

December 6, 2021

PROJECT REPORT

GROUP 20

DATA MANAGEMENT, WAREHOUSING, ANALYTICS (CSCI 5408)

PROF: SAURABH DEY

TA: DIVYASHREE BANGALORE SUBBARAYA

Submitted by-
JANHAVI SONAWANE (B00881787)
KRISHNA SANJAYBHAI JADAV (B00884964)
PRACHI KABTIYAL (B00862464)
RAHUL SUNIL MOJE(B00874101)
RUHI RAJNISH TYAGI (B00872269)

PROJECT OVERVIEW

The project aims at development of a Database Management platform, capable of performing most of the database operations such as query execution, transaction and log management, reverse engineering, and export. The implementation includes validation of the query requested by the user and parsing of the data in the correct format for accurate processing. Parsing would consider the various data types and load the data in-memory for seamless handling of the modification and retrieval requests.

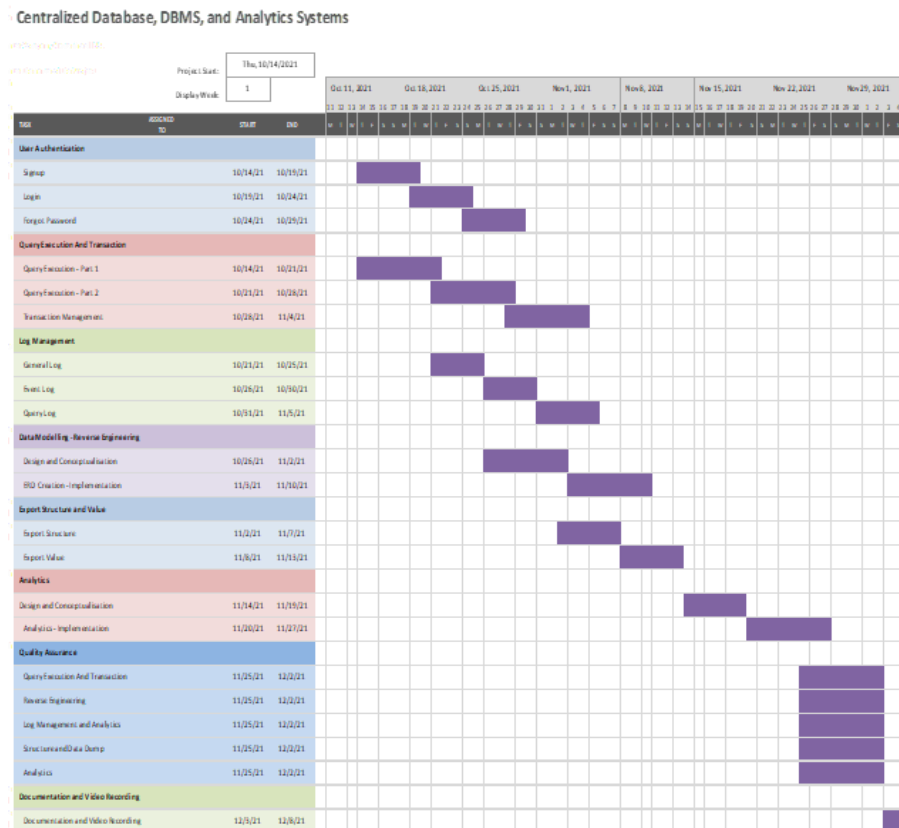
As part of the project, we would also be developing Reverse engineering feature to extract various tables, their relationships among each other along with the cardinalities. Moreover, the Analytics module will be providing user with the information like the number of valid queries done on an xyz database by an *abc* user. Log management module plays an integral role for execution of transactions and for the analysis.

In a nutshell, the whole idea of this project is to implement and understand the functioning of a typical centralized database, its management system and analytics operations.

1. Strength of Team Members and Contribution

Name	Strengths	Contribution
Janhavi Sonwane	Java SQL	Transaction Processing
Krishna Sanjaybhai Jadav	Java SQL	Export Data Model
Prachi Kabtiyal	Java SQL	Database Design Login and Security Query Implementation Log Management
Rahul Sunil Moje	Java SQL	Analytics
Ruhi Rajnish Tyagi	Java SQL	Transaction Processing

2. Gantt Chart



3. Module 1: Database Design

1. **Data Structure:** We have used a class to store details of the query under execution. The structure of the class includes-

- **Database:** Selected database is stored in a string variable.
- **Table Information:**
 - Table name
 - Hash map of columns to data types
 - List of primary keys
 - List of unique columns
 - List of not null columns
 - Hash map containing mapping of foreign key to the referenced table and column
- **Update Information:**
 - Column name to update
 - RHS value (new value)
- **Where condition:**
 - Where condition string
 - Column name (LHS)
 - Operator (like =, !=, is null, is not null, >, >=, <, <=)
 - RHS value
- **Query result**
 - List of Hash Map. Each Hash map contains information of each row.

```
public class QueryDetails {  
    String database;  
    String table_name;  
    HashMap<String,String> column_to_datatype;  
    List<HashMap<String,String>> values;  
    String where_clause;  
    String operator;  
    String lhs_column;  
    String rhs_value;  
    String set_lhs_column;  
    String set_rhs_value;  
    List<String> not_null_columns;  
    List<String> unique_columns;  
    List<String> primary_keys;  
    HashMap<String,HashMap<String,String>> column_to_referencetable_to_column;  
}
```

2. Valid Data Types:

- nvarchar
- integer
- float
- date (YYYY-MM-DD)

3. File Format for Persistent Storage:

- **.tsv:** Each value is stored tilde separated. First row of each file is tilde separated column names. The values start from the second row.

```
workspace > uvrWBPPSp3UIU > db1 > ≡ t1.tsv
1  name~adate~marks
2  prachi~2021-01-01~10.0
3  seema~2021-01-02~5.0
4
```

4. Data Dictionary/ Metadata

- **File format:** Same as persistent story. We have used a tilde separated file for creating meta data. Metadata file (in each database) stores the information of the tables and their properties.
- **Headers:** Headers are
 - "table_name",
 - "columns" (semi colon separated),
 - "data_types" (semi colon separated),
 - "primary_keys" (semi colon separated),
 - "unique_keys" (semi colon separated),
 - "not_null_keys" (semi colon separated),
 - "foreign_key"- Combination of
<column_name>#<referenced_table>#<referenced_column>. All combinations are joined with semi colon.

```
workspace > uvrWBPPSp3UIU > db1 > metadata > ≡ table_info.tsv
1  table_name~columns~data_types~primary_keys~unique_keys~not_null_keys~foreign_key
2  t1~name;adate;marks~nvarchar;date;float~name~~~
3  t2~bdate;name~date;nvarchar~name~~~bdate#t1#adate
4  t3~name;bmarks~nvarchar;float~name~~~bmarks#t1#marks
5  simple_table~cnvarchar;cinteger;cdate;cfloat~nvarchar;integer;date;float~~~~
6  simple_table2~cnvarchar;cinteger;cdate;cfloat~nvarchar;integer;date;float~~~~
7  simple_table3~cnvarchar;cinteger;cdate;cfloat~nvarchar;integer;date;float~~~~
```

4. Module 2: Query Implementation

The implementation of this module is divided into two parts:

