2020-2021 FALL SEMESTER

MARMARA UNIVERSITY

DEPARTMENT OF INDUSTRIAL ENGINEERING

IE3045.1 Introduction to Management Information Systems

TERM PROJECT

Prepared by:

Group Name: THE DATA

Hatice Sena TEMİZ	150818822
İremnur BOYACI	150818821
Selcan AKBULUT	150816043
Mert HIRKA	150319573
Nilvana KARABULUT	150319664

Instructor: Prof. Dr. Beytullah Gültekin ÇETİNER

Date of Submission of The Report: 09.02.2021

I.PROJECT NAME: DATABASE OF MARMARA UNIVERSITY

II.PROJECT OBJECTIVE

Marmara University's data are physically stored on folders. However, due to coronavirus, inspectors are prohibited from going to schools. In this case, university administrators made an agreement with a software developer to transmit university data to inspectors. The software developer will transfer the data of the university to the internet using MySQL. As a result, the inspectors will be able to check the university's data on the internet. At the same time, thanks to these data, it will be easier for university administrators to store university information and to control the data.

III.ENTITY NAMES

- course,
- section,
- department,
- requisite,
- time_slot,
- takes,
- user_role,
- teachers,
- user,
- user_salary,
- role,
- user_number,
- user_balance,
- user_fax,
- user_finaid_map,
- user_address,
- finaid_main,
- user_email,
- hold main,
- user_hold_map

Relational Sentences:

Each SECTION may be having one or only one COURSE

Each COURSE may be assigned to one or more SECTION

Each SECTION may be given by one or more TEACHERS

Each TEACHERS may be assigned to one or only one SECTION

Each COURSE may be managed by one or only one DEPARTMENT

Each DEPARTMENT may be having one or more COURSE

Each COURSE may be having one or more REQUISITE

Each REQUISITE may be used for one or only one COURSE

Each SECTION may be having one or more TAKES

Each TAKES may be assigned to one or only one SECTION

Each HOLD MAIN may be managed by one or more USER HOLD MAP

Each USER HOLD MAP may be used for one or only one HOLD MAIN

Each USER may be registered for one or more USER NUMBER

Each USER NUMBER may be assigned as one or only one USER

Each USER may be under one or more USER ADDRESS

Each USER ADDRESS may be consisting of one or only one USER

Each USER may be an entity under one or more USER MAIL

Each USER MAIL may be defined as one or only one USER

Each USER may be developing one or more HOLD MAIN

Each HOLD MAIN may be containing one or only one USER

Each USER may be registered for one or more USER HOLD MAP

Each USER HOLD MAP may be for one or only one USER

Each TIME SLOT may be assigned to one or more SECTION

Each SECTION may be assigned to one or only one TIME SLOT

Each USER ROLE may be given by one or only one TAKES

Each TAKES may be given by one or more USER ROLE

Each TEACHERS may be assigned to one or only one USER ROLE

Each USER ROLE may be assigned to one or more TEACHERS

Each ROLE may be defined as one or more USER ROLE

Each USER ROLE may be defined as one or only one ROLE

Each USER ROLE may be for one or only one USER

Each USER may be for one or more USER ROLE

Each USER may be given by one or more USER SALARY

Each USER SALARY may be assigned to one or only one USER

Each USER BALANCE may be used for one or only one USER

Each USER may be having one or more USER BALANCE

Each USER may be assigned to one or more FAX

Each FAX may be having one or only one USER

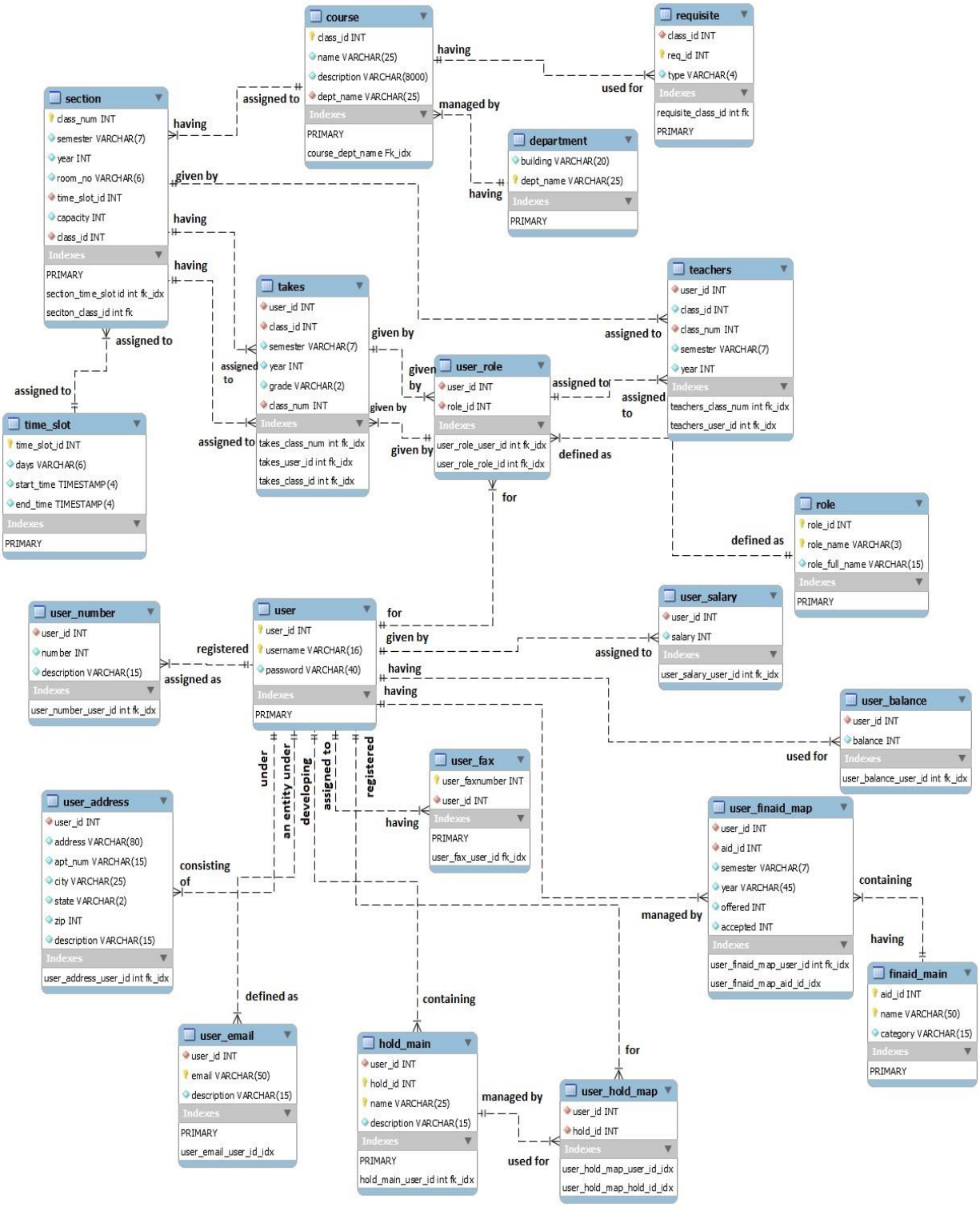
Each USER may be having one or more USER FINAID MAP

Each USER FINAID MAP may be managed by one or only one USER

Each USER FINAID MAP may be containing one or only one FINAID MAIN

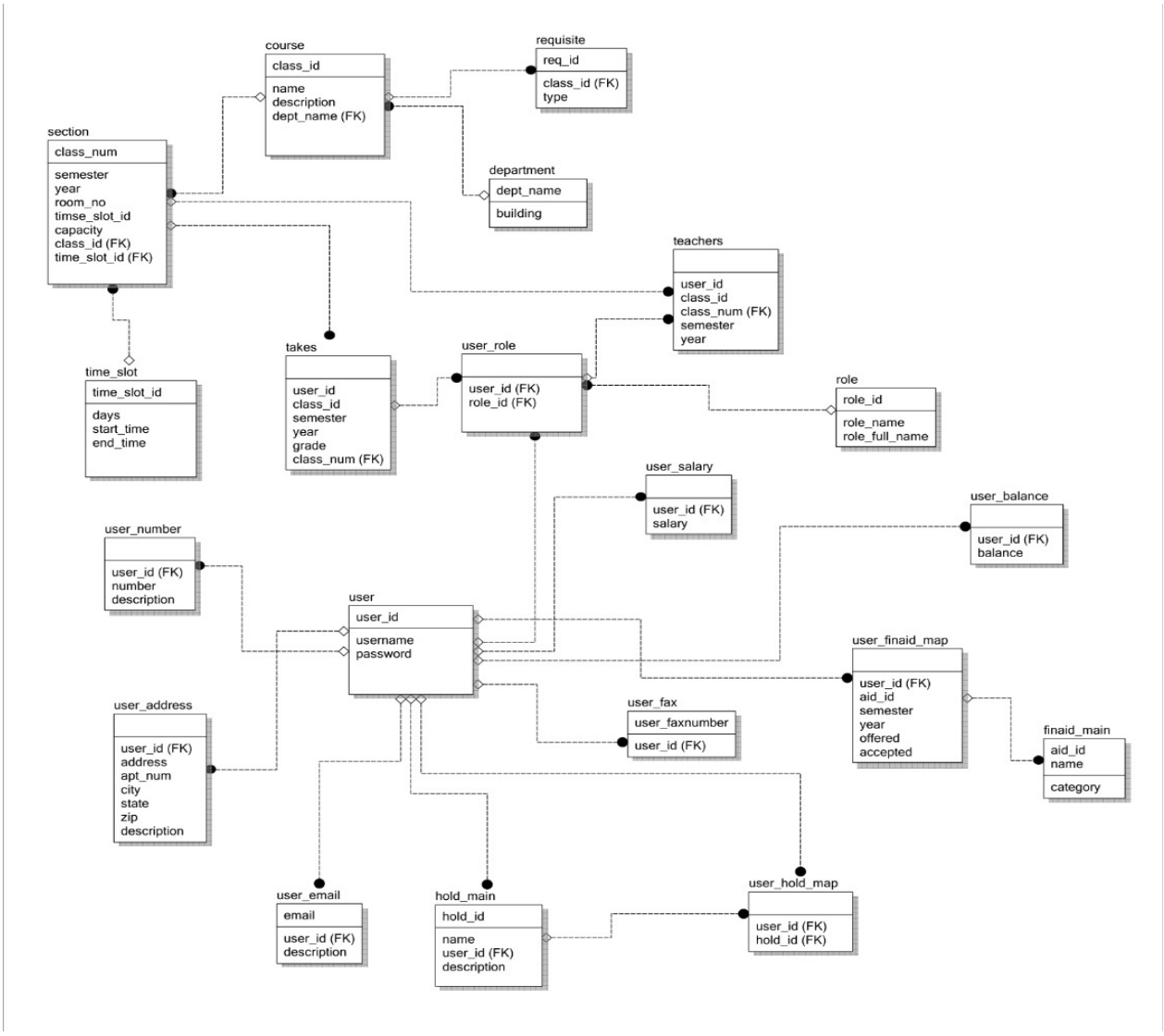
Each FINAID MAIN may be having one or more USER FINAID MAP

IV.ENTITY RELATIONSHIP (ER) DIAGRAM BY USING ORACLE BARKER NOTATIONS

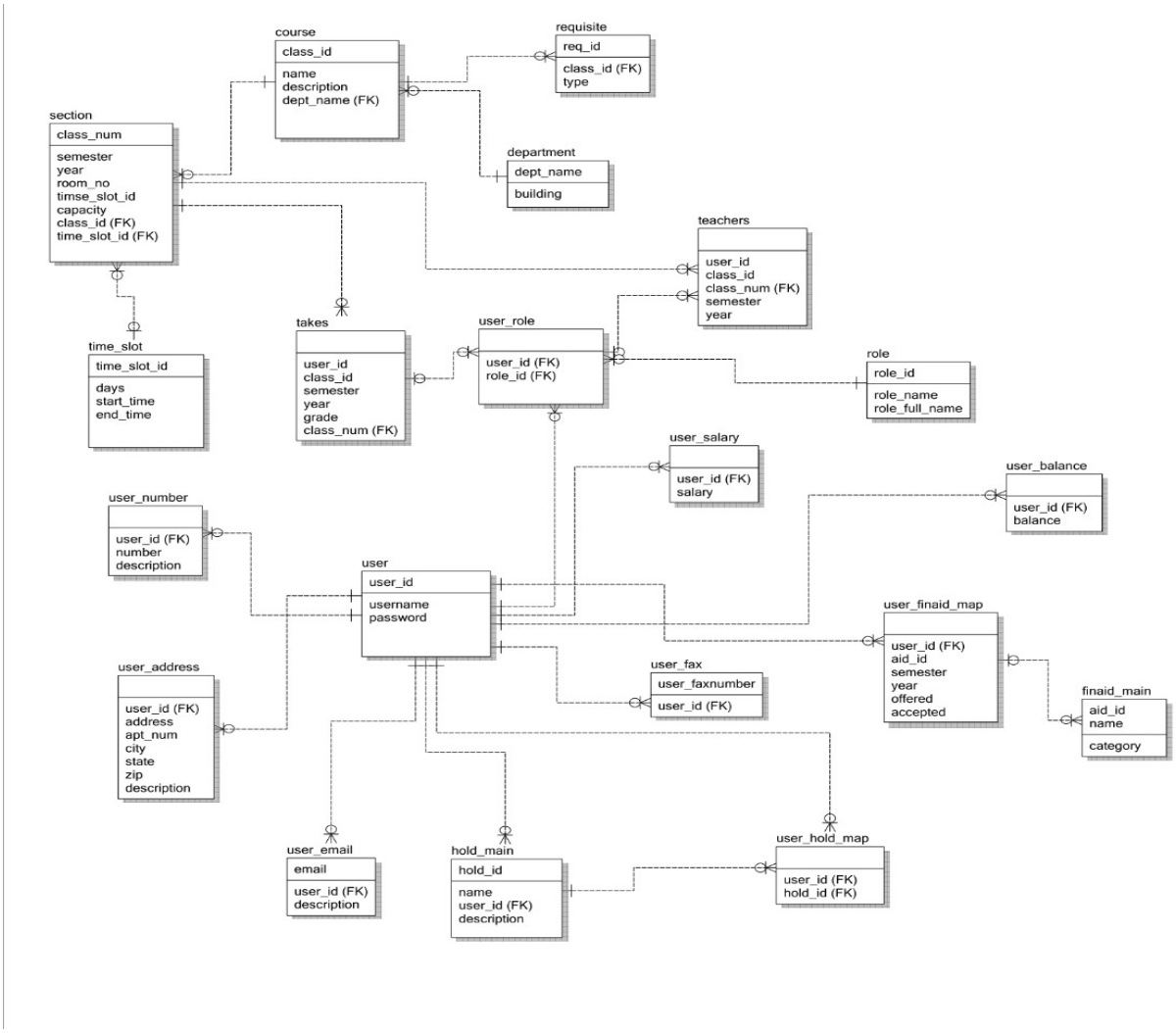


V.ENTITY RELATIONSHIP (ER) DIAGRAM BY USING ERWIN IDEF1X NOTATIONS

IDEF1X NOTATIONS:



IE NOTATIONS:



VI.SELECTION OF DBMS AND WHY YOU CHOSE IT

We preferred to use MYSQL in this project. There are some reasons for this. MYSQL is advantageous in terms of data security. MySQL requires less operational storage space. It also provides the memory caches needed for fast, improved performance. In addition to these, the ease of use directed us to this. It is also an advantage for us that we can download it for free.

VII.SQL DATA DEFINITION LANGUAGE- DDL STATEMENTS FOR ALL DATABASE (20 TABLES AND RELATIONS)

```
CREATE TABLE course
(
class_id          INTEGER NOT NULL,
name              VARCHAR(25) NOT NULL,
description       VARCHAR(8000) NOT NULL,
dept_name         VARCHAR(25) NOT NULL
);

ALTER TABLE course
ADD PRIMARY KEY (class_id);

CREATE TABLE department
(
dept_name         VARCHAR(25) NOT NULL,
building          VARCHAR(20) NOT NULL
);
ALTER TABLE department
ADD PRIMARY KEY (dept_name);

CREATE TABLE finaid_main
(
aid_id            INTEGER NOT NULL,
name              VARCHAR(50) NOT NULL,
category          VARCHAR(15) NOT NULL
);
```

```
ALTER TABLE finaid_main
ADD PRIMARY KEY (aid_id,name);

CREATE TABLE hold_main
(
user_id            INTEGER NOT NULL,
hold_id            INTEGER NOT NULL,
name               VARCHAR(25) NOT NULL,
description        VARCHAR(15) NOT NULL
);

ALTER TABLE hold_main
ADD PRIMARY KEY (hold_id);

CREATE TABLE requisite
(
class_id           INTEGER NOT NULL,
req_id             INTEGER NOT NULL,
type               VARCHAR(4) NOT NULL
);

ALTER TABLE requisite
ADD PRIMARY KEY (req_id);
```

```
CREATE TABLE role
(
role_id             INTEGER NOT NULL,
role_name           VARCHAR(3) NULL,
role_full_name      VARCHAR(15) NOT NULL
);

ALTER TABLE role
ADD PRIMARY KEY (role_id);

CREATE TABLE section
(
class_num           INTEGER NOT NULL,
semester            VARCHAR(7) NOT NULL,
year                INTEGER NOT NULL,
room_no             VARCHAR(6) NOT NULL,
timse_slot_id       INTEGER NOT NULL,
capacity            INTEGER NOT NULL,
class_id            INTEGER NOT NULL,
time_slot_id        INTEGER NULL
);

ALTER TABLE section
ADD PRIMARY KEY (class_num);
```

```
CREATE TABLE takes
(
user_id            INTEGER NOT NULL,
class_id           INTEGER NOT NULL,
semester           VARCHAR(7) NOT NULL,
year               INTEGER NOT NULL,
grade              VARCHAR(2) NOT NULL,
class_num          INTEGER NOT NULL
);

CREATE TABLE teachers
(
user_id            INTEGER NOT NULL,
class_id           INTEGER NOT NULL,
class_num          INTEGER NOT NULL,
semester           VARCHAR(7) NOT NULL,
year               INTEGER NOT NULL
);

CREATE TABLE time_slot
(
time_slot_id       INTEGER NOT NULL,
days               VARCHAR(6) NOT NULL,
start_time         TIMESTAMP NOT NULL,
end_time           TIMESTAMP NOT NULL
);

ALTER TABLE time_slot
ADD PRIMARY KEY (time_slot_id);
```

```
CREATE TABLE user
(
user_id             INTEGER NOT NULL,
username            VARCHAR(16) NULL,
password            VARCHAR(40) NOT NULL
);
ALTER TABLE user
ADD PRIMARY KEY (user_id);

CREATE TABLE user_address
(
user_id             INTEGER NOT NULL,
address             VARCHAR(80) NOT NULL,
apt_num             VARCHAR(15) NOT NULL,
city                VARCHAR(25) NOT NULL,
state               VARCHAR(2) NOT NULL,
zip                 INTEGER NOT NULL,
description         VARCHAR(15) NOT NULL
);
CREATE TABLE user_balance
(
user_id             INTEGER NOT NULL,
balance             INTEGER NOT NULL
);
```

```
CREATE TABLE user_email
(
user_id             INTEGER NOT NULL,
email               VARCHAR(50) NOT NULL,
description         VARCHAR(15) NOT NULL
);

ALTER TABLE user_email
ADD PRIMARY KEY (email);

CREATE TABLE user_fax
(
user_faxnumber      INTEGER NOT NULL,
user_id             INTEGER NOT NULL
);

ALTER TABLE user_fax
ADD PRIMARY KEY (user_faxnumber);

CREATE TABLE user_finaid_map
(
user_id             INTEGER NOT NULL,
aid_id              INTEGER NOT NULL,
semester            VARCHAR(7) NOT NULL,
year                VARCHAR(45) NOT NULL,
offered             INTEGER NOT NULL,
accepted            INTEGER NOT NULL
);
```

```
CREATE TABLE user_hold_map
(
user_id            INTEGER NOT NULL,
hold_id            INTEGER NOT NULL
);

CREATE TABLE user_number
(
user_id            INTEGER NOT NULL,
number             INTEGER NOT NULL,
description        VARCHAR(15) NOT NULL
);

CREATE TABLE user_role
(
user_id            INTEGER NOT NULL,
role_id            INTEGER NOT NULL
);

CREATE TABLE user_salary
(
user_id            INTEGER NOT NULL,
salary             INTEGER NOT NULL
);

ALTER TABLE course
ADD CONSTRAINT R_7 FOREIGN KEY (dept_name) REFERENCES department
(dept_name);
```

```
ALTER TABLE hold_main
ADD CONSTRAINT R_53 FOREIGN KEY (user_id) REFERENCES user (user_id);

ALTER TABLE requisite
ADD CONSTRAINT R_5 FOREIGN KEY (class_id) REFERENCES course (class_id);

ALTER TABLE section
ADD CONSTRAINT R_3 FOREIGN KEY (class_id) REFERENCES course (class_id);

ALTER TABLE section
ADD CONSTRAINT R_14 FOREIGN KEY (time_slot_id) REFERENCES time_slot
(time_slot_id);

ALTER TABLE takes
ADD CONSTRAINT R_11 FOREIGN KEY (class_num) REFERENCES section
(class_num);

ALTER TABLE teachers
ADD CONSTRAINT R_9 FOREIGN KEY (class_num) REFERENCES section (class_num);

ALTER TABLE user_address
ADD CONSTRAINT R_59 FOREIGN KEY (user_id) REFERENCES user (user_id);

ALTER TABLE user_balance
ADD CONSTRAINT R_63 FOREIGN KEY (user_id) REFERENCES user (user_id);
ALTER TABLE user_email
ADD CONSTRAINT R_58 FOREIGN KEY (user_id) REFERENCES user (user_id);
```

```
ALTER TABLE user_fax
ADD CONSTRAINT R_51 FOREIGN KEY (user_id) REFERENCES user (user_id);

ALTER TABLE user_finaid_map
ADD CONSTRAINT R_61 FOREIGN KEY (user_id) REFERENCES user (user_id);

ALTER TABLE user_hold_map
ADD CONSTRAINT R_55 FOREIGN KEY (user_id) REFERENCES user (user_id);

ALTER TABLE user_hold_map
ADD CONSTRAINT R_57 FOREIGN KEY (hold_id) REFERENCES hold_main (hold_id);

ALTER TABLE user_number
ADD CONSTRAINT R_60 FOREIGN KEY (user_id) REFERENCES user (user_id);

ALTER TABLE user_role
ADD CONSTRAINT R_28 FOREIGN KEY (user_id) REFERENCES user (user_id);

ALTER TABLE user_role
ADD CONSTRAINT R_48 FOREIGN KEY (role_id) REFERENCES role (role_id);

ALTER TABLE user_salary
ADD CONSTRAINT R_49 FOREIGN KEY (user_id) REFERENCES user (user_id);
```

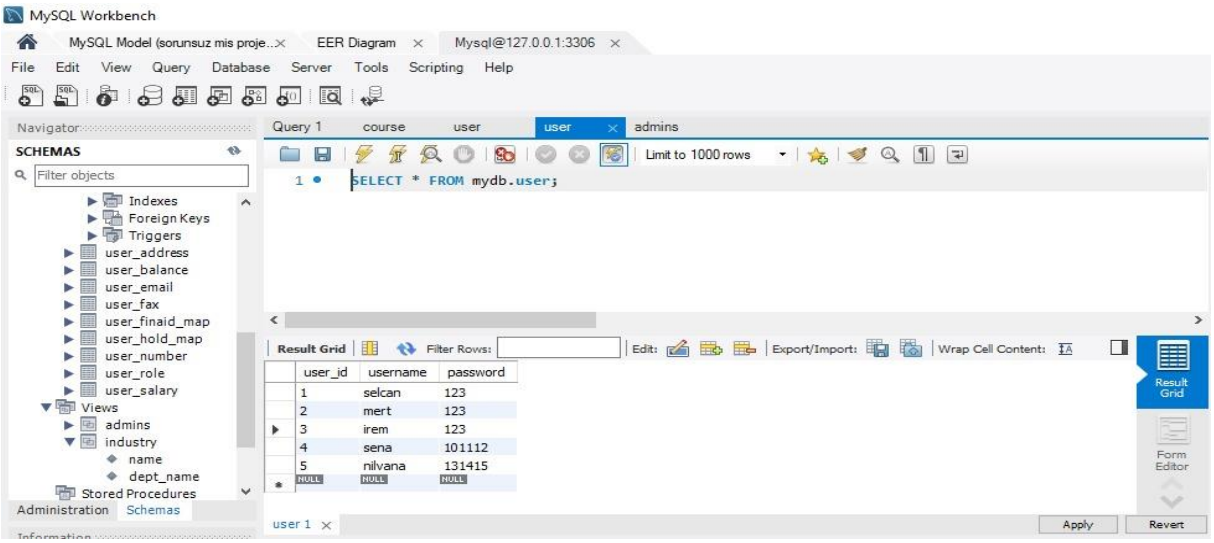
VIII.SELECTION OF THREE TABLES AND RELATIONS TO SHOW SQL EXPERIENCE

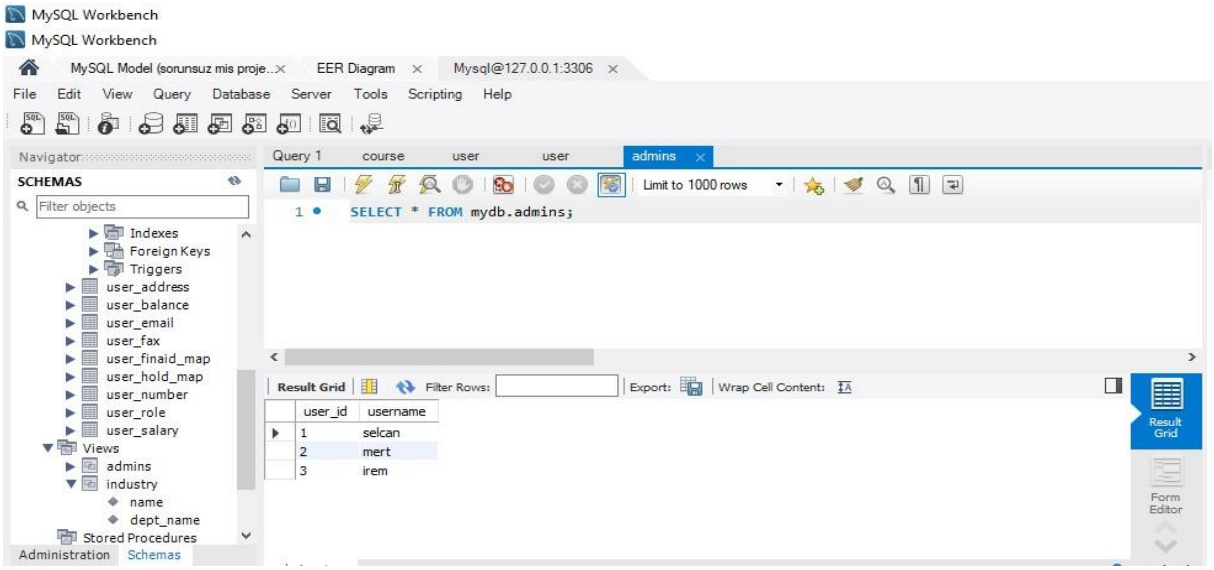
In this part, USER, ROLE and FINAID_MAIN table were selected as three tables.

A.ADDITIONAL DDL STATEMENTS -DATA DEFINITION LANGUAGE

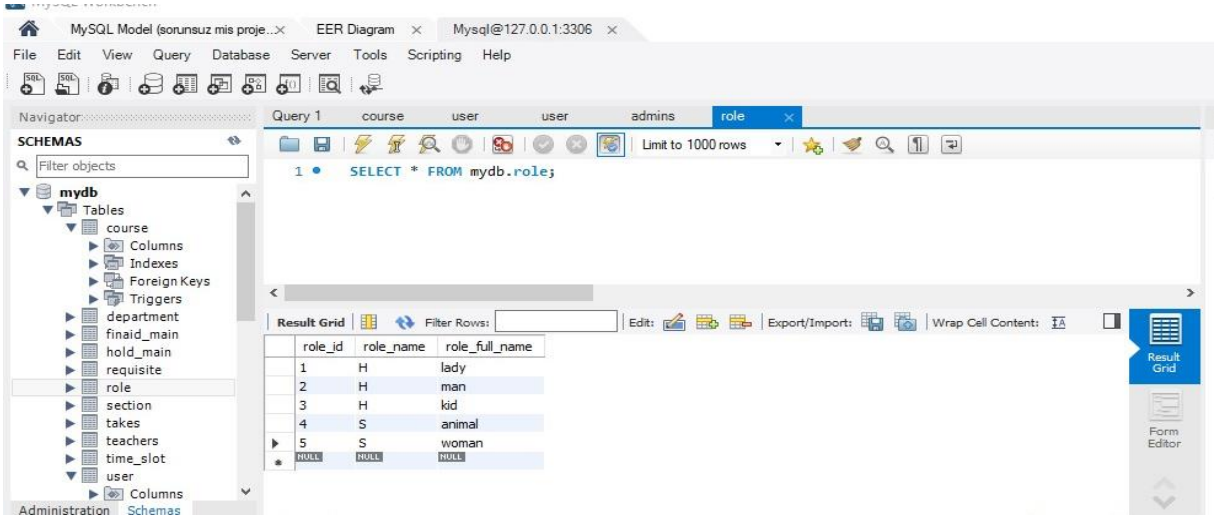
A.1. CREATE/ALTER VIEW EXAMPLES:

