



# ADDRESS CLEANER NL DOCUMENTATION

## SNOWFLAKE APPLICATION

Version	1.0
Version date	March 24, 2026
Classification	Public

## Colofon

### Contact

#### Support

**E** [support@edm.nl](mailto:support@edm.nl)

**T** +31-(0)88-8357071

#### General

**E** [info@edm.nl](mailto:info@edm.nl)

**T** +31-(0)88-8357072

### Disclaimer

© EDM DQ b.v., 2026 - All rights reserved. No part of this document may be reproduced or made public, in any form or by any means, without prior written permission from EDM DQ b.v.

# Table of Contents

- 1. Content Overview ..... 4**
  - Overview..... 4*
  - Data Definitions..... 5*
  - Examples..... 5*
  - Application workflow ..... 8*
  - SQL procedure workflow .....10*
  - In-app data management..... 9*
  
- 2. Snowflake Components .....12**
  - Application Installation..... 12*
  - Logging..... 12*
  - Grants and access .....13*
  - Usage and costs.....14*

# 1. Application Overview

## *Purpose*

EDM Address Cleaner NL is a Snowflake Native Application that cleans, standardizes, and validates Dutch address data. It operates entirely within the consumer's Snowflake account, meaning no data leaves the environment at any point during processing. The application is designed to improve the quality and consistency of address data stored in Snowflake. It is particularly suited for organizations that maintain Dutch customer, prospect, or location datasets and require up-to-date validation of addresses for downstream analytics, governance, or reporting.

## *Key Capabilities*

The application performs the following operations on address data:

- **Splitting:** separates combined address fields (e.g., street name and house number stored in a single column) into individual components.
- **Cleaning:** removes non-standard characters, trims whitespace, and adjusts formatting inconsistencies across all address fields.
- **Standardization:** formats address values according to Dutch conventions (uppercase postal codes, standardized street names, etc.).
- **Validation:** matches each address record against our monthly updated reference dataset to determine whether the address is a valid, registered Dutch address.
- **Scoring:** assigns a match score to each record, classifying it into one of five quality categories.
- **Enrichment:** adds missing address values to validated addresses, such as street names based on postal code and house number (addition) combinations, or the other way around.

## *Usage Options*

The application supports two usage modes:

- **Streamlit UI:** A step-by-step visual interface for interactive, ad-hoc cleaning processes. Suitable for one-time data quality checks or when a visual interface is preferred.
- **SQL Procedures:** Direct SQL calls for integration into automated data pipelines, scheduled tasks, or scripted workflows. No UI interaction required.

## 2. Data Specification

### *Address Reference Data*

The application validates addresses against our reference database, which contains over 10 million address records covering every officially registered address in the Netherlands, as well as historical addresses that were officially registered once, but which are now no longer active.

The data provided within the application is updated monthly.

### *Input Data Fields*

The application accepts five address fields as input. Each field must be mapped to a column in the input table, either through the Streamlit UI or as parameters in the SQL procedure call. All fields accept VARCHAR values.

Field	Description
<b>Street</b>	Street name (straatnaam). If your input contains a combined address in a single column (e.g., "Keizersgracht 100 A"), map that column as Street. The application will split it automatically.
<b>House number</b>	House number (huisnummer). Numeric value, typically 1–99999. May contain non-numeric characters that will be split to a house number addition during processing.
<b>House number addition</b>	House number addition (huisnummertoevoeging). Alphanumeric suffix such as "A", "II", "HS", "BIS", or "3-hoog". The application strips known non-address tokens (e.g., "HOOG", "BEL", "SOUS") during cleaning.
<b>Postal code</b>	Dutch postal code (postcode). Format: 4 digits + 2 letters (e.g., "1234 AB"). The application accepts variations such as "1234AB", "1234ab", and "1234 ab" and normalizes them.
<b>City</b>	City or place name (woonplaats). The application handles common variations (e.g., "S GRAVENHAGE" for "Den Haag", "IJSELSTEIN UT").