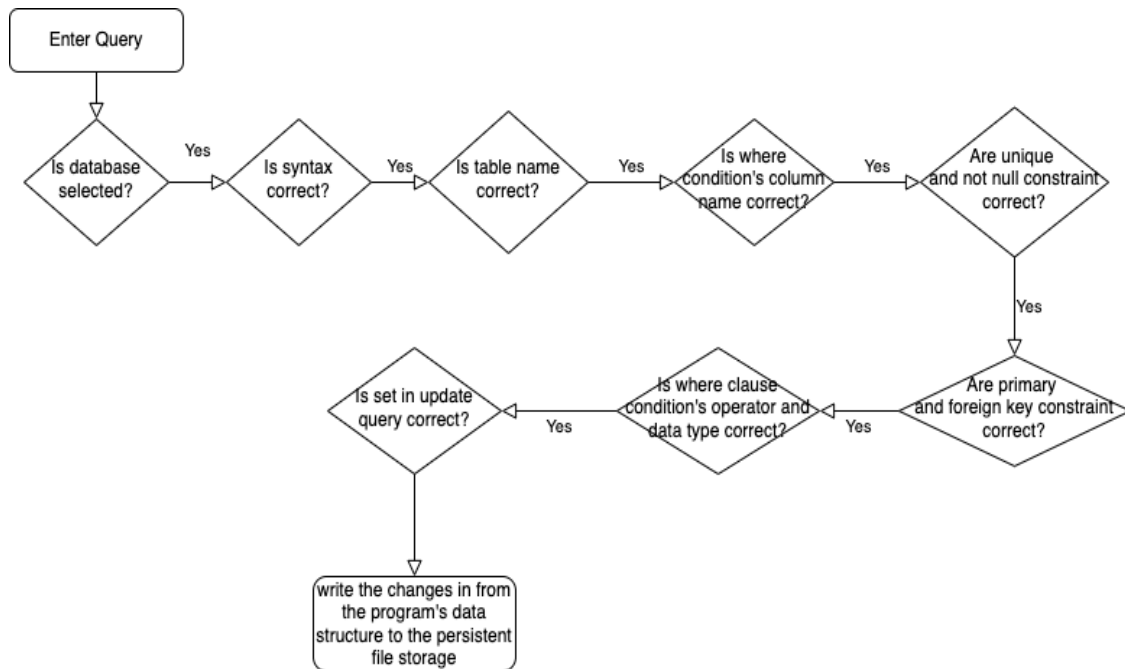
- **Query Validation:** In this sub-module, each query is parsed and validated based on some rules and criteria. Following are the validations for individual queries-
 - **Create Database:**
 - Syntax error: In case, entered query does not follow the sql format.
 - Database already exists: In case, a database with the same name already exists in the logged in user's workspace.
 - **Use Database:**
 - Syntax error: In case, entered query does not follow the sql format.
 - Database does not exist: In case, database is not present or was never created.
 - **Create Table:**
 - Syntax error: In case, entered query does not follow the sql format.
 - Table already exists in database: In case the user is trying to create a table with duplicate name.
 - Incorrect datatype <incorrect datatype name>. Correct data types are nvarchar, integer, float, date: In case, user gives a wrong data type in the input query.
 - Incorrect primary key= <incorrect column name>: In case, primary key does not match any of the column names provided in the query.
 - <table_name>: reference table does not exist: In case, user has referenced a foreign key to a table that does not exist.
 - <referenced_column>: referenced column does not exist in <referenced_table>: In case, foreign key is referencing a column of another table and that column is not present.
 - Incompatible datatypes of foreign key <key name> and <reference_column>: In case, the data type of foreign key and the referenced column does not match.
 - **Insert:**
 - Syntax error: In case, entered query does not follow the sql format.
 - Table not found: In case of incorrect table name that is not present in the workspace.
 - Primary key cannot be null
 - Not null constraint
 - Primary key constraint
 - Unique key constraint
 - Referenced table does not contain the foreign key

- **Select:**
 - Syntax error: In case, entered query does not follow the sql format.
 - Table not found
 - Invalid operator in where condition
 - Invalid operator for <datatype name> datatype. Valid operators are <list of valid operators>
 - Invalid column in where condition
- **Update:**
 - Syntax error: In case, entered query does not follow the sql format.
 - Table not found
 - Invalid operator in where condition
 - Invalid operator for <datatype name> datatype. Valid operators are <list of valid operators>
 - Invalid column in where condition
 - Invalid set clause: If “=” is not present
 - Invalid column in set clause
- **Delete:**
 - Syntax error: In case, entered query does not follow the sql format.
 - Table not found
 - Invalid operator in where condition
 - Invalid operator for <datatype name> datatype. Valid operators are <list of valid operators>
 - Invalid column in where condition
- **Drop Table:**
 - Syntax error: In case, entered query does not follow the sql format.
 - Table not found
 - Cannot delete the table since it is referenced by another table-<table_name>
- **Query Execution:**
 - **Create Database:**
 - A folder name “db name” is created inside the workspace of current logged in user.
 - **Use Database:**
 - Entered database is assigned to the selected database variable in the program.
 - Henceforth, every operation is performed inside the selected database folder.

- **Create Table:**
 - Entry of the table information is created in table_info.tsv file. It contains table name, columns, datatypes, primary key, foreign key, unique and not null constraints.
- **Insert:**
 - Values are parsed into right data format (string, integer, float, date) and inserted in <table_name>.tsv file.
 - After loading the data from query to program data structure (list of hash map), the entries are appended into the table.
- **Select:**
 - <table_name>.tsv is read as string and then converted in list of hash map.
 - Columns are shown according to asterisk (*) or column names mentioned in the query.
 - Program iterates over each row and apply filter condition. If the condition matches, the row is inserted to another list “filtered_rows”.
 - “filtered_rows” are printed on console using java.util.formatter to display the result in table format.
 - Total number of rows is also displayed.
- **Update:**
 - <table_name>.tsv is read as string and then converted in list of hash map.
 - Program iterates over each row and apply filter condition. If the condition matches, the column value is updated to the one mentioned in the query.
 - The updated list is inserted to the tsv file.
- **Delete:**
 - <table_name>.tsv is read as string and then converted in list of hash map.
 - Program iterates over each row and apply filter condition. If the condition does not match, the row is inserted to another list “filtered_rows”.
 - The program writes the filtered list to the table’s persistent storage.
- **Drop Table:**
 - Drop table query removes table’s entry from the metadata file.
 - The query deletes the persistent storage .tsv file from the database.

Project video demonstrates execution of all queries. It also shows the validations in case an error is encountered while parsing the query.

Flow Diagram



5. Module 3: Transaction Processing

Transactions support only DML operations [1] i.e., insert, delete, and update queries on the database. The transactions must comply with A.C.I.D. properties [2], therefore the transactions are not performed directly onto the database.

Data structures used:

- List: to store the queries in a transaction.
- LinkedHashMap: to store the output of the queries in the transaction (in-memory).

