

CSCI 5408
DATA MANAGEMENT AND
WAREHOUSING

ASSIGNMENT – 1

Problem2: Testcases and evidence of testing

Banner ID: B00948977

Git Assignment Link:

https://git.cs.dal.ca/sukumaran/csci5408_f23_b00948977_balaji_sukumaran/-/tree/main/Assignment1

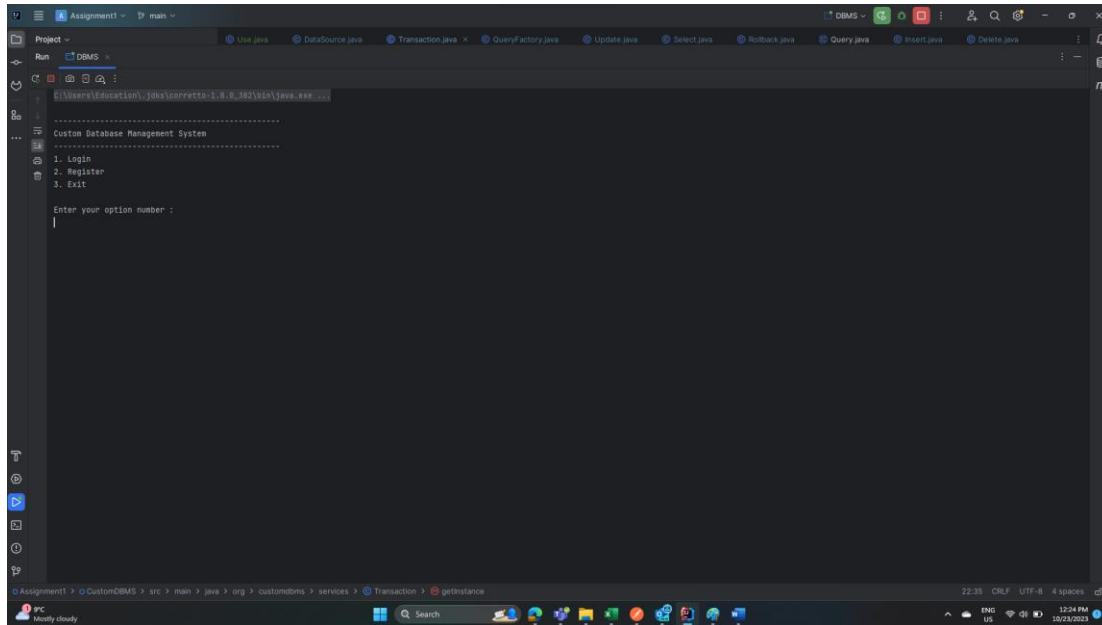
Table of contents

Section 1: Task A: Authentication (login and registration) testing.....	1
Section 2: Novelty Task: Data File.....	6
Section 3: Task B Implementation of Queries (DDL & DML) testing.....	7
Section 4: Task C, Implementation of transaction testing.....	16

Section 1: Task A Authentication (login and registration) testing

Step 1: Registration flow

Was able to open the application, following is the main menu.



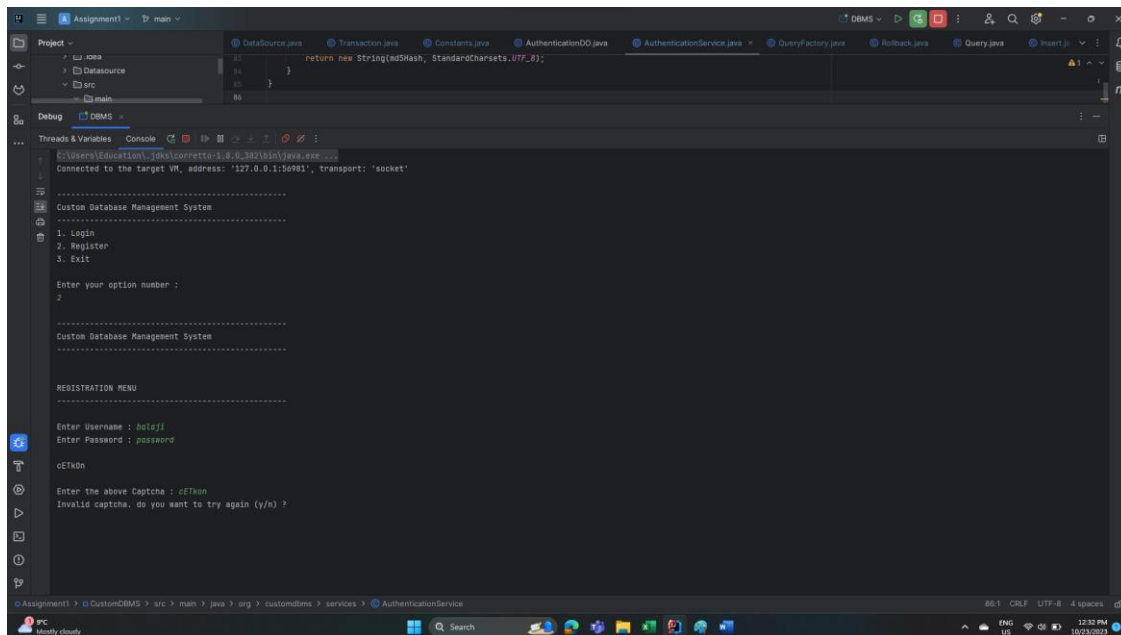
The screenshot shows a Java IDE with a project named 'Assignment'. The 'Run' tab is active, displaying the output of a Java application. The application is titled 'Custom Database Management System' and presents a main menu with three options: 1. Login, 2. Register, and 3. Exit. Below the menu, it prompts the user to 'Enter your option number :'. The IDE's interface includes a project explorer on the left, a code editor at the top, and a status bar at the bottom.

```
Custom Database Management System
-----
1. Login
2. Register
3. Exit

Enter your option number :
|
```

Figure 1: Custom DBMS main menu

Select option 2 to register but type an **invalid captcha**



The screenshot shows the same Java IDE, but now the 'Debug' tab is active. The application has moved to the registration menu. It prompts the user to 'Enter your option number :', and the user has entered '2'. The application then displays the 'REGISTRATION MENU' and prompts for 'Enter Username :', 'Enter Password :', and 'cETkdn'. It then displays 'Enter the above Captcha : cETkdn' and 'Invalid captcha, do you want to try again (y/n) ?'. The IDE's interface is consistent with the previous screenshot.

```
Custom Database Management System
-----
1. Login
2. Register
3. Exit

Enter your option number :
2

Custom Database Management System
-----

REGISTRATION MENU
-----

Enter Username : bulaji
Enter Password : password
cETkdn

Enter the above Captcha : cETkdn
Invalid captcha, do you want to try again (y/n) ?
```

Figure 2: Registration menu invalid captcha

Retry again and give correct captcha. **User registered successfully.**

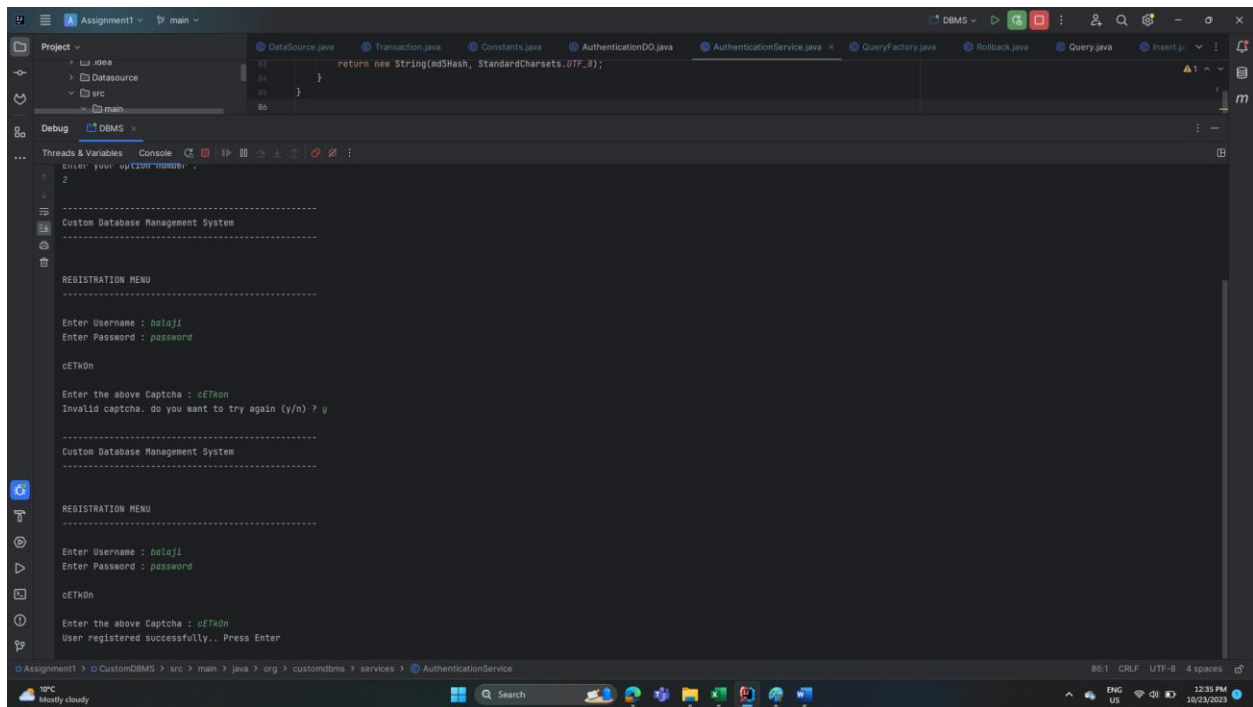


Figure 3: user registered successfully.

Credentials stored in the systems folder based on the application config and password is encrypted in the MD5 hashing algorithm.

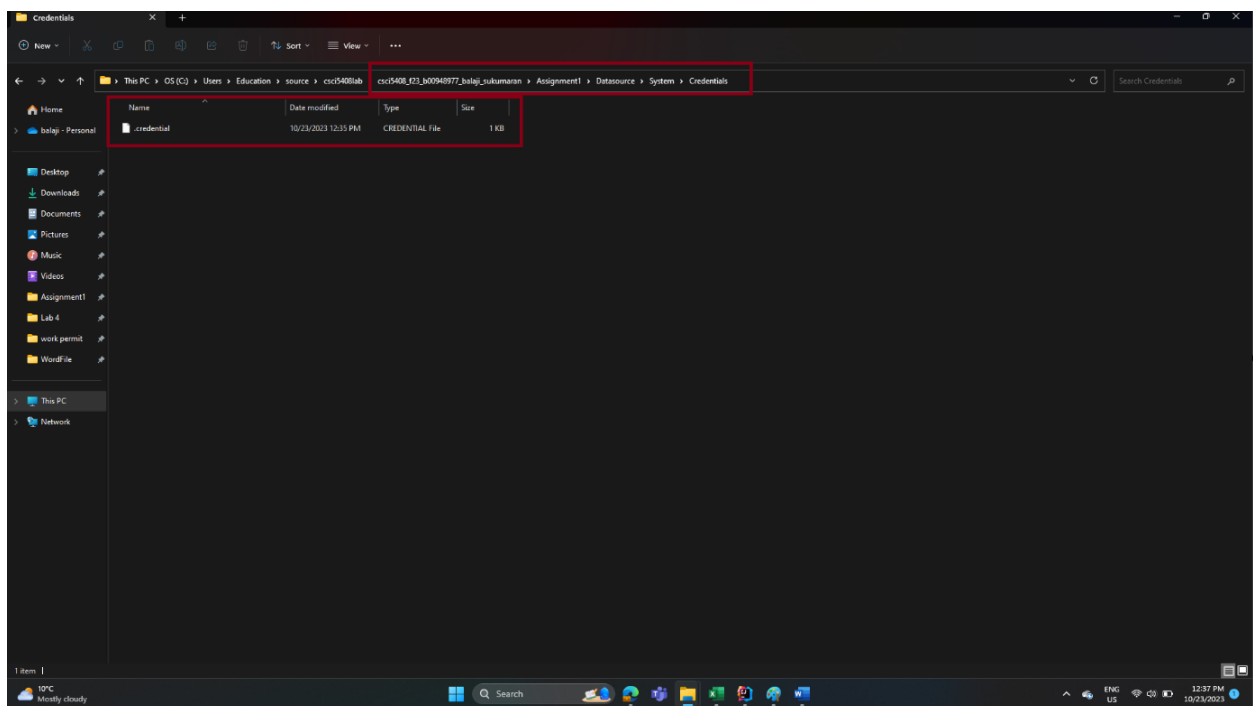
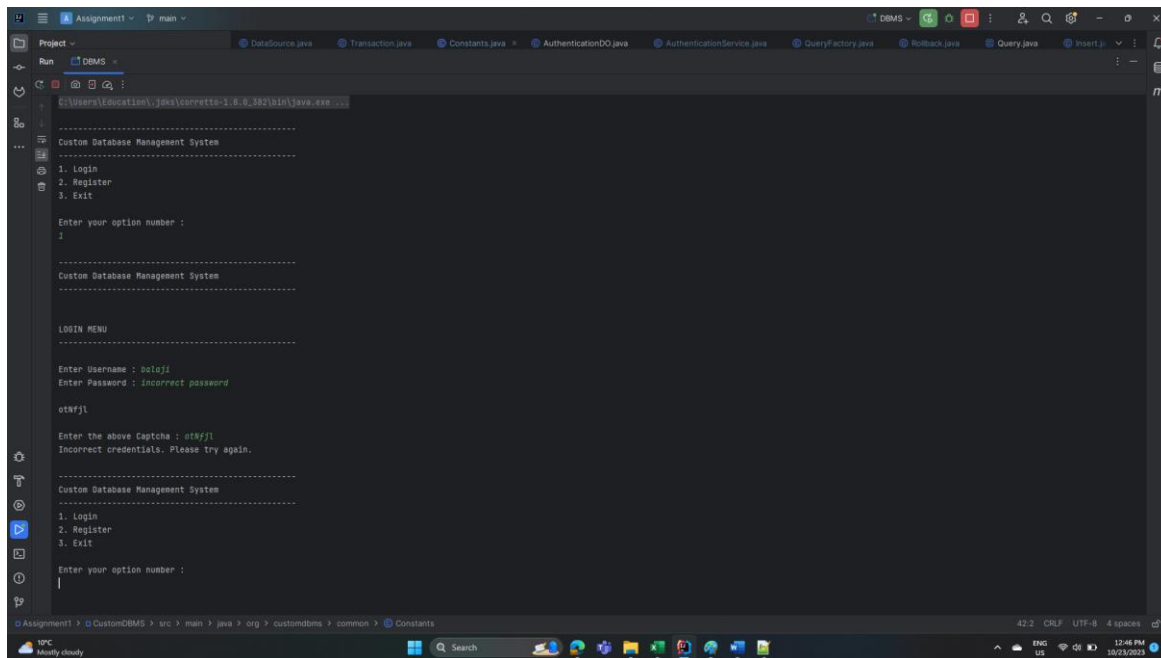


