

CSCI 5408

DATA MANAGEMENT AND WAREHOUSING

Tiny DB Sprint 1 Report

GitLab Assignment Link: <https://git.cs.dal.ca/mkalathiya/tinydb>

Submitted by:

Aditya Chaudhary (B00971587)

Rutvik Mansukhbhai Vaghani(B00978840)

Mansi Kalathiya(B00979173)

Table of Contents

Background Research.....	3
Architecture Diagram.....	6
Pseudocode.....	7
Testing Evidence:.....	10
Sprint 1 Meeting Logs:.....	24
References	25

Table of Figures:

Figure 1 : Architecture Diagram.....	6
Figure 2 : Registration of the New User.....	10
Figure 3 : User register successfully.....	10
Figure 4 : Register again with the same name.....	11
Figure 5 : User login with perfect credential	11
Figure 6 : User login with invalid credential	12
Figure 7 : Create database successfully	12
Figure 8 : Create the table without using the database	13
Figure 9 : Use specific database.....	13
Figure 10 : Create a table.....	14
Figure 11 : Create another table.....	14
Figure 12 : Empty file structure of the table.....	15
Figure 13 : Metadata file of the table	15
Figure 14 : Insert the data in the table without the selecting the database	16
Figure 15 : Insert the data into table	16
Figure 16 : After inserting all the data into table	17
Figure 17 : Select the whole table	17
Figure 18 : select the table which has no data into table	18
Figure 19 : Select the table which are not created.....	18
Figure 20 : Select the specific Column into table	19
Figure 21 : select the Specific the row from the table.....	19
Figure 22 : select the all the row by the condition for filtering the row	20
Figure 23 : Update the details the which are not present in the table	20
Figure 24 : Before Updating the details into tables	21
Figure 25 : Update the details of the table	21
Figure 26 : Update the data with incorrect details.....	22
Figure 27 : Before the deleting the data from the table.....	22
Figure 28 : Delete the row from the table.....	23
Figure 29 : Drop table by giving the incorrect data	23
Figure 30 : Drop the table from the database	24

Background Research

A Database Management System is a software application that helps to perform various operation such as create, maintain, and manipulate databases. It provides an interactive interface to the users to interact with the application and helps them to perform various tasks. The primary goal of a DBMS is to provide an effective way to store, retrieve, and manipulate data efficiently.

Essential Functions of a DBMS:

- **Storage and Retrieval of Data:** A DBMS is primarily used for storing large volumes of data in an organized manner. It focuses on easy and efficient data retrieval.
- **Data Manipulation:** It allows the user to perform insertion of new records, updating of existing record and deletion of records, the DBMS ensures all these operations were performed while maintaining data integrity.
- **Data Administration:** A DBMS provides tools for the effective administration of the database, by managing users access, providing functionalities for backup and recovery. This helps to maintain the security of the database.
- **Transaction Management:** A DBMS nowadays mainly used for transaction management to ensure the stored data is consistent and reliable. DBMS provide a mechanism that handles sequences of operations, all performed as a single unit.
- **Concurrency Control:** DBMS allows multiuser to interact with the application. In multi-user environment, the DBMS provides mechanisms for control concurrent access to the database, that helps many users to work with the same data simultaneously without interfering with each other.
- **Data Integrity:** DBMS ensures that data within a database is secure and data integrity is maintained. This is accomplished by enforcing several integrity rules. Basic types of integrity constraints, like primary key and foreign key, enforce logical relationships between data items.
- **Security:** Another most essential function of a DBMS is to prevent access to data by unauthorized users and allow access to users with specific permissions. It implements security measures, such as authentication, authorization, and encryption, to ensure that data is safeguarded.

Existing Research on DBMS:

1. Title: Database System: Concepts and Design

Summary:

This paper focuses on the designing and implementing databases. It provides and extensive overview of how different methodologies can be applied for database design. It also reflects how data integrity, security measures, and the use of SQL benefits in managing and manipulating data. The report also discusses the applications of various database models and architecture.

2. Title: Evolution of Autonomic Database Management Systems

Summary:

This paper focuses on reducing human intervention in transition of DBMS to automatic DBMS, which helps to highlight precautions needs to take while developing DBMS. It evaluates three leading DBMSs—IBM's DB2, Oracle, and Microsoft SQL Server—on their autonomic characteristics. The study helps to give insights about self creation of DBMS, self optimization, self healing, and self protection features in these systems.

3. Title: An Overview of Database Management Systems and their Applications along with the queries for processing the System

Summary:

This paper provides a comprehensive overview of database management systems (DBMS). It highlights how DBMS facilitate various operations like retrieval of data, storage, and management of data. The paper describes various levels of data abstraction: internal, conceptual, and external schemas, each abstraction practice serving different roles from data storage to end-user interaction. It shows various data models to illustrate how data can be organised and managed. The paper also highlights practical applications of DBMS in various sectors like banking, education, industry, military, online shopping, credit card transactions, and social media.

Data Structures we use in our TinyDB:

In selecting the data structure for TinyDB, our objective is achieving simplicity, effectiveness and being as easy to process as possible. We used arrays and hash maps for in-memory data manipulation and plain text files for persistence. For commands processing, arrays are obviously your best choice since they give you fast access and simple iterations but hash maps beat them in a heartbeat when it comes to querying user profiles and looking up database metadata. We crafted a custom file format that can be easily parsed for accurate data and better represent the specified requirements rather than using JSON or XML. This approach balances performance with simplicity, facilitating quick development and reliable data management. Arrays provide fast access to the elements by their index and this is useful for typical scenarios like iteration over query results or commands that apply on many records. Hash maps provide constant time complexity for search operations, making them ideal for user profile management and metadata storage, where rapid access and updates are necessary. This efficiency will be particularly advantageous in future modules, such as transaction processing, where quick in-memory operations are required before committing changes to persistent storage. Hash Maps are used to implement search operations with linear time complexity. It is useful for managing user profiles and metadata, where quick search and updates are required. This will be highly beneficial for future modules such as transaction processing where in memory operations need to be quick before changes are committed to persistent storage

Overview of our TinyDB:

TinyDB is a Java-based database management system that was created independently of third-party frameworks. It emphasises fundamental DBMS functions such database and table creation, data insertion, querying (including select, update, and delete operations with basic WHERE conditions), and table dropping. It provides an interactive command-line interface (CLI). For query and data processing, the system uses linear data structures; to guarantee persistent storage, data and metadata are stored in a unique file format. It differentiates between standard queries and transactional operations by implementing transaction processing with ACID features. The creation of JSON-formatted logs for user queries, database status, events, and query execution times is a component of log management. TinyDB also facilitates reverse engineering by using database metadata to create text-based Entity-Relationship Diagrams (ERDs). In order to maintain data integrity and reflect recent revisions, it also permits the export of database structures and values in SQL format. Basic console-based UI features are included in the system, and login security is provided using hashed credentials kept in text files.

