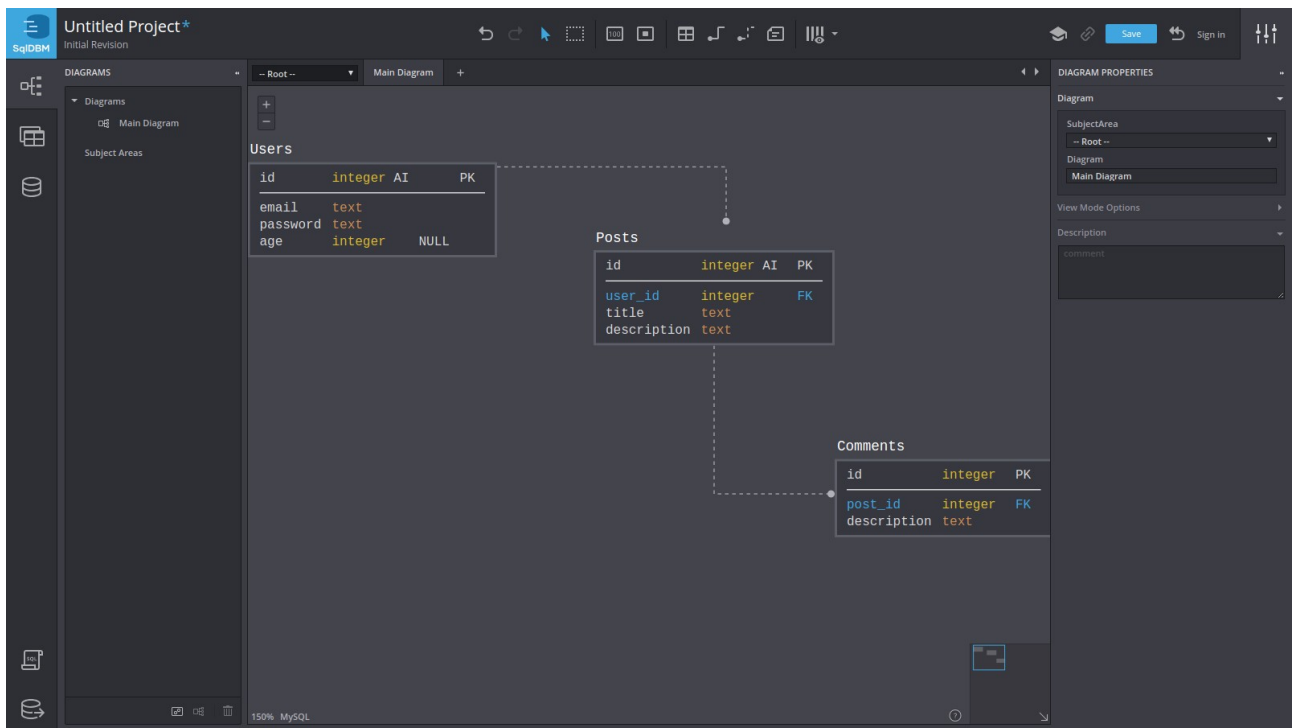


Big Data Lab Assignment 1

Group members

1. Bereket Abera
2. Bereket Gebredingle
3. Nathanael Tarekegn
4. Kaleb wondowssen

1. The database diagram



2. The package we used to connect to the database is [RSQLite](#).

3. 3.1 Creating tables

```
install.packages("RSQLite")
library(DBI)

connection <- dbConnect(RSQLite::SQLite(), ":memory:")

##### CREATING TABLES
dbSendQuery(connection, "CREATE TABLE users (
  id integer PRIMARY KEY,
  email text NOT NULL,
  password text NOT NULL,
  age integer)")

dbSendQuery(connection, "CREATE TABLE posts (
