CSCI 5408
Data Management and Warehousing

Sprint Report - 2

Group members:

Kenee Ashok Patel (B00969805)

Vraj Sunilkumar Shah (B00979965)

Shivani Uppe (B00976573)

GitLab Project Link: https://git.cs.dal.ca/kenee/dbms-builder-11

Table of Contents

Table of Figures

1. Pseudocode

1.1. Log Management

1.1.1. Log Manager

```
Class LogManager:

  CONSTANTS:
    GENERAL_LOGS_FILE = "Databases/general_logs.json"
    EVENT_LOGS_FILE = "Databases/event_logs.json"
    QUERY_LOGS_FILE = "Databases/query_logs.json"

  METHOD logGeneral(action, executionTime, dbState):
    Create JSON object logEntry
    Set "action" in logEntry to action
    Set "executionTime" in logEntry to executionTime
    Set "dbState" in logEntry to dbState
    Call appendToFile with GENERAL_LOGS_FILE and logEntry

  METHOD logEvent(eventType, details, timestamp):
    Create JSON object logEntry
    Set "eventType" in logEntry to eventType
    Set "details" in logEntry to details
    Set "timestamp" in logEntry to timestamp
    Call appendToFile with EVENT_LOGS_FILE and logEntry

  METHOD logQuery(query, executionTime, timestamp):
    Create JSON object logEntry
    Set "query" in logEntry to query
    Set "execution_time" in logEntry to executionTime
    Set "timestamp" in logEntry to timestamp
    Call appendToFile with QUERY_LOGS_FILE and logEntry

  METHOD appendToFile(filePath, logEntry):
    TRY:
      Open file at filePath in append mode
      Write logEntry to file
      Close file
    CATCH IOException:
        Print stack trace
```

1.2. Export structure and value

1.2.1. Generate SQL Dump

```
function generateSQLDump(dbName):
    // Initialize an empty list for the SQL dump
    List<String> sqlDump = new ArrayList<>()

    // Add CREATE DATABASE and USE DATABASE statements
    sqlDump.add("CREATE DATABASE " + dbName + ";")
    sqlDump.add("USE " + dbName + ";")

    // Get the directory containing the database tables
    File dbDirectory = new File(DATABASES_DIRECTORY + File.separator + dbName)
    File[] tableFiles = dbDirectory.listFiles((dir, name) -> name.endsWith(".txt"))

    if (tableFiles is not null) then
        for each tableFile in tableFiles do
            // Get table name by removing ".txt" extension
            String tableName = tableFile.getName().replace(".txt", "")

            // Get CREATE TABLE SQL statement
            String createTableSQL = getCreateTableSQLQuery(dbName, tableName)
            sqlDump.add(createTableSQL)
            // Get rows of the table
            List<Map<String, String>> rows = getRows(dbName, tableName)
            for each row in rows do
                // Get INSERT INTO TABLE SQL statement for each row
                String insertRowSQL = getInsertRowSQLQuery(tableName, row)
                sqlDump.add(insertRowSQL)
            end for
        end for
    end if

    // Build the content of the dump file
    StringBuilder dumpContent = new StringBuilder()
    for each sql in sqlDump do
        dumpContent.append(sql).append(System.lineSeparator())
    end for

    // Write the dump content to a file
    String sqlDumpFilePath = "Databases/" + dbName + "_dump.sql"
    writeToFile(sqlDumpFilePath, dumpContent.toString())
end function
```

1.2.2. Get CREATE TABLE SQL Query

```
function getCreateTableSQLQuery(dbName, tableName) returns String:

    // Get the columns of the table
    List<Column> columns = getTableColumns(dbName, tableName)
    StringBuilder sql = new StringBuilder("CREATE TABLE ").append(tableName).append(" (")

    for i = 0 to columns.size() - 1 do
```

```
        Column column = columns.get(i)
        sql.append(column.name()).append(" ").append(column.type())

        for each constraint in column.constraints() do
            sql.append(" ").append(constraint)
        end for

        if (i < columns.size() - 1) then
            sql.append(", ")
        end if
    end for

    sql.append(");")
    return sql.toString()
end function
```

1.2.3. Get INSERT INTO TABLE SQL Query

```
function getInsertRowSQLQuery(tableName, row) returns String:

    StringBuilder sql = new StringBuilder("INSERT INTO ").append(tableName).append(" (")
    StringBuilder values = new StringBuilder(" VALUES (")

    int i = 0
    for each entry in row.entrySet() do
        sql.append(entry.getKey())
        values.append("'").append(entry.getValue()).append("'")
        if (i < row.size() - 1) then
            sql.append(", ")
            values.append(", ")
        end if

        i++
    end for

    sql.append(")").append(values).append(");")
    return sql.toString()
end function
```

1.2.4. Write content to a file

```
function writeToFile(filePath, content):

    File file = new File(filePath)

    try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) then
        writer.write(content)
    catch IOException e then
        print "Failed to write to file: " + filePath + " - " + e.getMessage()
    end try
end function
```

1.3. User interface and Login security

1.3.1. Showing landing menu

```
function showLandingMenu():
    boolean shouldShowMainMenu = false
```

```
   LandingMenuOption landingMenuOption = printLandingMenuAndGetSelectedOption()


   when (landingMenuOption)
      is Register: shouldShowMainMenu = registerUser()
      is Login: shouldShowMainMenu = loginUser()
      is Exit: System.exit(0)


   if  (shouldShowMainMenu) then showMainMenu()
   else showLandingMenu()


// print landing menu and get selected option
function printLandingMenuAndGetSelectedOption() returns LandingMenuOption:
   print landing menu options
   print "Select an option between 1 and 3: "


   string selectedMenuOptionString = scanner.nextLine()
   if (isMenuOptionInvalid(selectedMenuOptionString, LandingMenuOption.length))
         then
               print "Oops wrong input provided, please try again.\n"
               return printLandingMenuAndGetSelectedOption()
   return LandingMenuOption.values()[Integer.parseInt(selectedMenuOptionString) - 1]


// check if menu option is invalid
function isMenuOptionInvalid(string selectedMenuOptionString, int totalOptionCount) returns
boolean
   if (selectedMenuOptionString is null or selectedMenuOptionString is blank)
         then return true
```

```
try {
    int selectedMenuOption = Integer.parseInt(selectedMenuOptionString)
    return (selectedMenuOption <= 0 or selectedMenuOption > totalOptionCount)
} catch (NumberFormatException e) {
    return true
}
```

1.3.2. Register

```
function registerUser() returns boolean:
    // get user ID
    print "Enter UserID: "
    string userId = scanner.nextLine()
    if (userId is blank or isUserRegistered)
        then print error message and return false

    // get password
    print "Enter Password: "
    string password = scanner.nextLine()
    if (password is blank)
        then print error message and return false

    // get security question
    print "Enter Security Question: "
    string securityQuestion = scanner.nextLine()
    if (securityQuestion is blank)
        then print error message and return false
    // get security answer
    print "Enter Security Answer: "
    string securityAnswer = scanner.nextLine()
    if (securityAnswer is blank) {
        then print error message and return false

    // register user
    boolean isUserRegistered = userAuthService.registerUser(userId, password, securityQuestion,
securityAnswer)
    print registration status message
    return isUserRegistered
```

1.3.3. Login

```
function loginUser() returns boolean:
    // get user ID
    print "Enter UserID: "
    string userId = scanner.nextLine()
    if (userId is blank or !userAuthService.validateUserIdToLogin(userId))
        then print error message and return false

    // get password
    print "Enter Password: "
    string password = scanner.nextLine()
    if (password is blank)
        then print error message and return false

    // validate password and get security question
    string securityQuestion =
userAuthService.validatePasswordAndGetSecurityQuestion(password)
    if (securityQuestion is null)
        then print error message and return false

    // get security answer
    print "Please answer this question: " + securityQuestion
    string securityAnswer = scanner.nextLine()
    boolean isSecurityAnswerValid = userAuthService.validateSecurityAnswer(securityAnswer)
    if (!isSecurityAnswerValid)
        print error message and return false

    return true
```

1.3.4. Show main menu

```
function showMainMenu():
    MainMenuOption mainMenuOption = printMainMenuAndGetSelectedOption()

    when (mainMenuOption)
        is WriteQueries: startAcceptingQueries()
        is ExportStructureAndValue: exportStructureAndValue()
        is Erd: print "Currently, generating ERD is not supported"
        is Exit: System.exit(0)

    showMainMenu()

// print main menu and get selected option
```

```
function printMainMenuAndGetSelectedOption() returns MainMenuOption:
    print main menu options
    print "Select an option between 1 and 4: "


    string selectedMenuOptionString = scanner.nextLine()
    if (isMenuOptionInvalid(selectedMenuOptionString, MainMenuOption.values().length))
        then
            print "Oops wrong input provided, please try again.\n"
            return printMainMenuAndGetSelectedOption()
    return MainMenuOption.values()[Integer.parseInt(selectedMenuOptionString) - 1]


// check if menu option is invalid
function isMenuOptionInvalid(string selectedMenuOptionString, int totalOptionCount) returns
boolean
    if (selectedMenuOptionString is null or selectedMenuOptionString is blank)
        then return true
    try {
        int selectedMenuOption = Integer.parseInt(selectedMenuOptionString)
        return (selectedMenuOption <= 0 or selectedMenuOption > totalOptionCount)
    } catch (NumberFormatException e) {
        return true
    }
```

