

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

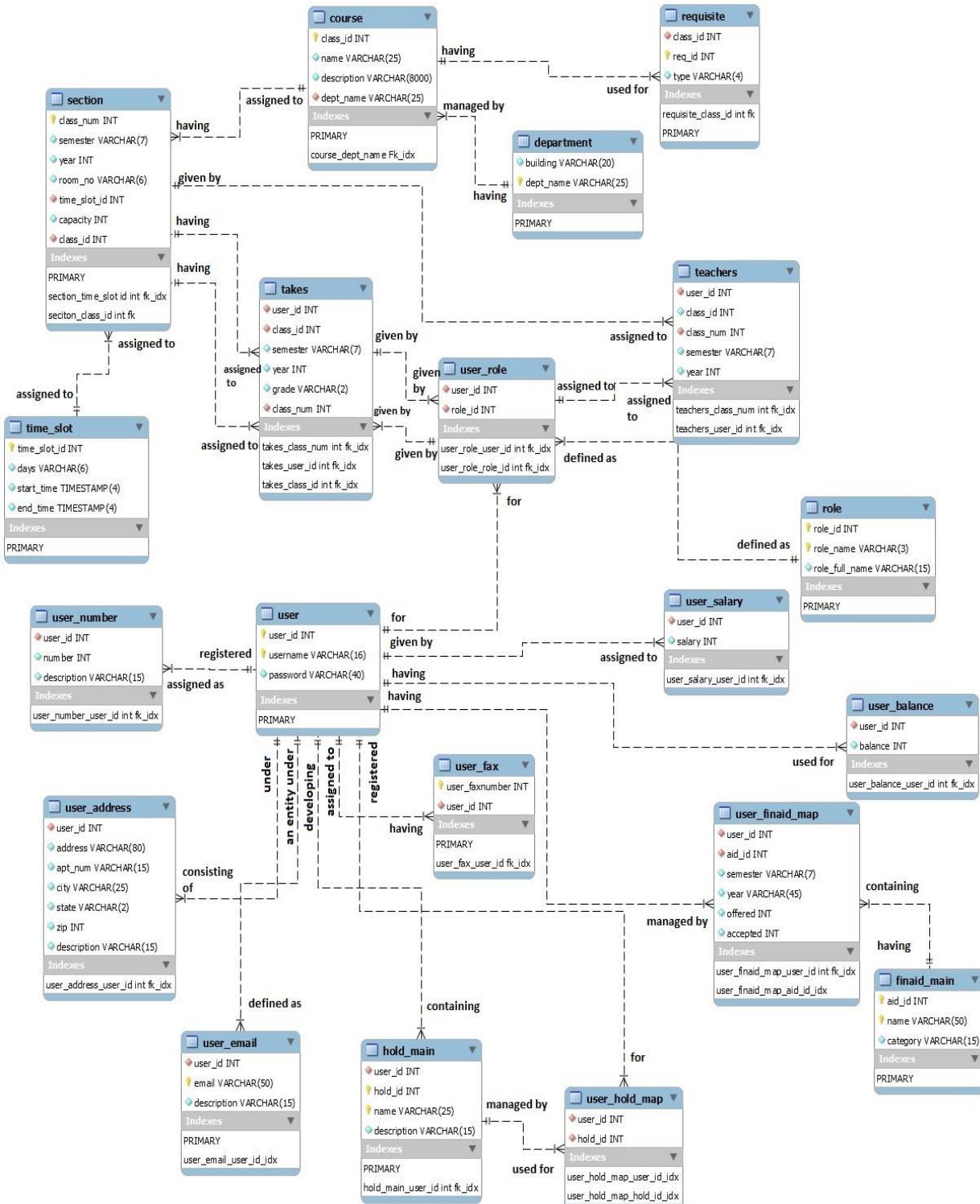
Each USER may be having one or more USER FINAID MAP