Architecture Diagram

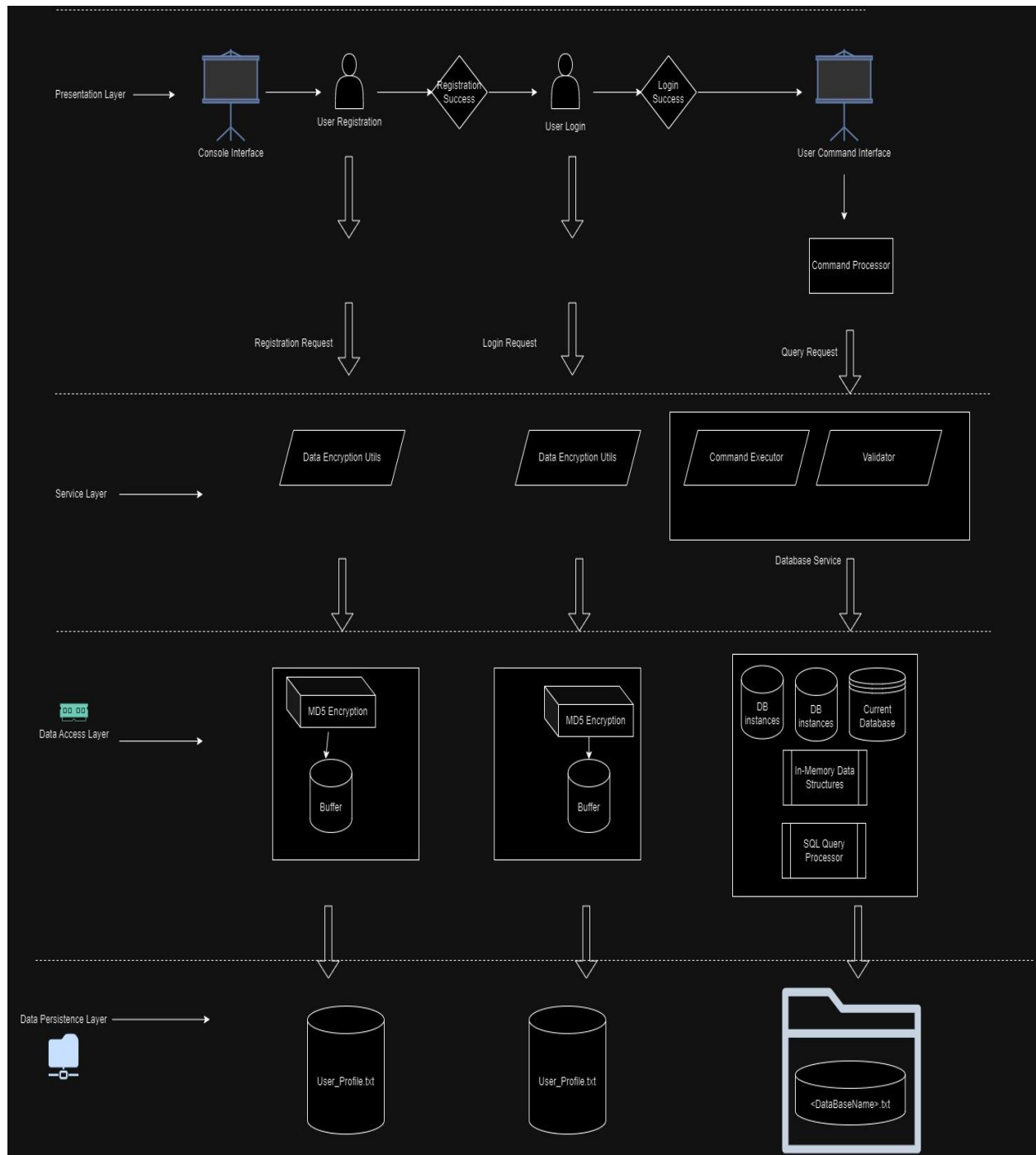


Figure 1 : Architecture Diagram

Pseudocode

Function UserProfileManager:

```
FILE_NAME = "User_Profile.txt"
```

```
function loadProfiles():  
    profiles = {}  
    if not fileExists(FILE_NAME):  
        return profiles  
    try:  
        with open(FILE_NAME, 'r') as file:  
            for line in file:  
                parts = line.strip().split(',')  
                if len(parts) == 4:  
                    username, password, firstName, lastName = parts  
                    profiles[username] = [password, firstName, lastName]  
    except IOError as e:  
        print("Error reading user profiles:", e)  
    return profiles
```

```
Function saveProfiles(profiles):
```

```
    try:  
        with open(FILE_NAME, 'w') as file:  
            for username, data in profiles.items():  
                file.write(username + ',' + ','.join(data) + '\n')  
    except IOError as e:  
        print("Error saving user profiles:", e)
```

Function createDatabase(dbName):

```
if databaseExists(dbName):  
    print("Database already exists.")  
    return False  
createDirectory(dbName)  
print("Database created successfully.")  
return True
```

```
Function createTable(tableName, tableStructure):
```

```
if not activeDatabase():  
    print("Use a database first.")
```

```
return
```

```
if tableExists(tableName):  
    print("Table already exists.")  
    return
```

```
createDirectory(tableName)  
writeMetadata(tableName, tableStructure)  
print("Table created successfully.")
```

Function deleteRecord(tableName, condition):

```
if not activeDatabase():  
    print("Use a database first.")  
    return
```

```
if not tableExists(tableName):  
    print("Table does not exist.")  
    return
```

```
filteredData = filterData(tableName, condition)  
writeData(tableName, filteredData)  
print("Record deleted successfully.")
```

Function insertRecord(tableName, columns, values):

```
if not activeDatabase():  
    print("Use a database first.")  
    return
```

```
if not tableExists(tableName):  
    print("Table does not exist.")  
    return
```

```
validateColumns(tableName, columns)  
writeData(tableName, combineData(columns, values))  
print("Record inserted successfully.")
```

Function updateRecord(tableName, columnName, value, condition):

```
if not activeDatabase():  
    print("Use a database first.")  
    return
```



```
if not tableExists(tableName):  
    print("Table does not exist.")  
    return
```

```
filteredData = filterData(tableName, condition)  
updateData(filteredData, columnName, value)  
writeData(tableName, filteredData)  
print("Record updated successfully.")
```

Function selectData(tableName, columns, condition):

```
if not activeDatabase():  
    print("Use a database first.")  
    return
```

```
if not tableExists(tableName):  
    print("Table does not exist.")  
    return
```

```
data = readData(tableName)  
filteredData = filterData(data, condition)  
printData(selectColumns(filteredData, columns))
```

Function dropTable(tableName):

```
if not activeDatabase():  
    print("Use a database first.")  
    return
```

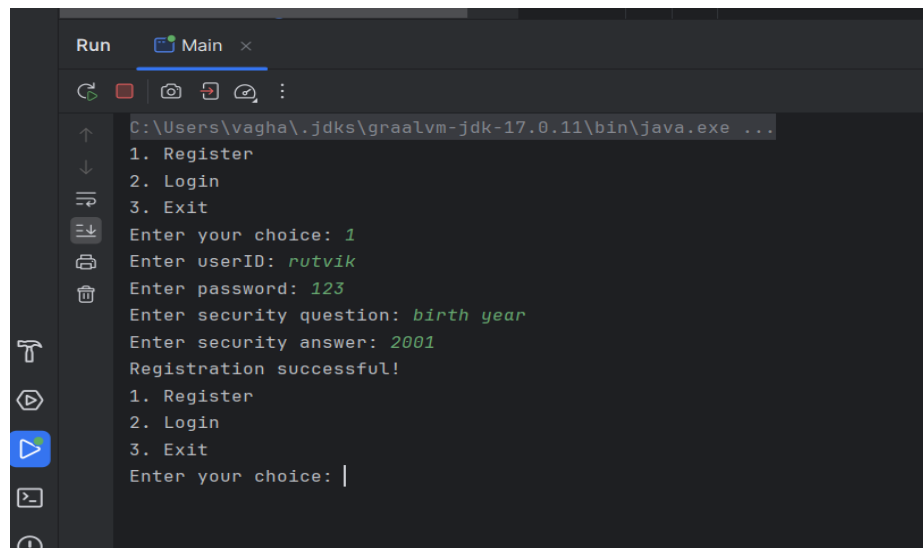
```
if not tableExists(tableName):  
    print("Table does not exist.")  
    return
```

```
deleteDirectory(tableName)  
deleteMetadata(tableName)  
print("Table dropped successfully.")
```