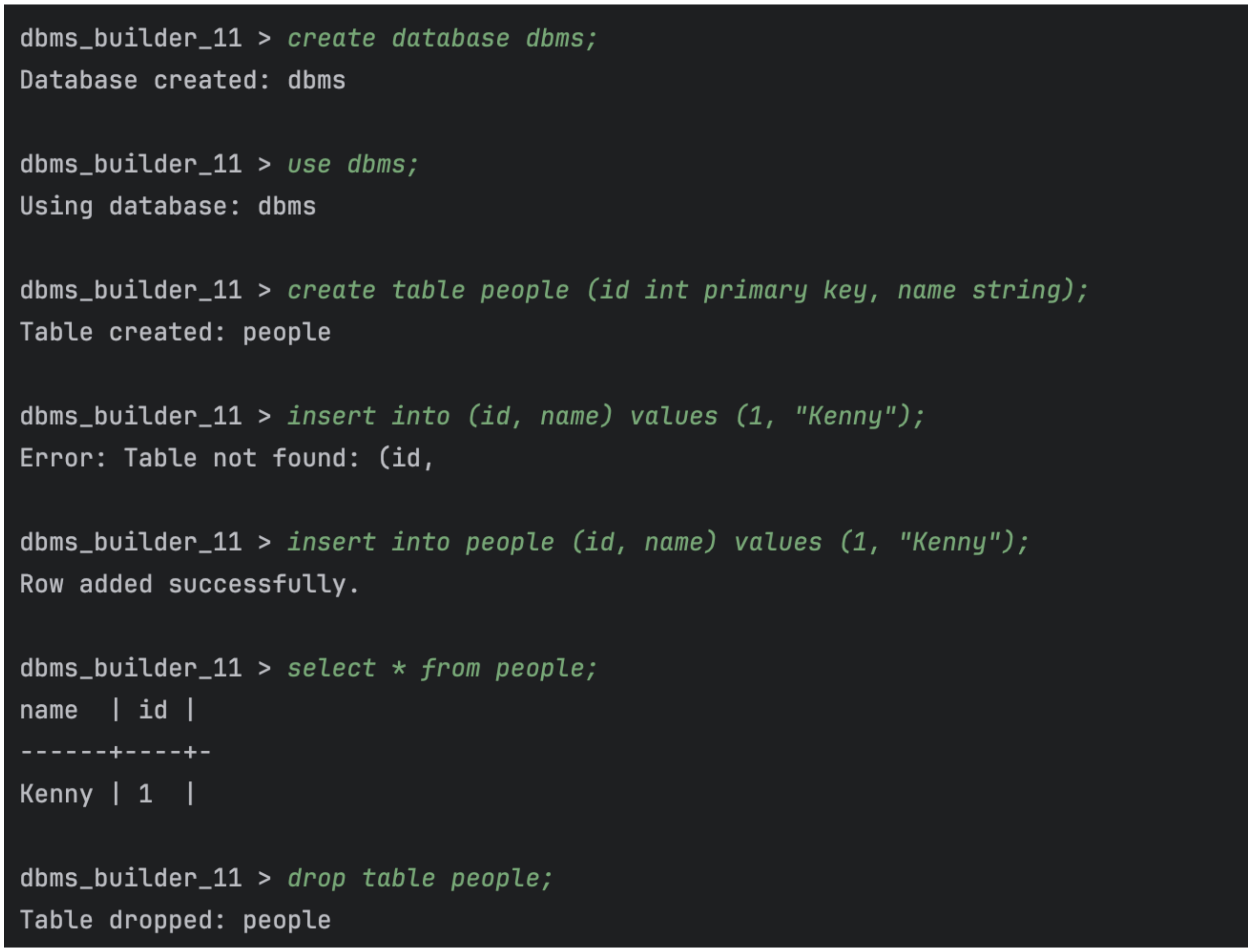
2. Git code repository link

Link: https://git.cs.dal.ca/kenee/dbms-builder-11

3. Test cases and evidence of testing

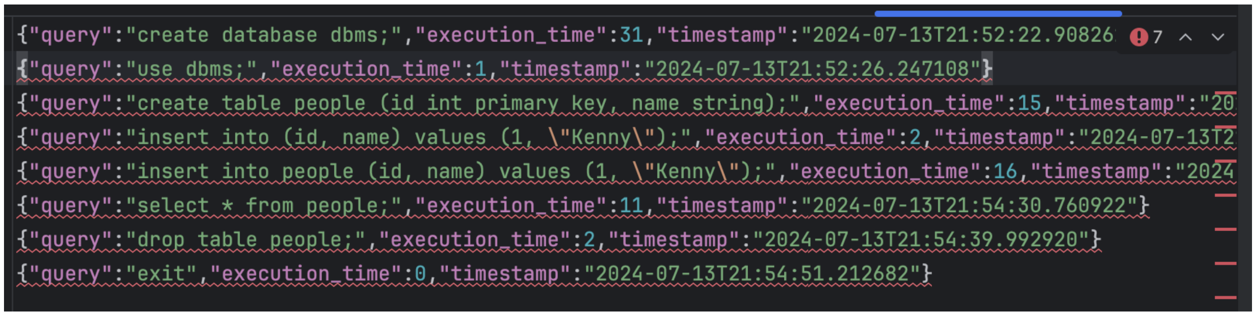
3.1. Log Management

Logging queries



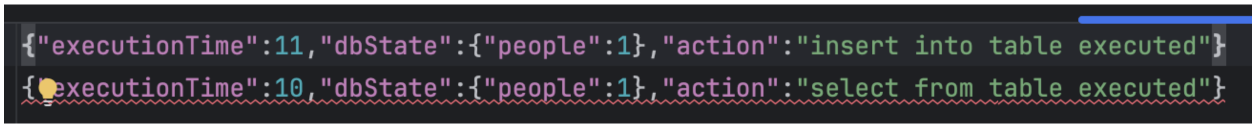
```
dbms_builder_11 > create database dbms;
Database created: dbms

dbms_builder_11 > use dbms;
Using database: dbms

dbms_builder_11 > create table people (id int primary key, name string);
Table created: people

dbms_builder_11 > insert into (id, name) values (1, "Kenny");
Error: Table not found: (id,

dbms_builder_11 > insert into people (id, name) values (1, "Kenny");
Row added successfully.

dbms_builder_11 > select * from people;
name  | id |
------+----+-
Kenny | 1  |

dbms_builder_11 > drop table people;
Table dropped: people
```

Figure 1 Queries ran to generate logs



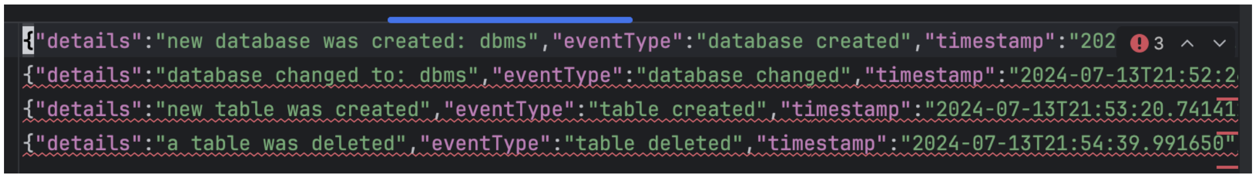
```
{"query":"create database dbms;","execution_time":31,"timestamp":"2024-07-13T21:52:22.90826
{"query":"use dbms;","execution_time":1,"timestamp":"2024-07-13T21:52:26.247108"}
{"query":"create table people (id int primary key, name string);","execution_time":15,"timestamp":"20
{"query":"insert into (id, name) values (1, \"Kenny\");","execution_time":2,"timestamp":"2024-07-13T2
{"query":"insert into people (id, name) values (1, \"Kenny\");","execution_time":16,"timestamp":"2024
{"query":"select * from people;","execution_time":11,"timestamp":"2024-07-13T21:54:30.760922"}
{"query":"drop table people;","execution_time":2,"timestamp":"2024-07-13T21:54:39.992920"}
{"query":"exit","execution_time":0,"timestamp":"2024-07-13T21:54:51.212682"}
```

Figure 2 Content that was filled in query_logs.json



```
{"executionTime":11,"dbState":{"people":1},"action":"insert into table executed"}
{"executionTime":10,"dbState":{"people":1},"action":"select from table executed"}
```

