CSCI 5408 DATA MANAGEMENT AND WAREHOUSING

Group:

DBMS_Builder- 6

Sprint 1 Report

Group Members:

Name	Enrollment Number
Vedant Patel	B00984592
Jems Patel	B00984406
Rushil Borad	B00977837

Group Project Git Lab Link for code:
https://git.cs.dal.ca/jems/csci_5408_s24_group06

Contents

Background Research

The TinyDB project aims to create a simplified database management system from scratch, focusing on core database functionalities including query processing, transaction management, and data storage. The primary objective is to develop a system that efficiently handles SQL operations while adhering to the principles of data structure and database design, basically mimicking the standard SQL databases and their operations. This involves a thorough understanding of various linear data structures, file management techniques, and database architecture.

For this project, we researched various data structures and their suitability for different database operations. Arrays and ArrayLists were chosen for their simplicity and efficiency in handling sequential data and dynamic list operations, respectively. Arrays provide fixed-size data handling, which is optimal for processing SQL query parts, while ArrayLists offer dynamic resizing, making them ideal for managing tables and records that can grow or shrink over time.

Another reason for selecting the Arrays and ArraysList over the other data structure like linked list is because of the simple indexing of the data structure. While writing the query we need to validate, and we thought that for that we need to break down the whole query and then need to validate so that can be done easily with the help of the Array and ArrayList.

In terms of persistent storage, we explored different file formats and settled on a custom text-based format due to its flexibility and ease of implementation. This format ensures that each database is represented by a directory containing text files for each table. The schema and data within these files are organized in a readable format, which simplifies the process of data retrieval and manipulation. The choice of using a text-based format over more complex formats like JSON or XML aligns with the project requirements of avoiding common serialization formats and maintaining straightforward data management.

This combination of using Arrays and ArrayLists for in-memory operations, along with a custom text-based file structure for persistent storage, forms the backbone of TinyDB. This approach ensures that the database system is not only efficient in terms of processing but also maintains a clear and organized structure for long-term data storage and retrieval. The research and decisions made in selecting these components reflect a balance between simplicity, efficiency, and the specific requirements of building a minimal yet functional database management system.

Architecture Diagram

Class Diagram:

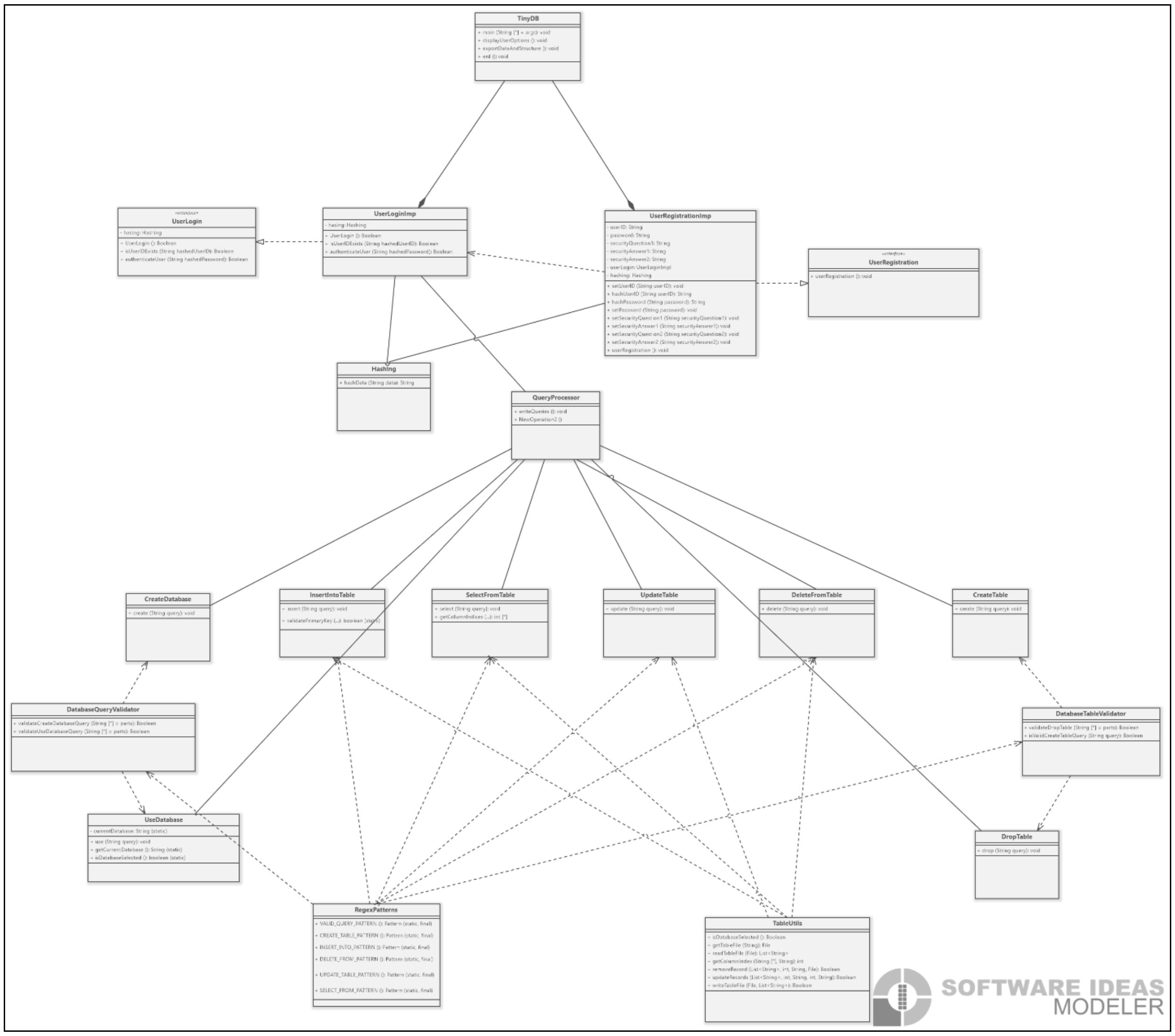

TinyDB
+ main (String [*] = args): void
+ displayUserOptions (): void
+ exportDataAndStructure (): void
+ erd (): void
«interface»
UserLogin
- hasing: Hashing
- UserLogin (): Boolean
- isUserIDExists (String hashedUserID): Boolean
- authenticateUser (String hashedPassword): Boolean
UserLoginImp
- hasing: Hashing
+ UserLogin (): Boolean
+ isUserIDExists (String hashedUserID): Boolean
+ authenticateUser (String hashedPassword): Boolean
UserRegistrationImp
- userID: String
- password: String
- securityQuestion1: String
- securityAnswer1: String
- securityAnswer2: String
- userLogin: UserLoginImpl
- hashing: Hashing
+ setUserID (String userID): void
+ hashUserID (String userID): String
+ hashPassword (String password): String
+ setPassword (String password): void
+ setSecurityQuestion1 (String securityQuestion1): void
+ setSecurityAnswer1 (String securityAnswer1): void
+ setSecurityQuestion2 (String securityQuestion2): void
+ setSecurityAnswer2 (String securityAnswer2): void
+ userRegistration (): void
«interface»
UserRegistration
+ userRegistration (): void
Hashing
+ hashData (String data): String
QueryProcessor
+ writeQueries (): void
+ NewOperation2 ()
CreateDatabase
+ create (String query): void
InsertIntoTable
+ insert (String query): void
+ validatePrimaryKey (...): boolean (static)
SelectFromTable
+ select (String query): void
+ getColumnIndices (...): int [*]
UpdateTable
+ update (String query): void
DeleteFromTable
+ delete (String query): void
CreateTable
+ create (String query): void
DatabaseQueryValidator
+ validateCreateDatabaseQuery (String [*] = parts): Boolean
+ validateUseDatabaseQuery (String [*] = parts): Boolean
DatabaseTableValidator
+ validateDropTable (String [*] = parts): Boolean
+ isValidCreateTableQuery (String query): Boolean
UseDatabase
- currentDatabase: String (static)
+ use (String query): void
+ getCurrentDatabase (): String (static)
+ isDatabaseSelected (): boolean (static)
DropTable
+ drop (String query): void
RegexPatterns
+ VALID_QUERY_PATTERN (): Pattern (static, final)
+ CREATE_TABLE_PATTERN (): Pattern (static, final)
+ INSERT_INTO_PATTERN (): Pattern (static, final)
+ DELETE_FROM_PATTERN (): Pattern (static, final)
+ UPDATE_TABLE_PATTERN (): Pattern (static, final)
+ SELECT_FROM_PATTERN (): Pattern (static, final)
TableUtils
- isDatabaseSelected (): Boolean
- getTableFile (String): File
- readTableFile (File): List<String>
- getColumnIndex (String [*], String): int
- removeRecord (List<String>, int, String, File): Boolean
- updateRecords (List<String>, int, String, int, String): Boolean
- writeTableFile (File, List<String>): Boolean
SOFTWARE IDEAS MODELER

Figure 1 : Architecture Diagram (Class Diagram) of overall program

Pseudocodes

Pseudocode for TinyDB Main Application

```
1. Initialize Components:
    o Initialize user login and registration components.
    o Set up a scanner for user input.
2. Main Loop:
    o Display options for login or registration.
    o Get user input.
    o Validate input to ensure it is a number.
    o Based on user choice:
        ▪ Login:
            ▪ Attempt user login.
            ▪ If successful, display user options and exit the main loop.
        ▪ Register:
            ▪ Proceed with user registration.
        ▪ If input is invalid, show an error message.
3. Display User Options:
    o Display options for writing queries, exporting data and structure, generating ERD, or
      exiting.
    o Get user input.
    o Validate input to ensure it is a number.
    o Based on user choice:
        ▪ Write Queries:
            ▪ Call the function to write queries.
        ▪ Export Data and Structure:
            ▪ Call the function to export data and structure.
        ▪ Generate ERD:
            ▪ Call the function to generate ERD.
        ▪ Exit:
            ▪ Exit the application.
        ▪ If input is invalid, show an error message.
4. Export Data and Structure:
    o Display message indicating the start of the export process.
    o Implement the export logic.
5. Generate ERD:
    o Display message indicating the start of ERD generation.
    o Implement the ERD generation logic.
```

Pseudocode for User Registration Process

```
1. Get User ID Input:
    o Repeat until a unique ID is entered:
        ▪ If the entered ID already exists:
            ▪ Show an error message.
        ▪ Else:
            ▪ Hash the valid ID.
            ▪ Store the hashed ID.
2. Get Password Input:
    o Hash the entered password.
    o Store the hashed password.
3. Get First Security Question Input:
    o Store the first security question.
4. Get First Security Answer Input:
    o Store the first security answer.
5. Get Second Security Question Input:
    o Store the second security question.
6. Get Second Security Answer Input:
    o Store the second security answer.
7. Write User Data to File:
    o Open "User_Profile.txt" file for writing.
    o Write the hashed ID, hashed password, and security questions and answers
      to the file.
    o Close the file.
8. Show Success Message:
    o Display a message indicating successful registration.
```

Pseudocode for Hashing Class

```
1. Initialize Hashing Utility:
    o Set up the hashing utility with the SHA-256 algorithm.
2. Hash Data:
    o Input: Data to be hashed.
    o Process:
        ▪ Convert the data into bytes.
        ▪ Apply the SHA-256 algorithm to the byte data.
        ▪ Convert the hashed byte data into a hexadecimal string.
    o Output: Return the hexadecimal string representing the hashed data.
```

Pseudocode for User Login Process

1. **Initialize Hashing Utility:**
    - Set up the hashing utility for use in the class.
2. **User Login Process:**
    - Get user ID input.
    - Repeat until a valid user ID is entered:
        - Hash the entered user ID.
        - Check if the hashed user ID exists.
        - If it doesn't exist, show an error message.
    - Get password input.
    - Repeat until authentication is successful:
        - Hash the entered password.
        - Check if the hashed password is correct.
        - If the password is incorrect, show an error message.
        - If the password is correct, prompt the user to answer a randomly selected security question.
        - If the security answer is correct, show a success message and complete the login process.
        - If the security answer is incorrect, show an error message.
3. **Check if User ID Exists:**
    - Open the user profile file.
    - Read each line from the file.
    - For each user entry, check if the hashed user ID matches.
    - If a match is found, return true.
    - If no match is found after checking all entries, return false.
4. **Authenticate User:**
    - Open the user profile file.
    - Read each line from the file.
    - For each user’s entry, check if the hashed password matches.
    - If a match is found:
        - Randomly select a security question.
        - Prompt the user to answer the selected security question.
        - Check if the security answer is correct.
        - If the answer is correct, return true.
        - If the answer is incorrect, return false.
    - If no match is found after checking all entries, return false.

Pseudocodes for Database Operations in TinyDB

For Create Database Query

1. **Parse the Query:**
   - Split the input query into parts.
2. **Validate the Query:**
   - Check if the query is a valid "CREATE DATABASE" query.
   - If the query is invalid, show an error message and exit.
3. **Extract Database Name:**
   - Extract the database name from the query parts.
4. **Check if Database Exists:**
   - Check if a directory with the database name already exists.
   - If it exists, show an error message indicating the database already exists.
5. **Create Database Directory:**
   - Attempt to create a new directory for the database.
   - If the directory is created successfully, show a success message.
   - If the directory creation fails, show an error message indicating the failure.

For Use Database Query

1. **Initialize Database Name:**
   - Split the query string to extract the database name.
2. **Validate Query:**
   - Validate if the query format is correct using validateUseDatabaseQuery.
3. **Check Database Existence:**
   - Create a file object pointing to the directory of the specified database.
   - If the directory exists and is valid:
     - Set the current database.
     - Display a success message.
   - If the directory does not exist:
     - Display an error message.
4. **Handle Invalid Query:**
   - If the query format is invalid, display an error message.
5. **Get Current Database:**
   - Return the name of the currently selected database.
6. **Check if Database is Selected:**
   - Return whether a database has been selected or not.

Pseudocodes for Table Operations in TinyDB

Create Table

1. **Check Database Selection:**
   - Ensure a database is currently selected. If not, terminate the operation.
2. **Validate Query:**
   - Validate if the query format is correct using isValidCreateTableQuery.
3. **Match Query Pattern:**
   - Use regex to extract the table name and column definitions from the query.
4. **Check Table Existence:**
   - Create a file object pointing to the table file.
   - If the table already exists, display an error message.
5. **Format Columns:**
   - Split the columns definition.
   - Format each column, identifying primary keys and adding necessary markers.
6. **Write to File:**
   - Create and write the formatted columns to the table file.
   - Display a success message.
   - If an error occurs during file operations, display an error message.
7. **Handle Invalid Query:**
   - If the query format is invalid, display an error message.

Drop Table

1. **Check Database Selection:**
   - Ensure a database is currently selected. If not, terminate the operation.
2. **Validate Query:**
   - Split the query and validate its format using validateDropTable.
3. **Check Table Existence:**
   - Create a file object pointing to the table file.
   - If the table does not exist, display an error message.
4. **Delete Table:**
   - Attempt to delete the table file.
   - If successful, display a success message.
   - If deletion fails, display an error message.
5. **Handle Invalid Query:**
   - If the query format is invalid, display an error message.

Pseudocodes for Data operations in TinyDB

Insert Into Table

1. **Check Database Selection:**
    - Ensure a database is currently selected. If not, terminate the operation.
2. **Validate Query:**
    - Use regex to match and validate the INSERT INTO query format.
3. **Extract Table and Values:**
    - Extract the table name and values from the query.
4. **Check Table Existence:**
    - Obtain the table file using TableUtils.getTableFile.
    - If the table does not exist, terminate the operation.
5. **Read Table Schema:**
    - Read the first line of the table file to get the column definitions.
    - Check if the table file is empty or null.
6. **Validate Value Count:**
    - Split the column definitions and the values.
    - Ensure the count of values matches the count of columns.
    - If the counts do not match, display an error message and terminate the operation.
7. **Validate Primary Key:**
    - Identify the primary key column in the column definitions.
    - Ensure the primary key value is not null.
    - Check for duplicate primary key values in the existing records of the table.
    - If validation fails, display an error message and terminate the operation.
8. **Format Values:**
    - Trim and clean the values.
    - Format the values appropriately for insertion.
9. **Write Values to File:**
    - Append the formatted values to the table file.
    - Display a success message.
    - If an error occurs during file operations, display an error message.
10. **Handle Invalid Query:**
    - If the query format is invalid, display an error message.

Select From Table

1. **Check Database Selection:**
    - Ensure a database is currently selected. If not, terminate the operation.
2. **Validate Query:**
    - Use regex to match and validate the SELECT FROM query format.
3. **Extract Query Components:**
    - Extract the columns to be selected, table name, condition column, and condition value from the query.
4. **Check Table Existence:**
    - Obtain the table file using TableUtils.getTableFile.
    - If the table does not exist, display an error message and terminate the operation.
5. **Read Table Data:**
    - Read the contents of the table file.
    - If the table file is empty or could not be read, display an error message and terminate the operation.
6. **Extract Headers:**
    - Read the first line of the table file to get the column headers.
7. **Validate Columns:**
    - Determine the indices of the columns to be selected based on the headers.
    - If a requested column is not found, display an error message and terminate the operation.
8. **Validate Condition Column:**
    - If a condition column is specified, determine its index.
    - If the condition column is not found, display an error message and terminate the operation.
9. **Search for Matching Records:**
    - Initialize a flag to check if any matching records are found.
    - Iterate through each line in the table (excluding the header).
    - Split each line into column values.
    - If the condition column is specified, check if the condition value matches the corresponding column value.
10. **Display Headers:**
    - If a matching record is found, display the headers for the selected columns (only once).
11. **Display Matching Records:**
    - For each matching record, display the values of the selected columns.
12. **Handle No Matching Records:**
    - If no matching records are found, display an error message indicating no matches.
13. **Handle Invalid Query:**
    - If the query format is invalid, display an error message.