Figure 4: credential file folder structure

Step 2: Login Flow,

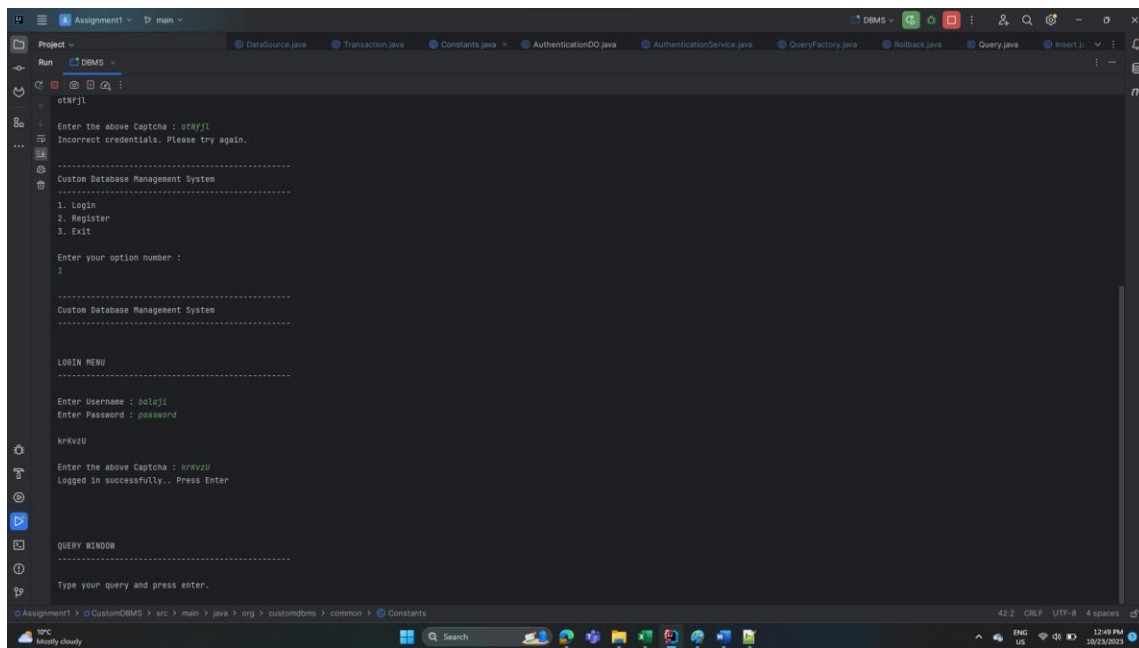
Tried login using **incorrect password** and the system has showed the error.



```
Project - Assignment1 - main - DBMS - DataSource.java Transaction.java Constants.java AuthenticationDO.java AuthenticationService.java QueryFactory.java Rollback.java Query.java Insert.java  
Run DBMS  
C:\Users\Educator\java\corretto-1.8.0_302\bin\java.exe ...  
-----  
Custom Database Management System  
-----  
1. Login  
2. Register  
3. Exit  
Enter your option number :  
1  
-----  
Custom Database Management System  
-----  
LOGIN MENU  
-----  
Enter Username : otaaji  
Enter Password : incorrect password  
otaaji  
Enter the above Captcha : otaaji  
Incorrect credentials. Please try again.  
-----  
Custom Database Management System  
-----  
1. Login  
2. Register  
3. Exit  
Enter your option number :  
|
```

Figure 7: Login using incorrect password.

Tried login using correct password and the system has authenticated the user successfully and opened the query window.



```
Project - Assignment1 - main - DBMS - DataSource.java Transaction.java Constants.java AuthenticationDO.java AuthenticationService.java QueryFactory.java Rollback.java Query.java Insert.java  
Run DBMS  
C:\Users\Educator\java\corretto-1.8.0_302\bin\java.exe ...  
otaaji  
Enter the above Captcha : otaaji  
Incorrect credentials. Please try again.  
-----  
Custom Database Management System  
-----  
1. Login  
2. Register  
3. Exit  
Enter your option number :  
1  
-----  
Custom Database Management System  
-----  
LOGIN MENU  
-----  
Enter Username : otaaji  
Enter Password : password  
NPKVZU  
Enter the above Captcha : NPKVZU  
Logged in successfully.. Press Enter  
-----  
QUERY WINDOW  
-----  
Type your query and press enter.
```

Figure 8: Login successful using user id and password

Internally the password is hashed in the following method using MD5 algorithm.

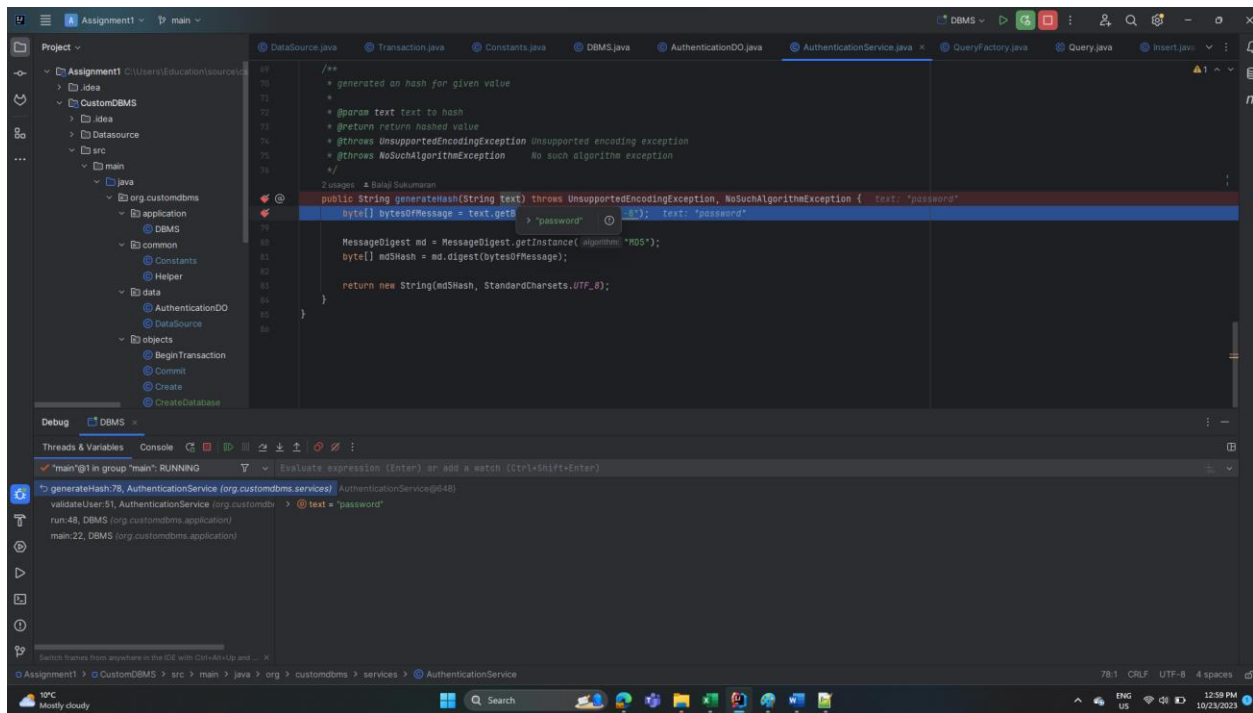


Figure 9: User password getting MD 5 hashed in this method

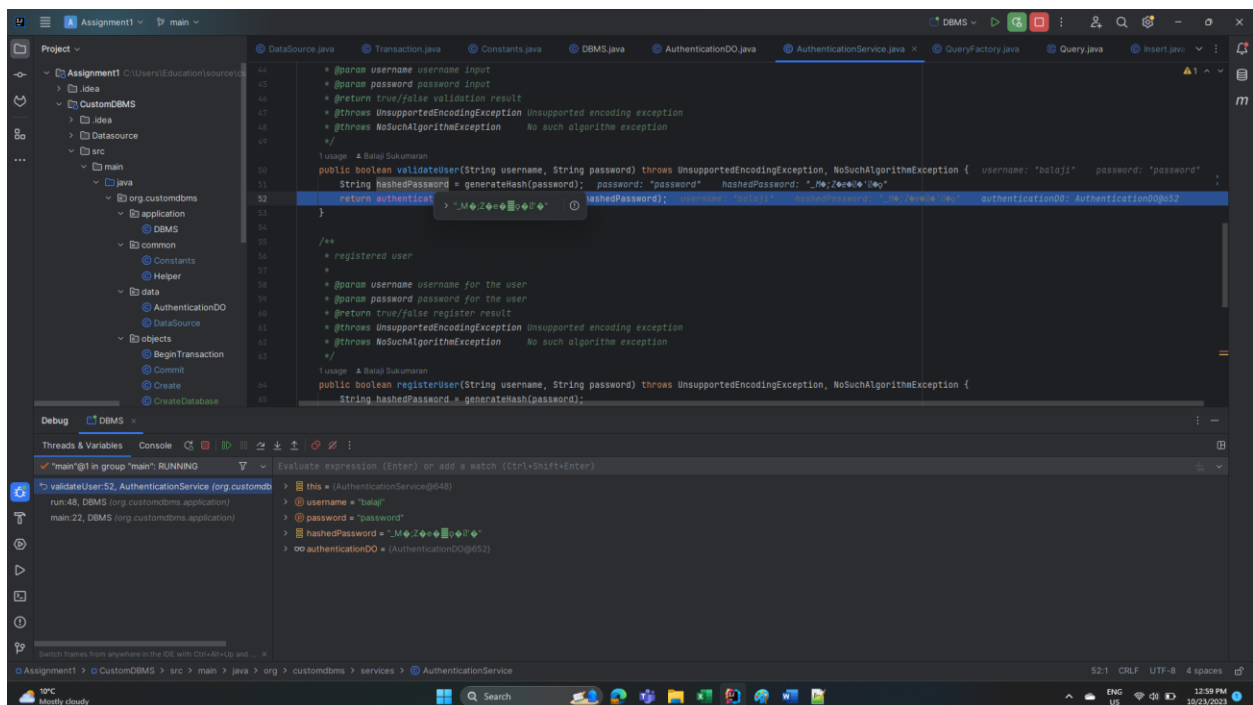
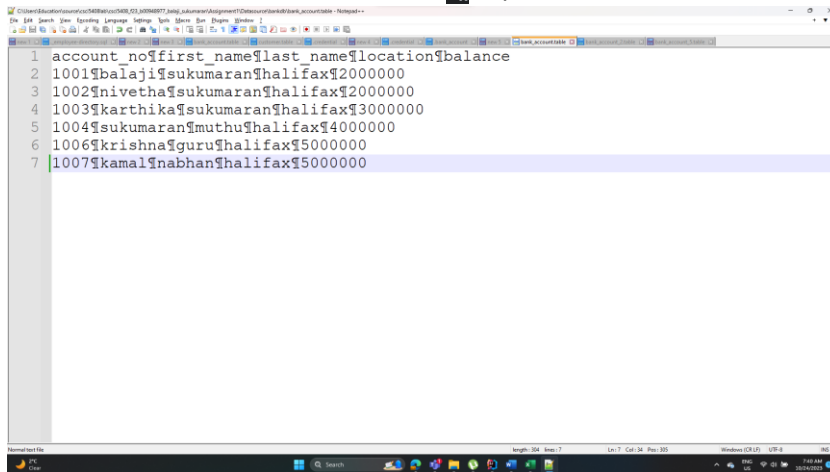


Figure 10: Hashed password

Section 2: Novelty Task: Data File

- the Data Files are stored in a Pilcrow separated file.



```
1 account_no§first_name§last_name§location§balance
2 1001§balaji§sukumaran§halifax§2000000
3 1002§nivetha§sukumaran§halifax§2000000
4 1003§karthika§sukumaran§halifax§3000000
5 1004§sukumaran§muthu§halifax§4000000
6 1006§krishna§guru§halifax§5000000
7 1007§kamal§nabhan§halifax§5000000
```

Figure 11: Pilcrow separated file

- Table meta-data will be stored in `.(table_name)` extension. For example table: `bank_account` meta data file will be stored as `.bank_account`

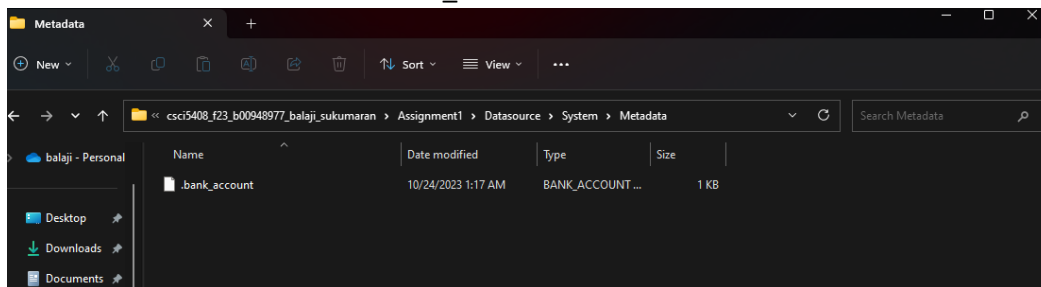


Figure 12: meta data file stored as `.(tablename)`

- Table file will be stored in `.table` extension. For example table: `bank_account` table file will be stored as `bank_account.table`.