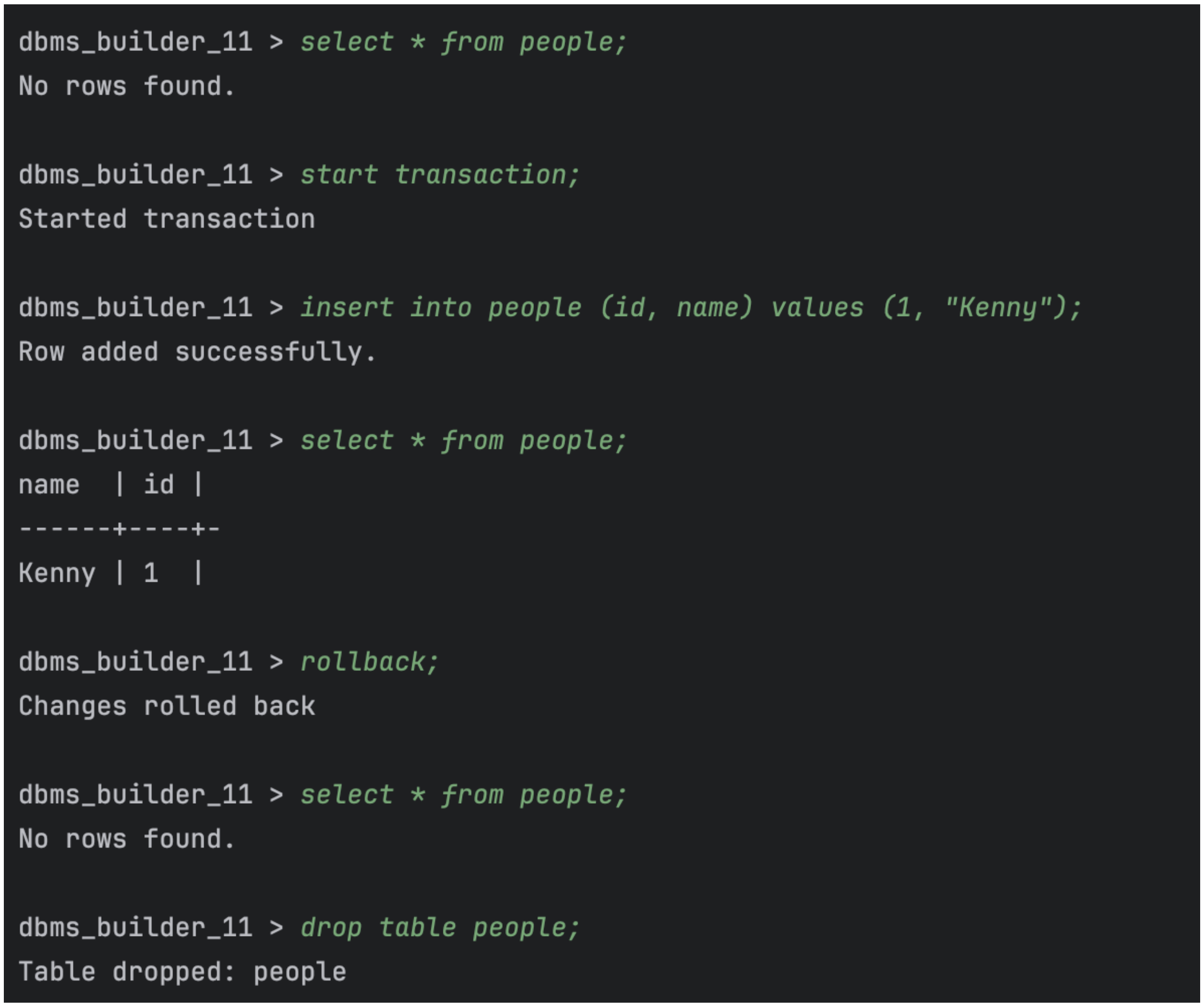
Figure 3 Content that was filled in general_logs.json



```
{"details":"new database was created: dbms","eventType":"database created","timestamp":"202
{"details":"database changed to: dbms","eventType":"database changed","timestamp":"2024-07-13T21:52:2
{"details":"new table was created","eventType":"table created","timestamp":"2024-07-13T21:53:20.74141
{"details":"a table was deleted","eventType":"table deleted","timestamp":"2024-07-13T21:54:39.991650"
```

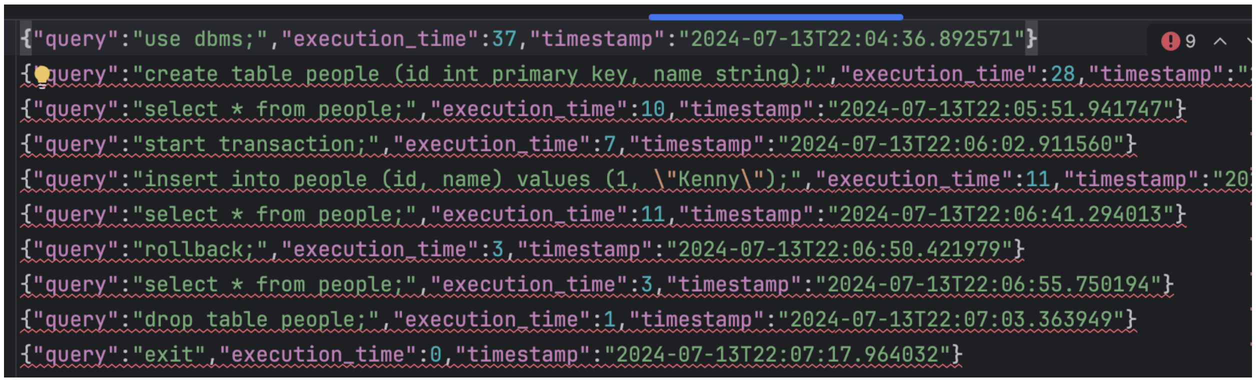
Figure 4 Content that was filled in event_logs.json

Logging of Transactional queries



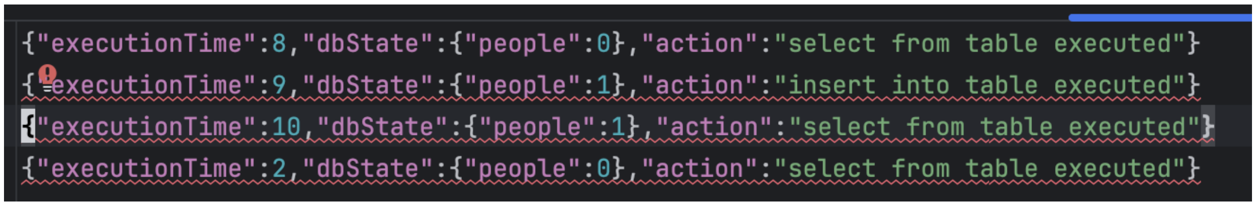
```
dbms_builder_11 > select * from people;
No rows found.

dbms_builder_11 > start transaction;
Started transaction

dbms_builder_11 > insert into people (id, name) values (1, "Kenny");
Row added successfully.

dbms_builder_11 > select * from people;
name  | id |
------+----+-
Kenny | 1  |

dbms_builder_11 > rollback;
Changes rolled back

dbms_builder_11 > select * from people;
No rows found.

dbms_builder_11 > drop table people;
Table dropped: people
```

Figure 5 Queries ran for transactional logging



```
{"query":"use dbms;","execution_time":37,"timestamp":"2024-07-13T22:04:36.892571"}
{"query":"create table people (id int primary key, name string);","execution_time":28,"timestamp":"
{"query":"select * from people;","execution_time":10,"timestamp":"2024-07-13T22:05:51.941747"}
{"query":"start transaction;","execution_time":7,"timestamp":"2024-07-13T22:06:02.911560"}
{"query":"insert into people (id, name) values (1, \"Kenny\");","execution_time":11,"timestamp":"20
{"query":"select * from people;","execution_time":11,"timestamp":"2024-07-13T22:06:41.294013"}
{"query":"rollback;","execution_time":3,"timestamp":"2024-07-13T22:06:50.421979"}
{"query":"select * from people;","execution_time":3,"timestamp":"2024-07-13T22:06:55.750194"}
{"query":"drop table people;","execution_time":1,"timestamp":"2024-07-13T22:07:03.363949"}
{"query":"exit","execution_time":0,"timestamp":"2024-07-13T22:07:17.964032"}
```

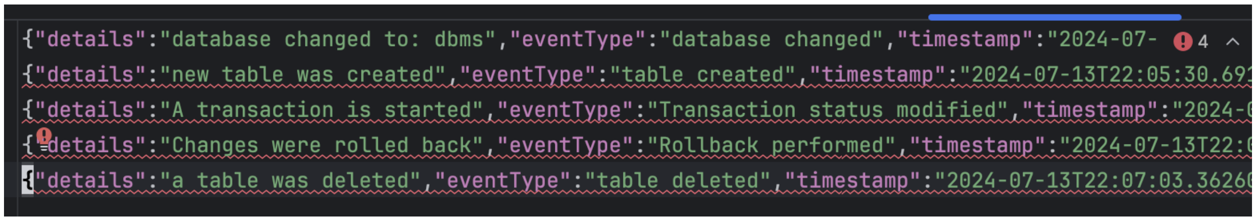
Figure 6 query_logs.json after running the transactional queries



```
{"executionTime":8,"dbState":{"people":0},"action":"select from table executed"}
{"executionTime":9,"dbState":{"people":1},"action":"insert into table executed"}
{"executionTime":10,"dbState":{"people":1},"action":"select from table executed"}
{"executionTime":2,"dbState":{"people":0},"action":"select from table executed"}
```

Figure 7 general_logs.json after running the transactional queries

You can notice the number of records in people table, which is 0 after rollback was performed, indicating reflection of transaction rollback in the number of records.