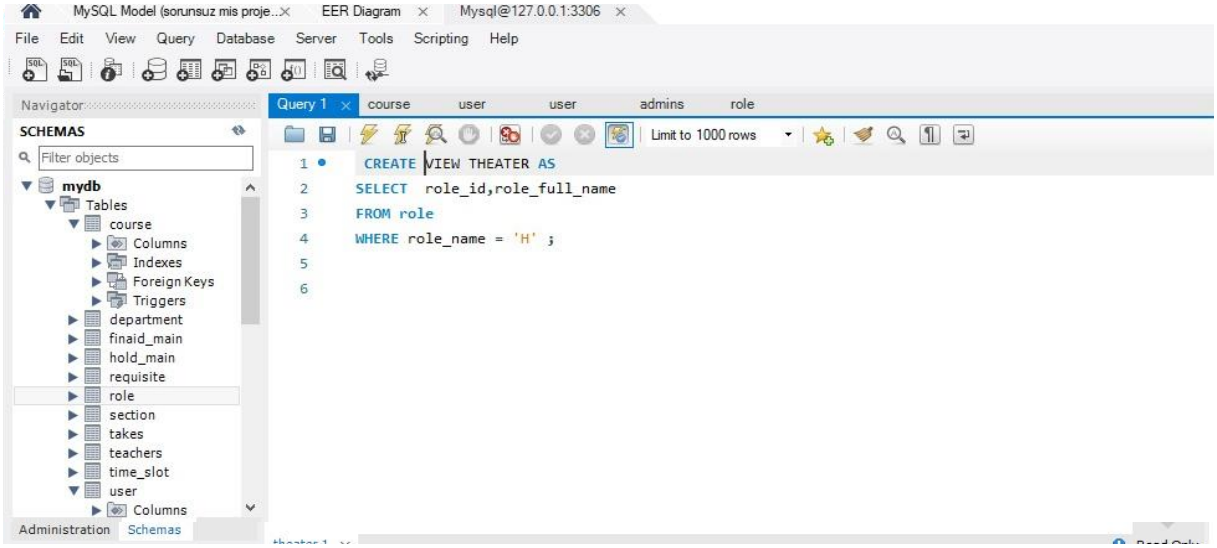
- **CREATE VIEW For USER :**



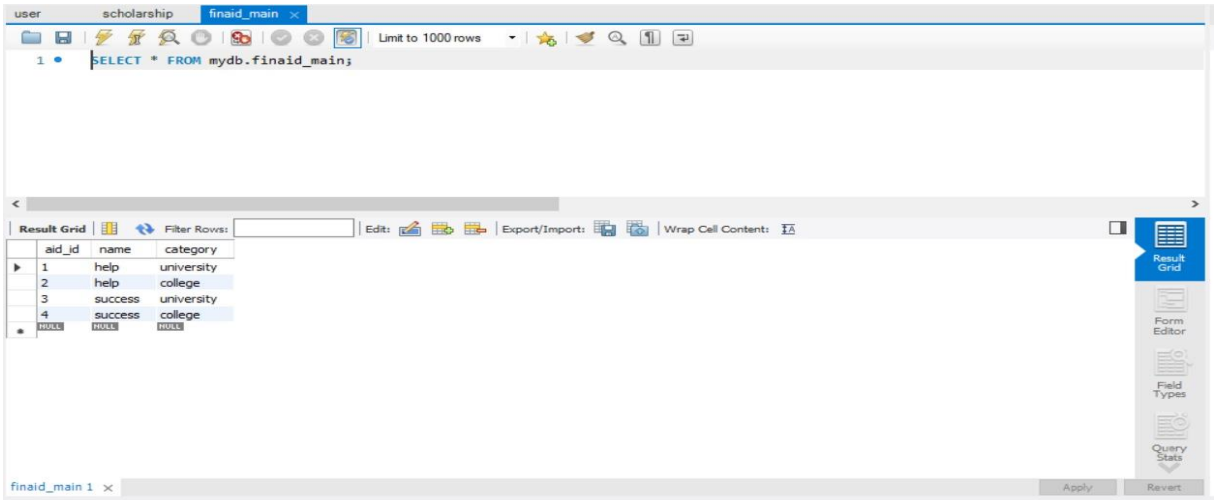


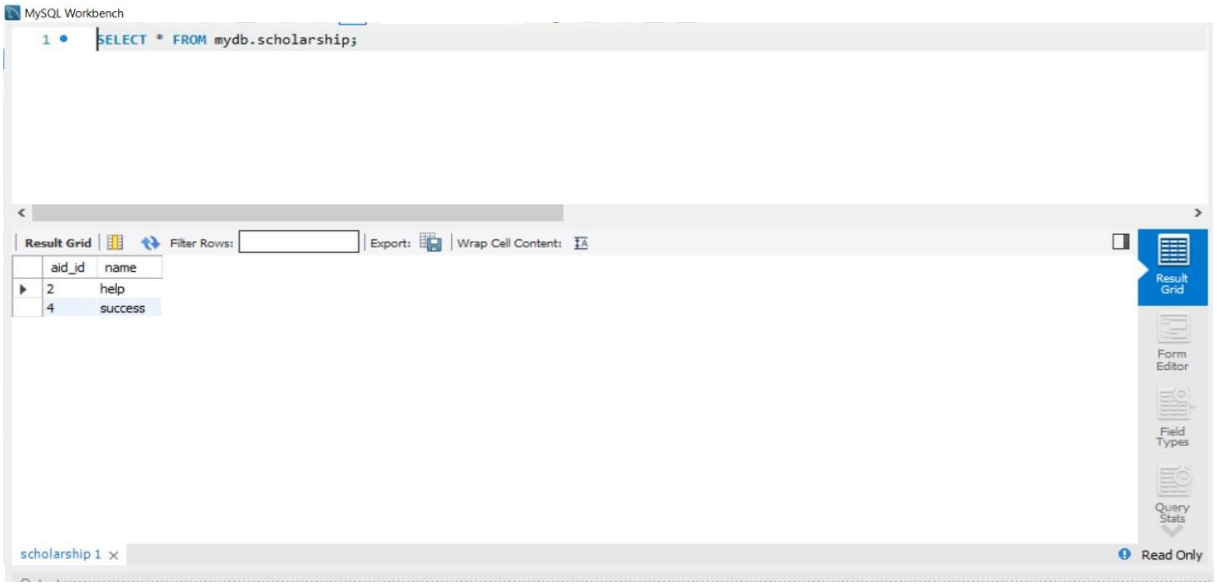
- **CREATE VIEW For ROLE:**



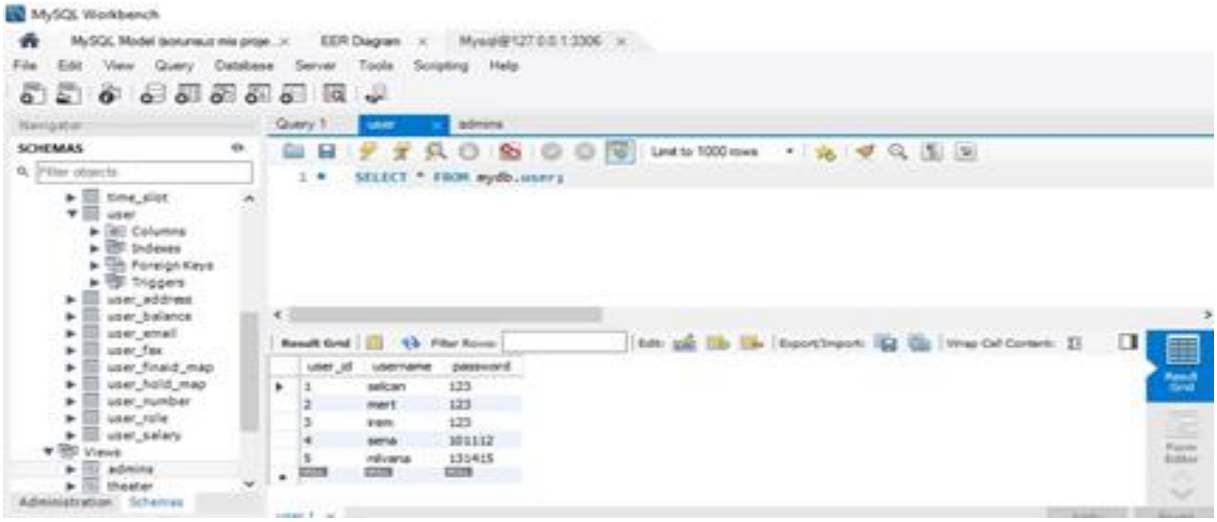


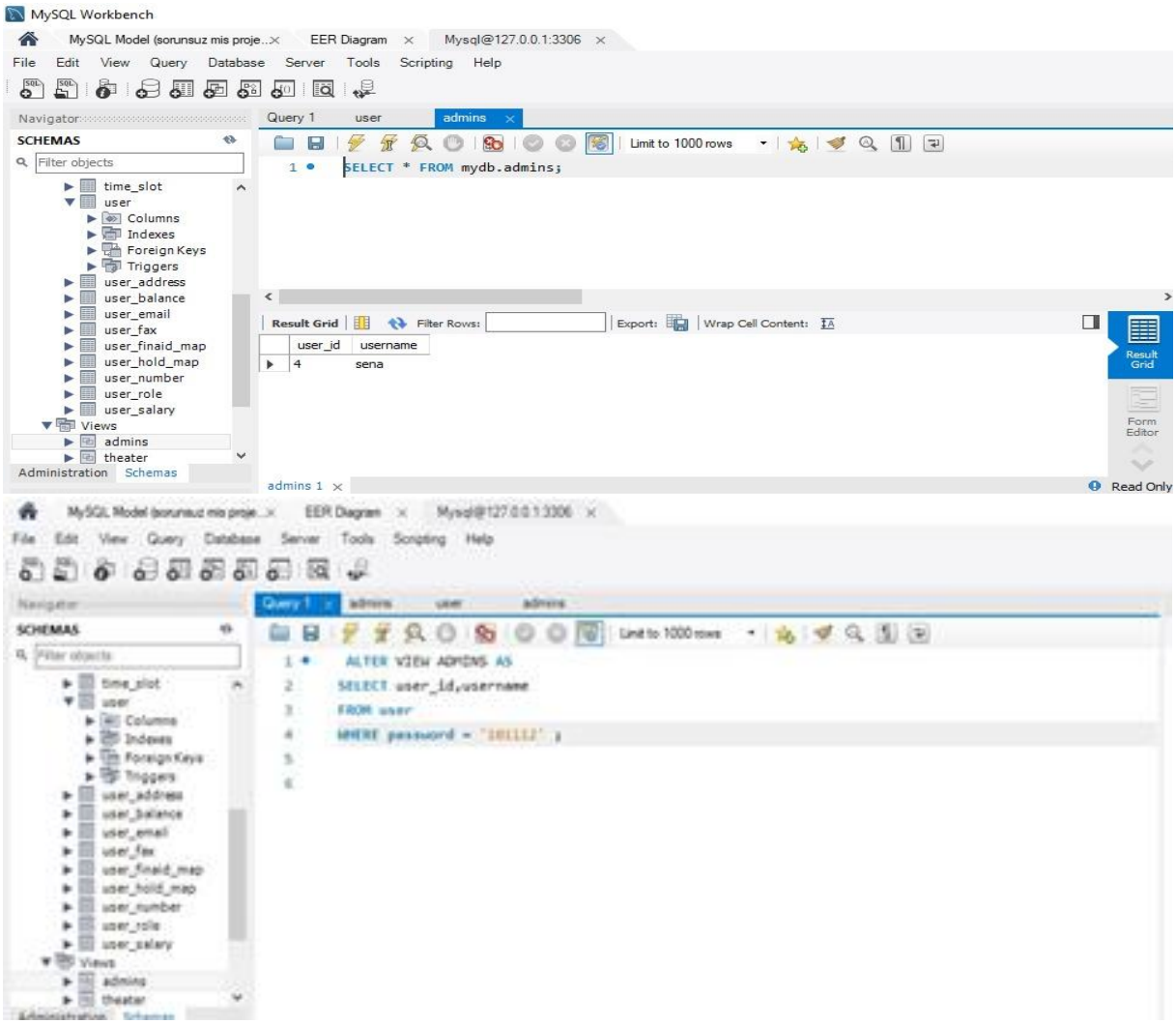
- **CREATE VIEW For FINAID_MAIN :**



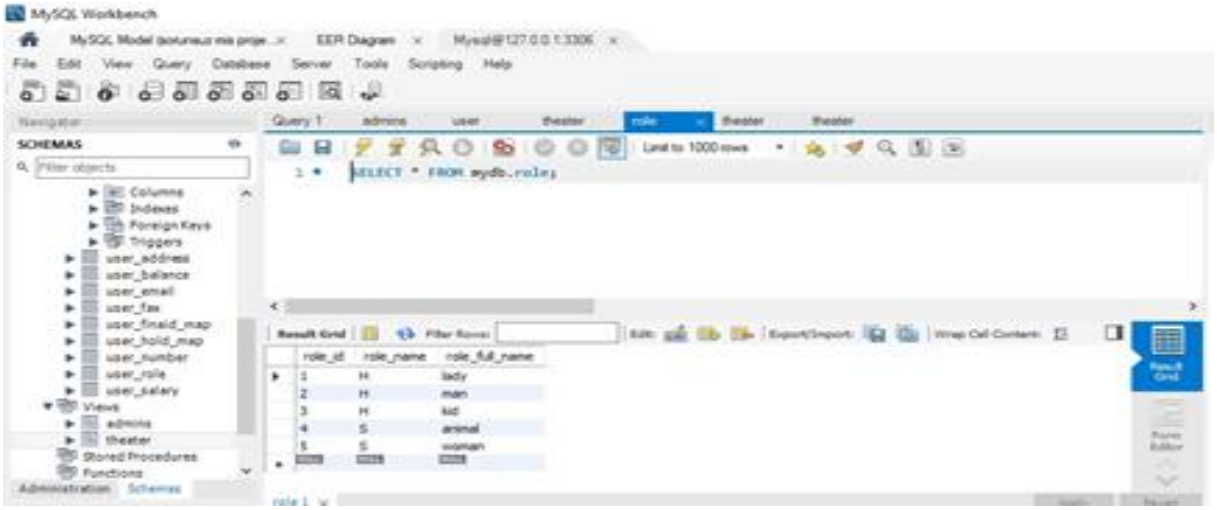


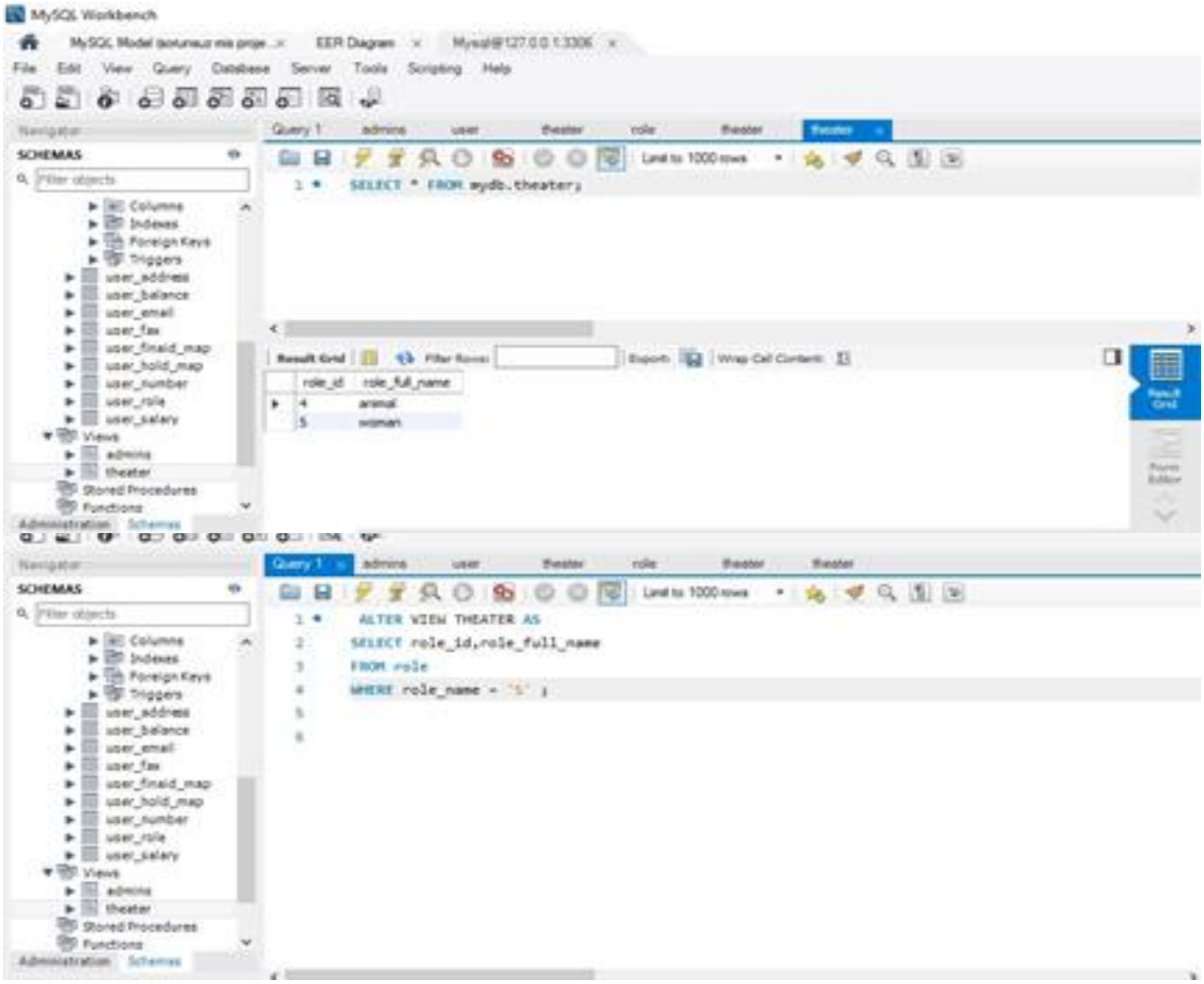
- **ALTER VIEW for USER:**



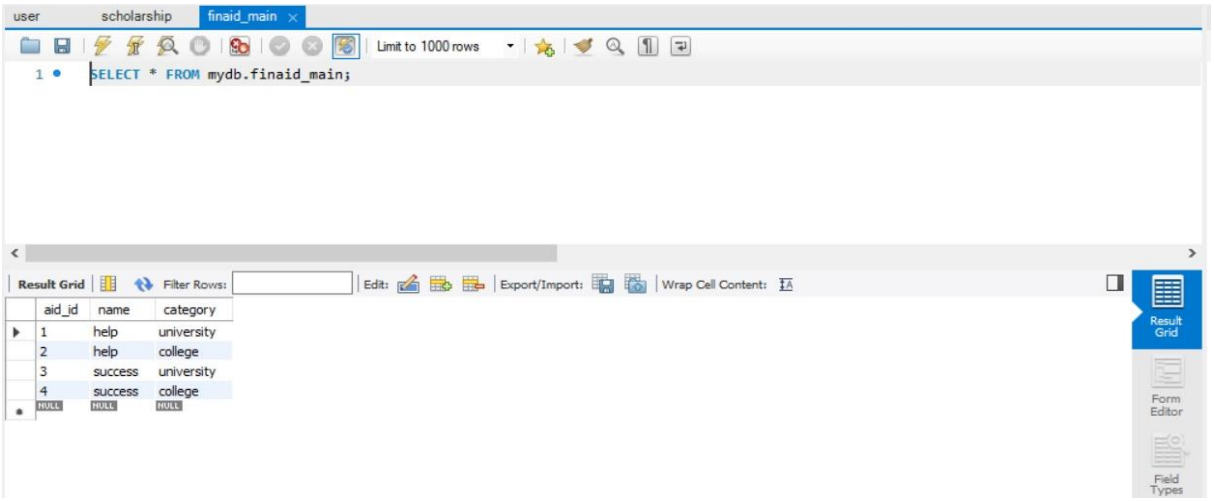


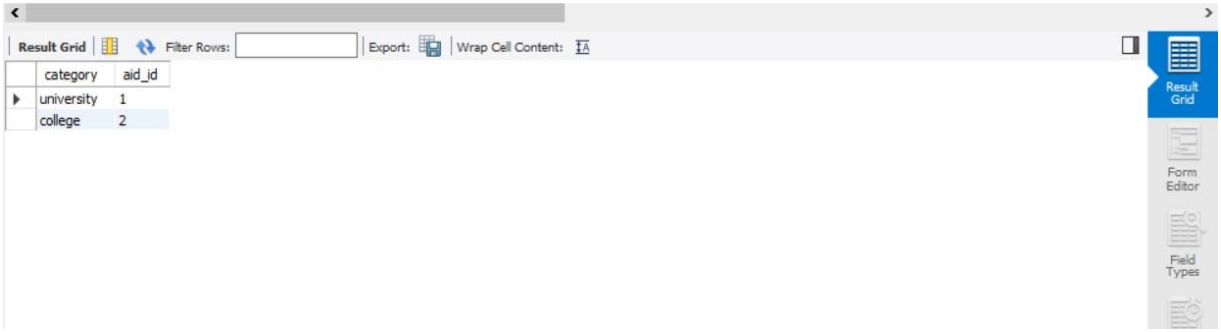
- **ALTER VIEW for ROLE:**





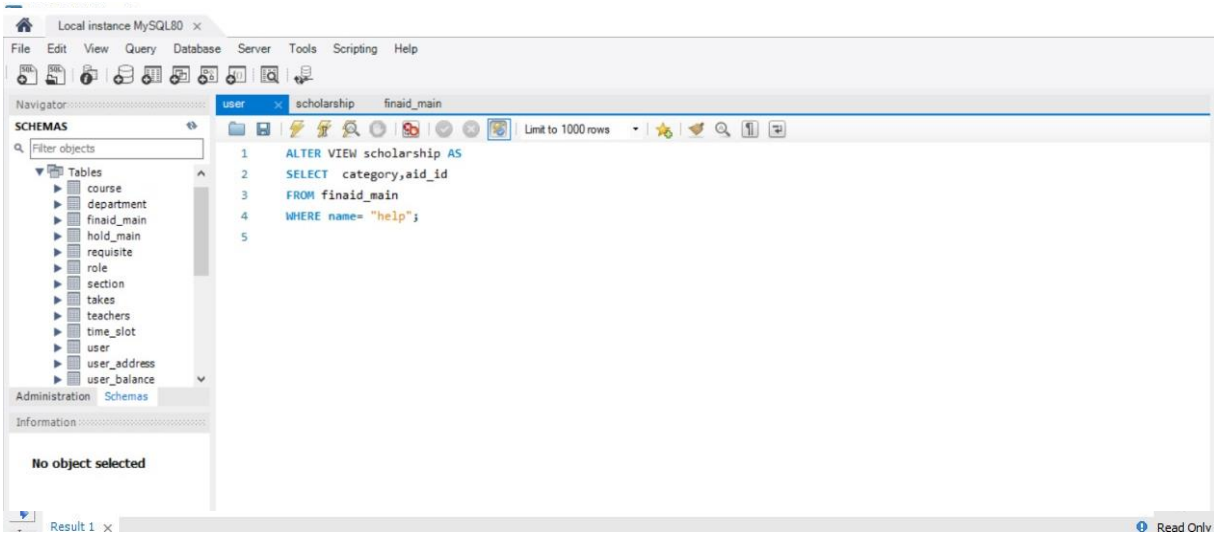
- **ALTER VIEW for FINAID_MAIN:**



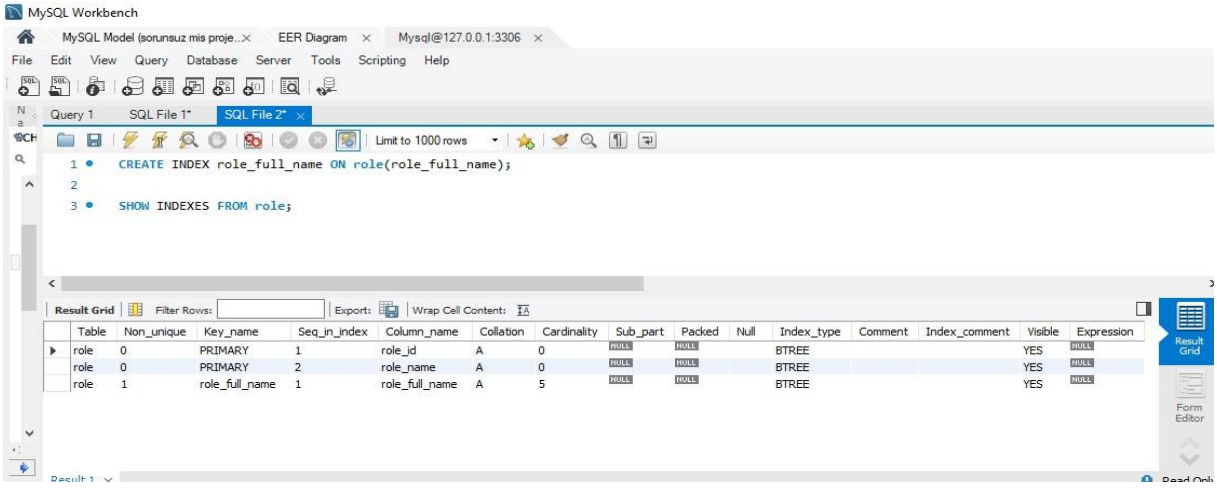


A.2 CREATE/ALTER INDEX EXAMPLES

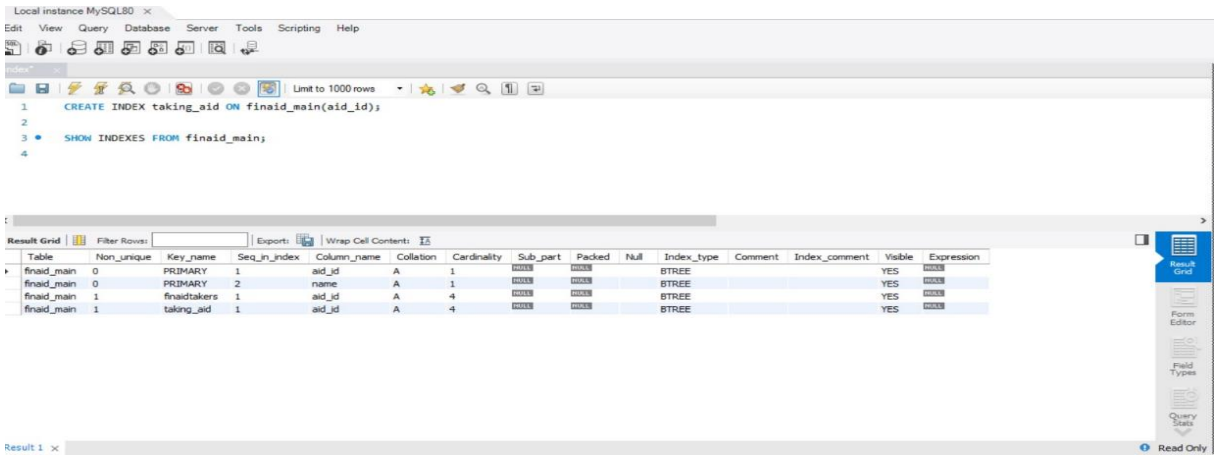
- **CREATE INDEX for USER:**



- **CREATE INDEX for ROLE:**

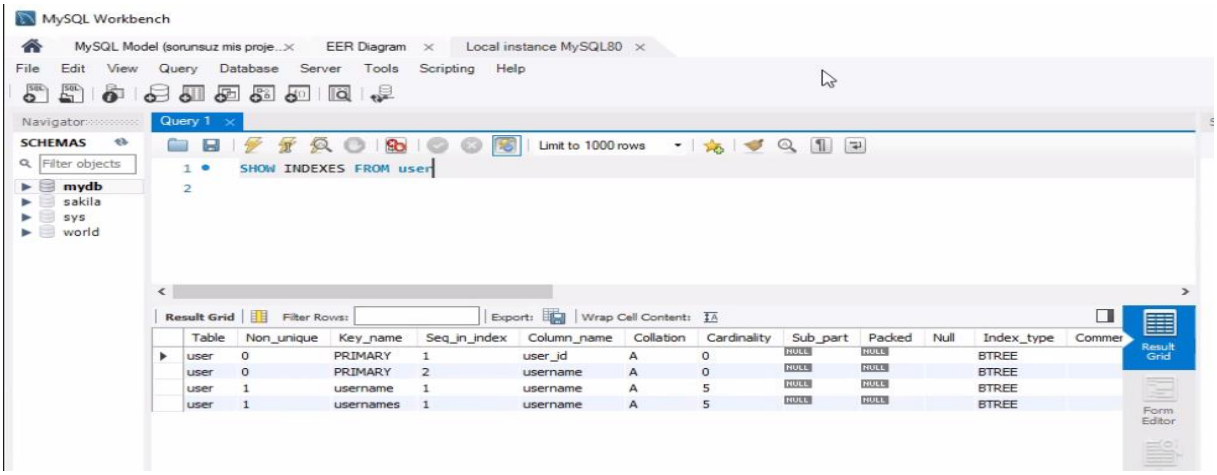


- **CREATE INDEX for FINAID_MAIN :**

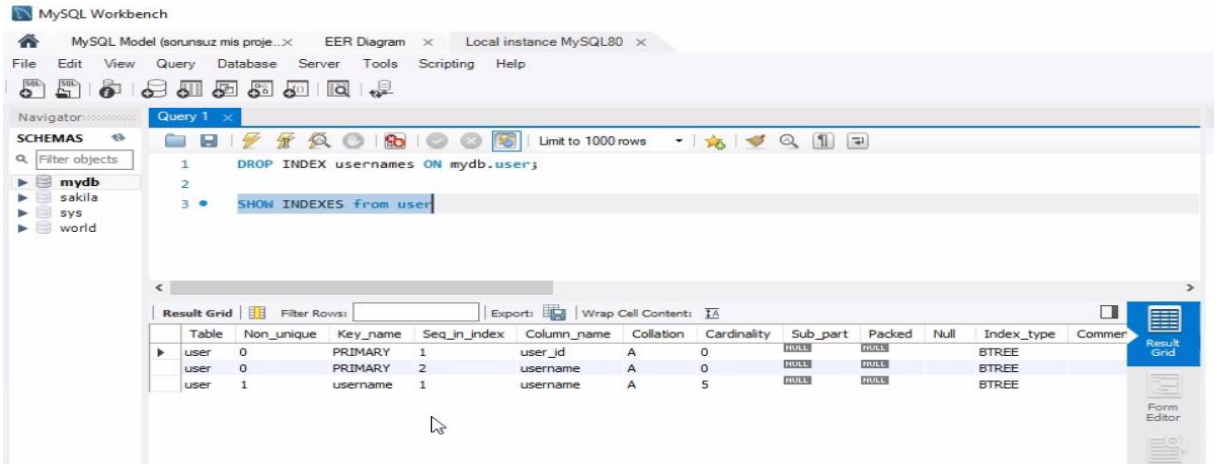


- **ALTER INDEX(DROP) for USER**

Before ALTER INDEX

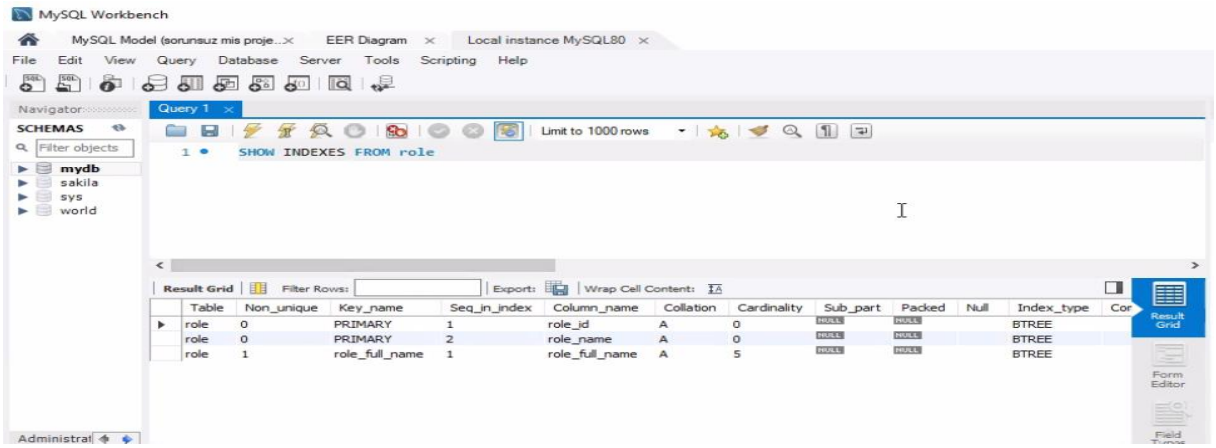


After ALTER INDEX

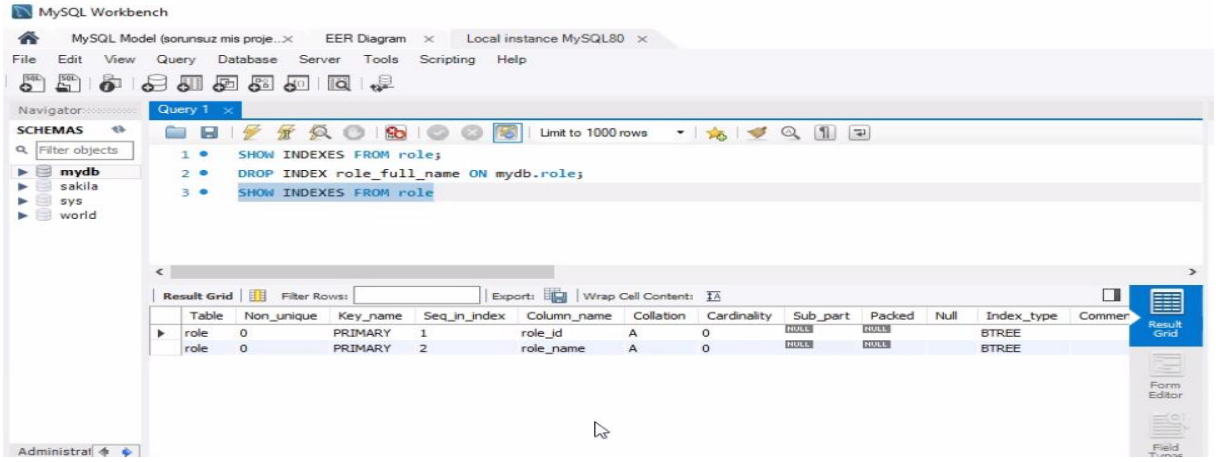


- **ALTER INDEX(DROP) for ROLE:**

Before ALTER INDEX

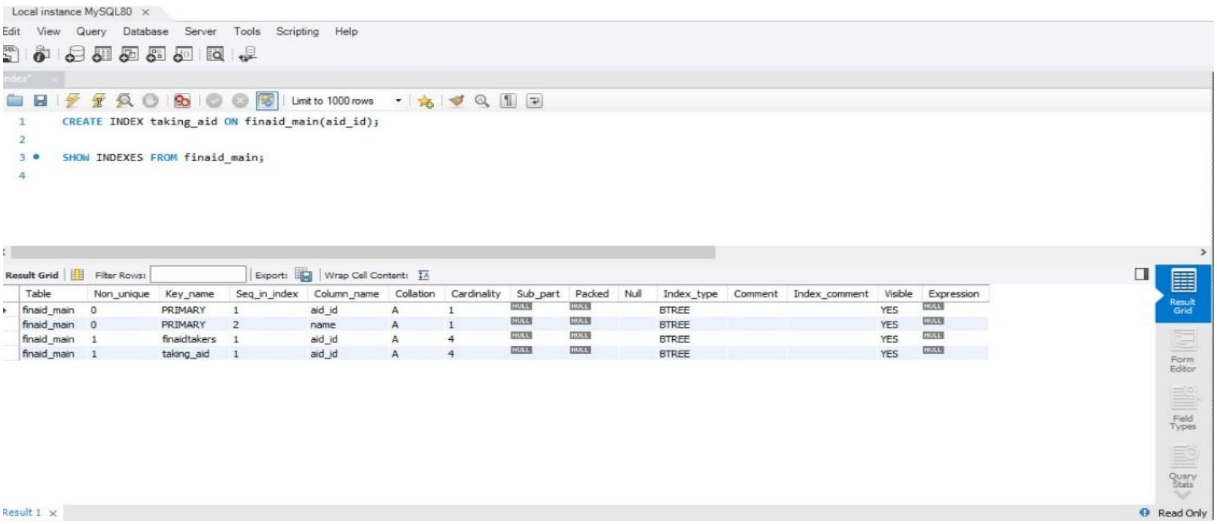


After ALTER INDEX

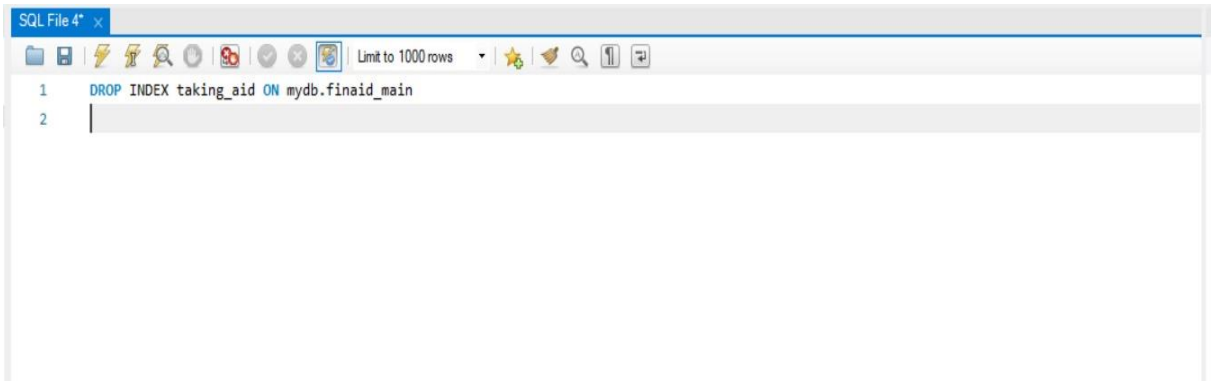


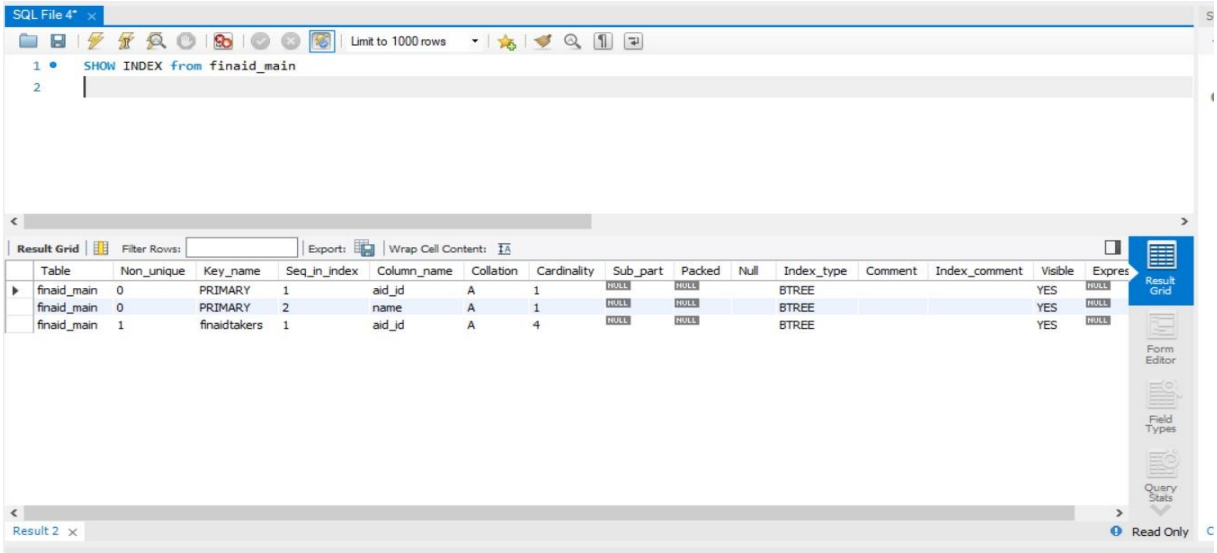
- **ALTER INDEX(DROP) for Finaid_main:**

