba17d9541285",
      "name": "Split - RSS Fields"
    },
    {
      "parameters": {
        "jsCode": "const output = [];\nconst now = new Date();\nconst
twoDaysAgo = new Date();\ntwoDaysAgo.setDate(now.getDate() - 7); //
only include last 7 days\n\nfor (const item of $input.all()) {\n
const isoDate = new Date(item.json.isoDate);\n  if (isoDate >=
twoDaysAgo && isoDate <= now) {\n    output.push(item);\n
}\n}\nreturn output;\n"
      },
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [
```

```
        1824,
        -592
      ],
      "id": "a2594981-0bde-4680-b8f3-0b21ac659feb",
      "name": "Filter - Recent Items"
    },
    {
      "parameters": {
        "url": "https://www.googleapis.com/customsearch/v1",
        "sendQuery": true,
        "queryParameters": {
          "parameters": [
            {
              "name": "key",
              "value": "AIzaSyA94keq6W6SAvJjap7MKFzZ6AM5GrS1LEA"
            },
            {
              "name": "cx",
              "value": "e3e0cfc950cdd43a7"
            },
            {
              "name": "q",
              "value": "={{ $('When chat message
received').item.json.chatInput }}"
            }
          ]
        },
        "options": {}
      },
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 4.3,
      "position": [
```

```
        2944,
        -336
      ],
      "id": "09ed2472-969f-4c69-8cd4-e0fdcc2ef17b",
      "name": "Google Search"
    },
    {
      "parameters": {
        "jsCode": "let inputText = $input.first().json.chatInput ||
$('When chat message received').first().json.chatInput;\n\n// Regex
patterns\nconst amazonRegex = /(https?:\\/\\/(?:www\\.)?amazon\\.[a-
z.]+\\/[^\\s]+)/gi;\n\n// Extract all links\nlet amazonLinks =
inputText.match(amazonRegex) || [];\n\n// Function to classify
links\nfunction classifyLinks(links, productPattern,
collectionPattern) {\n  let products = [];\n  let collections = [];\n
for (let link of links) {\n    if (productPattern.test(link)) {\n
products.push(link);\n    } else if (collectionPattern.test(link)) {\n
collections.push(link);\n    } else {\n      // Unknown type, you can
log or ignore\n    }\n  }\n  return { products, collections
};\n}\n\n// Amazon: products have /dp/ or /gp/product/, collections
/gp/bestsellers/ or /s?\nlet amazonClassified =
classifyLinks(amazonLinks, /\\/dp\\/|\\/gp\\/product\\//,
/\\/gp\\/bestsellers\\/|\\/s\\?/);\n\n// Return grouped arrays\nreturn
[\n  {\n    json: {\n      amazon_products:
amazonClassified.products,\n      amazon_collections:
amazonClassified.collections\n    }\n  }\n];"
      },
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [
        1088,
        -1072
      ],
      "id": "c05993de-a2aa-4de1-9e54-2ba1199c99f9",
      "name": "Find URLs"
    },
    {
      "parameters": {
```

```
        "jsCode": "// Get input JSON\nlet inputData =
$input.first().json;\n\n// Function to clean links (remove trailing
commas and whitespace)\nfunction cleanLinks(links) {\n    return
(links || []).map(l => l.trim().replace(/,+$/,
\"\")).filter(Boolean);\n}\n\n// Extract Amazon links: support old
'amazon' array or already separated arrays\nlet allAmazonLinks =
[];\nif (inputData.amazon) {\n    allAmazonLinks =
cleanLinks(inputData.amazon);\n} else {\n    // If already split
arrays exist, merge them\n    allAmazonLinks =
cleanLinks([...(inputData.amazon_products || []),
...(inputData.amazon_collections || [])]);\n}\n\n// Classify links:
\n// Products have /dp/ or /gp/product/\n// Collections have
/gp/bestsellers/ or /s?\nlet amazon_products = [];\nlet
amazon_collections = [];\n\nallAmazonLinks.forEach(link => {\n    if
(/\\/dp\\/|\\/gp\\/product\\//.test(link)) {\n
amazon_products.push(link);\n    } else if
(/\\/gp\\/bestsellers\\/|\\/s\\?/.test(link)) {\n
amazon_collections.push(link);\n    }\n});\n\n// Return only Amazon
products and collections\nreturn [\n    {\n        json: {\n
amazon_products,\n            amazon_collections\n        }\n
}\n];"

      },

      "type": "n8n-nodes-base.code",

      "typeVersion": 2,

      "position": [

        1376,

        -1072

      ],

      "id": "4d622080-709c-43ff-8eff-d66c74bc2ab2",

      "name": "Amazon Scraper"