Update Table

1. **Check Database Selection:**
    - Ensure a database is currently selected. If not, terminate the operation.
2. **Validate Query:**
    - Use regex to match and validate the UPDATE TABLE query format.
3. **Extract Query Components:**
    - Extract the table name, update column, update value, condition column, and condition value from the query.
4. **Check Table Existence:**
    - Obtain the table file using TableUtils.getTableFile.
    - If the table does not exist, terminate the operation.
5. **Read Table Data:**
    - Read the contents of the table file.
    - If the table file is empty or could not be read, terminate the operation.
6. **Extract Headers:**
    - Read the first line of the table file to get the column headers.
7. **Validate Update and Condition Columns:**
    - Determine the indices of the update and condition columns based on the headers.
    - If the update column is not found, display an error message and terminate the operation.
    - If the condition column is not found, display an error message and terminate the operation.
8. **Update Records:**
    - Initialize a flag to check if any records are updated.
    - Iterate through each line in the table (excluding the header).
    - Split each line into column values.
    - If the condition column value matches the condition, update the update column value.
9. **Write Updated Data to File:**
    - If records are updated, write the updated data back to the table file.
    - Display a success message.
    - If no records are updated, display an error message indicating no matches.
10. **Handle File Writing Errors:**
    - If an error occurs during file operations, display an error message.
11. **Handle Invalid Query:**
    - If the query format is invalid, display an error message.

Delete From Table

1. **Check Database Selection:**
    - Ensure a database is currently selected. If not, terminate the operation.
2. **Validate Query:**
    - Use regex to match and validate the DELETE FROM query format.
3. **Extract Query Components:**
    - Extract the table name, column, and value from the query.
4. **Check Table Existence:**
    - Obtain the table file using TableUtils.getTableFile.
    - If the table does not exist, display an error message and terminate the operation.
5. **Read Table Data:**
    - Read the contents of the table file.
    - If the table file is empty or could not be read, display an error message and terminate the operation.
6. **Handle Deletion Without WHERE Clause:**
    - If both column and value are null, delete all records (except the header) from the table.
    - Write the updated data (header only) back to the table file.
    - Display a success message indicating all records were deleted.
7. **Validate Condition Column:**
    - If a column is specified, determine its index based on the headers.
    - If the column is not found, display an error message and terminate the operation.
8. **Delete Matching Records:**
    - Initialize a flag to check if any matching records are found.
    - Iterate through each line in the table (excluding the header).
    - Split each line into column values.
    - If the specified column value matches, remove the record.
9. **Write Updated Data to File:**
    - If records are deleted, write the updated data back to the table file.
    - Display a success message.
    - If no matching records are found, display an error message indicating no matches.
10. **Handle File Writing Errors:**
    - If an error occurs during file operations, display an error message.
11. **Handle Invalid Query:**
    - If the query format is invalid, display an error message.

Functional Testing:

Authentication (Module 7):

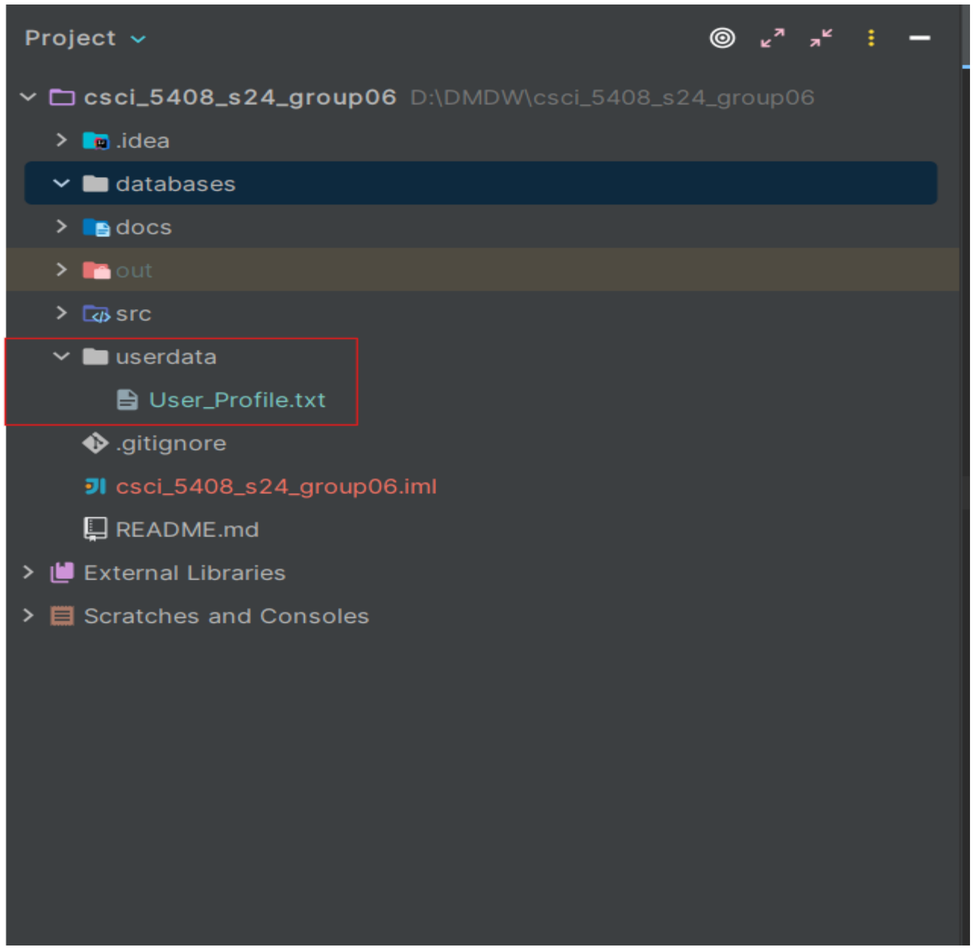
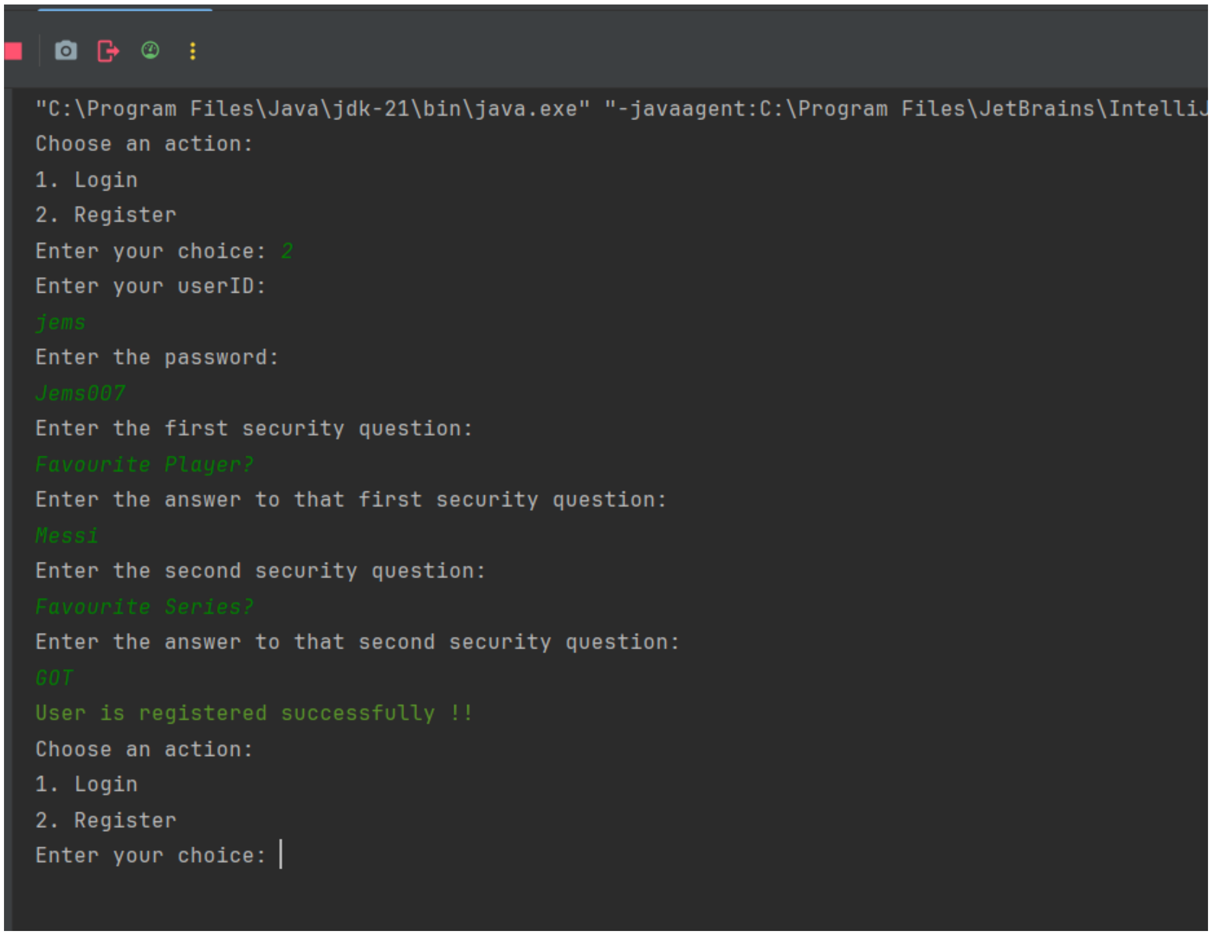
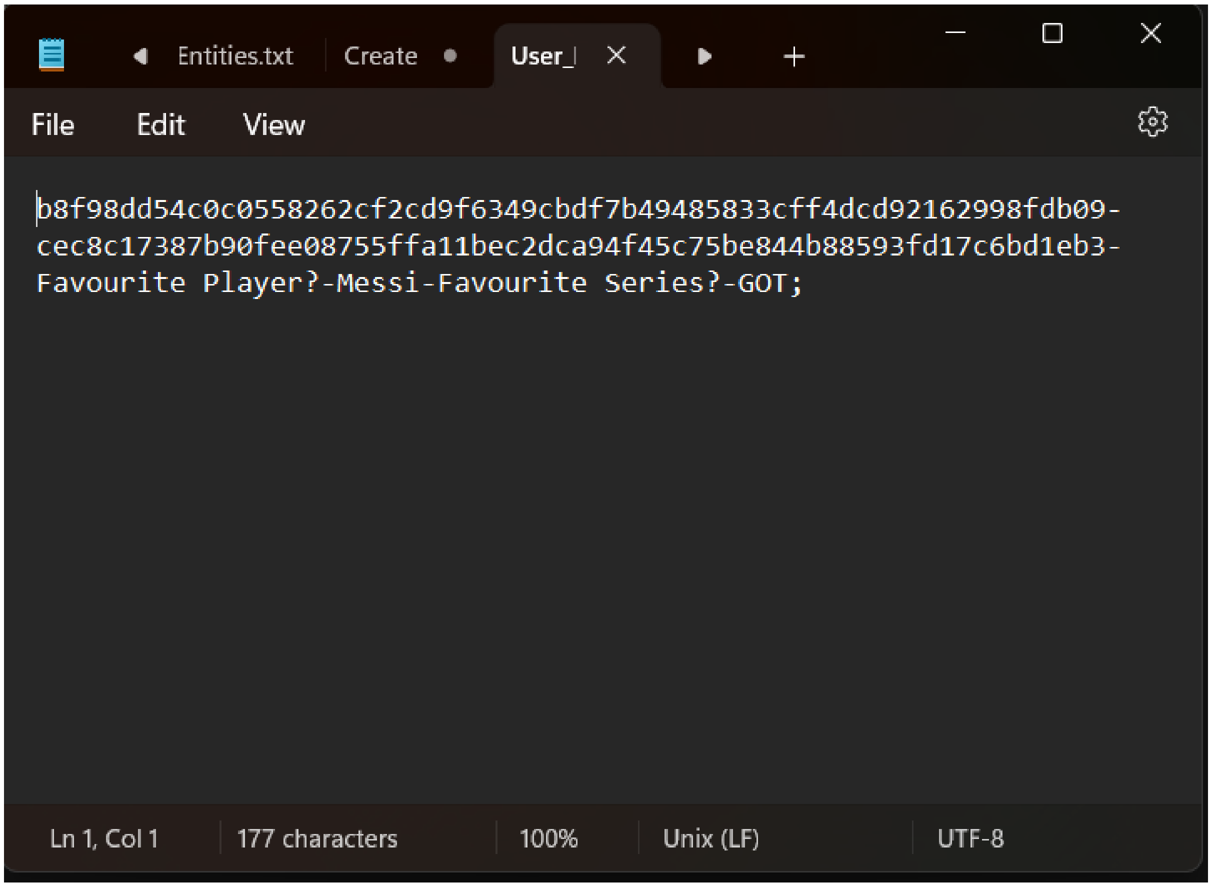


Figure: User profile file for the storing user data



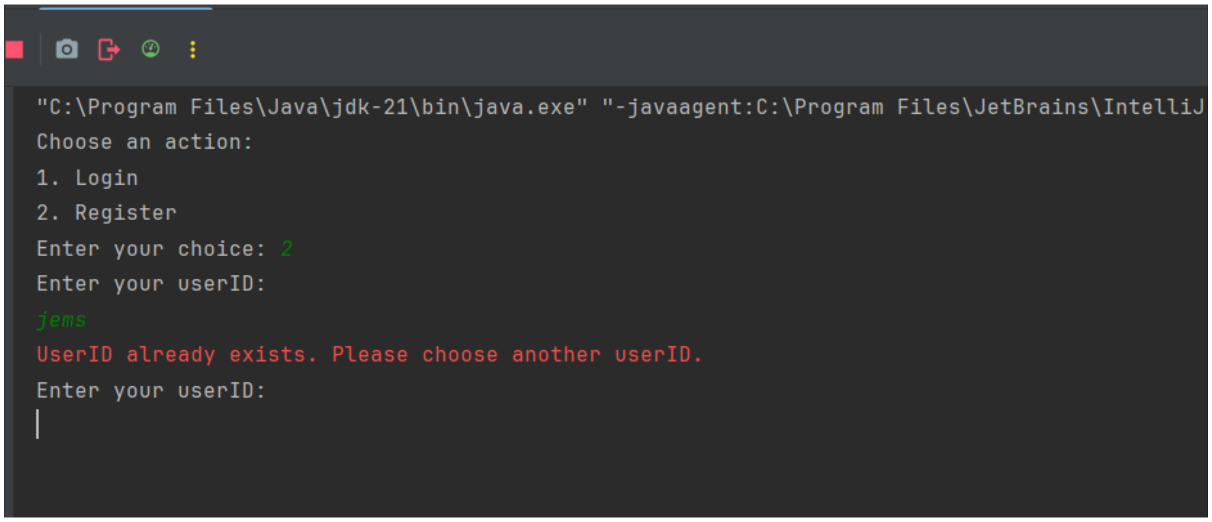
```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Choose an action:
1. Login
2. Register
Enter your choice: 2
Enter your userID:
jems
Enter the password:
Jems007
Enter the first security question:
Favourite Player?
Enter the answer to that first security question:
Messi
Enter the second security question:
Favourite Series?
Enter the answer to that second security question:
GOT
User is registered successfully !!
Choose an action:
1. Login
2. Register
Enter your choice:
```

Figure: Console for authenticating the user using username and password along with security question



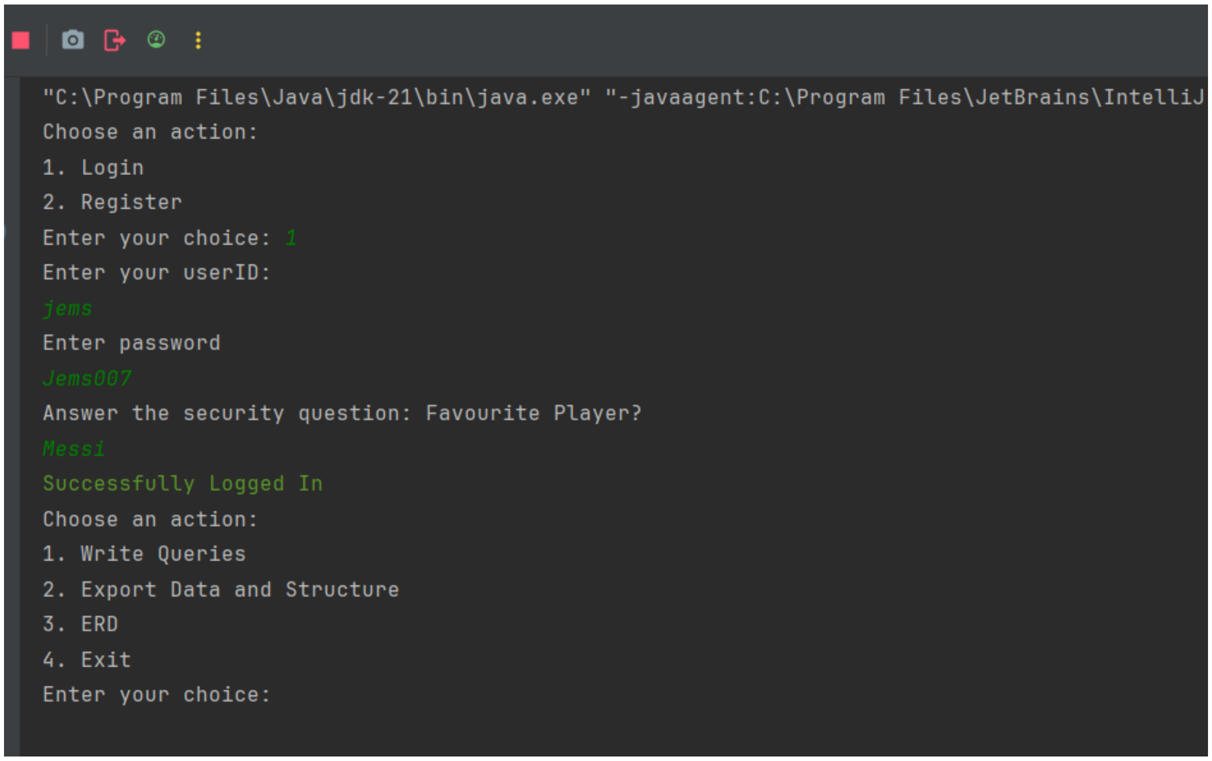
```
b8f98dd54c0c0558262cf2cd9f6349cbdf7b49485833cff4dcd92162998fdb09-
cec8c17387b90fee08755ffa11bec2dca94f45c75be844b88593fd17c6bd1eb3-
Favourite Player?-Messi-Favourite Series?-GOT;
```