Before ALTER INDEX



After ALTER INDEX

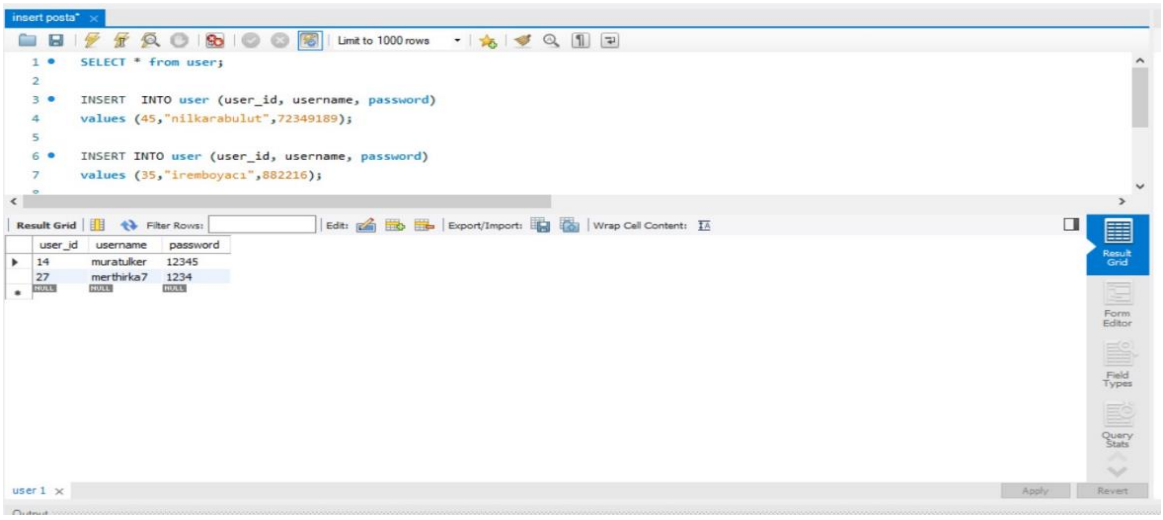


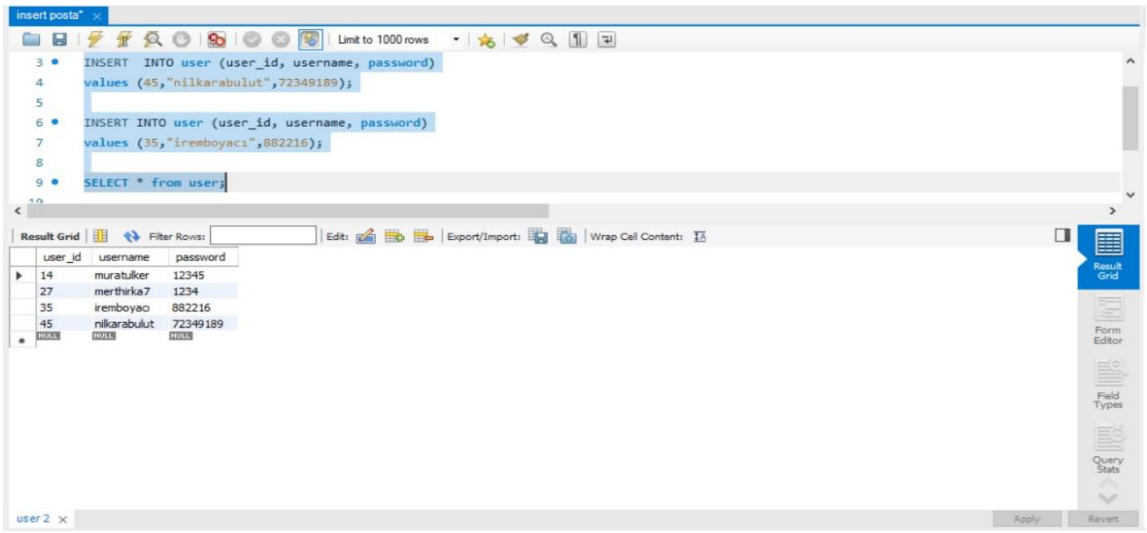


B.DML STATEMENTS -DATA MANIPULATION LANGUAGE

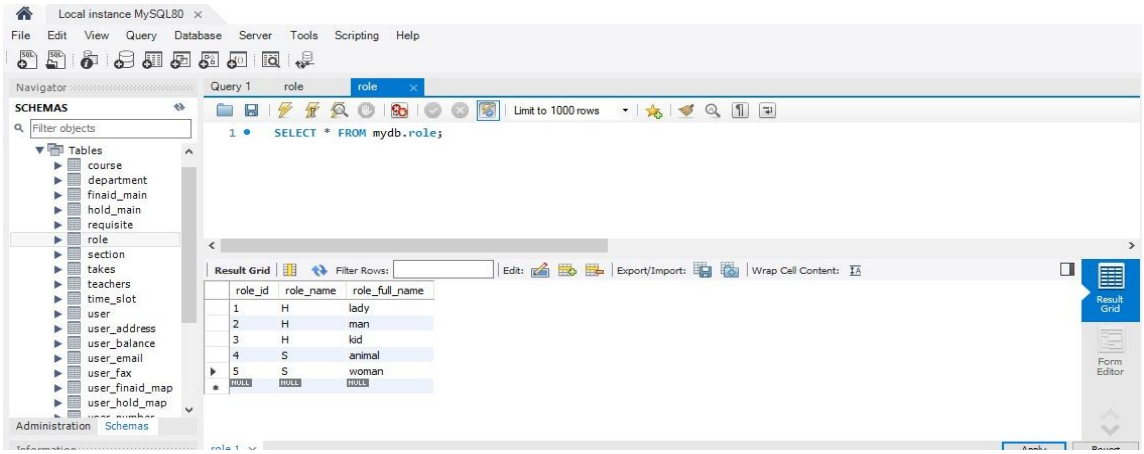
B.1. INSERT INTO STATEMENTS (OPEN ENDED)

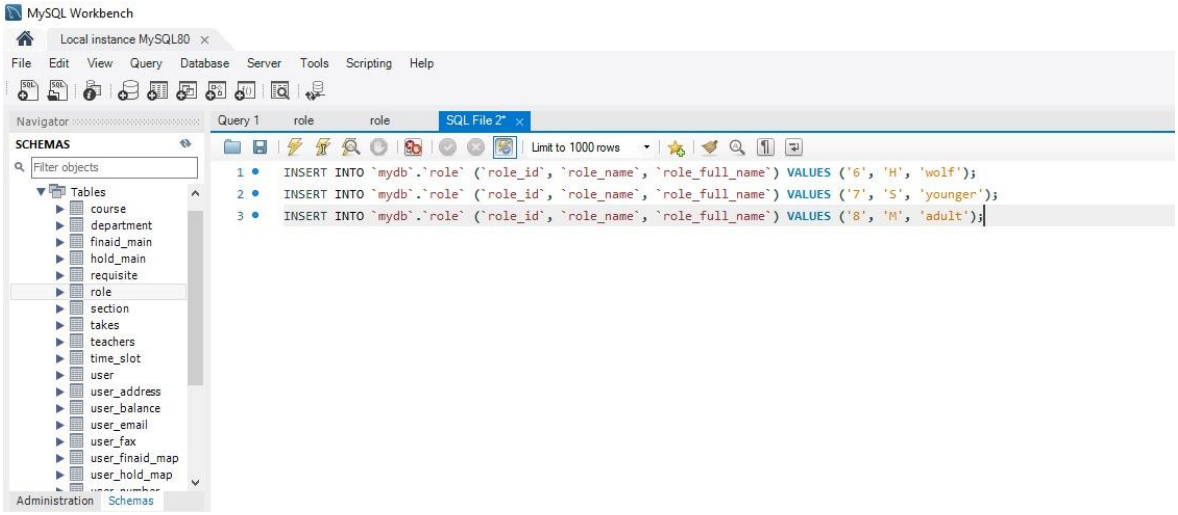
For USER:

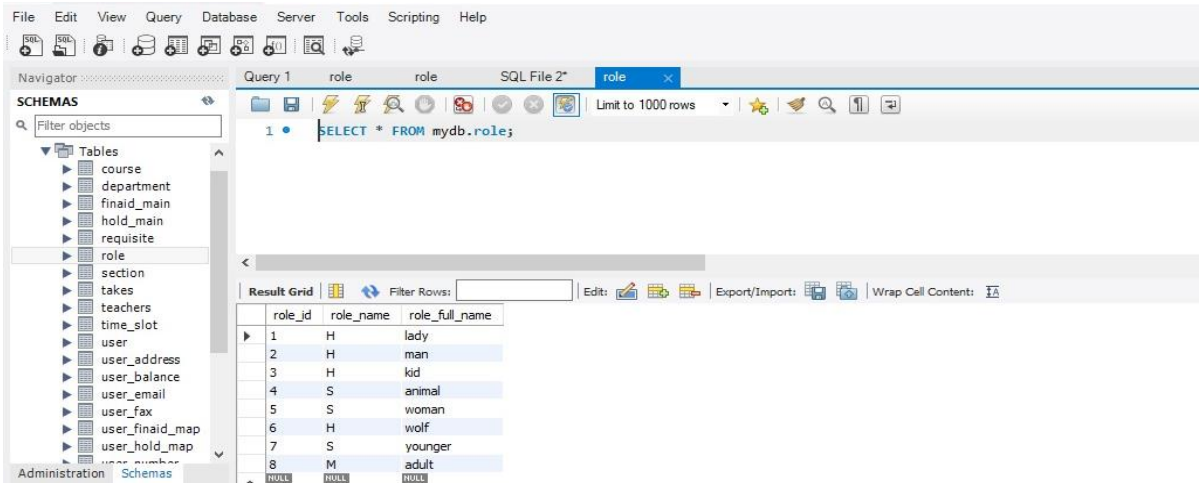




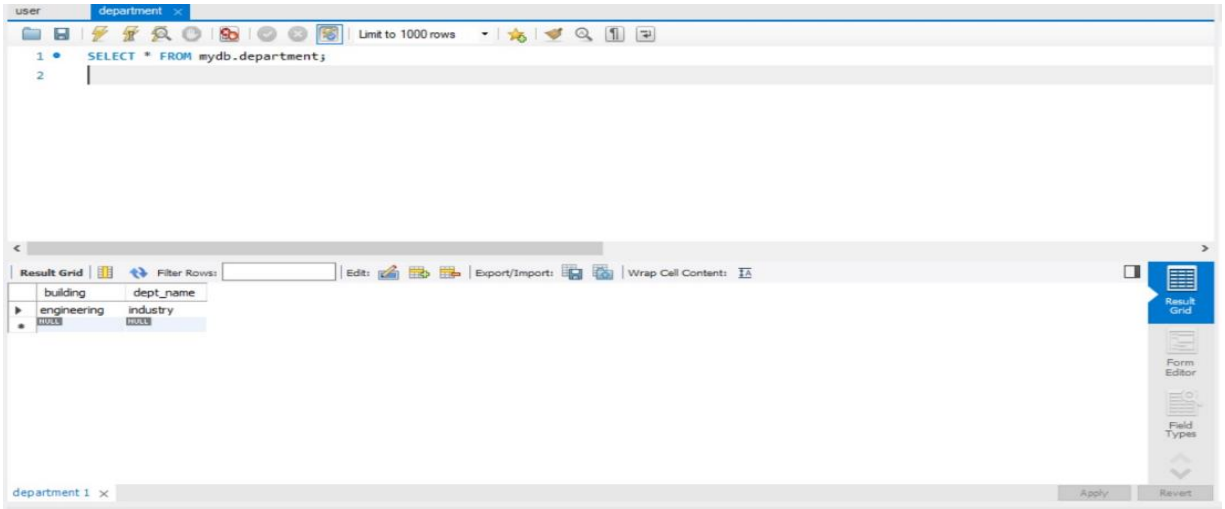
For ROLE :

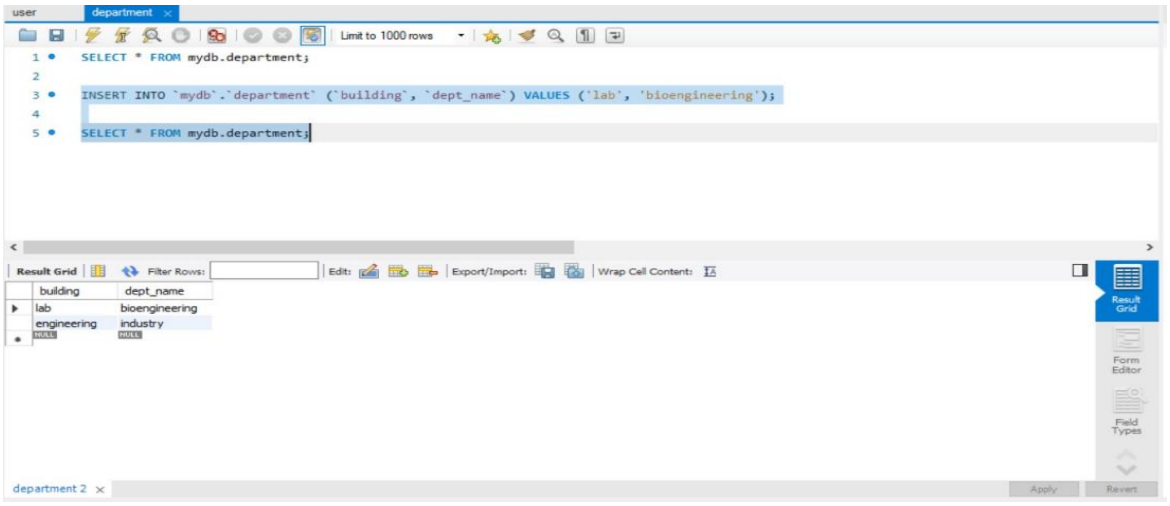




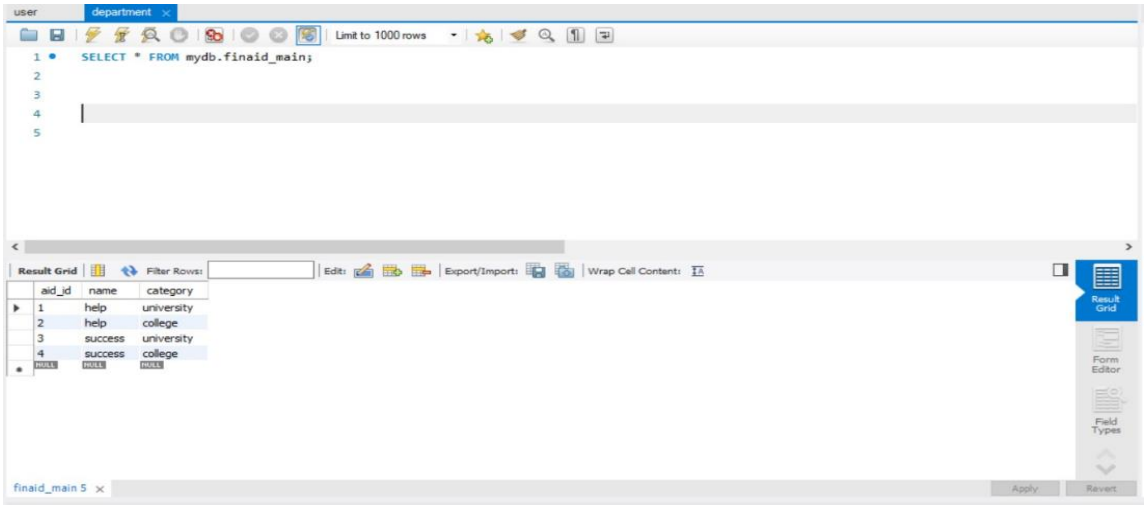


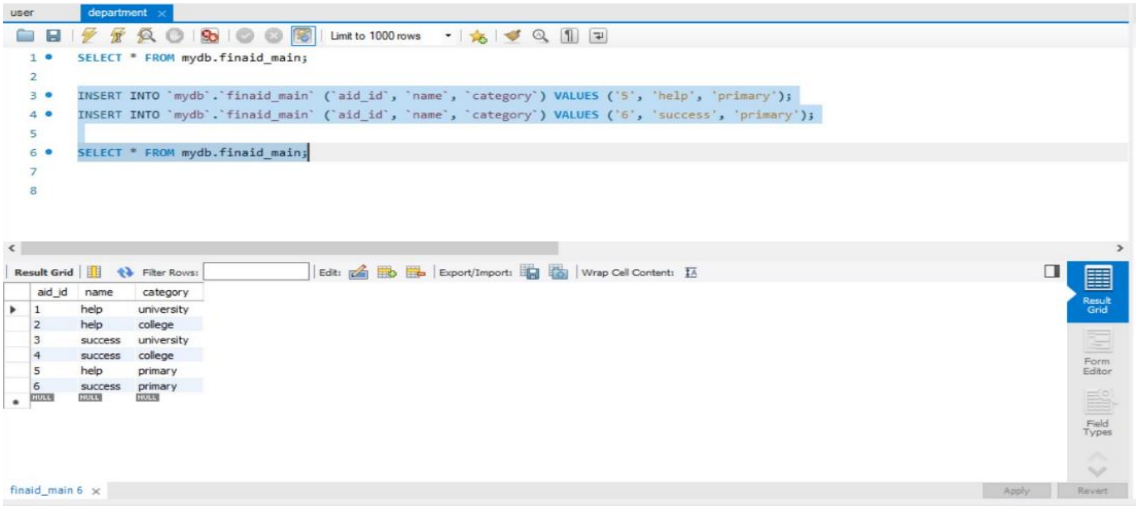
For DEPARTMENT:





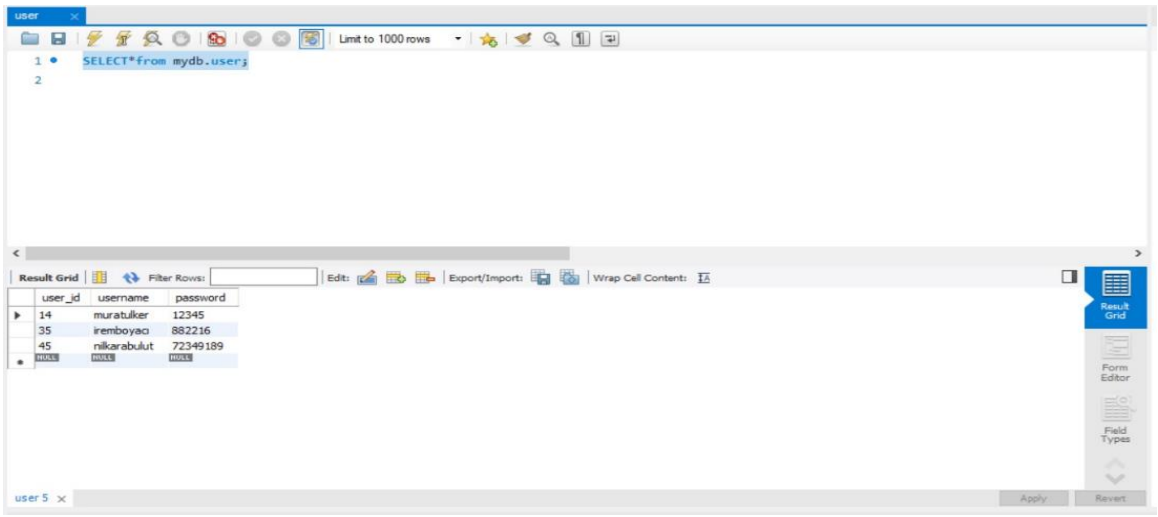
For FINAID_MAIN:

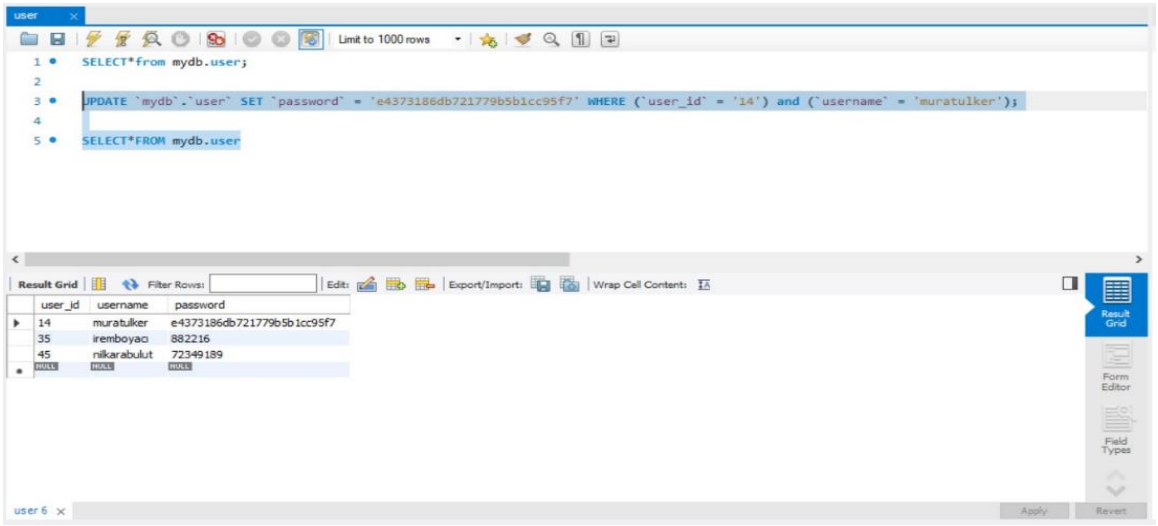




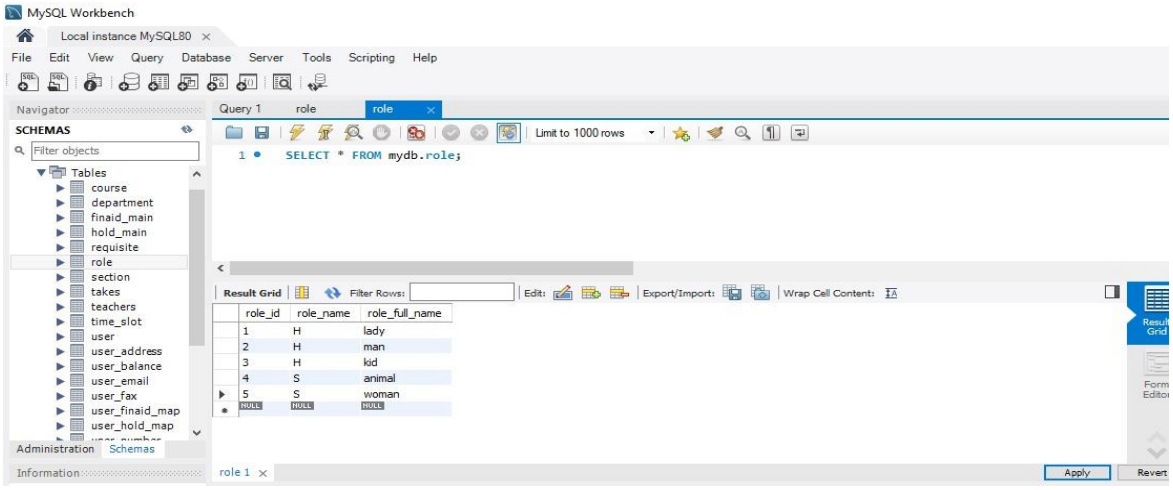
- **B.2. UPDATE STATEMENTS (OPEN ENDED)**

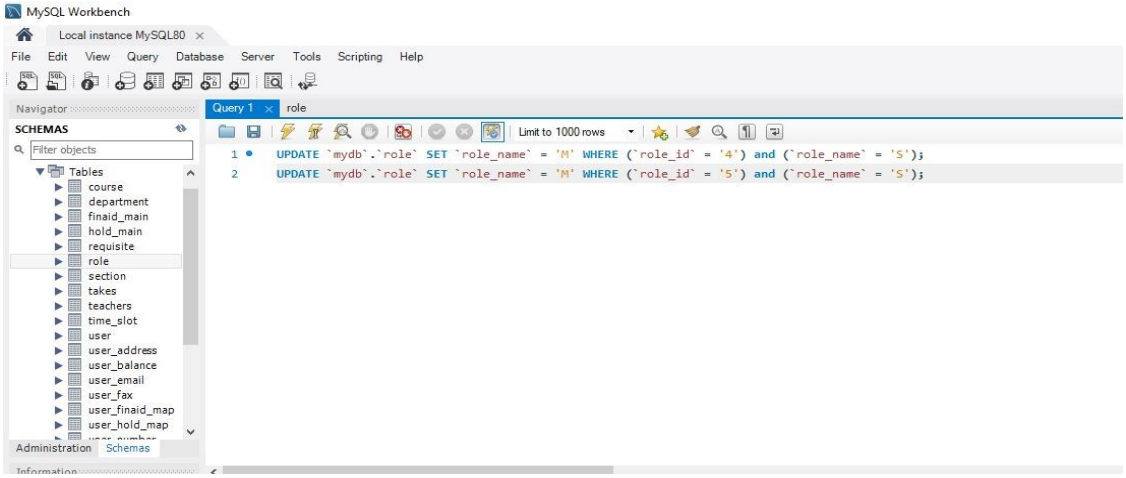
For USER:

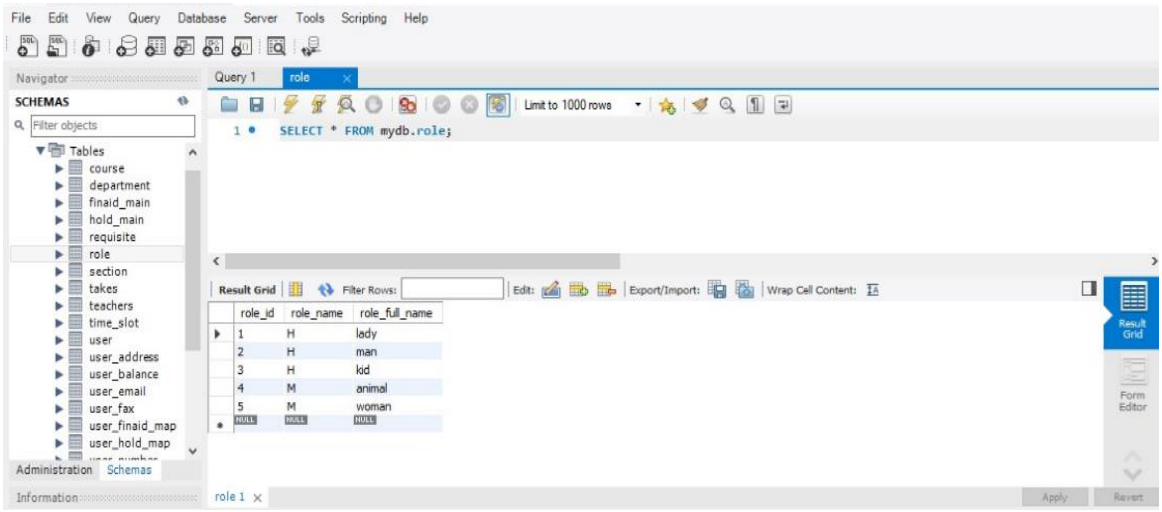




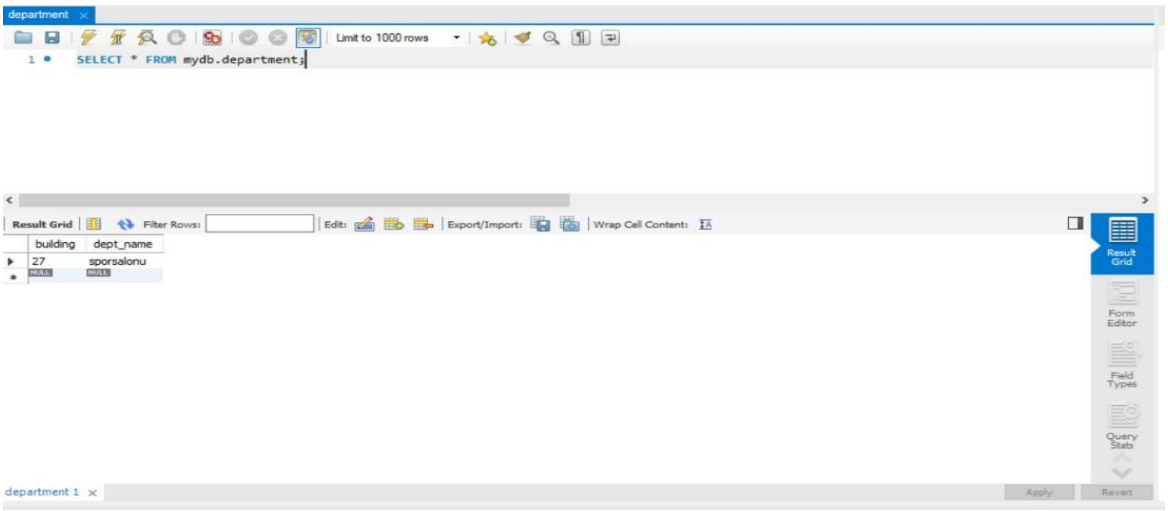
For ROLE :