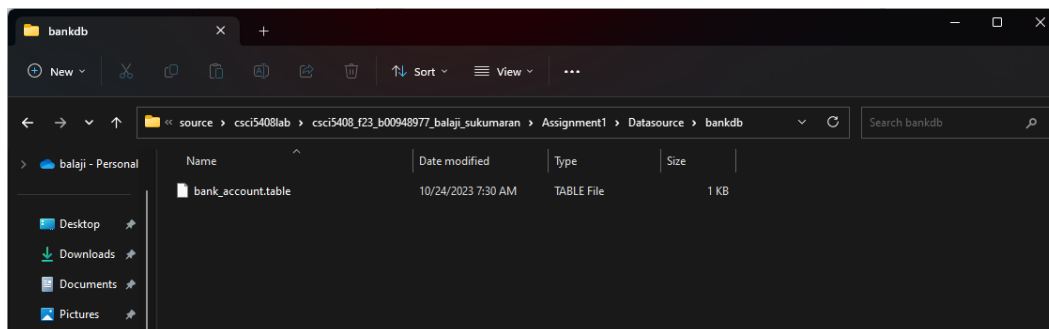


Figure 13: table file stored in `.table` extension

Section 3: Task B Implementation of Queries (DDL & DML) testing.

Step 1: Create database.

Entered “create database bankDB;” in the query window it executed successfully.

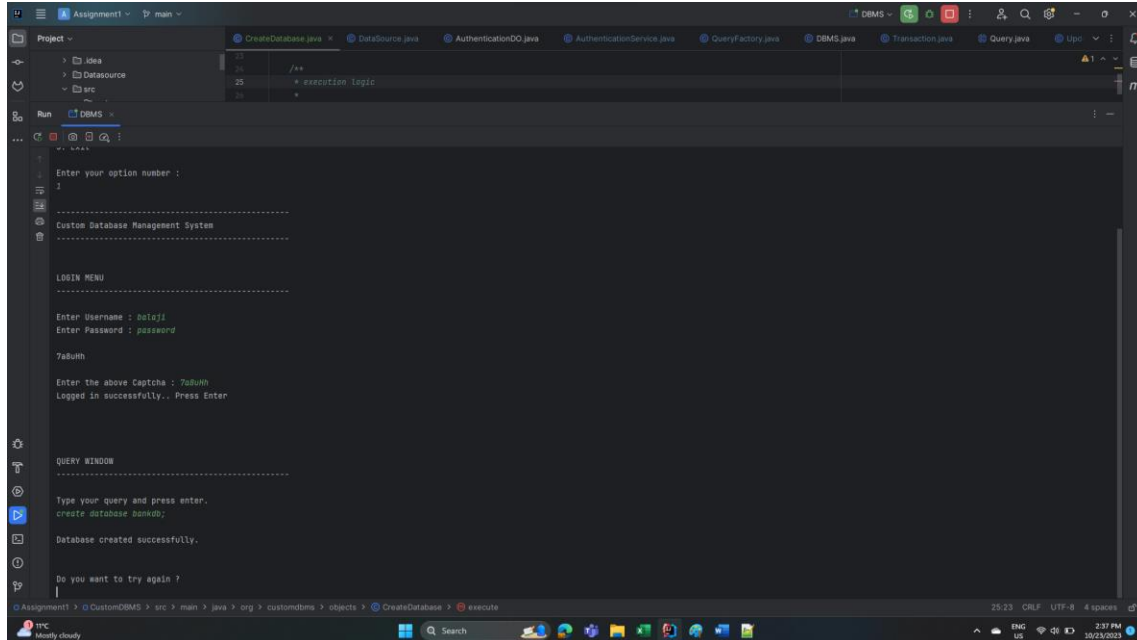


Figure 14: database created successfully

Internally it created a folder in the file system.

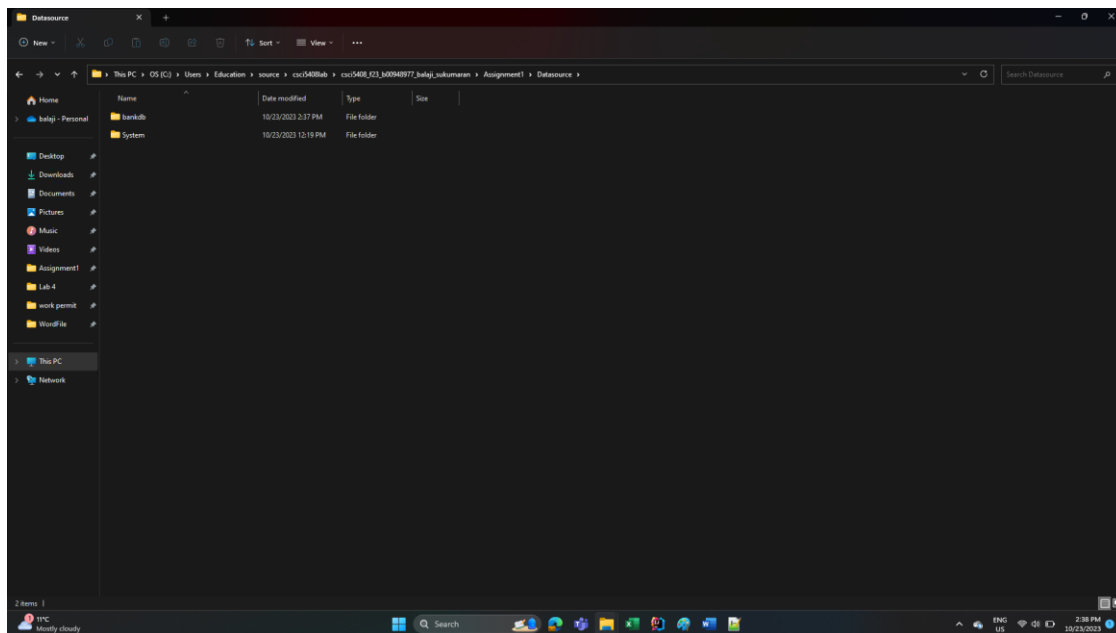


Figure 15: Folder created in the file system

Step 2: Tried to use the created database.

Execute “use bankdb”

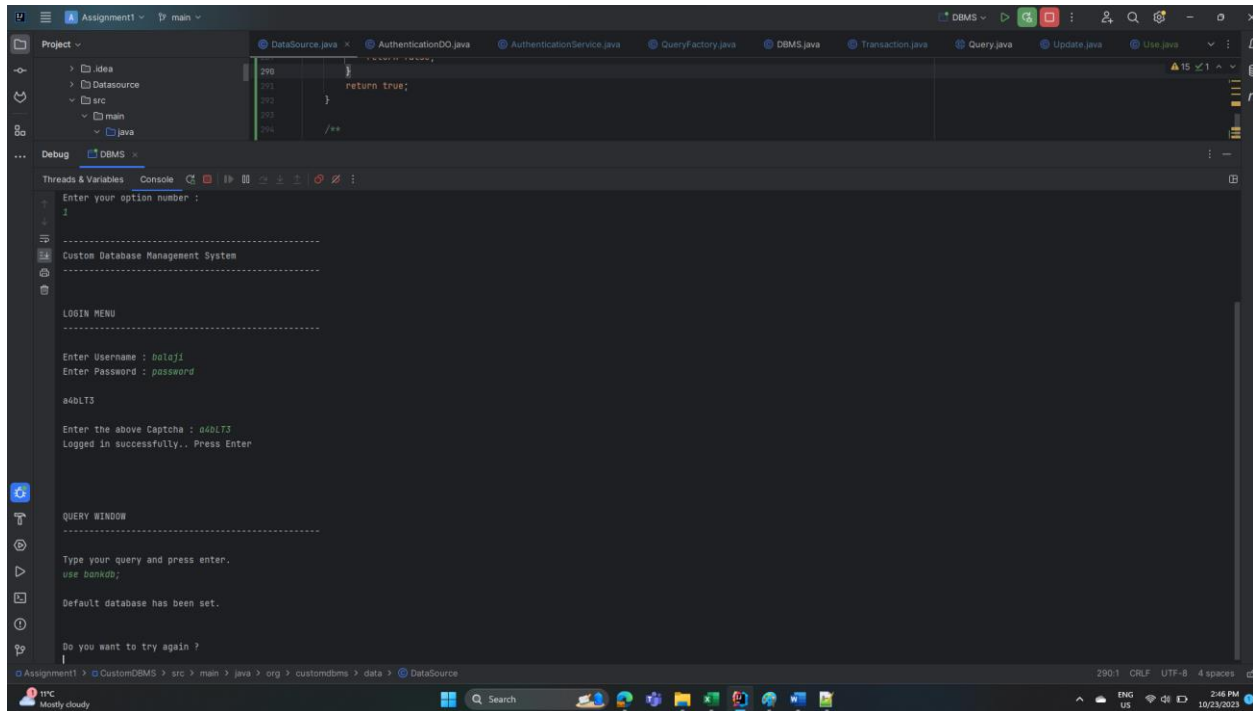


Figure 16: Default database chosen.

Internally, It marks the application where all the tables exists

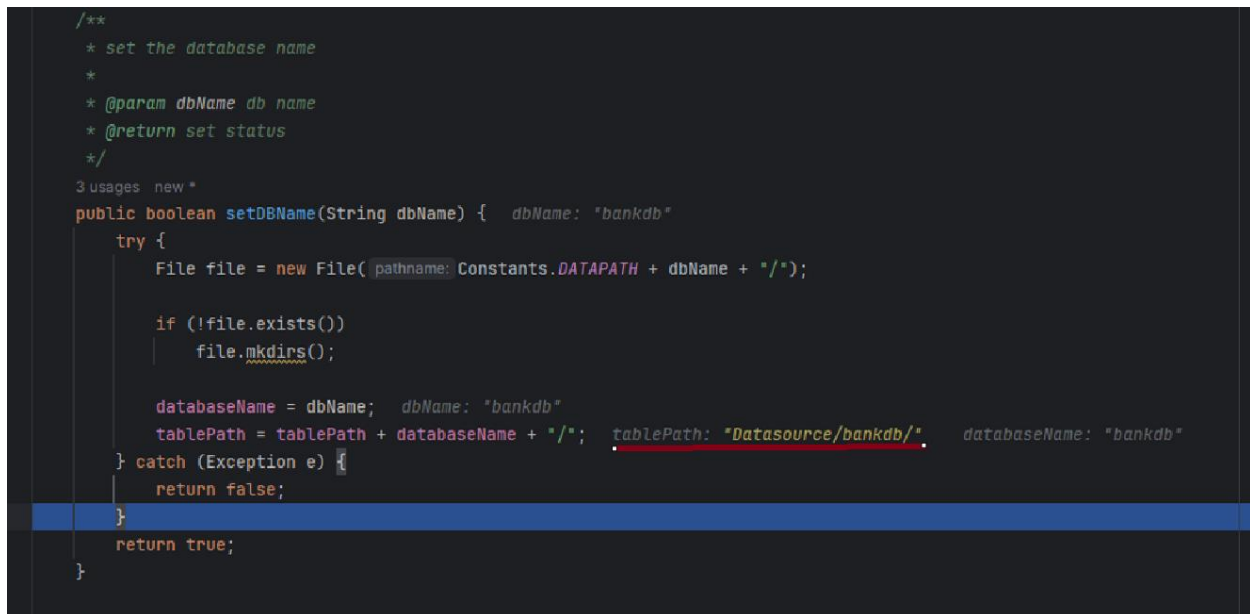


Figure 17: table access path has been set.

Step 3: Create a Table

Created a table using the “create table bank_account(account_no int primary key, first_name varchar(50), last_name varchar(50), location varchar(10), balance int)”. query executed successfully.