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

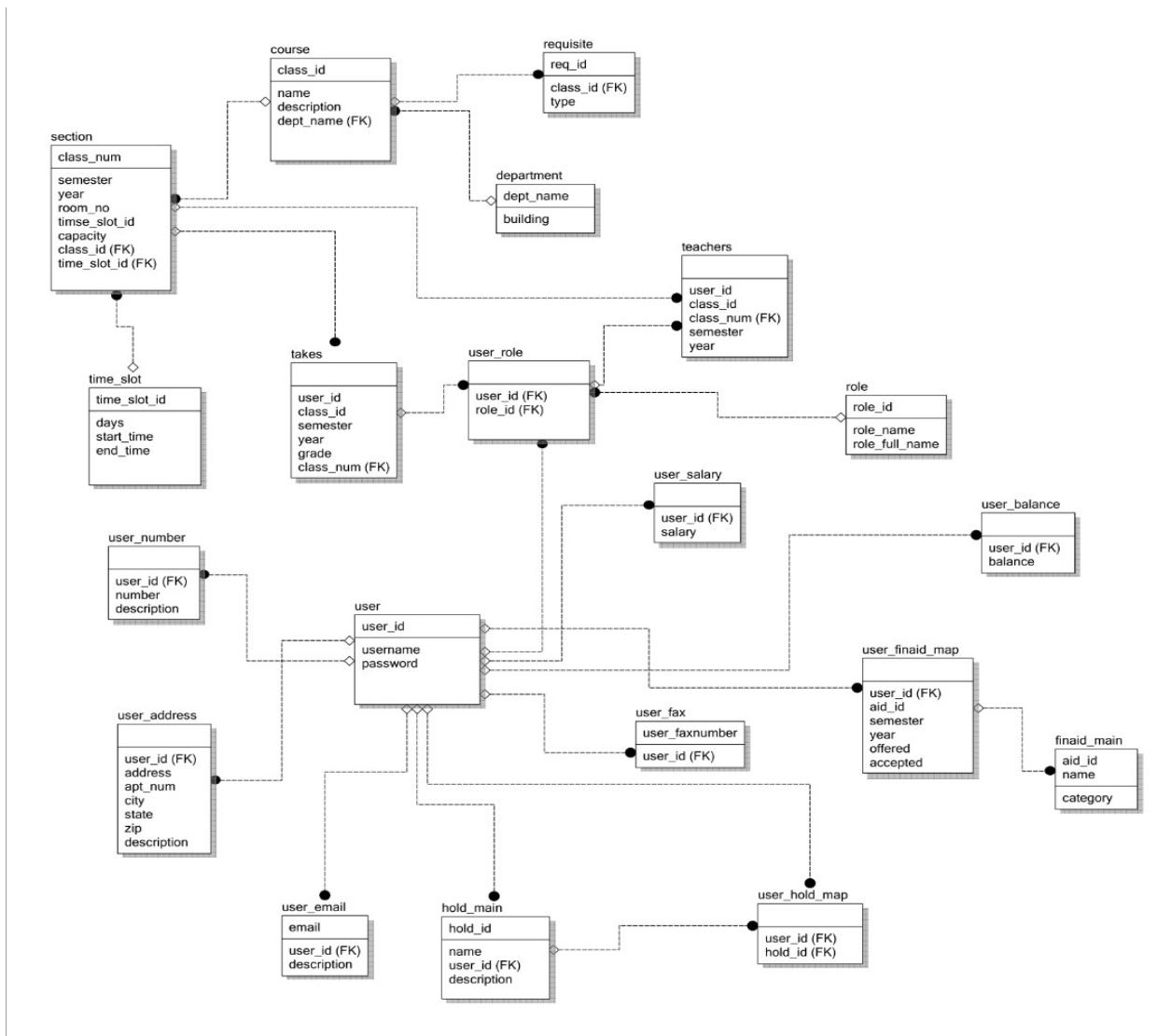
Each FINAID MAIN may be having one or more USER FINAID MAP