Figure: user details stored in encrypted forms



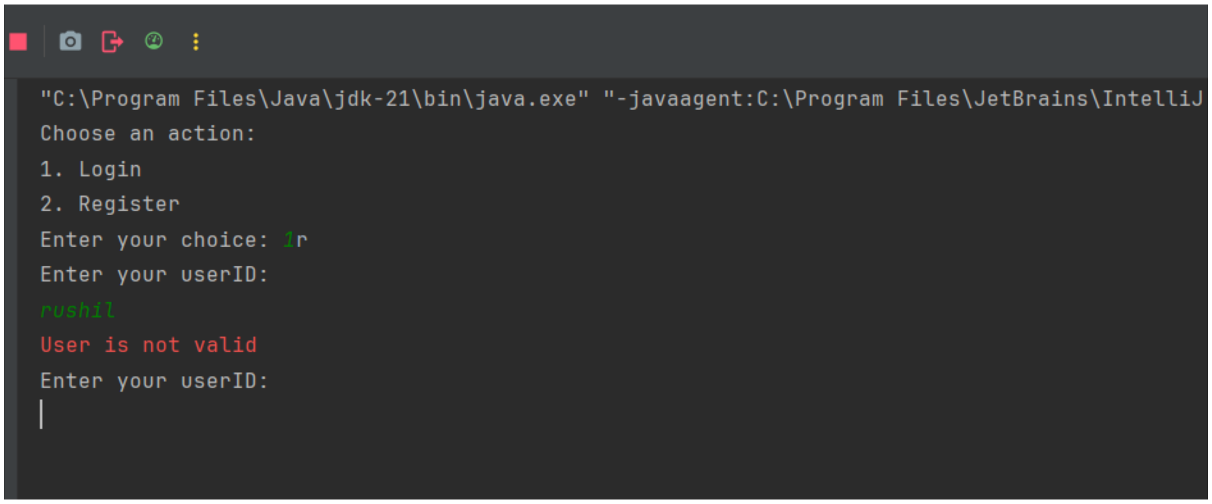
```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Choose an action:
1. Login
2. Register
Enter your choice: 2
Enter your userID:
jems
UserID already exists. Please choose another userID.
Enter your userID:
```

Figure: registering with same userId



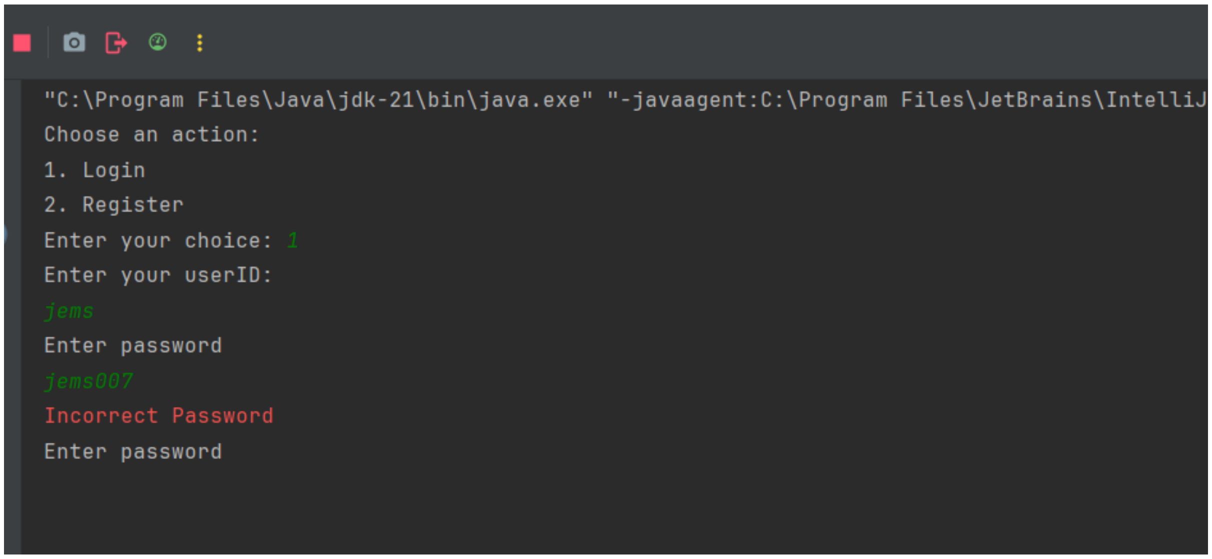
```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Choose an action:
1. Login
2. Register
Enter your choice: 1
Enter your userID:
jems
Enter password
Jems007
Answer the security question: Favourite Player?
Messi
Successfully Logged In
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: Login valid user with right security question



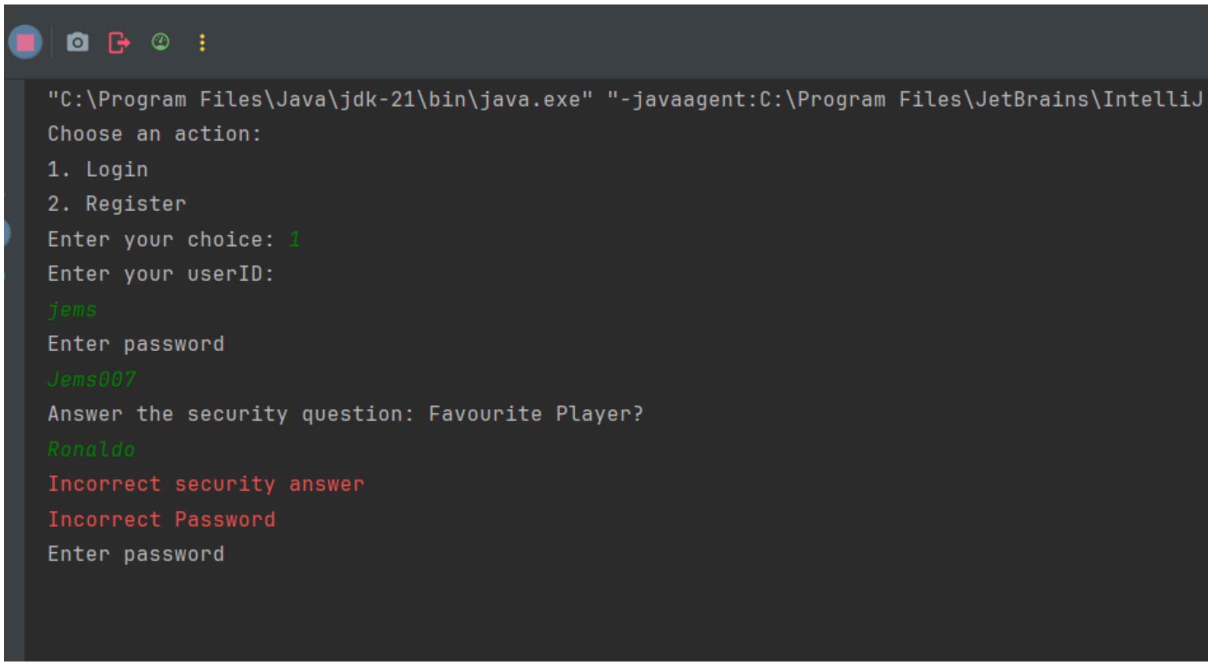
```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Choose an action:
1. Login
2. Register
Enter your choice: 1r
Enter your userID:
rushil
User is not valid
Enter your userID:
```

Figure: Login with invalid userId



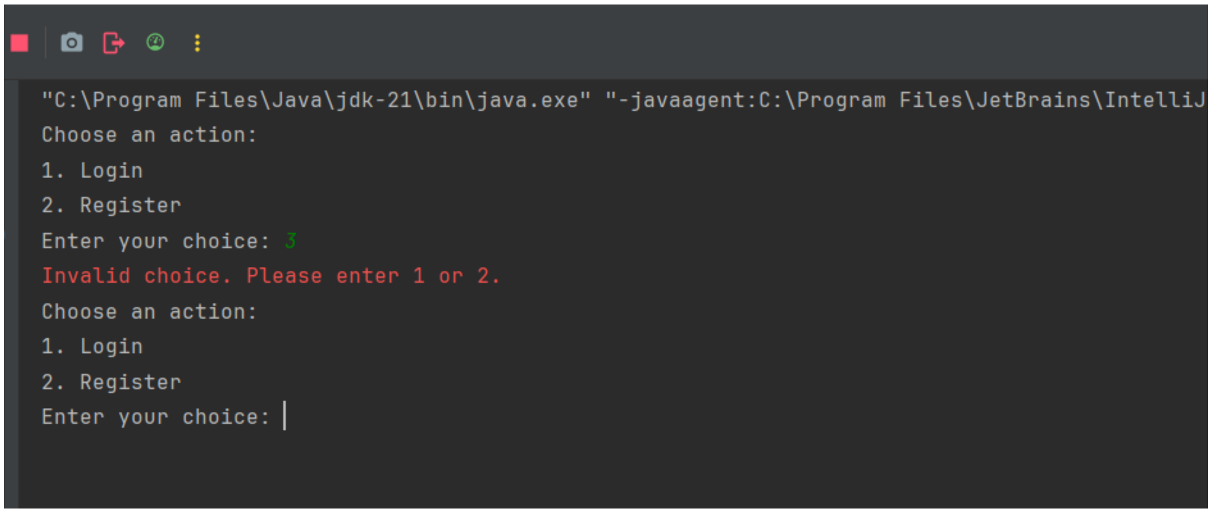
```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Choose an action:
1. Login
2. Register
Enter your choice: 1
Enter your userID:
jems
Enter password
jems007
Incorrect Password
Enter password
```

Figure: Login with invalid password



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Choose an action:
1. Login
2. Register
Enter your choice: 1
Enter your userID:
jems
Enter password
Jems007
Answer the security question: Favourite Player?
Ronaldo
Incorrect security answer
Incorrect Password
Enter password
```

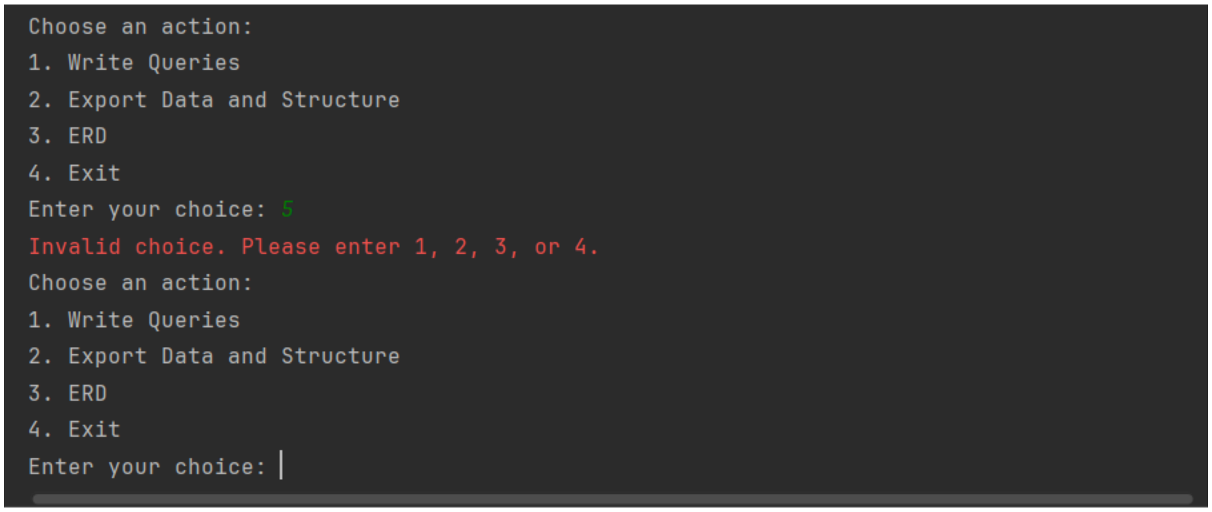
Figure: Login with incorrect security answer



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Choose an action:
1. Login
2. Register
Enter your choice: 3
Invalid choice. Please enter 1 or 2.
Choose an action:
1. Login
2. Register
Enter your choice: 
```

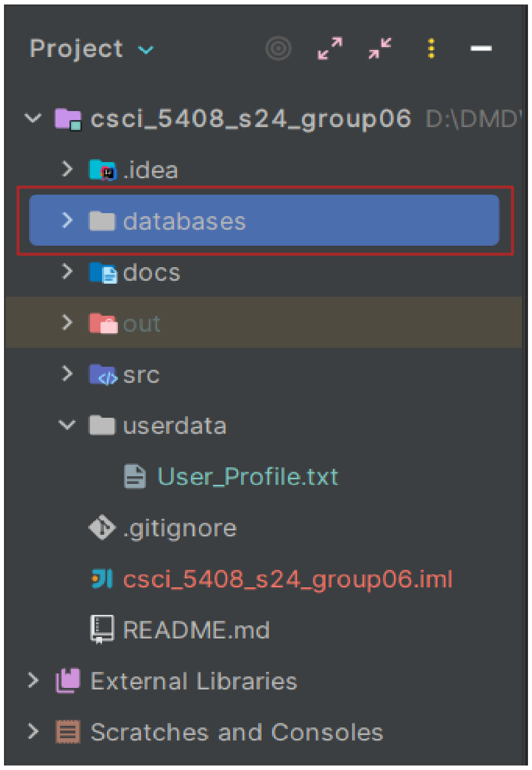
Figure: choosing invalid option for the system

Query Processing (Module 2):



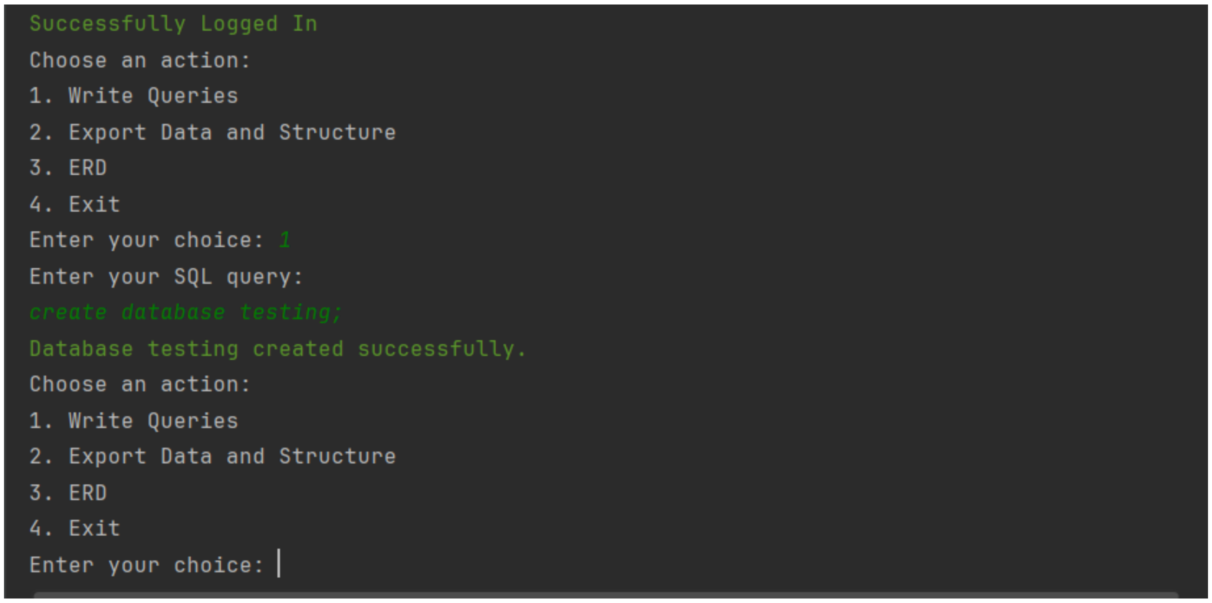
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 5
Invalid choice. Please enter 1, 2, 3, or 4.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 
```

Figure: Invalid choice for the actions



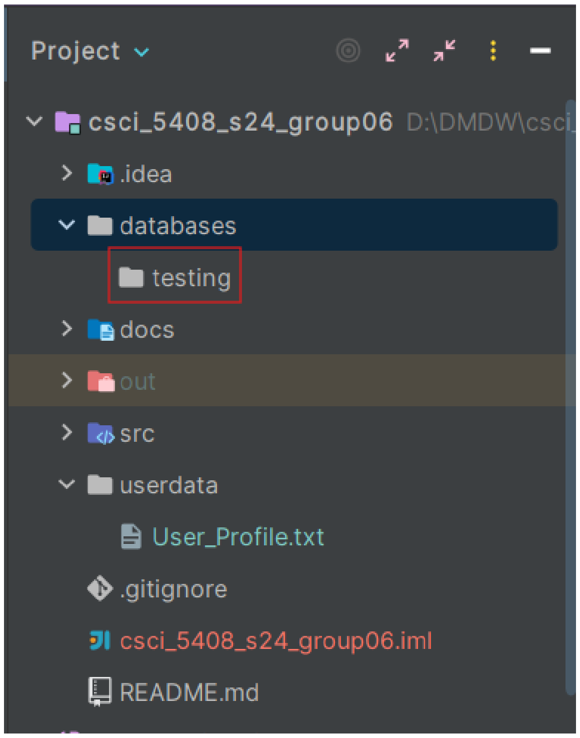
```
Project
csci_5408_s24_group06 D:\DMD'
  .idea
  databases
  docs
  out
  src
  userdata
    User_Profile.txt
  .gitignore
  csci_5408_s24_group06.iml
  README.md
External Libraries
Scratches and Consoles
```