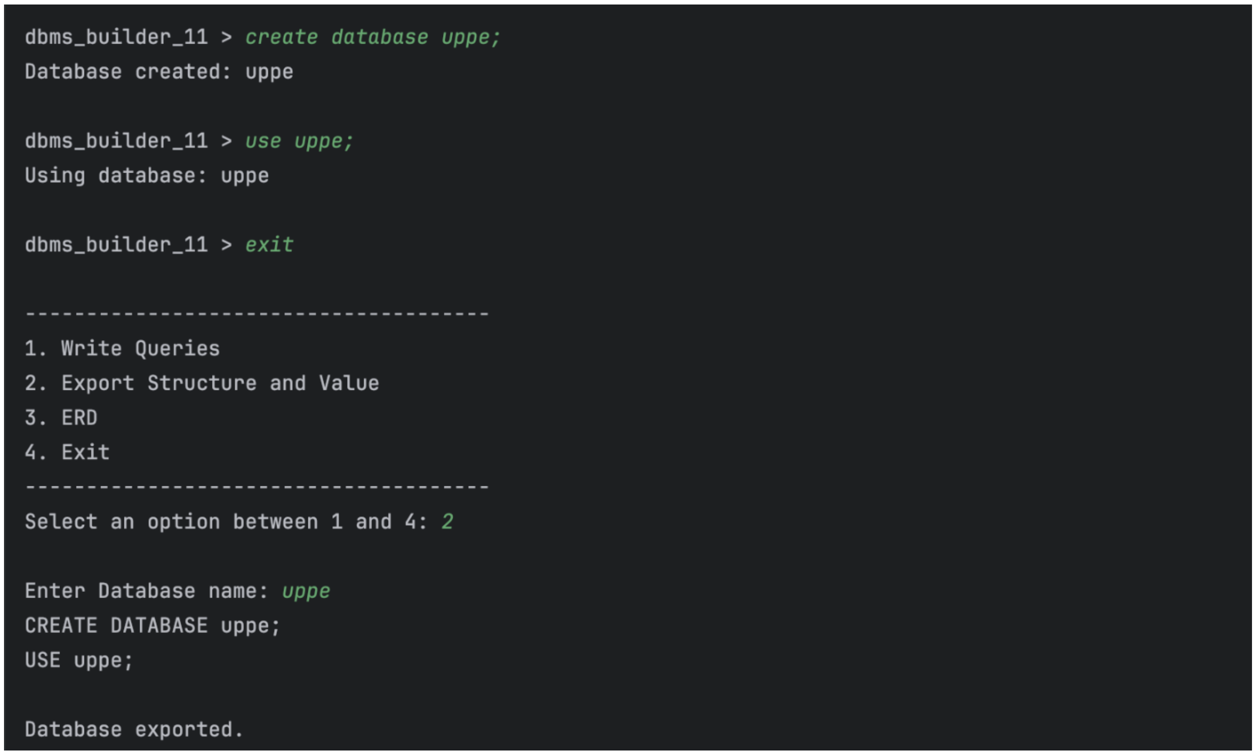


```
{"details":"database changed to: dbms","eventType":"database changed","timestamp":"2024-07-
{"details":"new table was created","eventType":"table created","timestamp":"2024-07-13T22:05:30.69
{"details":"A transaction is started","eventType":"Transaction status modified","timestamp":"2024-
{"details":"Changes were rolled back","eventType":"Rollback performed","timestamp":"2024-07-13T22:
{"details":"a table was deleted","eventType":"table deleted","timestamp":"2024-07-13T22:07:03.3626
```

Figure 8 event_logs.json after running the transactional queries

3.2. Export structure and value

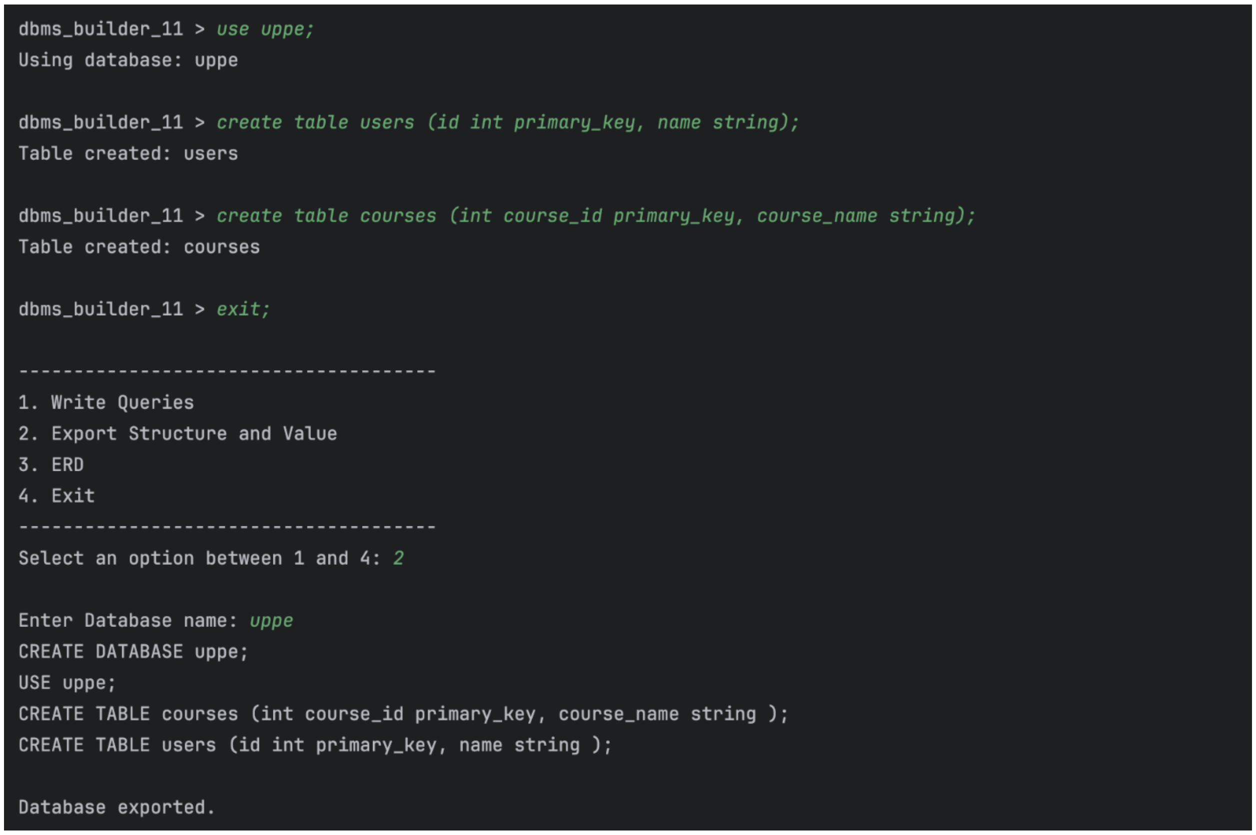
Exporting an empty database



```
dbms_builder_11 > create database uppe;
Database created: uppe

dbms_builder_11 > use uppe;
Using database: uppe

dbms_builder_11 > exit

--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: 2

Enter Database name: uppe
CREATE DATABASE uppe;
USE uppe;

Database exported.
```

Figure 9: Exporting an empty database.

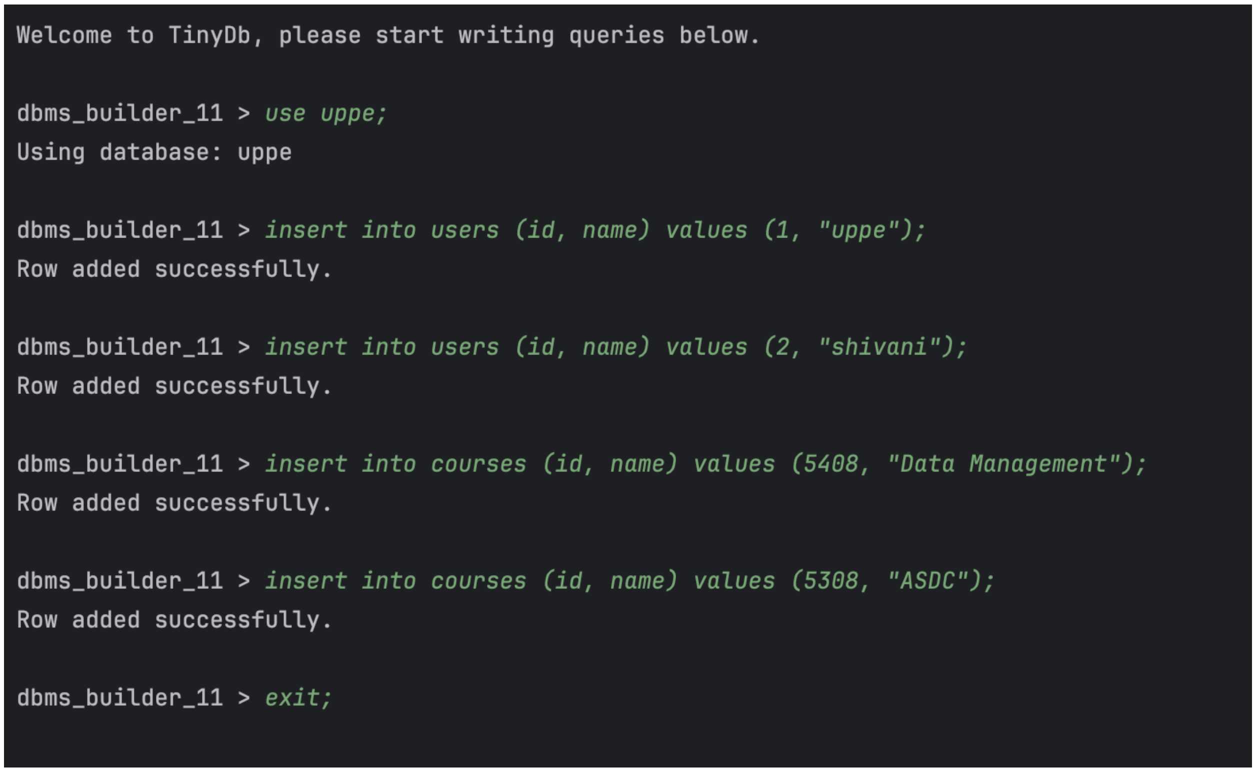
Exporting a database which has empty tables



```
dbms_builder_11 > use uppe;
Using database: uppe

dbms_builder_11 > create table users (id int primary_key, name string);
Table created: users

dbms_builder_11 > create table courses (int course_id primary_key, course_name string);
Table created: courses

dbms_builder_11 > exit;

--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: 2

Enter Database name: uppe
CREATE DATABASE uppe;
USE uppe;
CREATE TABLE courses (int course_id primary_key, course_name string );
CREATE TABLE users (id int primary_key, name string );

Database exported.
```