Figure 3.1 demonstrates the flow of the program:

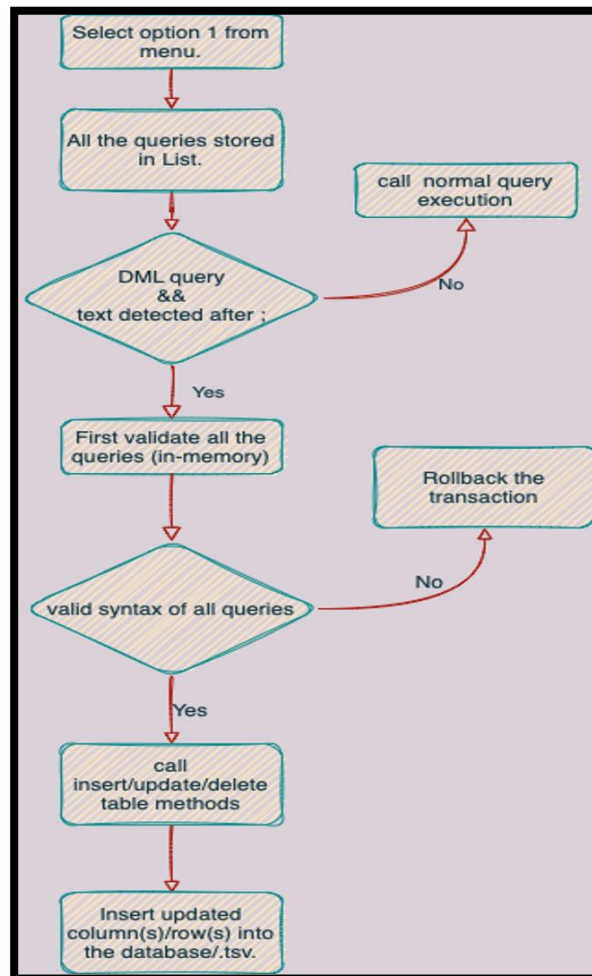
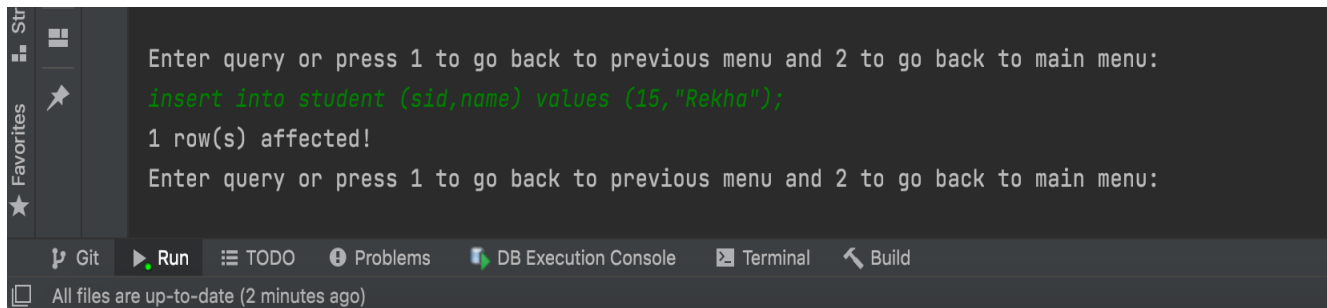


Figure 3.1 The flow of the program for performing transaction on the database.

After successfully login into the system and selecting the option to write queries, the following tests are running to show how and when transactions are detected by the program:

TEST 1: One query is not a transaction (despite being insert, delete, or update).

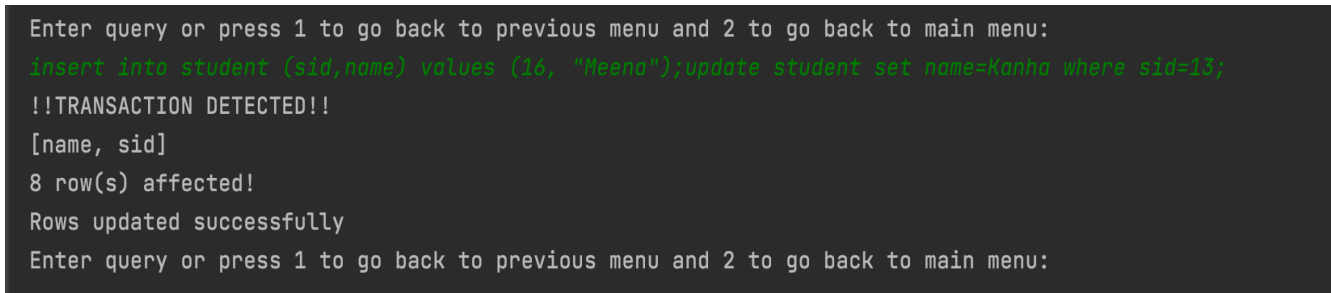


```

Enter query or press 1 to go back to previous menu and 2 to go back to main menu:
insert into student (sid,name) values (15,"Rekha");
1 row(s) affected!
Enter query or press 1 to go back to previous menu and 2 to go back to main menu:
  
```

Figure 3.2 The screenshot shows what is NOT a transaction.

TEST 2: Two or more than two queries detects a transaction.



```

Enter query or press 1 to go back to previous menu and 2 to go back to main menu:
insert into student (sid,name) values (16, "Meena");update student set name=Kanha where sid=13;
!!TRANSACTION DETECTED!!
[name, sid]
8 row(s) affected!
Rows updated successfully
Enter query or press 1 to go back to previous menu and 2 to go back to main menu:
  
```

Figure 3.3 The screenshot shows what IS a transaction.



1	name~sid
2	Tanvi~1
3	Ruhi~2
4	"Manish"~11
5	Roma~12
6	Kanha~13
7	Kanha~14
8	Rekha~15
9	Meena~16

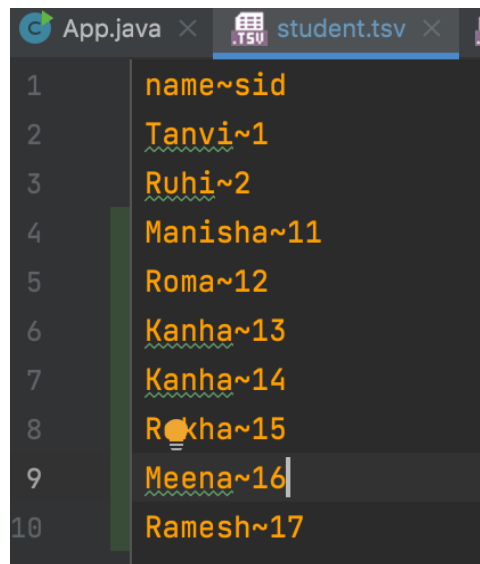
Figure 3.4 The screenshot shows changes in the database after all the queries are validated against the syntax.

TEST 3: Different number of queries in ascending order (Up to five).

THREE QUERIES:

```
Enter query or press 1 to go back to previous menu and 2 to go back to main menu:
delete from student where sid=1;update student set name=Manisha where sid=11;insert into student (sid,name) values (17,"Ramesh");
!!TRANSACTION DETECTED!!
9 row(s) affected!
Rows updated successfully
1 row(s) affected!
Enter query or press 1 to go back to previous menu and 2 to go back to main menu:
```

Figure 3.5 The screenshot shows a transaction with THREE different queries.