Figure: Directory for databases operations



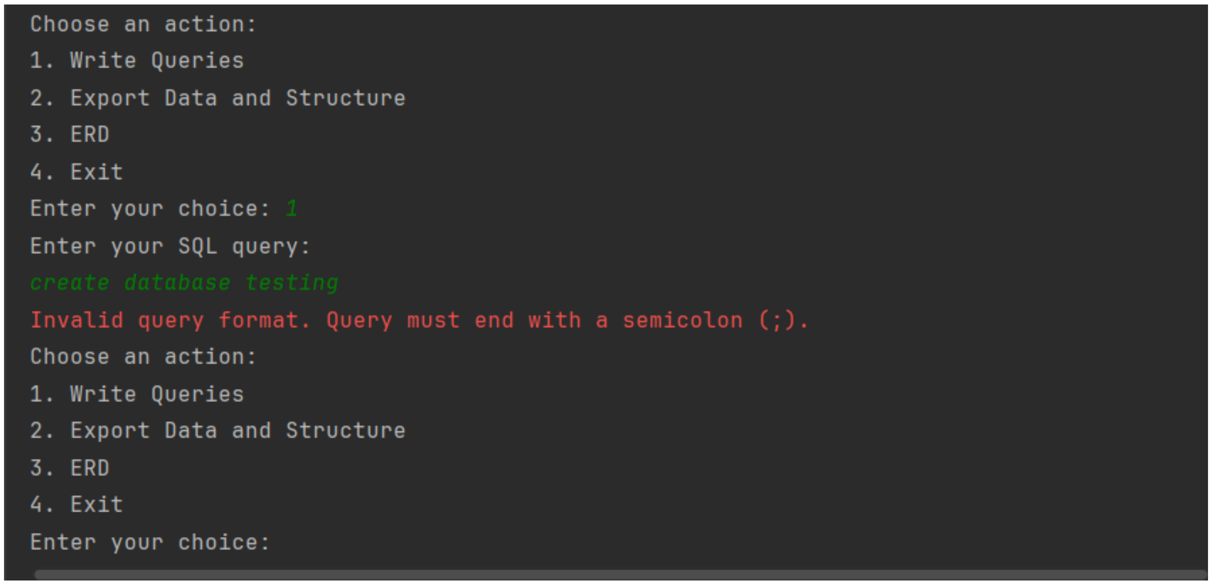
```
Successfully Logged In
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
create database testing;
Database testing created successfully.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 
```

Figure : Creating the databases after successful authentication



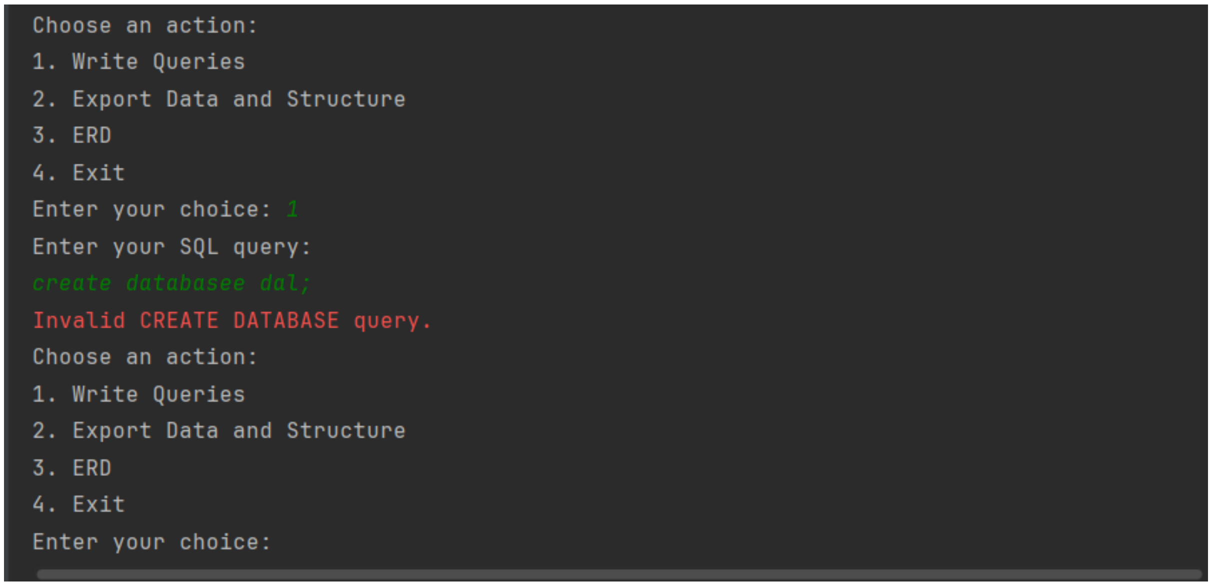
```
Project
csci_5408_s24_group06  D:\DMDW\csci_
  .idea
  databases
    testing
  docs
  out
  src
  userdata
    User_Profile.txt
  .gitignore
  csci_5408_s24_group06.iml
  README.md
```

Figure: testing directory inside the databases



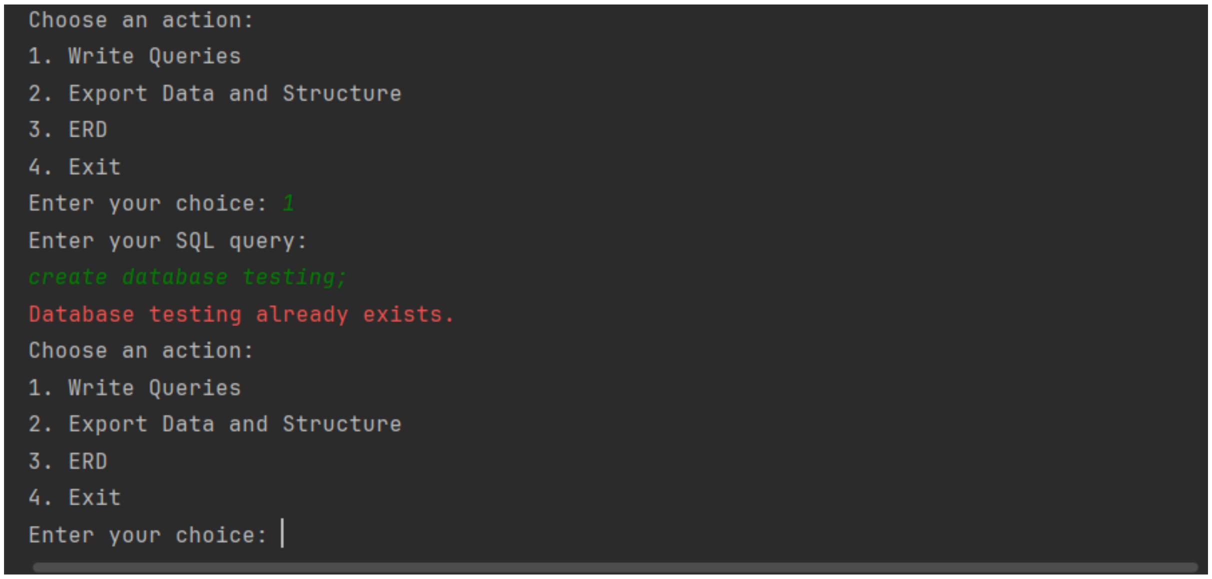
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
create database testing
Invalid query format. Query must end with a semicolon (;).
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure : performing valid query without semicolon



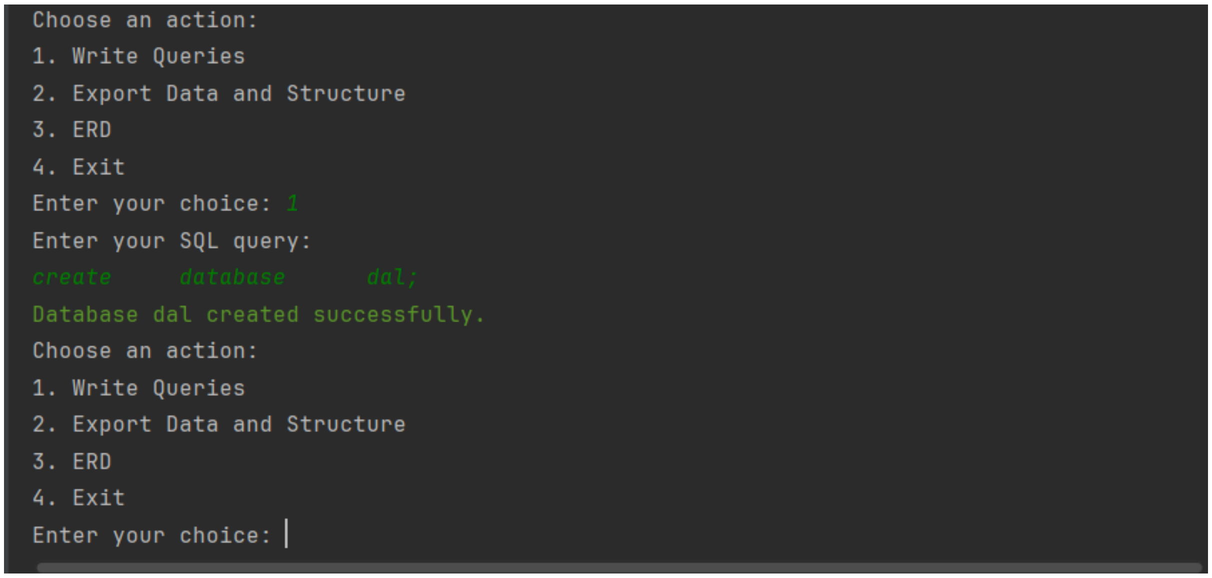
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
create databasee dal;
Invalid CREATE DATABASE query.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: invalid query format with wrong database spelling



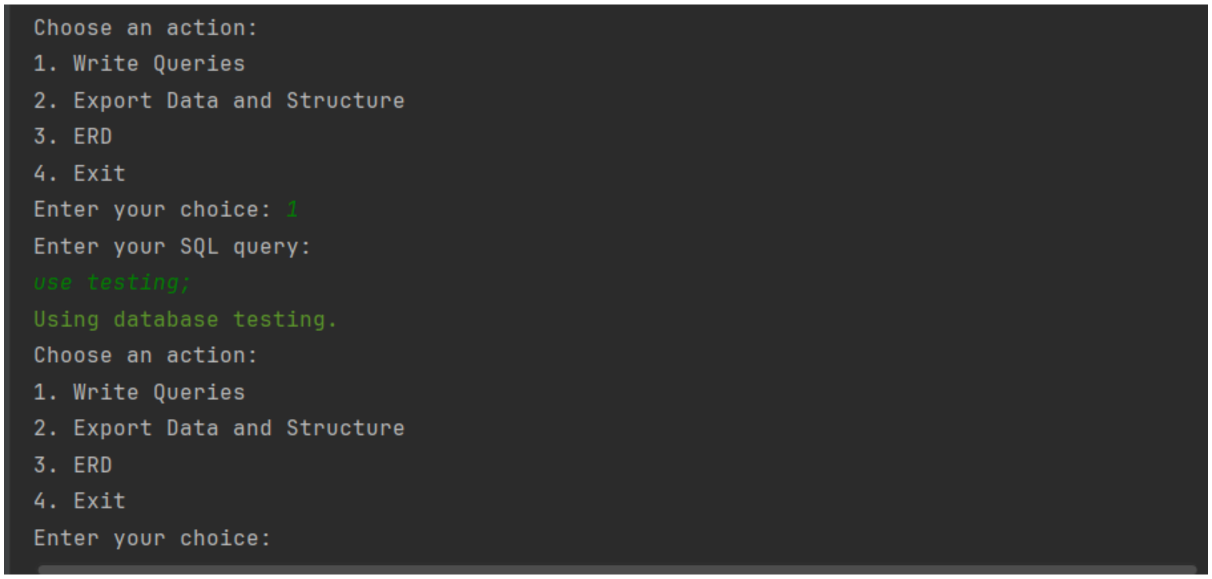
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
create database testing;
Database testing already exists.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 
```

Figure: Creating the same database again



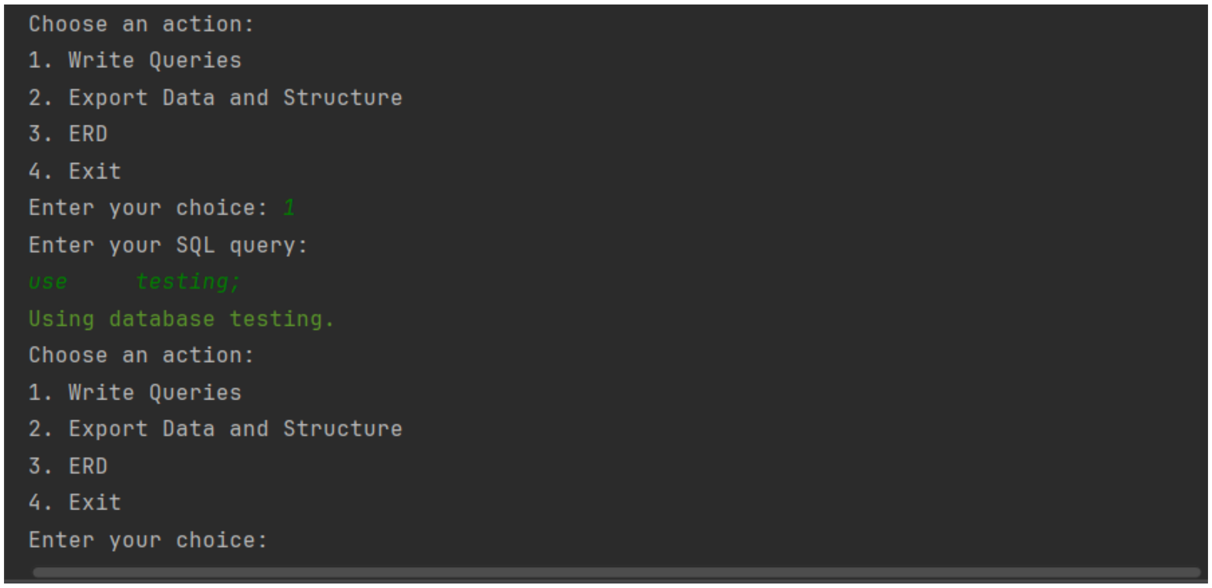
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
create      database      dal;
Database dal created successfully.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 
```

Figure: Creating database with valid format



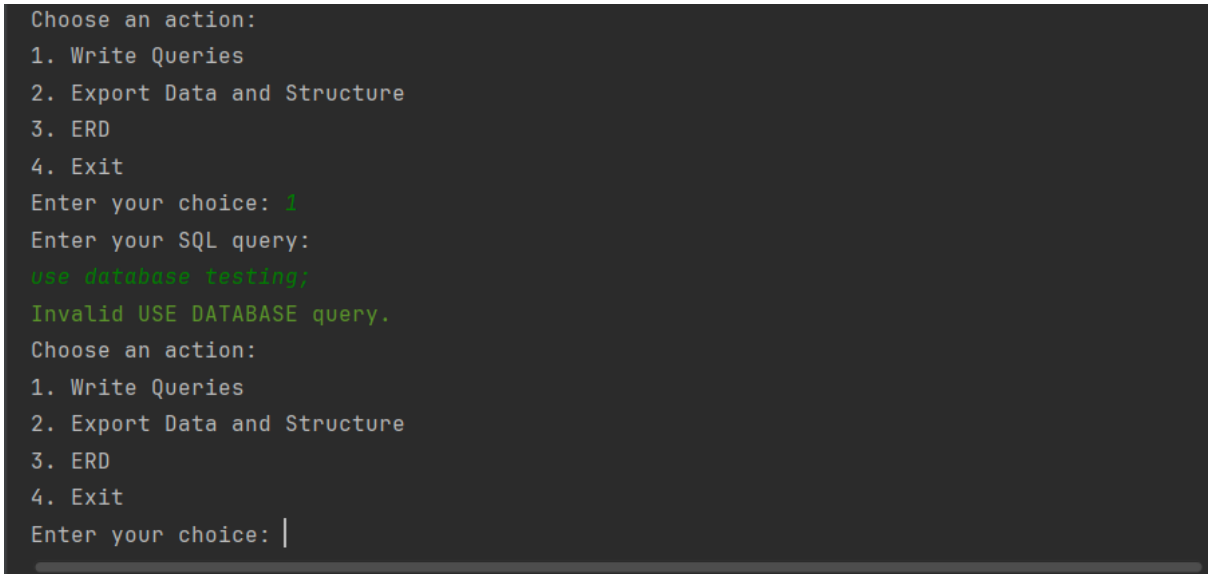
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
use testing;
Using database testing.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: use query with valid format