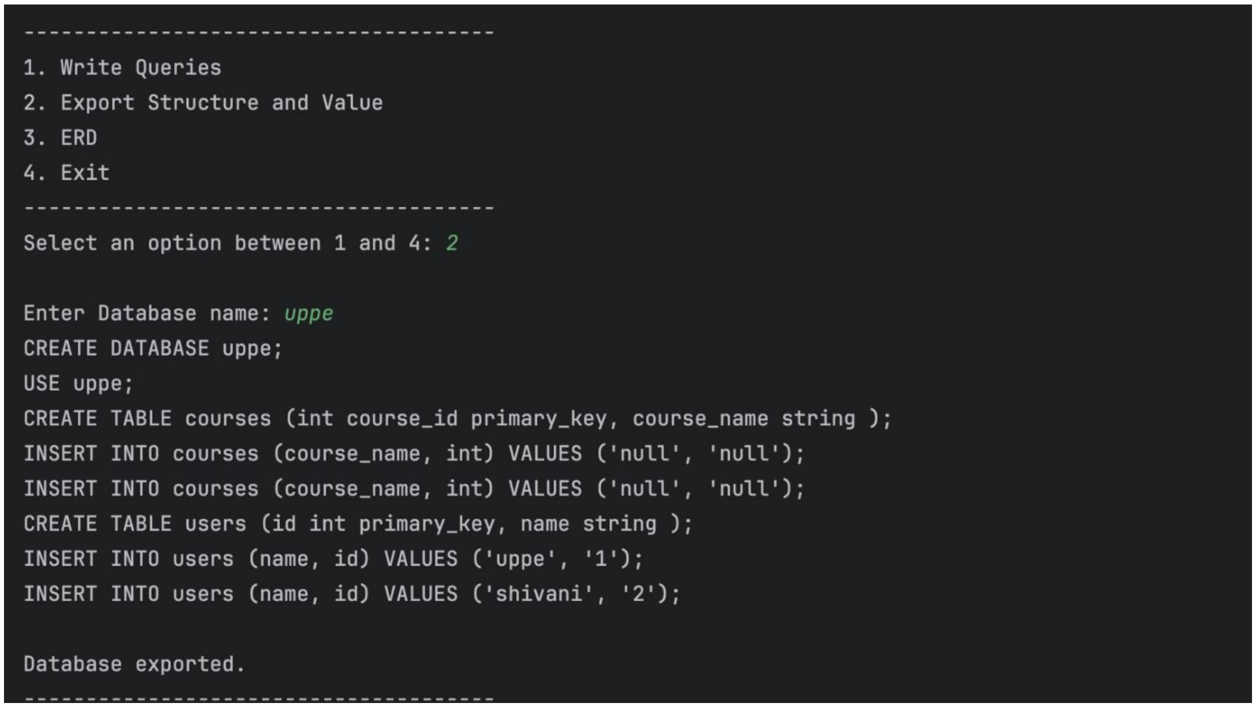
Figure 10: Exporting a database which has empty tables.

Exporting a database which has tables and rows in the tables



```
Welcome to TinyDb, please start writing queries below.

dbms_builder_11 > use uppe;
Using database: uppe

dbms_builder_11 > insert into users (id, name) values (1, "uppe");
Row added successfully.

dbms_builder_11 > insert into users (id, name) values (2, "shivani");
Row added successfully.

dbms_builder_11 > insert into courses (id, name) values (5408, "Data Management");
Row added successfully.

dbms_builder_11 > insert into courses (id, name) values (5308, "ASDC");
Row added successfully.

dbms_builder_11 > exit;
```

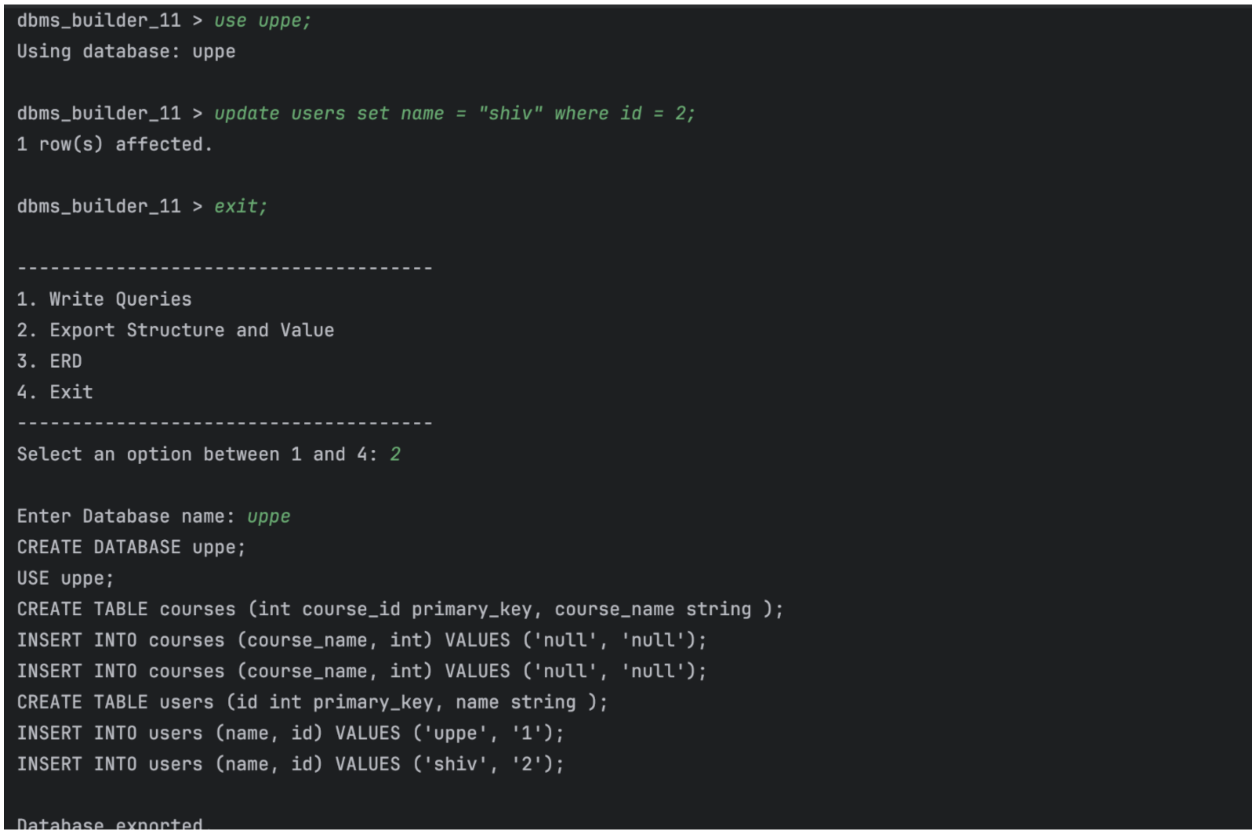
Figure 11: Inserting values into tables



```
-------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
-------------------------------------
Select an option between 1 and 4: 2

Enter Database name: uppe
CREATE DATABASE uppe;
USE uppe;
CREATE TABLE courses (int course_id primary_key, course_name string );
INSERT INTO courses (course_name, int) VALUES ('null', 'null');
INSERT INTO courses (course_name, int) VALUES ('null', 'null');
CREATE TABLE users (id int primary_key, name string );
INSERT INTO users (name, id) VALUES ('uppe', '1');
INSERT INTO users (name, id) VALUES ('shivani', '2');

Database exported.
-------------------------------------
```

Figure 12: Exporting a database which has tables and rows in the tables.