The cleaning process requires at least one of the following combinations of mapped fields:

- Postal code + House number
- Street + House number + City
- Street only (when addresses are combined in a single column)

### Output Data Fields

The output table is created in the OUTPUT schema of the application database. It contains all original columns from the input table plus the following additional columns:

Column	Type	Description
EDM_PROCESS_ID	NUMBER	Internal row identifier assigned during processing.
OUTPUT_STREET	VARCHAR	Cleaned and validated street name.
OUTPUT_HOUSENUMBER	VARCHAR	Cleaned and validated house number.
OUTPUT_HOUSENUMBERADDITION	VARCHAR	Cleaned and validated house number addition.
OUTPUT_POSTALCODE	VARCHAR	Standardized postal code (uppercase, no spaces).
OUTPUT_CITY	VARCHAR	Validated city/place name (woonplaats).
OUTPUT_MUNICIPALITY	VARCHAR	Municipality (gemeente) derived from postal code.
OUTPUT_PROVINCE	VARCHAR	Province (provincie) derived from postal code.
ADDRESS_CORRECTION	BOOLEAN	TRUE if the output differs from the input for any address field; NULL if no changes were made.
MATCHSCORE	VARCHAR	Color-coded quality indicator: GREEN, ORANGE, or RED. See Match Score Categories below.
MATCHSCORE_DESC	VARCHAR	Human-readable match score description. See Match Score Categories below.

### Match Score Categories

Each record in the output table is assigned one of the following match score categories:

Score	MATCHSCORE_DESC	Color	Description
1	Valid address	GREEN	Address matched directly without corrections.
2	Valid address after correction	GREEN	Address matched after automatic corrections to one or more fields.
3	Incomplete address	ORANGE	Partial match. Postal code + house number exists, but the specific addition could not be resolved.
4	Historical address	RED	Address found but marked as no longer active.
5	No address match	RED	No matching address was found after all matching attempts.

### *Sample Data*

The application includes a built-in sample dataset in the INPUT.SAMPLE\_DATA table. This table contains 10 representative Dutch address records with various data quality issues (inconsistent casing, missing spaces in postal codes, combined street + number values, non-standard city names). It can be used to test the application workflow without granting access to production data.

### 3. Application Workflow (Streamlit UI)

The Streamlit interface provides a guided, step-by-step workflow organized across five tabs. Each tab corresponds to a stage in the cleaning process.

#### Tab 1: Select Input Table

In this tab, you select the table containing the address data to be cleaned. The application uses the Snowflake reference system to request SELECT access. When you select a table, Snowflake will display a permission pop-up asking you to grant the application read access.

#### Tab 2: Map Columns

For each column in the selected input table, a dropdown menu is displayed allowing you to map it to one of the five address fields (Street, House number, House number addition, Postal code, City) or leave it as "Not applicable".

The application validates the mapping before proceeding:

- Each address field may only be mapped to a single column.
- At least one valid combination of fields must be mapped (see Input Data Fields).
- No reserved column names may be present in the input table.

Once the mapping is confirmed, the application creates an internal mapped table and proceeds to the verification step.

### Map Columns

The screenshot displays the 'Map Columns' interface. At the top, there is a 'Need help?' button. Below it, four columns are listed, each with a dropdown menu for mapping:

- ID\_CONSUMER: (empty dropdown)
- STREETNAME\_CONSUMER: Street
- HOUSENUMBER\_CONSUMER: Houenumber
- HOUSENUMBERADDITION\_CONSUMER: Houenumberaddition

The dropdown for 'HOUSENUMBERADDITION\_CONSUMER' is open, showing a list of options: Street, Houenumber, Houenumberaddition, Postalcode, and City. At the bottom of the interface is a 'Confirm mapping' button.

### Tab 3: Verify Mapping

This tab displays a preview of the mapped data. The columns prefixed with "IN" (e.g., INSTREET, INHOUSENUMBER) show the values that will be used as input for the cleaning process. Verify that these columns contain the expected data before proceeding. This tab also provides a collapsible section containing the equivalent SQL script for the current mapping. This script can be copied and used directly in a data pipeline (see SQL Procedure Workflow). Select "Start Cleaning Process" to begin. Processing time depends on the number of records in the input table.

### Tab 4: Results

- After the cleaning process is completed, this tab displays a summary of the results:
- **Match Scores:** A breakdown of all five match score categories with record counts and percentages.
  - **Output Table Preview:** A sample of the output table with all original and new columns.
  - **Ready-to-use SQL Queries:** Pre-generated SQL queries for querying the output table directly from a Snowflake worksheet.

The output table is stored in the OUTPUT schema of the application database and remains available until explicitly deleted.

Output table preview

	MATCHSCORE	MATCHSCORE_DESC	ADDRESS_CORRECTION	OUTPUT_STREET
4	GREEN	Valid address		Palmslagruwe
2	GREEN	Valid address after correction	true	Ariënswei
3	RED	Historical address		Mercatorlaan

### Tab 5: Output Management

This tab provides an overview of all previous cleaning processes and their output tables. For each process, the following information is displayed: input table name, output table name, row count, number of address corrections, counts per match score category, and processing date.