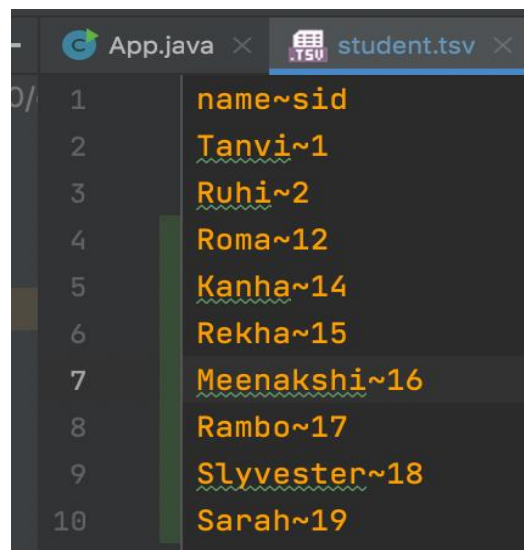
	name~sid
1	Tanvi~1
2	Ruhi~2
3	Manisha~11
4	Roma~12
5	Kanha~13
6	Kanha~14
7	Roha~15
8	Meena~16
9	Ramesh~17
10	

Figure 3.6 The screenshot shows changes in the database after all the queries are validated against the syntax.

FOUR QUERIES:

```
Enter query or press 1 to go back to previous menu and 2 to go back to main menu:
insert into student (sid,name) values (19,"Sarah");update student set name=Rambo where sid=17;delete from student where sid=11;delete from student where sid=13;
!!TRANSACTION DETECTED!!
[name, sid]
Rows updated successfully
10 row(s) affected!
9 row(s) affected!
Enter query or press 1 to go back to previous menu and 2 to go back to main menu:
```

Figure 3.7 The screenshot shows a transaction with FOUR different queries.



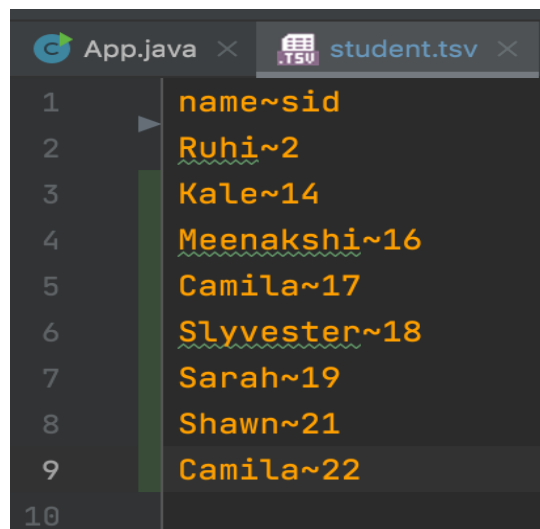
	name~sid
1	Tanvi~1
2	Ruhi~2
3	Roma~12
4	Kanha~14
5	Rekha~15
6	Meenakshi~16
7	Rambo~17
8	Sylvester~18
9	Sarah~19
10	

Figure 3.8 The screenshot shows changes in the database after all the queries are validated against the syntax.

FIVE QUERIES:

```
Enter query or press 1 to go back to previous menu and 2 to go back to main menu:
insert into student (sid,name) values (22,"Camila");update student set name=Shawn where sid=21;update student set name=Kale where sid=14;delete from student where sid=12;
!!TRANSACTION DETECTED!!
[name, sid]
Rows updated successfully
[name, sid]
Rows updated successfully
8 row(s) affected!
8 row(s) affected!
Rows updated successfully
Enter query or press 1 to go back to previous menu and 2 to go back to main menu:
|
```

Figure 3.9 The screenshot shows a transaction with FIVE different queries.



	name~sid
1	
2	<u>Ruhi~2</u>
3	<u>Kale~14</u>
4	<u>Meenakshi~16</u>
5	<u>Camila~17</u>
6	<u>Sylvester~18</u>
7	<u>Sarah~19</u>
8	<u>Shawn~21</u>
9	<u>Camila~22</u>
10	

Figure 3.10 The screenshot shows changes in the database after all the queries are validated against the syntax.

6. Module 4: Log Management

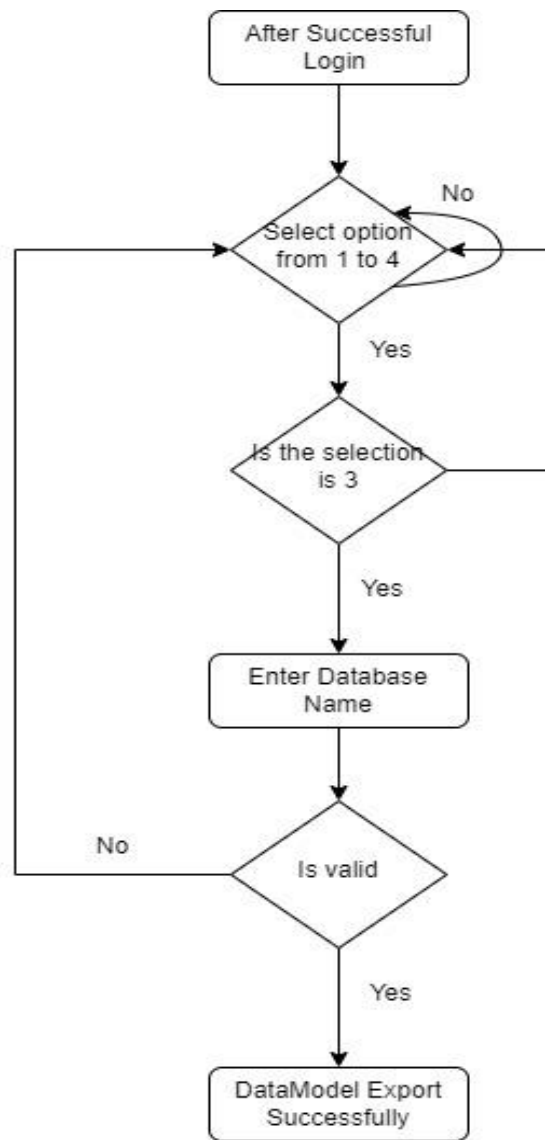
- A log file (system.log) is created in every user's workspace.
- Logs are stored in the JSON format.
- Each json object is an entry of database operation-
 - Login
 - Create database query
 - Use database query
 - Create table query
 - Insert into table query
 - Select query
 - Update table query
 - Delete from table query
 - Drop table query
- Properties for each log (json entry) are-
 - Timestamp
 - Username
 - Query type: update/insert/select etc.
 - Query: Exact query entered by the user
 - Result: Result of the executed query
 - Error: Error occurred while executing the query
- Program is segregated into two components-
 - Model: A model class "Log.java" is a structure to store the values of the properties
 - Writer: A "LogWriter.java" class loads the system.log file in the memory and appends an object of "Log" class in the file.

```

system.logs
workspace > uvrWBPPSp3UIU > logs > system.logs
41 }, {
42   "result": "Using database db1",
43   "database": "db1",
44   "query": "use database db1",
45   "query_type": "USE_DATABASE",
46   "user": "P",
47   "table_name": "",
48   "timestamp": "2021-12-02T10:40:33.326530"
49 }, {
50   "result": "Table created successfully",
51   "database": "db1",
52   "query": "create table table1 (column1 nvarchar, column2 integer, column3 float, column4 date);",
53   "query_type": "CREATE_TABLE",
54   "user": "P",
55   "table_name": "table1",
56   "timestamp": "2021-12-02T10:41:20.696291"
57 }, {
58   "result": "P logged in successfully",
59   "database": "login",
60   "query": "",
61   "query_type": "LOGIN",
62   "user": "P",
63   "table_name": "",
64   "timestamp": "2021-12-02T10:47:24.323241"