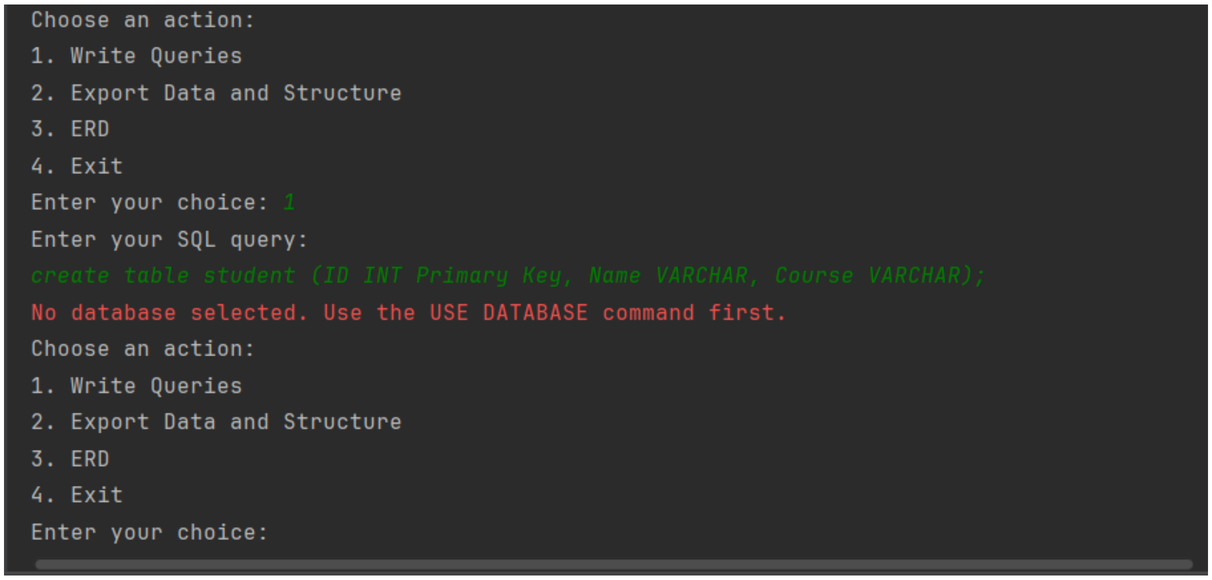
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
use     testing;
Using database testing.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: performing use query with right format



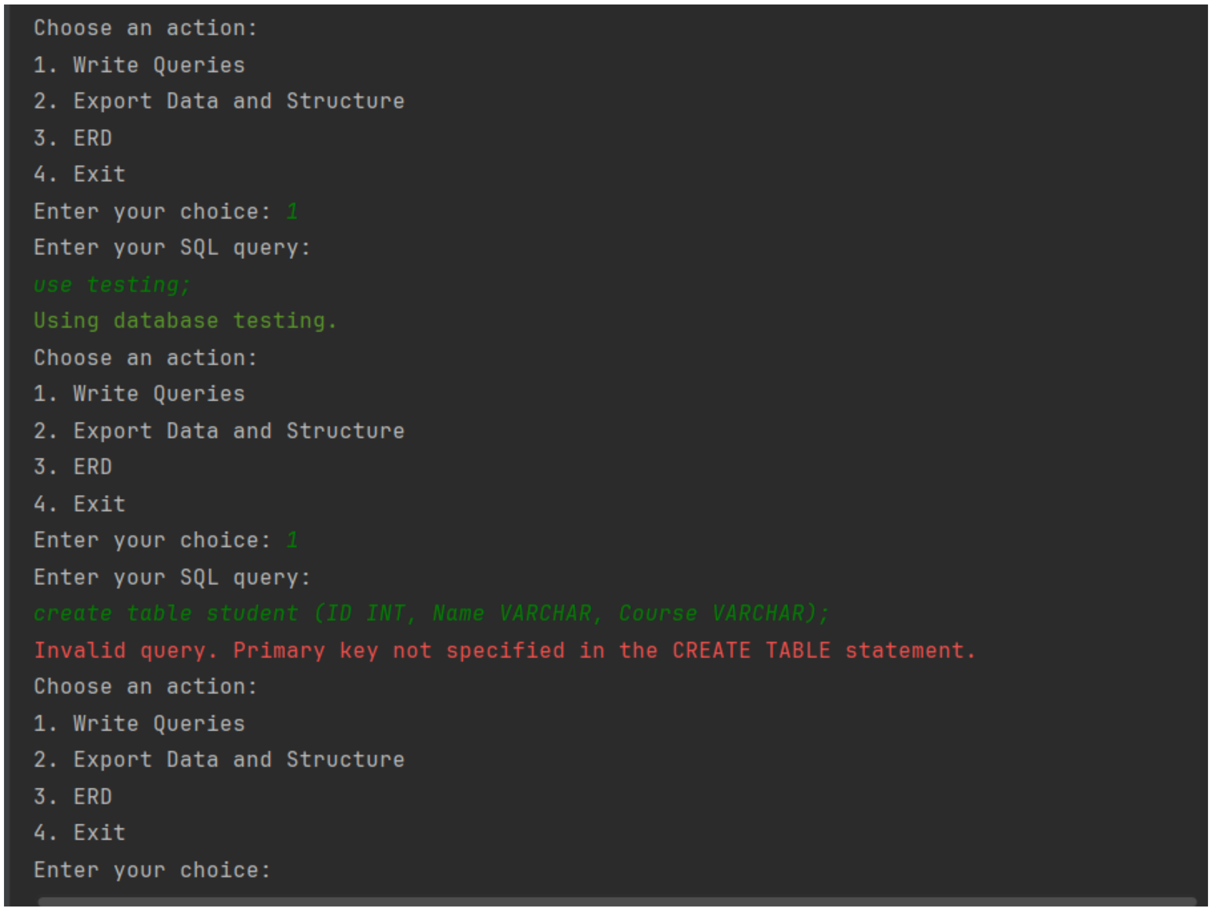
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
use database testing;
Invalid USE DATABASE query.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: use query with wrong format



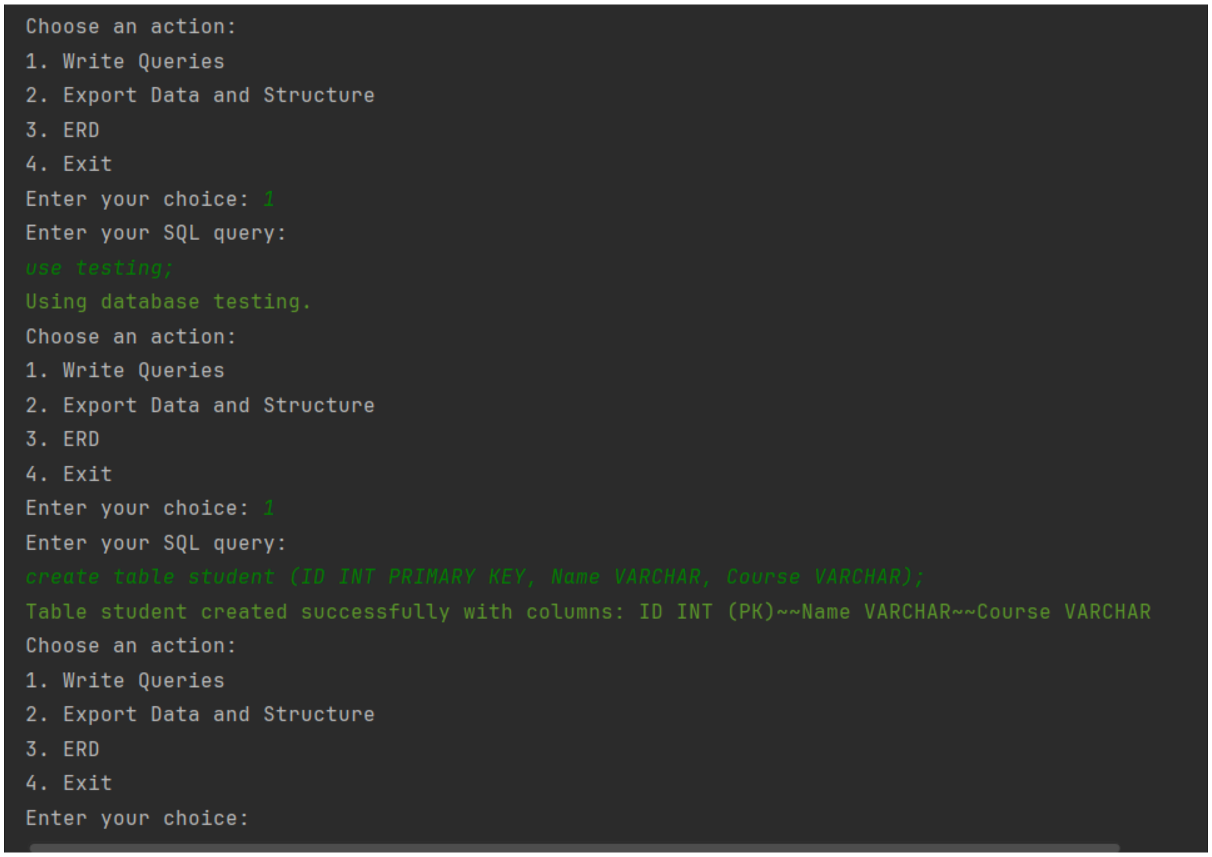
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
create table student (ID INT Primary Key, Name VARCHAR, Course VARCHAR);
No database selected. Use the USE DATABASE command first.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: Creating table without using the database



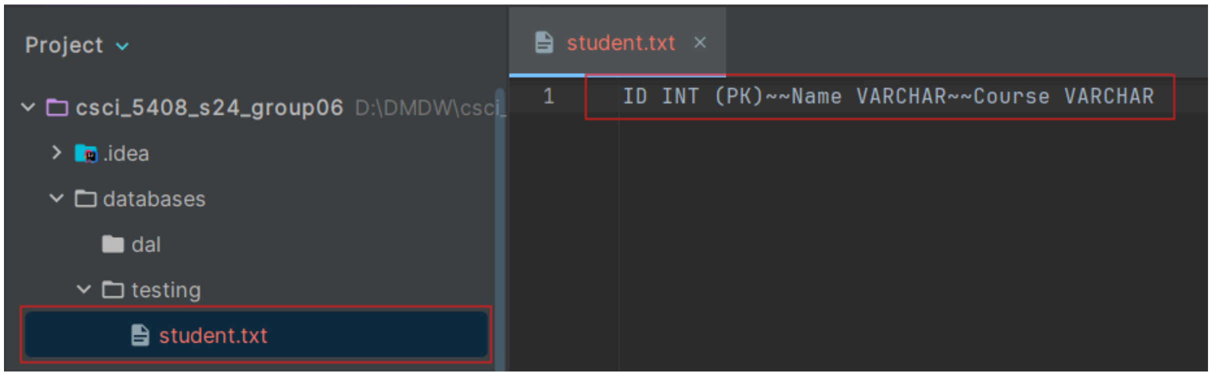
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
use testing;
Using database testing.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
create table student (ID INT, Name VARCHAR, Course VARCHAR);
Invalid query. Primary key not specified in the CREATE TABLE statement.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: creating table without specifying key



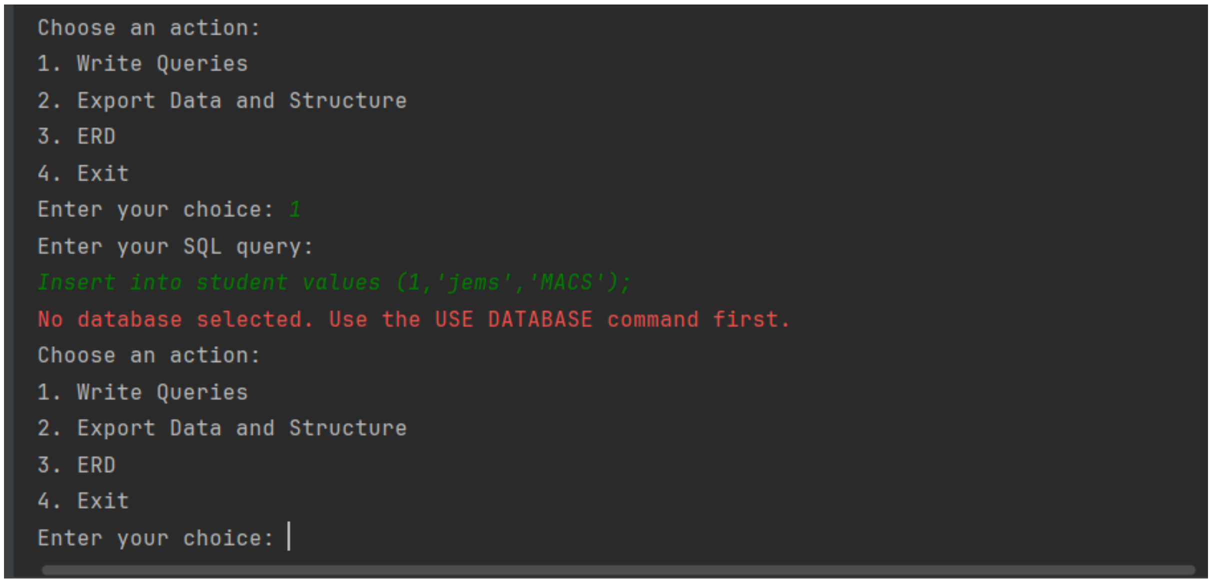
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
use testing;
Using database testing.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
create table student (ID INT PRIMARY KEY, Name VARCHAR, Course VARCHAR);
Table student created successfully with columns: ID INT (PK)~~Name VARCHAR~~Course VARCHAR
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: valid create table format with primary key specification



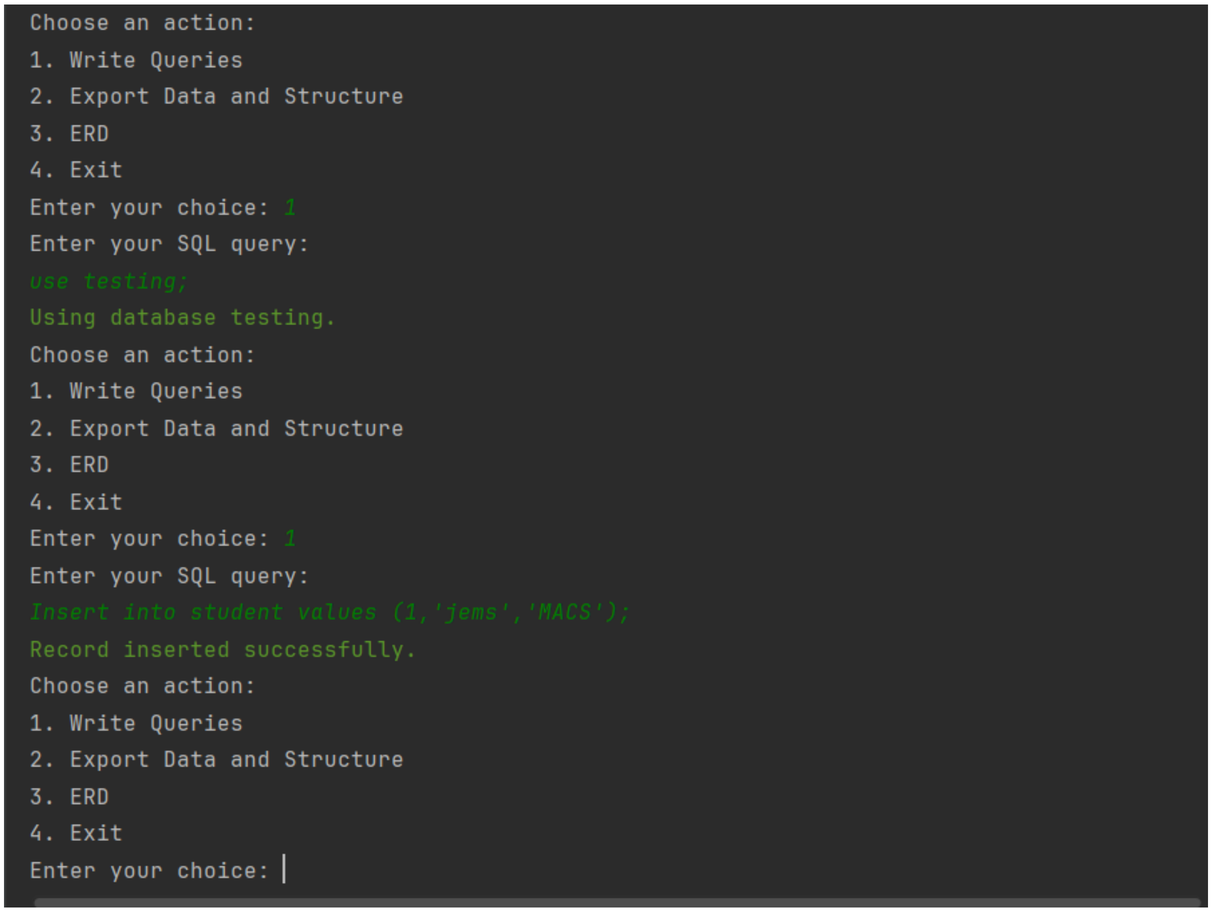
```
ID INT (PK)~~Name VARCHAR~~Course VARCHAR
```

Figure: creation of table with custom structure with double tilde



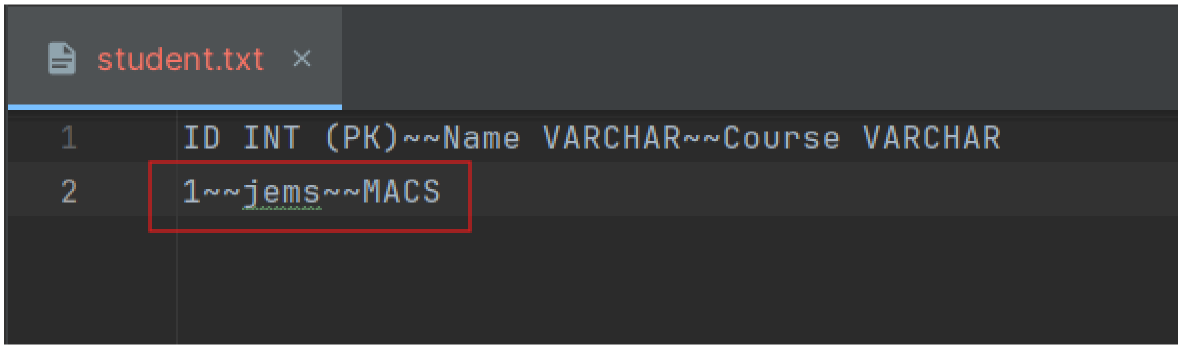
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
Insert into student values (1,'jems','MACS');
No database selected. Use the USE DATABASE command first.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: inserting into the table without selecting the database