    },

    {

      "parameters": {

        "jsCode": "// Get input JSON\nlet inputData =
$input.first().json;\n\n// Get Amazon product links safely\nlet
amazonProducts = (inputData.amazon_products || []).map(l =>
l.trim().replace(/,+$/, \"\")).filter(Boolean);\n\n// Return each
Amazon product link as its own object\nreturn amazonProducts.map(link
=> ({\n    json: { url: link }\n}));"

      },

      "type": "n8n-nodes-base.code",
```

```
      "typeVersion": 2,
      "position": [
        2496,
        -1168
      ],
      "id": "a214907f-3f4c-4658-9c8a-9dda88d7579b",
      "name": "Amazon Products"
    },
    {
      "parameters": {
        "url": "={{ $json.url }}",
        "sendHeaders": true,
        "headerParameters": {
          "parameters": [
            {
              "name": "User-Agent",
              "value": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36"
            }
          ]
        },
        "options": {
          "timeout": 10000
        }
      },
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 4.3,
      "position": [
        2720,
        -1168
      ],
      "id": "1860ef20-09e2-470c-b507-efff1a7b38d4",
```

```
      "name": "Get HTML Content"
    },
    {
      "parameters": {
        "jsCode": "const cheerio = require('cheerio');\nconst results = [];\n\nfor (const item of $input.all()) {\n  try {\n    const url = item.json.url;\n    const html = item.json.data;\n    const $ = cheerio.load(html);\n    \n    const name = $('#productTitle').text().trim();\n    const price = $('span.a-price span.a-price-whole').first().text().trim();\n    \n    const details = [];\n    $('#feature-bullets ul.a-unordered-list li span.a-list-item').each((i, el) => {\n      const text = $(el).text().trim();\n      if (text && text.length > 0) {\n        details.push(text);\n      }\n    });\n    \n    const pattern = $('#inline-twister-expanded-dimension-text-pattern_name').text().trim();\n    \n    results.push({\n      url,\n      name,\n      price,\n      details,\n      pattern,\n      scraped_at: new Date().toISOString()\n    });\n  } catch (error) {\n    console.error('Error processing item:', error);\n    results.push({\n      url: item.json.url || 'Unknown URL',\n      name: '',\n      price: '',\n      details: [],\n      pattern: '',\n      error: error.message,\n      scraped_at: new Date().toISOString()\n    });\n  }\n}\n\n// Return a single item containing all scraped data as an array\nreturn [{\n  json: {\n    products: results,\n    total_scraped: results.length,\n    successful_scrapes: results.filter(r => !r.error).length,\n    failed_scrapes: results.filter(r => r.error).length,\n    batch_scraped_at: new Date().toISOString()\n  }\n}];\nfor (const item of $input.all()) {\n  try {\n    const url = item.json.url;\n    const html = item.json.data;\n    const $ = cheerio.load(html);\n    \n    const name = $('#productTitle').text().trim();\n    const price = $('span.a-price span.a-price-whole').first().text().trim();\n    \n    const details = [];\n    $('#feature-bullets ul.a-unordered-list li span.a-list-item').each((i, el) => {\n      const text = $(el).text().trim();\n      if (text && text.length > 0) {\n        details.push(text);\n      }\n    });\n    \n    const pattern = $('#inline-twister-expanded-dimension-text-pattern_name').text().trim();\n    \n    results.push({\n      url,\n      name,\n      price,\n      details,\n      pattern,\n      scraped_at: new Date().toISOString()\n    });\n  } catch (error) {\n    console.error('Error processing item:', error);\n    results.push({\n      url: item.json.url || 'Unknown URL',\n      name: '',\n      price: '',\n      details: [],\n      pattern: '',\n      error: error.message,\n      scraped_at: new Date().toISOString()\n    });\n  }\n}\n\n// Return a single item containing all scraped data as an array\nreturn [{\n  json: {\n    products: results,\n    total_scraped: results.length,\n    successful_scrapes: results.filter(r => !r.error).length,\n    failed_scrapes: results.filter(r => r.error).length,\n    batch_scraped_at: new Date().toISOString()\n  }\n}];"
```