  id integer PRIMARY KEY,
  title text NOT NULL,
  description text NOT NULL,
  user_id integer NOT NULL,
  FOREIGN KEY (user_id) REFERENCES users(id))")

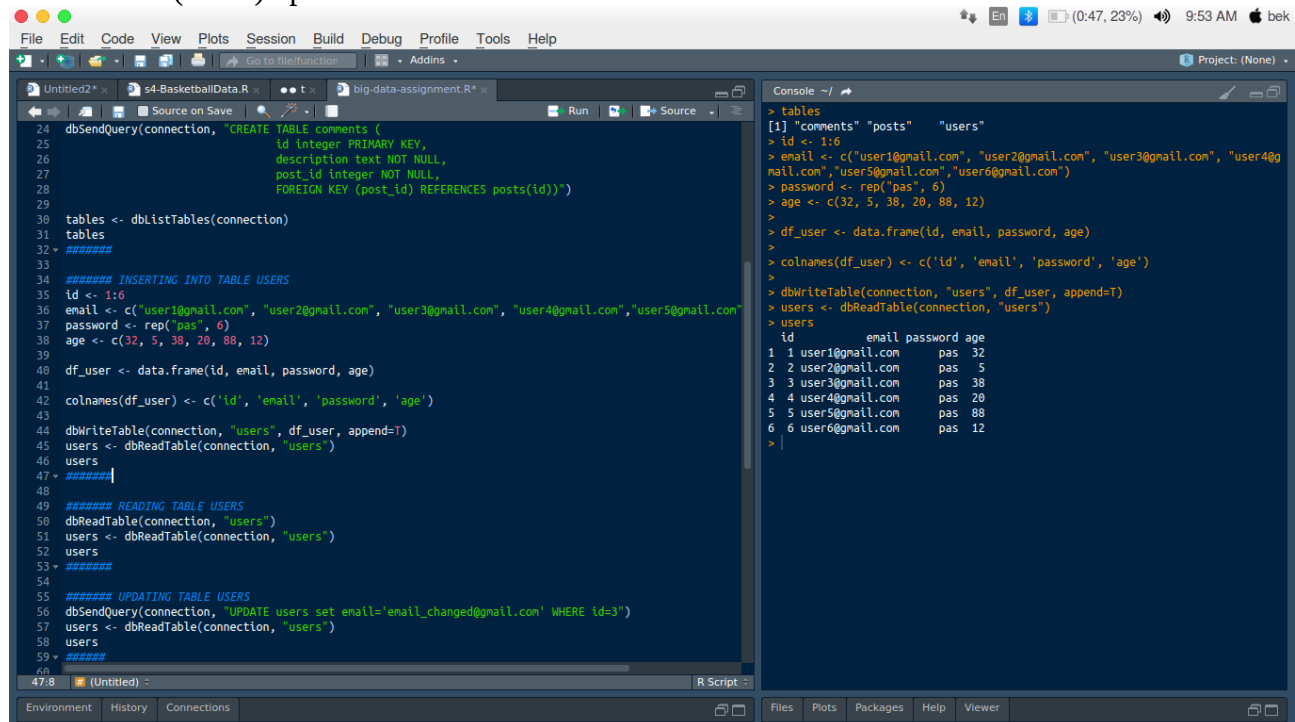
dbSendQuery(connection, "CREATE TABLE comments (
  id integer PRIMARY KEY,
  description text NOT NULL,
  post_id integer NOT NULL,
  FOREIGN KEY (post_id) REFERENCES posts(id))")

tables <- dbListTables(connection)
tables
#####

##### INSERTING INTO TABLE USERS
id <- 1:6
email <- c("user1@mail.com", "user2@mail.com", "user3@mail.com", "user4@mail.com", "user5@mail.com")
```