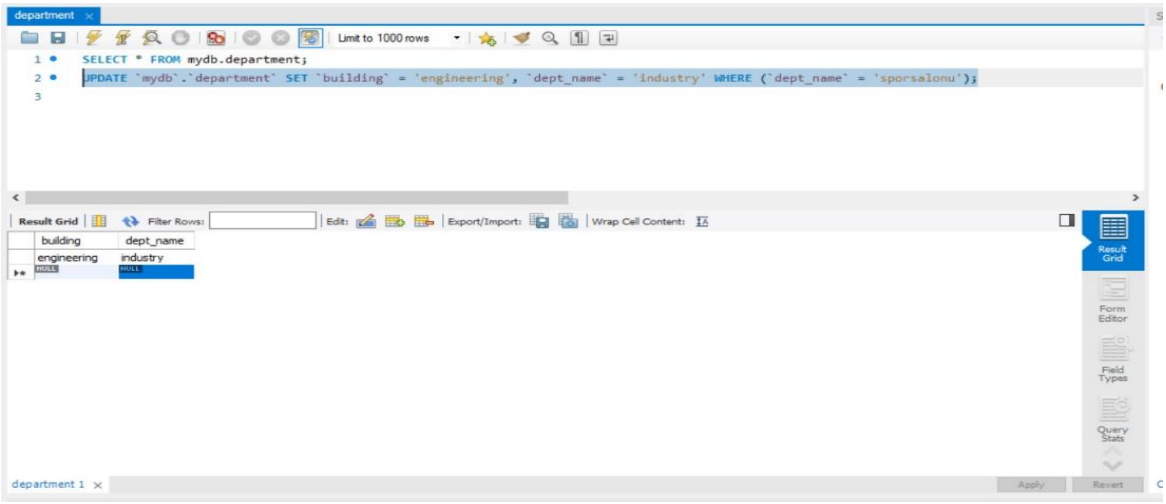




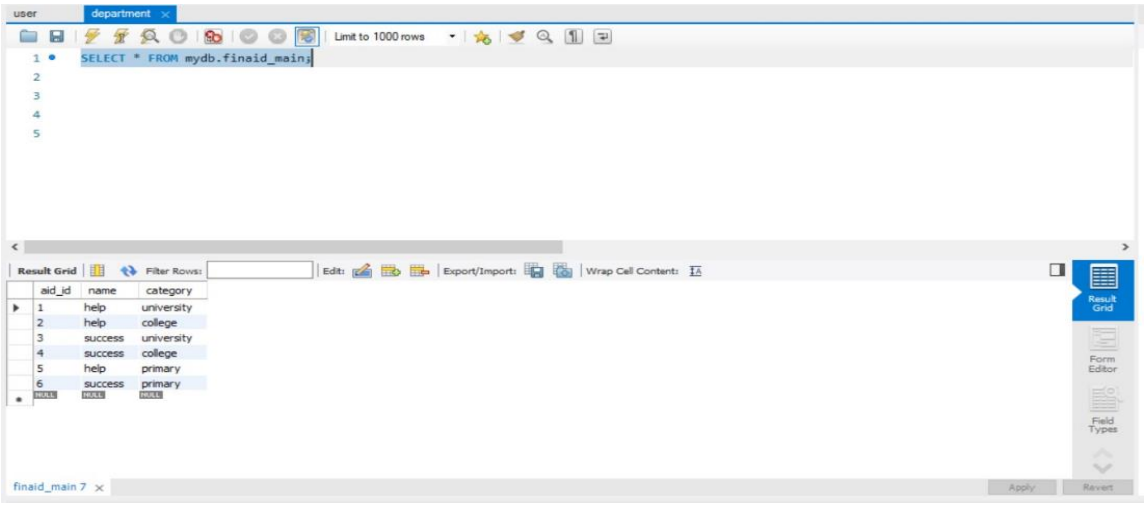


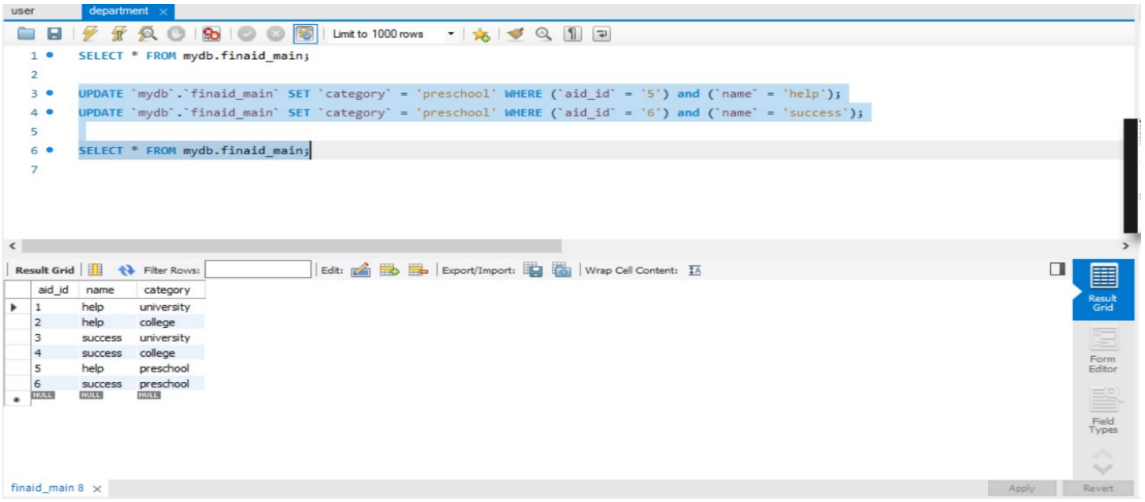
For DEPARTMENT:





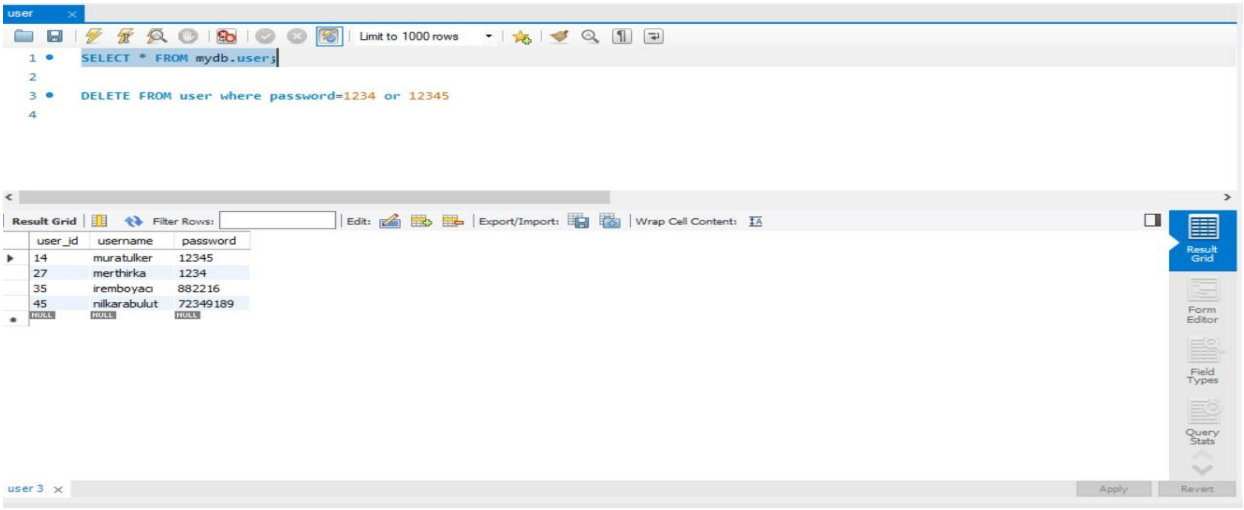
For FINAID_MAIN :

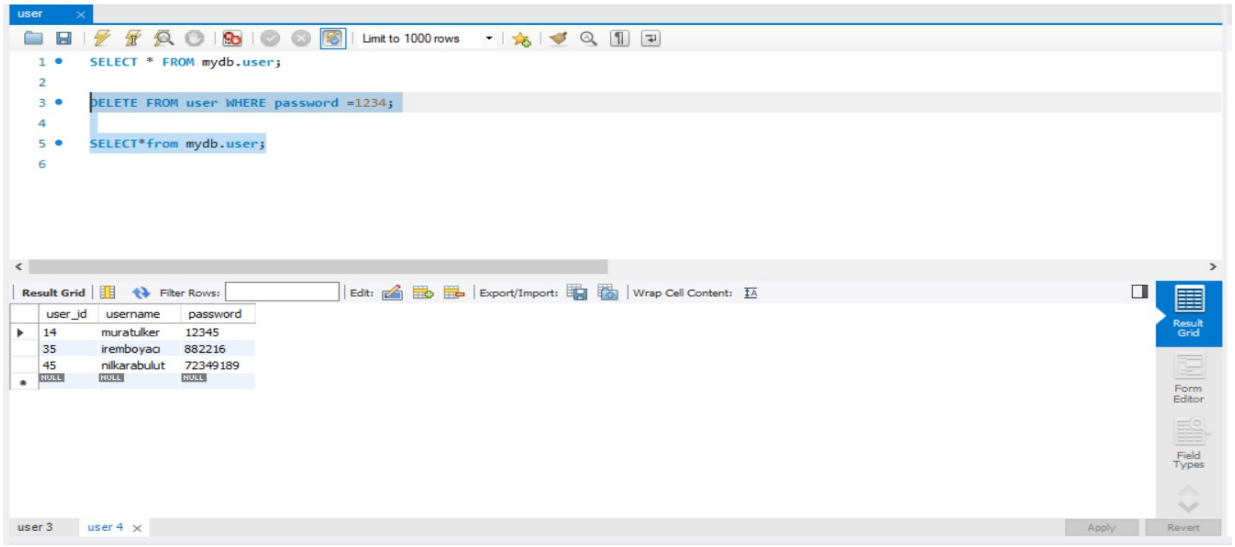




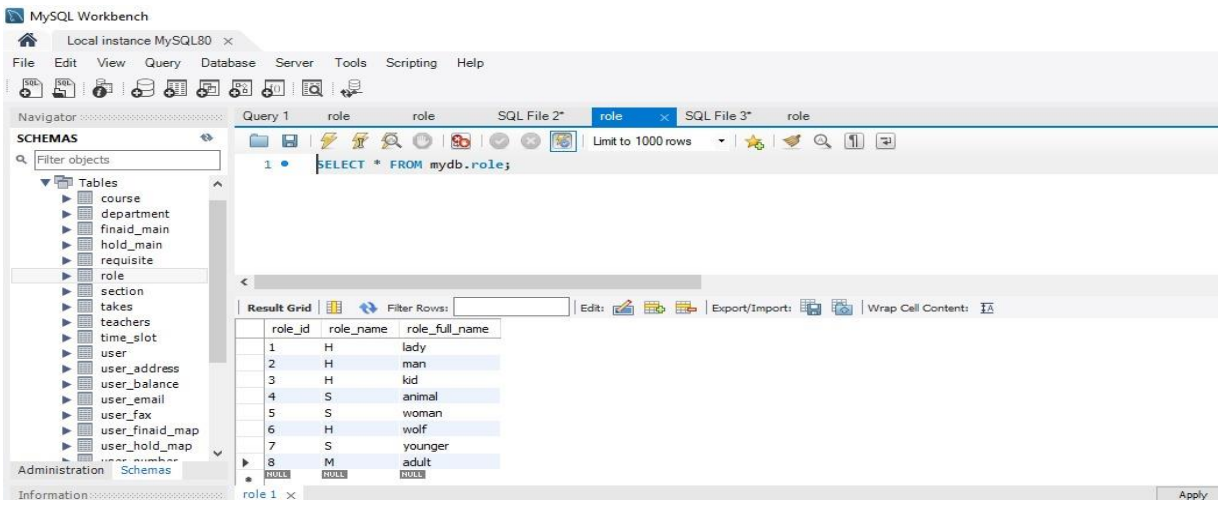
B.3. DELETE STATEMENTS (OPEN ENDED)

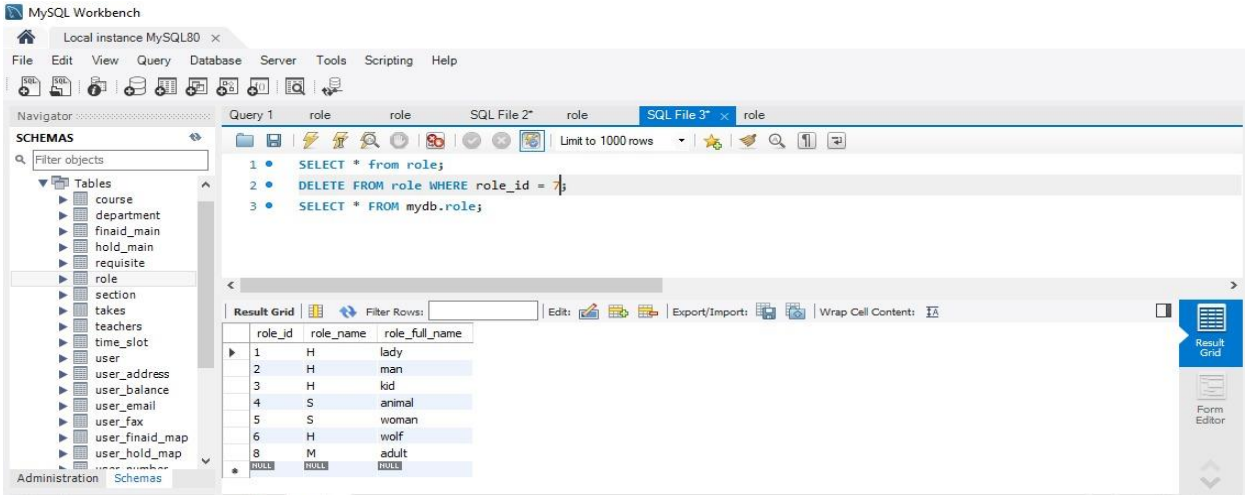
For USER:



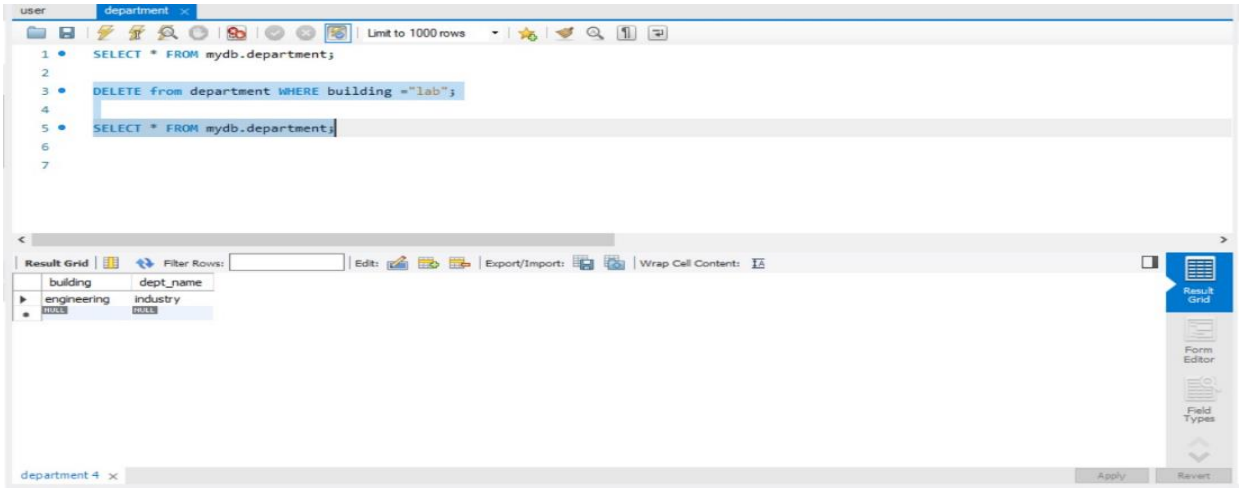


For ROLE:

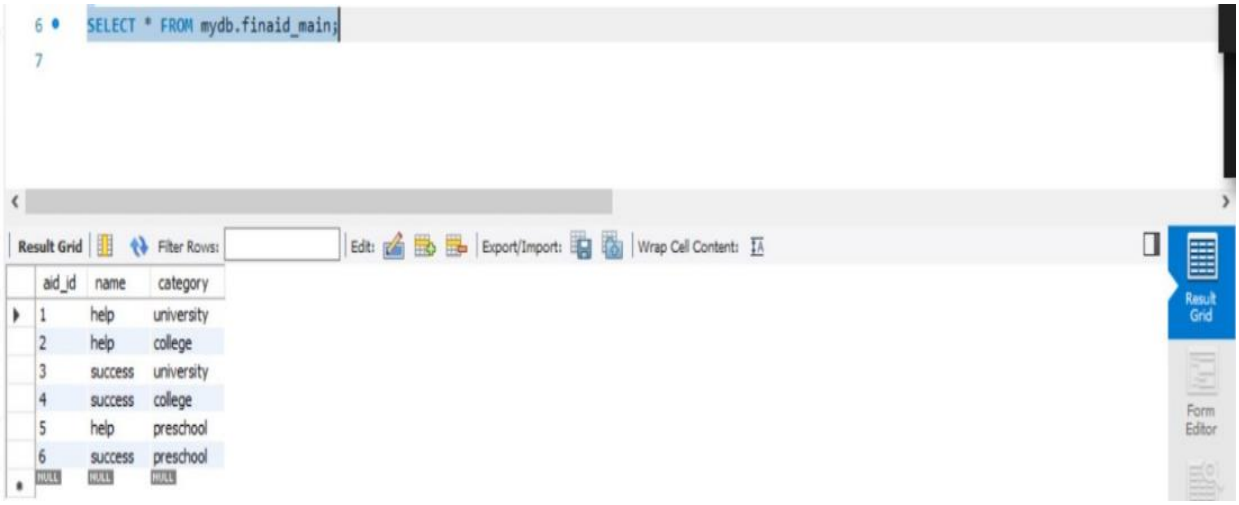


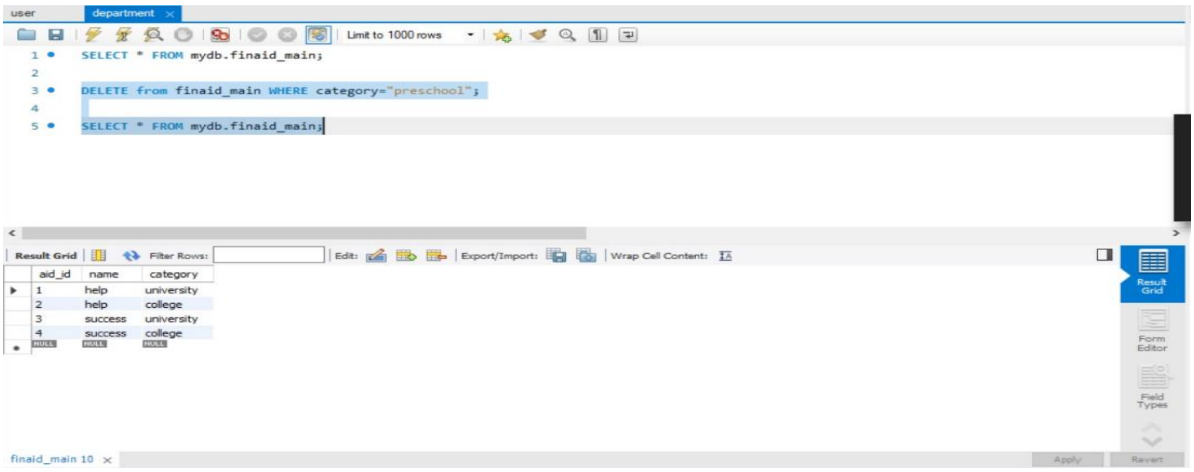


For DEPARTMENT:



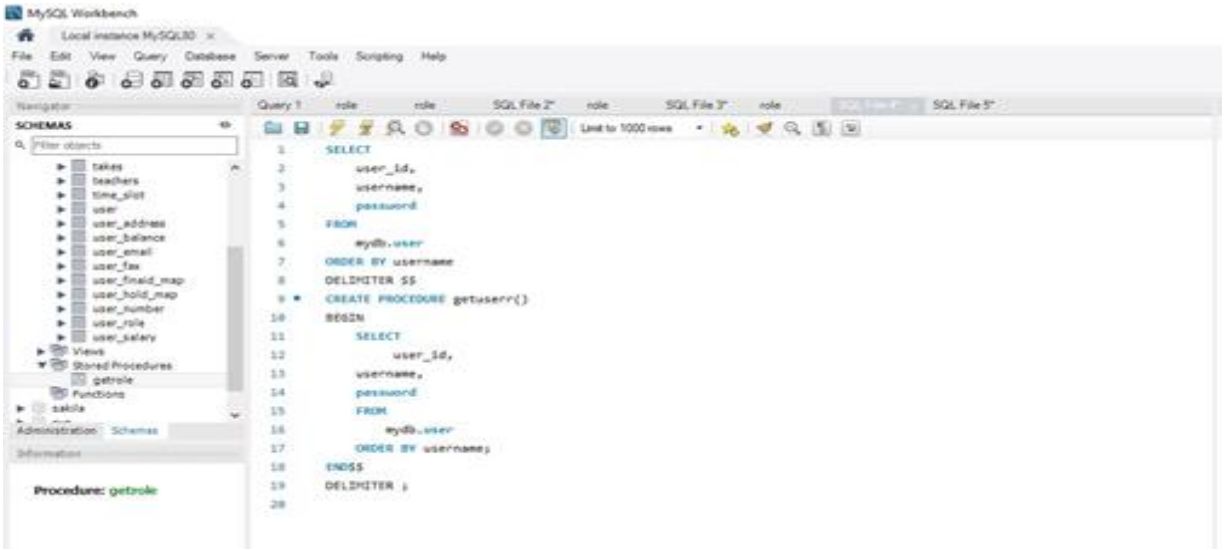
For FINAID_MAIN:

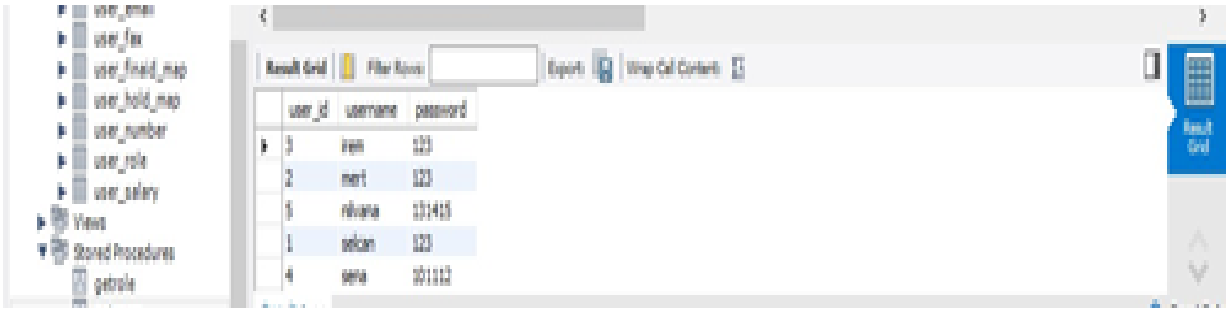




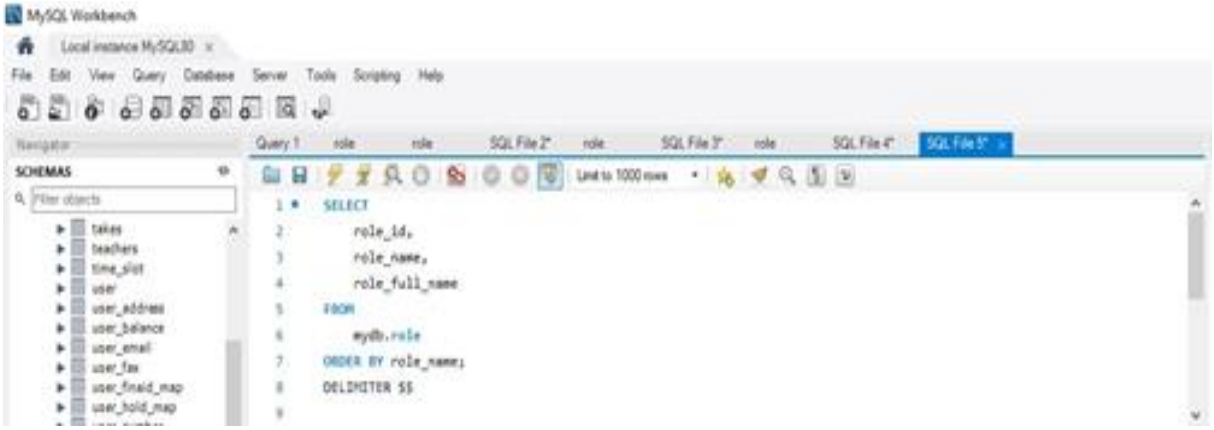
B.4. YOU SHOULD WRITE AT LEAST ONE STORED PROCEDURE OR FUNCTION (TRIGGER)

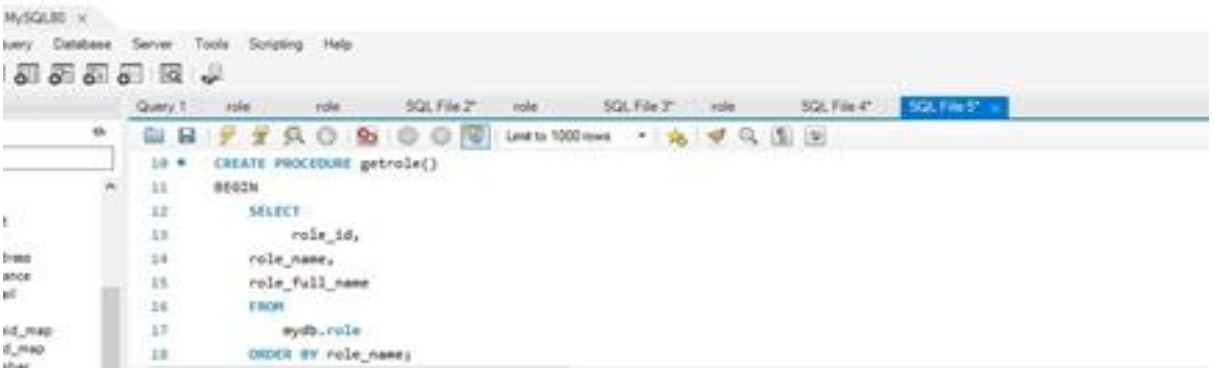
For USER:

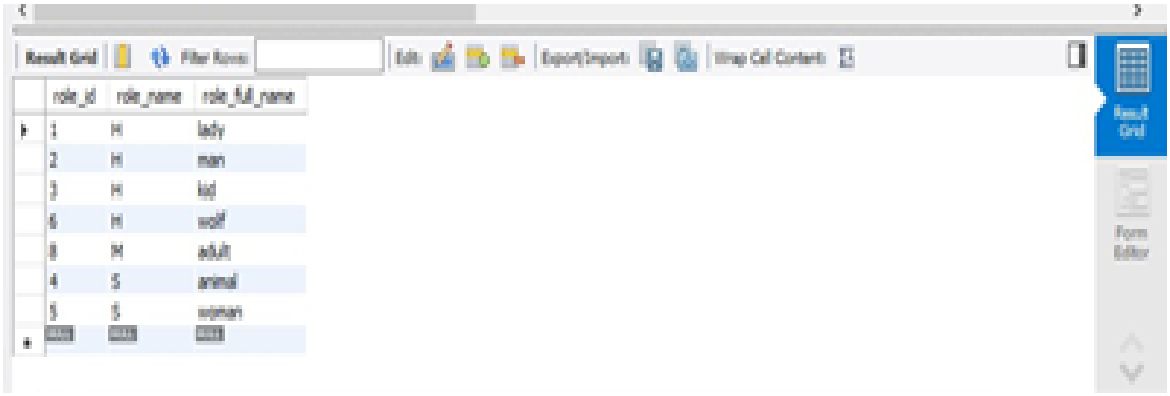




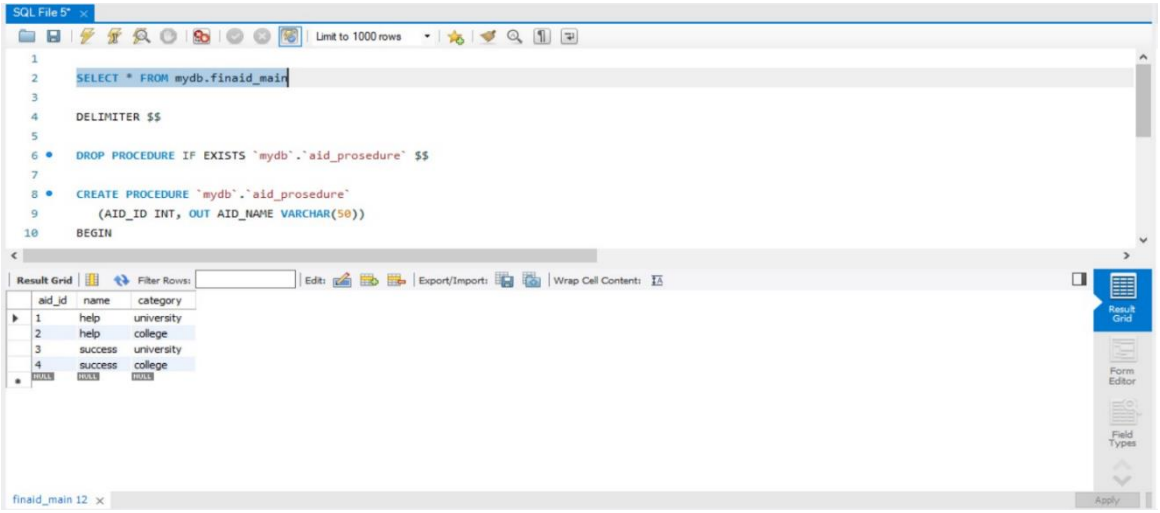
For ROLE:


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
SCHEMAS
takes
teachers
time_slot
user
user_address
user_balance
user_email
user_fax
user_fnaid_map
user_hold_map
Query 1 role role SQL File 2* role SQL File 3* role SQL File 4* SQL File 5*
Limit to 1000 rows
SELECT
role_id,
role_name,
role_full_name
FROM
mydb.role
ORDER BY role_name;
DELIMITER $$

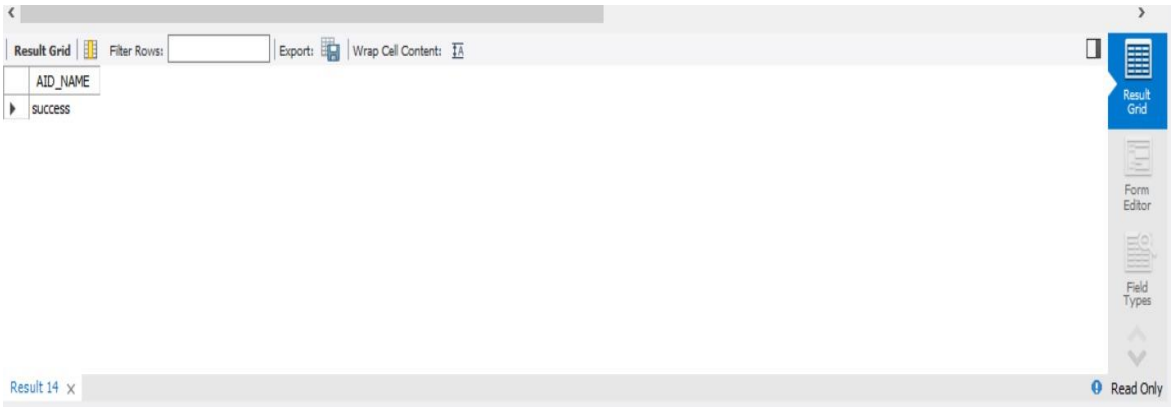

MySQL80
Database Server Tools Scripting Help
Query 1 role role SQL File 2* role SQL File 3* role SQL File 4* SQL File 5*
Limit to 1000 rows
CREATE PROCEDURE getrole()
BEGIN
SELECT
role_id,
role_name,
role_full_name
FROM
mydb.role
ORDER BY role_name;


Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cal Content:
role_id role_name role_full_name
1 M lady
2 M man
3 M kid
6 M wolf
8 M adult
4 S animal
5 S woman
Result Grid
Form Editor

For FINAID_MAIN:



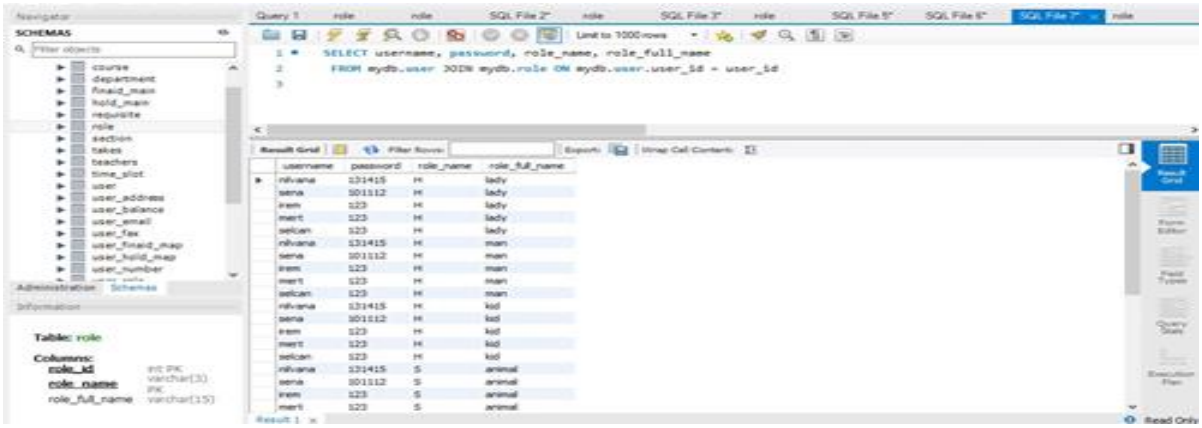

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `mydb`.`aid_prosedure` $$

CREATE PROCEDURE `mydb`.`aid_prosedure`
   (AID_ID INT, OUT AID_NAME VARCHAR(50))
BEGIN
   SELECT  finaid_main.name AID_NAME
   FROM finaid_main
   WHERE finaid_main.aid_id =AID_ID;
END $$

DELIMITER ;

SET @AID_ID=3;
CALL aid_prosedure(@AID_ID,@AID_NAME);