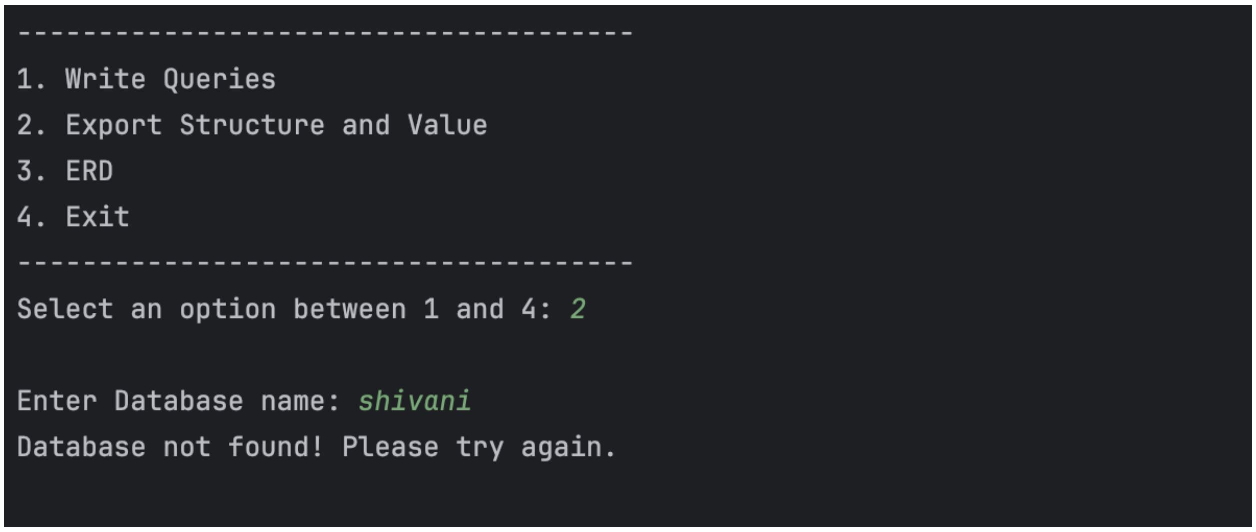
Exporting a database after updating a table



```
dbms_builder_11 > use uppe;
Using database: uppe

dbms_builder_11 > update users set name = "shiv" where id = 2;
1 row(s) affected.

dbms_builder_11 > exit;

--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: 2

Enter Database name: uppe
CREATE DATABASE uppe;
USE uppe;
CREATE TABLE courses (int course_id primary_key, course_name string );
INSERT INTO courses (course_name, int) VALUES ('null', 'null');
INSERT INTO courses (course_name, int) VALUES ('null', 'null');
CREATE TABLE users (id int primary_key, name string );
INSERT INTO users (name, id) VALUES ('uppe', '1');
INSERT INTO users (name, id) VALUES ('shiv', '2');

Database exported
```

Figure 13: Exporting a database after updating a table.

Exporting a database which does not exist



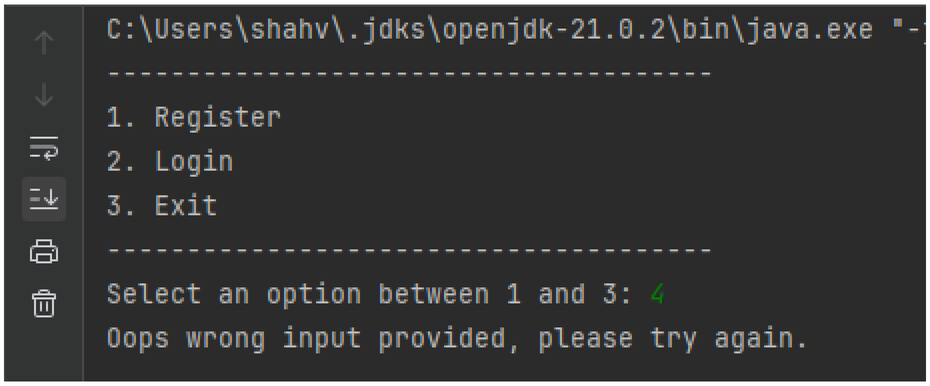
```
--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: 2

Enter Database name: shivani
Database not found! Please try again.
```

Figure 14: Exporting a database which does not exist.

3.3. User interface and Login security

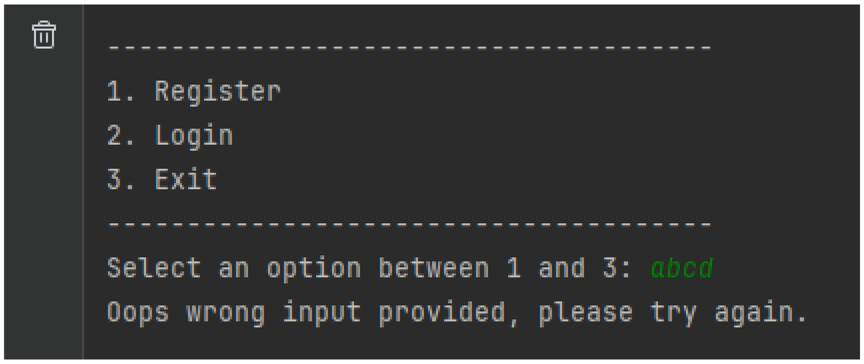
Landing menu

User inputs wrong number for menu selection:



```
C:\Users\shahv\.jdks\openjdk-21.0.2\bin\java.exe "-
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 4
Oops wrong input provided, please try again.
```

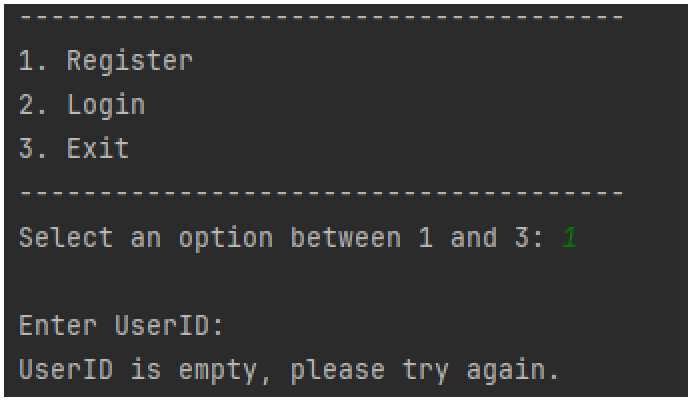
Figure 15: Landing menu when user inputs invalid number - 4



```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: abcd
Oops wrong input provided, please try again.
```

Figure 16: Landing menu when user inputs invalid input "abcd"

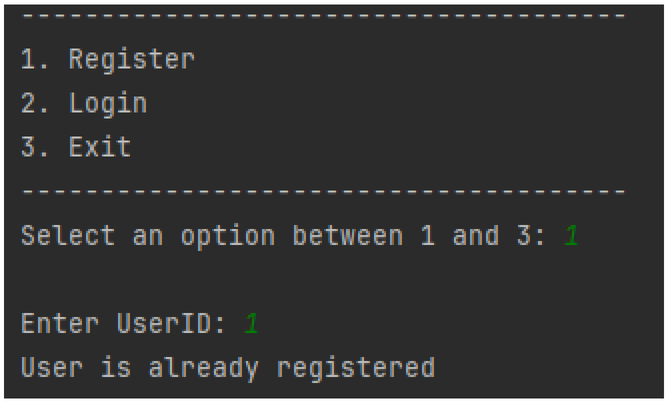
User provides empty user id:



```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 1

Enter UserID:
UserID is empty, please try again.
```

Figure 17: Landing menu when user inputs empty user id

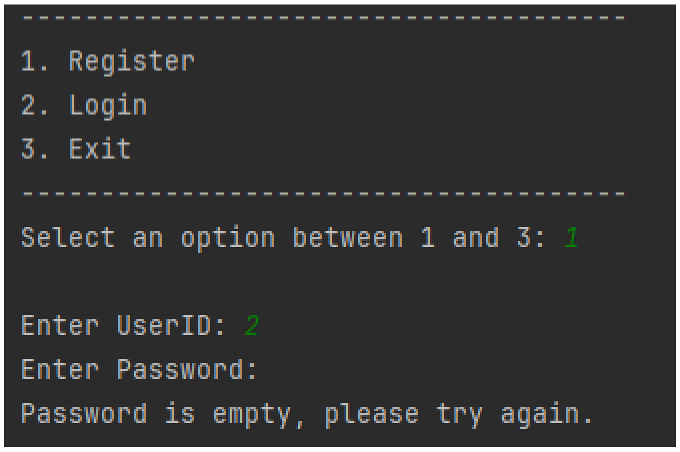
User tries to register with already registered user id:



```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 1

Enter UserID: 1
User is already registered
```

Figure 18: Landing menu when user inputs registered user id

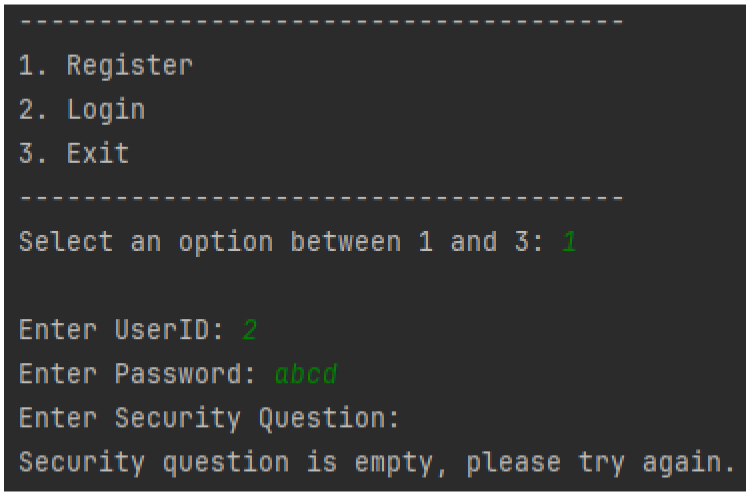
User provides empty password:



```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 1

Enter UserID: 2
Enter Password:
Password is empty, please try again.
```

