This code is designed for database normalization and transformation based on user-provided functional dependencies and multivalued dependencies.

**Main Script:** `main()`

The `main()` function serves as the entry point of the script. It prompts the user for input, including the file path, functional dependencies, multivalued dependencies, composite keys, and the desired normalization form. Based on the user's choices, it performs database normalization and displays the SQL commands for creating normalized tables.

Input:

- `file\_name`: The path to a CSV file containing the database.

- `functional\_dependencies`: Functional dependencies provided by the user.

- `multivalued\_dependencies`: Multivalued dependencies provided by the user.

- `composite\_keys`: Composite keys specified by the user.

- `normalization\_form`: The desired normalization form (1NF, 2NF, 3NF, BCNF, 4NF, or 5NF) chosen by the user.

Output:

- The script displays information about whether the input data is already in the chosen normalization form or not.

- It generates and prints SQL commands for creating normalized tables based on the chosen normalization form.

Import Statements:

- `import csv`: This module allows reading CSV files.

- `from datetime import datetime`: Used for datetime conversion in determining column data types.

**Database Structure Analysis Functions:**

1. `get\_columns(csv\_file\_path)` Function:

- Purpose: This function reads the CSV file, extracts the header and sample data, and determines the data types for each column.

- Input: The file path to the CSV file.

- Output: A dictionary mapping column names to their determined data types.

2. `is\_atomic(value)` Function:

- Purpose: This function checks if a value is atomic (e.g., not a list, set, tuple, or dictionary).

- Input: The value to be checked.

- Output: Returns `True` if the value is atomic, `False` otherwise.

3. `is\_in\_1NF(file\_path)` Function:

- Purpose: Determines whether the input database is in the First Normal Form (1NF).

- Input: The file path to the CSV file.

- Output: Returns `True` if the database is in 1NF, `False` otherwise.

4. `is\_in\_2NF(columns, functional\_dependencies, composite\_key)` Function:

- Purpose: Checks if the database is in the Second Normal Form (2NF).

- Input: Columns, functional dependencies, and composite keys.

- Output: Returns `True` if the database is in 2NF, `False` otherwise.

5. `is\_in\_3NF(tables, functional\_dependencies)` Function:

- Purpose: Determines if the database is in the Third Normal Form (3NF).

- Input: Tables and functional dependencies.

- Output: Returns `True` if the database is in 3NF, `False` otherwise.

6. `is\_in\_bcnf(tables, functional\_dependencies)` Function:

- Purpose: Checks if the database is in Boyce-Codd Normal Form (BCNF).

- Input: Tables and functional dependencies.

- Output: Returns `True` if the database is in BCNF, `False` otherwise.

7. `is\_superkey(X, candidate\_key, functional\_dependencies)` Function:

- Purpose: Determines if a set of attributes `X` is a superkey based on a candidate key and functional dependencies.

- Input: `X`, candidate key, and functional dependencies.

- Output: Returns `True` if `X` is a superkey, `False` otherwise.

**Normalization Functions:**

8. `convert\_to\_1nf()` Function:

- Purpose: Converts the database into the First Normal Form (1NF).

- Output: Returns a message indicating that there's no need to convert since the database is already in 1NF. (To be extended for actual conversion.)

9. `convert\_to\_2nf(columns, functional\_dependencies, composite\_key)` Function:

- Purpose: Converts the database into the Second Normal Form (2NF).

- Input: Columns, functional dependencies, and composite keys.

- Output: Returns a dictionary representing normalized tables in 2NF.

10. `convert\_to\_3nf(tables\_2NF, functional\_dependencies)` Function:

- Purpose: Converts the database into the Third Normal Form (3NF).

- Input: Tables in 2NF and functional dependencies.

- Output: Returns a dictionary representing normalized tables in 3NF.

11. `convert\_to\_bcnf(tables\_3NF, functional\_dependencies)` Function:

- Purpose: Converts the database into Boyce-Codd Normal Form (BCNF).

- Input: Tables in 3NF and functional dependencies.

- Output: Returns a dictionary representing normalized tables in BCNF.

12. `convert\_to\_4nf(bcnf\_tables, multivalued\_dependencies)` Function:

- Purpose: Converts the database into the Fourth Normal Form (4NF).

- Input: Tables in BCNF and multivalued dependencies.

- Output: Returns a dictionary representing normalized tables in 4NF.

**User Input Functions:**

13. `get\_functional\_dependencies\_from\_user()` Function:

- Purpose: Allows the user to input functional dependencies and parses them into a dictionary.

- Output: Returns a dictionary of functional dependencies.

14. `get\_multivalued\_dependencies\_from\_user()` Function:

- Purpose: Allows the user to input multivalued functional dependencies and parses them into a dictionary.

- Output: Returns a dictionary of multivalued dependencies.

15. `get\_composite\_keys()` Function:

- Purpose: Allows the user to input a list of composite keys and returns the list.

- Output: Returns a list of composite keys.

**SQL Command Generation and Printing Functions:**

16. `generate\_table\_command(table\_name, columns, primary\_key=None, foreign\_keys=None)` Function:

- Purpose: Generates an SQL command for creating a table with specified attributes, primary key, and foreign keys.

- Input: Table name, columns, primary key, and foreign keys.

- Output: Returns an SQL `CREATE TABLE` command as a string.

17. `print\_1nf\_commands(table\_name, columns)` Function:

- Purpose: Prints SQL commands to create a 1NF table based on the provided table name and columns.

- Input: Table name and columns.

- Output: Prints the SQL command to the console.

18. `print\_nf\_commands(output\_2nf)` Function:

- Purpose: Prints SQL commands to create tables in a normalized form (2NF in this case) based on the provided tables and details.

- Input: A dictionary of tables and their details.

- Output: Prints the SQL commands for each table to the console.

User Input Functions:

- `get\_filename()`: Provides a file path.

- `get\_multivalued\_dependencies\_from\_user()`: Allows the user to input multivalued functional dependencies.