SELECT AID_NAME;
```



C. DQL STATEMENTS -DATA QUERY LANGUAGE

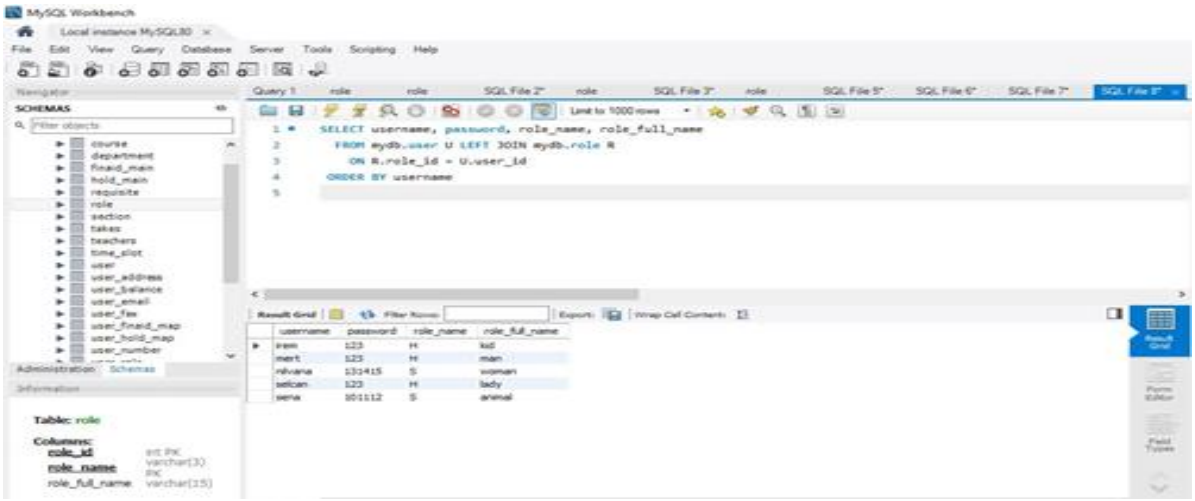
C.1. SIMPLE JOINS - JOINING TWO TABLES

- User and role are joined.

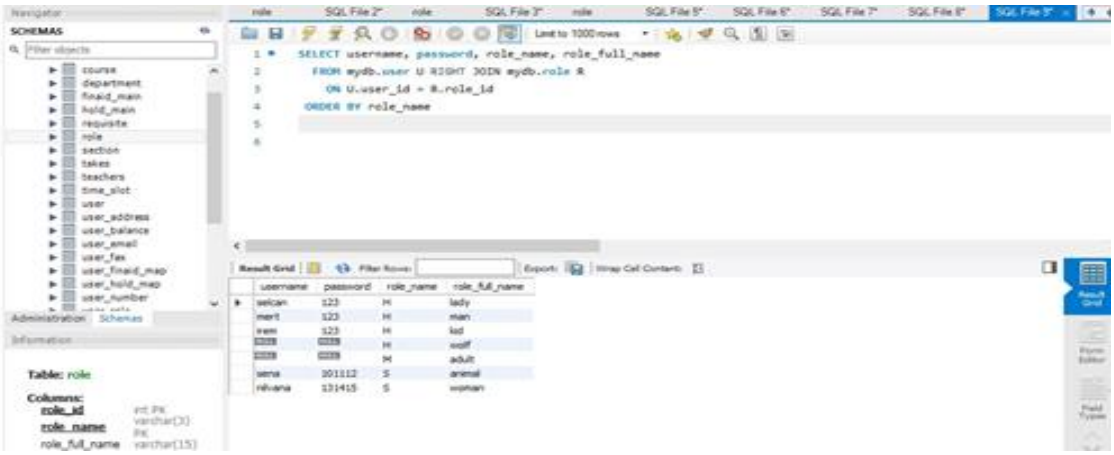


C.2. LEFT/RIGHT INNER/OUTER JOINS

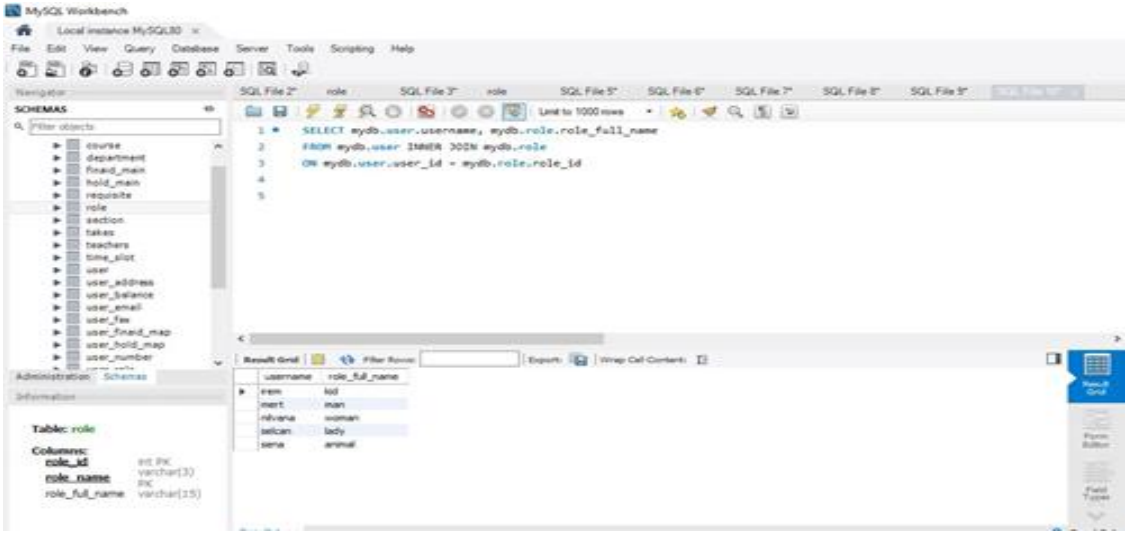
- **LEFT JOINS:**



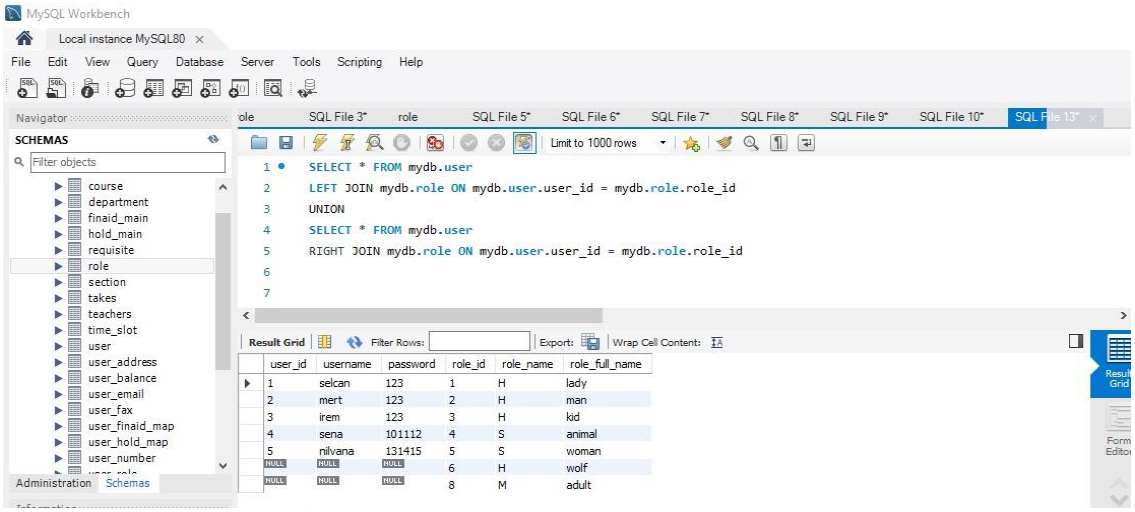
- **RIGHT JOINS:**



- **INNER JOINS:**

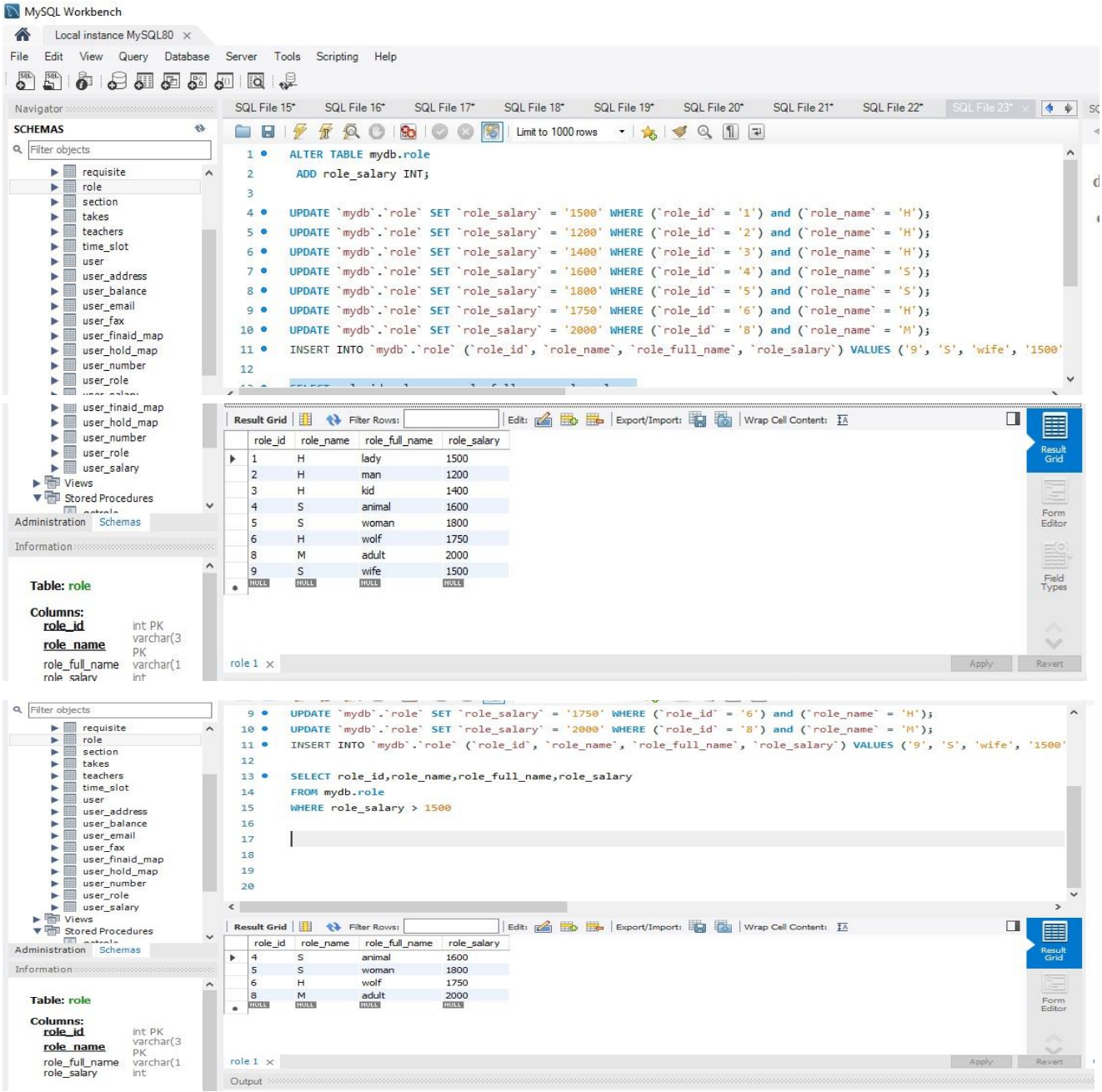


- **OUTER JOINS:**



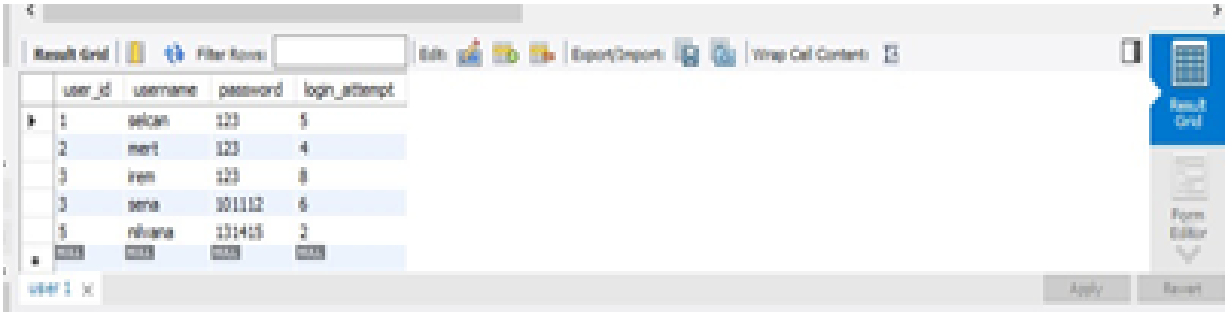
C.3. NESTED QUERIES

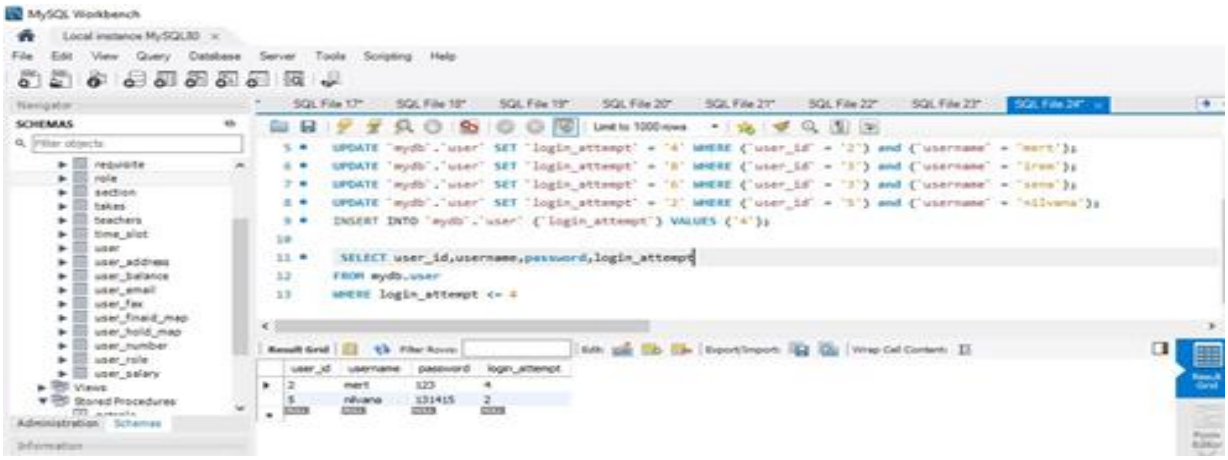
For ROLE:



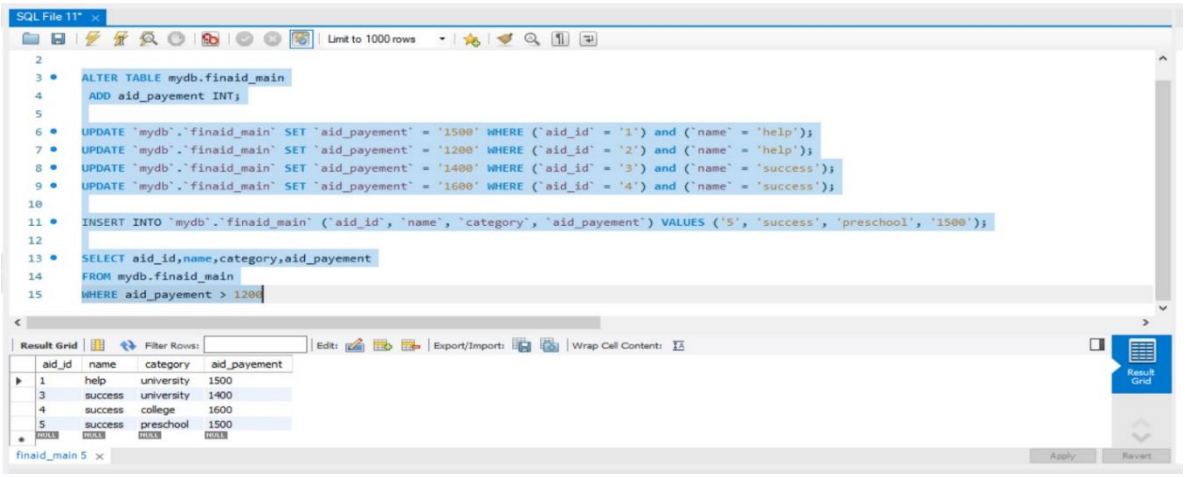
For USER :


MySQL Workbench
ALTER TABLE mydb.user
ADD login_atteept INT;
UPDATE 'mydb'.'user' SET 'login_attempt' = '5' WHERE ('user_id' = '1') and ('username' = 'selcan');
UPDATE 'mydb'.'user' SET 'login_attempt' = '4' WHERE ('user_id' = '2') and ('username' = 'mert');
UPDATE 'mydb'.'user' SET 'login_attempt' = '8' WHERE ('user_id' = '3') and ('username' = 'irem');
UPDATE 'mydb'.'user' SET 'login_attempt' = '6' WHERE ('user_id' = '4') and ('username' = 'sena');
UPDATE 'mydb'.'user' SET 'login_attempt' = '2' WHERE ('user_id' = '5') and ('username' = 'nilvana');
INSERT INTO 'mydb'.'user' ('login_attempt') VALUES ('4');


user_id username password login_attempt
1 selcan 123 5
2 mert 123 4
3 irem 123 8
4 sena 101112 6
5 nilvana 131415 2

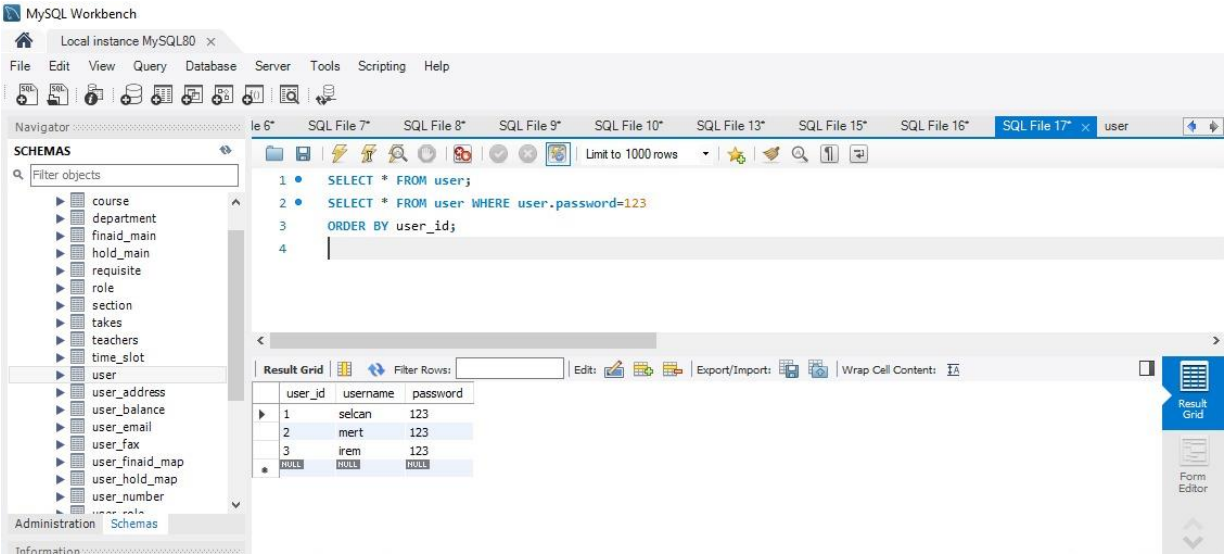

MySQL Workbench
SELECT user_id,username,password,login_attempt
FROM mydb.user
WHERE login_attempt <= 4

For FINAID_MAIN:

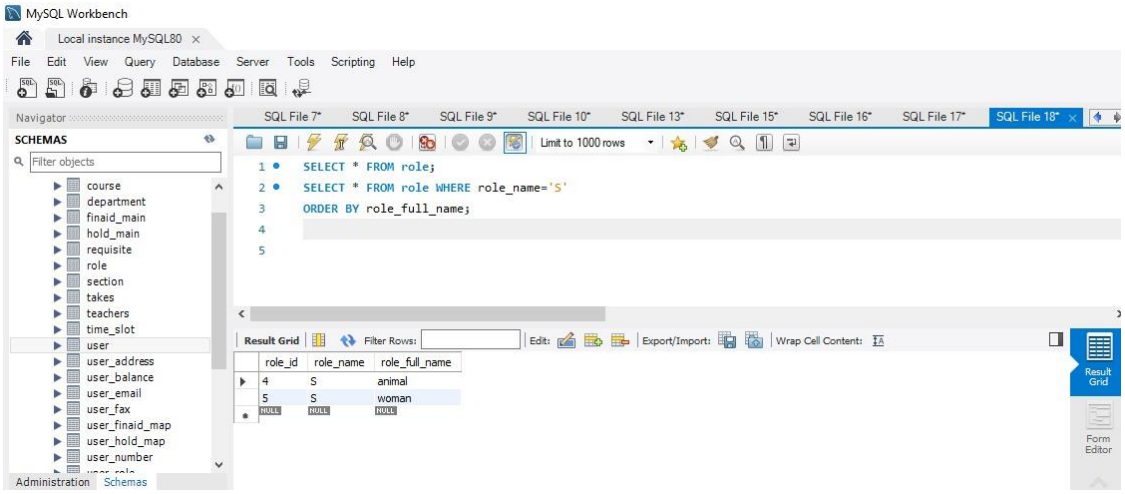


C.4. SORTING SUCH AS ORDER BY STATEMENTS:

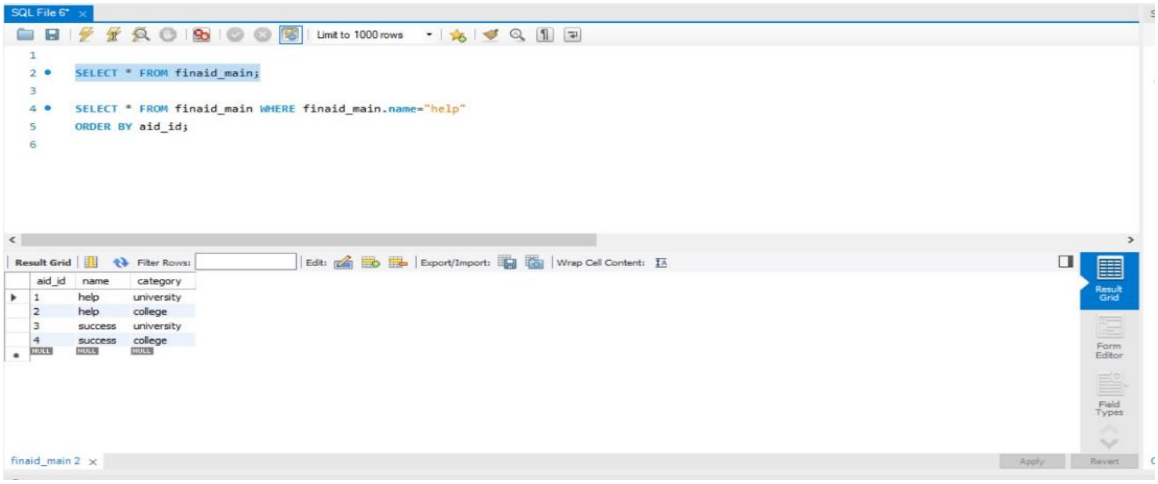
For USER:

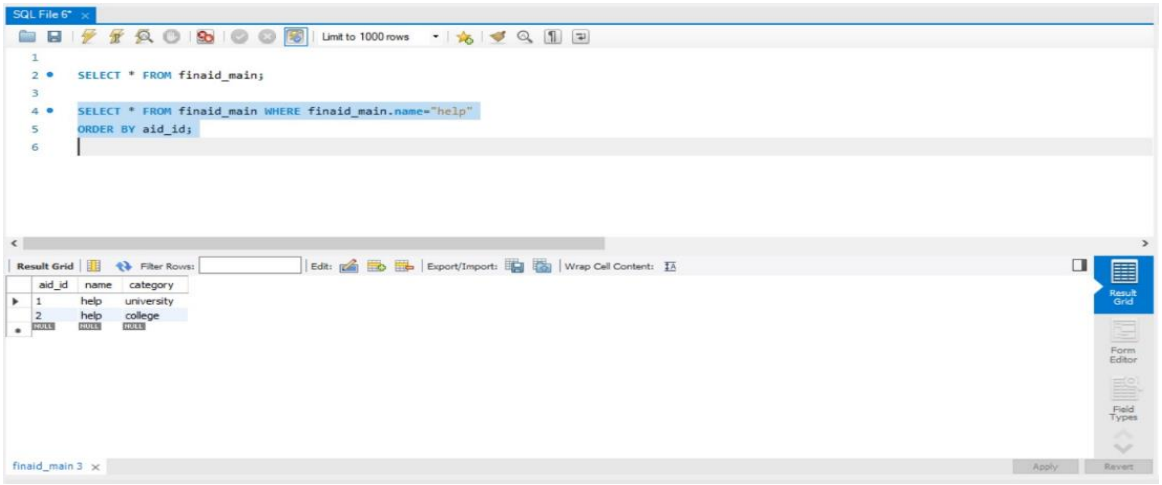


For ROLE:



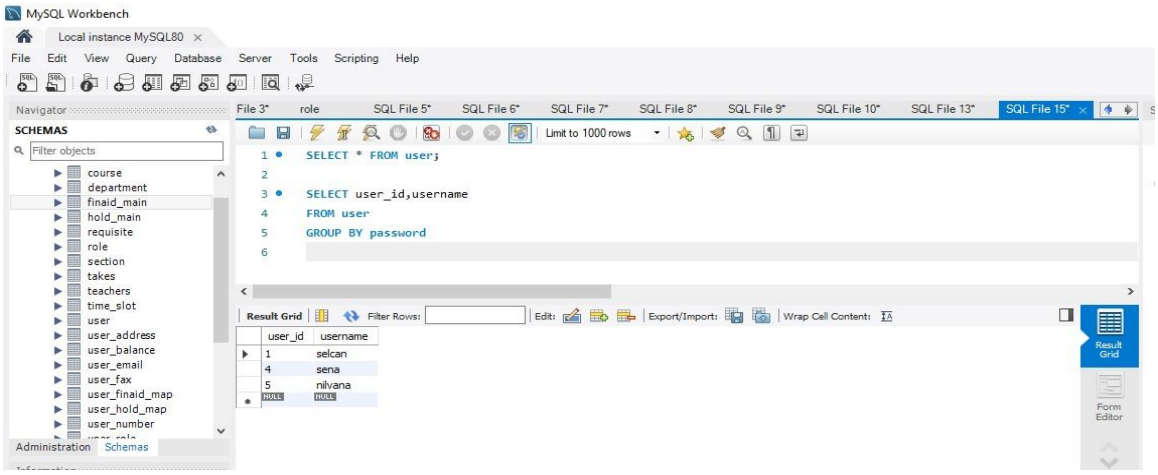
For FINAID_MAIN:



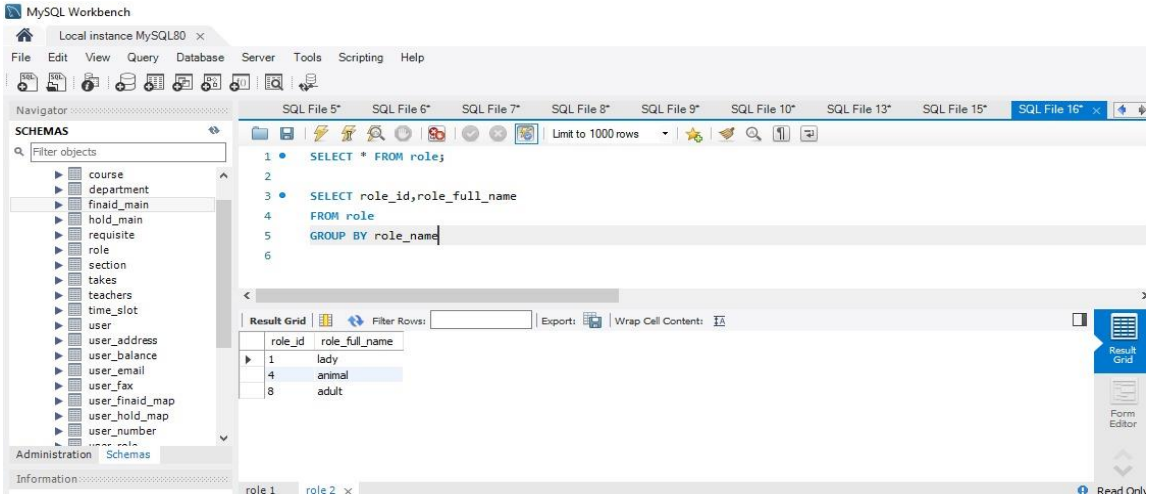


C.5. GROUP BY AND AGGREGATION FUNCTIONS

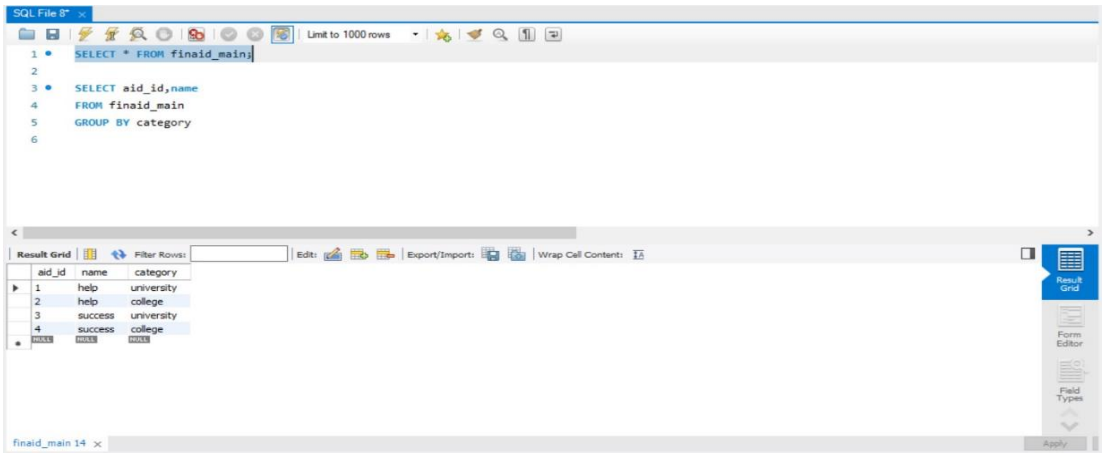
For USER:

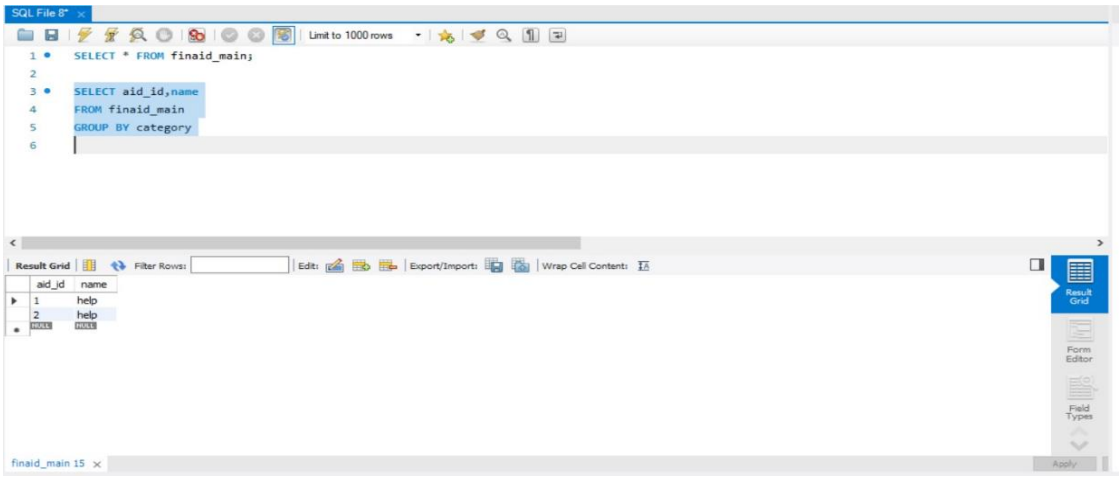


For ROLE:



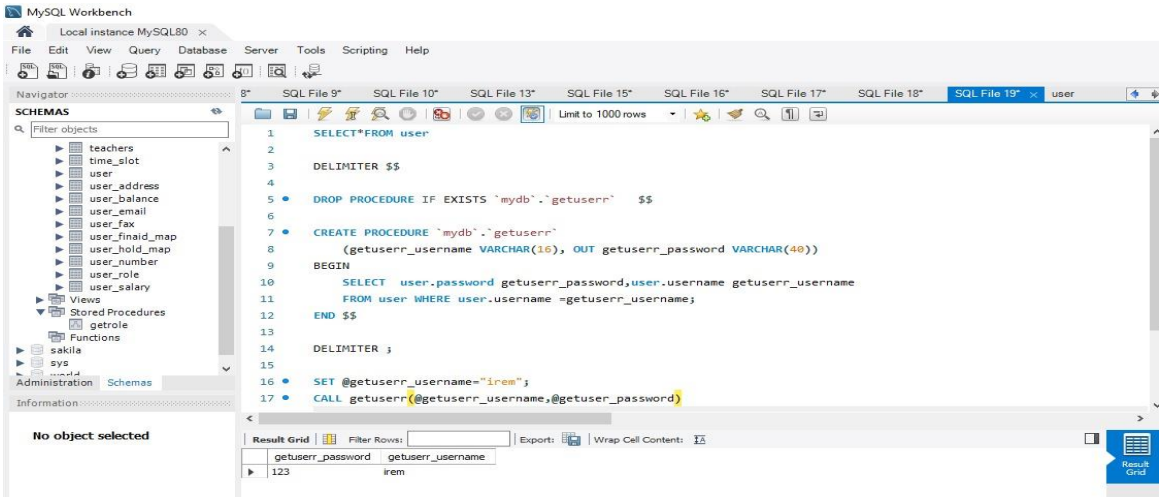
For FINAID_MAIN:



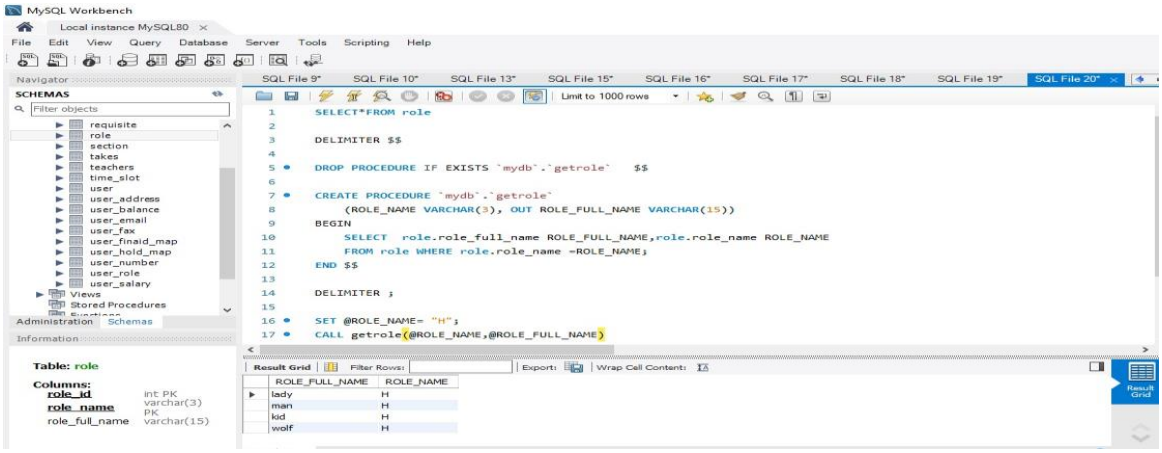


C.6. USING SELECT STATEMENTS WITH STORED PROCEDURE/FUNCTIONS

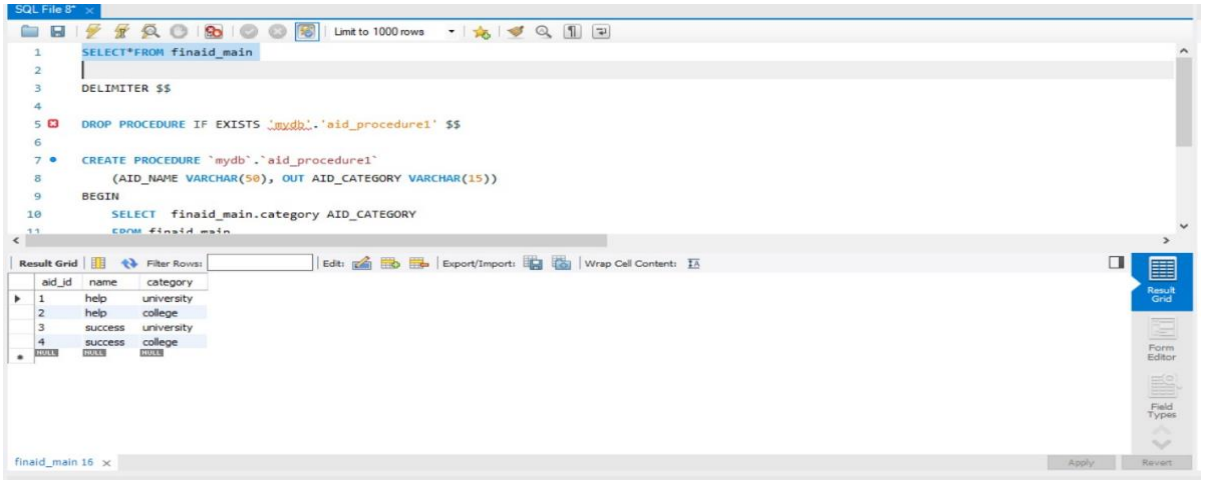
For USER:

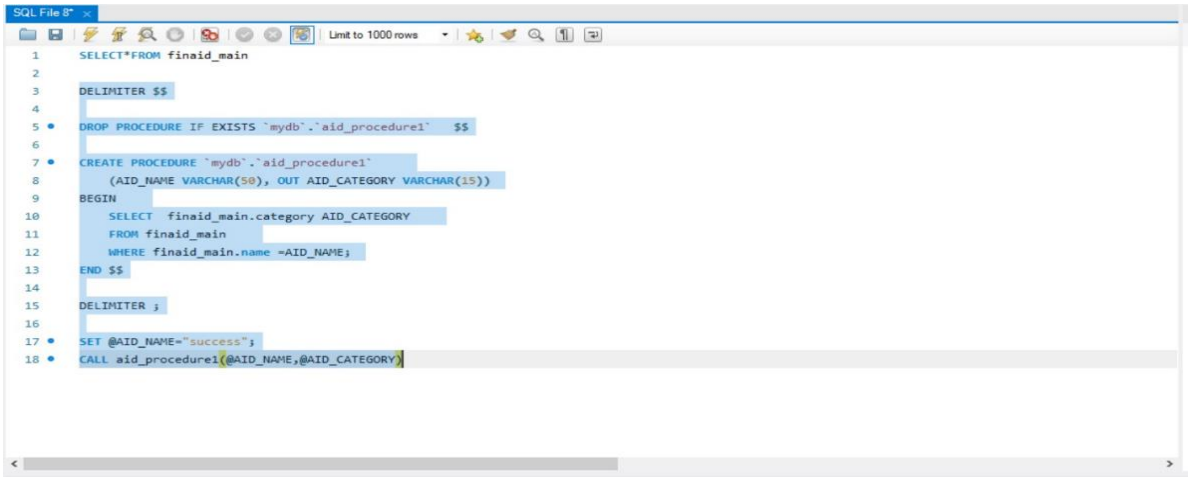


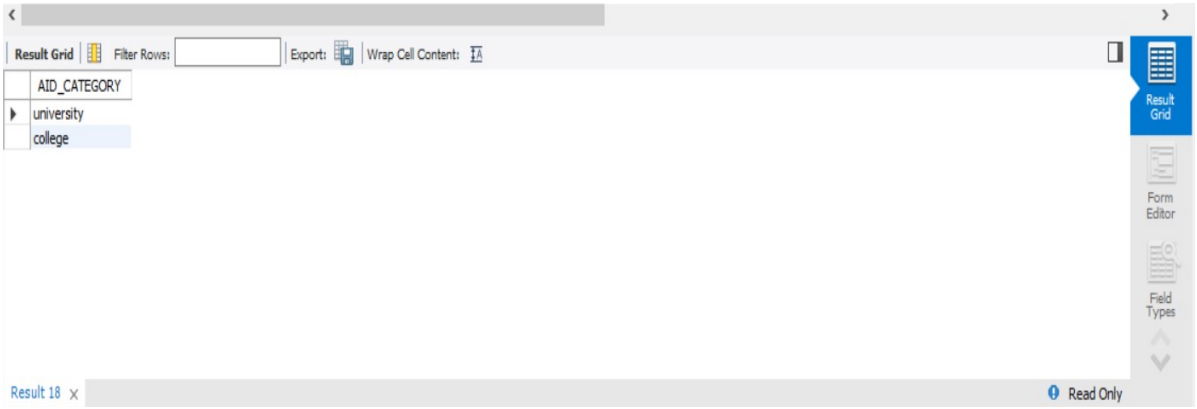
For ROLE:



For FINAID_MAIN:



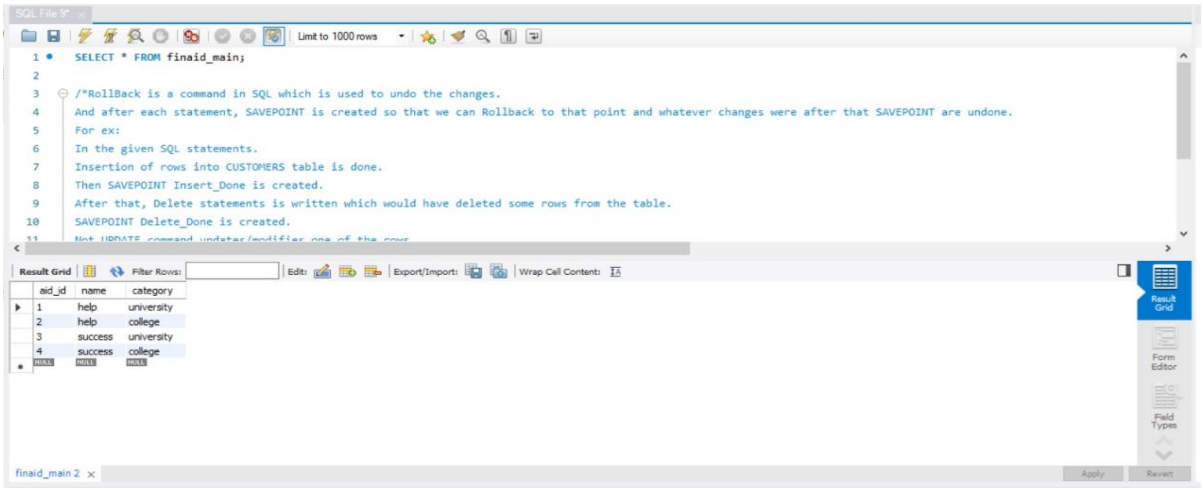


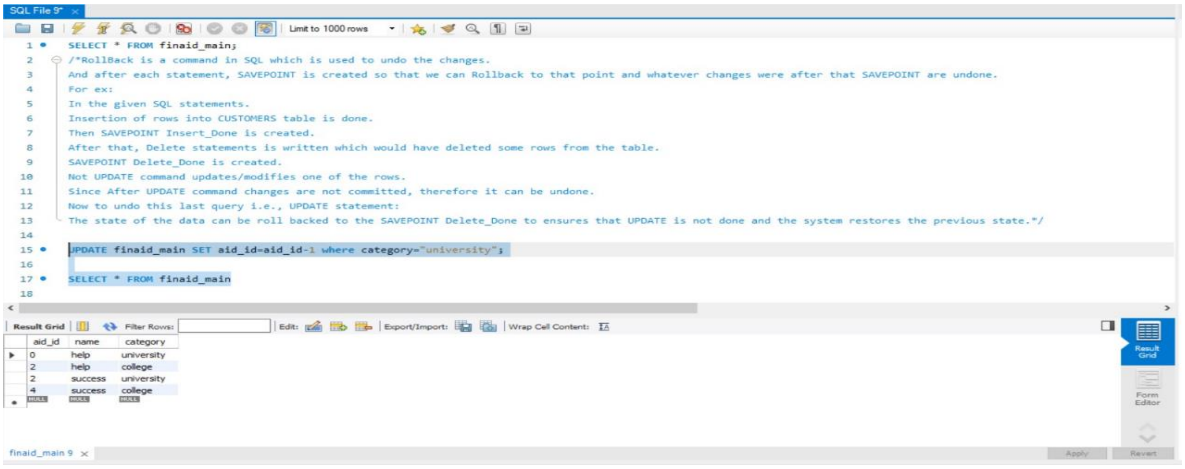


D. DCL STATEMENTS -DATA CONTROL LANGUAGE

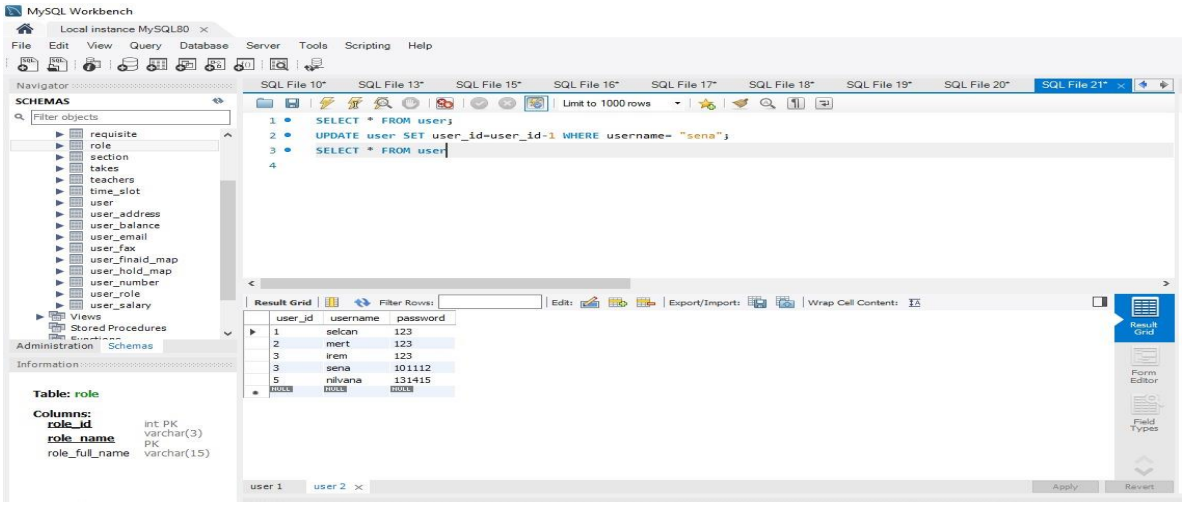
D.1. TRANSACTION PROCESSING (ROLLBACK/COMMIT ETC) (ORACLE SAVEPOINT)

For FINAID_MAIN:

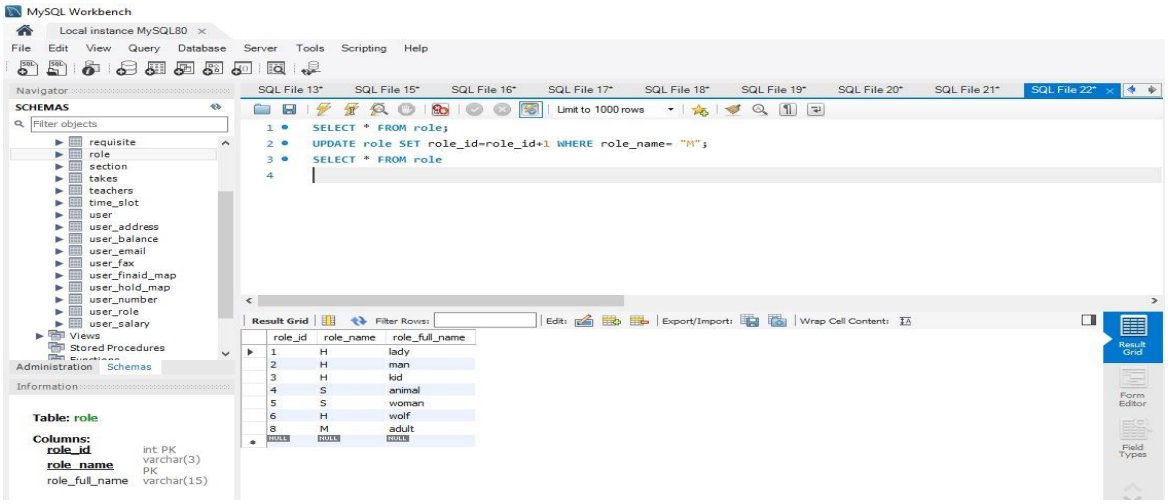




For USER:

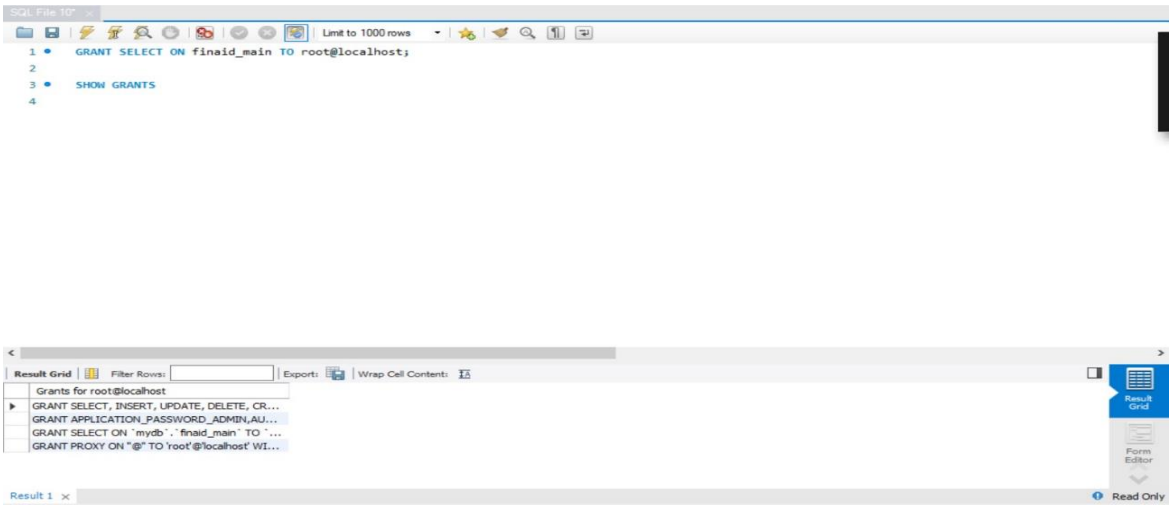


For ROLE:

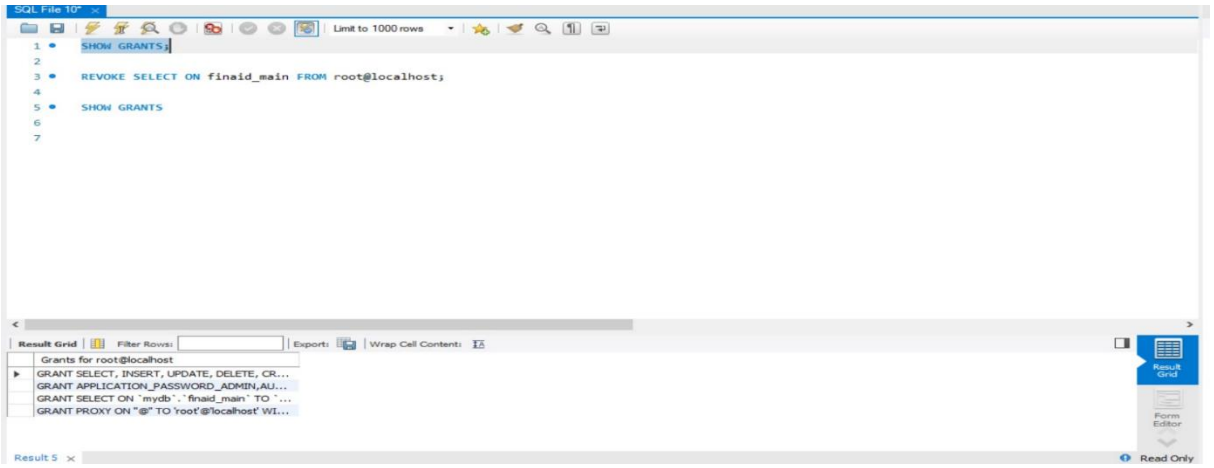


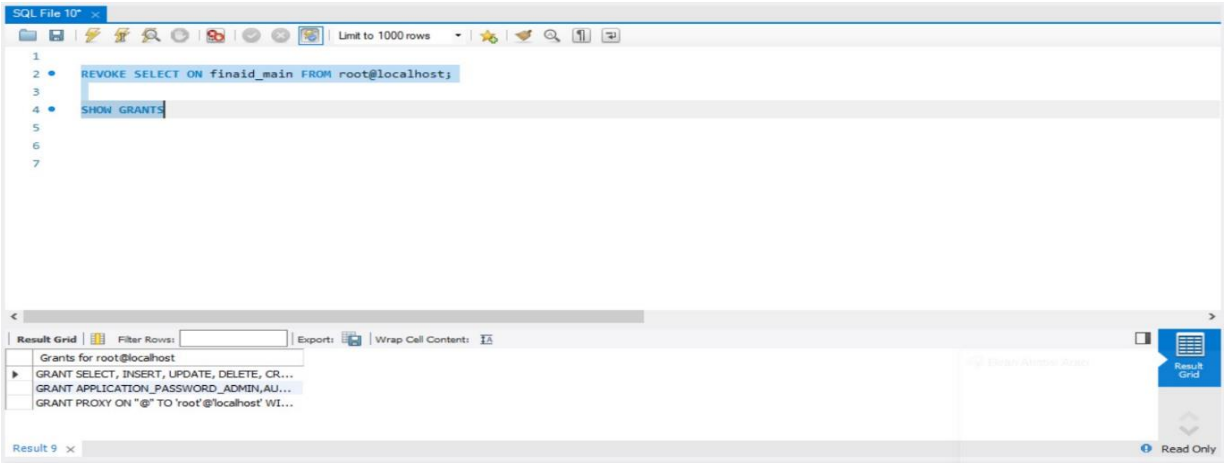
D.2. SECURITY FUNCTIONS (GRANT/REVOKE)

- **GRANT**



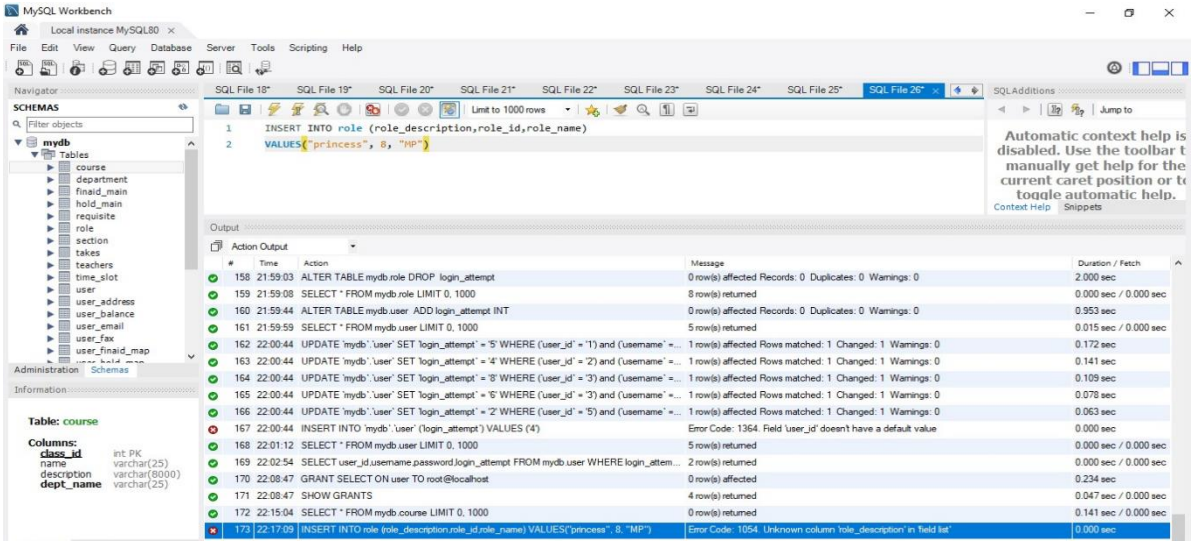
- **REVOKE**


SQL File 10*
Limit to 1000 rows
SHOW GRANTS;
REVOKE SELECT ON finaid_main FROM root@localhost;
