Option 1 (most common): Composite model

Keep your SQL table **live** via DirectQuery, and bring the NPPES side in as a small **Imported** table that you refresh periodically.

How it works

- **DirectQuery table**: your live SQL rows (already changing constantly).
- **Imported table**: NPPES "Candidates" fetched once per distinct (state, city) using the `NppesFetch` M function we built (or a **Dataflow**).
- **Join in the model**: relate your SQL table to the imported mapping using a key (ideally a stable `id`).

What you click

1. **Get Data** → **SQL Server** → **DirectQuery** → pick your live table (must have a stable key, e.g., `id`).
2. **Transform Data** → add the two helper functions:
   - `NppesFetch` (API + pagination)
   - `StreetTokens` (address tokens)
3. **New query** → build **Candidates** (distinct `state, city` → call `NppesFetch` → add tokens).
4. **New query** → **KeysForMatch**: *Imported* copy of just the keys you need from SQL (e.g., `id, state, city, address1`)
   - This is a tiny Import table (OK to refresh every 30–60 min).
   - Add `Tok = StreetTokens([address1])`.
5. In Power Query, **merge** `KeysForMatch` with `Candidates` twice:
   - On (`state, city, Tok`) → rank 1/2 (address+taxonomy / address+any)
   - On (`state, city`) → rank 3/4 (city+taxonomy / city+any)
   - Pick best per `id` → this yields **Mapping(id → NPI)** (Imported).
6. **Close & Apply**, then in Model view make a **relationship**:
   - SQL (DirectQuery) table `[id]` → Mapping (Imported) `[id]` (one-to-one or many-to-one).
7. Use `NPI` from **Mapping** alongside live SQL fields in visuals.

Pros: your main table stays live; NPPES piece refreshes on a schedule (hourly/daily).
Cons: the NPI mapping updates only when the Imported tables refresh.

If you're on **Power BI Pro**: up to 8 refreshes/day. **Premium**: up to 48/day. For most NPPES use cases (daily changes), this is usually fine.

Option 2 (all live, no Import): External enrichment service

If you truly need *everything* live with **zero Import**, do the enrichment **outside** Power BI:

Architecture

- A small service (Azure Function / Python job) that:
  - listens/polls your SQL for new/changed rows,
  - calls NPPES once per distinct (state, city) with caching,
  - writes the resolved **NPI** into a **separate store you control** (e.g., an **Azure SQL** table you're allowed to create).
- In Power BI, build a **composite model with two DirectQuery sources**:
  - DirectQuery to your original SQL table,
  - DirectQuery to the enrichment table (`id → NPI`),
  - Create a model relationship on `id`.

Pros: fully live, no Import.
Cons: you need a place to write the mapping (a DB you can create), plus a small background job.