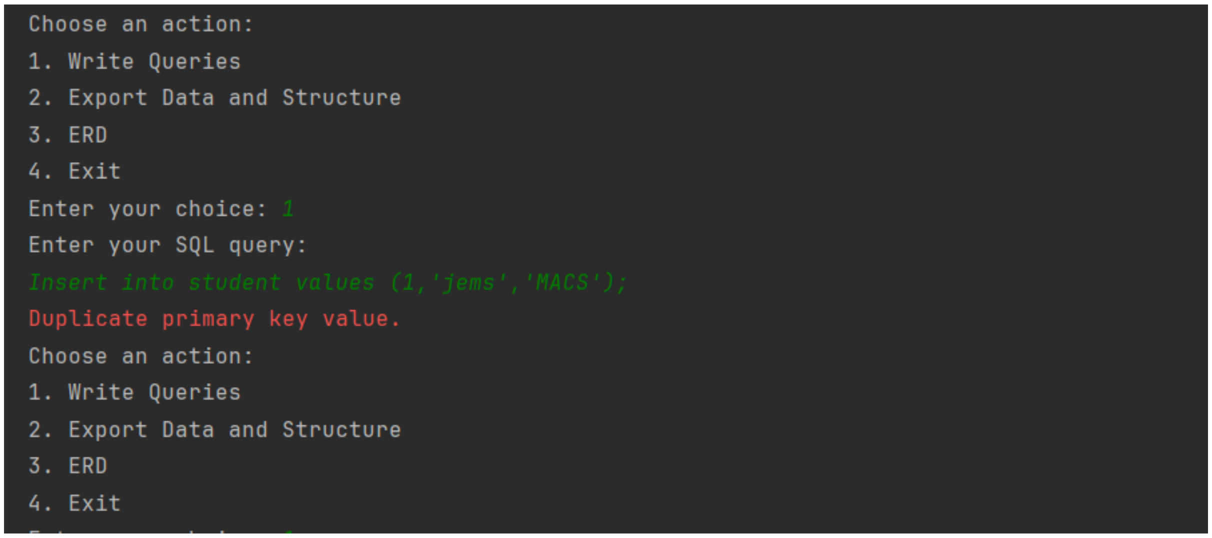
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
use testing;
Using database testing.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
Insert into student values (1,'jems','MACS');
Record inserted successfully.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: inserting into students with the usage of database



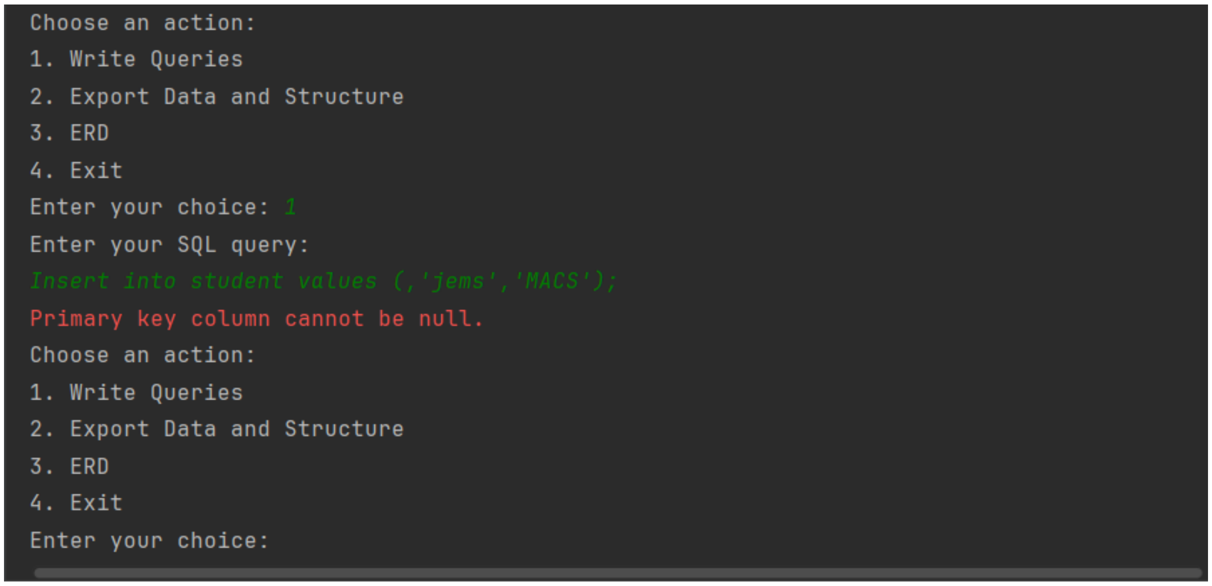
```
ID INT (PK)~~Name VARCHAR~~Course VARCHAR
1~~jems~~MACS
```

Figure: Data inserted into the table with double tilde separation



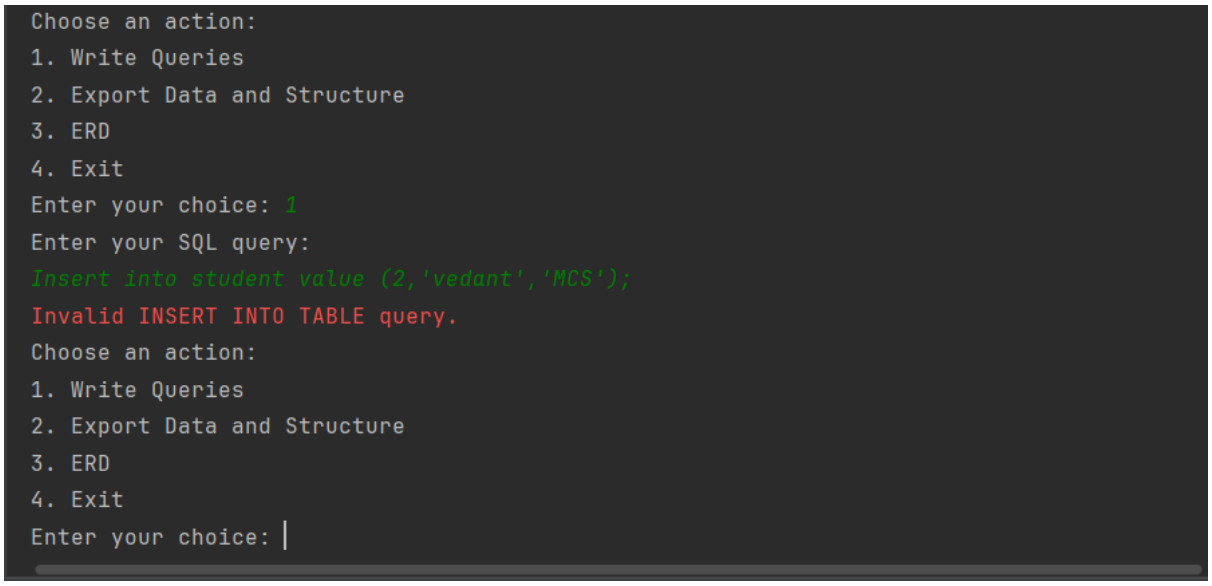
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
Insert into student values (1,'jems','MACS');
Duplicate primary key value.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
```

Figure: Inserting the same data with duplicate key



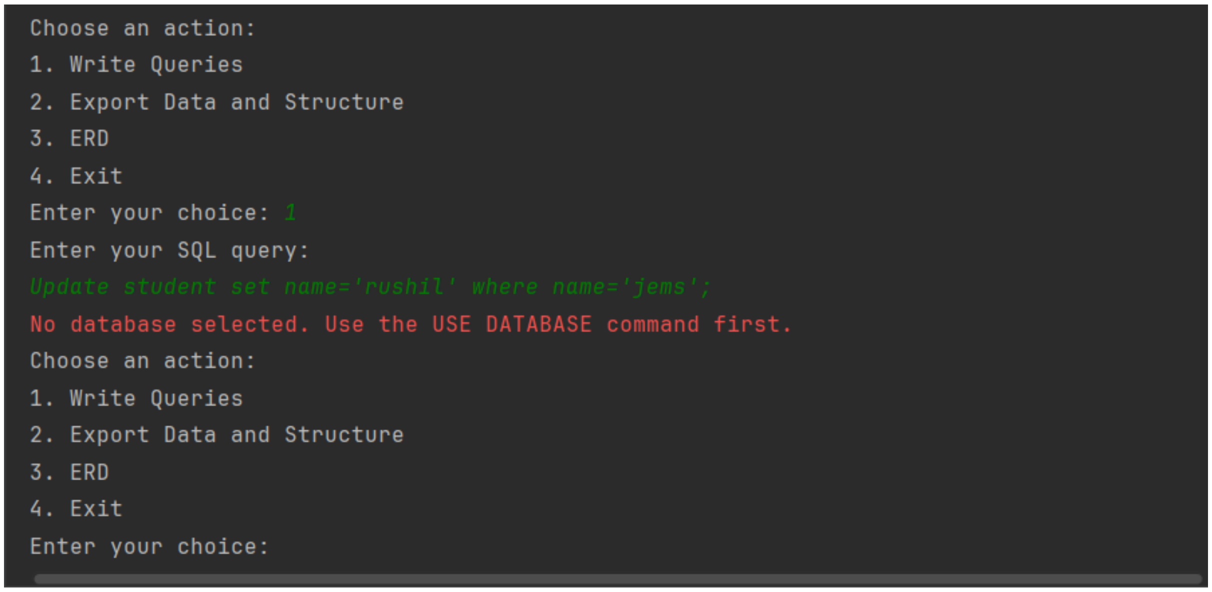
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
Insert into student values (,'jems','MACS');
Primary key column cannot be null.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: Inserting the data with null primary key



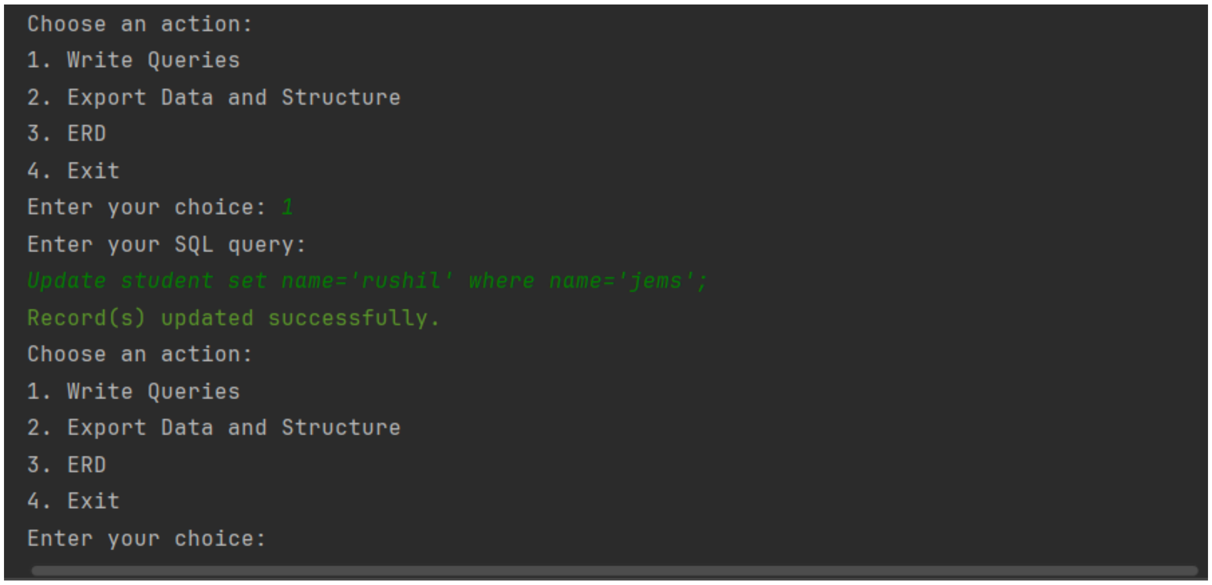
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
Insert into student value (2,'vedant','MCS');
Invalid INSERT INTO TABLE query.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 
```

Figure: Inserting with invalid query format



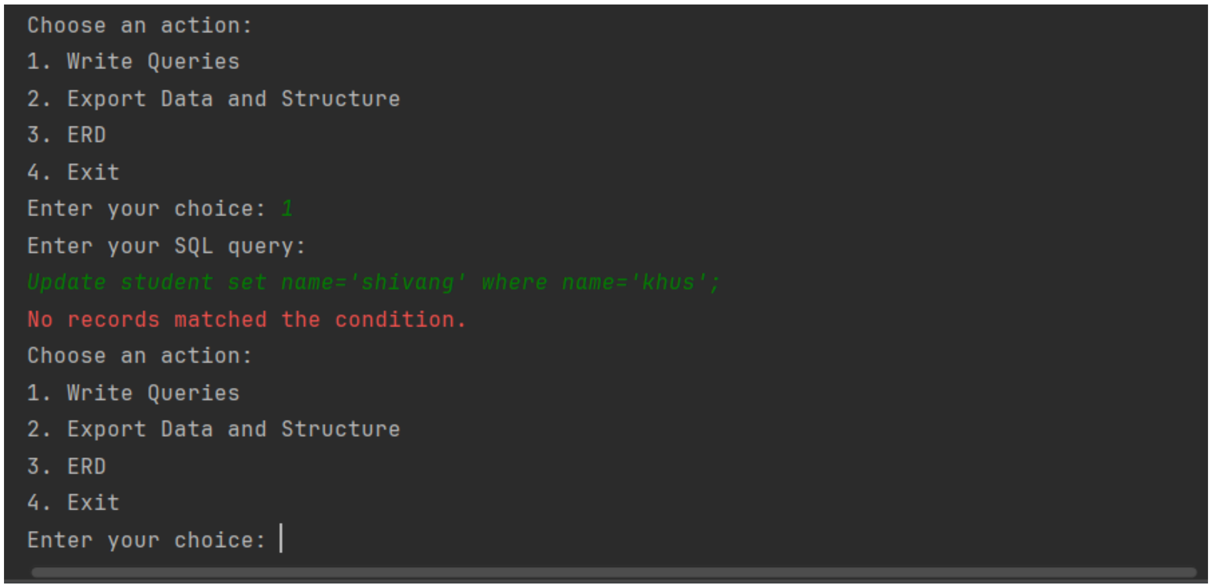
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
Update student set name='rushil' where name='jems';
No database selected. Use the USE DATABASE command first.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: updating the table without selecting the database



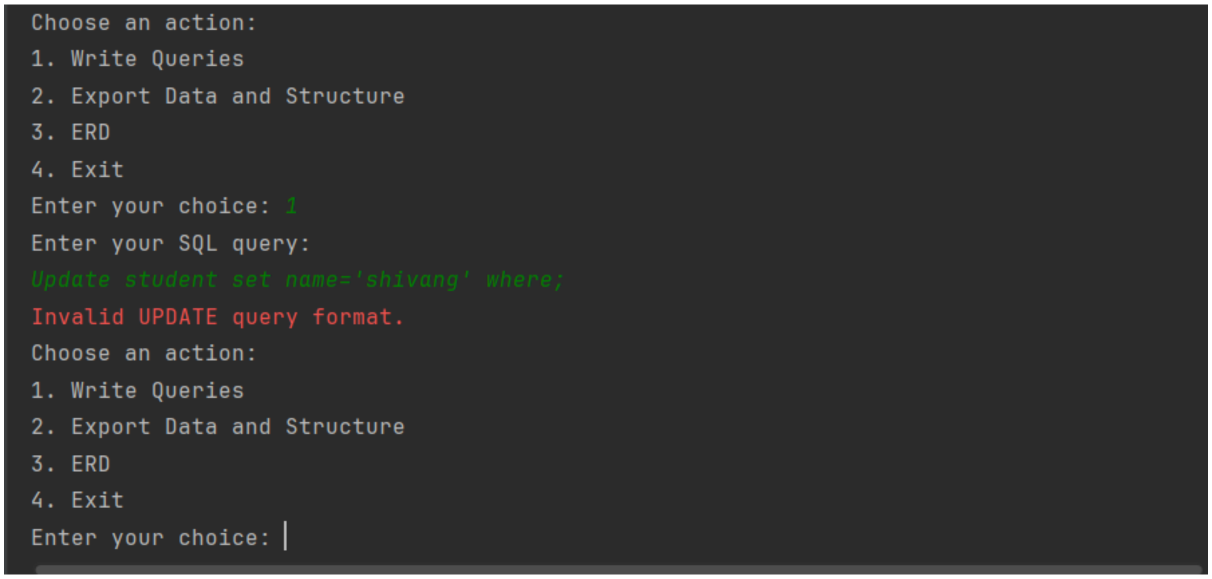
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
Update student set name='rushil' where name='jems';
Record(s) updated successfully.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: Updating the user with valid format