SHOW GRANTS
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Grants for root@localhost
GRANT SELECT, INSERT, UPDATE, DELETE, CR...
GRANT APPLICATION_PASSWORD_ADMIN,AU...
GRANT SELECT ON `mydb`.`finaid_main` TO `...
GRANT PROXY ON ''@'' TO 'root'@'localhost' WI...
Result 5
Read Only

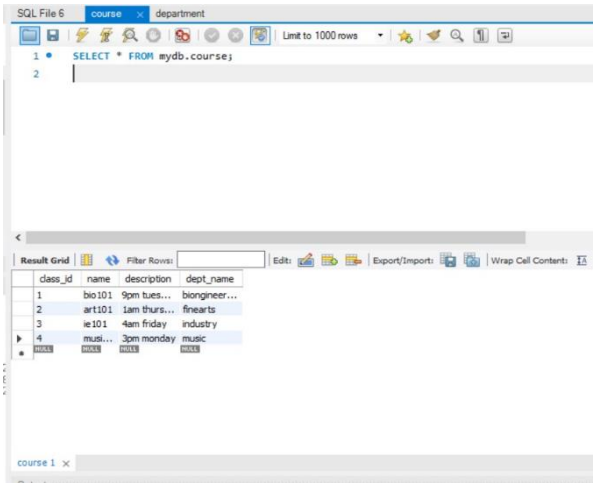

SQL File 10*
Limit to 1000 rows
REVOKE SELECT ON finaid_main FROM root@localhost;
SHOW GRANTS
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Grants for root@localhost
GRANT SELECT, INSERT, UPDATE, DELETE, CR...
GRANT APPLICATION_PASSWORD_ADMIN,AU...
GRANT PROXY ON ''@'' TO 'root'@'localhost' WI...
Result 9
Read Only

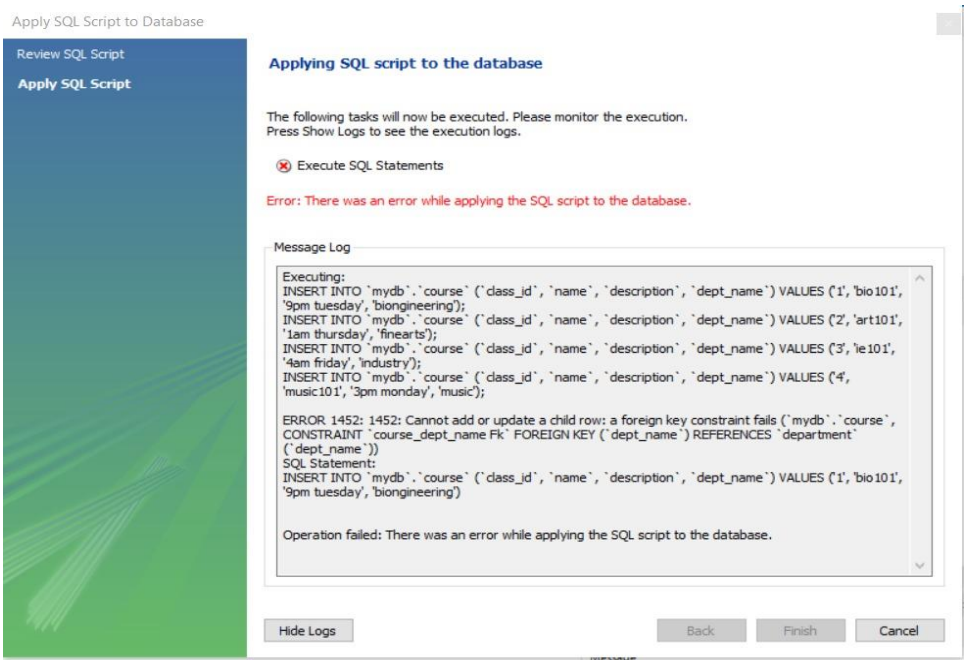
IX.INTEGRITY TESTS

1) A new line can not be insert into the table because role_description which does not exist in master table. As a shown below figure, it gives error as Unknown column 'role_description' in 'field list'.

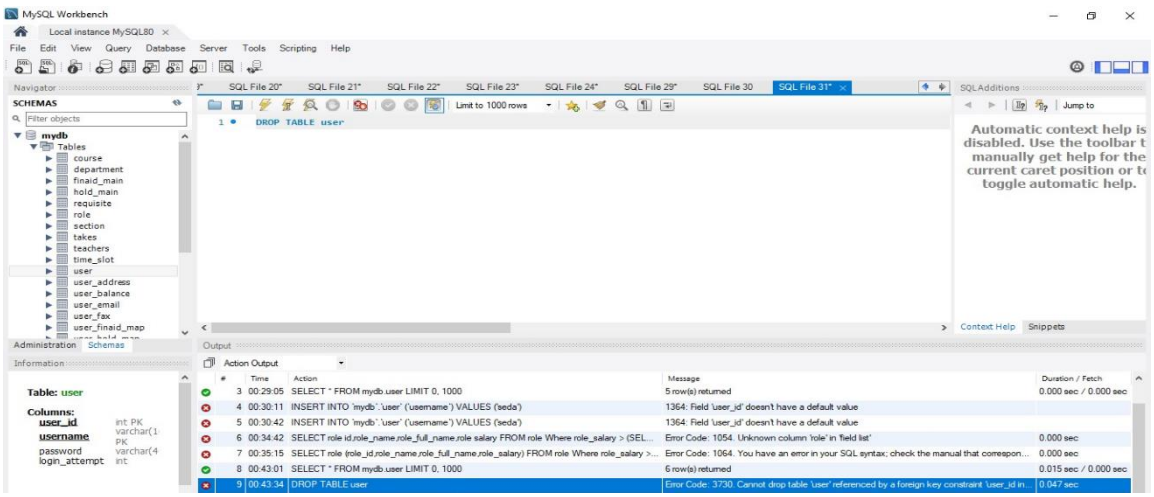


2) While applying in the SQL script to the database, it gives error as ‘a foreign key contraint fails’ . Since, we can not add or update a child row.

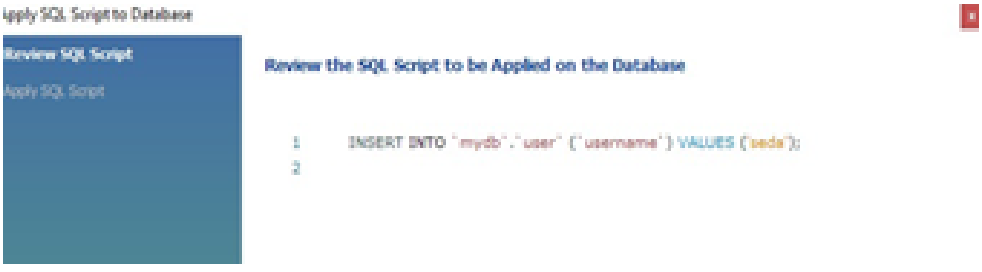




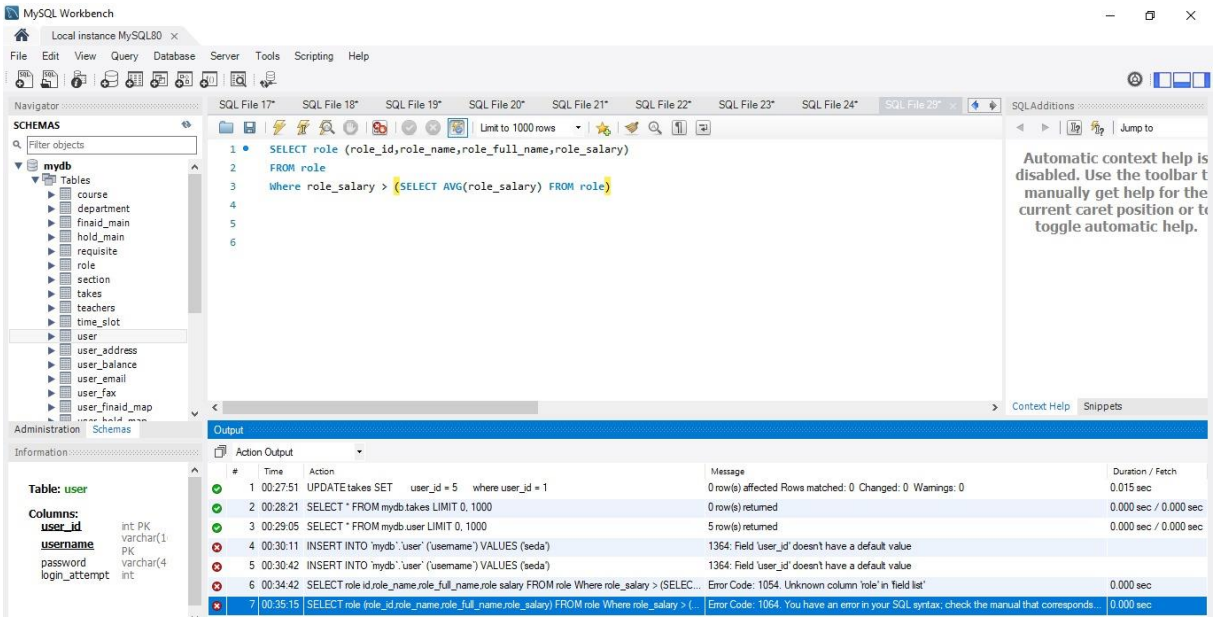
3) When applying the SQL script to the database, it gives error as "It can not drop table 'user' referenced by a foreign key constraint 'user_id int fk' on table 'user_role' ".



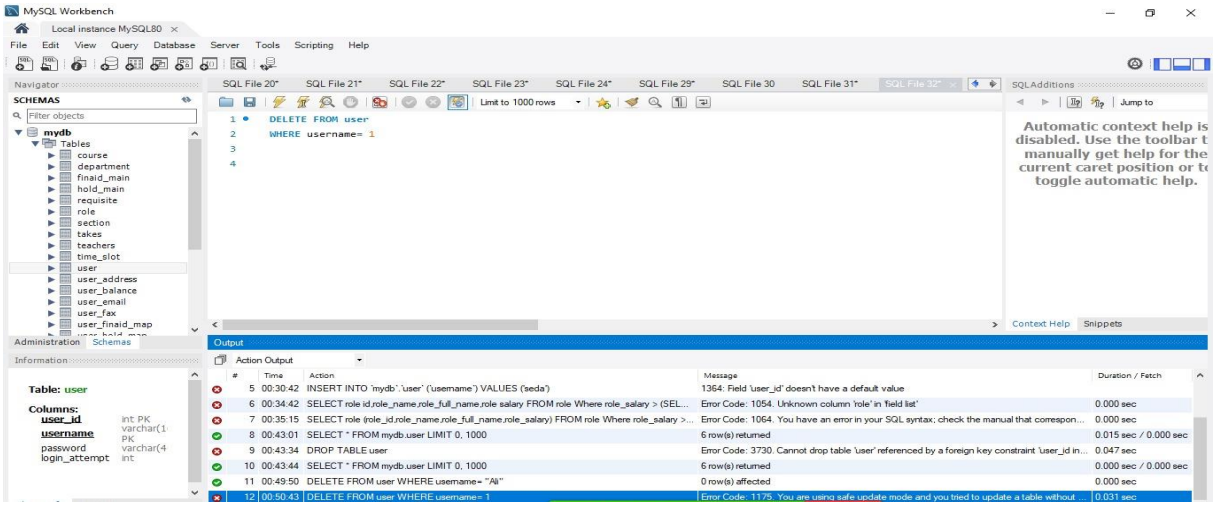
4) This gives error when applying the SQL script to the database. Because, field ‘user_id’ does not a default value.



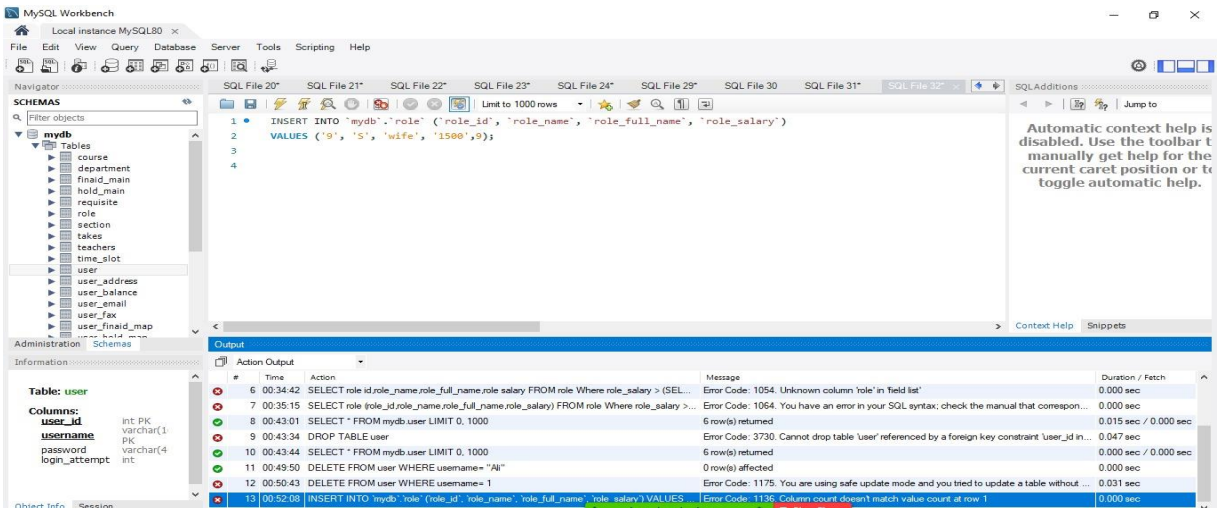
5) It gives syntax error as a "You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '(role_id,role_name,role_full_name,role_salary) FROM role Where role_salary > (SE' at line 1) "



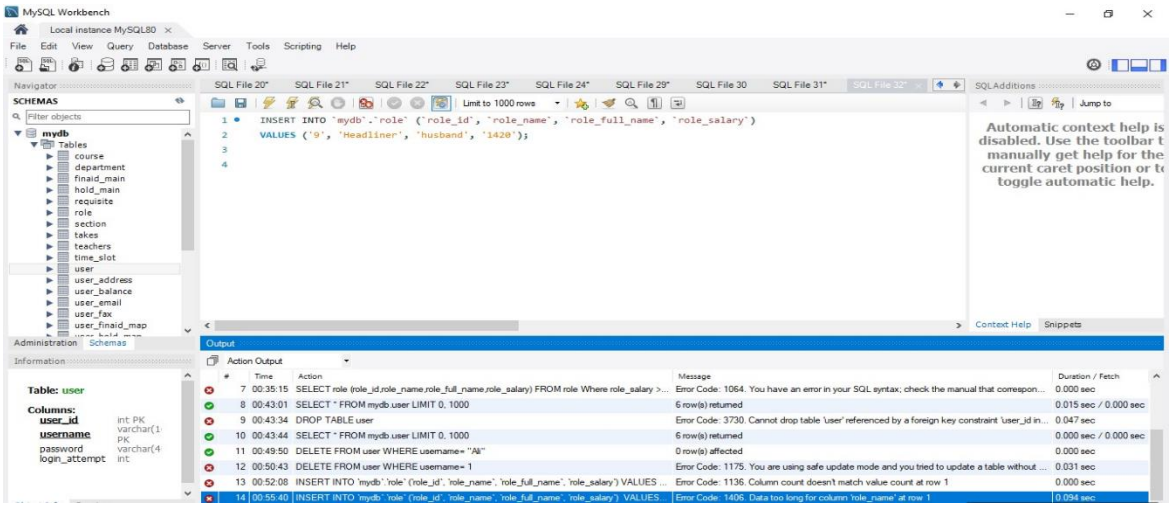
6) When SQL script is executed, it gives error as "You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column. Cannot use range access on index 'PRIMARY' due to type or collation conversion on field 'username' To disable safe mode, toggle the option in Preferences -> SQL Editor and reconnect.



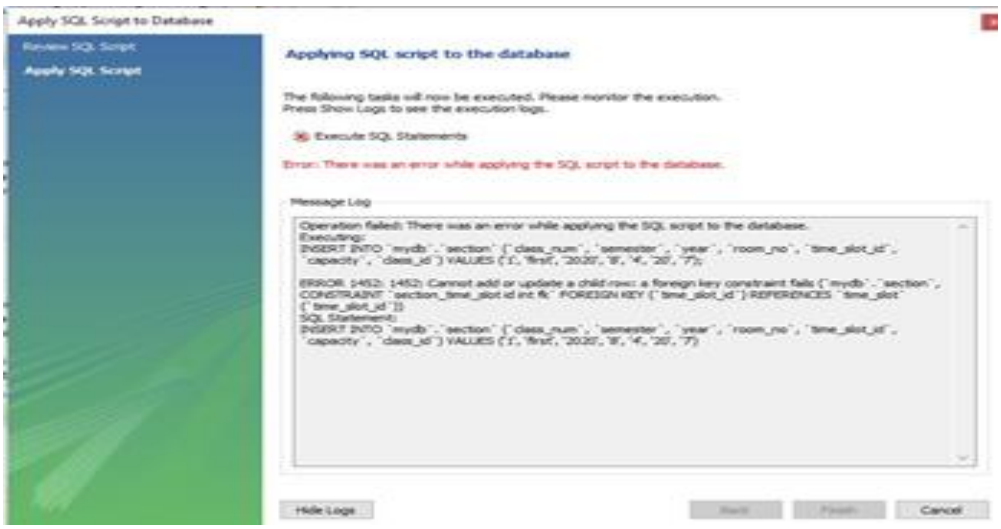
7) This script will not execute. Because, it gives error as "Column count doesn't match value count at row 1."



8) This script does not execute to database. It gives error as "Data too long for column 'role_name' at row 1". Because, we assigned to varchart of role_name as "3" .



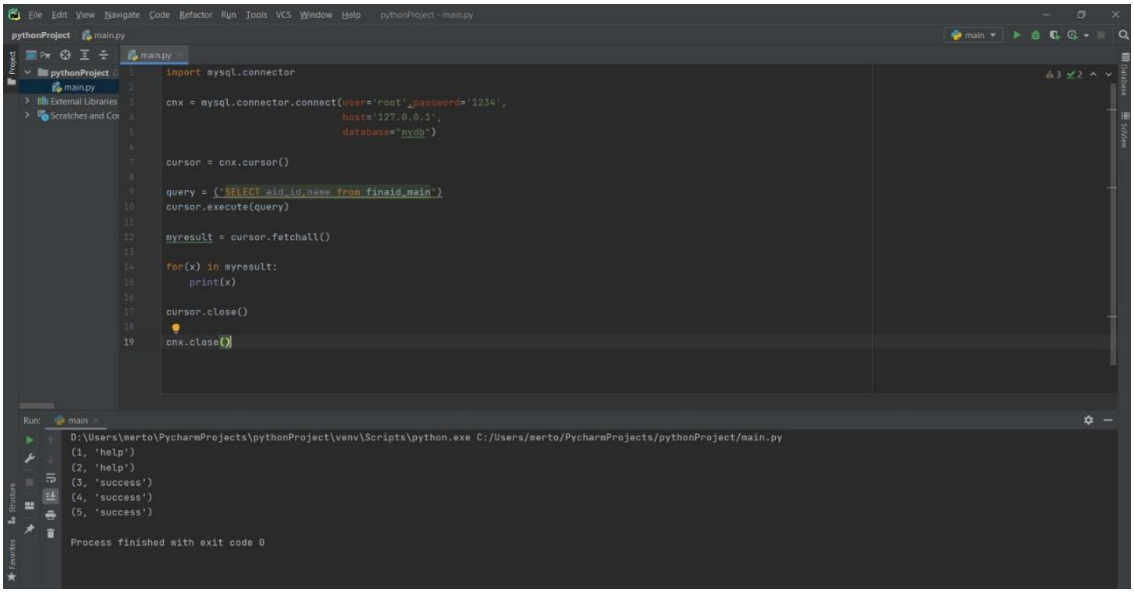
9) There was an error while applying in the SQL script to the database. It gives error as 'a foreign key contraint fails'. Since, we cannot add or update a child row.



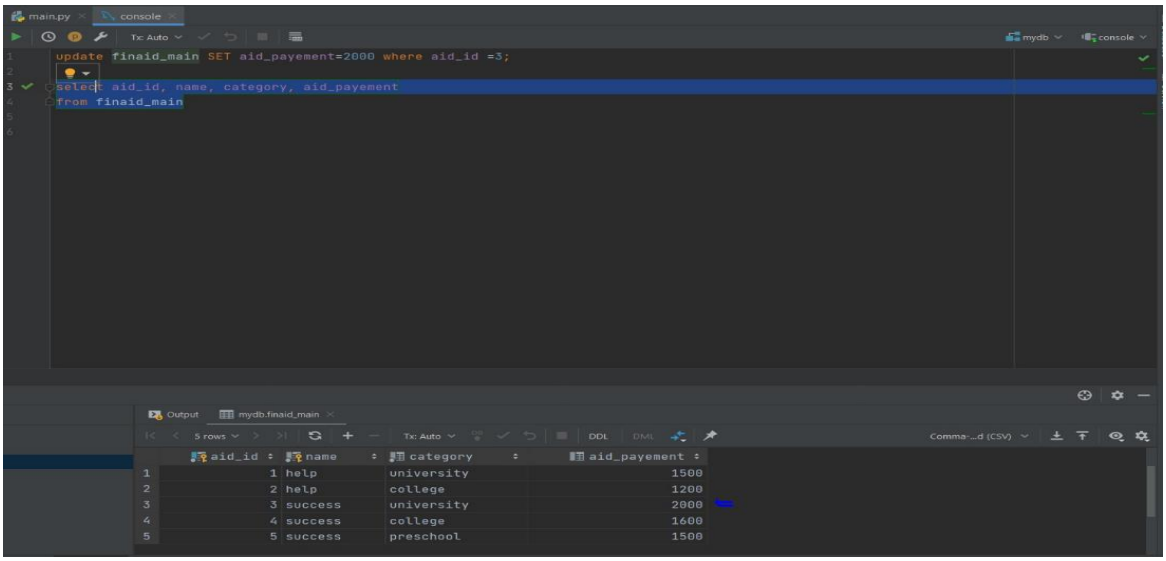
X.MANAGING DATABASE FROM A FRONT-END BY USING PYTHON

A.1. DESIGN SIMPLE FORM TO PROCESS ONE TABLE

- **SELECT**

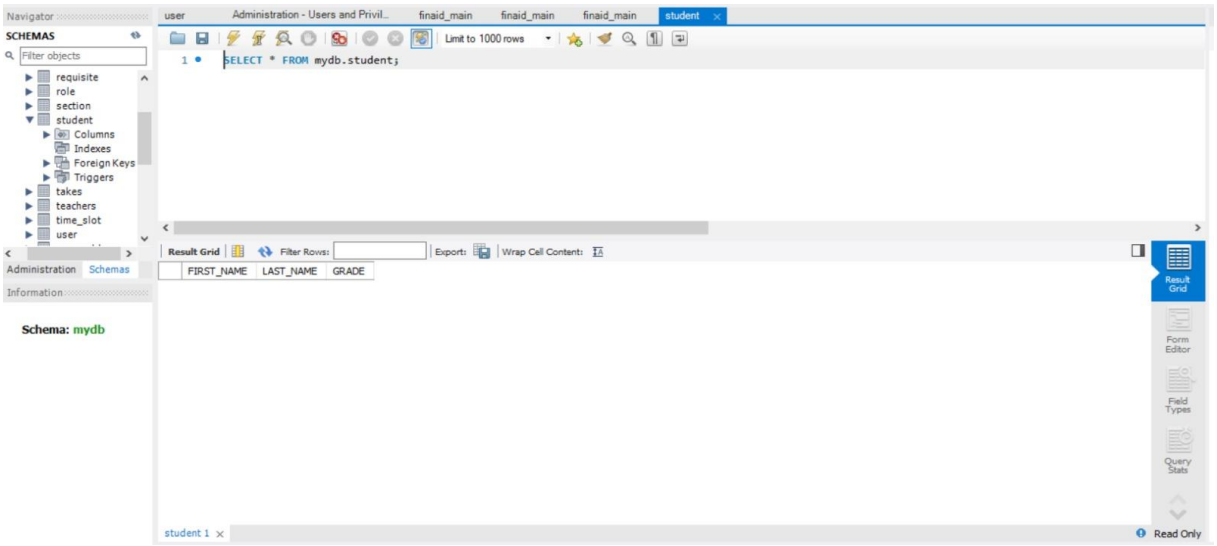


- **UPDATE**



- **CREATE**

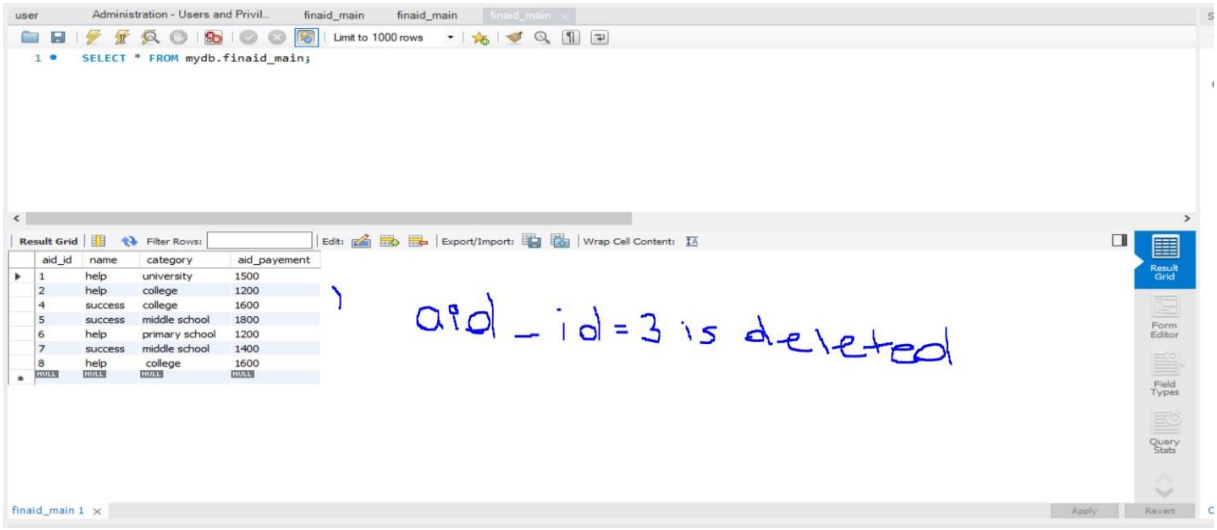
```python
import mysql.connector

