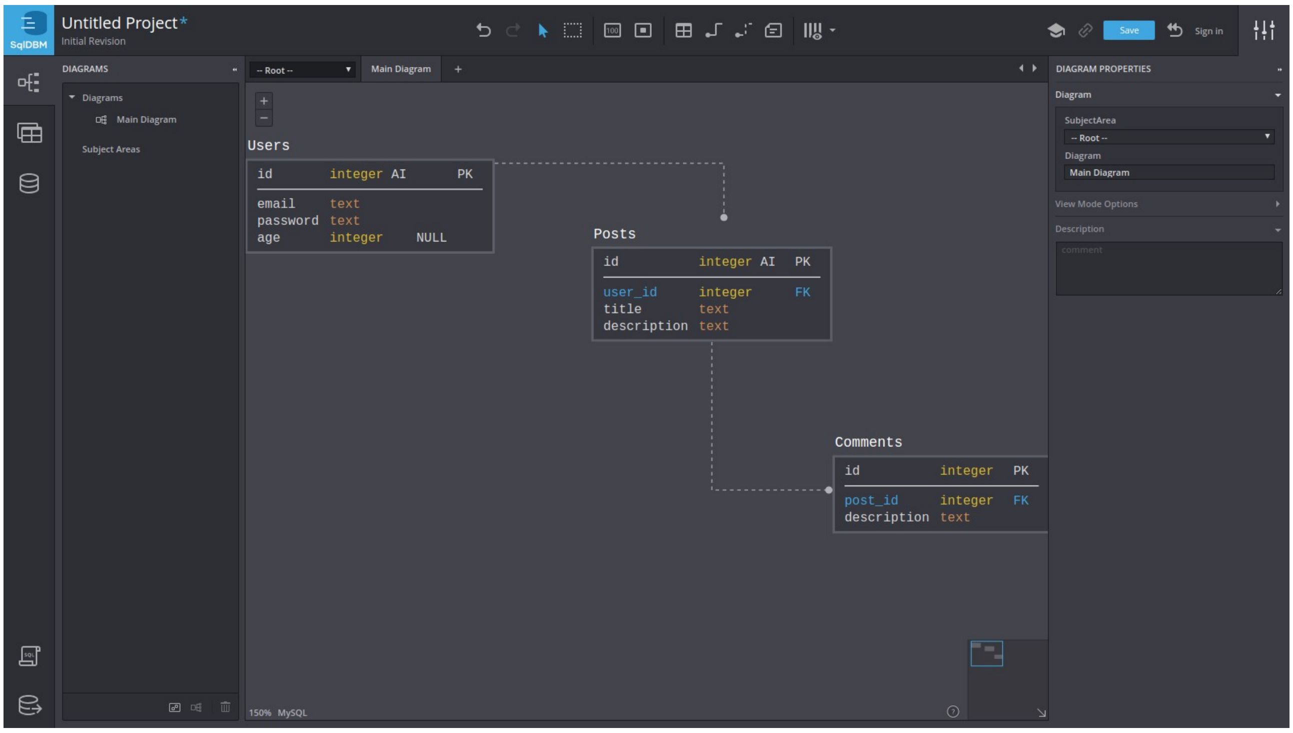
Big Data Lab Assignment 1

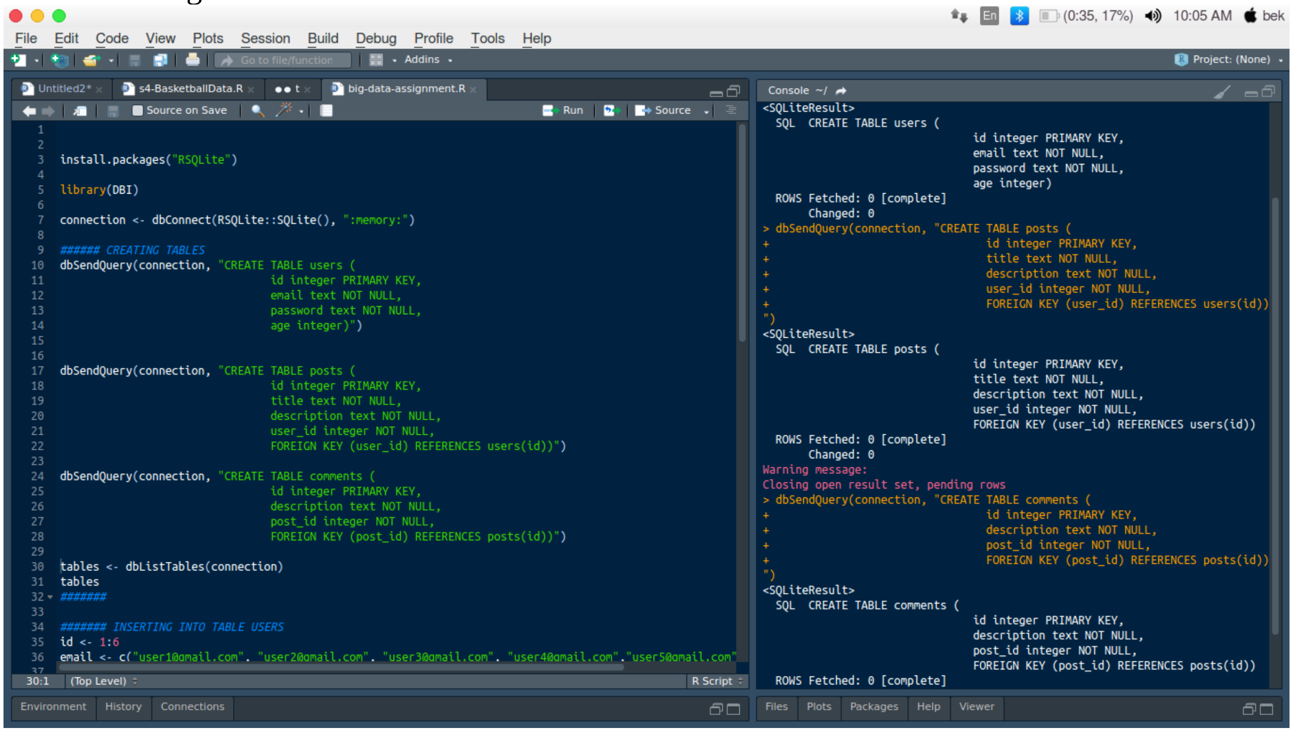
Group members
1.Bereket Abera
2.Bereket Gebredingle
3.Nathenael Tarekegn
4.Kaleb wondowssen

1. The database diagram



2. The package we used to connect to the database is **RSQLite**.

3. 3.1 Creating tables

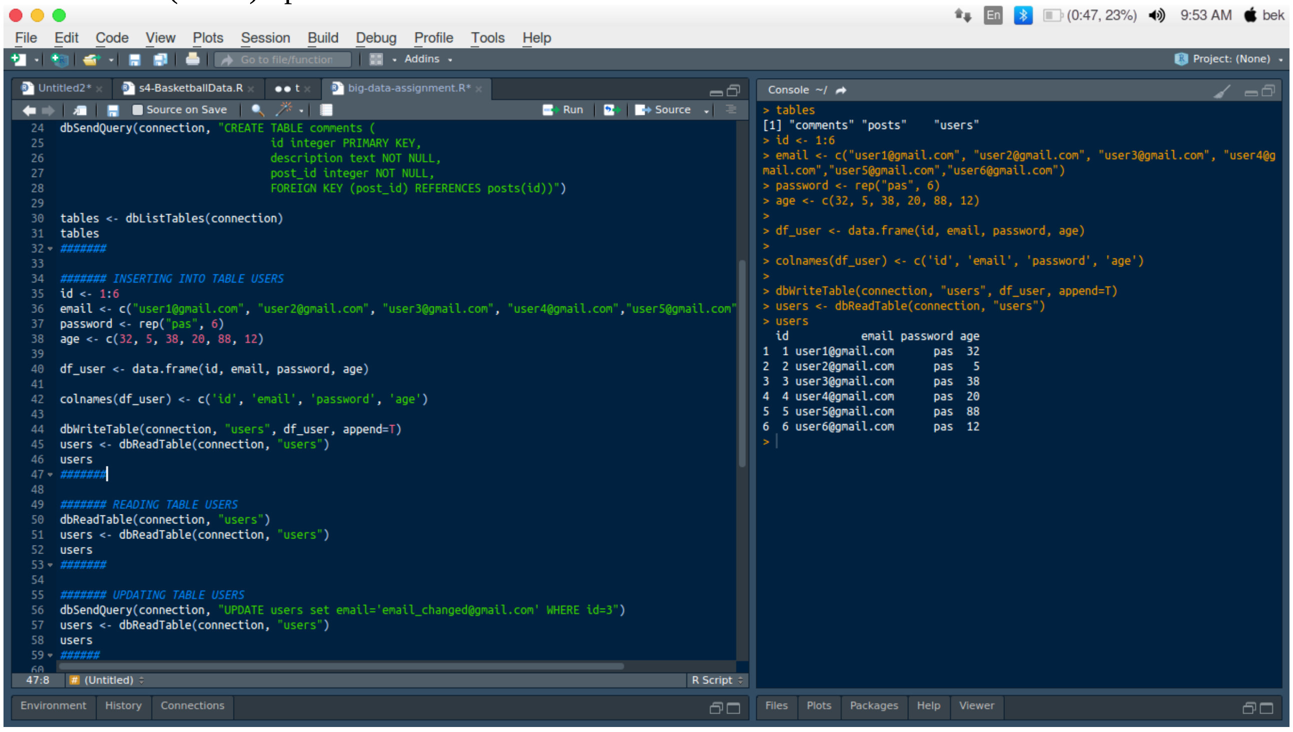


```
install.packages("RSQLite")

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@gmail.com", "user2@gmail.com", "user3@gmail.com", "user4@gmail.com","user5@gmail.com"
```

```
<SQLiteResult>
  SQL  CREATE TABLE users (
                                id integer PRIMARY KEY,
                                email text NOT NULL,
                                password text NOT NULL,
                                age integer)
  ROWS Fetched: 0 [complete]
       Changed: 0
> 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))
")
<SQLiteResult>
  SQL  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))
  ROWS Fetched: 0 [complete]
       Changed: 0
Warning message:
Closing open result set, pending rows
> 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))
")
<SQLiteResult>
  SQL  CREATE TABLE comments (
                                id integer PRIMARY KEY,
                                description text NOT NULL,
                                post_id integer NOT NULL,
                                FOREIGN KEY (post_id) REFERENCES posts(id))
  ROWS Fetched: 0 [complete]
```