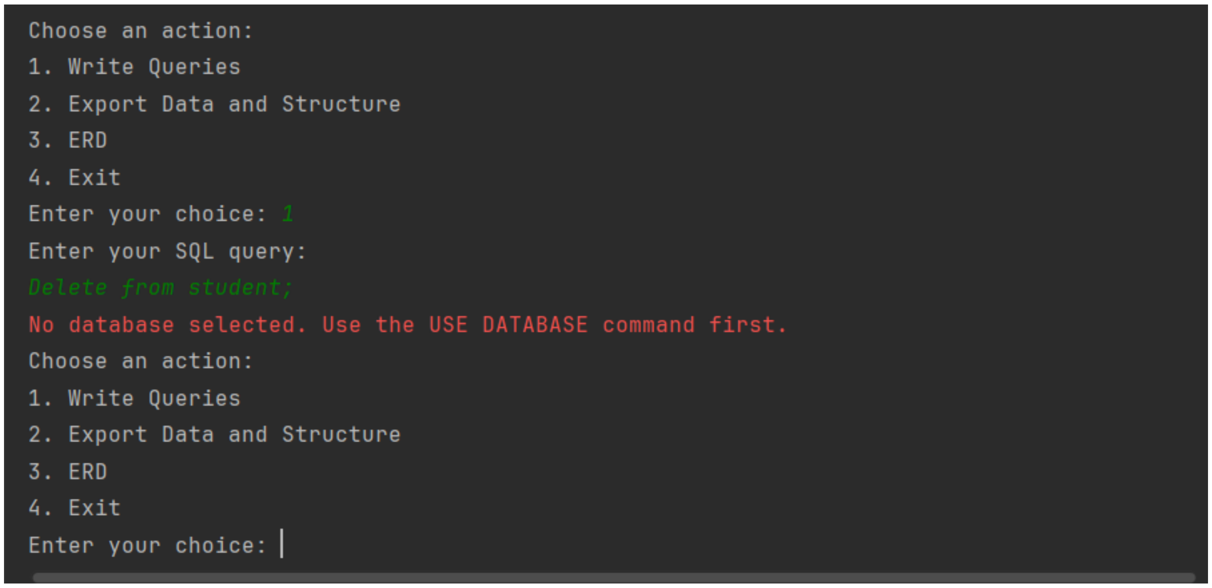
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
Update student set name='shivang' where name='khus';
No records matched the condition.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: updating the user with non- existing record



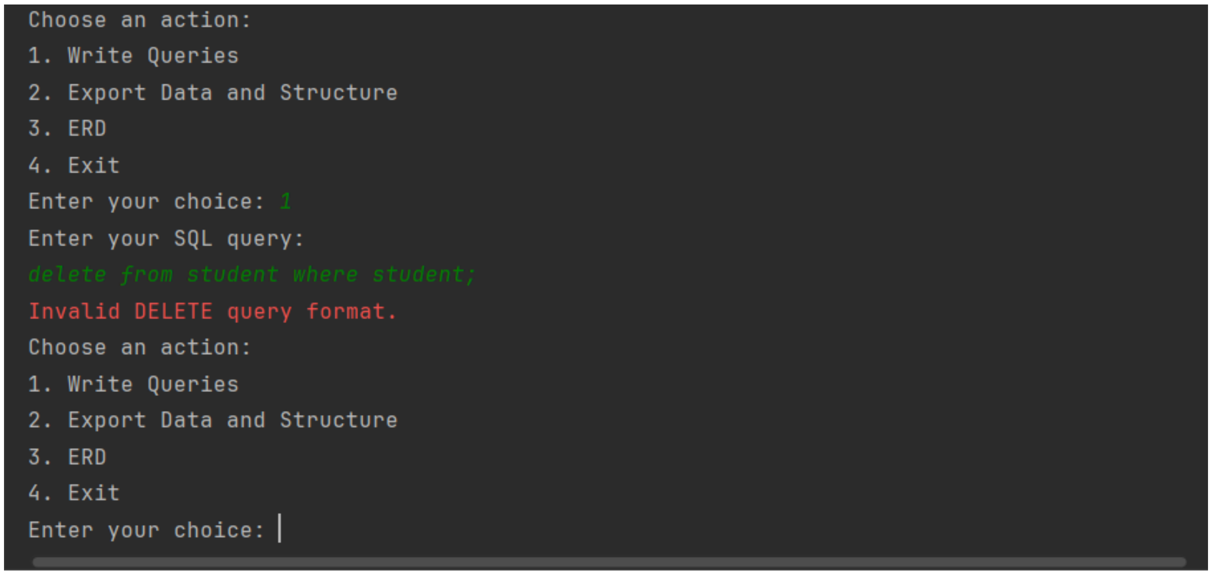
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
Update student set name='shivang' where;
Invalid UPDATE query format.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 
```

Figure: updating the user with invalid Query format



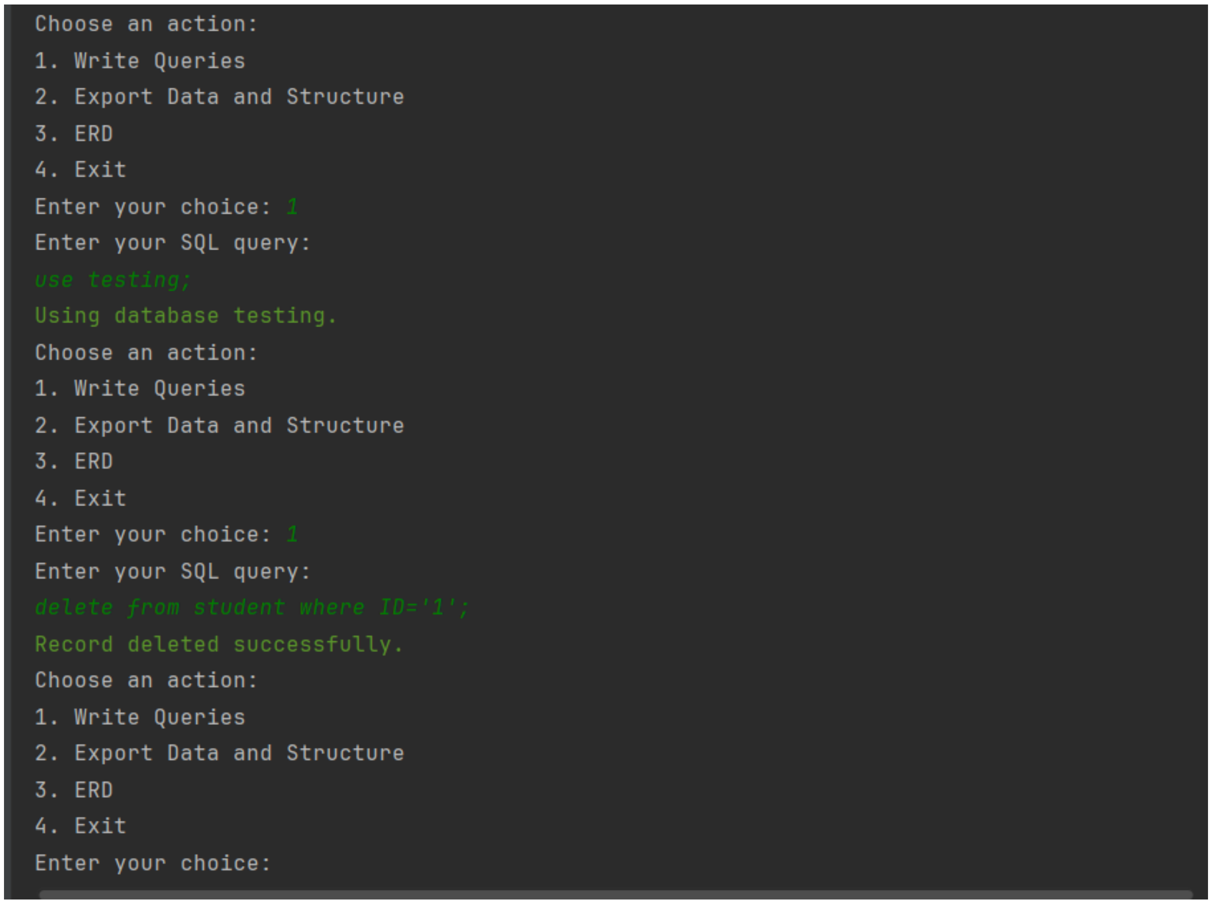
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
Delete from student;
No database selected. Use the USE DATABASE command first.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 
```

Figure: Deleting without selecting the database



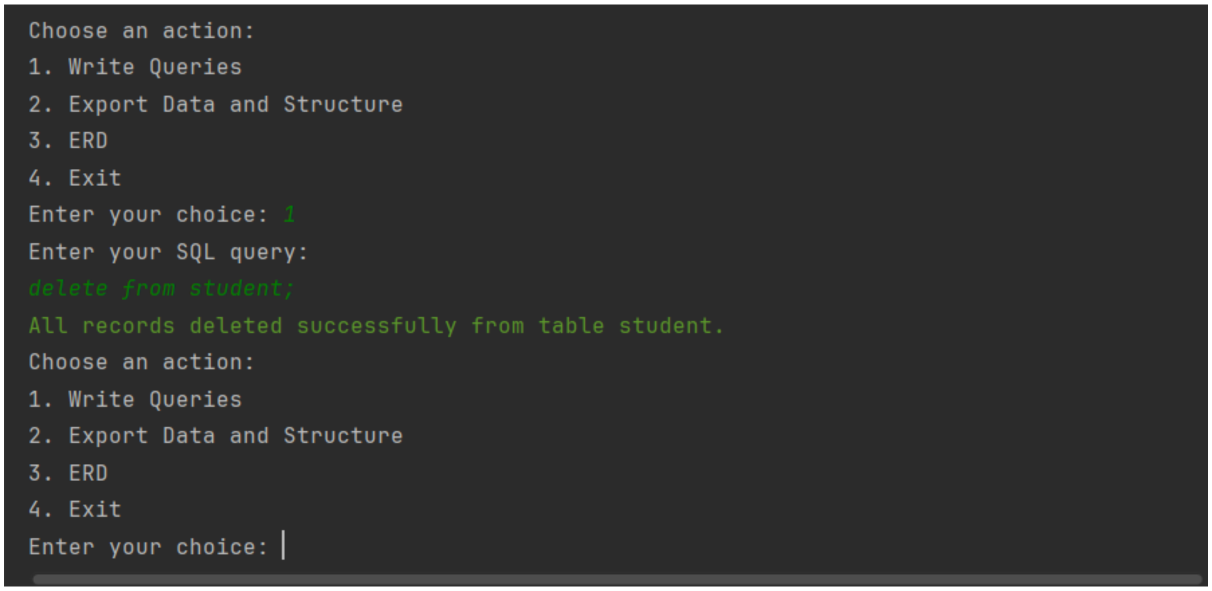
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
delete from student where student;
Invalid DELETE query format.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 
```

Figure: Deleting from student with invalid delete query.



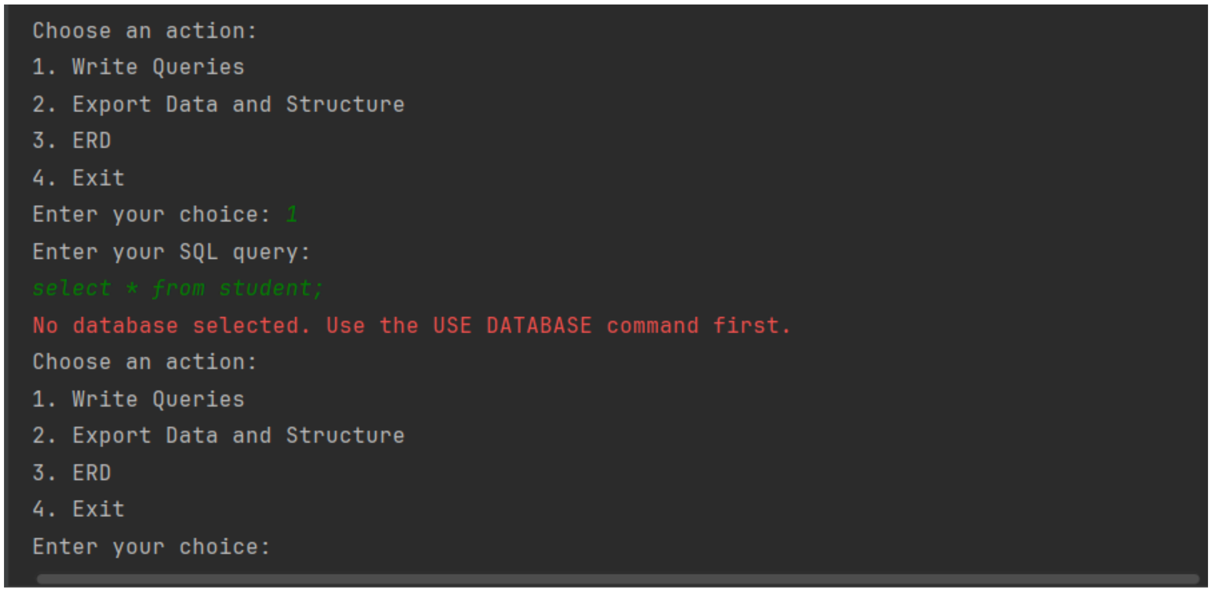
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
use testing;
Using database testing.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
delete from student where ID='1';
Record deleted successfully.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: Deleting the record with valid delete query.



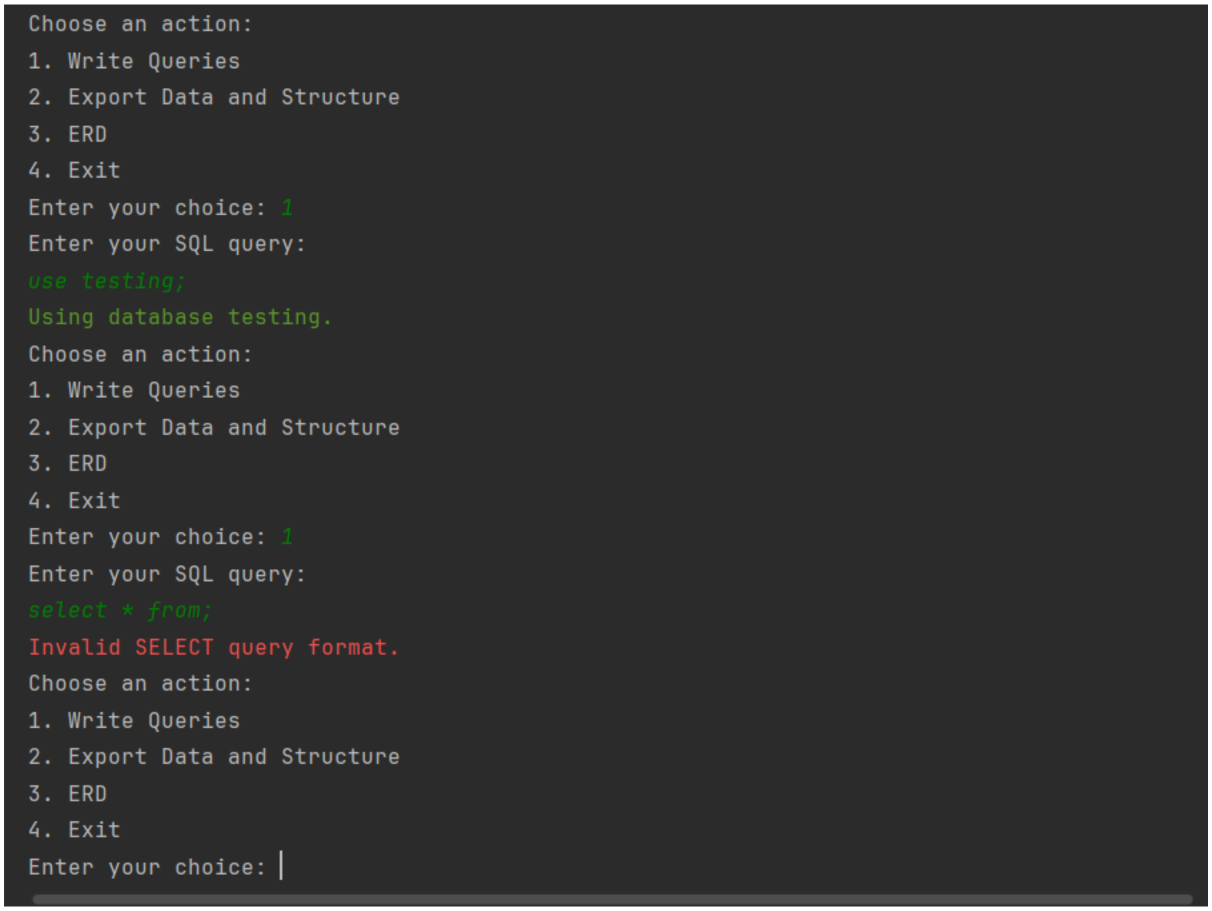
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
delete from student;
All records deleted successfully from table student.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 
```