cnx = mysql.connector.connect(user='root',password='1234',
                              host='127.0.0.1',
                              database="mydb")
cursor = cnx.cursor()

#Dropping EMPLOYEE table if already exists.
cursor.execute("DROP TABLE IF EXISTS STUDENT")

#Creating table as per requirement
sql ='''CREATE TABLE STUDENT(
   FIRST_NAME CHAR(20) NOT NULL,
   LAST_NAME CHAR(20) NOT NULL,
   GRADE INT NOT NULL
)'''
cursor.execute(sql)

cnx.close()
```



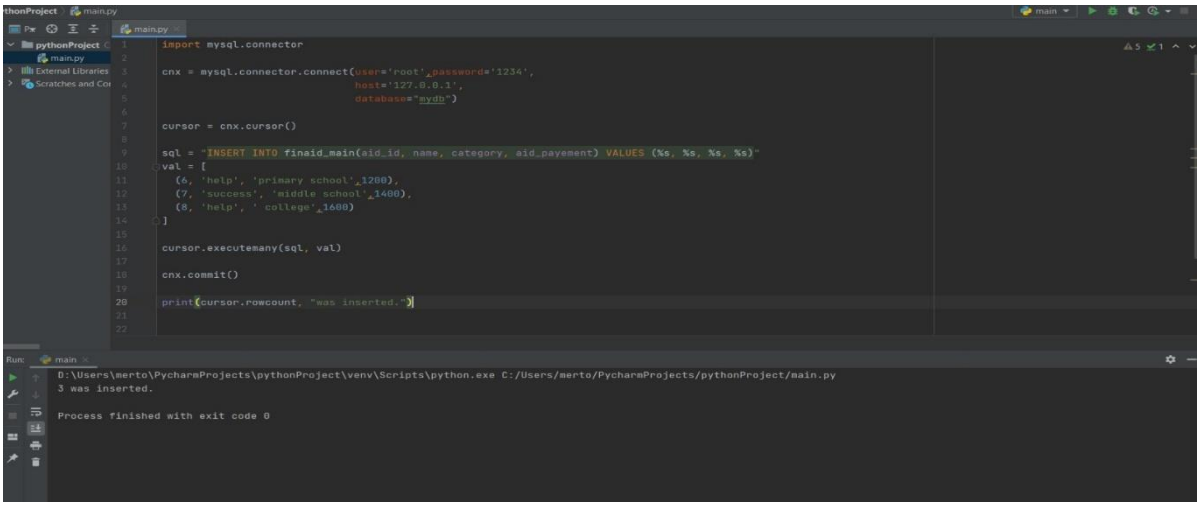
- **DELETE**

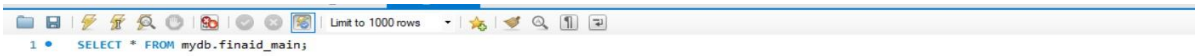
```python
    (8, 'help', ' college',1600)
]

cursor.executemany(sql, val)

cnx.commit()

print(cursor.rowcount, "was inserted.")
"""

query = " DELETE FROM finaid_main  WHERE aid_payement = '2000';"

cursor.execute(query)
cnx.commit()

print(cursor.rowcount, "record(s) deleted")

cnx.close()

cursor.close()
```

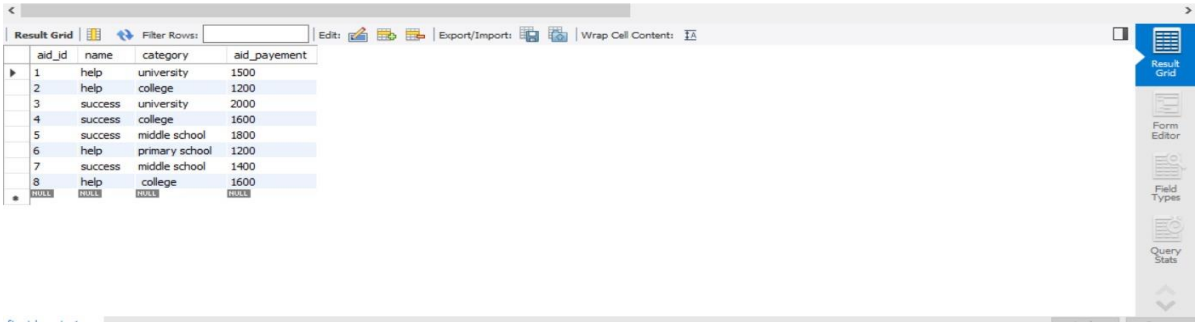
```
D:\Users\merto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/merto/PycharmProjects/pythonProject/main.py
1 record(s) deleted

Process finished with exit code 0
```


user
Administration - Users and Privil...
finaid_main
finaid_main
finaid_main
Limit to 1000 rows
1 • SELECT * FROM mydb.finaid_main;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
aid_id name category aid_payement
1 help university 1500
2 help college 1200
4 success college 1600
5 success middle school 1800
6 help primary school 1200
7 success middle school 1400
8 help college 1600
aid_id=3 is deleted
Result Grid
Form Editor
Field Types
Query Stats
finaid_main 1
Apply
Revert

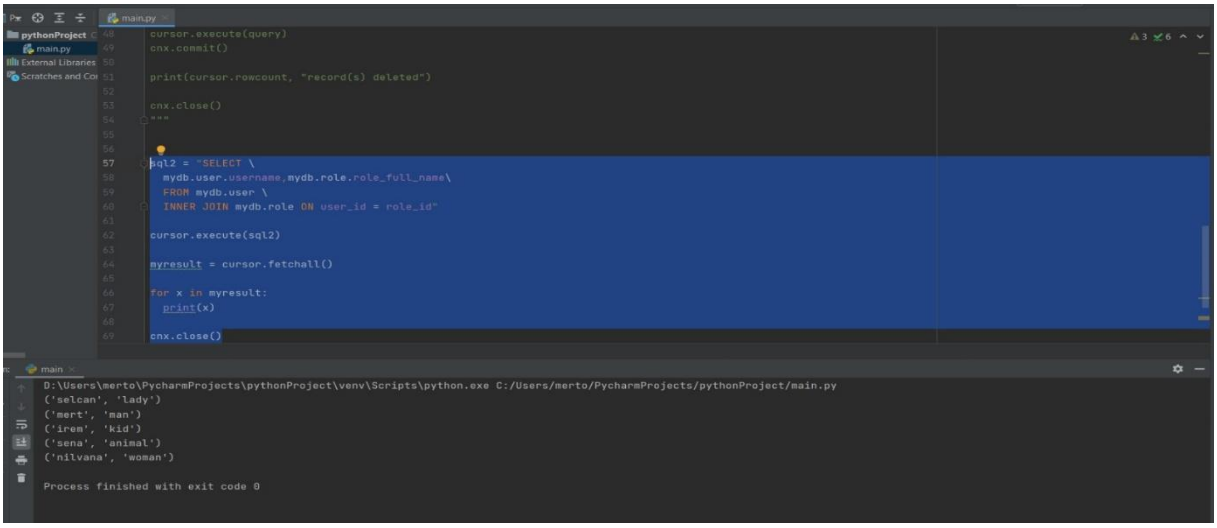
- **INSERT**

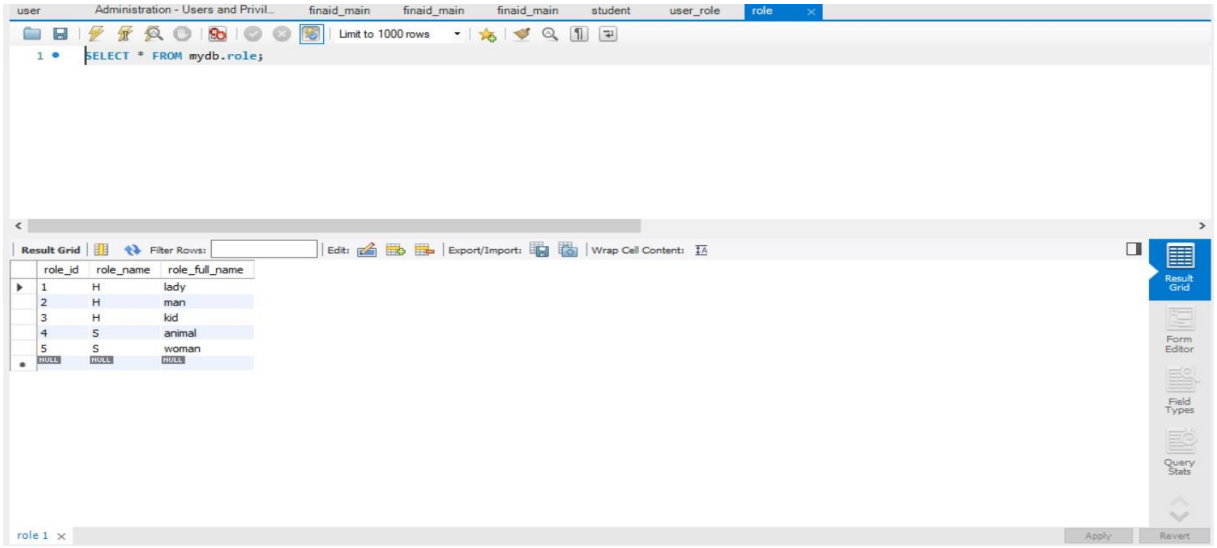

main.py
import mysql.connector
cnx = mysql.connector.connect(user='root',password='1234',
host='127.0.0.1',
database="mydb")
cursor = cnx.cursor()
sql = "INSERT INTO finaid_main(aid_id, name, category, aid_payement) VALUES (%s, %s, %s, %s)"
val = [
(6, 'help', 'primary school',1200),
(7, 'success', 'middle school',1400),
(8, 'help', ' college',1600)
]
cursor.executemany(sql, val)
cnx.commit()
print(cursor.rowcount, "was inserted.")
Run: main
D:\Users\merto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/merto/PycharmProjects/pythonProject/main.py
3 was inserted.
Process finished with exit code 0

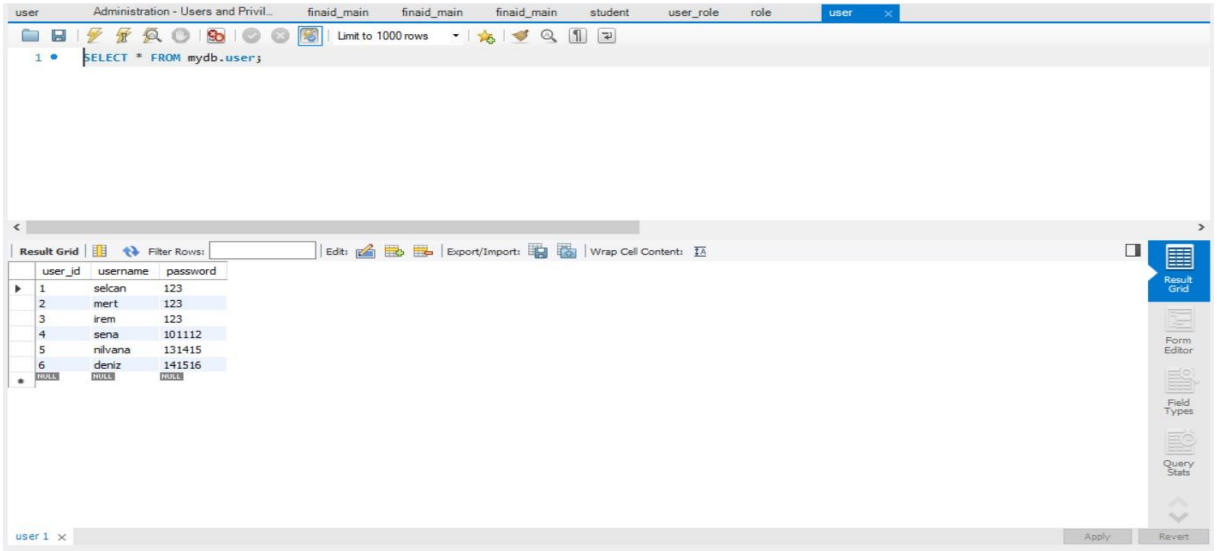

Limit to 1000 rows
1 • SELECT * FROM mydb.finaid_main;


Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
aid_id name category aid_payement
1 help university 1500
2 help college 1200
3 success university 2000
4 success college 1600
5 success middle school 1800
6 help primary school 1200
7 success middle school 1400
8 help college 1600
Result Grid
Form Editor
Field Types
Query Stats

A.2. DESIGN A MASTER/DETAIL TABLE (TWO TABLES AT THE SAME TIME)

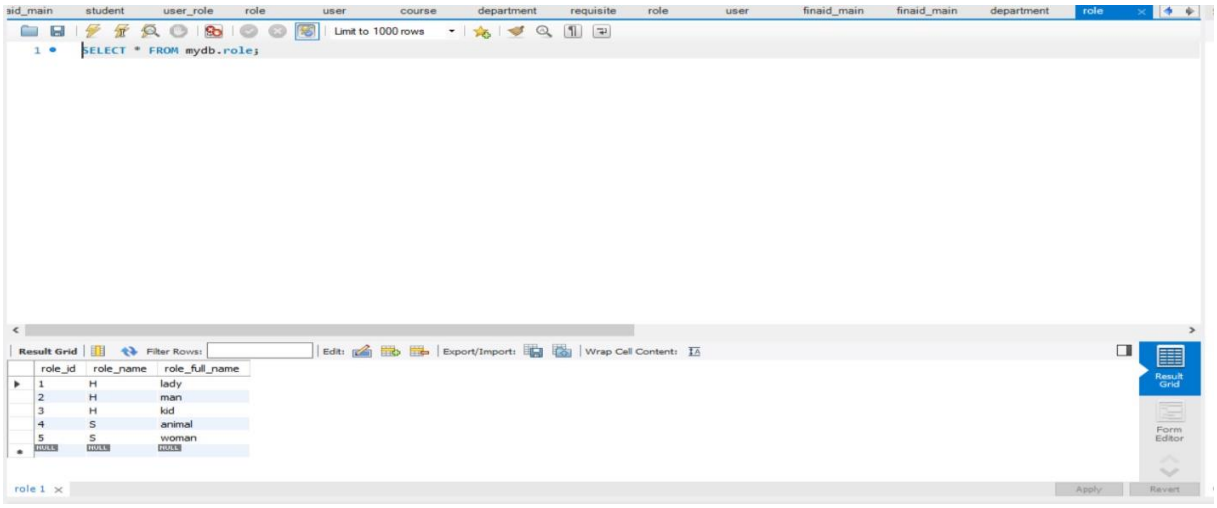


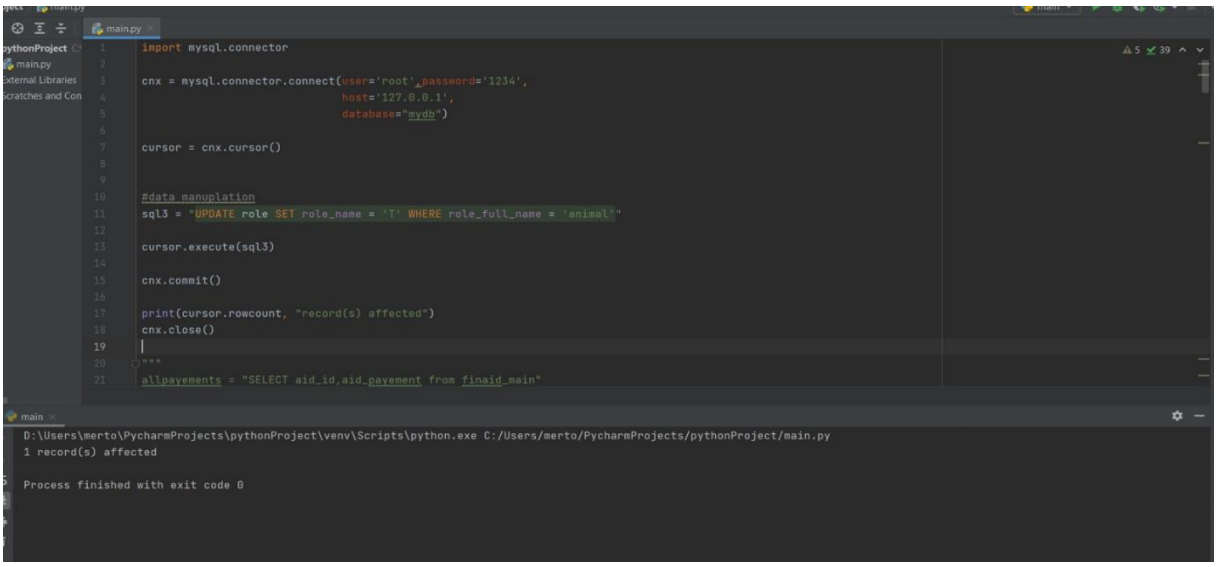


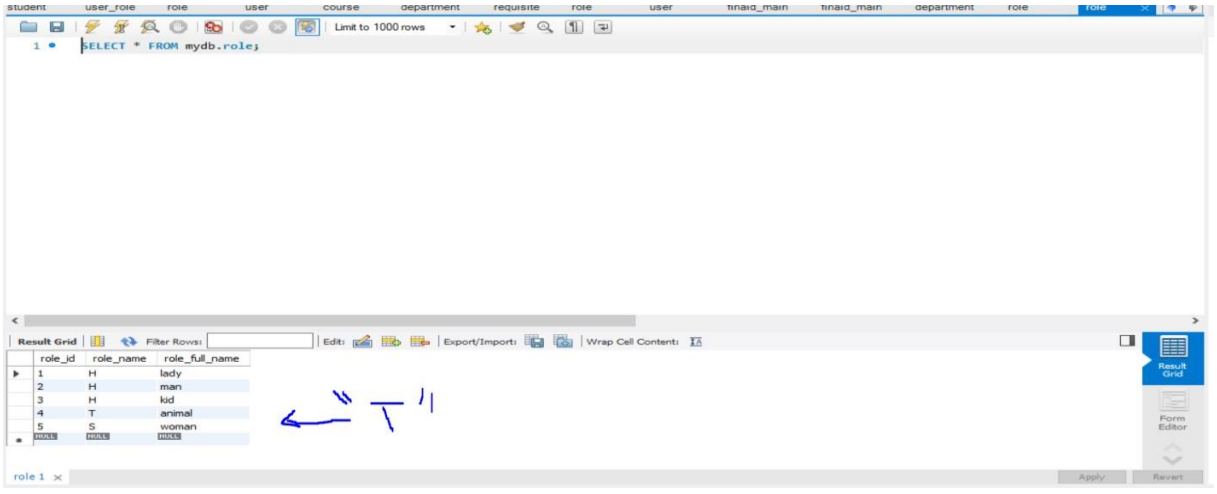


A.3. OPEN ENDED. SHOW A PROCESS ON THE FORM TO MANIPULATE THE DATA IN 3 TABLES.

- **DATA MANIPULATION & STORED PROCEDURE**







```
import pymysql
from werkzeug.security import generate_password_hash, check_password_hash

try:
    conn = pymysql.connect(host='localhost', database='mydb', user='root', password='1234')
    cur = conn.cursor()
    _hashed_password = generate_password_hash('secret')
    cur.callproc('sp_createUser',('Soumitra Roy','contact@roytuts.com',_hashed_password))
    data = cur.fetchall()
    if len(data) == 0:
        conn.commit()
        print('User information saved successfully !')
    else:
        print('error: ', str(data[0]))
except Exception as e:
    print(e)
finally:
    cur.close()
    conn.close()
```

```
D:\Users\merto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/merto/PycharmProjects/pythonProject/deneme.py
(1305, 'PROCEDURE mydb.sp_createUser does not exist')

Process finished with exit code 0
```

- **INSERT**

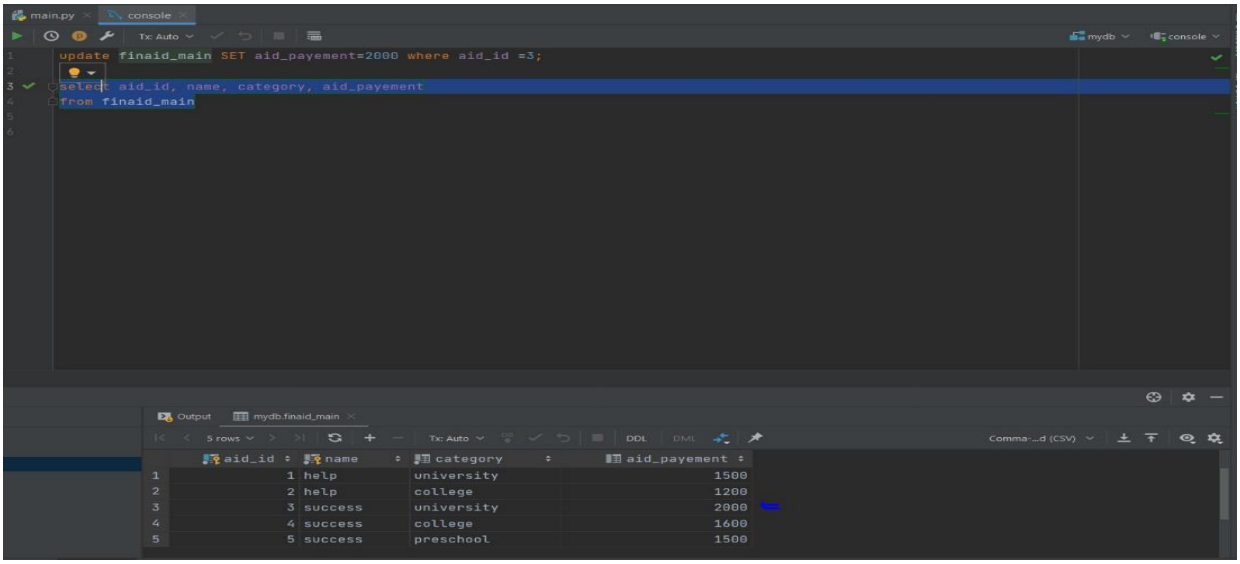
```
import mysql.connector

cnx = mysql.connector.connect(user='root',password='1234',
                              host='127.0.0.1',
                              database="mydb")

cursor = cnx.cursor()

sql = "INSERT INTO finaid_main(aid_id, name, category, aid_payement) VALUES (%s, %s, %s, %s)"
val = [
  (6, 'help', 'primary school',1200),
  (7, 'success', 'middle school',1400),
  (8, 'help', ' college',1600)
]

cursor.executemany(sql, val)

cnx.commit()

print(cursor.rowcount, "was inserted.")
```

```
D:\Users\merto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/merto/PycharmProjects/pythonProject/main.py
3 was inserted.