3.2 CRUD operation

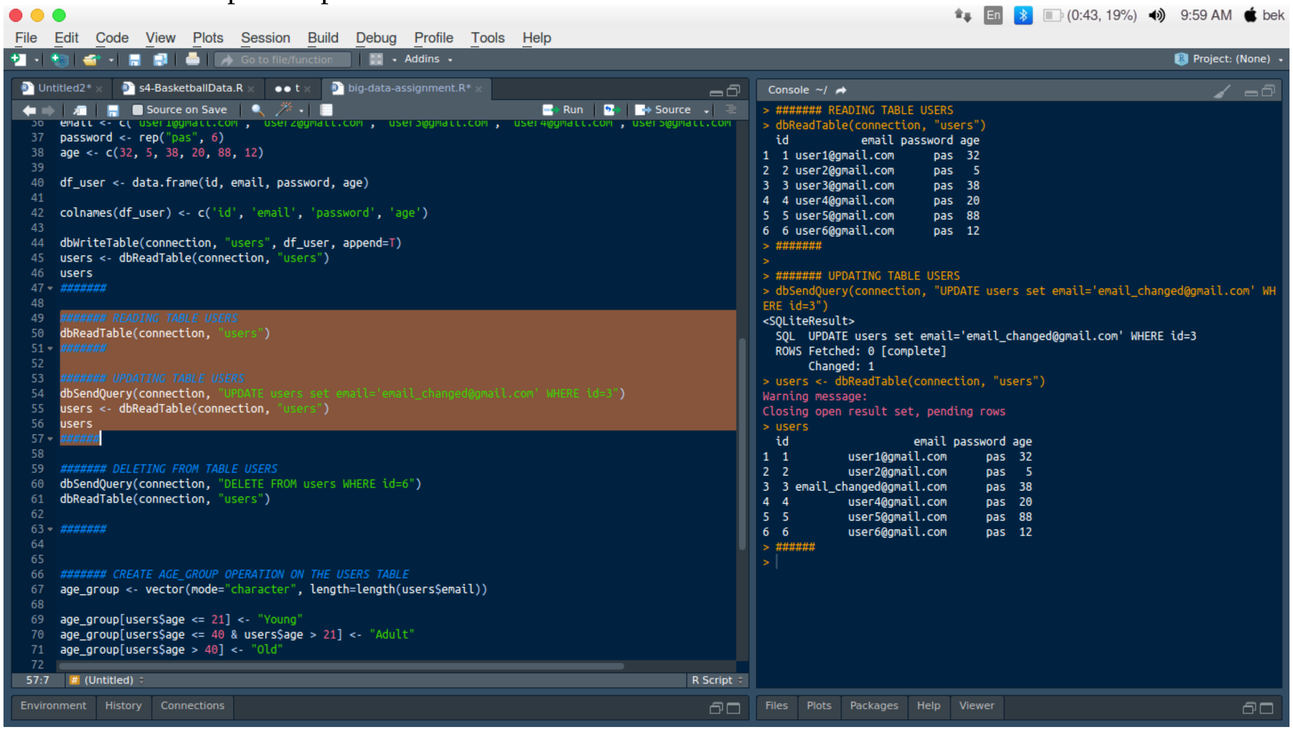
3.2.1 Create (insert) operation



```r
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@gmail.com", "user2@gmail.com", "user3@gmail.com", "user4@gmail.com","user5@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
#######

####### READING TABLE USERS
dbReadTable(connection, "users")
users <- dbReadTable(connection, "users")
users
#######

####### UPDATING TABLE USERS
dbSendQuery(connection, "UPDATE users set email='email_changed@gmail.com' WHERE id=3")
users <- dbReadTable(connection, "users")
users
######
```

```
> tables
[1] "comments" "posts"    "users"
> id <- 1:6
> email <- c("user1@gmail.com", "user2@gmail.com", "user3@gmail.com", "user4@g
mail.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
>
```