```

7. Module 5: Data Modelling- Reverse Engineering



- After successful login, the system provides four options,


```
Choose from following options (1/2/3/4):
1. Write Queries
2. Export
3. Data Model
4. Analytics
3
Enter Database Name
dummydatabase

***** Error Occurred *****

Database dummydatabase not exists.

***** End *****
```

- For Data model export, user has to select option 3.
- After that the system would ask for database name that user want to export.
- Before exporting, system checks the database exists or not.

```
Choose from following options (1/2/3/4):
1. Write Queries
2. Export
3. Data Model
4. Analytics
3
Enter Database Name
dbtest
DataModel Export Successfully!! Check your workspace
```

- Once the user enters the correct database name, the system will create a ERExport directory in the user workspace. The folder contains the text file below,

dbtest_1638894598431.txt - Notepad

File Edit Format View Help

2021/12/07 12:29:58

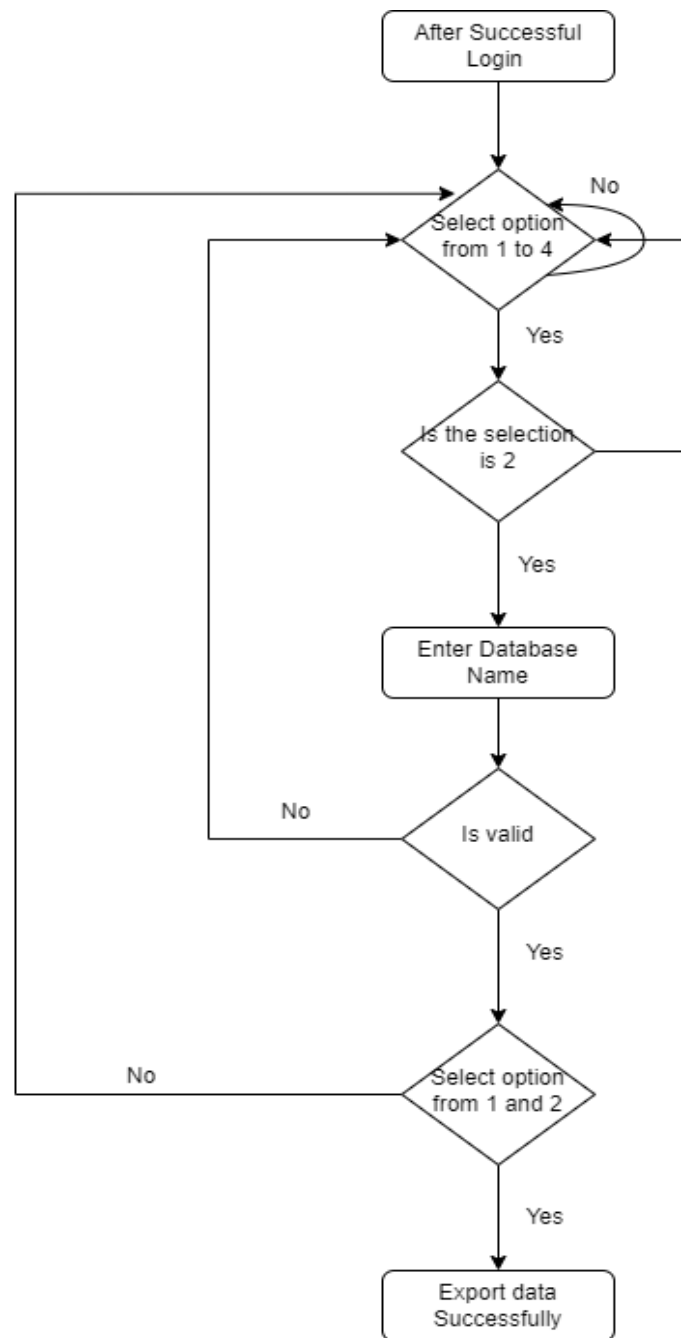
Database Name: dbtest

Table Name	country				
ColumnName	ColumnType	PrimaryKey	NotNull	Unique	ForeignKey
coname	nvarchar	NO	NO	NO	NO
coid	integer	YES	YES	NO	NO

Table Name	city				
ColumnName	ColumnType	PrimaryKey	NotNull	Unique	ForeignKey
cname	nvarchar	NO	NO	NO	NO
coid	integer	NO	NO	NO	country(coid)
cid	integer	YES	YES	NO	NO

Table Name	t1				
ColumnName	ColumnType	PrimaryKey	NotNull	Unique	ForeignKey
col2	nvarchar	NO	NO	NO	NO
col3	integer	NO	NO	NO	NO
col1	integer	NO	NO	NO	NO

8. Module 6: Export Structure & Value



- After successful login, the system provides four options,

```
Choose from following options (1/2/3/4):
1. Write Queries
2. Export
3. Data Model
4. Analytics
2
Enter Database Name
dummydatabase

***** Error Occurred *****

Database dummydatabase not exists.

***** End *****
```

- For Export, the user has to select option 2.
- After that the system would ask for database name that user want to export.
- Before exporting, system checks the database exists or not.
- Once the user enters the correct database name, the system would ask the user for the options below,

```
Choose from following options (1/2/3/4):
1. Write Queries
2. Export
3. Data Model
4. Analytics
2
Enter Database Name
dbtest
Select option 1/2:
1. Export Structure
2. Export Structure And Value
1
Export Successfully!! Check your workspace
```

- Once the user selects any option, the system will create an Export directory in the user workspace.
- If user selects Export Structure option then the system will generate the SQLDUMP file of entered database as below,

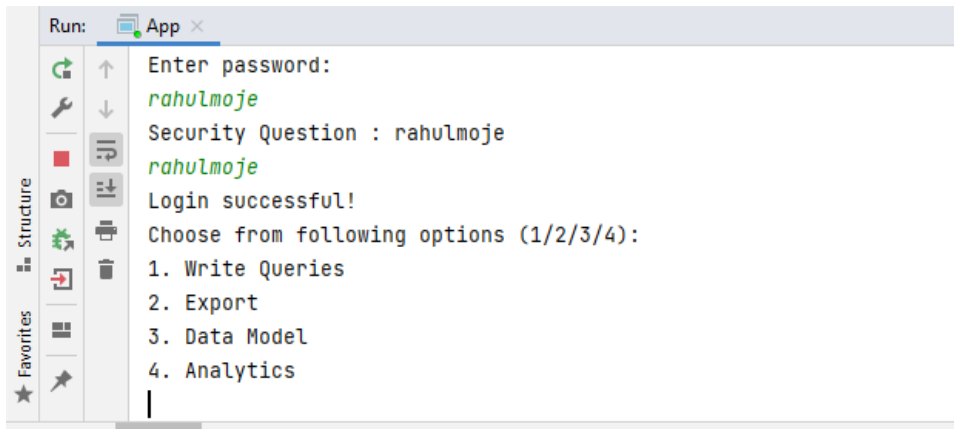
```
dbtest_1638897007450.sql X
workspace > uvwvCUa.DW > dbtest > Export > dbtest_1638897007450.sql
1 CREATE DATABASE dbtest;
2 USE dbtest;
3 CREATE TABLE country(coname nvarchar,coid integer NOT NULL,PRIMARY KEY (coid));
4 CREATE TABLE city(cname nvarchar,coid integer,cid integer NOT NULL,PRIMARY KEY (cid),FOREIGN KEY (coid) REFERENCES country(coid));
5 CREATE TABLE ti(col2 nvarchar,col3 integer,col1 integer);
6
```

