**Code Description:**

The provided code is designed to normalize a given dataset to a specified Normal Form (1NF, 2NF, 3NF, BCNF, 4NF, or 5NF) for a relational database. It follows a series of steps to achieve normalization based on user-defined functional dependencies and multivalued dependencies.

**Detailed Documentation:**

**Import Libraries:**

Imports the csv library for handling CSV files.

Global Variables and Data Structures:

Defines various data structures to hold column names, data, functional dependencies, multivalued dependencies, etc.

These structures are used to hold and manage information during the normalization process.

**Input Parsing (Function: input\_parser()):**

Reads a CSV file, extracts column names and data.

Takes user input to define functional and multivalued dependencies.

Requests user input for the desired normal form and the composite key.

**Normalization Functions:**

Checks for Normal Forms i.e.,

Functions (is\_1nf(), is\_2nf(), is\_3nf(), is\_bcnf(), is\_4nf(), is\_5nf()) validate if the given relations satisfy the conditions of the specified normal forms.

These functions examine functional dependencies and the composite key to check for partial dependencies, transitive dependencies, BCNF anomalies, and more.

**Dependency Handling:**

Functions (resolve\_partial\_dependencies(), resolve\_transitive\_dependencies(), resolve\_bcnf\_anomalies()) modify the table structure to resolve partial, transitive, and BCNF anomalies based on the normalization checks.

**SQL Query Generation:**

Function (sqlQueries()) generates SQL queries for creating normalized tables.

Dependency Builder (Function: dependency\_builder()):

Builds a map of dependencies (how many times each attribute appears in the dependency).

**Main Execution:**

Calls input\_parser() to gather user inputs and initiate the normalization process based on the chosen normal form.

Performs checks for the desired normal form and modifies the table structure accordingly.

Generates SQL queries for the normalized tables.

**Dataset Relationships:**

**Functional Dependencies:**

StudentID -> FirstName, LastName

Course, Professor -> classroom

Course -> CourseStart, CourseEnd

Professor -> ProfessorEmail

**Multivalued Dependencies:**

Course ->> Professor

Course ->> classroom

StudentID ->> Course

StudentID ->> Professor

**Flow:**

**Input Gathering:**

Reads the dataset and collects information on functional and multivalued dependencies, the composite key, and the desired normal form from the user.

**Normalization Steps:**

Based on the chosen normal form, the code checks the dataset for compliance with that form.

If the dataset does not satisfy the conditions of the chosen normal form, the code proceeds to resolve issues such as partial dependencies, transitive dependencies, and BCNF anomalies.

**SQL Query Generation:**

After normalization, the code generates SQL queries to create the normalized tables based on the changes made during the normalization process.

**Methodology:**

The code starts by obtaining necessary inputs and then systematically checks the dataset against the chosen normal form's conditions. If the dataset doesn't meet these criteria, it proceeds to modify the data structure by resolving partial dependencies, transitive dependencies, and BCNF anomalies. Finally, it generates SQL queries to create the normalized tables based on the alterations made during normalization.

Overall, the code aims to provide a framework for normalizing a dataset to the desired normal form, leveraging user-defined dependencies to guide the normalization process.