Normalization Project Report

1. Importing Libraries:

required libraries and modules, including 'csv', 'pandas', 'numpy', and 'copy'.

2. Global Variables:

'global_map': A global dictionary to map attributes to the key they belong to.

3. Reading CSV File:

'read_csv_file(file_name)': Reads a CSV file and converts it into a list of dictionaries, where each dictionary represents a row in the table.

4. Parsing Functional Dependencies:

'parse_functional_dependencies()': Takes user input to input functional dependencies and returns a list of these dependencies.

5. Decompose to 1NF:

'decompose_to_1NF(fds, data, multi_value_attributes)': Decomposes the table into the first normal form (1NF), specifically handling multivalued attributes.

6. Find Normal Form:

'find_normal_form(data, fds, key, target_nf_choice, mvds)': Determines the normal form of the table and performs normalization steps accordingly. It checks for 1NF, 2NF, 3NF, BCNF, and 4NF based on the user's choice and functional dependencies.

7. Check for 1NF:

'is_in_1NF(data)': Checks if the table is in the first normal form by identifying multivalued attributes.

8. Check for 2NF:

'is_in_2NF(fds, key)': Checks if the table is in the second normal form by identifying violated functional dependencies.

9. Decompose to 2NF:

'decompose_to_2NF(data, primary_key, violated_fds, correct_fds, fds)': Decomposes the table into the second normal form, particularly handling violated functional dependencies.

10. Populate Global Map:

'populateGlobalMap(fds, primary_key_set, data, violated_fds, correct_fds)': Populates the global map to map attributes to their respective primary keys.

11. Convert Data to DataFrame:

'convertDataToDataFrame(data)': Converts the data to a Pandas DataFrame for easier manipulation.

12. Check for 3NF:

'is_in_3NF(table_fds, tables_of_2nf_dfs, key)': Checks if the table is in the third normal form by verifying transitive dependencies.

13. Decompose to 3NF:

'decompose_to_3NF(transitive_dependencies, temp_table_fds, tables_of_2nf_dfs, faulty_dependency_indexes)': Decomposes the table into the third normal form, addressing transitive dependencies.

14. Check for BCNF:

'is_in_BCNF(table_fds, tables_of_2nf_dfs, candidate_key)': Checks if the table is in BoyceCodd Normal Form (BCNF).

15. Decompose to BCNF:

'decompose_to_BCNF(problematic_fds, table_fds, tables_of_2nf_dfs, key)': Decomposes the table into BCNF, handling problematic functional dependencies.

16. Check for 4NF:

'is_in_4NF(table_fds, tables_of_2nf_dfs, mvds, key)': Checks if the table is in the fourth normal form (4NF) and deals with multivalued dependencies (MVDs).

17. Decompose to 4NF:

'decompose_4nf(data, fds, mvds)': Implements a basic 4NF decomposition using given MVDs.

18. Decompose to 5NF:

'decompose_to_5NF(data, fds)': Identifies join dependencies and relevant attributes, then performs a losslessjoin decomposition based on the identified join dependencies.

19. Identify Join Dependencies:

'identify_join_dependencies(fds)': Identifies join dependencies and relevant attributes to assist in the losslessjoin decomposition.

20. Lossless Join Decomposition:

'lossless_join_decomposition(data, join_dependencies, relevant_attributes)': Performs a losslessjoin decomposition based on identified join dependencies.

21. Choose Attributes:

'choose_attributes(uncovered_attributes, relevant_attributes)': Helps select attributes for creating new decomposed tables in the losslessjoin decomposition.

22. Find MultiValued Dependencies:

'find_multivalued_dependencies()': Takes user input to input multivalued dependencies (MVDs) and returns a list of these dependencies.

23. Main Function:

'main()': This is the main function that orchestrates the normalization process. It reads data, parses functional dependencies, determines the user's choice for the highest normal form, and normalizes the table accordingly.