Project Report

Group Members:

Niharika Gurram [16360904]

Rohith Reddy Yarramreddy [12618055]

Approach:

Reading Excel File: The code begins by importing data from an Excel file named relationinput.xlsx using the pandas library. The data is loaded into a DataFrame called dataf.

Functional Dependencies File: A filename FD.txt is specified, which is presumably used to store or retrieve functional dependencies relevant to the relation being processed.

Extracting Column Names and Data: The column names of the DataFrame are extracted and stored in a list called colnames. The actual data from the DataFrame is converted to a list of lists (rows) and stored in the variable data.

User Input for Primary Keys: The code prompts the user to input the primary keys for the relation. The input is expected to be a comma-separated string. This string is split into individual key names, and any leading or trailing whitespace is removed from each key. The resulting list of primary keys is stored in the variable prim_key.

Detecting the Highest Normal Form of a Table:

The code prompts the user to decide if they want to detect the highest normal form of a table by entering 1 for "Yes" or 0 for "No", storing the input as highest_nf_of_ip. If the user inputs 1, the function detection_of_max_nf is called to analyze and print the table's highest normal form; if 0, no action is taken.

Normalization Process:

Option = 1 (1NF)

- The code returns data normalized to **1NF** only.
- Ensures atomicity of values without any repeating groups or nested relations.
- **1NF (First Normal Form)**: Ensures atomicity by breaking down attributes with repeating values into separate rows. If any attributes don’t satisfy 1NF, they are decomposed. Functional dependencies (FDs) are read from a file and applied to the relation.

Option = 2 (2NF)

- Returns data normalized to **1NF and 2NF**.
- Ensures atomicity (1NF) and removes partial dependencies on any candidate key (2NF).
- **2NF (Second Normal Form)**: After achieving 1NF, checks if relations are free of partial dependencies on any candidate key. If partial dependencies exist, the relations are decomposed to satisfy 2NF.

Option = 3 (3NF)

- Returns data normalized to **1NF, 2NF, and 3NF**.
- Ensures atomicity (1NF), no partial dependencies (2NF), and removes transitive dependencies (3NF).
- **3NF (Third Normal Form)**: Ensures each relation is in 3NF by checking for transitive dependencies. If any transitive dependencies are found, relations are further decomposed.

Option = 4 (BCNF)

- Returns data normalized to **1NF, 2NF, 3NF, and BCNF**.
- Ensures atomicity (1NF), no partial dependencies (2NF), no transitive dependencies (3NF), and removes non-superkey determinants (BCNF).
- **BCNF (Boyce-Codd Normal Form)**: Ensures that each relation satisfies BCNF by verifying that all determinants are superkeys. If a non-superkey determinant exists, the relation is decomposed further.

Option = 5 (4NF)

- Returns data normalized to **1NF, 2NF, 3NF, BCNF, and 4NF**.
- Ensures atomicity (1NF), no partial dependencies (2NF), no transitive dependencies (3NF), no non-superkey determinants (BCNF), and removes multi-valued dependencies (4NF).
- **4NF (Fourth Normal Form)**: Ensures that each BCNF-compliant relation also satisfies 4NF by checking for multi-valued dependencies. If any multi-valued dependencies are found, the relation is decomposed to eliminate them.

Option = 6 (5NF)

- Returns data normalized to **1NF, 2NF, 3NF, BCNF, 4NF, and 5NF**.
- Ensures atomicity (1NF), no partial dependencies (2NF), no transitive dependencies (3NF), no non-superkey determinants (BCNF), no multi-valued dependencies (4NF), and eliminates join dependencies (5NF).
- **5NF (Fifth Normal Form)**: Ensures that each 4NF-compliant relation also satisfies 5NF by removing join dependencies. If join dependencies are present, they are resolved by further decomposing the relation.

Assumptions:

Dependency Completeness: All functional and multivalued dependencies are accurately specified in the input files and are non-trivial.

Primary Key: Primary keys are clearly defined and are assumed to be accurate for each relation.

Challenges Faced:

Normalisation of 2NF is a bit difficult as it involves partial dependencies.

Normalisation of 1NF depending on atomicity is a bit difficult

Normalisation of 4NF is difficult as it involves MVD's

In 5NF testing with new data instance is difficult as it involves join dependencies

Performing on different datasets is difficult as it involves alots of constraints

Faced difficulty in automating the whole code.

Validation tests performed:

Performed validations on fd's and mvd's and also on input.

Tested with different datasets