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

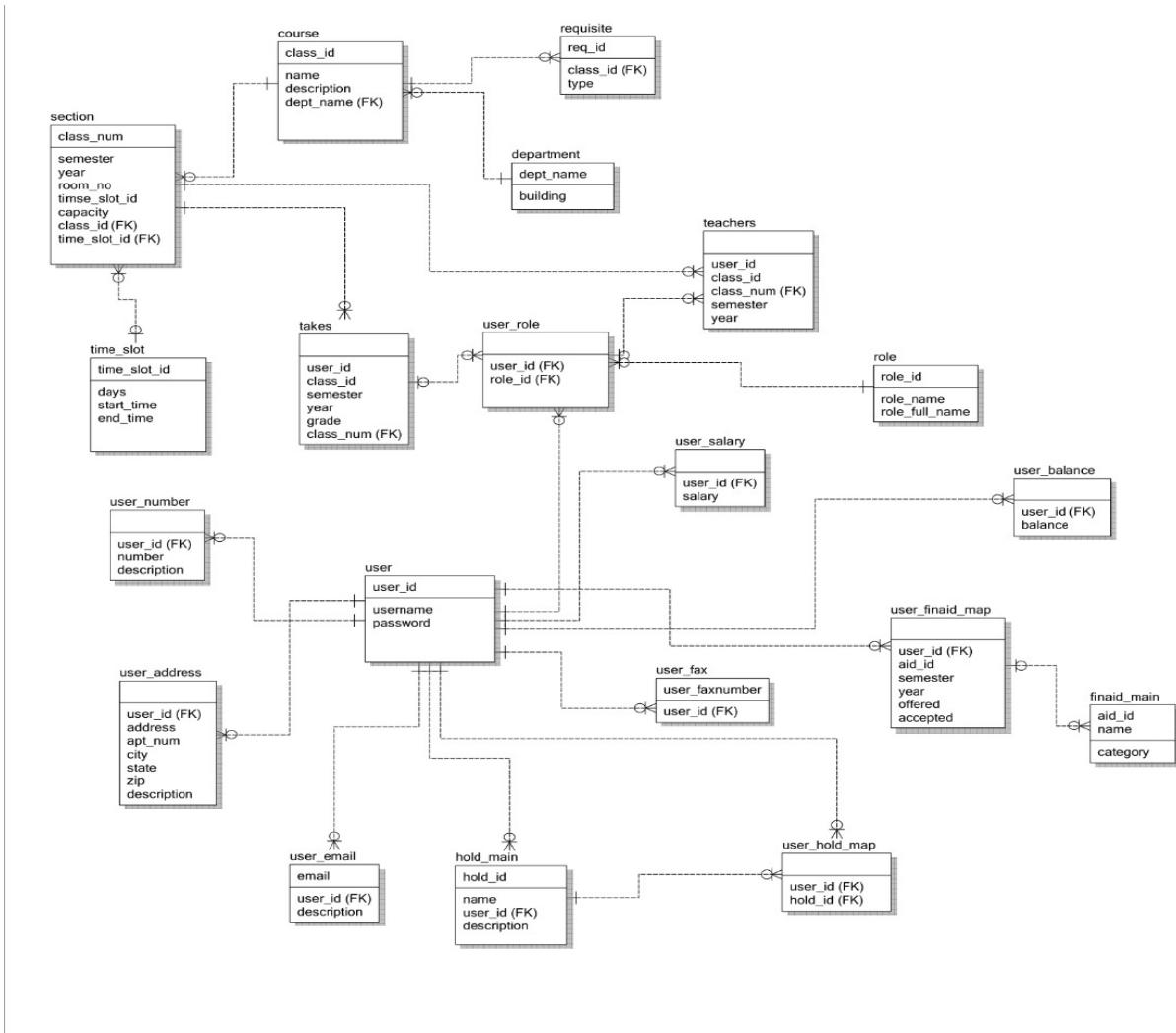


V. ENTITY RELATIONSHIP (ER) DIAGRAM BY USING ERWIN IDEF1X NOTATIONS

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 :

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the database is set to 'Mysql@127.0.0.1:3306'. The left sidebar shows the schema structure under 'SCHEMAS', including tables like 'user', 'course', and 'admins', and views like 'industry'. The main query editor window contains the following code and result:

```
1 • SELECT * FROM mydb.user;
```

user_id	username	password
1	selcan	123
2	mert	123
3	irem	123
4	sena	101112
5	nilvana	131415
*	HULL	HULL

The screenshot shows the MySQL Workbench interface. The database is still 'Mysql@127.0.0.1:3306'. The left sidebar shows the schema structure under 'SCHEMAS'. The main query editor window contains the following code and result:

```
1 • SELECT * FROM mydb.admins;
```

user_id	username
1	selcan
2	mert
3	irem

- **CREATE VIEW For ROLE:**

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'mydb' schema with its tables: course, department, finaid_main, hold_main, requisite, role, section, takes, teachers, time_slot, and user. The 'Tables' section is expanded. The main area is titled 'Query 1' and contains the following SQL code:

```
1 • SELECT * FROM mydb.role;
```