```
      },
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [
        2944,
        -1168
      ],
      "id": "07b47b1b-3990-4711-93c0-a16f4561f095",
      "name": "Get Amazon Product Detail"
    },
    {
      "parameters": {
        "jsCode": "// Get input JSON\nlet inputData =
$input.first().json;\n\n// Get Amazon collection links safely\nlet
amazonCollections = (inputData.amazon_collections || []).map(l =>
l.trim().replace(/,+$/, \"\")).filter(Boolean);\n\n// Return each
Amazon collection link as its own object\nreturn
amazonCollections.map(link => ({\n    json: { url: link }\n}));"
      },
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [
        1600,
        -976
      ],
      "id": "3cce60ee-354d-4393-956a-54b2fd75b6d3",
      "name": "Amazon Collections1"
    },
    {
      "parameters": {
        "url": "={{ $json.url }}",
        "options": {}
      },
      "type": "n8n-nodes-base.httpRequest",
```

```
  "typeVersion": 4.3,
  "position": [
    1824,
    -976
  ],
  "id": "de17eb1c-7b07-4d78-b414-59f4e8d536b0",
  "name": "Get HTML of website"
},
{
  "parameters": {
    "operation": "extractHtmlContent",
    "extractionValues": {
      "values": [
        {
          "key": "product_links",
          "cssSelector": "div div a[href*='/dp/']",
          "returnValue": "attribute",
          "attribute": "href",
          "returnArray": true
        }
      ]
    },
    "options": {}
  },
  "type": "n8n-nodes-base.html",
  "typeVersion": 1.2,
  "position": [
    2048,
    -976
  ],
  "id": "bce6dc5d-cdaa-44e8-8980-d5876dabccfd",
  "name": "Extract Amazon Product URLs"
```

```
    },
    {
      "parameters": {
        "jsCode": "for (const item of $input.all()) {\n  // Remove
duplicates from product_links array\n  const uniqueLinks = [...new
Set($input.first().json.product_links)];\n  \n  // Add base URL to
make complete URLs\n  const fullUrls = uniqueLinks.map(link => {\n
if (link.startsWith('/')) {\n      return
`https://www.amazon.in${link}`;\n    }\n    return link; // in case
it's already a full URL\n  });\n  \n  // Update the item with
processed URLs\n  item.json.product_links = fullUrls;\n}\n\nreturn
$input.all();"
      },
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [
        2272,
        -976
      ],
      "id": "9db48829-6a76-439d-bde6-31f50fbcc396",
      "name": "Make Amazon accessible URL1"
    },
    {
      "parameters": {
        "jsCode": "// Split the URLs into individual items for
processing one by one\nconst productLinks =
$input.first().json.product_links;\nconst results = [];\n\nfor (const
link of productLinks) {\n  results.push({\n    url: link\n
});\n}\n\nreturn results.map(r => ({ json: r }));"
      },
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [
        2496,
        -976
      ],
```

```
      "id": "95ce0771-587d-4702-adad-254b17f84a57",
      "name": "Split URLs1"
    },
    {
      "parameters": {
        "url": "={{ $json.url }}",
        "sendHeaders": true,
        "headerParameters": {
          "parameters": [
            {
              "name": "User-Agent",
              "value": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36"
            }
          ]
        },
        "options": {
          "timeout": 10000
        }
      },
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 4.3,
      "position": [
        2720,
        -976
      ],
      "id": "4ce8fe8b-c8f6-4389-8816-9985fd3c00a4",
      "name": "URL-wise HTML gather"
    },
    {
      "parameters": {
```

