Project Report: MyDB - A Simple Database Management System

1. Overview:

The MyDB project is a simple database management system implemented in Python, providing a command-line interface (CLI) for users to interact with basic database operations. The system includes three main classes: DatabaseFunctions, QueryLanguageProcessing, and MyDbCLI, each serving a specific purpose in the functionality of the database.

2. DatabaseFunctions Class:

2.1. LoadCsvFile Method:

Purpose: Loads data from a CSV file in chunks, converting values to appropriate types.

Parameters:

fileName: The name of the CSV file.

chunkSize: Optional parameter for the number of rows to load at a time.

2.2. createTable Method:

Purpose: Creates a new CSV file (table) with specified columns.

Parameters:

table_name: Name of the table to be created.

columns: List of column names.

2.3. insertInto Method:

Purpose: Inserts values into a specified table.

Parameters:

table_name: Name of the table to insert into.

columns: List of column names.

values: List of values to insert.

2.4. get Method:

Purpose: Retrieves data from a table based on specified columns and conditions.

Parameters:

table_name: Name of the table to query.

columns: List of column names to retrieve.

condition: A condition to filter rows.

2.5. displayTable Method:

Purpose: Displays the first 10 rows of a table.

Parameters:

table_name: Name of the table to display.

2.6. joinTable Method:

Purpose: Joins two tables based on common columns.

Parameters:

table1_name: Name of the first table.

table2_name: Name of the second table.

commonColumns: List of common columns for the join.

2.7. aggregate Method:

Purpose: Performs aggregate functions (SUM, AVG, COUNT, MAX, MIN) on a specific column.

Parameters:

tablename: Name of the table.

column: Name of the column for aggregation.
aggFunc: Aggregate function to perform.
group_by: Optional list of columns to group by.
2.8. update Method:
Purpose: Updates values in a table based on a condition.
Parameters:
tableName: Name of the table to update.
columns: List of columns to update.
values: List of values to set.
condition: Condition for updating rows.
2.9. deleteRow Method:
Purpose: Deletes rows from a table based on a condition.
Parameters:
tableName: Name of the table.
condition: Condition for deleting rows.
3. QueryLanguageProcessing Class:
3.1. getCondition Method:
Purpose: Translates natural language conditions into symbols.
Parameters:
strCondition: Natural language condition.
3.2. processStatement Method:
Purpose: Processes user input statements and invokes the appropriate database function.
Parameters:
statement: User input statement.
chunkSize: Optional parameter for the number of rows to process at a time.
4. MyDbCLI Class:
4.1. Command Methods:
Methods like do_GET, do_COMBINE, do_FIND, etc., are specific to handling user commands and delegate the processing to the QueryLanguageProcessing class.
4.2. do_help Method:
Purpose: Displays help information for available commands.
4.3. do_exit Method:
Purpose: Exits the MyDB CLI.
4.4. default Method:
Purpose: Handles commands not recognized by specific do_ methods by delegating to the QueryLanguageProcessing class.
5. Key Features:
Natural Language Processing: The project incorporates a basic natural language processing component for translating user-friendly statements into database operations, enhancing the user experience.

Database Operations: Essential database operations are covered, including creating tables, inserting and querying data, joining tables, aggregating data, updating records, and deleting records.

Error Handling: The code includes basic error handling to address potential issues such as file not found errors, unsupported aggregate functions, and general exceptions.

6. Supported Commands:
GET, CREATE, COMBINE, FIND, ADD, UPDATE, DELETE, SHOW, and EXIT are the supported commands, providing users with a diverse set of functionalities for interacting with the database.
7. Usage:
Users can run the MyDbCLI class to enter the command-line interface.
They can then issue commands such as GET, CREATE, COMBINE, FIND, ADD, UPDATE, DELETE, SHOW, and EXIT.
8. Future Improvements:
Security Measures: Implement security measures to prevent SQL injection attacks.
Transaction Management: Introduce transaction management for better data consistency.
Indexing: Implement indexing mechanisms to improve query performance.
GUI: Develop a graphical user interface (GUI) for a more user-friendly experience.
9. Conclusion:
The MyDB project provides a foundation for a simple yet effective database management system with natural language processing features. Users can perform operations using a command-line interface, making it accessible to a broader audience. The inclusion of error handling and potential future improvements will contribute to making MyDB a more robust and versatile database management system.