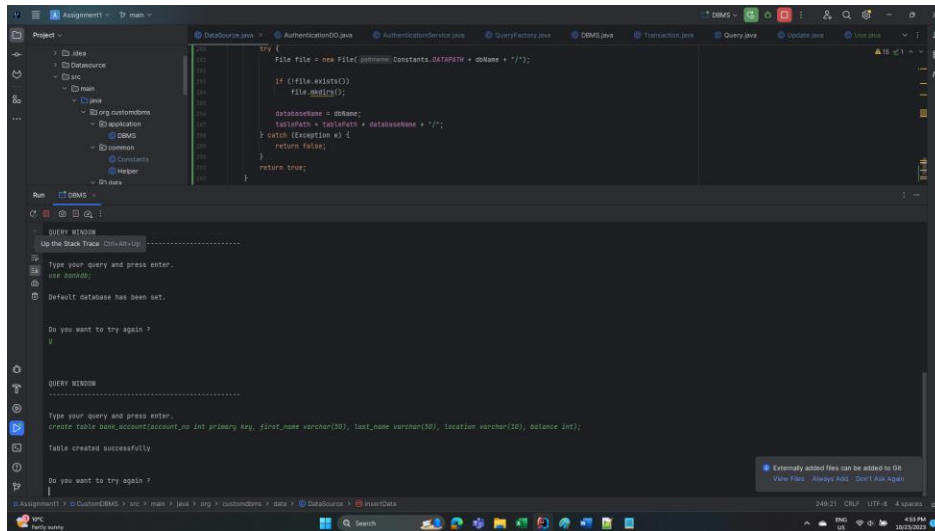


Figure 18: create table query

In the file system it created two files .bank_account contains the meta data about the table and the actual table is contained in bank_account.table

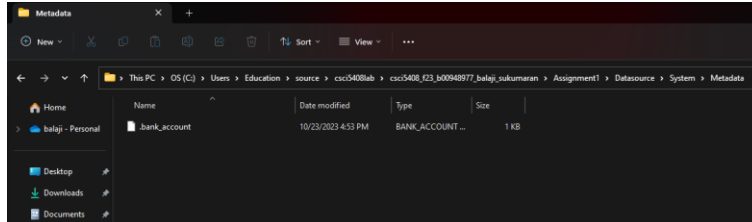


Figure 19: new file created for table metadata

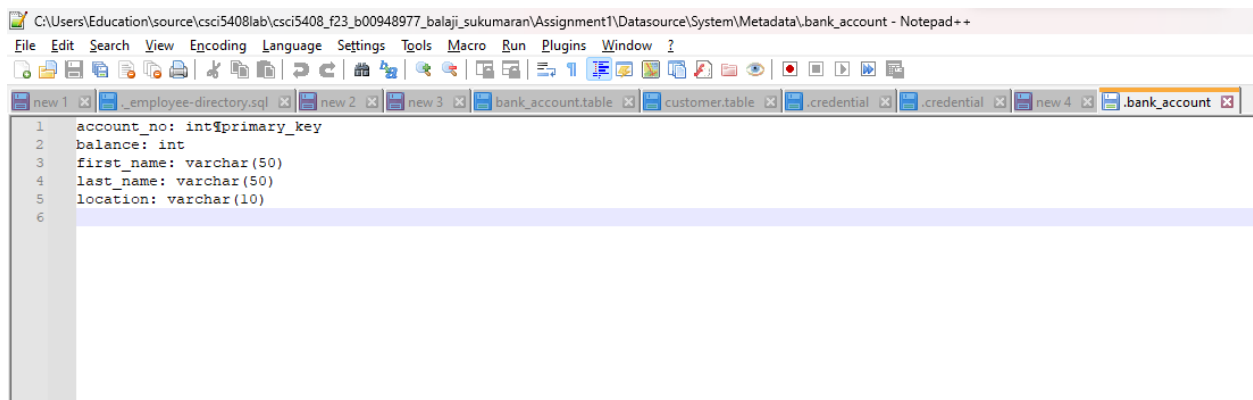


Figure 20: meta-data file contents

Table file is created under the database folder bankdb.

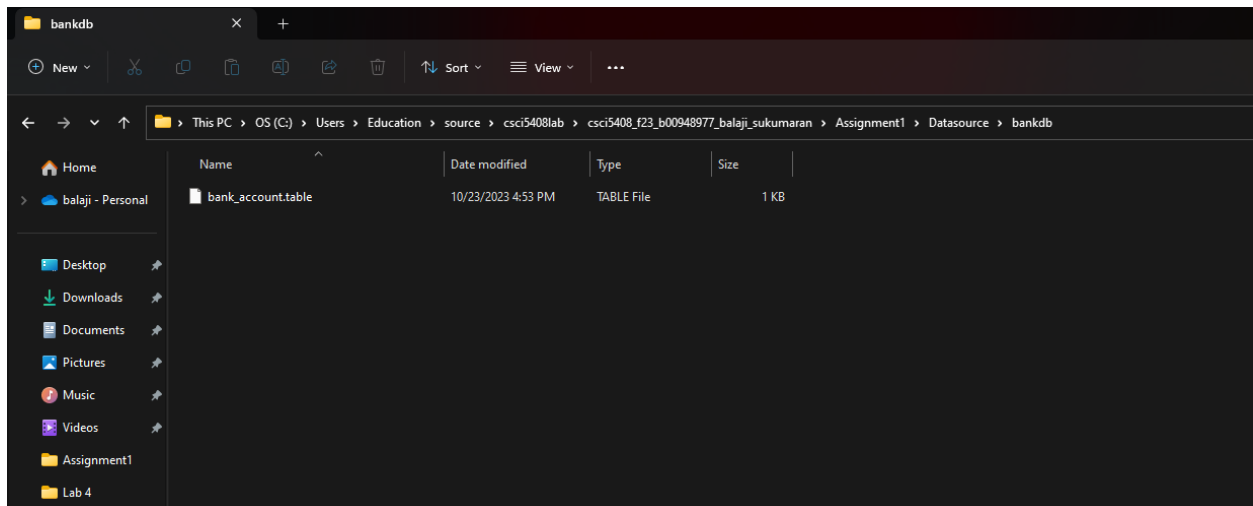


Figure 21: table file created.

Following is the table file contents.

It is a Pilcrow separated file.

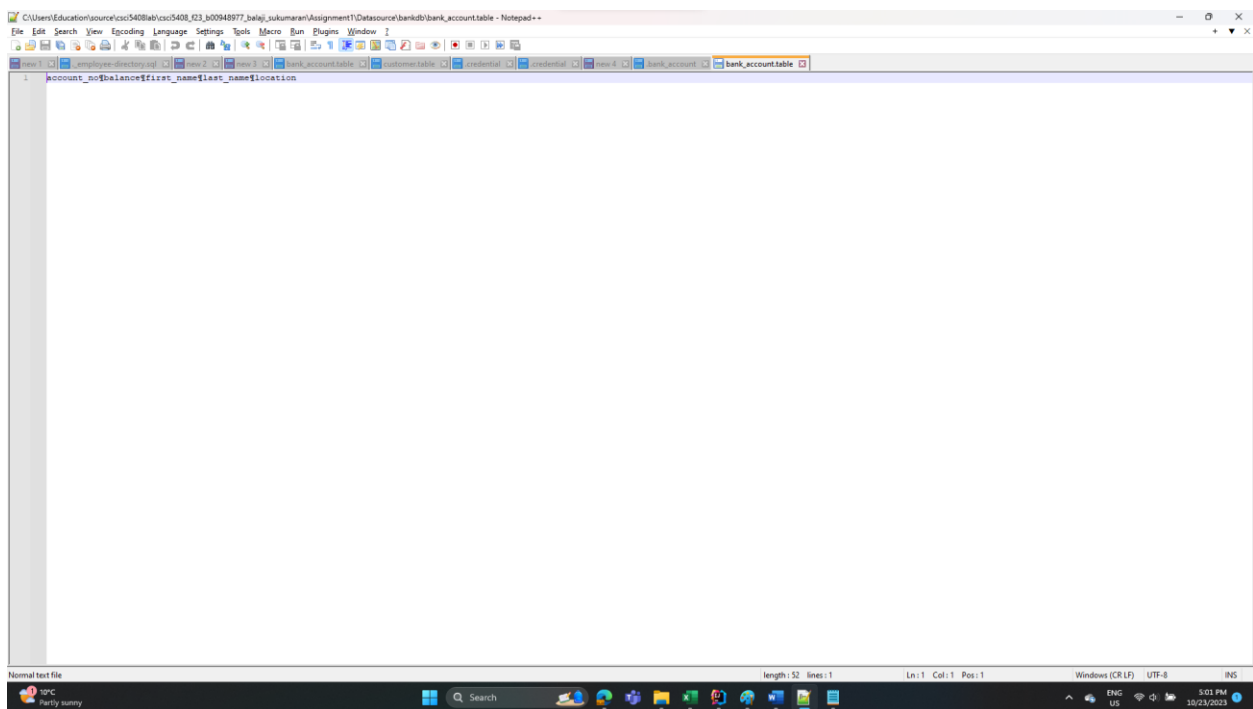
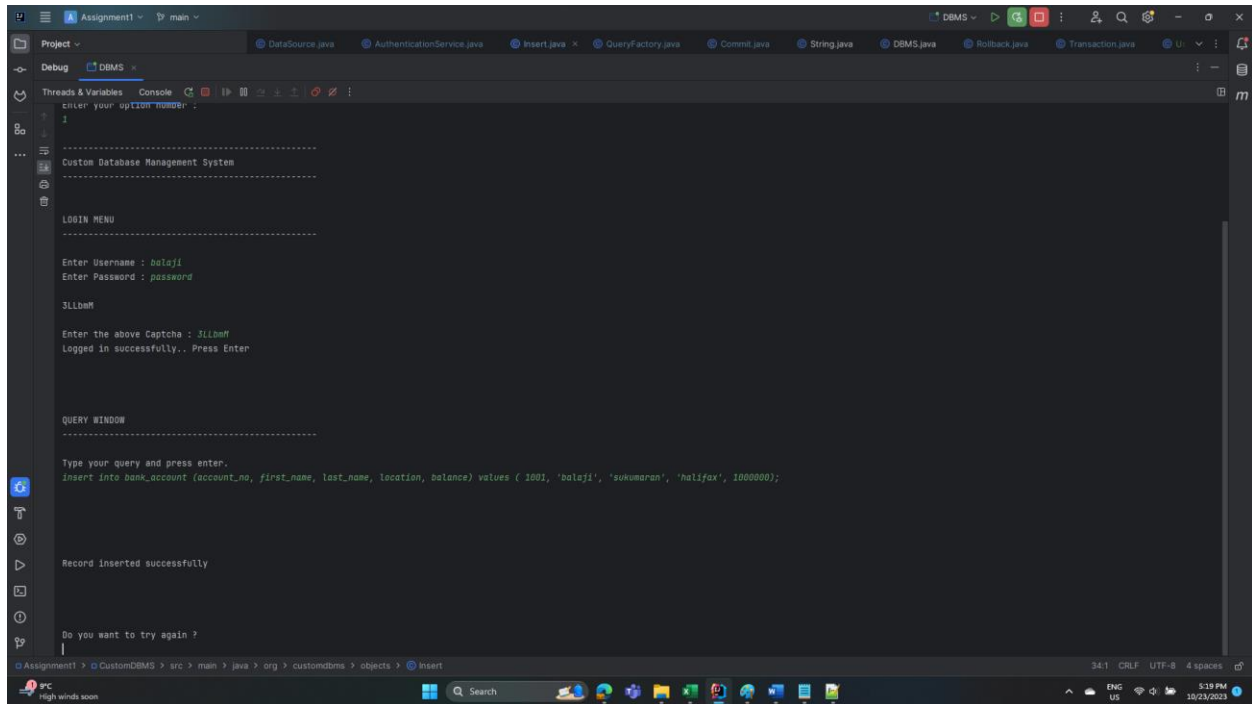


Figure 22: Pilcrow separated file.

Step 4: Insert into a table.

Tried to insert one row into the bank_account table. Record inserted successfully.



```
enter your username : balaji
password : password

-----
Custom Database Management System
-----

LOGIN MENU

Enter Username : balaji
Enter Password : password

JLlbnM
Enter the above Captcha : JLlbnM
Logged in successfully.. Press Enter

QUERY WINDOW

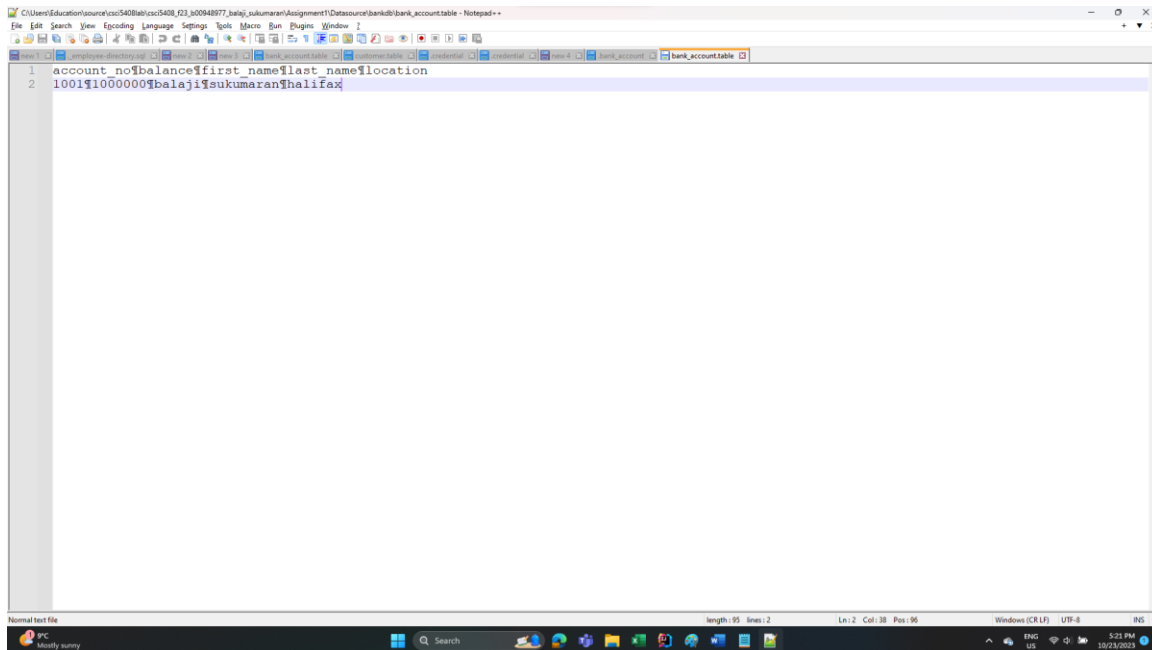
Type your query and press enter.
Insert into bank_account (account_no, first_name, last_name, location, balance) values ( 1001, 'balaji', 'sukumaran', 'halifax', 1000000);

Record inserted successfully

Do you want to try again ?
```

Figure 23: Record inserted successfully.