Figure 19: Landing menu when user inputs empty password

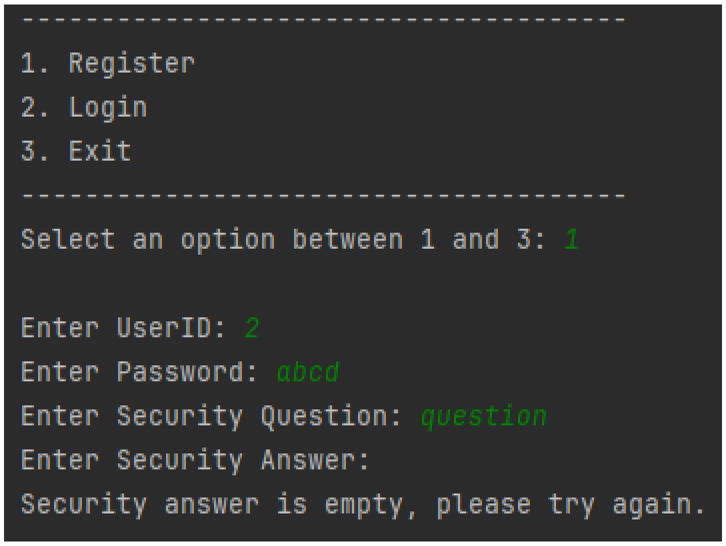
User provides empty security question:



```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 1

Enter UserID: 2
Enter Password: abcd
Enter Security Question:
Security question is empty, please try again.
```

Figure 20: Landing menu when user inputs empty security question

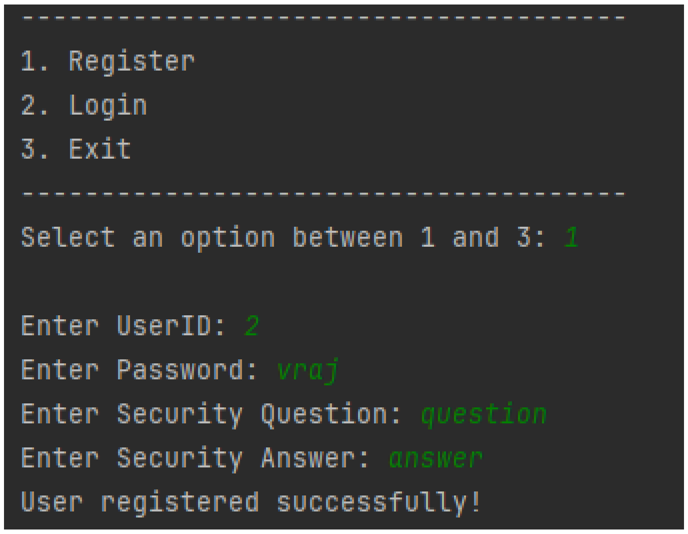
User provides empty security question’s answer:



```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 1

Enter UserID: 2
Enter Password: abcd
Enter Security Question: question
Enter Security Answer:
Security answer is empty, please try again.
```

Figure 21: Landing menu when user inputs empty security question's answer

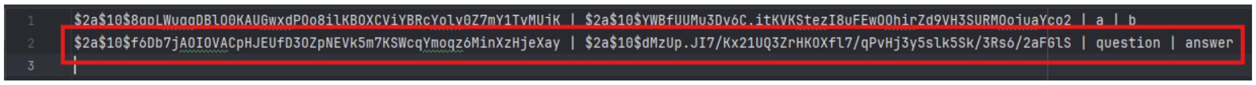
User provides valid credentials:



```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 1

Enter UserID: 2
Enter Password: vraj
Enter Security Question: question
Enter Security Answer: answer
User registered successfully!
```

Figure 22: Successful registration from landing menu

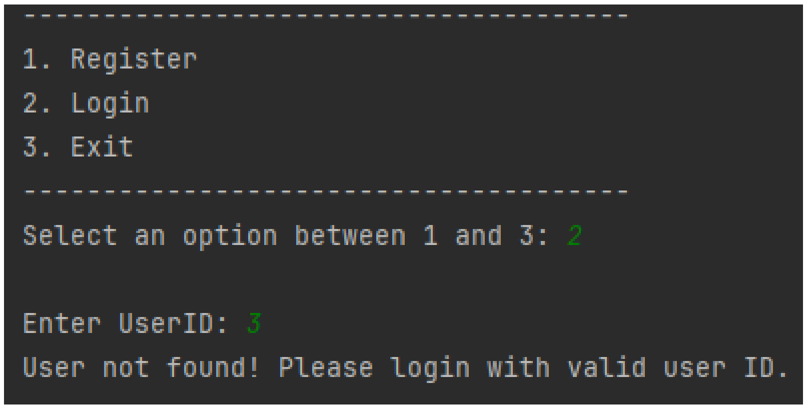
User_profile.txt file after registration



```
$2a$10$8apLWuqaDBlO0KAUGwxdPOo8ilKBOXCViYBRcYolv0Z7mY1TvMUjK | $2a$10$YWBfUUMu3Dv6C.itKVKStezI8uFEwOOhirZd9VH3SURMOojuaYco2 | a | b
$2a$10$f6Db7jAOIOVACpHJEUfD3OZpNEVk5m7KSWcqYmoqz6MinXzHjeXay | $2a$10$dMzUp.JI7/Kx21UQ3ZrHKOXfl7/qPvHj3y5slk5Sk/3Rs6/2aFGlS | question | answer
```

Figure 23: User profile text file after successful registration

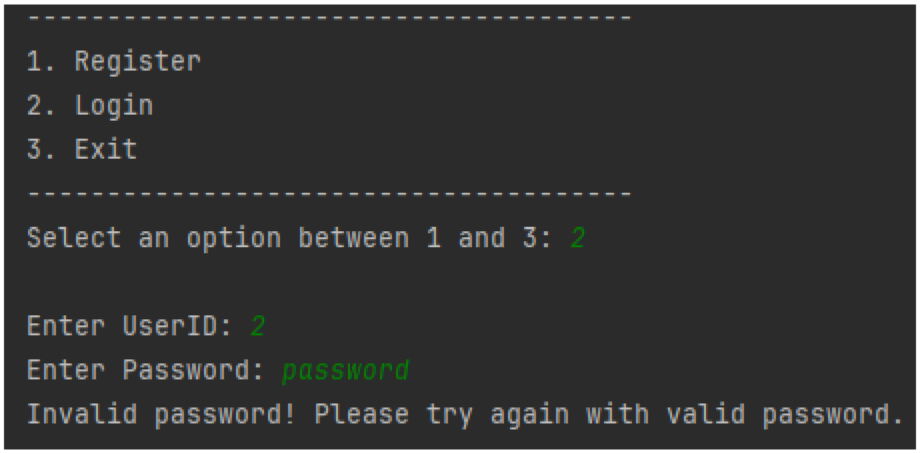
User tries to login with user id which is not registered:



```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 2

Enter UserID: 3
User not found! Please login with valid user ID.
```

Figure 24: Landing menu when user tries to login with unregistered user id

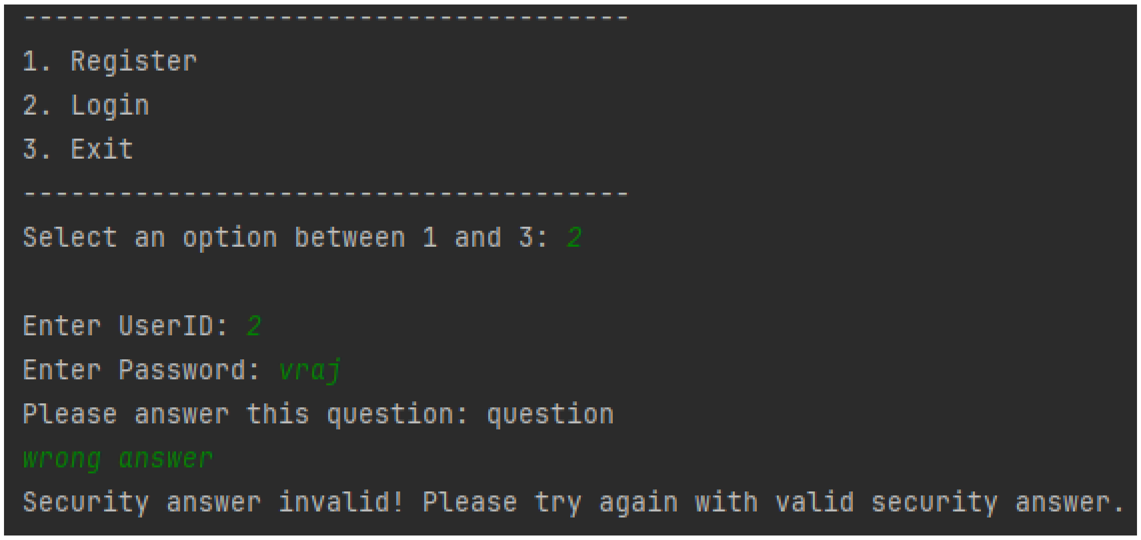
User provides wrong password during login:



```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 2

Enter UserID: 2
Enter Password: password
Invalid password! Please try again with valid password.
```

Figure 25: Landing menu when user tries to login with invalid password

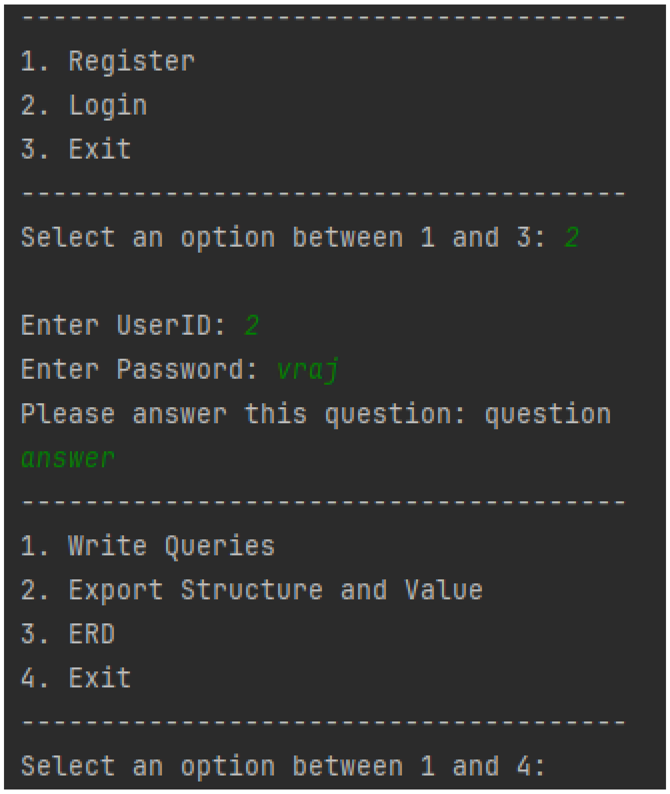
User provides wrong answer to security question:



```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 2

Enter UserID: 2
Enter Password: vraj
Please answer this question: question
wrong answer
Security answer invalid! Please try again with valid security answer.
```