3.2 CRUD operation

3.2.1 Create (insert) operation



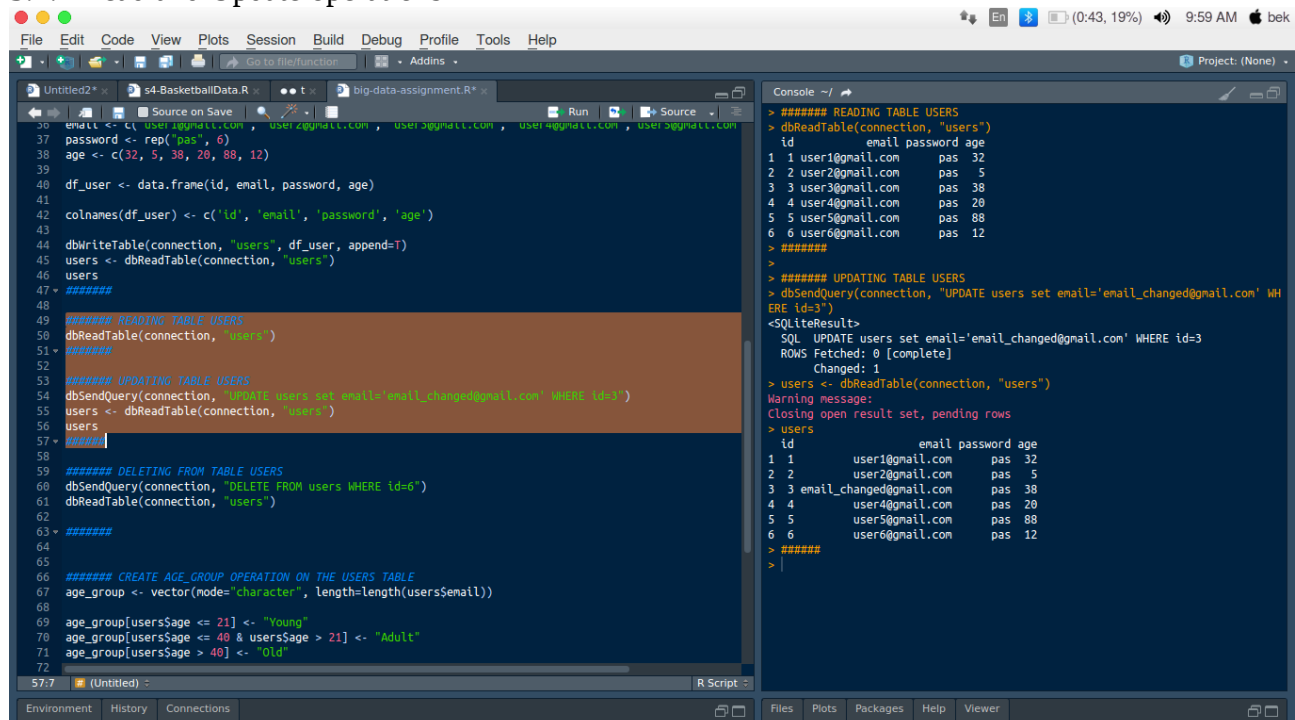
The screenshot shows the RStudio interface with a script editor on the left and a console on the right. The script editor contains R code for creating a database connection, creating a table, inserting data, and reading the data back. The console shows the output of the code, including the list of tables and the data read from the 'users' table.

```
24 dbSendQuery(connection, "CREATE TABLE comments (  
25   id integer PRIMARY KEY,  
26   description text NOT NULL,  
27   post_id integer NOT NULL,  
28   FOREIGN KEY (post_id) REFERENCES posts(id))")  
29  
30 tables <- dbListTables(connection)  
31 tables  
32 #####  
33  
34 ##### INSERTING INTO TABLE USERS  
35 id <- 1:6  
36 email <- c("user1@gmail.com", "user2@gmail.com", "user3@gmail.com", "user4@gmail.com", "user5@gmail.com", "user6@gmail.com")  
37 password <- rep("pas", 6)  
38 age <- c(32, 5, 38, 20, 88, 12)  
39  
40 df_user <- data.frame(id, email, password, age)  
41  
42 colnames(df_user) <- c('id', 'email', 'password', 'age')  
43  
44 dbWriteTable(connection, "users", df_user, append=T)  
45 users <- dbReadTable(connection, "users")  
46 users  
47 #####  
48  
49 ##### READING TABLE USERS  
50 dbReadTable(connection, "users")  
51 users <- dbReadTable(connection, "users")  
52 users  
53 #####  
54  
55 ##### UPDATING TABLE USERS  
56 dbSendQuery(connection, "UPDATE users set email='email_changed@gmail.com' WHERE id=3")  
57 users <- dbReadTable(connection, "users")  
58 users  
59 #####  
60
```

Console output:

```
> tables  
[1] "comments" "posts"    "users"  
> id <- 1:6  
> email <- c("user1@gmail.com", "user2@gmail.com", "user3@gmail.com", "user4@gmail.com", "user5@gmail.com", "user6@gmail.com")  
> password <- rep("pas", 6)  
> age <- c(32, 5, 38, 20, 88, 12)  
>  
> df_user <- data.frame(id, email, password, age)  
>  
> colnames(df_user) <- c('id', 'email', 'password', 'age')  
>  
> dbWriteTable(connection, "users", df_user, append=T)  
> users <- dbReadTable(connection, "users")  
> users  
   id      email password age  
1  1 user1@gmail.com    pas  32  
2  2 user2@gmail.com    pas   5  
3  3 user3@gmail.com    pas  38  
4  4 user4@gmail.com    pas  20  
5  5 user5@gmail.com    pas  88  
6  6 user6@gmail.com    pas  12  
>
```