- If the user selects Export Structure and Value option then the system will generate the SQLDUMP file of entered database as below,

```
dbtest_1638897353851.sql X
workspace > uvwvCUa.DW > dbtest > Export > dbtest_1638897353851.sql
1 CREATE DATABASE dbtest;
2 USE dbtest;
3 CREATE TABLE country(coname nvarchar,coid integer NOT NULL,PRIMARY KEY (coid));
4 INSERT INTO country(coname,coid) VALUES('India',1);
5 INSERT INTO country(coname,coid) VALUES('Canada',2);
6 CREATE TABLE city(cname nvarchar,coid integer,cid integer NOT NULL,PRIMARY KEY (cid),FOREIGN KEY (coid) REFERENCES country(coid));
7 INSERT INTO city(cname,coid,cid) VALUES('Surat',1,1);
8 INSERT INTO city(cname,coid,cid) VALUES('Halifax',2,2);
9 CREATE TABLE ti(col2 nvarchar,col3 integer,col1 integer);
10
```

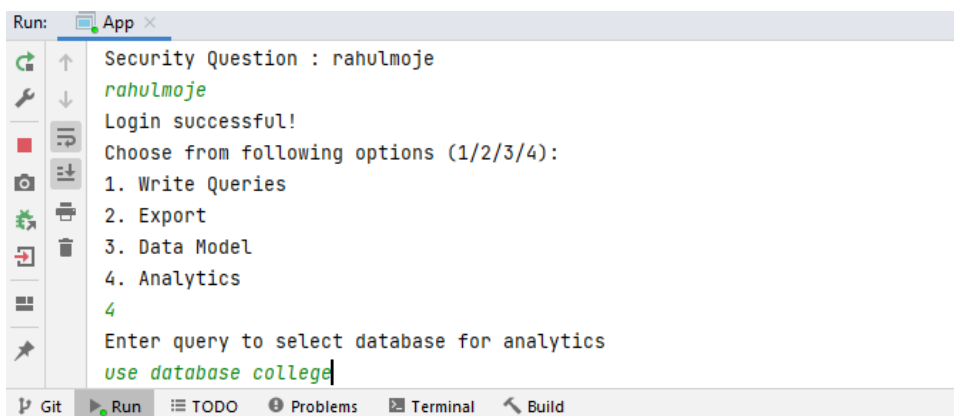
9. Module 7: Analytics

The analytics module provides different statistics to the user. In order to find analytics, the user has to select option number 4 after logging into the database.



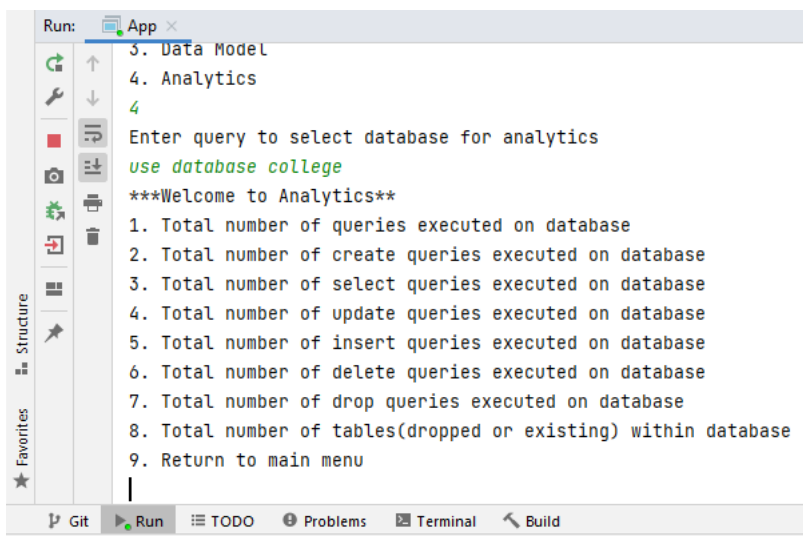
```
Run: App x
Enter password:
rahulmoje
Security Question : rahulmoje
rahulmoje
Login successful!
Choose from following options (1/2/3/4):
1. Write Queries
2. Export
3. Data Model
4. Analytics
|
```

Once the user selects option 4, the user has to enter on which database analytics has to be provided



```
Run: App x
Security Question : rahulmoje
rahulmoje
Login successful!
Choose from following options (1/2/3/4):
1. Write Queries
2. Export
3. Data Model
4. Analytics
4
Enter query to select database for analytics
use database college
```

If the database name is validated successfully, analytics menu is displayed



```
Run: App x
3. Data Model
4. Analytics
4
Enter query to select database for analytics
use database college
***Welcome to Analytics**
1. Total number of queries executed on database
2. Total number of create queries executed on database
3. Total number of select queries executed on database
4. Total number of update queries executed on database
5. Total number of insert queries executed on database
6. Total number of delete queries executed on database
7. Total number of drop queries executed on database
8. Total number of tables(dropped or existing) within database
9. Return to main menu
|
```

We are providing a total of eight analytics options to the user. The user will be able to find out the total number of queries executed on the database, total create queries, total select queries, total update queries, total insert queries, total delete queries, total drop queries and finally total number of tables (dropped or existing) within database. Suppose the user enters option 1, then the analytics module will find out total number of queries executed in the database

```

Run: App
7. Total number of drop queries executed on database
8. Total number of tables(dropped or existing) within database
9. Return to main menu
1. Total number of queries executed by RAHULMOJE is 7
***Welcome to Analytics**
1. Total number of queries executed on database
2. Total number of create queries executed on database
3. Total number of select queries executed on database
4. Total number of update queries executed on database
5. Total number of insert queries executed on database
6. Total number of delete queries executed on database
7. Total number of drop queries executed on database
8. Total number of tables(dropped or existing) within database
9. Return to main menu

