## Data Preprocessing and Normalization Report - Revised

\*\*Project:\*\* Data Preprocessing for SQLTEST Database

\*\*Date:\*\* October 27, 2023

\*\*1. Introduction\*\*

This report documents the revised data preprocessing and normalization process applied to the SQLTEST database, containing 59 tables. The goal was to improve the efficiency of the preprocessing pipeline by leveraging database introspection to partially automate the configuration process. The system now automatically generates a basic configuration for each table, reducing manual effort while maintaining the flexibility to handle diverse table structures.

\*\*2. Methodology\*\*

A Python class, `DataPreprocessor`, was developed to perform the following tasks:

\* \*\*Database Connection and Introspection:\*\* The class establishes a connection to the PostgreSQL database using SQLAlchemy. It then uses SQLAlchemy's `Inspector` object to retrieve schema information (column names and data types) directly from the database. This eliminates the need to manually specify all column names in the configuration.

\* \*\*Automated Configuration Generation:\*\* A new method, `auto\_generate\_config`, automatically generates a basic configuration for each table based on the inferred data types. It uses heuristics to identify likely categorical and timestamp columns: string columns (excluding description-like fields) are considered categorical, and columns with "TIMESTAMP" or "DATE" in their type string are considered timestamps.

\* \*\*Data Retrieval and Preprocessing:\*\* The `preprocess\_table` method retrieves data from individual tables using `pd.read\_sql\_table`. It handles missing values (NULLs) by replacing them with 0 for numeric columns and an empty string ('') for non-numeric columns. Categorical columns are encoded using scikit-learn's `LabelEncoder`, and timestamp columns are converted to dates.

\* \*\*Manual Configuration Override:\*\* The system allows manual overrides of the automatically generated configurations. If a specific configuration is provided for a table, it will be used; otherwise, the automatically generated configuration will be used. This allows for fine-tuning of the preprocessing for individual tables where necessary.

\*\*3. Automated Configuration Generation and Heuristics\*\*

The automated configuration generation relies on heuristics to identify categorical and timestamp columns. These heuristics are:

\* \*\*Categorical Columns:\*\* String columns (excluding columns with names like "description," "notes," or "comments") are assumed to be categorical.

\* \*\*Timestamp Columns:\*\* Columns with "TIMESTAMP" or "DATE" in their data type string are assumed to be timestamps.

These heuristics are not perfect and might require manual adjustments for specific tables. The system prioritizes manually provided configurations over automatically generated ones.

\*\*4. Libraries Used\*\*

\* \*\*psycopg2:\*\* PostgreSQL adapter for Python (used indirectly through SQLAlchemy).

\* \*\*pandas:\*\* Data manipulation and analysis library.

\* \*\*SQLAlchemy:\*\* SQL toolkit and ORM for database interaction and schema introspection.

\* \*\*scikit-learn:\*\* Machine learning library (used for `LabelEncoder`).

\* \*\*python-dateutil:\*\* Library for parsing dates and times.

\*\*5. Conclusion\*\*

The revised `DataPreprocessor` class provides a more efficient and scalable way to preprocess data from the SQLTEST database. The use of database introspection significantly reduces the manual effort required to configure the preprocessing steps. While some manual review and adjustment might still be necessary, the automated configuration generation greatly streamlines the process, making it more adaptable to changes in the database schema and more maintainable for a large number of tables. Further improvements could involve more sophisticated heuristics for data type inference and handling of more complex data types.