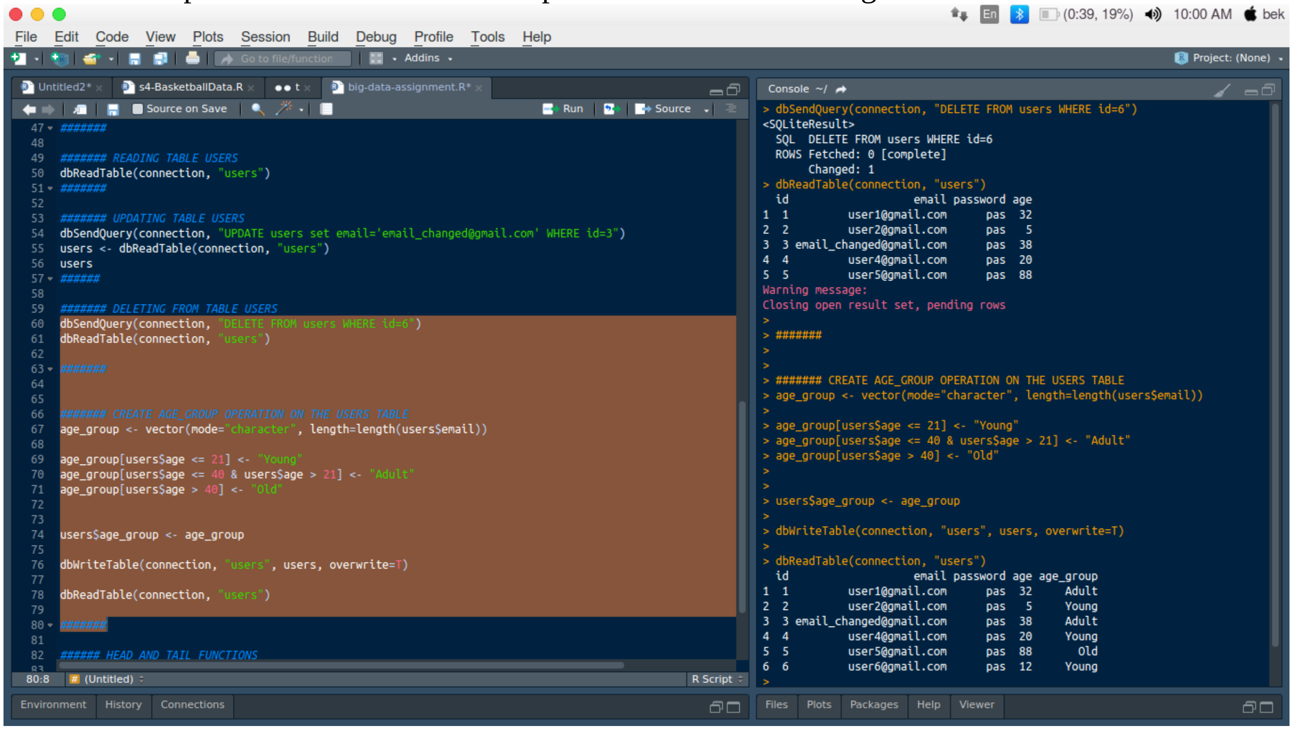
3.2.2 Read and Update operations



```r
email <- c("user1@gmail.com", "user2@gmail.com", "user3@gmail.com", "user4@gmail.com","user5@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
#######

####### READING TABLE USERS
dbReadTable(connection, "users")
#######

####### UPDATING TABLE USERS
dbSendQuery(connection, "UPDATE users set email='email_changed@gmail.com' WHERE id=3")
users <- dbReadTable(connection, "users")
users
######

####### DELETING FROM TABLE USERS
dbSendQuery(connection, "DELETE FROM users WHERE id=6")
dbReadTable(connection, "users")

#######


####### 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"
```

```
> ####### 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' WH
ERE 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
> ######
>
```