Testing Evidence:

Login & Signup:

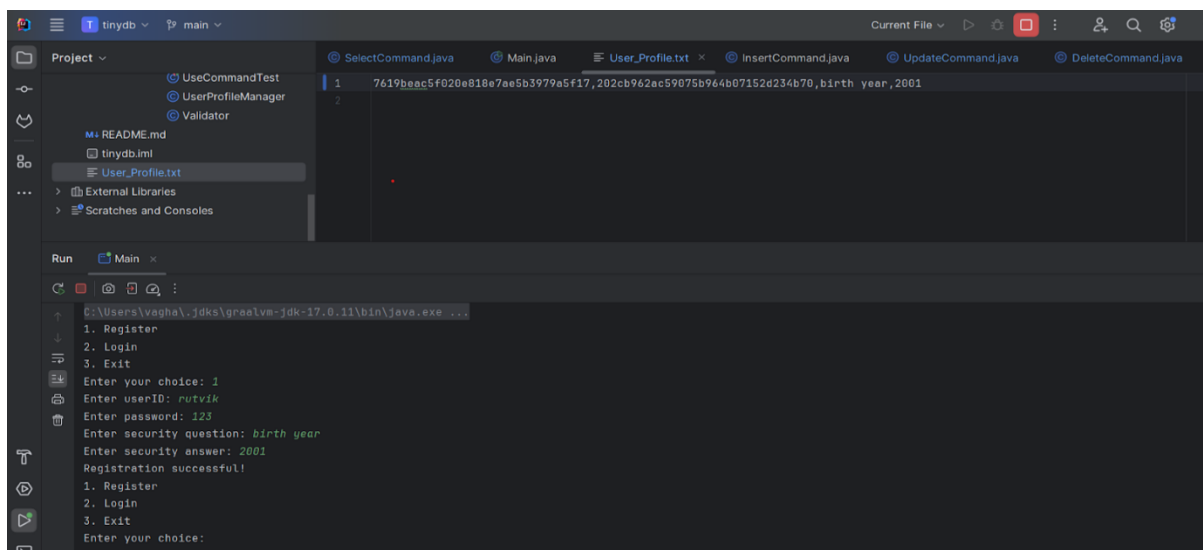
- Registration of the new user with the all the details like userID, password and security question.



```
Run Main x
C:\Users\vagha\.jdk\graalvm-jdk-17.0.11\bin\java.exe ...
1. Register
2. Login
3. Exit
Enter your choice: 1
Enter userID: rutvik
Enter password: 123
Enter security question: birth year
Enter security answer: 2001
Registration successful!
1. Register
2. Login
3. Exit
Enter your choice: |
```

Figure 2 : Registration of the New User

- User Registered successfully and the data of the user stored in file.



```
Project
- tinydb
  - UserCommandTest
  - UserProfileManager
  - Validator
  - README.md
  - tinydb.iml
  - User_Profile.txt
- External Libraries
- Scratches and Consoles

Run Main x
C:\Users\vagha\.jdk\graalvm-jdk-17.0.11\bin\java.exe ...
1. Register
2. Login
3. Exit
Enter your choice: 1
Enter userID: rutvik
Enter password: 123
Enter security question: birth year
Enter security answer: 2001
Registration successful!
1. Register
2. Login
3. Exit
Enter your choice:
```

Figure 3 : User register successfully

- The user want to register again with the same name it will not able to register

```

C:\Users\vagha\.jdk\graalvm-jdk-17.0.11\bin\java.exe ...
1. Register
2. Login
3. Exit
Enter your choice: 1
Enter userID: rutvik
Enter password: 123
Enter security question: birth year
Enter security answer: 2001
Registration successful!
1. Register
2. Login
3. Exit
Enter your choice: 1
Enter userID: rutvik
Enter password: 123
User already exists.
1. Register
2. Login
3. Exit
Enter your choice:

```

Figure 4 : Register again with the same name

- The user login with perfect credential then use can successfully login in system.

```

C:\Users\vagha\.jdk\graalvm-jdk-17.0.11\bin\java.exe ...
1. Register
2. Login
3. Exit
Enter your choice: 2
Enter userID: rutvik
Enter password: 123
Answer the security question birth year:
2001
Login successful!
Welcome to TinyDB!
TinyDB>

```

Figure 5 : User login with perfect credential

- user login and add the invalid credential then user cannot login to the system.

```

Run Main x
1. Register
2. Login
3. Exit
Enter your choice: 2
Enter userID: rutvik
Enter password: 12
Invalid userID or password.
1. Register
2. Login
3. Exit
Enter your choice:

```

tinydb > src > src > main > java > org > example > Main > main

Figure 6 : User login with invalid credential

Query Manipulation:

- Create the database after the login.

```

Project: tinydb
- databases
  - A
  - README.md
  - tinydb.iml
  - User_Profile.txt
- External Libraries

Main.java
7 public class Main {
8     public static void main(String[] args) {
9         while (true) {
10             System.out.println("1. Register");
11             System.out.println("2. Login");
12             System.out.println("3. Exit");
13             System.out.print("Enter your choice: ");
14             int choice = scanner.nextInt();
15             scanner.nextLine();
16         }
17     }
18 }

```

```

Run Main x
1. Register
2. Login
3. Exit
Enter your choice: 2
Enter userID: rutvik
Enter password: 123
Answer the security question birth year:
2001
Login successful!
Welcome to TinyDB!
TinyDB> create database a;
Database created successfully.
TinyDB>

```

Figure 7 : Create database successfully

- User Want to create the table without the selecting the database then return the error that database is not selected.

```

7 public class Main {
11     public static void main(String[] args) {
14         while (true) {
16             System.out.println("1. Register");
17             System.out.println("2. Login");
18             System.out.println("3. Exit");
19             System.out.print("Enter your choice: ");
20
21             int choice = scanner.nextInt();
22             scanner.nextLine();
23         }

```

```

C:\Users\vagha\.jdk\graalvm-jdk-17.0.11\bin\java.exe ...
1. Register
2. Login
3. Exit
Enter your choice: 2
Enter userID: rutvik
Enter password: 123
Answer the security question birth year:
2001
Login successful!
Welcome to TinyDB!
TinyDB> create database a;
Database created successfully.
TinyDB> create table people(id int,name string,age int);
Error: No database selected.
TinyDB>

```

Figure 8 : Create the table without using the database

- User Can use the database by giving the name of the database.

```

2. Login
3. Exit
Enter your choice: 2
Enter userID: rutvik
Enter password: 123
Answer the security question birth year:
2001
Login successful!
Welcome to TinyDB!
TinyDB> create database a;
Database created successfully.
TinyDB> create table people(id int,name string,age int);
Error: No database selected.
TinyDB> use a;
Using database: A
TinyDB>

```

Figure 9 : Use specific database

Create Table:

- User can create the table by using the specific database. There is two file ceated when table is created when one file is table name and another file is {tablename}_meta.txt file created.