The result grid shows the following data:

role_id	role_name	role_full_name
1	H	lady
2	H	man
3	H	kid
4	S	animal
5	S	woman
*	NULL	NULL

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'mydb' schema with its tables. The 'Tables' section is expanded. The main area is titled 'Query 1' and contains the following SQL code:

```
1 • CREATE VIEW THEATER AS
2   SELECT role_id,role_full_name
3   FROM role
4   WHERE role_name = 'H' ;
5
6
```

- CREATE VIEW For FINAID_MAIN :

The screenshot shows the MySQL Workbench interface with a query editor window. The title bar says "finaid_main". The SQL pane contains the command:

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

The results pane displays a table with three columns: aid_id, name, and category. The data is as follows:

aid_id	name	category
1	help	university
2	help	college
3	success	university
4	success	college
HULL	HULL	HULL

The screenshot shows the MySQL Workbench interface with a query editor window. The title bar says "scholarship". The SQL pane contains the command:

```
1 • SELECT * FROM mydb.scholarship;
```

The results pane displays a table with two columns: aid_id and name. The data is as follows:

aid_id	name
2	help
4	success

- ALTER VIEW for USER:

The screenshot shows the MySQL Workbench interface with a query editor window. The title bar says "user". The SQL pane contains the command:

```
1 • SELECT * FROM mydb.user;
```

The results pane displays a table with three columns: user_id, username, and password. The data is as follows:

user_id	username	password
1	selcan	123
2	mert	123
3	ben	123
4	bené	301132
5	nilvana	135415
6	BBB	BBB

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas

Query 1 user admins

```
1 • SELECT * FROM mydb.admins;
```

user_id	username
4	sena

Result Grid Filter Rows: Export: Wrap Cell Content:

admins 1 x

MySQL Model (sorusuz mis proj... x EER Diagram x Mysql@127.0.0.1:3306 x

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas

Query 1 x admins user admins

```
1 • ALTER VIEW ADMINS AS
2     SELECT user_id,username
3     FROM user
4     WHERE password = "123456";
```

- **ALTER VIEW for ROLE:**

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas

Query 1 role

```
1 • SELECT * FROM mydb.role;
```

role_id	role_name	role_full_name
1	H	lady
2	H	man
3	H	kid
4	S	animal
5	S	woman

```

1 *  SELECT * FROM mydb.role;
+-----+-----+
| role_id | role_full_name |
+-----+-----+
|      4 |       admin      |
|      5 |       woman      |
+-----+-----+
2 * ALTER VIEW THEATER AS
3   SELECT role_id,role_full_name
4   FROM role
5   WHERE role_name = '1';
6

```

- **ALTER VIEW for FINAID_MAIN:**

```

1 *  SELECT * FROM mydb.finaid_main;
+-----+-----+-----+
| aid_id | name    | category |
+-----+-----+-----+
|      1 | help    | university |
|      2 | help    | college   |
|      3 | success | university |
|      4 | success | college   |
+-----+-----+-----+

```

The screenshot shows a MySQL Workbench interface with a result grid. The grid has two columns: 'category' and 'aid_id'. There are two rows of data:

category	aid_id
university	1
college	2

A.2 CREATE/ALTER INDEX EXAMPLES

- CREATE INDEX for USER:

The screenshot shows a MySQL Workbench interface with a query editor and a result grid.

Query Editor:

```

ALTER VIEW scholarship AS
SELECT category,aid_id
FROM finaid_main
WHERE name= "help";

```

Result Grid:

category	aid_id
university	1
college	2

- CREATE INDEX for ROLE:

The screenshot shows a MySQL Workbench interface with a query editor and a result grid.

