CS5300 PROGRAMMING PROJECT01 DATABASE NORMALIZATION

TEAM: Sai Dinesh Reddy Gunnala (12597913), Anirudh Koganti (12612256), Asrithasai Gannamani (12595903), Sathwika Yasa (12615417), Urmilla Chowdary Nalluru (12601280)

Sample Input:

Dataset: The example data includes the following attributes: Course, Professor, ProfessorEmail, CourseStart, CourseEnd, FirstName, LastName, and StudenetID. It has five tuples in it. The dataset is stored as a comma-separated values (.csv) file.

Functional Dependencies: The dataset's functional dependencies are as follows:

StudentId->FirstName,LastName.

Course->CourseStart, CourseEnd, Professor.

Professor->ProfessorEmail

Multi-valued dependencies:

The multi-valued dependencies used for the dataset are:

Course -> CourseStart

Course -> CourseEnd

Functional dependencies and multi-valued dependencies are taken as .txt file (input.txt).

User Input:

1. Select the highest normal form to normalize from the below list

   1 for 1NF

   2 for 2NF

   3 for 3NF

   B for BCNF

   4 for 4NF

   5 for 5NF:
2. Primary key used (studentID, Course)

Core Components:

1. **Input Parser:** The txt file, which is used to contain the functional dependencies, and the csv file, which is used to contain the table, are read by the parser function. It breaks down the text into recognized strings of characters for further analysis.
2. **Normalizer:** The input dataset is broken down by the normalizer into the necessary normal form using the specified functional dependencies. The provided dataset is broken down into the user-required normal form using normalizing techniques.
3. **SQL Query Generator:** It refers to a program or feature that helps in creating SQL queries.

Deliverables:

Source Code: we used Python programming language to normalize the given dataset

Code Description:

1. To carry out the necessary tasks, the Python libraries are imported.
2. The input dataset is given as .csv file. The read method is used to read the data from the csv file.
3. The open method is used to read the functional dependencies txt file.
4. Getting the input of the user required normal form and the primary keys.
5. The parser method is used to convert the given input files into string characters.
6. Checking for multi-valued dependencies
7. is_superkey is defined to check for the superkey.
8. The bcnf_decomposition function is defined to decompose the relation into BCNF normal form.
9. The is_in_1NF method is defined to check whether the dataset is in 1NF or not.
10. The is_in_2NF method is defined to check whether the dataset is in 2NF or not.
11. The is_in_3NF method is defined to check whether the dataset is in 3NF or not.
12. The is_in_bcnf method is defined to check whether the dataset is in BCNF or not.
13. is_in_4NF method is defined to check whether the dataset is in 4NF or not.
14. The in_5NF method is defined to check whether the dataset is in 5NF or not.
15. The first_normaization_form function is defined to decompose the given dataset into 1NF.
16. second_normalization_form function is defined to decompose the given dataset into the 2NF.
17. third_normalization_form function is defined to decompose the given dataset into the 3NF.
18. bc_normal_form function is defined to decompose the given dataset into the BCNF.
19. fourth_normalization_form function is defined to decompose the given dataset into 4NF.
20. The decomposing_to_5NF function is defined to check for the lossless decomposition of 5NF.
21. fifth_normalization_form function is defined to decompose the given dataset into 5NF.
22. output function is defined to generate the user required output results.
23. The sql_query_generator_for_1NF function is defined to print the 1NF sql query as an output.

24. The sql_query_generator_for_2_3 function is defined to print the 2NF and 3NF sql queries as output.
25. The sql_query_generator_for_BCNF_4_5 function is defined to print the BCNF, 4NF, and 5NF sql queries as output.
26. Checking for the normal form of the given dataset, whether it is in any type of normal form or not.
27. Using the functions is_in_1nf, is_in_2nf, is_in_3nf, is_in_4nf, is_in_bcnf, and is_in_5nf from the variable high_normalform, one may determine the highest normal form of the provided dataset.

Code Execution:

1. Choose the desired normal form for normalizing the table:

   Type '1' for 1NF

   Type '2' for 2NF

   Type '3' for 3NF

   Type 'B' for BCNF

   Type '4' for 4NF

   Type '5' for 5NF

2. Next, choose an option:

   Type '1' to determine the highest normal form of the given relation

   Type '2' if you don't need to find the highest normal form

3. Provide the primary key values for this relationship, separated by commas. For example, the primary key could be "StudentID, Course."
4. If the given table is not in the selected normal form, it will be decomposed until it meets the criteria, and you will receive queries for the decomposed tables.
5. If the given table already satisfies the chosen normal form, you will receive a query for the original table.
6. For 5NF, you will receive candidate keys for each table. For instance, the student table's candidate key is "StudentID," the course table's candidate key is "Course," and the professor table's candidate key is "Professor."
7. Finally, the script will display the highest normal form achieved by the given table. In this case, the table currently satisfies only 1NF.