```

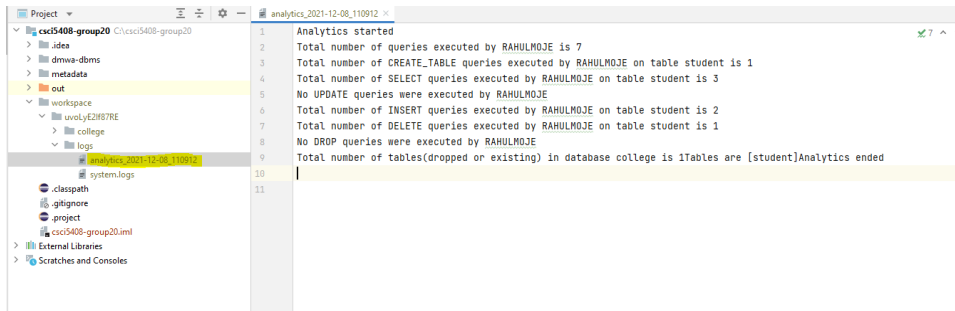
The analytics module uses system.logs to generate these results. Every event in the system.log is checked against the analytics option and if it matches, then a counter is incremented. A HashMap is also used to store counts of different type of queries. For example - suppose we need to find out the total number of create queries, the analytics module will scan through system.logs individual events, check if the database is correct, check if the log is related to create query and if true, add it to count. Finally, the total count is displayed to the user. A sample event for a create query in system.log looks like this

```

system.logs
1 [{"result":"RAHULMOJE logged in successfully","database":"login","query":"","query_type":"LOGIN",
  "user":"RAHULMOJE","table_name":"","timestamp":"2021-12-08T10:58:24.880220400"},{"result":"Using database
  college","database":"college","query":"use database college","query_type":"USE_DATABASE","user":"RAHULMOJE",
  "table_name":"","timestamp":"2021-12-08T10:59:17.527093400"},{"result":"Table created successfully",
  "database":"college","query":"create table student (name nvarchar, id integer)","query_type":"CREATE_TABLE",
  "user":"RAHULMOJE","table_name":"student","timestamp":"2021-12-08T10:59:43.814705500"},{"result":"1 row(s)
  inserted successfully","database":"college","query":"insert into student (name,id) values ('rahul',1)",
  "query_type":"INSERT","user":"RAHULMOJE","table_name":"student","timestamp":"2021-12-08T11:01:00.548628500"},
  {"result":"1 row(s) inserted successfully","database":"college","query":"insert into student (name,id) values
  ('sanjay',2)","query_type":"INSERT","user":"RAHULMOJE","table_name":"student","timestamp":"2021-12-08T11:01:33.
  870027700"},{"result":"1 row(s)","database":"college","query":"select * from student where id=1",
  "query_type":"SELECT","user":"RAHULMOJE","table_name":"student","timestamp":"2021-12-08T11:01:59.499567100"},
  {"result":"1 row(s)","database":"college","query":"select * from student where id=2","query_type":"SELECT",
  "user":"RAHULMOJE","table_name":"student","timestamp":"2021-12-08T11:02:50.491230700"},{"result":"0 row(s)",
  "database":"college","query":"select * from student where id=3","query_type":"SELECT","user":"RAHULMOJE",
  "table_name":"student","timestamp":"2021-12-08T11:03:12.140172800"},{"result":"1 rows deleted successfully",
  "database":"college","query":"delete from student where name='sanjay'", "query_type":"DELETE","user":"RAHULMOJE",
  "table_name":"student","timestamp":"2021-12-08T11:03:44.680242300"},{"result":"RAHULMOJE logged in successfully",
  "database":"login","query":"","query_type":"LOGIN","user":"RAHULMOJE","table_name":"","
  timestamp":"2021-12-09T16:19:32.879313900"},{"result":"RAHULMOJE logged in successfully","database":"login",
  "query":"","query_type":"LOGIN","user":"RAHULMOJE","table_name":"","timestamp":"2021-12-09T16:49:43.816995100"}]

```

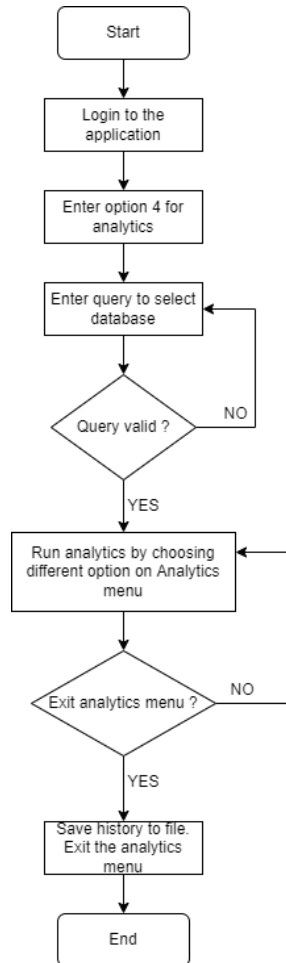
After the user exits the analytics menu, a file is also saved into the workspace folder that tracks what options the user chose and what were the results. The file pattern is analytics_ddMMyyHHmmss.txt



The screenshot shows an IDE with a project named 'csc5408-group20'. The console window displays the following analytics results:

```
1 Analytics started
2 Total number of queries executed by RAHULMOJE is 7
3 Total number of CREATE_TABLE queries executed by RAHULMOJE on table student is 1
4 Total number of SELECT queries executed by RAHULMOJE on table student is 3
5 No UPDATE queries were executed by RAHULMOJE
6 Total number of INSERT queries executed by RAHULMOJE on table student is 2
7 Total number of DELETE queries executed by RAHULMOJE on table student is 1
8 No DROP queries were executed by RAHULMOJE
9 Total number of tables(dropped or existing) in database college is 1Tables are [student]Analytics ended
10
11
```

Flowchart -



10. Module 8: User Interface & Login Security

- User is prompted with two options- 1. Login and 2. Register
- One selecting “Login”, user has to enter “username”, “password”, “security question’s answer”.
- If username or password is wrong, a message is shown on console “Incorrect username/password. Please try again” and user is redirected back to the previous menu of (Login and Register).

```
Choose from following options (1/2):
1. Login
2. Register
1

***** Login *****

Enter username:
prachi
Enter password:
wrongpassword
Incorrect username/password. Please try again
Choose from following options (1/2):
1. Login
2. Register
```

- On selecting “Register”, user is asked to enter a username, password, a security question, and its answer. Following validations are run before registering a user-
 - Username already exists
 - Input should be in alpha numeric format

```
Choose from following options (1/2):
1. Login
2. Register
2

***** Register *****

Enter username:
p
Enter password:
p
Enter a security question:
incorrect
Answer of the security question:
incorrect
Username already exists. Please try again.
Enter username:
prachi kabtiyal
Input should be in alpha numeric format.
Enter username:
```

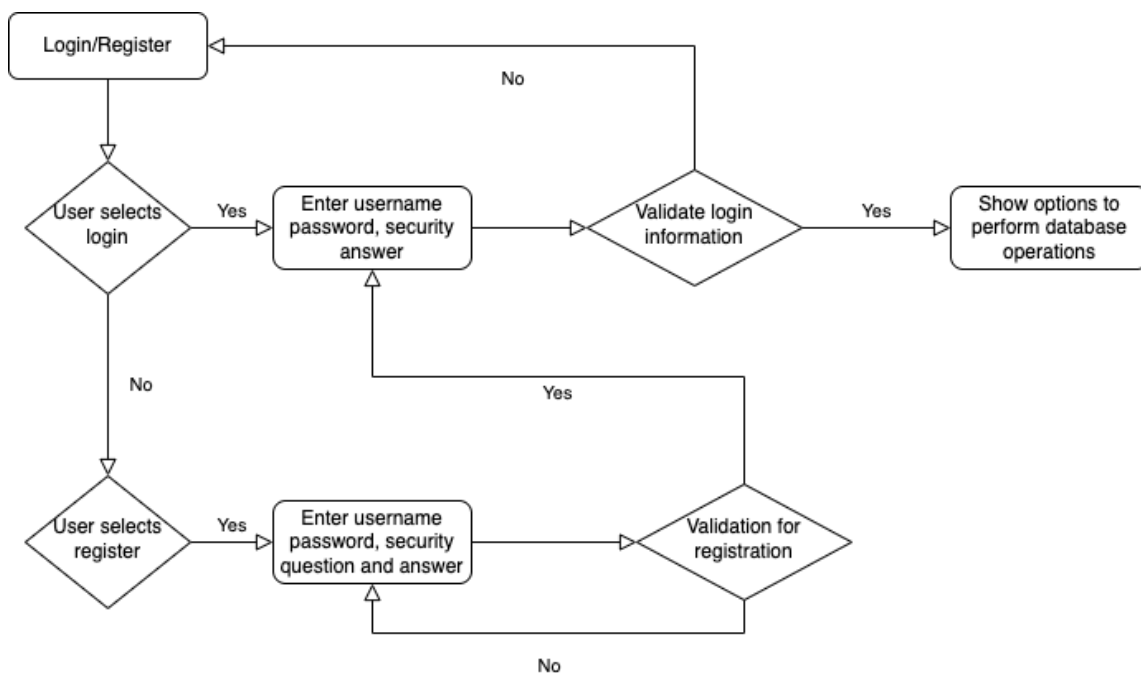
- Registered users’ information is stores in a metadata file named “USER_PROFILE.txt”.
- The file contains json object. Each key is an encrypted username, and its value is another json containing password, security question and its answer.
- Username and password are stored in an encrypted format. Data Encryption Standard (DES)

is used to encrypt the password and username.