Query Editor:

```

CREATE INDEX role_full_name ON role(role_full_name);
SHOW INDEXES FROM role;

```

Result Grid:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
role	0	PRIMARY	1	role_id	A	0	NULL	NULL	NULL	BTREE		YES	NULL	
role	0	PRIMARY	2	role_name	A	0	NULL	NULL	NULL	BTREE		YES	NULL	
role	1	role_full_name	1	role_full_name	A	5	NULL	NULL	NULL	BTREE		YES	NULL	

- **CREATE INDEX** for FINAID_MAIN :

```

Local instance MySQL80 ×
Edit View Query Database Server Tools Scripting Help
File Navigator Query Editor Database Browser Results
Result Grid Filter Rows: Export: Wrap Cell Content: 
1 CREATE INDEX taking_aid ON finaid_main(aid_id);
2
3 • SHOW INDEXES FROM finaid_main;
4

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:
Table Non_unique Key_name Seq_in_index Column_name Collation Cardinality Sub_part Packed Null Index_type Comment Index_comment Visible Expression
finaid_main 0 PRIMARY 1 aid_id A 1 NULL NULL BTREE YES
finaid_main 0 PRIMARY 2 name A 1 NULL NULL BTREE YES
finaid_main 1 finaidtakers 1 aid_id A 4 NULL NULL BTREE YES
finaid_main 1 taking_aid aid_id A 4 NULL NULL BTREE YES

Result 1 × Read Only

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

Before ALTER INDEX

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
File Navigator Query Editor Database Browser Results
Query 1 ×
Schemas mydb sakila sys world
SHOW INDEXES FROM user
1
2

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:
Table Non_unique Key_name Seq_in_index Column_name Collation Cardinality Sub_part Packed Null Index_type Comment
user 0 PRIMARY 1 user_id A 0 NULL NULL BTREE
user 0 PRIMARY 2 username A 0 NULL NULL BTREE
user 1 username 1 username A 5 NULL NULL BTREE
user 1 usernames 1 username A 5 NULL NULL BTREE

After ALTER INDEX

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
File Navigator Query Editor Database Browser Results
Query 1 ×
Schemas mydb sakila sys world
DROP INDEX usernames ON mydb.user;
1
2
3 • SHOW INDEXES FROM user

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:
Table Non_unique Key_name Seq_in_index Column_name Collation Cardinality Sub_part Packed Null Index_type Comment
user 0 PRIMARY 1 user_id A 0 NULL NULL BTREE
user 0 PRIMARY 2 username A 0 NULL NULL BTREE
user 1 username 1 username A 5 NULL NULL BTREE

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

Before ALTER INDEX

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: mydb

Query 1: SHOW INDEXES FROM role

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment
role	0	PRIMARY	1	role_id	A	0	NULL	NULL	NULL	BTREE	
role	0	PRIMARY	2	role_name	A	0	NULL	NULL	NULL	BTREE	
role	1	role_full_name	1	role_full_name	A	5	NULL	NULL	NULL	BTREE	

After ALTER INDEX

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: mydb

Query 1: SHOW INDEXES FROM role;

2. DROP INDEX role_full_name ON mydb.role;

3. SHOW INDEXES FROM role;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment
role	0	PRIMARY	1	role_id	A	0	NULL	NULL	NULL	BTREE	
role	0	PRIMARY	2	role_name	A	0	NULL	NULL	NULL	BTREE	

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

Before ALTER INDEX

```

Local instance MySQL80 ×
Edit View Query Database Server Tools Scripting Help
File | Favorites | Home | Database | Scripts | Tools | Help | 
indexes*
File | Favorites | Home | Database | Scripts | Tools | Help | 
Limit to 1000 rows | Filter Rows: | Export: | Wrap Cell Content: 
1 CREATE INDEX taking_aid ON finaid_main(aid_id);
2
3 • SHOW INDEXES FROM finaid_main;
4

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
finaid_main	0	PRIMARY	1	aid_id	A	1	HULL	HULL		BTREE			YES	NULL
finaid_main	0	PRIMARY	2	name	A	1	HULL	HULL		BTREE			YES	NULL
finaid_main	1	finaidtakers	1	aid_id	A	4	HULL	HULL		BTREE			YES	NULL
finaid_main	1	taking_aid	1	aid_id	A	4	HULL	HULL		BTREE			YES	NULL