Figure 26: Landing menu when user tries to login with invalid security answer

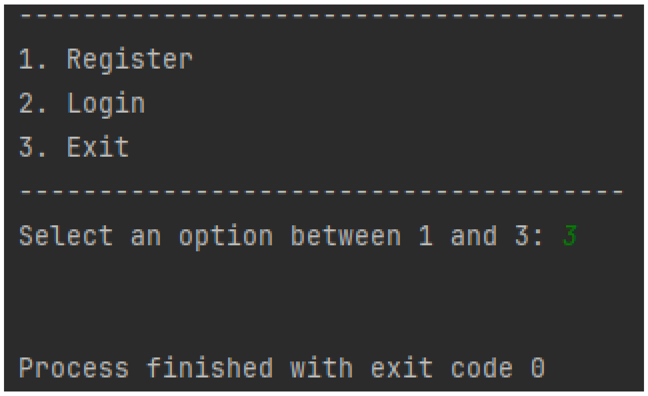
User provides valid credentials and answer to login:



```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 2

Enter UserID: 2
Enter Password: vraj
Please answer this question: question
answer
--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4:
```

Figure 27: Successful login from landing menu

User selects exit option:

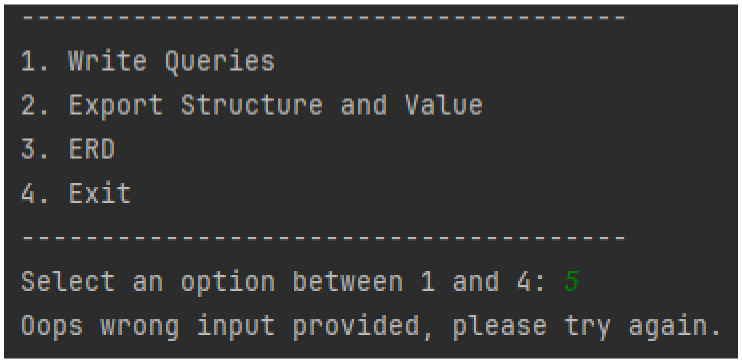


```
--------------------------------------
1. Register
2. Login
3. Exit
--------------------------------------
Select an option between 1 and 3: 3


Process finished with exit code 0
```

Figure 28: Landing menu after user selects Exit option

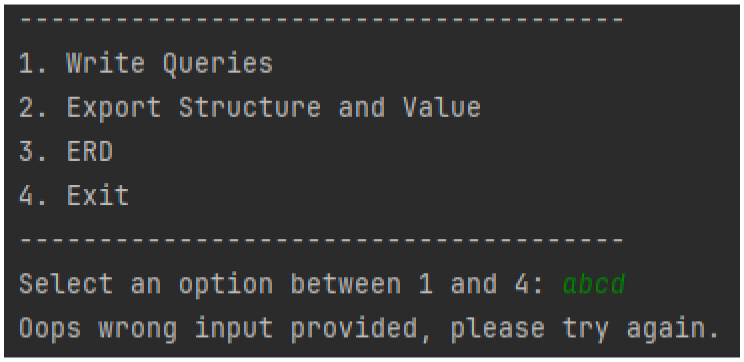
Main menu

User inputs wrong number for menu selection:



```
--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: 5
Oops wrong input provided, please try again.
```

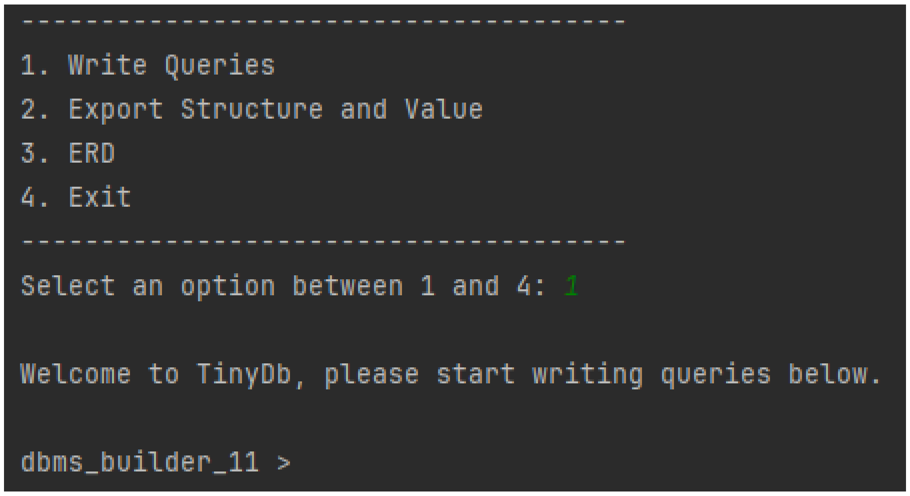
Figure 29: Main menu when user inputs invalid number - 5



```
--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: abcd
Oops wrong input provided, please try again.
```

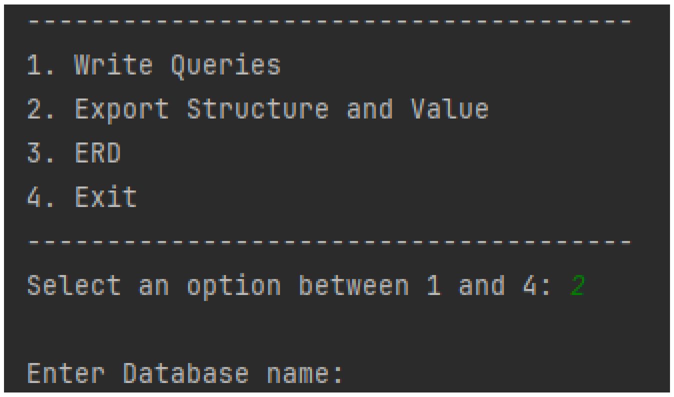
Figure 30: Main menu when user inputs invalid input "abcd"

Like landing menu, main menu selection will divert execution to individual functionalities i.e. 1 will start accepting queries, 2 will prompt user to input database to export, 3 will be implemented in upcoming sprint and 4 for exiting the application.



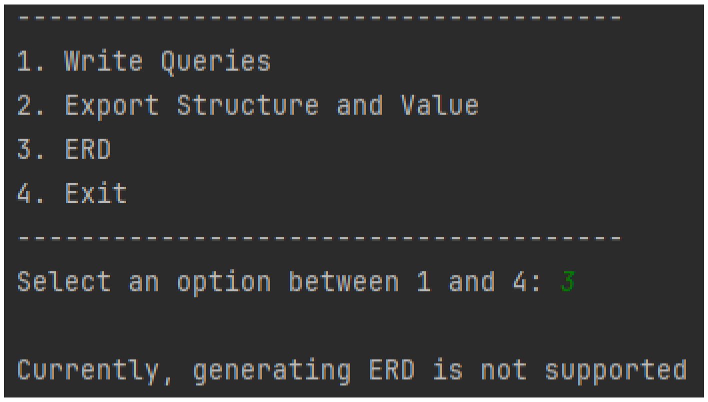
```
--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: 1

Welcome to TinyDb, please start writing queries below.

dbms_builder_11 >
```

Figure 31: Main menu after user selects Write Queries option



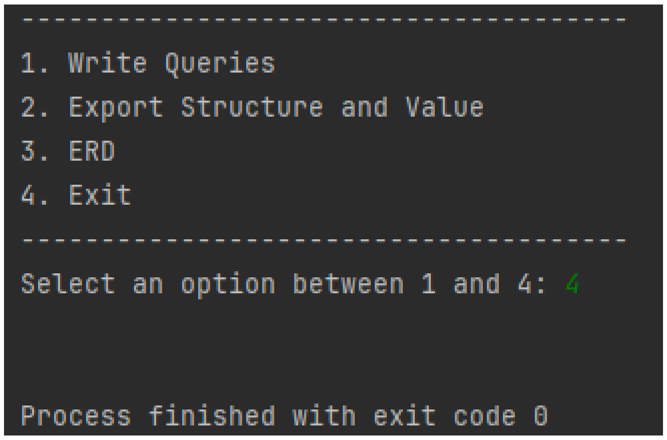
```
--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: 2

Enter Database name:
```

Figure 32: Main menu after user selects Export Structure and Value option



```
--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: 3

Currently, generating ERD is not supported
```

Figure 33: Main menu after user selects ERD option



```
--------------------------------------
1. Write Queries
2. Export Structure and Value
3. ERD
4. Exit
--------------------------------------
Select an option between 1 and 4: 4


Process finished with exit code 0
```

Figure 34: Main menu after user selects Exit option