- Library used for encryption- org.apache.commons.codec.digest.Crypt
- Once the user has entered correct username, password, and security question's answer, they are given 4 options-
 - Write Queries
 - Export
 - Data Model
 - Analytics

```
Login successful!  
Choose from following options (1/2/3/4):  
1. Write Queries  
2. Export  
3. Data Model  
4. Analytics  
1  
Select option 1/2/3:  
1. Query  
2. See Query Guide  
█
```

Flow Diagram



11. Meeting logs

meetingAttendanceReport (3)

Meeting	Summary							
Total Number of	Participants		3					
Meeting	Title	DM	WA group project		Command line interface			
Meeting	Start Time		me 11/9/2021, 5:00:28 PM					
Meeting	End Time		11/9/2021, 5:25:32 PM					
Meeting	Id	354d4125-cd65-4630-bfaa-d0475462a1fd						
Full Name	email	Join Time	Leave Time	Duration	Email	Role	Participant	
Prachi K	abtiyal	11/9/2021, 5:00:28 PM	11/9/2021, 5:25:32 PM	8 PM	11/9/2021, 5:25:32 PM	21, 5:25:32 PM	24 PM	2
Rahul Su	nil Moje	11/9/2021, 5:00:28 PM	11/9/2021, 5:25:32 PM	5 7 PM	11/9/2021, 5:25:32 PM	021, 5:25:32 PM	:2 6 PM	
Krishna	Sanjaybh	ai Jadav	11/9/2021, 5:00:28 PM	1, 5:04:32 PM	46 PM	M 11/9/2021, 5:25:32 PM	21, 5:25:32 PM	, 5:2

meetingAttendanceReport(DMWA KT)

Meeting Summary											
Total Number of Participants 2											
Meeting Title DMWA KT											
Meeting Start Time 11/28/2021				9:29:47 AM							
Meeting End Time 11/28/2021				10:05:03 AM							
Meeting Id d16ce27f-5959-410d-a1c2-05dac0114b96											
Full Name Join Time Leave Time Duration Email Role Participant ID (UPN)											
Prachi Kabtiyal 11/28/2021				9:29:47 AM	11/28/2021	10:05:00 AM	35m 13s	pr522601@dal.ca	Organizer	pr522601@dal.ca	
Janhavi Sonawane 11/28/2021				9:29:56 AM	11/28/2021	10:05:03 AM	35m 7s	jn476959@dal.ca	Presenter	jn476959@dal.ca	

meetingAttendanceReport

Meeting Summary																										
Total Number of Participants 5																										
Meeting Title																										
Meeting Start Time		12/2/2021					6:00:29 PM																			
Meeting End Time		12/2/2021					6:22:52 PM																			
Meeting Id		da4fc506-a905-4325-8a6d-0393cd764603																								
Full Name							Join Time		Leave Time		Duration		Email		Role		Participant ID (UPN)									
Prachi Kabtiyal							12/2/2021						6:00:29 PM		12/2/2021		6:22:50 PM		22m 20s		pr522601@dal.ca		Organizer		pr522601@dal.ca	
Krishna Sanjaybhai Jadav							12/2/2021						6:00:52 PM		12/2/2021		6:22:47 PM		21m 54s		kr447707@dal.ca		Presenter		kr447707@dal.ca	
Janhavi Sonawane							12/2/2021						6:00:52 PM		12/2/2021		6:22:52 PM		21m 59s		jn476959@dal.ca		Presenter		jn476959@dal.ca	
Rahul Sunil Moje							12/2/2021						6:01:12 PM		12/2/2021		6:22:51 PM		21m 38s		rh547954@dal.ca		Presenter		rh547954@dal.ca	
Ruhi Tyagi							12/2/2021						6:01:52 PM		12/2/2021		6:22:51 PM		20m 59s		rh679297@dal.ca		Presenter		rh679297@dal.ca	

meetingAttendanceReport (2)

Meeting Summary										
Total Number of Participants				6						
Meeting Title										
Meeting Start Time				12/9/2021		5:10:08 PM				
Meeting End Time				12/9/2021		5:37:25 PM				
Meeting Id				a5165392-1b37-4cfa-adf7-052405c7fd78						
Full Name	Join Time	Leave Time	Duration	Email	Role	Participant ID (UPN)				
Prachi Kabtiyal	12/9/2021					5:10:08 PM	12/9/2021	5:37:19 PM	27m 11s	pr522601@dal.ca Organizer pr522601@dal.ca
Ruhi Tyagi	12/9/2021					5:10:17 PM	12/9/2021	5:37:21 PM	27m 4s	rh679297@dal.ca Presenter rh679297@dal.ca
Janhavi Sonawane	12/9/2021					5:10:35 PM	12/9/2021	5:37:25 PM	26m 49s	jn476959@dal.ca Presenter jn476959@dal.ca
Krishna Jadav	12/9/2021					5:11:05 PM	12/9/2021	5:37:23 PM	26m 17s	kr447707@dal.ca Presenter kr447707@dal.ca
Rahul Sunil Moje	12/9/2021					5:14:52 PM	12/9/2021	5:37:21 PM	22m 29s	rh547954@dal.ca Presenter rh547954@dal.ca
Saurabh Dey	12/9/2021					5:18:46 PM	12/9/2021	5:37:19 PM	18m 32s	sr695037@dal.ca Presenter sr695037@dal.ca

References

- [1] *"Transactions apply only to the Data Manipulation Language (DML) portion of the SQL language (such as INSERT, UPDATE, and DELETE). Transactions do not apply to the Data Control Language (DCL) or Data Definition Language (DDL) portions (such as CREATE, DROP, ALTER, and so on) of the SQL language. DCL and DDL commands always force a commit, which in turn commits everything done before them."*
"Transactions", *Docs.oracle.com*, 2021. [Online]. Available:
https://docs.oracle.com/cd/B19306_01/win.102/b14308/cpp00018.htm. [Accessed December 9, 2021].
- [2] *"The ACID properties are meant for the transaction that goes through a different group of tasks, and there we come to see the role of the ACID properties."*
(ACID Properties in DBMS - javatpoint, 2021),www.javatpoint.com. 2021. *ACID Properties in DBMS javatpoint*. [online] Available at: <<https://www.javatpoint.com/acid-properties-in-dbms>> [Accessed December 9, 2021].
- [3] M. Gappa, "Let's build a SQL parser in go!," *Mariano Gappa's Blog*. [Online]. Available:
<https://marianogappa.github.io/software/2019/06/05/lets-build-a-sql-parser-in-go/>. [Accessed December 9, 2021].