```
        "jsCode": "const cheerio = require('cheerio');\nconst results
= [];\n\nfor (const item of $input.all()) {\n  try {\n    const url =
item.json.url;\n    const html = item.json.data;\n    const $ =
cheerio.load(html);\n    \n    const name =
$('#productTitle').text().trim();\n    const price = $('span.a-price
span.a-price-whole').first().text().trim();\n    \n    const details =
[];\n    $('#feature-bullets ul.a-unordered-list li span.a-list-
item').each((i, el) => {\n      const text = $(el).text().trim();\n
if (text && text.length > 0) {\n        details.push(text);\n      }\n
});\n    \n    const pattern = $('#inline-twister-expanded-dimension-
text-pattern_name').text().trim();\n    \n    results.push({\n
url,\n      name,\n      price,\n      details,\n      pattern,\n
scraped_at: new Date().toISOString()\n    });\n  } catch (error) {\n
console.error('Error processing item:', error);\n    results.push({\n
url: item.json.url || 'Unknown URL',\n      name: '',\n      price:
'',\n      details: [],\n      pattern: '',\n      error:
error.message,\n      scraped_at: new Date().toISOString()\n    });\n
}\n}\n\n// Return a single item containing all scraped data as an
array\nreturn [{\n  json: {\n    products: results,\n
total_scraped: results.length,\n    successful_scrapes:
results.filter(r => !r.error).length,\n    failed_scrapes:
results.filter(r => r.error).length,\n    batch_scraped_at: new
Date().toISOString()\n  }\n}];\nfor (const item of $input.all()) {\n
try {\n    const url = item.json.url;\n    const html =
item.json.data;\n    const $ = cheerio.load(html);\n    \n    const
name = $('#productTitle').text().trim();\n    const price = $('span.a-
price span.a-price-whole').first().text().trim();\n    \n    const
details = [];\n    $('#feature-bullets ul.a-unordered-list li span.a-
list-item').each((i, el) => {\n      const text =
$(el).text().trim();\n      if (text && text.length > 0) {\n
details.push(text);\n      }\n    });\n    \n    const pattern =
$('#inline-twister-expanded-dimension-text-
pattern_name').text().trim();\n    \n    results.push({\n      url,\n
name,\n      price,\n      details,\n      pattern,\n      scraped_at:
new Date().toISOString()\n    });\n  } catch (error) {\n
console.error('Error processing item:', error);\n    results.push({\n
url: item.json.url || 'Unknown URL',\n      name: '',\n      price:
'',\n      details: [],\n      pattern: '',\n      error:
error.message,\n      scraped_at: new Date().toISOString()\n    });\n
}\n}\n\n// Return a single item containing all scraped data as an
array\nreturn [{\n  json: {\n    products: results,\n
total_scraped: results.length,\n    successful_scrapes:
results.filter(r => !r.error).length,\n    failed_scrapes:
results.filter(r => r.error).length,\n    batch_scraped_at: new
Date().toISOString()\n  }\n}];"

      },

      "type": "n8n-nodes-base.code",

      "typeVersion": 2,

      "position": [
```

```
        2944,
        -976
      ],
      "id": "acfbcba9-407a-44bd-a5c1-90c329927623",
      "name": "Amazon Scraper2"
    },
    {
      "parameters": {
        "numberInputs": 4
      },
      "type": "n8n-nodes-base.merge",
      "typeVersion": 3.2,
      "position": [
        3168,
        -720
      ],
      "id": "c8de5473-d201-4003-91d5-de94480db8a5",
      "name": "Merge - All Sources"
    },
    {
      "parameters": {
        "jsCode": "// n8n JavaScript Code to Process and Return All Data in One Array\n// Combines all products and articles into a single array\n\nconst allData = [];\n\n// Loop through all input items\nfor (const item of $input.all()) {\n  let dataArray = item.json;\n  \n  // Handle if data is wrapped in a \"data\" property\n  if (dataArray.data && Array.isArray(dataArray.data)) {\n    dataArray = dataArray.data;\n  }\n  \n  // If it's already an array, use it directly\n  if (Array.isArray(dataArray)) {\n    // Process each item in the array\n    dataArray.forEach(data => {\n      // Check if this is a products object (has products array)\n      if (data.products && Array.isArray(data.products)) {\n        // Process each product\n        data.products.forEach(product => {\n          if (!product.error) { // Skip failed scrapes\n            allData.push({\n              type: 'product',\n              name: product.name,\n              price: product.price,\n              url: product.url,\n              pattern: product.pattern,\n              details: product.details,\n              scraped_at: product.scraped_at,\n              myNewField: 1,\n
```

```
// Additional processing fields\n              price_numeric:
parseInt(product.price.replace(/[^\\d]/g, '')) || 0,\n
has_details: product.details && product.details.length > 0,\n
category: product.name.toLowerCase().includes('mat') ? 'mat' : \n
product.name.toLowerCase().includes('rug') ? 'rug' : 'carpet',\n
detail_count: product.details ? product.details.length : 0\n
});\n          }\n        });\n        \n        // Also add summary
data for the batch\n        allData.push({\n          type:
'scrape_summary',\n          total_scraped: data.total_scraped,\n
successful_scrapes: data.successful_scrapes,\n
failed_scrapes: data.failed_scrapes,\n          batch_scraped_at:
data.batch_scraped_at,\n          myNewField: 1\n        });\n
}\n      \n      // Check if this is a news article\n      else if
(data.title && data.link) {\n        allData.push({\n          type:
'article',\n          title: data.title,\n          link: data.link,\n
pubDate: data.pubDate,\n          isoDate: data.isoDate,\n
content: data.content,\n          contentSnippet:
data.contentSnippet,\n          myNewField: 1,\n          //
Additional processing fields\n          word_count:
data.contentSnippet ? data.contentSnippet.split(' ').length : 0,\n
has_content: Boolean(data.content),\n          publish_date: new
Date(data.isoDate),\n          domain: data.link.split('/')[2] ||
data.link,\n          source: data.link.split('/')[2] ===
'corporate.target.com' ? 'Target' : \n
data.link.split('/')[2] === 'ruginsider.com' ? 'Rug Insider' :
'Other'\n        });\n      }\n      \n      // Handle any other data
structure\n      else {\n        allData.push({\n          ...data,\n
type: 'other',\n          myNewField: 1,\n          processed_at: new
Date().toISOString()\n        });\n      }\n    });\n  }\n  // If it's
not an array, process as single item\n  else {\n    allData.push({\n
...dataArray,\n      type: 'single_item',\n      myNewField: 1,\n
processed_at: new Date().toISOString()\n    });\n  }\n}\n\n// Return
all data as a single array in one item\nreturn [{ json: { allData:
allData, totalItems: allData.length, processedAt: new
Date().toISOString() } }];"
      },
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [
        3392,
        -688
      ],
      "id": "10eef869-a36d-4836-bd30-73bdf936086b",
      "name": "Combine All Data"
    },
```