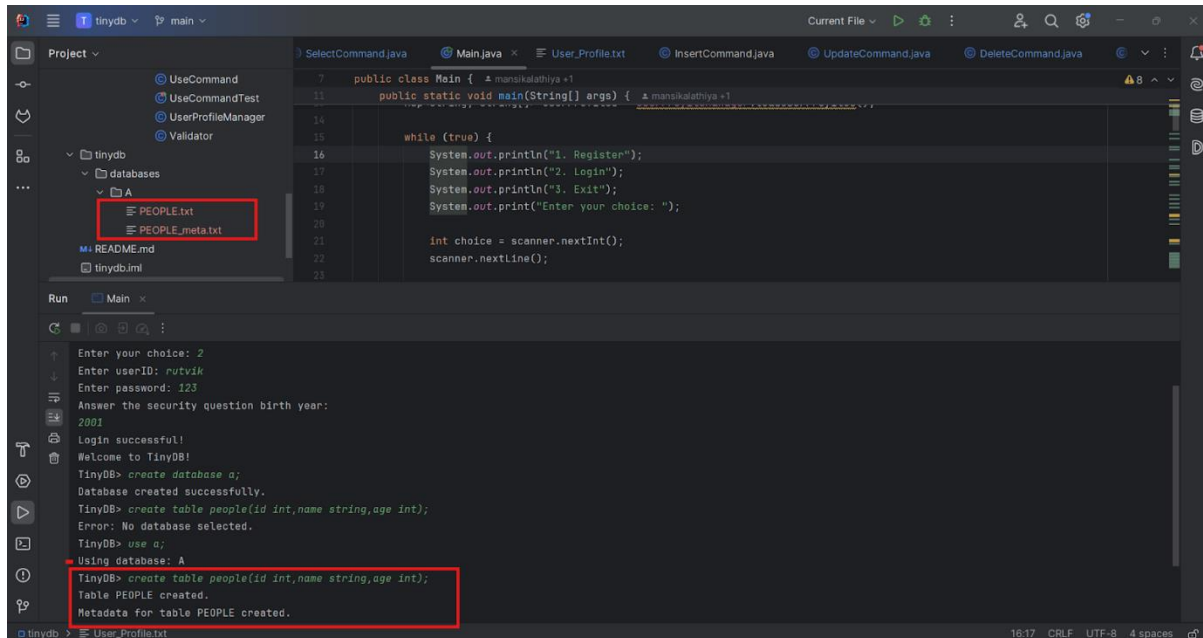


Figure 10 : Create a table

- Create the another table of the by gining the details.

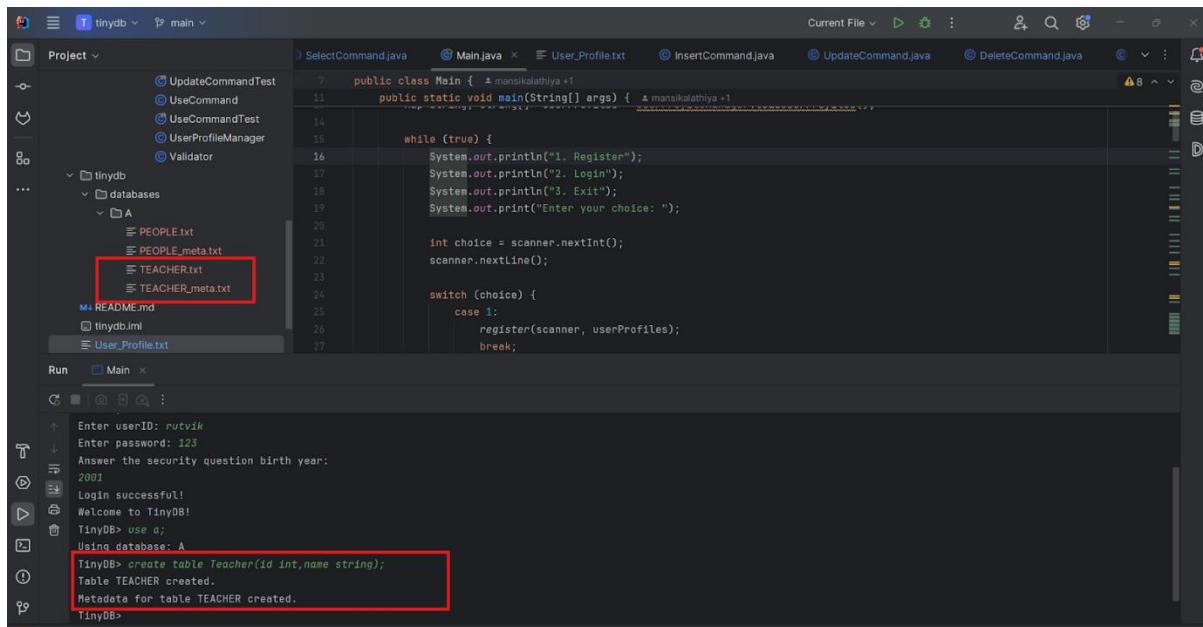


Figure 11 : Create another table

- Before the inserting the data into table the structure of the file is.

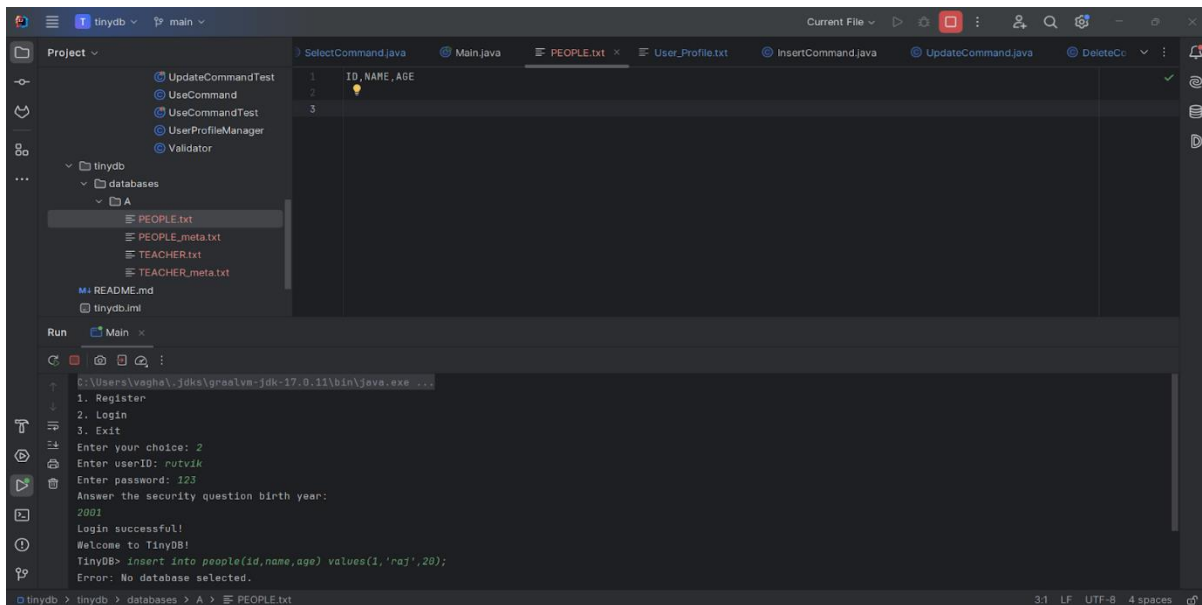


Figure 12 : Empty file structure of the table

- Metadata file structure of the data when there is the table is created then the structure of the meta file.

