**Nielsen Commodity code**

**1. How Commodity Code Determination Works**

Commodity code is a unique identifier for product categories, critical for procurement automation and compliance. In the system, the code is determined as follows. If the item exists in the catalog, the linked commodity code is used directly. If it is a new, custom, or non-catalog item, the code is assigned using an LLM (GPT-4 via SAP AI Core) based on the product description and vendor profile. If a PDF quotation is uploaded, the code is extracted automatically for each quoted line item using text analysis and classification logic.

**2. Tables and Fields**

The main files used are:  
**materials.csv** contains the list of materials and their commodity codes. Key fields include MATERIAL\_DESCRIPTION, COMMODITY\_CODE, and PRICE.  
**vendors.csv** contains vendor data, including their preferred commodity codes and contract status. Main fields are VENDOR\_NAME, Vendor Commodity Code List, and CONTRACT\_EXISTS.  
**Nielsen Ariba Material Commodity Codes.csv** is the reference file with all possible commodity codes and their descriptions, with fields Commodity Code and Commodity Description.  
The PDF quotation itself contains unstructured data for analysis.

**3. Step-by-Step Logic**

For catalog items, the process is direct: user queries are matched against the catalog, and the commodity code is taken from the matched entry.  
For non-catalog (custom) requests, an LLM prompt is used that presents the user’s text, the available commodity codes, and the available vendors, then asks the LLM to recommend the most appropriate code and vendor, with explanations. The prompt for this is:

|  |
| --- |
| **You are a procurement expert analyzing a purchase request for a product not in our catalog.**  **User Request: "{user\_text}"**  **Available Commodity Codes in our system: [...list...]**  **Available Vendors: [...list...]**  **Instructions:**  **1. Analyze the user's text to understand what they want to buy**  **2. Recommend the MOST APPROPRIATE commodity code from our available codes**  **3. Recommend the BEST vendor who can supply this type of product**  **4. Consider vendor contracts and specializations**  **5. Provide reasoning for your recommendations**  **Respond in JSON format: { "recommended\_commodity\_code": "...", ... }** |

For PDF quotations, the process first extracts the vendor name and item descriptions from the document using LLM. For each item, the system then uses this function:  
get\_best\_commodity\_code\_for\_item(item\_description, vendor\_name, vendors\_df, all\_codes\_df)  
This calls a prompt like:

|  |
| --- |
| **You are an expert in procurement classification. You must determine the best Commodity Code for the given item description.**  **Item Description: "..."**  **Vendor: "..."**  **Vendor's Preferred Codes: ... (if any)**  **Here is the full list of available Commodity Codes: ... (code and description)**  **Instructions:**  **- If the vendor is known and has preferred codes, prioritize those codes unless there is a clear mismatch.**  **- For ambiguous names (like "notebook"), match based on vendor's industry.**  **- If you are not sure, choose the code that matches most closely based on both description and vendor profile.**  **- Reply with one line only in the format: <Commodity Code> - <Commodity Description>** |

The result is always a valid commodity code from the reference list, taking into account both text and vendor context.

**4. Why the System Is Flexible and Accurate**

The system achieves flexibility by combining LLM semantic analysis with structured vendor data. The model does not just match keywords, but interprets meaning and context, including the vendor’s specialization. Each vendor record contains a list of preferred commodity codes (Vendor Commodity Code List), so the system will prioritize those codes unless there is a clear mismatch. For ambiguous cases, like “notebook” for DELL versus OFFICE DEPOT, the classification logic uses the vendor profile to resolve the ambiguity. If no good match is found, the LLM selects the closest code from the master list.

The entire process is built on dynamic logic powered by LLMs (Large Language Models). There are no hardcoded rules or rigid if-then statements inside the system. Instead, the LLM is used to interpret the true meaning of the user’s request, item description, and vendor context, allowing for flexible and fuzzy matching even when the request does not exactly match catalog entries or typical commodity names. Because the logic is not hardwired but instead driven by the AI model’s semantic understanding, the system can adapt to a wide variety of phrasings and real-world purchase scenarios without requiring manual rule updates. This approach enables high flexibility and resilience in commodity code determination and vendor matching.