Result 1 × Read Only

After ALTER INDEX

```

SQL File 4* ×
File | Favorites | Home | Database | Scripts | Tools | Help | 
Limit to 1000 rows | Filter Rows: | Export: | Wrap Cell Content: 
1 DROP INDEX taking_aid ON mydb.finaid_main
2

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
finaid_main	0	PRIMARY	1	aid_id	A	1	HULL	HULL		BTREE			YES	NULL
finaid_main	0	PRIMARY	2	name	A	1	HULL	HULL		BTREE			YES	NULL
finaid_main	1	finaidtakers	1	aid_id	A	4	HULL	HULL		BTREE			YES	NULL

Result 2 × Read Only

B.DML STATEMENTS -DATA MANIPULATION LANGUAGE

B.1. INSERT INTO STATEMENTS (OPEN ENDED)

For USER:

The screenshot shows two separate MySQL Workbench sessions, each with its own tab labeled "insert posts*".

Session 1 (Top):

```
1 • SELECT * from user;
2
3 • INSERT INTO user (user_id, username, password)
4   values (45,"nilkarabulut",72349189);
5
6 • INSERT INTO user (user_id, username, password)
7   values (35,"iremboyaci",882216);
8
```

Session 2 (Bottom):

```
3 • INSERT INTO user (user_id, username, password)
4   values (45,"nilkarabulut",72349189);
5
6 • INSERT INTO user (user_id, username, password)
7   values (35,"iremboyaci",882216);
8
9 • SELECT * from user;
```

In both sessions, the "Result Grid" pane displays the following data:

user_id	username	password
14	muratulker	12345
27	merthirkat7	1234
35	iremboyaci	882216
45	nilkarabulut	72349189
NULL	NULL	NULL

For ROLE :

The screenshot shows a single MySQL Workbench session with a tab labeled "Local instance MySQL80*".

Query 1:

```
1 • SELECT * FROM mydb.role;
```

Result Grid:

role_id	role_name	role_full_name
1	H	lady
2	H	man
3	H	kid
4	S	animal
5	S	woman
NULL	NULL	NULL

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator Query 1 role role SQL File 2*

SCHEMAS Filter objects

- Tables
 - course
 - department
 - finaid_main
 - hold_main
 - requisite
 - role
 - section
 - takes
 - teachers
 - time_slot
 - user
 - user_address
 - user_balance
 - user_email
 - user_fax
 - user_finaid_map
 - user_hold_map
 - user_number

Administration Schemas

```

1 • INSERT INTO `mydb`.`role` ('role_id', 'role_name', 'role_full_name') VALUES ('6', 'H', 'wolf');
2 • INSERT INTO `mydb`.`role` ('role_id', 'role_name', 'role_full_name') VALUES ('7', 'S', 'younger');
3 • INSERT INTO `mydb`.`role` ('role_id', 'role_name', 'role_full_name') VALUES ('8', 'M', 'adult');

```

File Edit View Query Database Server Tools Scripting Help

Navigator Query 1 role role SQL File 2* role

SCHEMAS Filter objects

- Tables
 - course
 - department
 - finaid_main
 - hold_main
 - requisite
 - role
 - section
 - takes
 - teachers
 - time_slot
 - user
 - user_address
 - user_balance
 - user_email
 - user_fax
 - user_finaid_map
 - user_hold_map
 - user_number

Administration Schemas

```

1 • 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
	6	H	wolf
	7	S	younger
	8	M	adult
	HULL	HULL	HULL

For DEPARTMENT:

user department

File Edit View Query Database Server Tools Scripting Help

Navigator Query 1 department department SQL File 2*

SCHEMAS Filter objects

- Tables
 - building
 - dept_name
 - engineering
 - industry

Administration Schemas

```

1 • SELECT * FROM mydb.department;
2 •

```

Result Grid | Filter Rows: [] | Edit: [] | Export/Import: [] | Wrap Cell Content: []

	building	dept_name
▶	engineering	industry
*	HULL	HULL

department 1

Apply Revert

user department

```

1 • SELECT * FROM mydb.department;
2
3 • INSERT INTO `mydb`.`department` (`building`, `dept_name`) VALUES ('lab', 'bioengineering');
4
5 • SELECT * FROM mydb.department;

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Apply | Revert |

building	dept_name
lab	bioengineering
engineering	industry
NULL	NULL

department 2 × | Apply | Revert |

For FINAID_MAIN:

user finaid_main