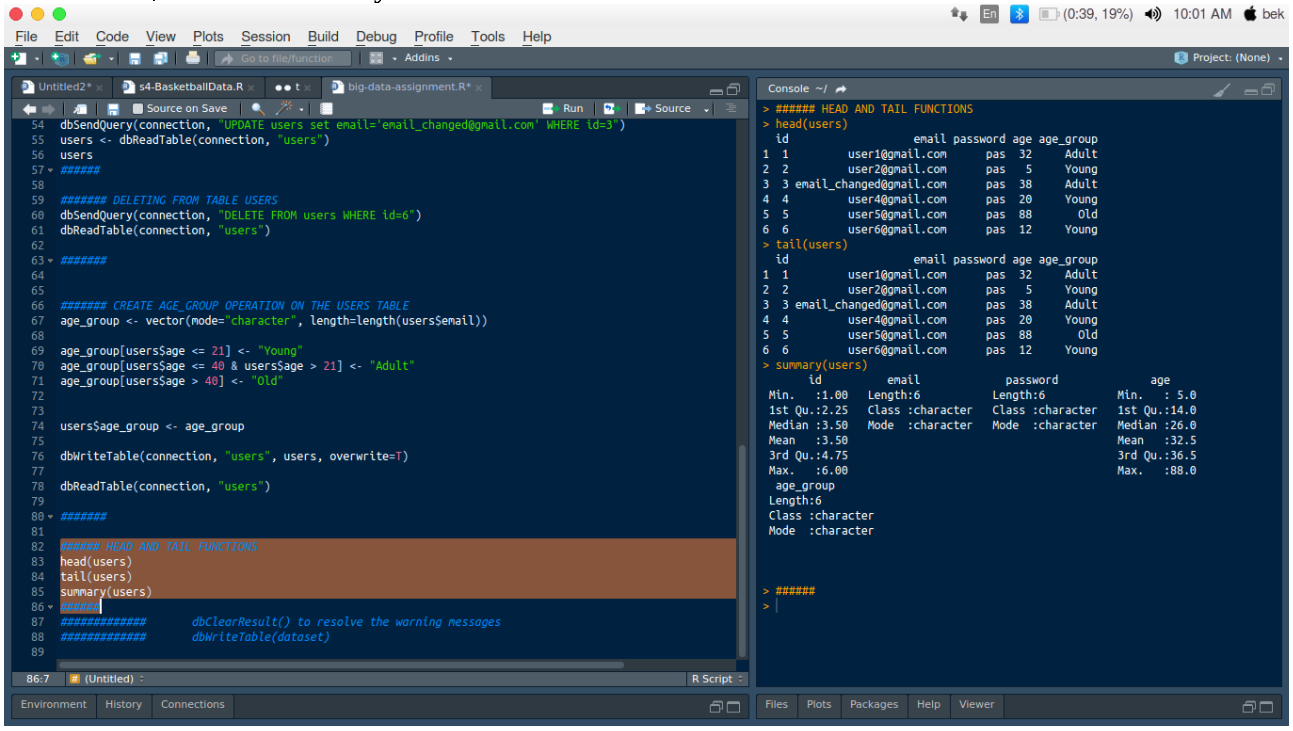
3.2.3 Delete operation and some kind of operation based on existing columns



```r
#######

####### READING TABLE USERS
dbReadTable(connection, "users")
#######

####### UPDATING TABLE USERS
dbSendQuery(connection, "UPDATE users set email='email_changed@gmail.com' WHERE id=3")
users <- dbReadTable(connection, "users")
users
######

####### DELETING FROM TABLE USERS
dbSendQuery(connection, "DELETE FROM users WHERE id=6")
dbReadTable(connection, "users")

#######



####### 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")

#######

###### HEAD AND TAIL FUNCTIONS
```

```
> 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



```r
dbSendQuery(connection, "UPDATE users set email='email_changed@gmail.com' WHERE id=3")
users <- dbReadTable(connection, "users")
users
######

####### DELETING FROM TABLE USERS
dbSendQuery(connection, "DELETE FROM users WHERE id=6")
dbReadTable(connection, "users")

#######


####### 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")

#######

###### HEAD AND TAIL FUNCTIONS
head(users)
tail(users)
summary(users)
######
#############       dbClearResult() to resolve the warning messages
#############       dbWriteTable(dataset)

```

```
> ###### 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           Length:6           Min.   : 5.0
 1st Qu.:2.25   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



> ######
> 
```