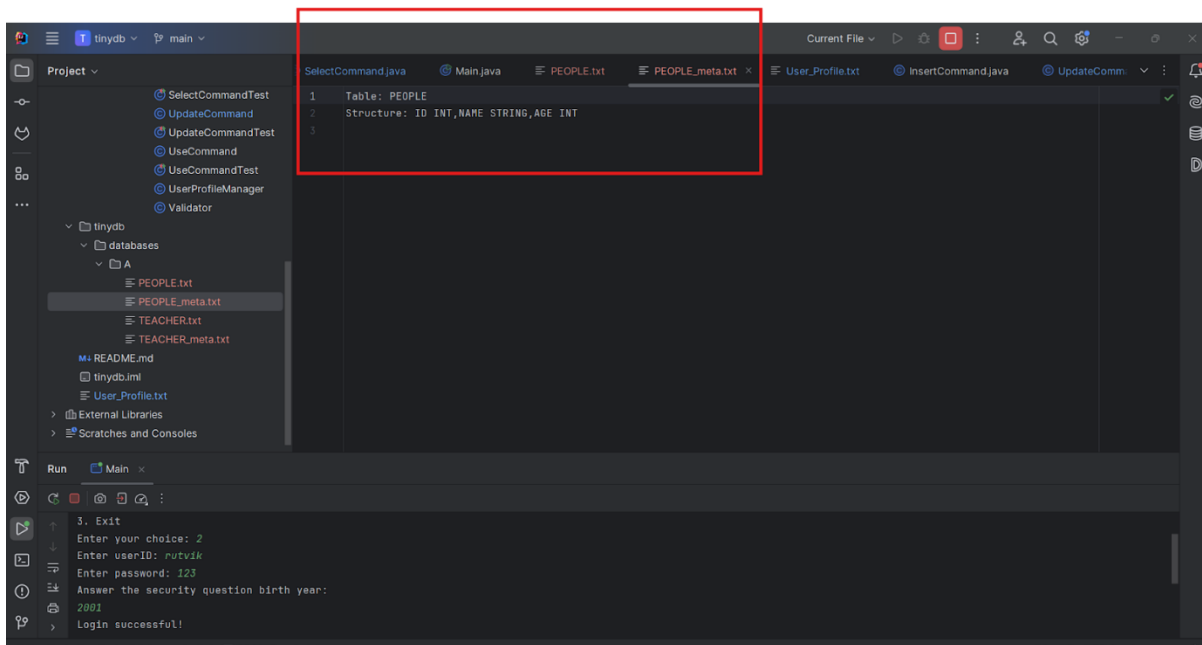


Figure 13 : Metadata file of the table

- User want to insert the data without the selecting the database then user can not the insert the data.

```

public class Main {
    public static void main(String[] args) {
        while (true) {
            System.out.println("1. Register");
            System.out.println("2. Login");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
        }
    }
}

```

```

1. Register
2. Login
3. Exit
Enter your choice: 2
Enter userID: rutvik
Enter password: 123
Answer the security question birth year:
2001
Login successful!
Welcome to TinyDB!
TinyDB> insert into people(id,name,age) values(1,'raj',20);
Error: No database selected.
TinyDB>

```

Figure 14 : Insert the data in the table without the selecting the database

- After selecting the database the user can insert the data into the table and the the data is added into the file.

```

ID, NAME, AGE
1, 'RAJ', 20

```

```

Enter your choice: 2
Enter userID: rutvik
Enter password: 123
Answer the security question birth year:
2001
Login successful!
Welcome to TinyDB!
TinyDB> insert into people(id,name,age) values(1,'raj',20);
Error: No database selected.
TinyDB> use a;
Using database: A
TinyDB> insert into people(id,name,age) values(1,'raj',20);
Record inserted successfully into table PEOPLE.
TinyDB>

```

Figure 15 : Insert the data into table

- After inserting the multiple data into the table it will added the data into the file.

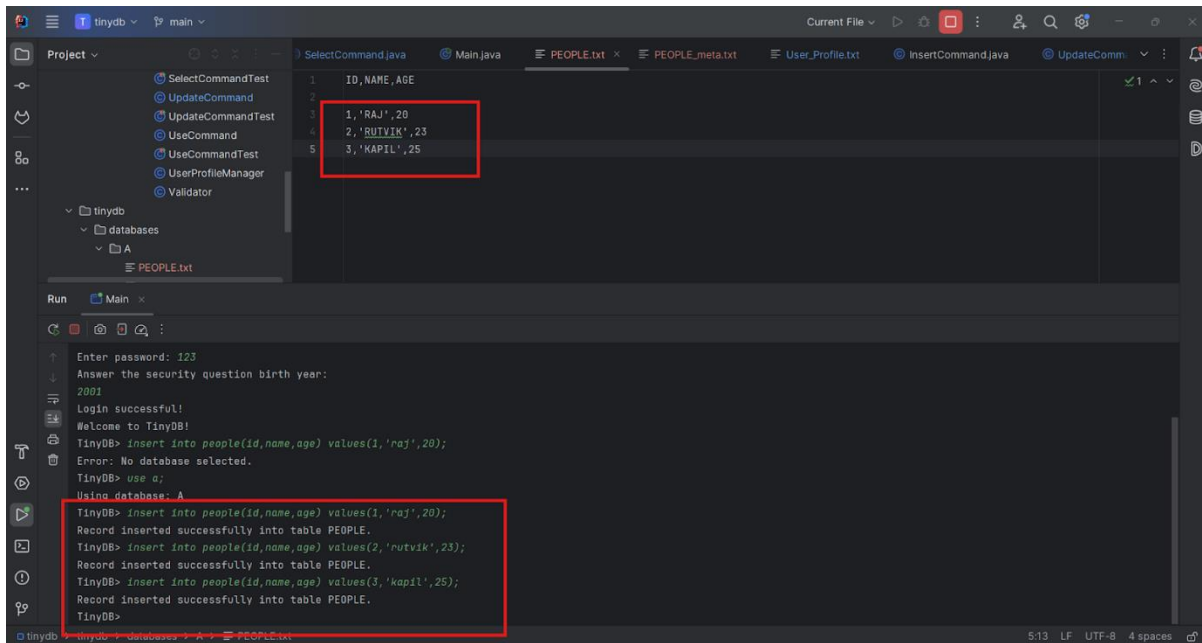


Figure 16 : After inserting all the data into table

Select Query:

- Selecting the whole table printing whole data of the table.

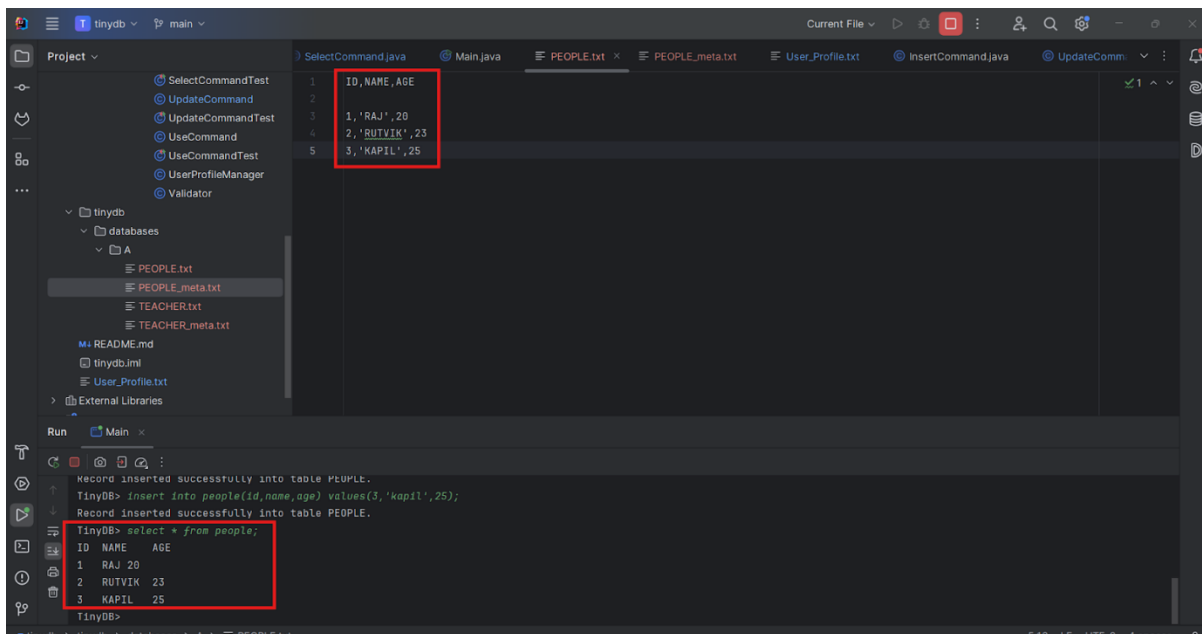


Figure 17 : Select the whole table

- Select the table which has no data.