```

1 • SELECT * FROM mydb.finaid_main;
2
3
4
5

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Apply | Revert |

aid_id	name	category
1	help	university
2	help	college
3	success	university
4	success	college
NULL	NULL	NULL

finaid_main 5 × | Apply | Revert |

user department

```

1 • SELECT * FROM mydb.finaid_main;
2
3 • INSERT INTO `mydb`.`finaid_main` (`aid_id`, `name`, `category`) VALUES ('5', 'help', 'primary');
4 • INSERT INTO `mydb`.`finaid_main` (`aid_id`, `name`, `category`) VALUES ('6', 'success', 'primary');
5
6 • SELECT * FROM mydb.finaid_main;
7
8

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Apply | Revert |

aid_id	name	category
1	help	university
2	help	college
3	success	university
4	success	college
5	help	primary
6	success	primary
NULL	NULL	NULL

finaid_main 6 × | Apply | Revert |

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

For USER:

The image shows two separate sessions in MySQL Workbench.

Session 1 (user 5):

```

1 • SELECT * from mydb.user;
2
3
4
5

```

Result Grid:

user_id	username	password
14	muratulker	12345
35	iremboyao	882216
45	nilkarabulut	72349189
NULL	NULL	NULL

Session 2 (user 6):

```

1 • SELECT * from mydb.user;
2
3 • UPDATE `mydb`.`user` SET `password` = 'e4373186db721779b5b1cc95f7' WHERE (`user_id` = '14') and (`username` = 'muratulker');
4
5 • SELECT * FROM mydb.user

```

Result Grid:

user_id	username	password
14	muratulker	e4373186db721779b5b1cc95f7
35	iremboyao	882216
45	nilkarabulut	72349189
NULL	NULL	NULL

For ROLE :

The image shows a single session in MySQL Workbench.

Query 1 (role):

```

1 • SELECT * FROM mydb.role;

```

Result Grid:

role_id	role_name	role_full_name
1	H	lady
2	H	man
3	H	kid
4	S	animal
5	S	woman
NULL	NULL	NULL

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Query 1 × role

```

1 • UPDATE `mydb`.`role` SET `role_name` = 'M' WHERE (`role_id` = '4') and (`role_name` = 'S');
2 • UPDATE `mydb`.`role` SET `role_name` = 'M' WHERE (`role_id` = '5') and (`role_name` = 'S');

```

SCHEMAS

Tables

- course
- department
- finaid_main
- hold_main
- requisite
- role
- section
- takes
- teachers
- time_slot
- user
- user_address
- user_balance
- user_email
- user_fax
- user_finaid_map
- user_hold_map
- user_numbers

Administration Schemas

Information

File Edit View Query Database Server Tools Scripting Help

Navigator

Query 1 × role

```
1 • SELECT * FROM mydb.role;
```

SCHEMAS

Tables

- course
- department
- finaid_main
- hold_main
- requisite
- role
- section
- takes
- teachers
- time_slot
- user
- user_address
- user_balance
- user_email
- user_fax
- user_finaid_map
- user_hold_map
- user_numbers

Administration Schemas

Information

role 1 ×

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	M	animal
5	M	woman
*	NULL	NULL

Result Grid Form Editor

Apply Revert

For DEPARTMENT:

department ×

File Edit View Query Database Server Tools Scripting Help

Navigator

Query 1 × department

```
1 • SELECT * FROM mydb.department;
```

SCHEMAS

Tables

- building
- dept_name

department 1 ×

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

building	dept_name
27	sporsalonu
*	NULL

Result Grid Form Editor

Field Types

Query Stats

Apply Revert

department

```

1 • SELECT * FROM mydb.department;
2 • UPDATE `mydb`.`department` SET `building` = 'engineering', `dept_name` = 'industry' WHERE (`dept_name` = 'sporsalonu');
3

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Query Stats | Apply | Revert | C

building	dept_name
engineering	industry
NULL	NULL

department.1 ×

For FINAID_MAIN :

user department

```

1 • SELECT * FROM mydb.finaid_main;
2
3
4
5

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Apply | Revert | C

aid_id	name	category
1	help	university
2	help	college
3	success	university
4	success	college
5	help	primary
6	success	primary
NULL	NULL	NULL

finaid_main 7 ×

user department