```
{
  "parameters": {
    "promptType": "define",
    "text": "={{ $json.allData }}",
    "options": {
      "systemMessage": "=You are an expert internet researcher and trend analyst for the \"rugs\" category. Produce a single self-contained HTML document (UTF-8) that reports current and emerging rug trends found today {{ $now }} up to the last 7 days (i.e., include only sources published or snapshot within the date range: {{ $now }} minus 7 days through {{ $now }}). If no rug-related signals appear in that window, return an HTML report stating that clearly and the date range checked.\n\nRules & scope (strict):\n• Only consider signals explicitly about rugs (area rugs, runners, round rugs, mats). Ignore any products or posts not related to rugs.\n• Sources to search include (but are not limited to): Amazon Best Sellers & New Releases, Walmart top products, Target product/press pages, IKEA blog, Wayfair, Pinterest boards & catalogs, retailer blogs, product catalogs, Google Search results, news sites, and retailer press releases.\n• Focus only on fresh trends from the last 7 days. Do not include older trends.\n• For each trend you include, cite the source name and snapshot/publish date (e.g., \"Amazon Best Sellers — Oct 9, 2025\") in the source line.\n• If a candidate source is a Google search result or scraped product page, include the product title, retailer, and snapshot date.\n• Scrape products, blogs, and news items and use those as the basis for trends. (If scraping is not possible in your environment, search and use public pages and metadata.)\n• Combine multiple sources for the same signal where possible (e.g., \"Amazon + Pinterest + Wayfair showing high interest in 'high-pile shag' this week\").\n\nData extraction & normalization (required for every rug trend):\nFor each trend, extract and normalize the following attributes. Always present attributes in a 2-column table using the Attributes table template below — do NOT use an inline bullet list for core attributes.\n\nAttributes to extract and normalize:\n• Size — examples: 2×3, 5×7, 8×10, runner, round. Use format 5×8 (lowercase x). If multiple sizes are common for the trend, list representative sizes separated by commas.\n• Color — dominant color or family (normalized; prefer terms like Grey, Cream, Earthy tones, Warm neutrals, Blue). Use singular/plural consistently.\n• Material — normalized to categories such as wool, jute, cotton, synthetic (polyester), viscose, polypropylene, etc.\n• Pattern — normalized to geometric, floral, abstract, distressed, kilim, solid, textured, etc.\n• Style — normalized to boho, minimalist, scandinavian, traditional, vintage, modern, transitional, etc.\n• Features — normalized tags like washable, low-pile, hand-knotted, eco-friendly, non-slip, reversible, high-pile, shag, etc.\n• Price — place the trend into price buckets using this exact formatting rule: $X to $Y or $Y+. Suggested buckets to use where applicable: under $25, $25 to $50, $50 to $75, $75 to
```

$200, $200 to $500, $500+. Always use $ and write ranges exactly as $75 to $200 etc. When you can, provide a short note with typical SKU price range in the analysis line (formatted like $75 to $200).\n\nFormatting & editorial rules (apply precisely):\n• Output must be valid HTML5, with embedded CSS in a <style> block at top that implements the style guide below.\n• Use Inter font (link to Google Fonts in the HTML head) and apply the typographic rules:\n • H1: Inter 700, 28px, color #444444, line-height 1.2\n • H2: Inter 600, 18px, color #444444, line-height 1.25\n • Body: Inter 400, 13px, color #000000, line-height 1.45\n • Label/meta: Inter 500 or italic 400, 11px, color #6B7280\n • Emphasis/key terms: Inter 600, same size as body, color #1F4E79\n• Colour palette exact hex:\n • Heading text: #444444\n • Body text: #000000\n • Accent/key labels: #1F4E79 (Dark Blue)\n • Secondary/muted: #6B7280\n • Table borders/dividers: #E6E6E6 or #F3F4F6\n• Page margins: set container padding to 32–40px. Use spacing rules: paragraph spacing 8–10px, section spacing 18–28px.\n• Use the Attributes table template for every trend. Table CSS must match this template:\n• Attribute table header left column must be Dark Blue #1F4E79 (Inter 600, 12px). Values right column black #000000. Source line (small grey) must be Inter 11px #6B7280 and placed directly below the trend H2 or under the attributes table.\n• For each trend include:\n1. H2 (trend title — Title Case)\n2. Source line (small grey) with source name(s) and snapshot/publish date(s) — format Oct 9, 2025 (MMM D, YYYY).\n3. Attributes table (Attribute | Values / examples) — follow the exact attribute names.\n4. A 1–2 sentence analysis (body text) in sentence case, with one compact actionable insight (e.g., \"Consider launching a washable, low-pile 5×8 in warm neutrals at $75 to $200 for Q4 promotion.\").\n5. Optional: one small image or screenshot placeholder (use <img> with alt text) if a visual supports the trend. Provide caption (Inter 11px #6B7280) under the image.\n• Always convert tokens like 75_200 to $75 to $200. Follow the \"Fixes to existing content\" rules from the style doc.\n• Sizes must use 5×8 format (lowercase x). Prices use commas for thousands (e.g., $1,250).\n• Use Oxford comma in lists.\n• For every factual/non-obvious claim (e.g., \"high-pile shag rising on Amazon + Pinterest this week\"), include the top 1–3 source lines (source name + snapshot date) and place them in a small \"Sources\" section at the end of the document (Appendix). Do not put full URLs in the HTML body; instead include the source title and date. If you scraped product titles, include them in the Appendix table with snapshot dates.\n• Accessibility: every <img> must include alt text. Use semantic HTML (headings, tables, paragraphs).\n\nOutput structure (required):\n• Title page: H1 with the main heading (Title Case) and one-line descriptor under it (small grey).\n• Executive summary: 2–3 short sentences summarizing the top 2–3 signals this week.\n• Trends: For each trend (ordered by strength of signal), one H2 lead, source line, Attributes table, 1–2 line analysis, optional image + caption.\n• Appendix: a small table listing raw sources with snapshot dates and short notes (e.g., \"Amazon Best Sellers — Oct 9, 2025 — top 10 show X\"). Use Inter 11px #6B7280 for the metadata lines.\n\nWhen extracting trends:\n• If a trend is supported by multiple product SKUs or multiple retailers, aggregate and show that in the analysis and in