3.2.2 Read and Update operations



The screenshot shows the RStudio interface with a script editor on the left and a console on the right. The script editor contains R code for reading data from a table, updating data, and deleting data. The console shows the output of the code, including the data read from the 'users' table and the results of the update and delete operations.

```
36 email <- c("user1@gmail.com", "user2@gmail.com", "user3@gmail.com", "user4@gmail.com", "user5@gmail.com", "user6@gmail.com")  
37 password <- rep("pas", 6)  
38 age <- c(32, 5, 38, 20, 88, 12)  
39  
40 df_user <- data.frame(id, email, password, age)  
41  
42 colnames(df_user) <- c('id', 'email', 'password', 'age')  
43  
44 dbWriteTable(connection, "users", df_user, append=T)  
45 users <- dbReadTable(connection, "users")  
46 users  
47 #####  
48  
49 ##### READING TABLE USERS  
50 dbReadTable(connection, "users")  
51 users <- dbReadTable(connection, "users")  
52 users  
53 #####  
54  
55 ##### UPDATING TABLE USERS  
56 dbSendQuery(connection, "UPDATE users set email='email_changed@gmail.com' WHERE id=3")  
57 users <- dbReadTable(connection, "users")  
58 users  
59 #####  
60  
61 ##### DELETING FROM TABLE USERS  
62 dbSendQuery(connection, "DELETE FROM users WHERE id=6")  
63 dbReadTable(connection, "users")  
64  
65 #####  
66  
67 ##### CREATE AGE_GROUP OPERATION ON THE USERS TABLE  
68 age_group <- vector(mode="character", length=length(users$email))  
69  
70 age_group[users$age <= 21] <- "Young"  
71 age_group[users$age >= 40 & users$age > 21] <- "Adult"  
72 age_group[users$age > 40] <- "Old"
```

Console output:

```
> ##### READING TABLE USERS  
> dbReadTable(connection, "users")  
   id      email password age  
1  1 user1@gmail.com    pas  32  
2  2 user2@gmail.com    pas   5  
3  3 user3@gmail.com    pas  38  
4  4 user4@gmail.com    pas  20  
5  5 user5@gmail.com    pas  88  
6  6 user6@gmail.com    pas  12  
>  
> ##### UPDATING TABLE USERS  
> dbSendQuery(connection, "UPDATE users set email='email_changed@gmail.com' WHERE id=3")  
<SQLiteResult>  
SQL UPDATE users set email='email_changed@gmail.com' WHERE id=3  
ROWS Fetched: 0 [complete]  
Changed: 1  
> users <- dbReadTable(connection, "users")  
Warning message:  
Closing open result set, pending rows  
> users  
   id      email password age  
1  1 user1@gmail.com    pas  32  
2  2 user2@gmail.com    pas   5  
3  3 email_changed@gmail.com    pas  38  
4  4 user4@gmail.com    pas  20  
5  5 user5@gmail.com    pas  88  
6  6 user6@gmail.com    pas  12  
> #####  
>
```

3.2.3 Delete operation and some kind of operation based on existing columns

The screenshot shows an R script in RStudio performing several database operations on a 'users' table. The script includes comments for each step: deleting a user, reading the table, updating an email, and creating an age group. The console output shows the results of these operations, including a warning message about closing the result set and the final state of the 'users' table.

```
47 #####
48
49 ##### READING TABLE USERS
50 dbReadTable(connection, "users")
51 #####
52
53 ##### UPDATING TABLE USERS
54 dbSendQuery(connection, "UPDATE users set email='email_changed@gmail.com' WHERE id=3")
55 users <- dbReadTable(connection, "users")
56 users
57 #####
58
59 ##### DELETING FROM TABLE USERS
60 dbSendQuery(connection, "DELETE FROM users WHERE id=6")
61 dbReadTable(connection, "users")
62
63 #####
64
65 ##### CREATE AGE_GROUP OPERATION ON THE USERS TABLE
66 age_group <- vector(mode="character", length=length(users$email))
67
68
69 age_group[users$age <= 21] <- "Young"
70 age_group[users$age <= 40 & users$age > 21] <- "Adult"
71 age_group[users$age > 40] <- "Old"
72
73
74 users$age_group <- age_group
75
76 dbWriteTable(connection, "users", users, overwrite=T)
77
78 dbReadTable(connection, "users")
79
80 #####
81
82 ##### HEAD AND TAIL FUNCTIONS
83
84
85
86
87 #####
88
89
```