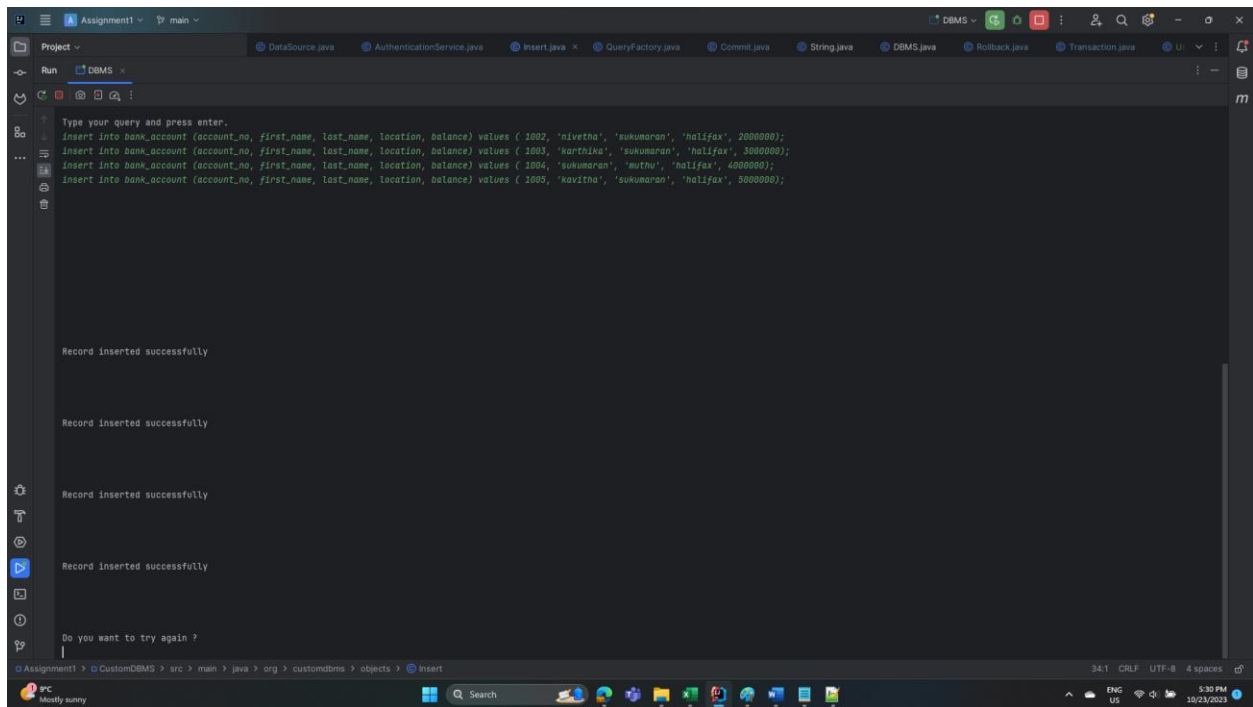
The updated table file looks like this.



```
1 account_no|balance|first_name|last_name|location
2 1001|1000000|balaji|sukumaran|halifax
```

Figure 24: Bank_account table file after single record insertion.

Tried to insert multiple records. One after another single record. All the records are inserted successfully.



The screenshot shows an IDE window with a project named 'Assignment1'. The 'Run' tab is active, displaying a SQL query editor and its output. The query editor contains four 'INSERT INTO bank_account' statements, each with a unique account number and name. The output window shows four 'Record inserted successfully' messages, one for each statement. The IDE's status bar at the bottom indicates the file is 'Insert' and the encoding is 'UTF-8'.

```
Type your query and press enter.
Insert into bank_account (account_no, first_name, last_name, location, balance) values ( 1002, 'nivetha', 'sukumaran', 'halifax', 2000000);
Insert into bank_account (account_no, first_name, last_name, location, balance) values ( 1003, 'karthika', 'sukumaran', 'halifax', 3000000);
Insert into bank_account (account_no, first_name, last_name, location, balance) values ( 1004, 'sukumaran', 'muthu', 'halifax', 4000000);
Insert into bank_account (account_no, first_name, last_name, location, balance) values ( 1005, 'kavitha', 'sukumaran', 'halifax', 5000000);

Record inserted successfully

Record inserted successfully

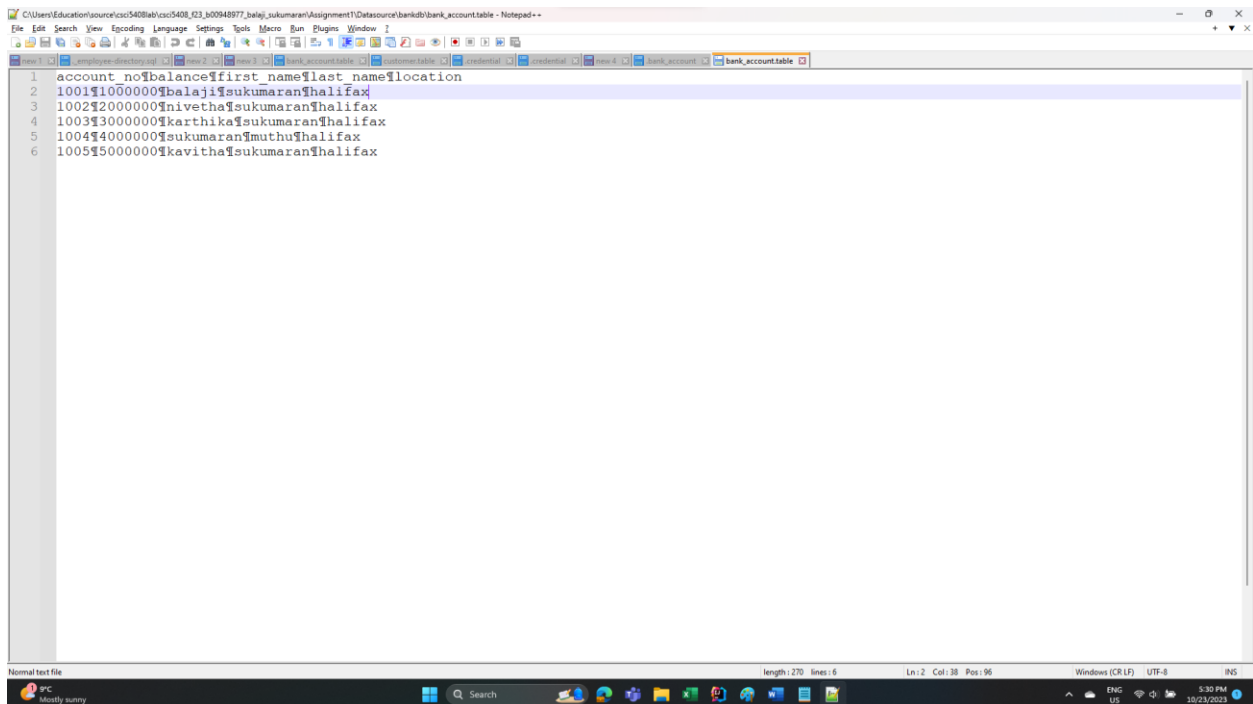
Record inserted successfully

Record inserted successfully

Do you want to try again ?
```

Figure 25: Inserting multiple records successfully.

The updated table file all the records looks like this.



The screenshot shows a Notepad++ window with the file 'bank_account.table' open. The file contains six lines of text, representing the table's structure and five records. The first line is a header with placeholders for account number, balance, first name, last name, and location. The subsequent five lines contain the data for each record, separated by semicolons.

```
1 account_no|balance|first_name|last_name|location
2 1001|1000000|balaji|sukumaran|halifax
3 1002|2000000|nivetha|sukumaran|halifax
4 1003|3000000|karthika|sukumaran|halifax
5 1004|4000000|sukumaran|muthu|halifax
6 1005|5000000|kavitha|sukumaran|halifax
```

Figure 26: bank_account.table file with 5 records

Step 5: Select from the bank_account table.

Executed “Select * from bank_account” query

```
Record inserted successfully
Record inserted successfully
Record inserted successfully

Do you want to try again ?
y

QUERY WINDOW
-----
Type your query and press enter.
select * from bank_account

account_no  balance  first_name  last_name  location
-----
1001        1000000  balaaji    sukumaran  halifax
1002        2000000  nivetha    sukumaran  halifax
1003        3000000  karthika   sukumaran  halifax
1004        4000000  sukumaran  muthu      halifax
1005        5000000  kavitha    sukumaran  halifax

Do you want to try again ?
|
```

Figure 27: select all from bank_account

Selected few columns and applied a where clause. The records were fetched correctly

```
Enter your option number :
1

Custom Database Management System
-----

LOGIN MENU

Enter Username : balaaji
Enter Password : password
dsK2ux

Enter the above Captcha : dsK2ux
Logged in successfully.. Press Enter

QUERY WINDOW
-----
Type your query and press enter.
select account_no, first_name, last_name, balance from bank_account where balance > 3000000

account_no  first_name  last_name  balance
-----
1004        sukumaran  muthu      4000000
1005        kavitha    sukumaran  5000000

Do you want to try again ?
```

Figure 28: select and where clause

Step 6: update the bank_account table.

Execute the “update bank_account set location='toronto' where account_no = 1001”. Record updated successfully.

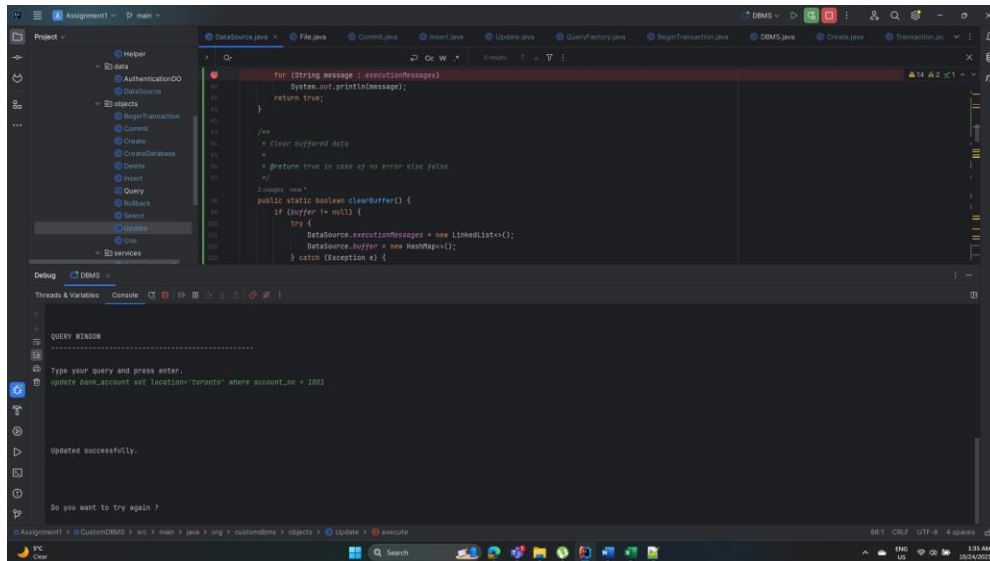


Figure 29: Record updated successfully.

The file in the file system got updated according

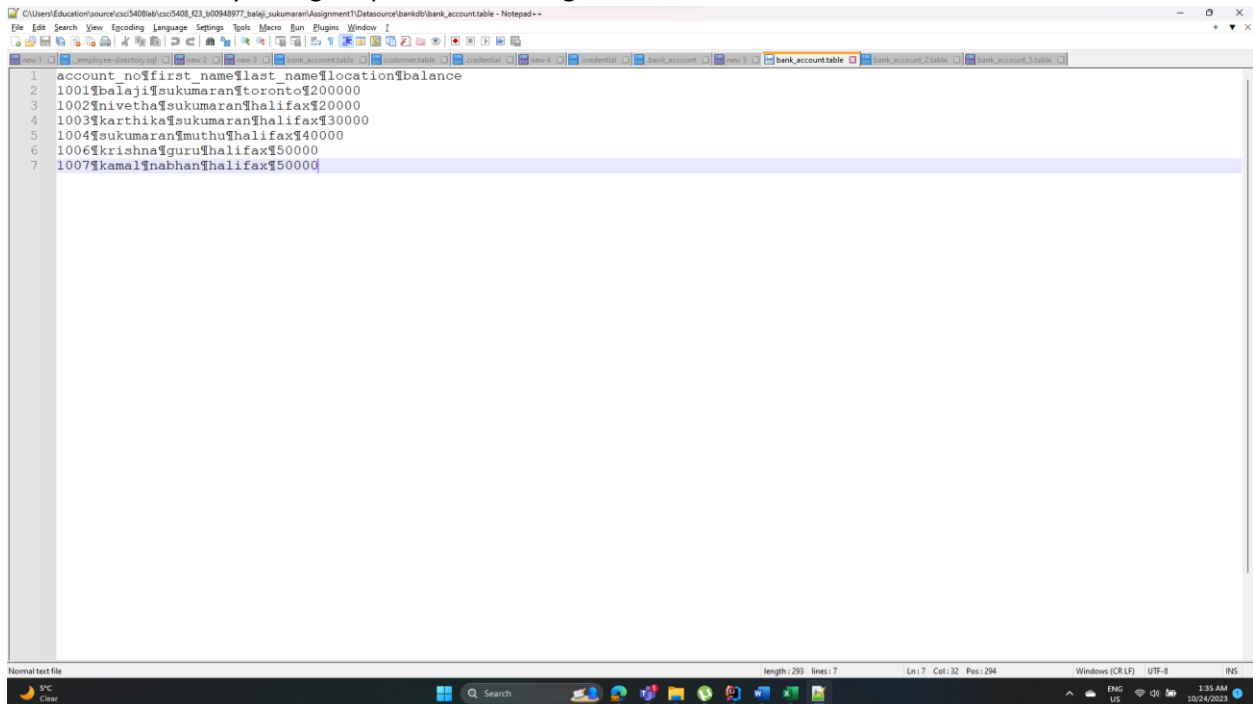


Figure 30: bank_account table file after update

Step 7: Delete records from bank_account table.

Executed “delete from bank_account where account_no=1001”. Record executed successfully.