```
the Appendix (list top SKUs with their price if available).\n• If the
data is ambiguous, still include the trend but explicitly mark
uncertainty (e.g., \"signal strength: low — only 1 retailer this
week\").\n• If the trend is non-rug or outside the 7-day window,
ignore it.\n\nDeliverable:\n• Return only the final HTML document (no
extra text or commentary).\n• The HTML must be ready to render and
follow the style guide above.\n• Ensure all price buckets and
attribute normalizations follow the editorial rules.\n\nIf you cannot
access live web pages in this environment, substitute \"snapshot\"
with the search date and include a note in the Appendix that the
content was collected via search metadata on the date range: ({{ $now
}} minus 7 days through {{ $now }}).\nEnd of prompt."
        }
      },
      "type": "@n8n/n8n-nodes-langchain.agent",
      "typeVersion": 3,
      "position": [
        3616,
        -688
      ],
      "id": "a2c5851a-0f48-426e-8da4-f540af767548",
      "name": "Analyze All Data"
    },
    {
      "parameters": {
        "options": {}
      },
      "type": "@n8n/n8n-nodes-langchain.lmChatGoogleGemini",
      "typeVersion": 1,
      "position": [
        3696,
        -464
      ],
      "id": "44cd0d1b-33f4-4f07-8a96-aecbee247d33",
      "name": "Google Gemini Chat Model2",
      "credentials": {
```