Console output:

```
> dbSendQuery(connection, "DELETE FROM users WHERE id=6")
<SQLiteResult>
SQL DELETE FROM users WHERE id=6
ROWS Fetched: 0 [complete]
Changed: 1

> dbReadTable(connection, "users")
  id email password age
1 1 user1@gmail.com pas 32
2 2 user2@gmail.com pas 5
3 3 email_changed@gmail.com pas 38
4 4 user4@gmail.com pas 20
5 5 user5@gmail.com pas 88

Warning message:
Closing open result set, pending rows

> #####
>
> ##### CREATE AGE_GROUP OPERATION ON THE USERS TABLE
> age_group <- vector(mode="character", length=length(users$email))
>
> age_group[users$age <= 21] <- "Young"
> age_group[users$age <= 40 & users$age > 21] <- "Adult"
> age_group[users$age > 40] <- "Old"
>
> users$age_group <- age_group
>
> dbWriteTable(connection, "users", users, overwrite=T)
>
> dbReadTable(connection, "users")
  id email password age age_group
1 1 user1@gmail.com pas 32 Adult
2 2 user2@gmail.com pas 5 Young
3 3 email_changed@gmail.com pas 38 Adult
4 4 user4@gmail.com pas 20 Young
5 5 user5@gmail.com pas 88 Old
6 6 user6@gmail.com pas 12 Young
```

3.2.3 Head , tail and summary of the table

The screenshot shows an R script in RStudio performing head, tail, and summary operations on a 'users' table. The script includes comments for each step: deleting a user, reading the table, updating an email, and creating an age group. The console output shows the results of these operations, including a warning message about closing the result set and the final state of the 'users' table.

```
54 dbSendQuery(connection, "UPDATE users set email='email_changed@gmail.com' WHERE id=3")
55 users <- dbReadTable(connection, "users")
56 users
57 #####
58
59 ##### DELETING FROM TABLE USERS
60 dbSendQuery(connection, "DELETE FROM users WHERE id=6")
61 dbReadTable(connection, "users")
62
63 #####
64
65 ##### CREATE AGE_GROUP OPERATION ON THE USERS TABLE
66 age_group <- vector(mode="character", length=length(users$email))
67
68
69 age_group[users$age <= 21] <- "Young"
70 age_group[users$age <= 40 & users$age > 21] <- "Adult"
71 age_group[users$age > 40] <- "Old"
72
73
74 users$age_group <- age_group
75
76 dbWriteTable(connection, "users", users, overwrite=T)
77
78 dbReadTable(connection, "users")
79
80 #####
81
82 ##### HEAD AND TAIL FUNCTIONS
83 head(users)
84 tail(users)
85 summary(users)
86 #####
87 ##### dbClearResult() to resolve the warning messages
88 ##### dbWriteTable(dataset)
89
```

Console output:

```
> ##### HEAD AND TAIL FUNCTIONS
> head(users)
  id email password age age_group
1 1 user1@gmail.com pas 32 Adult
2 2 user2@gmail.com pas 5 Young
3 3 email_changed@gmail.com pas 38 Adult
4 4 user4@gmail.com pas 20 Young
5 5 user5@gmail.com pas 88 Old
6 6 user6@gmail.com pas 12 Young

> tail(users)
  id email password age age_group
1 1 user1@gmail.com pas 32 Adult
2 2 user2@gmail.com pas 5 Young
3 3 email_changed@gmail.com pas 38 Adult
4 4 user4@gmail.com pas 20 Young
5 5 user5@gmail.com pas 88 Old
6 6 user6@gmail.com pas 12 Young

> summary(users)
  id email password age
Min. :1.00 Length:6 Min. : 5.0
1st Qu.:12.25 Class :character Class :character 1st Qu.:14.0
Median :3.50 Mode :character Mode :character Median :26.0
Mean :3.50 Mean :32.5
3rd Qu.:4.75 3rd Qu.:36.5
Max. :6.00 Max. :88.0
age_group
Length:6
Class :character
Mode :character

> #####
>
```