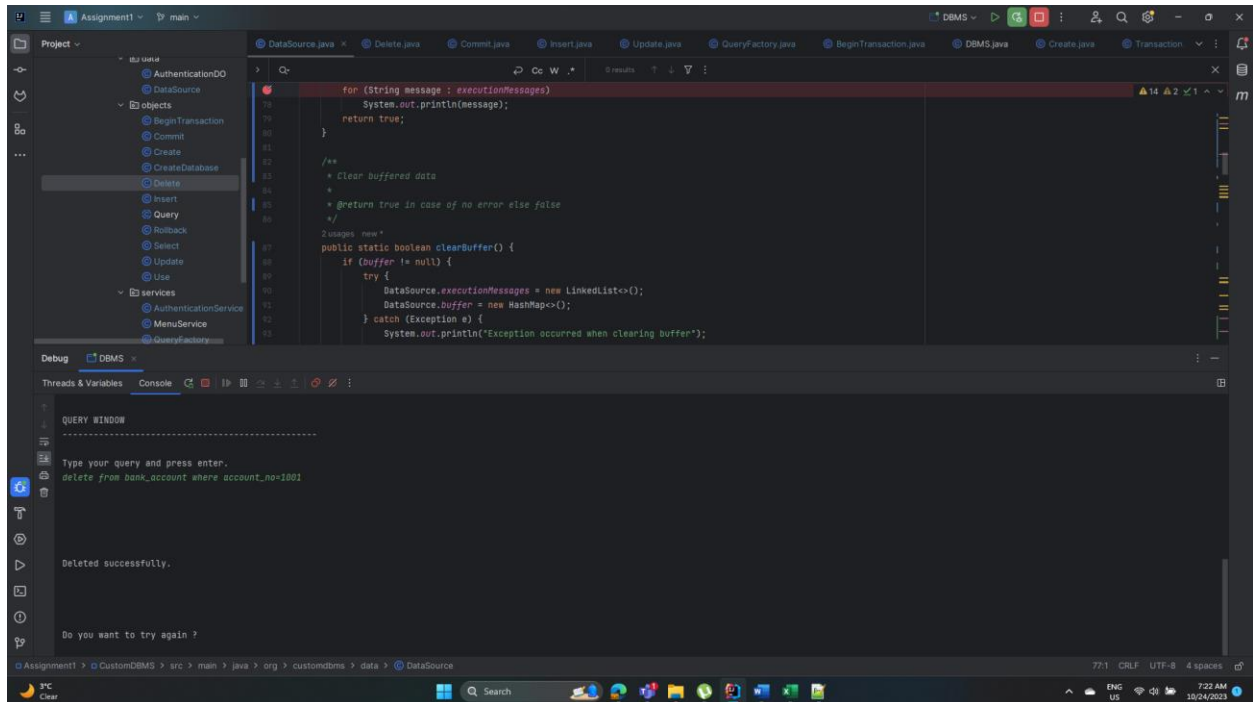


Figure 31: Record deleted successfully.

The table file in the filesystem has been updated to the following (record: 1001 is deleted)

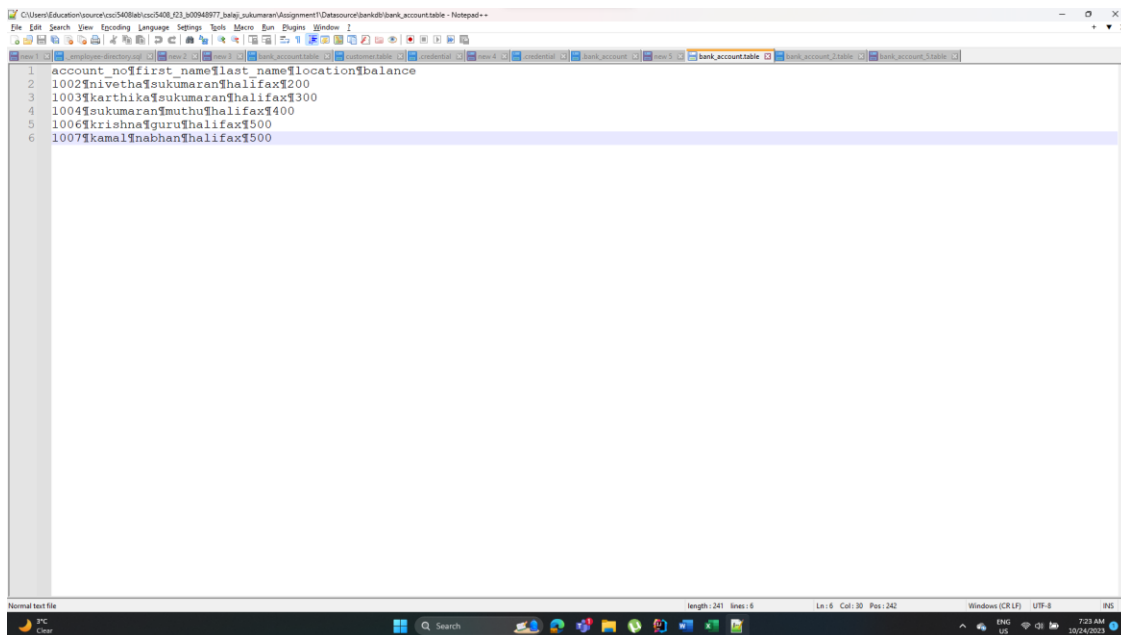


Figure 32: Record deleted from the table file.

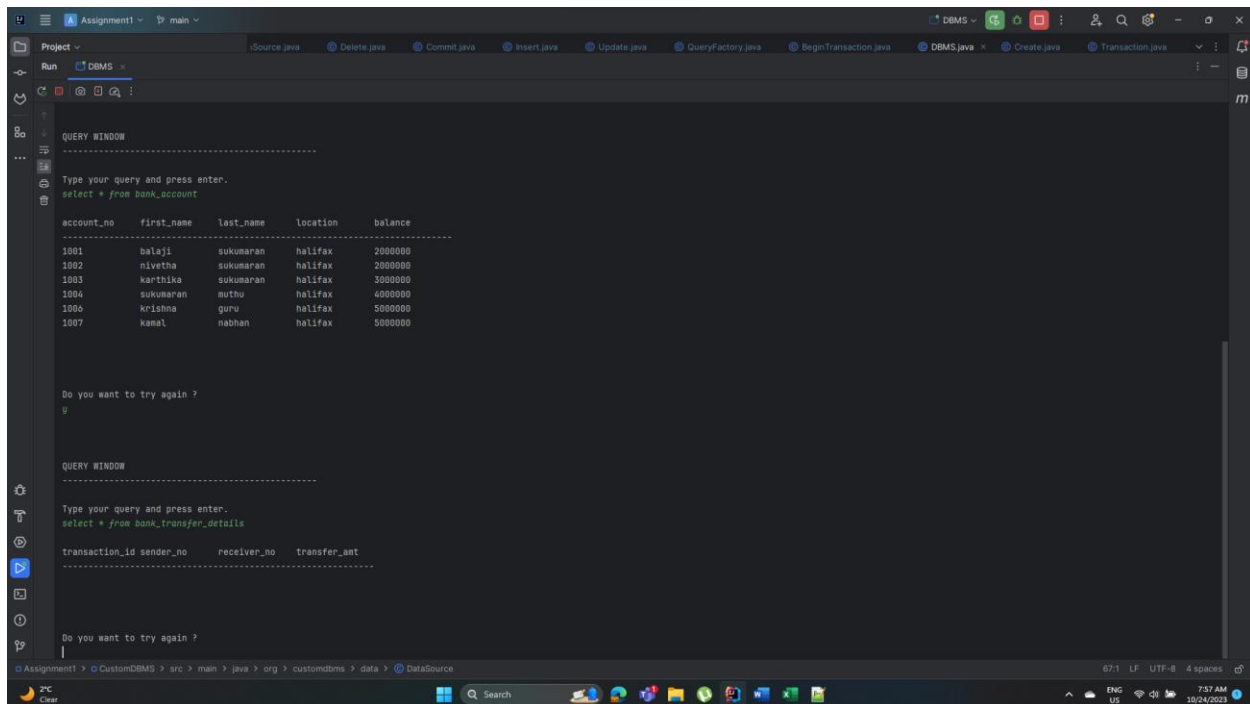
Section 4: Task C, Implementation of transaction testing

Table setup

Bank_account: bank_account {account_no, first_name, last_name, location, balance}

Bank_transfer details: bank_transfer_details {transaction_id, sender_no, receiver_no, transfer_amt}

We have 5 bank accounts in **bank_account** table and once a transfer has been made it should be logged in the **bank transfer details table**



```
Project: Assignment1 | main
Run: DBMS
Source.java | Delete.java | Commit.java | Insert.java | Update.java | QueryFactory.java | BeginTransaction.java | DBMS.java | Create.java | Transaction.java

QUERY WINDOW
-----
Type your query and press enter.
select * from bank_account

account_no  first_name  last_name  location  balance
-----
1001        balaji      sukumaran  halifax   2000000
1002        nivetha     sukumaran  halifax   2000000
1003        karthika    sukumaran  halifax   3000000
1004        sukumaran   muthu      halifax   4000000
1006        krishna     guru       halifax   5000000
1007        kamal       nabhan     halifax   5000000

Do you want to try again ?
y

QUERY WINDOW
-----
Type your query and press enter.
select * from bank_transfer_details

transaction_id sender_no  receiver_no  transfer_amt
-----
1           1001       1002         1000000

Do you want to try again ?
y

Assignment1 | CustomDBMS | src | main | java | org | customdbms | data | DataSource
67-1 LF UTF-8 4 spaces
```

Figure 33: bank_account and bank_transfer_details tables

Commit Flow and its intermediate state of the buffer.

Trying to execute the following query to simulate a banking transaction.

1. Account no 1001 (Balaji) is sending \$1000 to Account no(Nivetha) 1002.
2. This transaction should be logged in the **bank_transfer_details** table
3. Commit the transaction and persist the changes

Query:

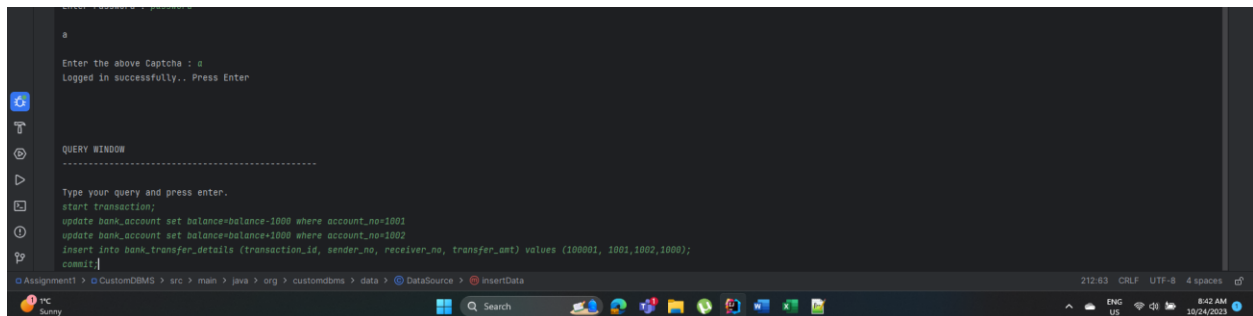
start transaction;

update bank_account set balance=balance-1000 where account_no=1001

update bank_account set balance=balance+1000 where account_no=1002

insert into bank_transfer_details (transaction_id, sender_no, receiver_no, transfer_amt) values (100001, 1001,1002,1000);

commit;



```
Enter the above Captcha : a
Logged in successfully.. Press Enter

QUERY WINDOW
-----
Type your query and press enter.
start transaction;
update bank_account set balance=balance-1000 where account_no=1001
update bank_account set balance=balance+1000 where account_no=1002
insert into bank_transfer_details (transaction_id, sender_no, receiver_no, transfer_amt) values (100001, 1001,1002,1000);
commit;
```

Figure 34: executing the transaction query.

When the flow executes till the first update (Money deducted from Balaji's account). Before transferring the amount to Nivetha's account. The buffer contains the following.

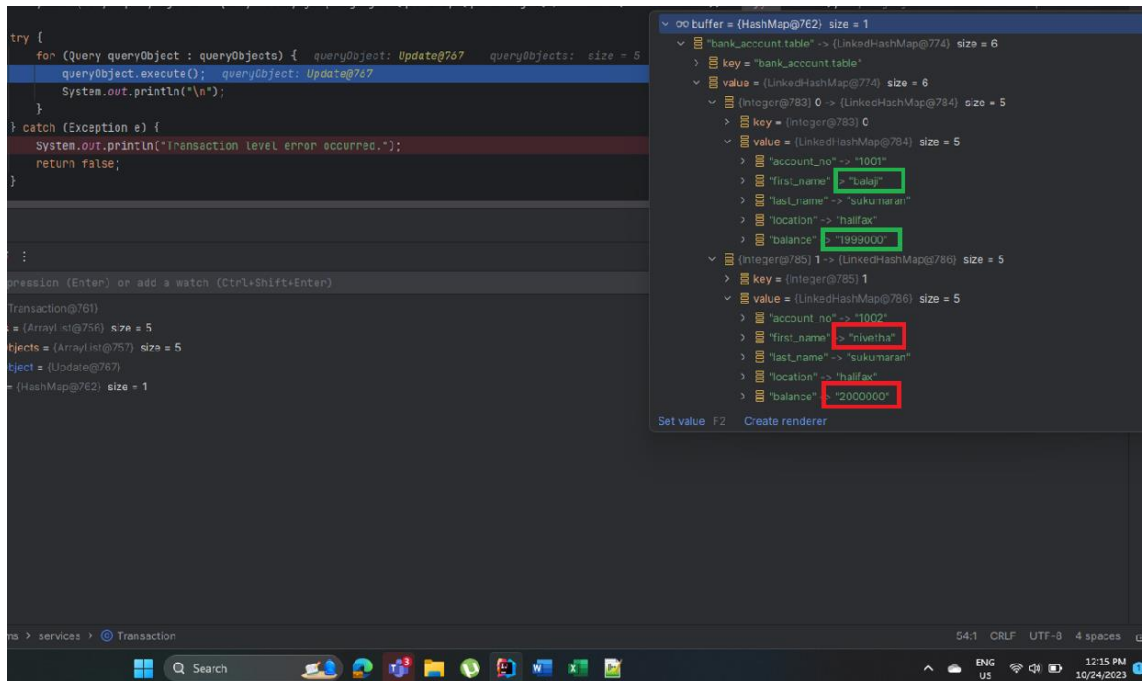


Figure 35: Buffer after the first update.