```
        "googlePalmApi": {
          "id": "RjFd103QS7sBuu6k",
          "name": "Google Gemini(PaLM) Api account"
        }
      }
    },
    {
      "parameters": {
        "jsCode": "let html = $input.first().json.output;\n\n// Remove
markdown fences like ```html or ```\nhtml =
html.replace(/```html|```/g, \"\");\n\n// Remove all \\n, \\r, \\t,
and excessive spaces\nhtml = html.replace(/\\\\[nrt]/g, \"
\").replace(/\\s{2,}/g, \" \");\n\n// Remove duplicate or misplaced
<html> / <body> wrappers\nhtml = html.replace(/<\\/?body><\\/?body>/g,
\"\")\n           .replace(/<\\/?html><\\/?html>/g, \"\")\n
.replace(/<body><!DOCTYPE html>/g, \"<!DOCTYPE html>\");\n\n// Trim
leading/trailing spaces\nhtml = html.trim();\n\n// Optional: Ensure
the final HTML starts and ends properly\nif
(!html.startsWith(\"<!DOCTYPE html>\")) {\n  html = `<!DOCTYPE
html><html><body>${html}</body></html>`;\n}\n\n// Return clean
minified HTML\nreturn [{ json: { cleaned_minified_html: html } }];"
      },
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [
        3968,
        -784
      ],
      "id": "6a5ec256-f8ea-4368-8a1b-a84317a76a41",
      "name": "Clean AI Output"
    },
    {
      "parameters": {
        "jsCode": "const text =
$input.first().json.cleaned_minified_html;\nconst buffer =
Buffer.from(text, 'utf8');\nconst binaryData = {\n  data:
buffer.toString('base64'),\n  mimeType: 'application/octet-stream',\n
```

```
fileName: 'file.html',\n};\nitems[0].binary = { data: binaryData
};\nreturn items;"
      },
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [
        4192,
        -784
      ],
      "id": "5e770873-f7e7-4e1f-8559-25de4e8809f8",
      "name": "Download HTML File"
    },
    {
      "parameters": {
        "html": "{{ $json.cleaned_minified_html }}"
      },
      "type": "n8n-nodes-base.html",
      "typeVersion": 1.2,
      "position": [
        4416,
        -784
      ],
      "id": "3e566411-babe-429c-a80d-5f0dc554453a",
      "name": "See HTML Output here"
    },
    {
      "parameters": {
        "assignments": {
          "assignments": [
            {
              "id": "089e219d-d11f-4adc-bced-76a4be559319",
              "name": "Project 2",
```

```
          "value": "={{ $json.output }}",
          "type": "string"
        }
      ]
    },
    "options": {}
  },
  "type": "n8n-nodes-base.set",
  "typeVersion": 3.4,
  "position": [
    3968,
    -592
  ],
  "id": "3a71fe43-bf51-48f8-bcf3-d8dc948bddf3",
  "name": "Edit Fields"
},
{
  "parameters": {
    "workflowId": {
      "__rl": true,
      "value": "n5KARnKrVd0YZaUg",
      "mode": "list",
      "cachedResultUrl": "/workflow/n5KARnKrVd0YZaUg",
      "cachedResultName": "Project 5 Sheet"
    },
    "workflowInputs": {
      "mappingMode": "defineBelow",
      "value": {}
    },
    "options": {}
  },
  "type": "n8n-nodes-base.executeWorkflow",
```

```
    "typeVersion": 1.3,
    "position": [
      4192,
      -592
    ],
    "id": "08802312-29fb-405c-a344-154a960bf408",
    "name": "Call 'Project 5 Sheet'"
  }
],
"pinData": {},
"connections": {
  "When chat message received": {
    "main": [
      [
        {
          "node": "Find User Intent",
          "type": "main",
          "index": 0
        }
      ]
    ]
  },
  "Google Gemini Chat Model": {
    "ai_languageModel": [
      [
        {
          "node": "Find User Intent",
          "type": "ai_languageModel",
          "index": 0
        }
      ]
    ]
```

```
},
"Simple Memory": {
  "ai_memory": [
    [
      {
        "node": "Find User Intent",
        "type": "ai_memory",
        "index": 0
      }
    ]
  ]
},
"Find User Intent": {
  "main": [
    [
      {
        "node": "Code in JavaScript",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Code in JavaScript": {
  "main": [
    [
      {
        "node": "If",
        "type": "main",
        "index": 0
      }
    ]
```

```
    ]
  },
  "If": {
    "main": [
      [
        {
          "node": "RSS-Garden Decor",
          "type": "main",
          "index": 0
        },
        {
          "node": "RSS-Locust",
          "type": "main",
          "index": 0
        },
        {
          "node": "RSS-Obeetee",
          "type": "main",
          "index": 0
        },
        {
          "node": "RSS-Sisal",
          "type": "main",
          "index": 0
        },
        {
          "node": "Google Search",
          "type": "main",
          "index": 0
        },
        {
          "node": "Find URLs",
```