The screenshot shows the TinyDB IDE interface. The Project Explorer on the left lists files including PEOPLE.txt, PEOPLE_meta.txt, TEACHER.txt, and TEACHER_meta.txt. The main editor displays the TEACHER.txt file with a table structure: ID, NAME. The Run console at the bottom shows the following commands and output:

```
TinyDB> insert into people(id,name,age) values(1,'raj',20);
Error: No database selected.
TinyDB> use a;
Using database: A
TinyDB> insert into people(id,name,age) values(1,'raj',20);
Record inserted successfully into table PEOPLE.
TinyDB> insert into people(id,name,age) values(2,'rutvik',23);
Record inserted successfully into table PEOPLE.
TinyDB> insert into people(id,name,age) values(3,'kapil',25);
Record inserted successfully into table PEOPLE.
TinyDB> select * from people;
ID NAME AGE
1 RAJ 20
2 RUTVIK 23
3 KAPIL 25
TinyDB> SELECT * FROM TEACHER;
No matching records found.
```

Figure 18 : select the table which has no data into table

- Select the table which are not created.

The screenshot shows the TinyDB IDE interface. The Project Explorer on the left lists files including PEOPLE.txt, PEOPLE_meta.txt, TEACHER.txt, and TEACHER_meta.txt. The main editor displays the TEACHER.txt file with a table structure: ID, NAME. The Run console at the bottom shows the following commands and output:

```
TinyDB> select * from staff;
Error: Table does not exist.
TinyDB>
```

Figure 19 : Select the table which are not created

- Selecting the specific column in the table.

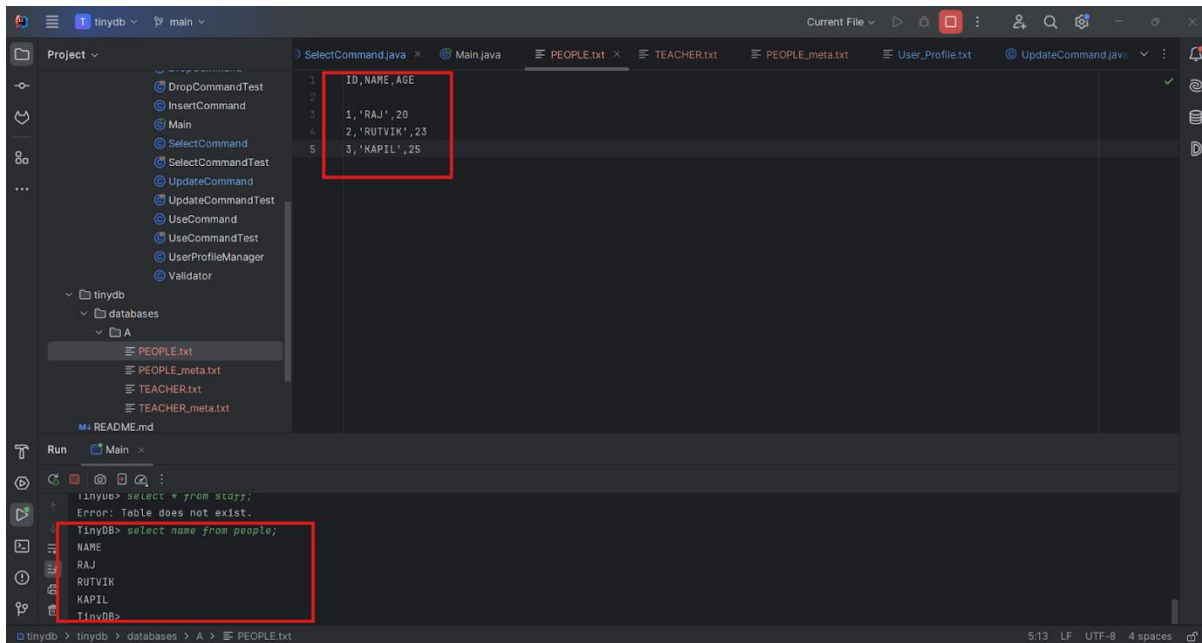


Figure 20 : Select the specific Column into table

- Selecting the specifying row from the table.

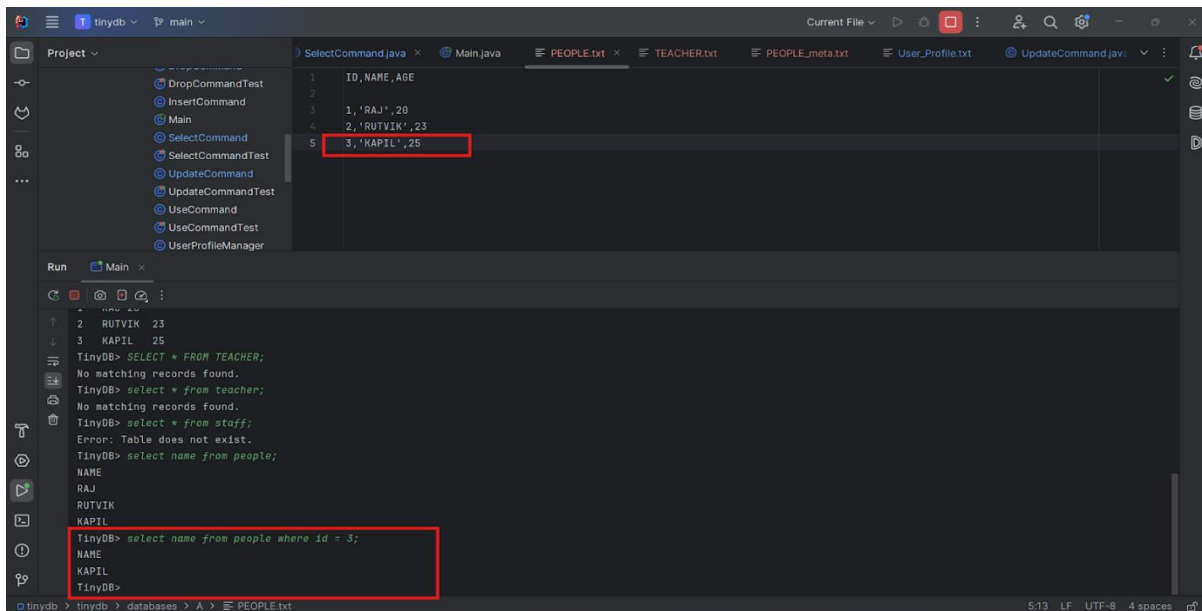


Figure 21 : select the Specific the row from the table

- Select the row the by the condition.