Figure : Deleting all the records from table



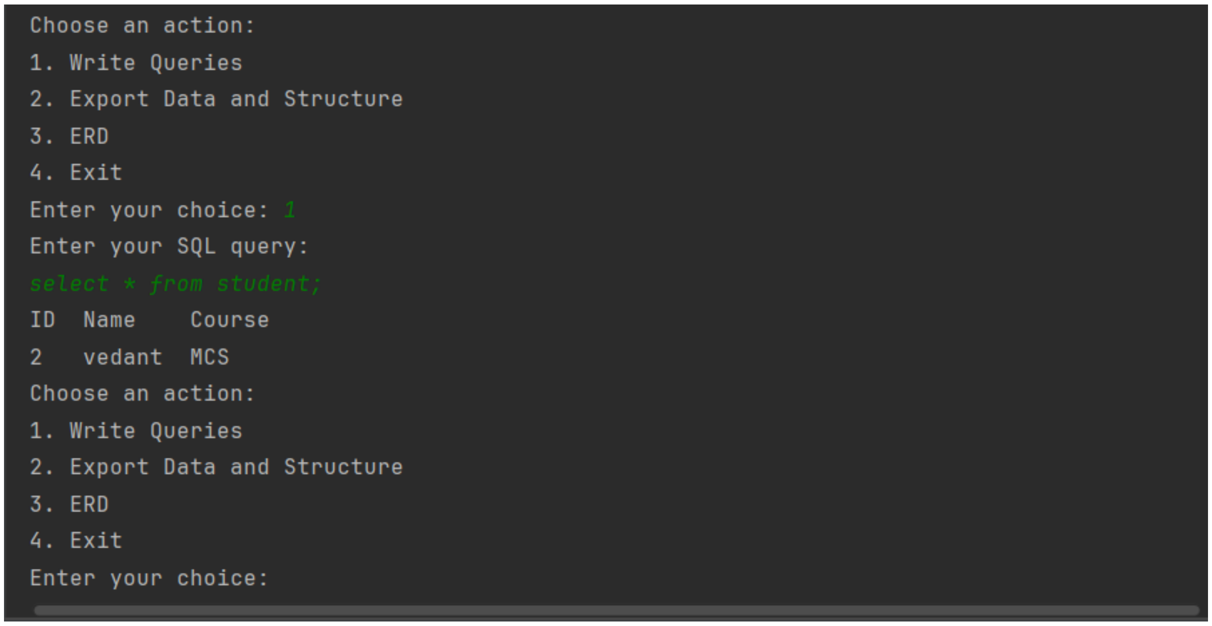
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
select * from student;
No database selected. Use the USE DATABASE command first.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: Deleting without selecting the database



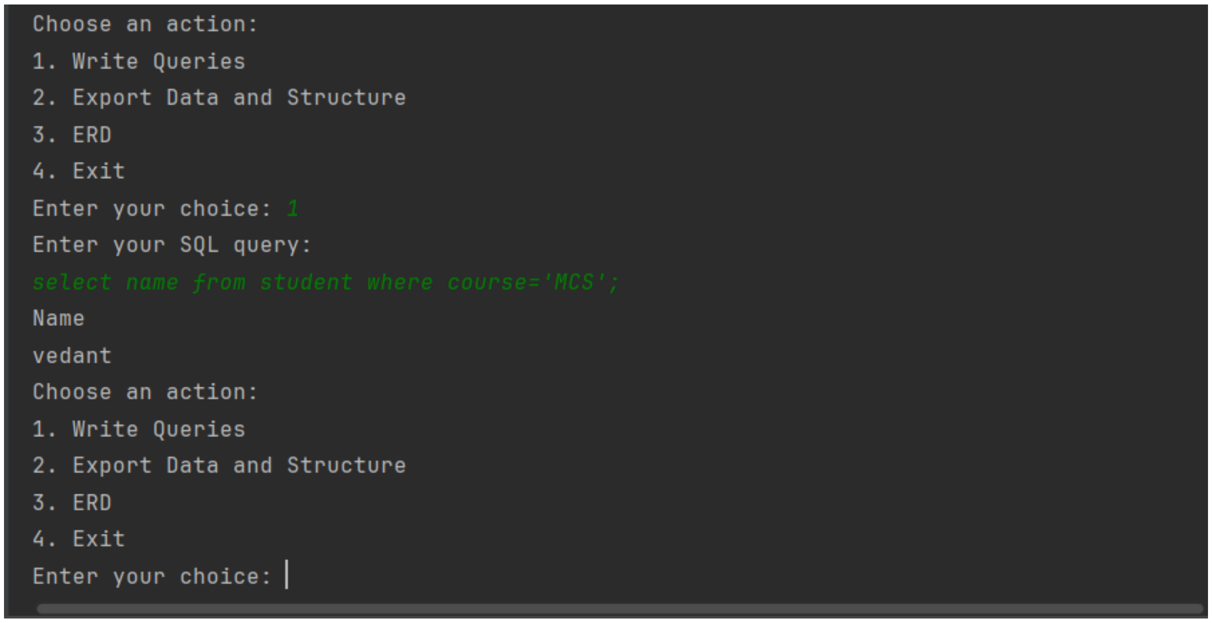
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
use testing;
Using database testing.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
select * from;
Invalid SELECT query format.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: Invalid delete query



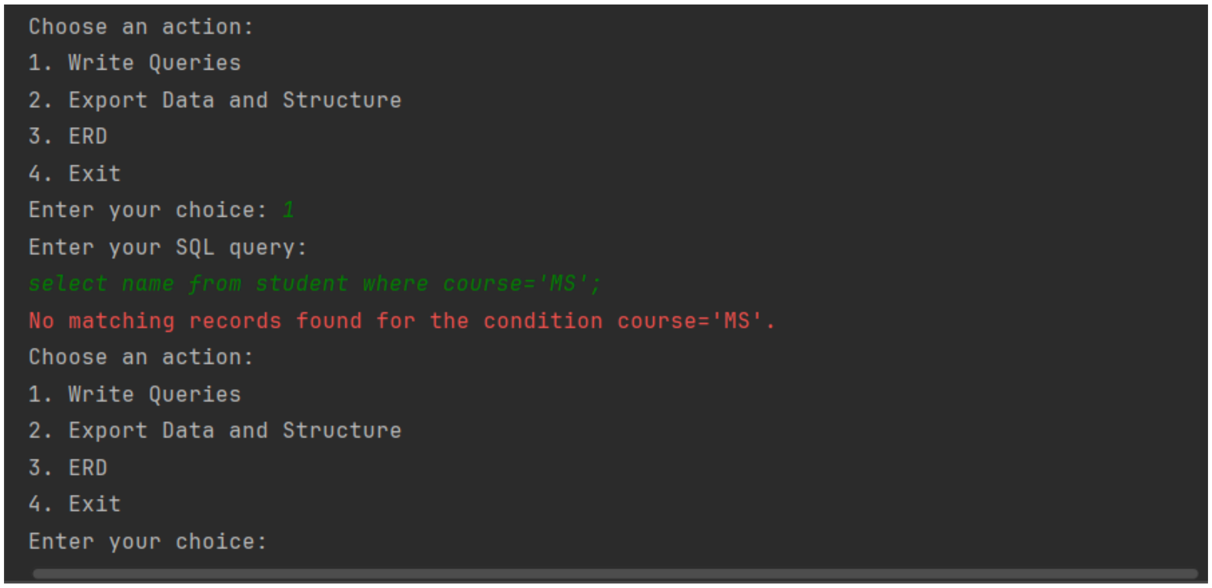
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
select * from student;
ID  Name    Course
2   vedant  MCS
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: obtaining data from student with valid select query



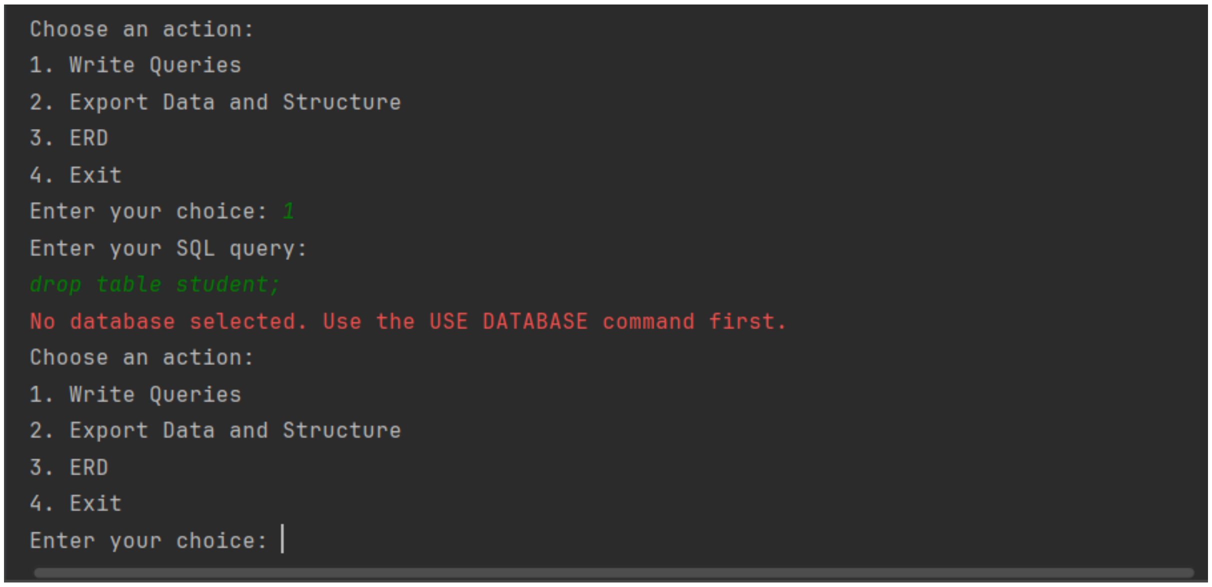
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
select name from student where course='MCS';
Name
vedant
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: Selecting from student with where clause



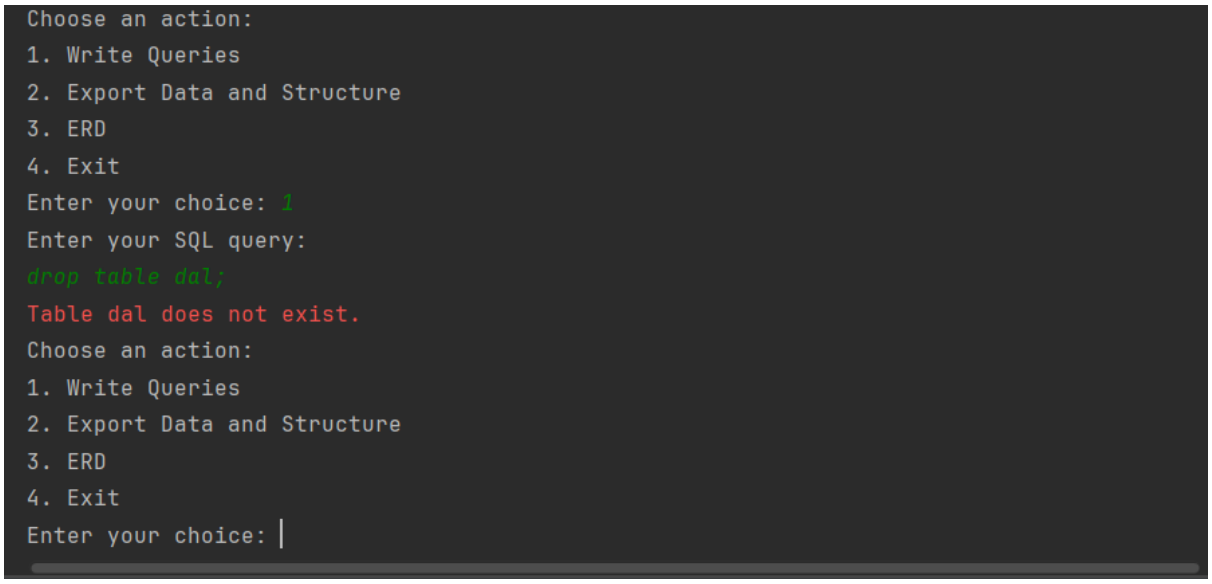
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
select name from student where course='MS';
No matching records found for the condition course='MS'.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice:
```

Figure: selecting the record that does not present



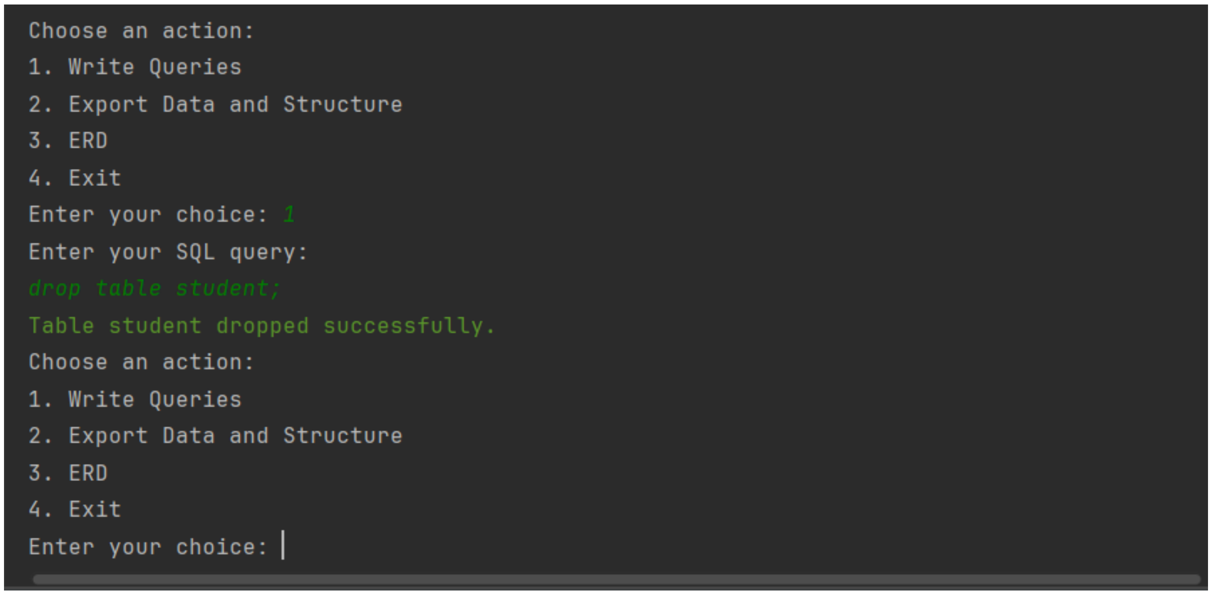
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
drop table student;
No database selected. Use the USE DATABASE command first.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 
```

Figure: Dropping the table without selecting the database



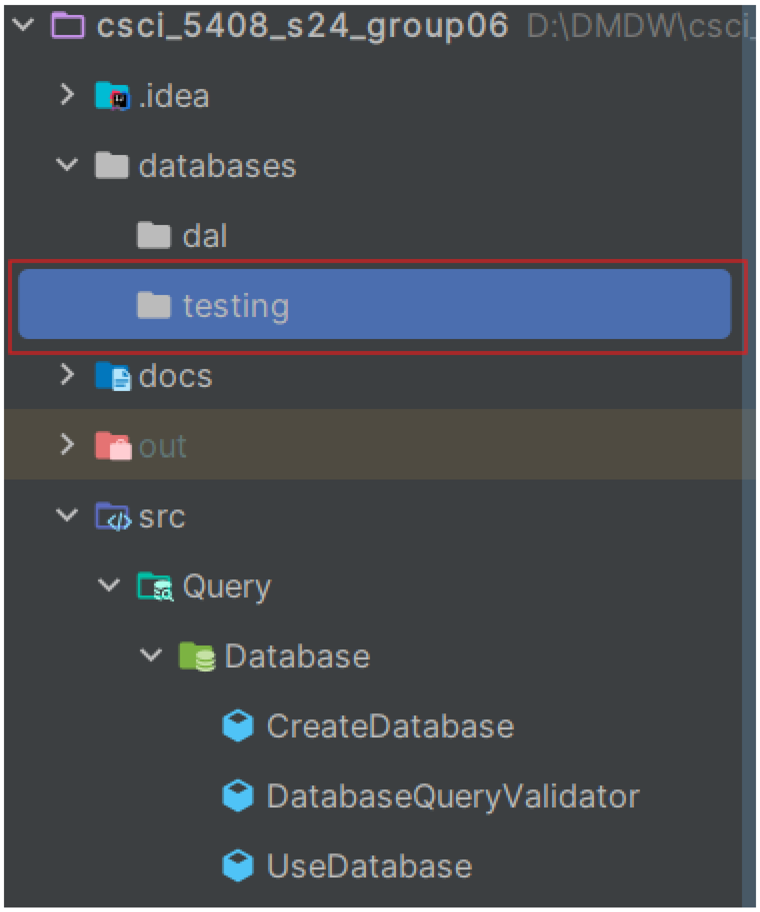
```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
drop table dal;
Table dal does not exist.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 
```

Figure: Dropping the table that does not exists



```
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: 1
Enter your SQL query:
drop table student;
Table student dropped successfully.
Choose an action:
1. Write Queries
2. Export Data and Structure
3. ERD
4. Exit
Enter your choice: |
```

Figure: Dropping the table with right query



```
csci_5408_s24_group06 D:\DMDW\csci_
  .idea
  databases
    dal
    testing
  docs
  out
  src
    Query
      Database
        CreateDatabase
        DatabaseQueryValidator
        UseDatabase
```

Figure: student table dropped from testing

References

[1] "Unified Modeling Language (UML) Class Diagrams - GeeksforGeeks." Accessed June 29, 2024. [Online]. Available: https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/

[2] "UML Class Diagram." Lucidchart. Accessed June 29, 2024. [Online]. Available: https://www.lucidchart.com/pages/uml-class-diagram

[3] "How Regex Works in SQL." Atlassian. Accessed June 29, 2024. [Online]. Available: https://www.atlassian.com/data/sql/how-regex-works-in-sql

[4] "Regular Expressions in SQL." Codedamn. Accessed June 29, 2024. [Online]. Available: https://codedamn.com/news/sql/regular-expressions-in-sql

[5] "Software Ideas." Software Ideas. Accessed June 29, 2024. [Online]. Available: https://www.softwareideas.net/

[6] "File Operations in Java." Javatpoint. Accessed June 29, 2024. [Online]. Available: https://www.javatpoint.com/file-operations-in-java

[7] "MessageDigest in Java." Javatpoint. Accessed June 29, 2024. [Online]. Available: https://www.javatpoint.com/messagedigest-in-java#:~:text=MessageDigest%20is%20the%20returned%20value,values%20into%20compressed%20numerical%20values

[8] "File mkdir Method in Java with Examples." GeeksforGeeks. Accessed June 29, 2024. [Online]. Available: https://www.geeksforgeeks.org/file-mkdir-method-in-java-with-examples/