```
        "type": "main",
        "index": 0
      }
    ],
    [
      {
        "node": "Basic LLM Chain",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Google Gemini Chat Model1": {
  "ai_languageModel": [
    [
      {
        "node": "Basic LLM Chain",
        "type": "ai_languageModel",
        "index": 0
      }
    ]
  ]
},
"RSS-Garden Decor": {
  "main": [
    [
      {
        "node": "Merge - RSS Feeds",
        "type": "main",
        "index": 0
      }
```

```
    ]
  ]
},
"RSS-Locust": {
  "main": [
    [
      {
        "node": "Merge - RSS Feeds",
        "type": "main",
        "index": 1
      }
    ]
  ]
},
"RSS-Obeetee": {
  "main": [
    [
      {
        "node": "Merge - RSS Feeds",
        "type": "main",
        "index": 2
      }
    ]
  ]
},
"RSS-Sisal": {
  "main": [
    [
      {
        "node": "Merge - RSS Feeds",
        "type": "main",
        "index": 3
```

```
      }
    ]
  ]
},
"Merge - RSS Feeds": {
  "main": [
    [
      {
        "node": "Split - RSS Fields",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Split - RSS Fields": {
  "main": [
    [
      {
        "node": "Filter - Recent Items",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Find URLs": {
  "main": [
    [
      {
        "node": "Amazon Scraper",
        "type": "main",
```

```
        "index": 0
      }
    ]
  ]
},
"Amazon Scraper": {
  "main": [
    [
      {
        "node": "Amazon Products",
        "type": "main",
        "index": 0
      },
      {
        "node": "Amazon Collections1",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Amazon Products": {
  "main": [
    [
      {
        "node": "Get HTML Content",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
```

```
"Get HTML Content": {
  "main": [
    [
      {
        "node": "Get Amazon Product Detail",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Amazon Collections1": {
  "main": [
    [
      {
        "node": "Get HTML of website",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Get HTML of website": {
  "main": [
    [
      {
        "node": "Extract Amazon Product URLs",
        "type": "main",
        "index": 0
      }
    ]
  ]
```

```
},
"Extract Amazon Product URLs": {
  "main": [
    [
      {
        "node": "Make Amazon accessible URL1",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Make Amazon accessible URL1": {
  "main": [
    [
      {
        "node": "Split URLs1",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Split URLs1": {
  "main": [
    [
      {
        "node": "URL-wise HTML gather",
        "type": "main",
        "index": 0
      }
    ]
```

```
    ]
  },
  "URL-wise HTML gather": {
    "main": [
      [
        {
          "node": "Amazon Scraper2",
          "type": "main",
          "index": 0
        }
      ]
    ]
  },
  "Filter - Recent Items": {
    "main": [
      [
        {
          "node": "Merge - All Sources",
          "type": "main",
          "index": 2
        }
      ]
    ]
  },
  "Get Amazon Product Detail": {
    "main": [
      [
        {
          "node": "Merge - All Sources",
          "type": "main",
          "index": 0
        }
```

```
    ]
  ]
},
"Google Search": {
  "main": [
    [
      {
        "node": "Merge - All Sources",
        "type": "main",
        "index": 3
      }
    ]
  ]
},
"Amazon Scraper2": {
  "main": [
    [
      {
        "node": "Merge - All Sources",
        "type": "main",
        "index": 1
      }
    ]
  ]
},
"Merge - All Sources": {
  "main": [
    [
      {
        "node": "Combine All Data",
        "type": "main",
        "index": 0
```

```
      }
    ]
  ]
},
"Combine All Data": {
  "main": [
    [
      {
        "node": "Analyze All Data",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Google Gemini Chat Model2": {
  "ai_languageModel": [
    [
      {
        "node": "Analyze All Data",
        "type": "ai_languageModel",
        "index": 0
      }
    ]
  ]
},
"Analyze All Data": {
  "main": [
    [
      {
        "node": "Clean AI Output",
        "type": "main",
```

```
        "index": 0
      },
      {
        "node": "Edit Fields",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Clean AI Output": {
  "main": [
    [
      {
        "node": "Download HTML File",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Download HTML File": {
  "main": [
    [
      {
        "node": "See HTML Output here",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
```

```
    "Edit Fields": {
      "main": [
        [
          {
            "node": "Call 'Project 5 Sheet'",
            "type": "main",
            "index": 0
          }
        ]
      ]
    }
  },
  "active": false,
  "settings": {
    "executionOrder": "v1"
  },
  "versionId": "a7f4d984-7322-4c25-a72e-e16f9955664c",
  "meta": {
    "templateCredsSetupCompleted": true,
    "instanceId":
"51fdafb3104f777c4b7b55f3a0359f97abc71602124cf0485bbacd488e70f639"
  },
  "id": "2CA8n2QkSD0WidsD",
  "tags": []
}
```