Data and Information Quality Report

Daniele Ferracuti 10580840
Samuele Pietro Galli 10710025
Dataset n.13

1.1. Project Goal

The goal of this project is to implement a Data Preparation Pipeline on the assigned dataset (HOTELS) to address and resolve key Data Quality issues. The pipeline is designed to follow these steps:

1. Data Profiling and Quality Assessment: Understanding the dataset structure and identifying potential quality issues.
2. Data Cleaning: Transforming, standardizing, detecting, and correcting errors, including:
    - Data Transformation and Standardization: Bringing all data to a consistent format and correcting typographical errors.
    - Error Detection and Correction: Managing missing values and detecting and addressing potential outliers.
    - Data Deduplication: Identifying and resolving non-exact duplicates.
3. Data Quality Assessment (Post-Cleaning): Verifying the effectiveness of the cleaning process.

The dataset assigned to our group represents structured information about hotel facilities in Milan, including details such as location, capacity, and categorization.

For this project, we mainly used the libraries pandas, numpy, and re. Pandas was essential for working with tabular data, allowing us to manage columns, transform datasets, and filter data efficiently. Numpy was used for numerical operations, ensuring faster computations on larger datasets. Finally, the RE library helped with text normalization and cleaning, which was crucial for handling unstructured data like addresses and descriptions.

Data Quality assessment

2.1. Preliminary assessment
We used head() to preview the first few rows, shape() to check its dimensions, and dtypes() to examine the data types of each column. To ensure data consistency, we applied duplicated() to detect duplicate rows and confirmed there were none. Columns were categorized into numerical and categorical groups using select_dtypes(), facilitating targeted analysis. For numerical columns, we plotted histograms with hist() to explore the cardinality and identify potential patterns or irregularities in the data distribution.
The data assessment began with an evaluation of distinctness, constancy, completeness, and uniqueness. Then we moved to analyze the accuracy ensuring that the data adhered to expected formats and logical constraints.

2.2 Accuracy:
Functions designed to ensure the syntactic correctness of the data by validating specific columns: for example, the check_ubicazione function verifies that addresses follow a consistent pattern, starting with a valid prefix such as "VIA" or "PZA" and including other required details. The check_isdigit function ensures that numeric values, such as house numbers or street codes, are valid integers or properly formatted floating-point numbers within a reasonable range. This validation guarantees that columns like Civico or Codice via contain logical and consistent values. For categorical fields, the check_whitelist function confirms that the values belong to a predefined set of acceptable options, as applied to fields like Tipo via or Tipo attività. check_numeric_list and check_list, validate columns that include lists of values, ensuring they adhere to a structured format, such as semicolon-separated numbers or alphanumeric codes.

2.3 Consistency:
We assessed the consistency of the dataset by verifying whether the relationships between different columns adhered to logical rules. safe_sum_split, was created to handle cases where numerical values were stored as semicolon-separated strings (example: "10;15;20"). This function splits the string, converts the parts to numbers, and computes their sum, even accounting for potential missing or invalid values.
The verifica_corrispondenza function was used to ensure the alignment of address information by reconstructing the full address from its component and comparing it to the recorded Ubicazione; if all these conditions were met, the row was considered consistent
For the rows where a discrepancy was identified, we used the estrai_valori function to extract the missing components directly from the Ubicazione field. This function employs regular expressions to identify and separate the various elements of the address: the street type (e.g., "CSO"), the street name (e.g., "BUENOS AIRES"), the house number, and the zone code (z.d.). The extracted values are stored in a dictionary (valori_df) to be used for data correction. We implemented the update_df_by_location function, to update the main dataset with the corrected values extracted from Ubicazione. This function iterated through the rows of valori_df and checked whether rows with the same Ubicazione value existed in the main dataset. For each match found, the specified columns (e.g., Tipo via, Descrizione via, Civico, ZD) were updated with the new values.
We then standardized the values in key columns. For example, abbreviations in the Tipo via column, such as "CSO" or "VLE," were replaced with their full forms, like "CORSO" and "VIALE," to ensure uniformity. Similarly, the Categoria column, which initially contained numeric values (e.g., "1", "2") or ambiguous labels (e.g., "I"), was transformed into descriptive labels, such as "1 STELLA" or "2 STELLE." Missing values in columns like Tipo attività struture extra and Insegna were filled with a default value, "Non Specificato," to reduce the presence of null values in the dataset. Next, we addressed the missing values in the Piani totali column by using information from related fields. For rows where Piani totali was missing but Piano piano had a value, the total floors were calculated based on the structure of Piano piano. If both Camere piano and Posti letto per piano were consistent, their values were used to infer the number of floors. Lastly, we filled missing values in Camere piano and Posti letto per piano for hotels with only one floor by copying the total number of rooms or beds directly into these columns. We also standardized the format of numerical columns, such as Camere, Posti letto, and Civico, to remove decimals and display values as integers for better readability.

2.4 Data Cleaning (Error Detection & Correction for Missing Values):
We filled the missing values in critical columns. For example, the Codice via column was filled with "not specified" to indicate unavailable values explicitly. For Piani totali, where the total number of floors was missing, we assumed that these hotels had only one floor, filling the column with a value of 1. Similarly, the Piano piano column, which represents the detailed floor structure, was set to "1" for hotels assumed to have a single floor. we used the total number of rooms (Camere) and beds (Posti letto), respectively To address missing data in Camere piano and Posti letto per piano.

2.5 Data Cleaning (Error Detection & Correction for Outliers) phase:
We applied a Z-score method (ZS) to the Posti letto column with a threshold of 3, identifying extreme values that deviated significantly from the mean. The detected outliers included unusually high bed counts, such as 725, 864, and 922 and we enforced a logical constraint on the Construction_year column by ensuring that all values were less than or equal to 2023.