The screenshot shows the TinyDB application interface. The 'Project' view on the left lists various command tests. The 'Main' window displays a SQL query: `select * from people;`. The results are shown in a table with columns ID, NAME, and AGE. The data rows are: (1, 'RAJ', 20), (2, 'RUTVIK', 23), and (3, 'KAPIL', 25). The 'Run' window shows the execution of the query, displaying the results in a table format.

ID	NAME	AGE
1	RAJ	20
2	RUTVIK	23
3	KAPIL	25

Figure 22 : select the all the row by the condition for filtering the row

Update Query:

- Update the details which are not in the table.

The screenshot shows the TinyDB application interface. The 'Main' window displays a SQL query: `update people set name = 'kapil' where id = 10;`. The results are shown in a table with columns ID, NAME, and AGE. The data rows are: (1, 'RAJ', 20), (2, 'RUTVIK', 23), and (3, 'KAPIL', 25). The 'Run' window shows the execution of the query, displaying the results in a table format.

ID	NAME	AGE
1	RAJ	20
2	RUTVIK	23
3	KAPIL	25

Figure 23 : Update the details the which are not present in the table

- Before the updating the details into the table. The data into the file.

```

Project
├── tinydb
│   ├── databases
│   │   └── A
│   │       ├── PEOPLE.txt
│   │       ├── PEOPLE_meta.txt
│   │       ├── TEACHER.txt
│   │       └── TEACHER_meta.txt
│   ├── README.md
│   ├── tinydb.iml
│   ├── User_Profile.txt
│   └── External Libraries
└── ...

Run
Main
TinyDB> insert into people(id,name,age) values(3,'kapil',25);
Record inserted successfully into table PEOPLE.
TinyDB> select * from people;
ID NAME AGE
1 RAJ 20
2 RUTVIK 23
3 KAPIL 25

```

Figure 24 : Before Updating the details into tables

- After the updating the data which are updating the data into file.

```

Project
├── tinydb
│   ├── databases
│   │   └── A
│   │       ├── PEOPLE.txt
│   │       ├── PEOPLE_meta.txt
│   │       ├── TEACHER.txt
│   │       └── TEACHER_meta.txt
│   ├── README.md
│   ├── tinydb.iml
│   ├── User_Profile.txt
│   └── External Libraries
└── ...

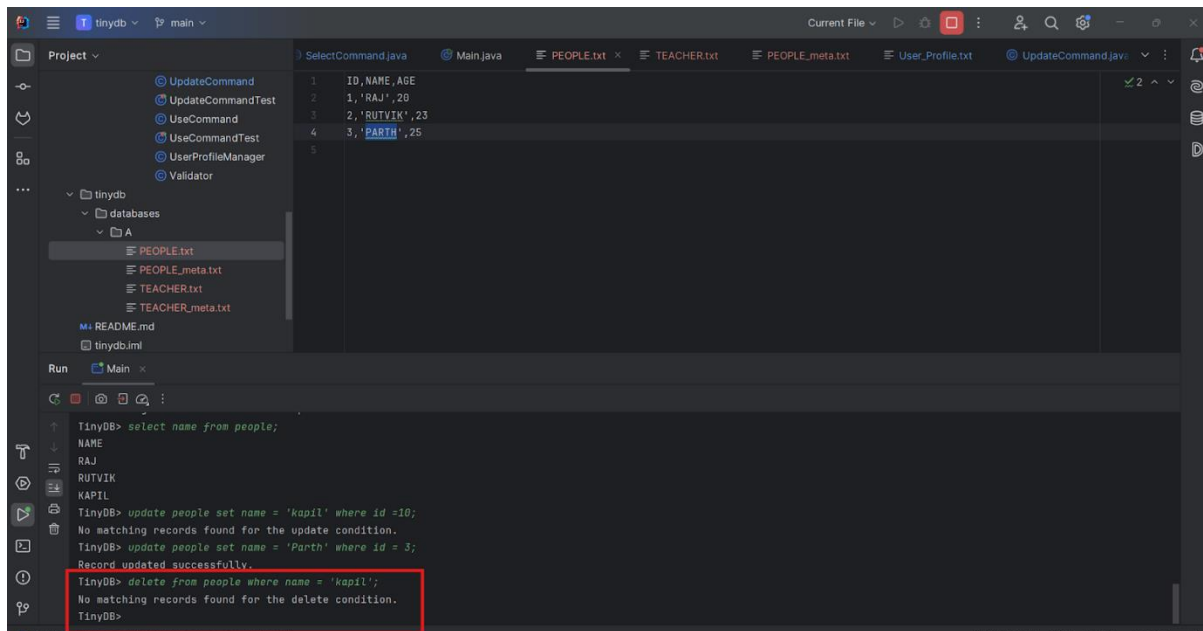
Run
Main
TinyDB> update people set name = 'kapil' where id = 10;
No matching records found for the update condition.
TinyDB> select name from people;
NAME
RAJ
RUTVIK
KAPIL
TinyDB> update people set name = 'kapil' where id = 10;
No matching records found for the update condition.
TinyDB> update people set name = 'Parth' where id = 3;
Record updated successfully.
TinyDB>

```

Figure 25 : Update the details of the table

Delete Query:

- Delete the data by giving the details which are not correct.

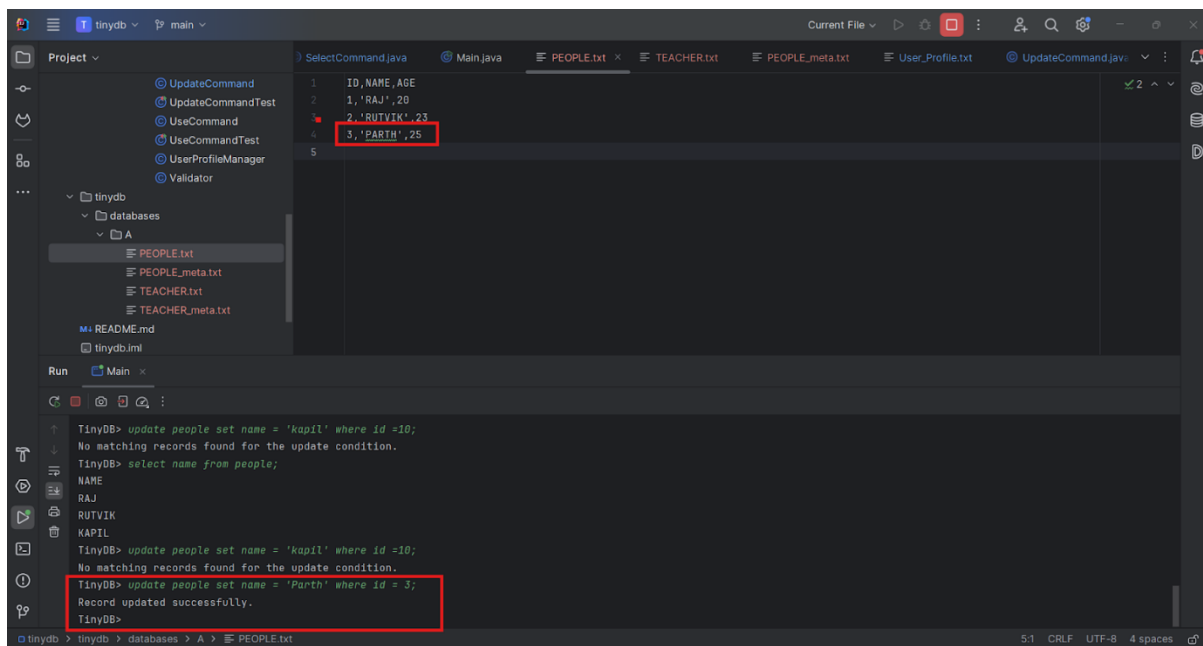


```
1 ID, NAME, AGE
2 1, 'RAJ', 20
3 2, 'RUTVIK', 23
4 3, 'PARTH', 25
5

TinyDB> select name from people;
NAME
RAJ
RUTVIK
KAPIL
TinyDB> update people set name = 'kapil' where id = 10;
No matching records found for the update condition.
TinyDB> update people set name = 'Parth' where id = 3;
Record updated successfully.
TinyDB> delete from people where name = 'kapil';
No matching records found for the delete condition.
TinyDB>
```