Process finished with exit code 0
```

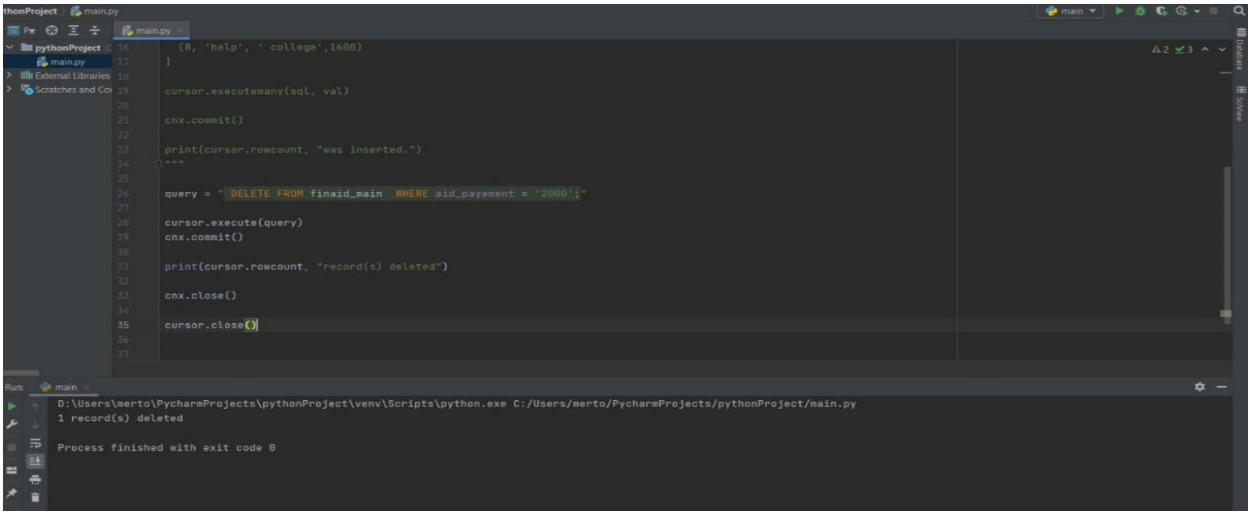
```
SELECT * FROM mydb.finaid_main;
```

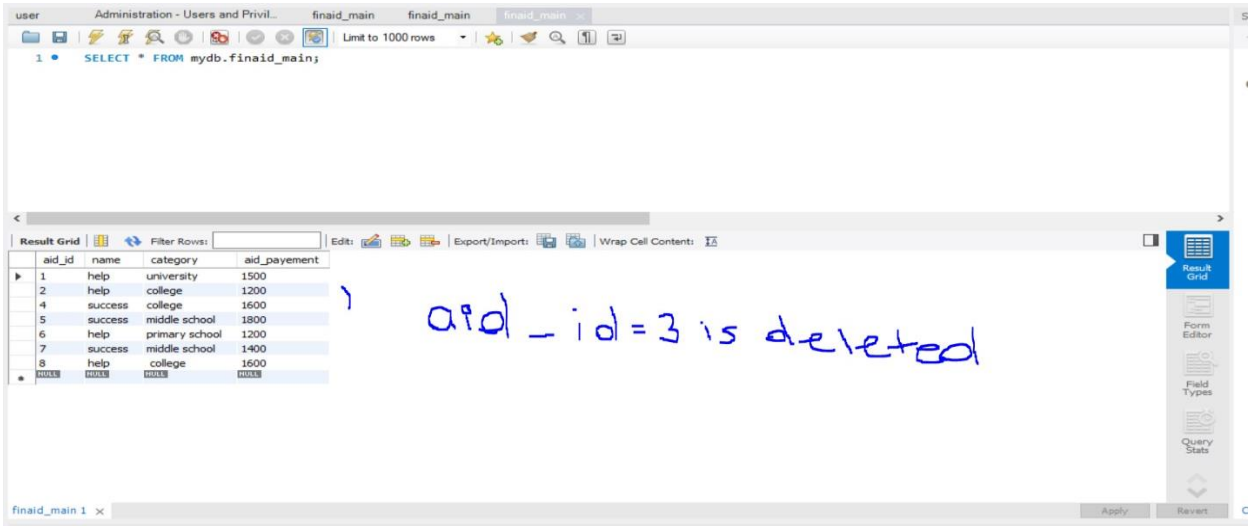
aid_id	name	category	aid_payement
1	help	university	1500
2	help	college	1200
3	success	university	2000
4	success	college	1600
5	success	middle school	1800
6	help	primary school	1200
7	success	middle school	1400
8	help	college	1600

- **UPDATE**

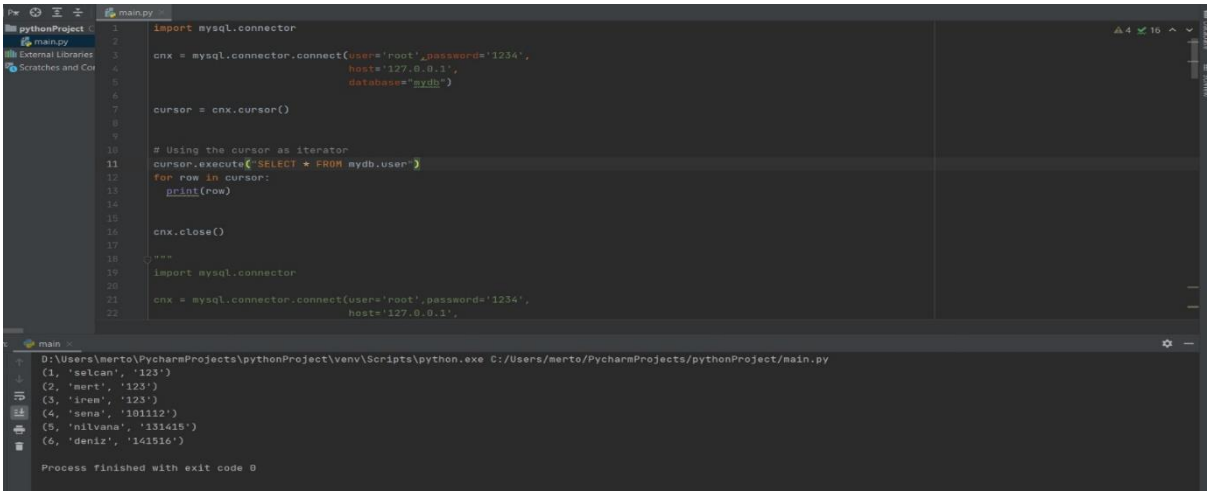


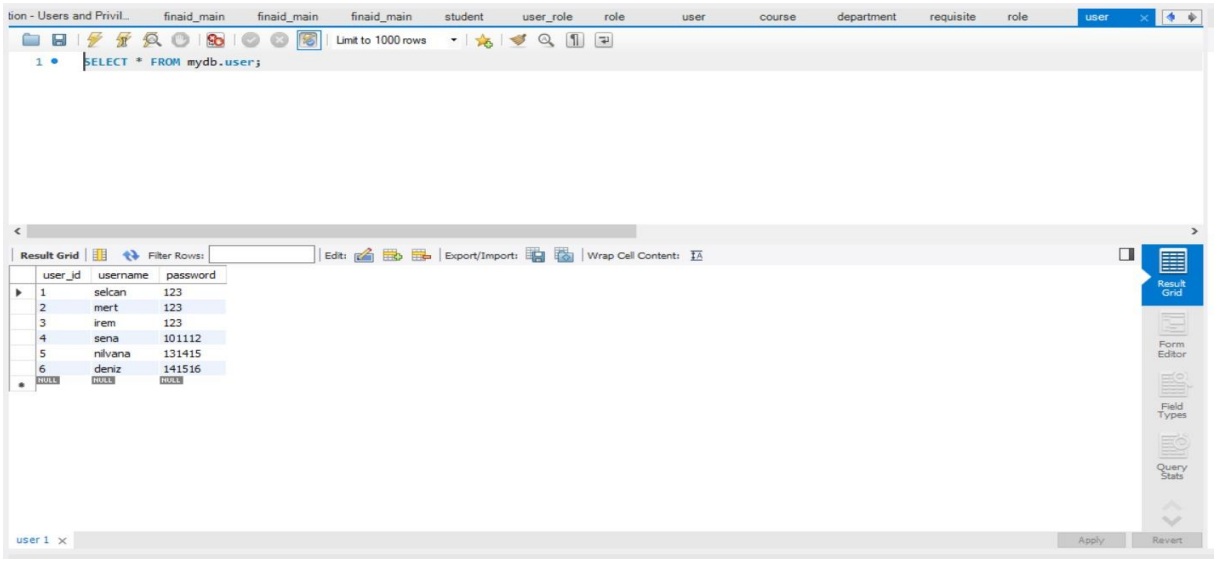
- **DELETE**

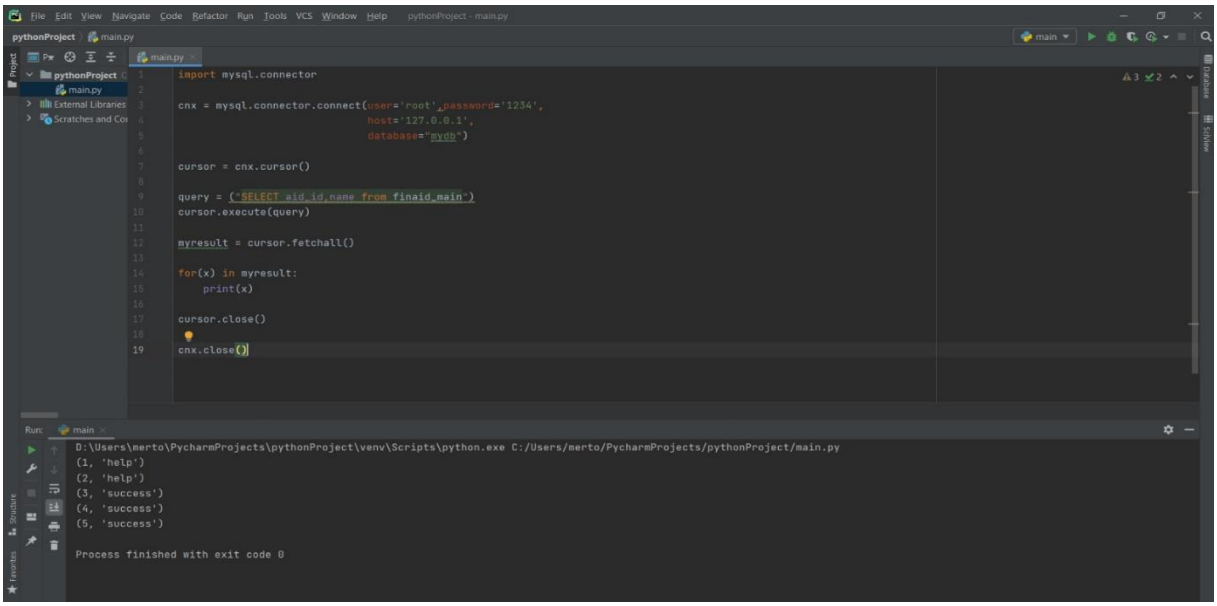


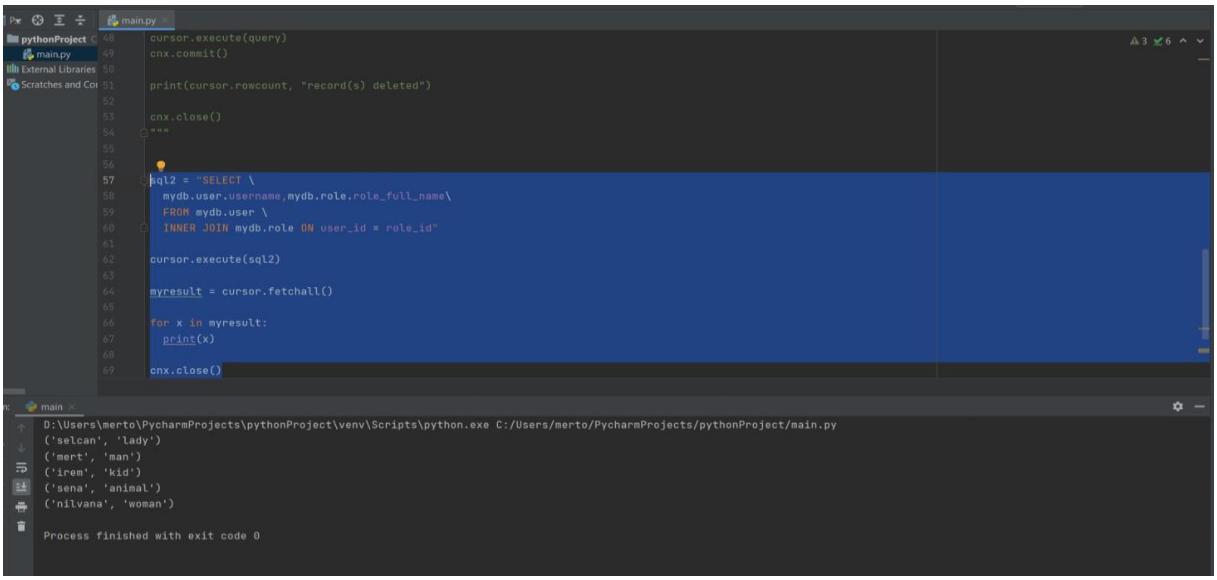


- **FOR**

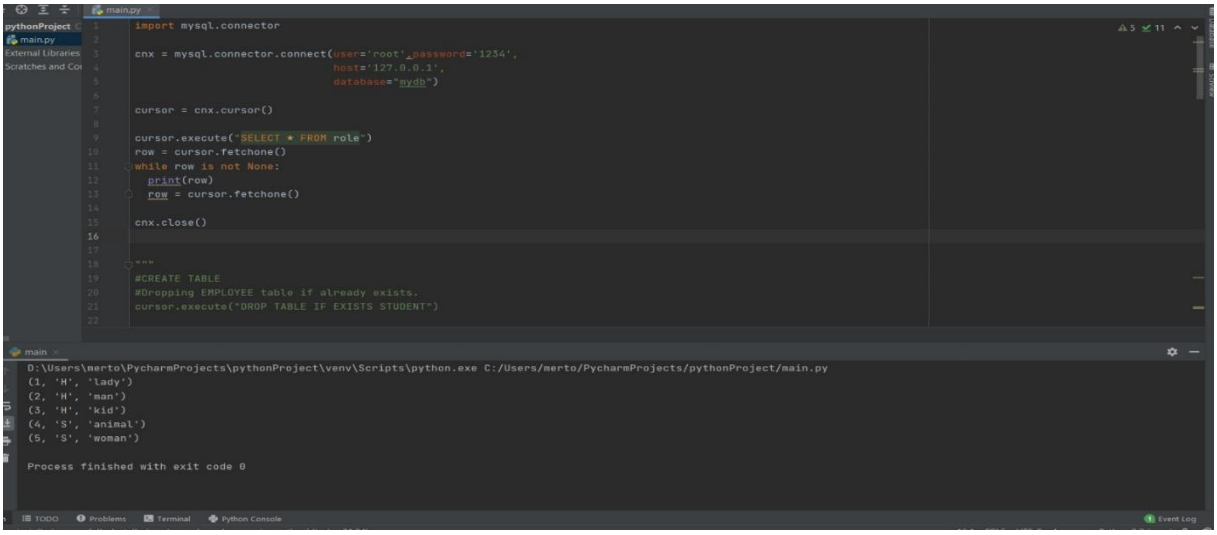

import mysql.connector
cnx = mysql.connector.connect(user='root',password='1234',
host='127.0.0.1',
database="mydb")
cursor = cnx.cursor()
# Using the cursor as iterator
cursor.execute("SELECT * FROM mydb.user")
for row in cursor:
print(row)
cnx.close()
"""
import mysql.connector
cnx = mysql.connector.connect(user='root',password='1234',
host='127.0.0.1',
D:\Users\merto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/merto/PycharmProjects/pythonProject/main.py
(1, 'selcan', '123')
(2, 'mert', '123')
(3, 'irem', '123')
(4, 'sena', '101112')
(5, 'nilvana', '131415')
(6, 'deniz', '141516')
Process finished with exit code 0

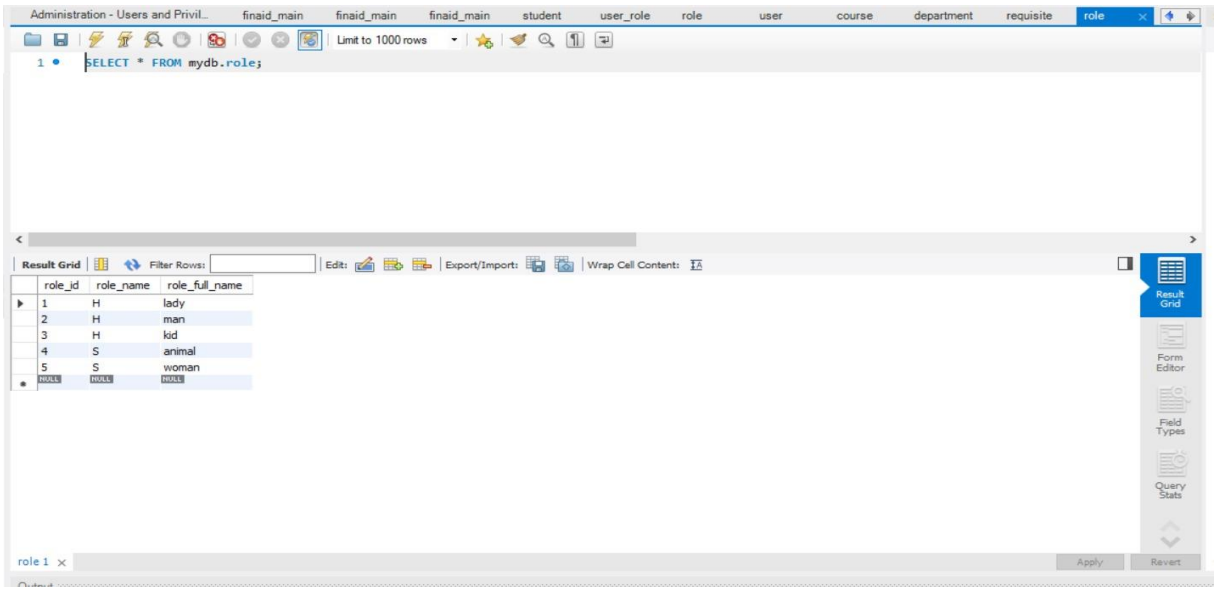

SELECT * FROM mydb.user;
Limit to 1000 rows
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
user_id username password
1 selcan 123
2 mert 123
3 irem 123
4 sena 101112
5 nilvana 131415
6 deniz 141516
Result Grid
Form Editor
Field Types
Query Stats
user 1
Apply
Revert


import mysql.connector
cnx = mysql.connector.connect(user='root',password='1234',
host='127.0.0.1',
database="mydb")
cursor = cnx.cursor()
query = ("SELECT aid_id,name from finaid_main")
cursor.execute(query)
myresult = cursor.fetchall()
for(x) in myresult:
print(x)
cursor.close()
cnx.close()
D:\Users\merto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/merto/PycharmProjects/pythonProject/main.py
(1, 'help')
(2, 'help')
(3, 'success')
(4, 'success')
(5, 'success')
Process finished with exit code 0

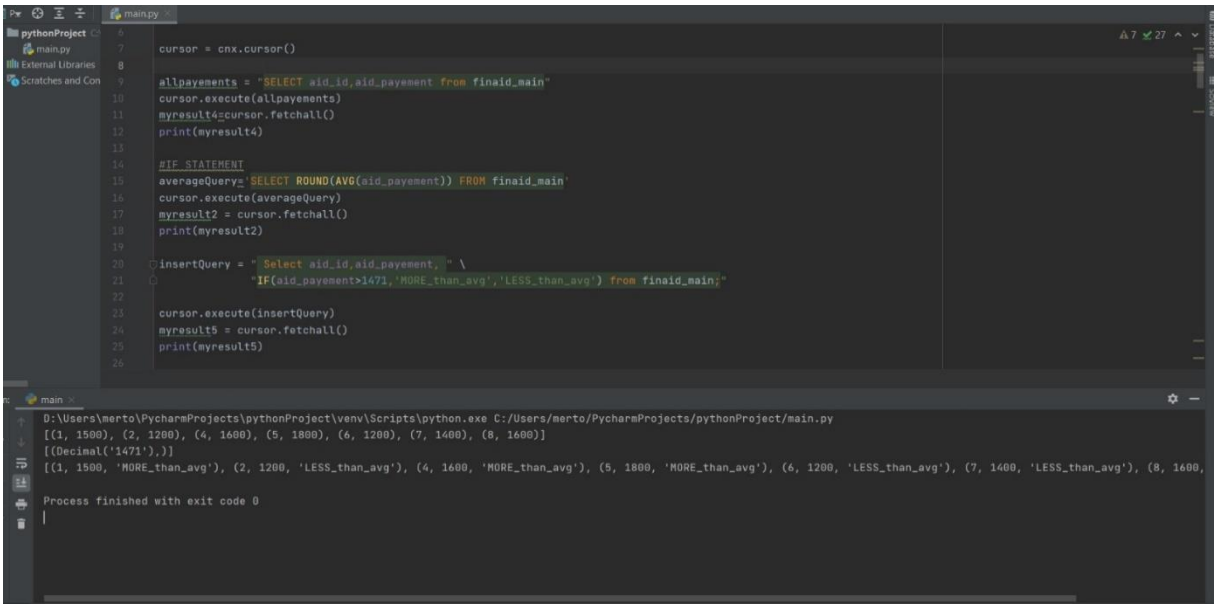

cursor.execute(query)
cnx.commit()
print(cursor.rowcount, "record(s) deleted")
cnx.close()
"""
sql2 = "SELECT \
mydb.user.username,mydb.role.role_full_name\
FROM mydb.user \
INNER JOIN mydb.role ON user_id = role_id"
cursor.execute(sql2)
myresult = cursor.fetchall()
for x in myresult:
print(x)
cnx.close()
D:\Users\merto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/merto/PycharmProjects/pythonProject/main.py
('selcan', 'lady')
('mert', 'man')
('irem', 'kid')
('sena', 'animal')
('nilvana', 'woman')
Process finished with exit code 0

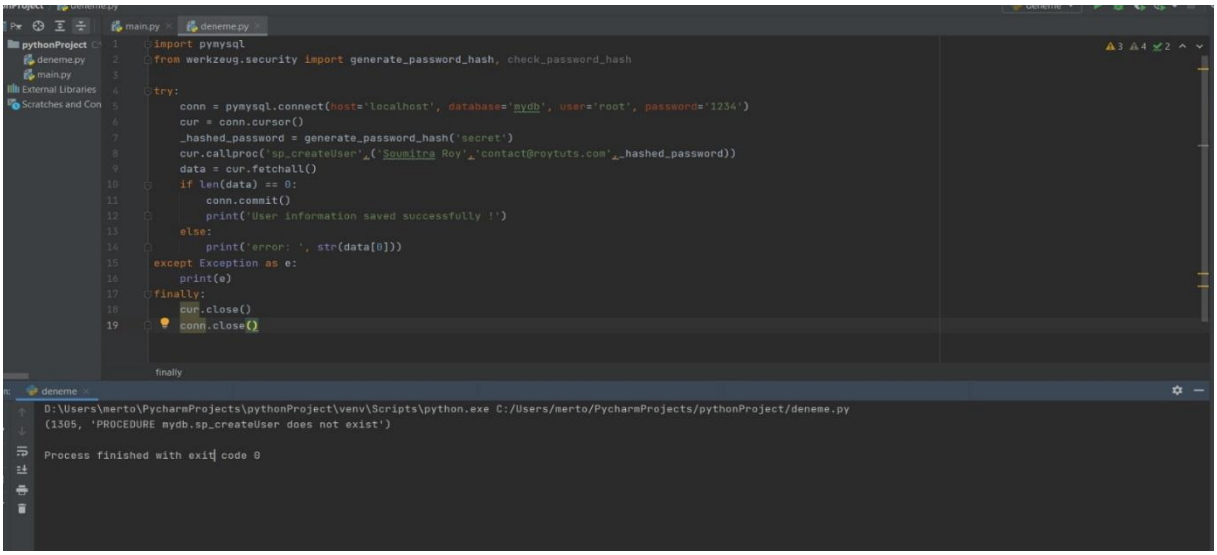
- **WHILE**


import mysql.connector
cnx = mysql.connector.connect(user='root',password='1234',
host='127.0.0.1',
database="mydb")
cursor = cnx.cursor()
cursor.execute("SELECT * FROM role")
row = cursor.fetchone()
while row is not None:
print(row)
row = cursor.fetchone()
cnx.close()
"""
#CREATE TABLE
#Dropping EMPLOYEE table if already exists.
cursor.execute("DROP TABLE IF EXISTS STUDENT")
D:\Users\merto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/merto/PycharmProjects/pythonProject/main.py
(1, 'H', 'lady')
(2, 'H', 'man')
(3, 'H', 'kid')
(4, 'S', 'animal')
(5, 'S', 'woman')
Process finished with exit code 0


Administration - Users and Privil...
finaid_main
finaid_main
finaid_main
student
user_role
role
user
course
department
requisite
role
Limit to 1000 rows
SELECT * FROM mydb.role;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
role_id role_name role_full_name
1 H lady
2 H man
3 H kid
4 S animal
5 S woman
Result Grid
Form Editor
Field Types
Query Stats
role 1
Apply
Revert

- **IF/THEN**


main.py
pythonProject
main.py
External Libraries
Scratches and Con
cursor = cnx.cursor()
allpayements = "SELECT aid_id,aid_payement from finaid_main"
cursor.execute(allpayements)
myresult4=cursor.fetchall()
print(myresult4)
#IF STATEMENT
averageQuery='SELECT ROUND(AVG(aid_payement)) FROM finaid_main'
cursor.execute(averageQuery)
myresult2 = cursor.fetchall()
print(myresult2)
insertQuery = " Select aid_id,aid_payement, " \
"IF(aid_payement>1471,'MORE_than_avg','LESS_than_avg') from finaid_main;"
cursor.execute(insertQuery)
myresult5 = cursor.fetchall()
print(myresult5)
main
D:\Users\merto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/merto/PycharmProjects/pythonProject/main.py
[(1, 1500), (2, 1200), (4, 1600), (5, 1800), (6, 1200), (7, 1400), (8, 1600)]
[(Decimal('1471'),)]
[(1, 1500, 'MORE_than_avg'), (2, 1200, 'LESS_than_avg'), (4, 1600, 'MORE_than_avg'), (5, 1800, 'MORE_than_avg'), (6, 1200, 'LESS_than_avg'), (7, 1400, 'LESS_than_avg'), (8, 1600,
Process finished with exit code 0


main.py
deneme.py
pythonProject
deneme.py
main.py
External Libraries
Scratches and Con
import pymysql
from werkzeug.security import generate_password_hash, check_password_hash
try:
conn = pymysql.connect(host='localhost', database='mydb', user='root', password='1234')
cur = conn.cursor()
_hashed_password = generate_password_hash('secret')
cur.callproc('sp_createUser',('Soumitra Roy','contact@roytuts.com',_hashed_password))
data = cur.fetchall()
if len(data) == 0:
conn.commit()
print('User information saved successfully !')
else:
print('error: ', str(data[0]))
except Exception as e:
print(e)
finally:
cur.close()
conn.close()
finally
deneme
D:\Users\merto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/merto/PycharmProjects/pythonProject/deneme.py
(1305, 'PROCEDURE mydb.sp_createUser does not exist')
Process finished with exit code 0

CONCLUSION

Thanks to this project, we gained extensive experience about MySQL. MySQL is based on a client-server model. MySQL is a relational database management system that helps manage the database stored in different tables on the computer. Another program we use, Erwin, is to increase awareness about data assets and to support decision-making processes by expanding access to data cataloging, data literacy. It is a data intelligence solution that combines and automation capabilities.

With this project, we had the chance to learn many new knowledge and skills. Firstly, we learned to create relational sentences with Oracle Barker Notations, then we transformed these sentences into photo format using Oracle. We also experienced showing this project with IDEF1X notations at Erwin. Next, we used SQL DDL (Data Definition Language) statements such as create / alter on data models. Besides, we learned to make changes on the database using DML (Database Management System) statements such as insert, update and delete. We had the opportunity to transform the database we created into Database Design Document using Erwin. Thus, we learned how to actually report the database.

In addition, we had the opportunity to improve our SQL skills in this section by choosing 3 of the tables that created by our. With the 9 tests we performed with the check Database Integrity task, we checked the allocation and structural integrity of all objects in the specified database and according to the results obtained, the database was better understood. Importantly, we were able to integrate MySQL to a different software such as Python.

On the other hand, this project gave us the ability to work as a group in a planned way. Moreover, we were faced with many problems while doing the project, we found solutions by researching these problems. Thus, our solution-oriented approach to problems improved. Lastly, in line with the last determined goal / problem, this project was like a Management Systems Simulation for us, and that's why we learned crisis management.