Now, when the flow goes executes the second update (Money transferred to Nivetha). Buffer will be in following state.

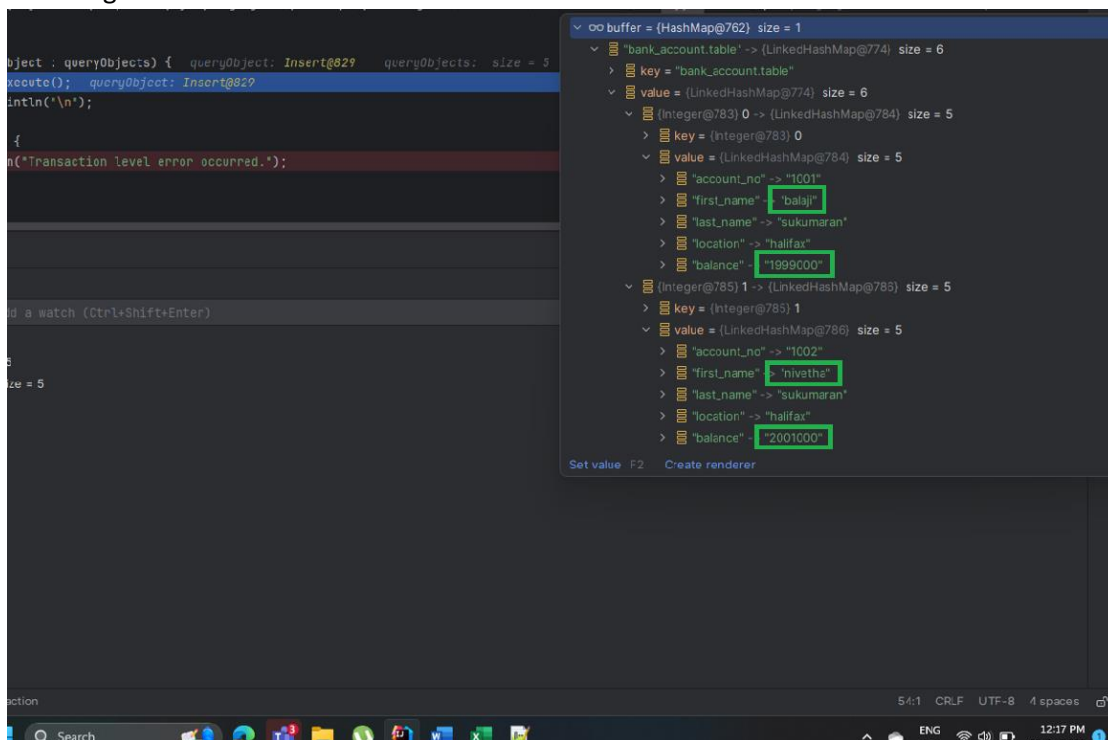


Figure 36: Buffer after the second update

Now, after logging this transaction, it the buffer will have two table states, bank_transfer_details and bank_account

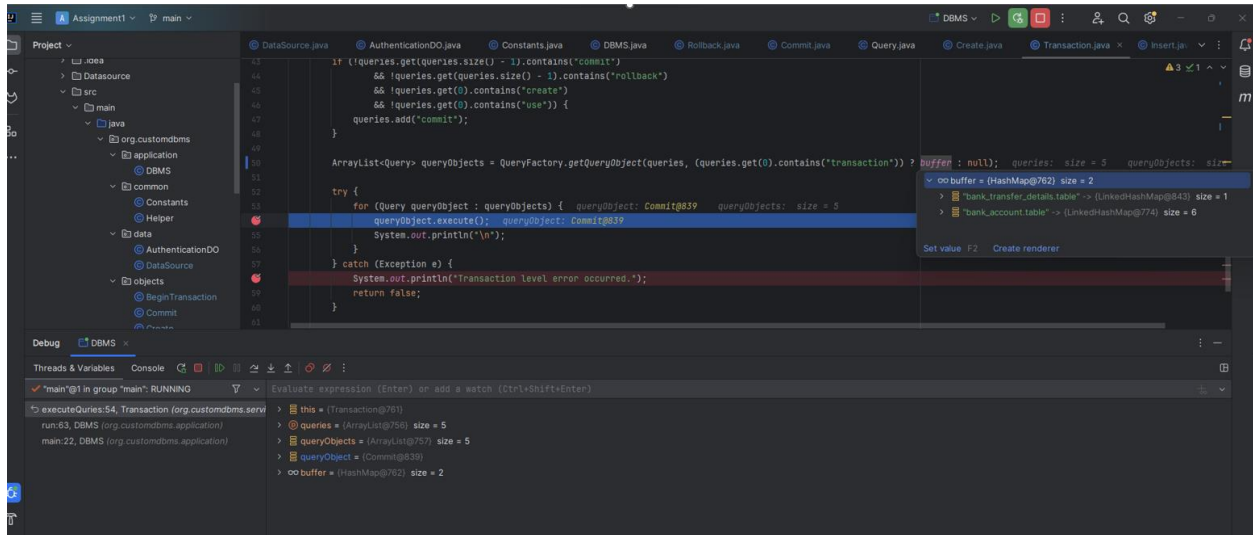


Figure 37: buffer with two table changes

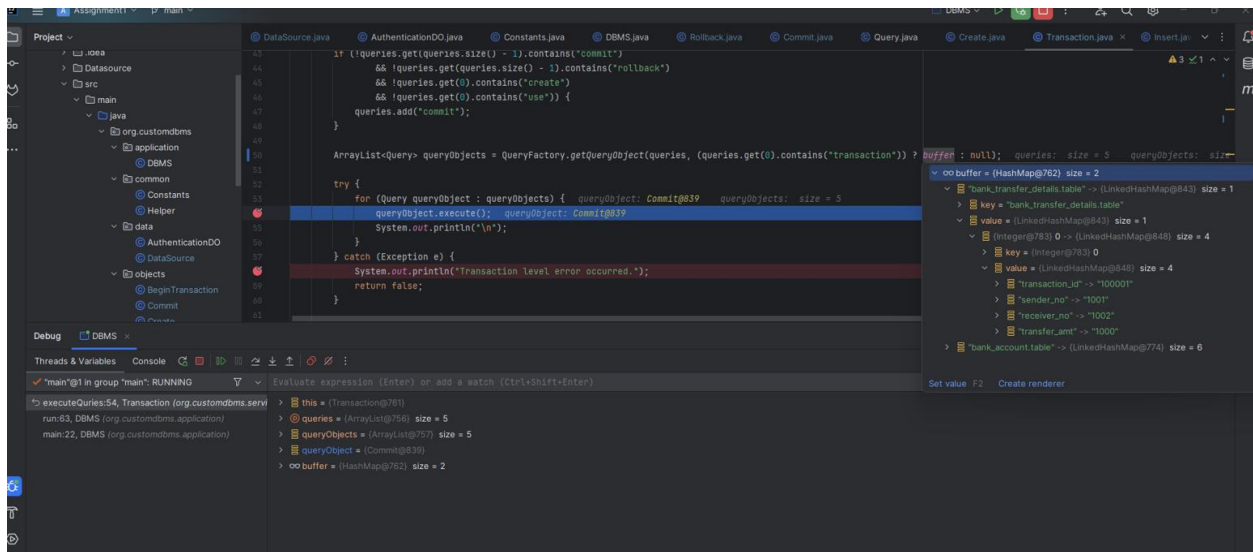
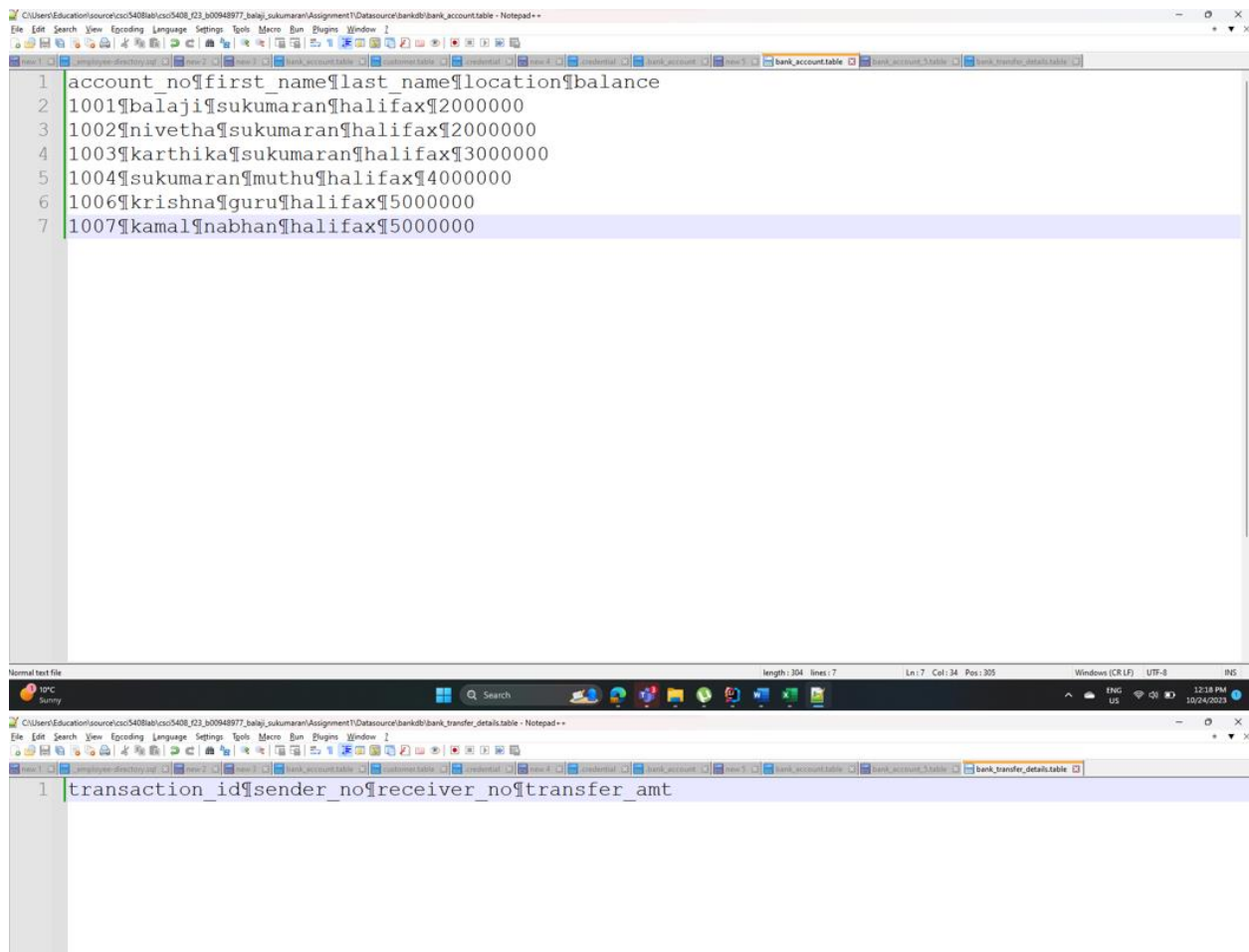


Figure 38: bank_transfer_details in buffer contains the transfer detail

Since, the transaction is not committed yet. The table files are not updated yet.



```
1 account_no|first_name|last_name|location|balance
2 1001|balaji|sukumaran|halifax|20000000
3 1002|nivetha|sukumaran|halifax|20000000
4 1003|karthika|sukumaran|halifax|30000000
5 1004|sukumaran|muthu|halifax|40000000
6 1006|krishna|guru|halifax|50000000
7 1007|kamal|nabhan|halifax|50000000
```

Figure 39: table file before committing.

Once the commit query, gets executed successfully. The control flow will try to persist the buffered changes once successful the table files get updated with buffered data.

The image shows a database query window and two table files. The query window displays a successful commit operation. Below it, two table files are shown: 'bank_account.table' and 'bank_transfer_detail.table'. Both tables have their respective data rows highlighted with red boxes.

```
QUERY WINDOW
-----
Type your query and press enter.
start transaction;
update bank_account set balance=balance-1000 where account_no=1001
update bank_account set balance=balance+1000 where account_no=1002
insert into bank_transfer_details (transaction_id, sender_no, receiver_no, transfer_amt) values (100001, 1001,1002,1000);
commit;
```

Updated successfully.

Updated successfully.

Record inserted successfully

Do you want to try again ?

1 account_no|first_name|last_name|location|balance
2 1001|balaji|sukumaran|halifax|1999000
3 1002|nivetha|sukumaran|halifax|2001000
4 1003|karthika|sukumaran|halifax|3000000
5 1004|sukumaran|muthu|halifax|4000000
6 1006|krishna|guru|halifax|5000000
7 1007|kamal|nabhan|halifax|5000000

1 transaction_id|sender_no|receiver_no|transfer_amt
2 100001|1001|1002|1000

Figure 40: Final state of the table file

Rollback flow and its intermediate state of the buffer.