Figure 26 : Update the data with incorrect details

- Before the Delete the data into file.



```
1 ID, NAME, AGE
2 1, 'RAJ', 20
3 2, 'RUTVIK', 23
4 3, 'PARTH', 25
5

TinyDB> update people set name = 'kapil' where id = 10;
No matching records found for the update condition.
TinyDB> select name from people;
NAME
RAJ
RUTVIK
KAPIL
TinyDB> update people set name = 'kapil' where id = 10;
No matching records found for the update condition.
TinyDB> update people set name = 'Parth' where id = 3;
Record updated successfully.
TinyDB>
```

Figure 27 : Before the deleting the data from the table

- Deleting the row by giving the correct data in the table.

The screenshot shows the TinyDB application interface. The left sidebar displays the project structure with a database named 'A' containing tables 'PEOPLE.txt', 'PEOPLE_meta.txt', 'TEACHER.txt', and 'TEACHER_meta.txt'. The main window shows a table with columns 'ID', 'NAME', and 'AGE'. The table contains two rows: (1, 'RAJ', 20) and (2, 'RUTVIK', 23). The console output shows the following commands and results:

```
TinyDB> update people set name = 'kapil' where id =10;
No matching records found for the update condition.
TinyDB> update people set name = 'Parth' where id = 3;
Record updated successfully.
TinyDB> delete from people where name = 'kapil';
No matching records found for the delete condition.
TinyDB> delete from people where name = 'parth';
Record deleted successfully.
TinyDB>
```

Figure 28 : Delete the row from the table

- Drop the table by giving the table name which are not present into the database.

The screenshot shows the TinyDB application interface. The left sidebar displays the project structure with a database named 'A' containing tables 'PEOPLE.txt', 'PEOPLE_meta.txt', 'TEACHER.txt', and 'TEACHER_meta.txt'. The main window shows a Java code editor with a 'Main' class. The console output shows the following commands and results:

```
TinyDB> select name from people;
NAME
RAJ
RUTVIK
KAPIL
TinyDB> update people set name = 'kapil' where id =10;
No matching records found for the update condition.
TinyDB> update people set name = 'Parth' where id = 3;
Record updated successfully.
TinyDB> delete from people where name = 'kapil';
No matching records found for the delete condition.
TinyDB> delete from people where name = 'parth';
Record deleted successfully.
TinyDB> drop table aa;
Table AA does not exist.
Metadata for table AA does not exist.
TinyDB>
```

Figure 29 : Drop table by giving the incorrect data

- Drop the table by giving the correct table name.

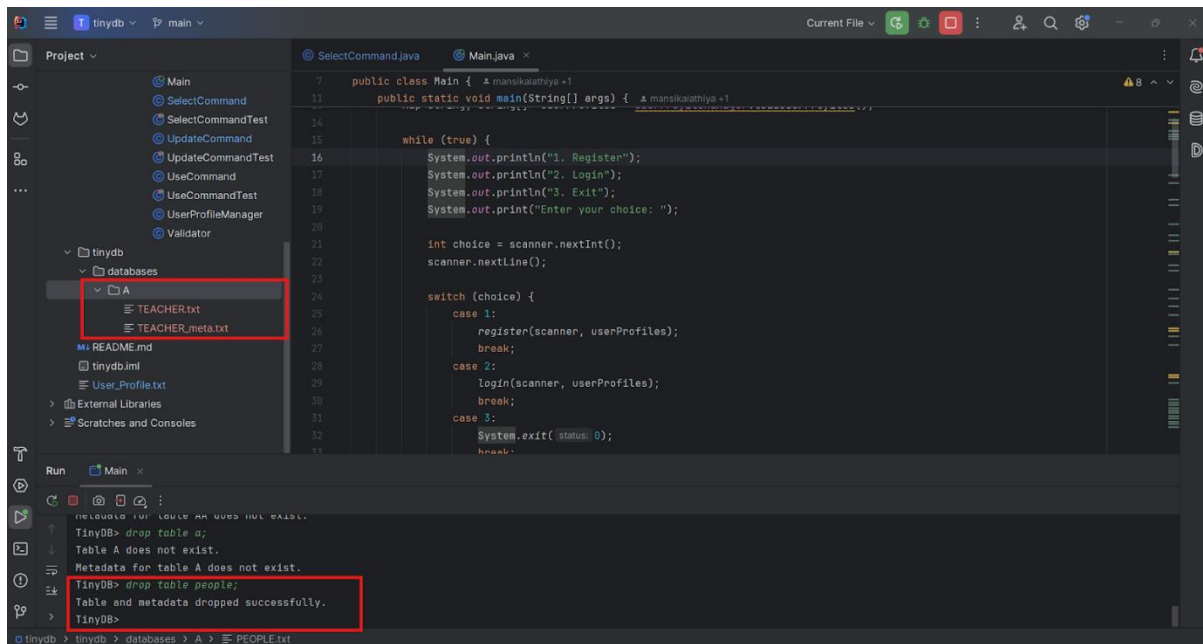


Figure 30 : Drop the table from the database

Sprint 1 Meeting Logs:

No	Date	Time	Attendees	Agenda	Meeting type	Meeting Recording Link
1	3/6/20024	3:00PM	Rutvik, Mansi, Aditya	Basic Discussion about selecting the proper data structure.	Online(MS Teams)	Meeting-20240603_151945-Meeting Recording.mp4
2	20/6/2024	3:00PM	Rutvik, Mansi, Aditya	Task Distribution and discussion of the how to complete the module in lesser day and complete the sprint.	Online(MS Teams)	Meeting-202406020_1513945-Meeting Recording.mp4 (sharepoint.com)

References

- [1] “Mr.BhojarajuG., Dr.M.M.Koganurmath, “Database System : Concepts and Design” , Conference Paper , · December 2003, <https://www.researchgate.net/publication/257298522>
- [2] “Abdul Mateen, BasitRaza, Muhammad Sher, Mian Muhammad Awais, TauqeerHussain, “Evolution of Autonomic Database Management Systems”, Conference Paper · March 2010, <https://www.researchgate.net/publication/224132778>
- [3] “D.V.Sathiya Vadivoo, S.Shanthini, A.Vinora, G.Mohana Priya, "An Overview of Database Management Systems and their Applications along with the queries for processing the System," *SSRG International Journal of Computer Science and Engineering* , vol. 4, no. 3, pp. 1-4, 2017, <https://doi.org/10.14445/23488387/IJCSE-V4I3P101>
- [4] “Structure of Database Management System," Geeks For Geeks”, [Online]. Available: <https://www.geeksforgeeks.org/structure-of-database-management-system/> [Accessed: June 28, 2024]
- [5] “Draw.io”, [Online]. Available: <https://app.diagrams.net/> [Accessed: June 29, 2024]