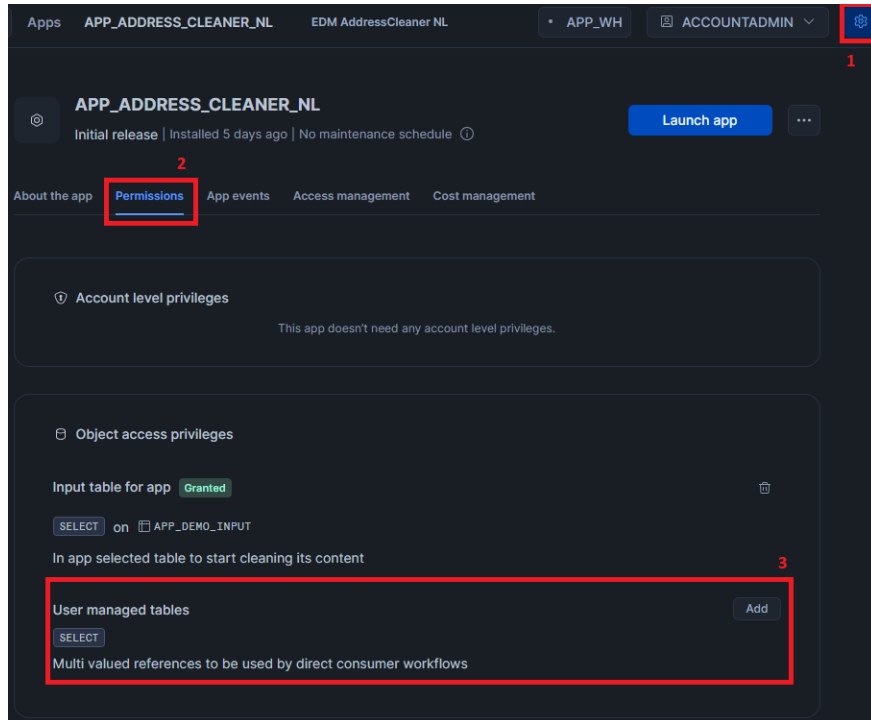
Output tables that are no longer needed can be deleted from this tab. Deleting an output table removes it from the OUTPUT schema and from the process history.

## 4. SQL Procedure Workflow

For automated or scripted workflows, the application can be invoked directly via SQL without using the Streamlit interface. This approach is recommended for integration into data pipelines, scheduled tasks, or any scenario where UI interaction is impractical.

### Step 1: Grant Table Access

Register the input table with the application using the permissions tab in the application. This grants the application SELECT access on the specified table.



### Step 2: Run the Cleaning Procedure

Call the cleaning procedure with the table location and column mapping as parameters. The database and schema parameters are optional. If omitted, the procedure will search all registered references for the given table name. The procedure returns the name of the output table on success, or "FALSE" if the process failed.

```
CALL EDM_CODE.EDM_ADDRESS_CLEANER_NL(  
  '<table_database>',      -- database of input table (optional)  
  '<table_schema>',      -- schema of input table (optional)  
  '<table_name>',        -- name of input table  
  '<street_column>',     -- column containing street name  
  '<houenumber_column>', -- column containing house number  
  '<addition_column>',   -- column containing house number addition  
  '<postalcode_column>', -- column containing postal code  
  '<city_column>',      -- column containing city name  
);
```

### Step 3: Query the Output

The output table is created in the OUTPUT schema with a timestamp-based name, SELECT grant is provided automatically.

```
SELECT * FROM OUTPUT.<tablename>_<DATE>_<TIME>;
```

*Tip: the output of the procedure in step 2 contains the output table name. This can be used directly in your following statements to work with the output.*

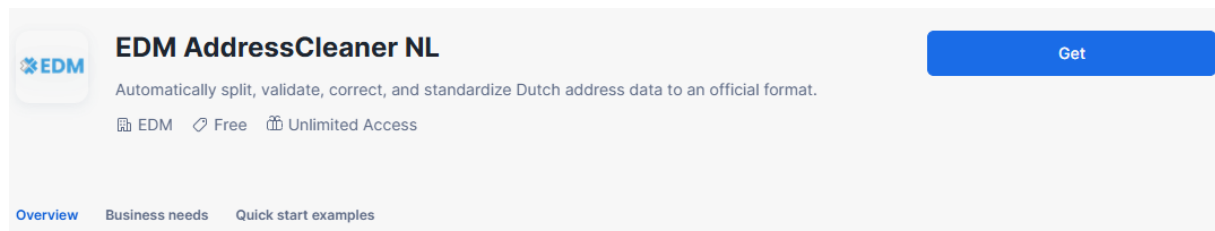
## 5. Snowflake Components

### Application Installation

The application is distributed as a Snowflake Native Application through the Snowflake Marketplace or via direct listing. Installation is handled entirely within Snowflake.