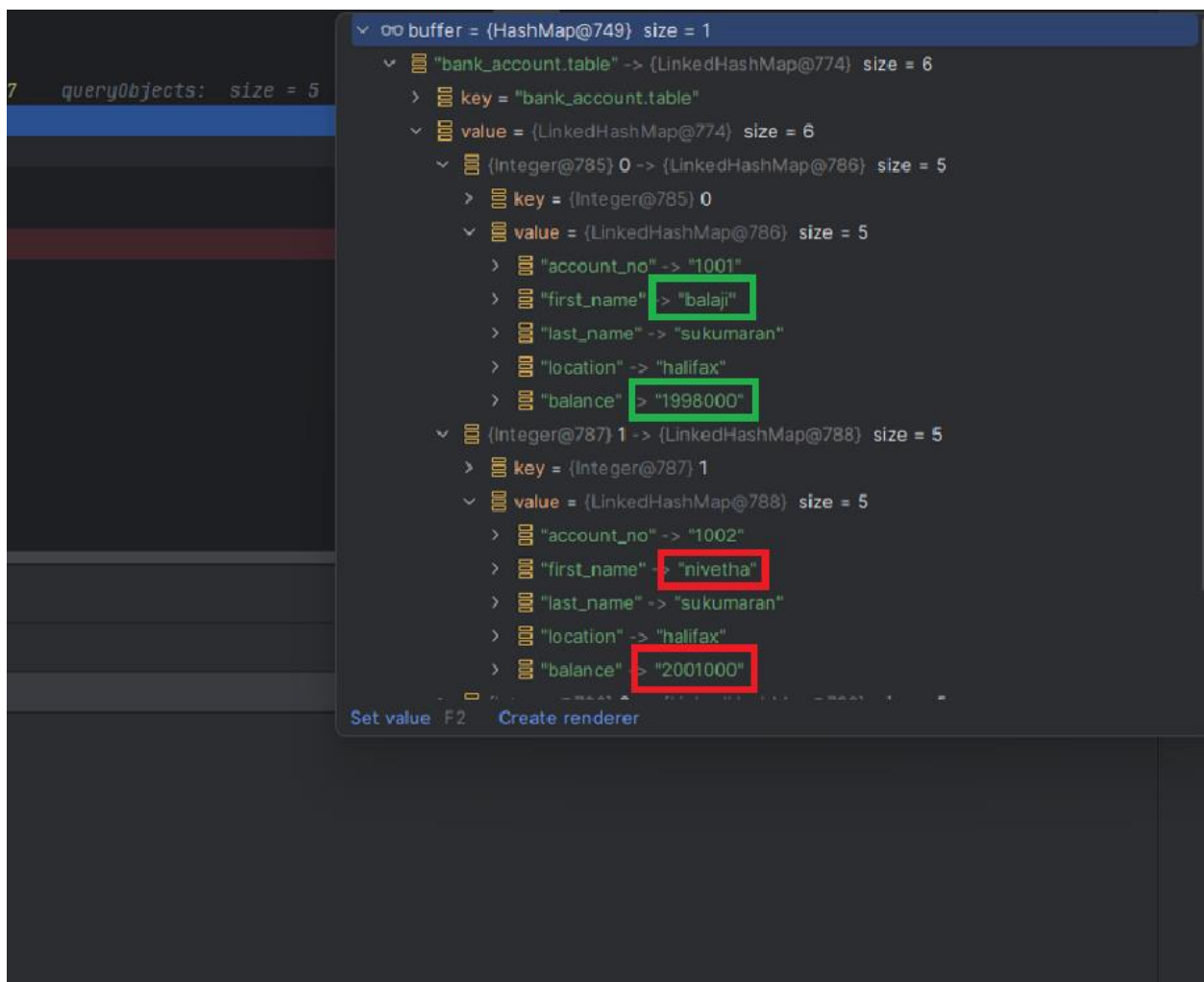
Executing the same query but with a rollback condition.



```
QUERY WINDOW
-----
Type your query and press enter.
start transaction;
update bank_account set balance=balance-1000 where account_no=1001
update bank_account set balance=balance+1000 where account_no=1002
insert into bank_transfer_details (transaction_id, sender_no, receiver_no, transfer_amt) values (100001, 1001,1002,1000);
rollback;
```

Figure 41: Transaction with rollback

After executing the first update, money deducted from Balaji's account. Changes are stored in the buffer.



```
7 queryObjects: size = 5
buffer = {HashMap@749} size = 1
  "bank_account.table" -> {LinkedHashMap@774} size = 6
    key = "bank_account.table"
    value = {LinkedHashMap@774} size = 6
      {Integer@785} 0 -> {LinkedHashMap@786} size = 5
        key = {Integer@785} 0
        value = {LinkedHashMap@786} size = 5
          "account_no" -> "1001"
          "first_name" -> "balaji"
          "last_name" -> "sukumaran"
          "location" -> "halifax"
          "balance" -> "1998000"
      {Integer@787} 1 -> {LinkedHashMap@788} size = 5
        key = {Integer@787} 1
        value = {LinkedHashMap@788} size = 5
          "account_no" -> "1002"
          "first_name" -> "nivetha"
          "last_name" -> "sukumaran"
          "location" -> "halifax"
          "balance" -> "2001000"
```

Figure 42: Buffer after first update

After executing the second update, Money credited to Nivetha's account. Changes are logged in buffer accordingly

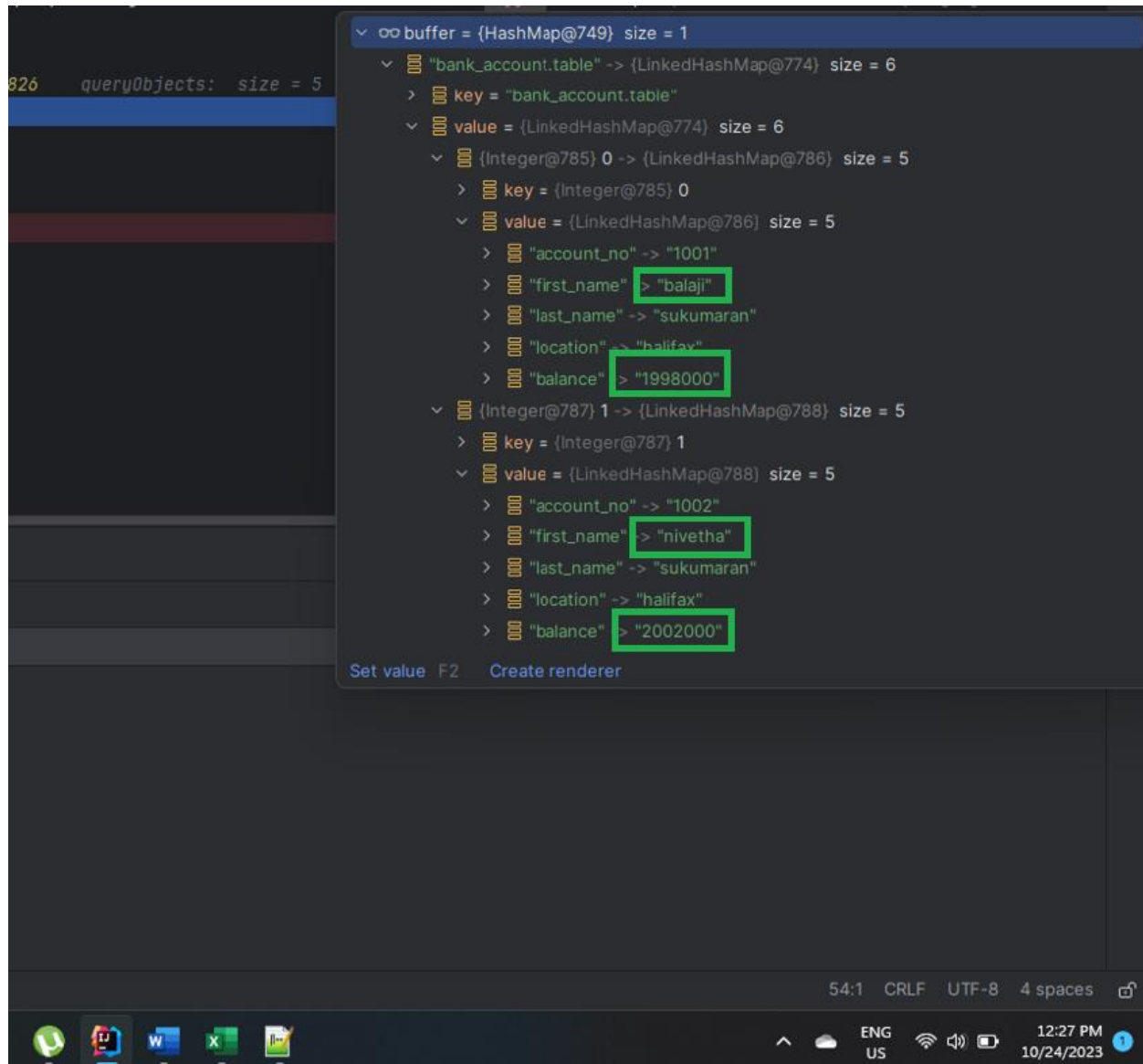


Figure 43: Buffer after second update

The flow will log the money transfer in the bank_account_transfer table and the buffer will hold two tables now and the new buffered bank_account_transfer table has the log.

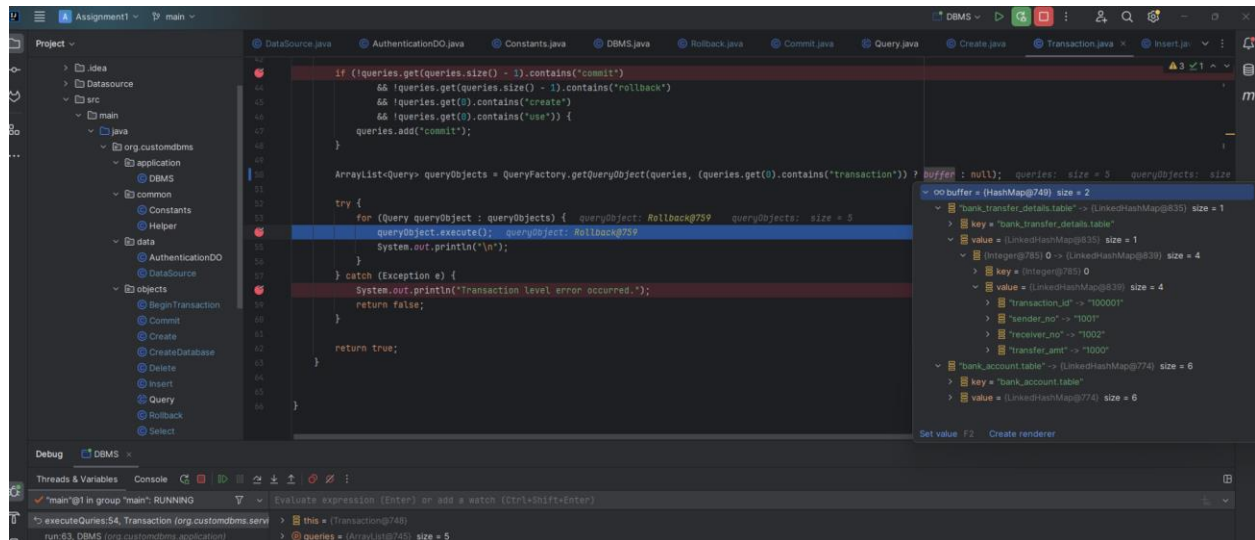


Figure 44: Buffer after logging the transaction.

Now, since the given query has rollback. The changes won't be reflected in the table file. The buffered records get cleared.

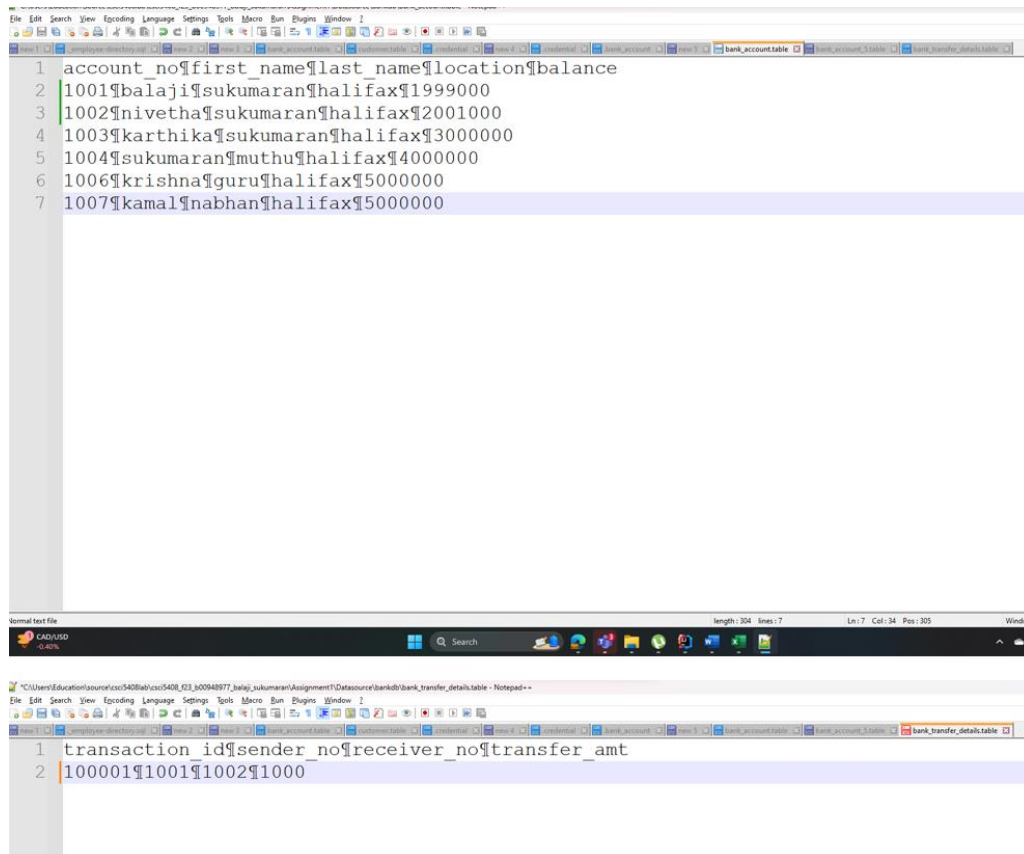


Figure 45: Final table file with old data