To install the application:

1. Locate the application in the Snowflake Marketplace or through a private listing shared by the provider.
2. Select "Get" to install the application. Snowflake creates the application object in your account.
3. Grant the EDM\_APP\_USER application role to the Snowflake roles that should have access to the application.
4. Open the application from the Snowflake sidebar to access the Streamlit interface.



The screenshot shows the Snowflake Marketplace listing for the application "EDM AddressCleaner NL". On the left is the EDM logo. The title "EDM AddressCleaner NL" is prominently displayed. Below the title is a description: "Automatically split, validate, correct, and standardize Dutch address data to an official format." Underneath the description are three icons: a document icon labeled "EDM", a price tag icon labeled "Free", and a shield icon labeled "Unlimited Access". On the right side of the listing is a blue "Get" button. At the bottom of the listing, there are three tabs: "Overview" (which is selected), "Business needs", and "Quick start examples".

Address data is a critical component of many customer and operational databases. In practice, address fields often contain inconsistent formatting, incomplete information, or data entry errors.

EDM AddressCleaner NL helps organizations improve the quality and consistency of their Dutch address data by automatically structuring and correcting address records.

#### Categories

Accelerating Advertising Revenue

Contact Data Enrichment

### Upgrading

Consumer accounts receive updates through Snowflake's release channel mechanism. Upgrades are scheduled by the provider and automatically executed in your environment.

### Application Logging

The application uses the Snowflake log functionality for structured logging throughout the cleaning process. Each procedure step emits log entries with the following structured metadata:

- **procedure:** Identifier of the current processing step
- **step:** Sub-step identifier within the procedure
- **level:** Log severity: info (process milestones), debug (SQL statement execution), or error (failures).

Logs can be accessed through the Snowflake event table and are visible to the consumer at any time. When log sharing is enabled, the provider will receive a copy of the application logs in your account. Logs created by the application will **NEVER** contain any

information about input data, user data or any other data that is not a direct component of the application.

### *Grants and Access*

#### *Application Roles*

The application defines one application role: EDM\_APP\_USER. This role must be granted to Snowflake roles that need access to the application interface and output data. The role provides:

- USAGE on the EDM\_CODE schema (contains the Streamlit app, procedures, and functions).
- USAGE on the OUTPUT schema and SELECT on the PROCESS\_HISTORY view.
- USAGE on the INPUT schema, including SELECT on SAMPLE\_DATA and CREATE TABLE permissions.
- USAGE on the EDM\_ADDRESS\_CLEANER\_NL procedure.

```
GRANT APPLICATION ROLE EDM_APP_USER TO ROLE "<your_role>";
```

#### *Table Access (Reference System)*

The application uses the Snowflake Native App reference system to access consumer tables. It does not request broad database-level grants. Two reference types are defined in the manifest:

- **app\_table (single-valued):** Used by the Streamlit UI to bind one table at a time through the in-app table selector. Managed via REGISTER\_SINGLE\_REFERENCE.
- **consumer\_tables (multi-valued):** Used for SQL procedure workflows where multiple tables may be registered. Managed via the permissions tab in the application settings.

Both reference types request only SELECT privilege. The application never modifies consumer tables. See section 4 (*SQL Procedure Workflow*) for instructions.

### *Security and Data Handling*

The application adheres to the Snowflake Native Application security model:

- All processing occurs within the consumer's Snowflake account. No data is transferred to external systems or the provider's account.
- The application only requests SELECT access on consumer tables. It never modifies or deletes source data.
- Stored procedures are executed as OWNER, ensuring they run within the application's security context.
- Telemetry event sharing (errors, warnings, debug logs) is optional and requires explicit consumer authorization during installation. Logs will never contain user information other than the account hash for identification (see Snowflake documentation).

- Table access grants can be revoked at any time through the permissions tab in the application interface.

#### *Usage and Costs*

The application runs within the consumer's Snowflake account and uses the consumer's computing resources (virtual warehouse). Processing costs are determined by:

- **Input table size:** Larger datasets require more computing time. The application deduplicates unique address combinations before validation to optimize performance.
- **Warehouse size:** A larger warehouse reduces wall-clock time but increases credit consumption per second. An XSMALL warehouse is recommended to minimize costs.
- **Storage:** Output tables are stored as transient tables in the OUTPUT schema. Storage costs depend on the size of the output and how long the tables are retained. Use the Output Management tab in the application to remove output tables that are no longer needed.

The process history table (OUTPUT.APP\_PROCESS\_HISTORY) tracks each cleaning run including the row count, allowing you to monitor usage patterns over time.