```

1 • SELECT * FROM mydb.finaid_main;
2
3 • UPDATE `mydb`.`finaid_main` SET `category` = 'preschool' WHERE (`aid_id` = '5') and (`name` = 'help');
4 • UPDATE `mydb`.`finaid_main` SET `category` = 'preschool' WHERE (`aid_id` = '6') and (`name` = 'success');
5
6 • SELECT * FROM mydb.finaid_main;
7

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Apply | Revert | C

aid_id	name	category
1	help	university
2	help	college
3	success	university
4	success	college
5	help	preschool
6	success	preschool
NULL	NULL	NULL

finaid_main 8 ×

B.3. DELETE STATEMENTS (OPEN ENDED)

For USER:

The image shows three separate MySQL Workbench sessions, each titled "user".

Session 1: Contains the following SQL statements:

```
1 • SELECT * FROM mydb.user;
2
3 • DELETE FROM user where password=1234 or 12345
4
```

Session 2: Shows the result grid after the delete operation:

user_id	username	password
14	muratulker	12345
27	metirthka	1234
35	iremboyacı	882216
45	nillkarabulut	72349189
*	NULL	NULL

Session 3: Contains the following SQL statements:

```
1 • SELECT * FROM mydb.user;
2
3 • DELETE FROM user WHERE password =1234;
4
5 • SELECT*from mydb.user;
```

Session 4: Shows the result grid after the delete operation:

user_id	username	password
14	muratulker	12345
35	iremboyacı	882216
45	nillkarabulut	72349189
*	NULL	NULL

For ROLE:

The image shows the MySQL Workbench interface with the "role" table selected in the Navigator.

Table Structure:

```
Tables
  course
  department
  finald_main
  hold_main
  requisite
  role
  section
  takes
  teachers
  time_slot
  user
  user_address
  user_balance
  user_email
  user_fax
  user_finaid_map
  user_hold_id_map
  user_number
```

Session 1: Contains the following SQL statement:

```
1 • SELECT * FROM mydb.role;
```

Result Grid:

role_id	role_name	role_full_name
1	H	lady
2	H	man
3	H	kid
4	S	animal
5	S	woman
6	H	wolf
7	S	younger
8	M	adult
*	NULL	NULL

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator role role SQL File 2* role SQL File 3* role

SCHEMAS

Tables course department finaid_main hold_main requisite role section taken teachers time_slot user user_address user_balance user_email user_fax user_finaid_map user_hold_map user_number

Administration Schemas

Query 1 role role SQL File 2* role SQL File 3* role

```

1 • SELECT * FROM role;
2 • DELETE FROM role WHERE role_id = 7;
3 • SELECT * FROM mydb.role;

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor

role_id	role_name	role_full_name
1	H	lady
2	H	man
3	H	kid
4	S	animal
5	S	woman
6	H	wolf
8	M	adult
*	NULL	NULL

For DEPARTMENT:

User department ×

File Edit View Query Database Server Tools Scripting Help

department 4 ×

```

1 • SELECT * FROM mydb.department;
2
3 • DELETE FROM department WHERE building = "lab";
4
5 • SELECT * FROM mydb.department;
6
7

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Field Types

building	dept_name
engineering	industry
*	NULL

For FINAID_MAIN:

File Edit View Query Database Server Tools Scripting Help

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Field Types

aid_id	name	category
1	help	university
2	help	college
3	success	university
4	success	college
5	help	preschool
6	success	preschool
*	NULL	NULL

The screenshot shows the MySQL Workbench interface. At the top, there is a query editor window titled 'user' with a tab 'department'. The query is:

```

1 • SELECT * FROM mydb.finaid_main;
2
3 • DELETE from finaid_main WHERE category="preschool";
4
5 • SELECT * FROM mydb.finaid_main;

```

Below the query editor is a results grid window titled 'finaid_main 10'. It displays the following data:

aid_id	name	category
1	help	university
2	help	college
3	success	university
4	success	college
•	ROLE	ROLE

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

For USER:

The screenshot shows the MySQL Workbench interface with the 'Local instance MySQL80' connection selected. In the left sidebar, under 'Schemas', the 'mydb' schema is selected. In the 'Stored Procedures' section, a procedure named 'getrole' is listed. The main area shows the SQL code for the procedure:

```

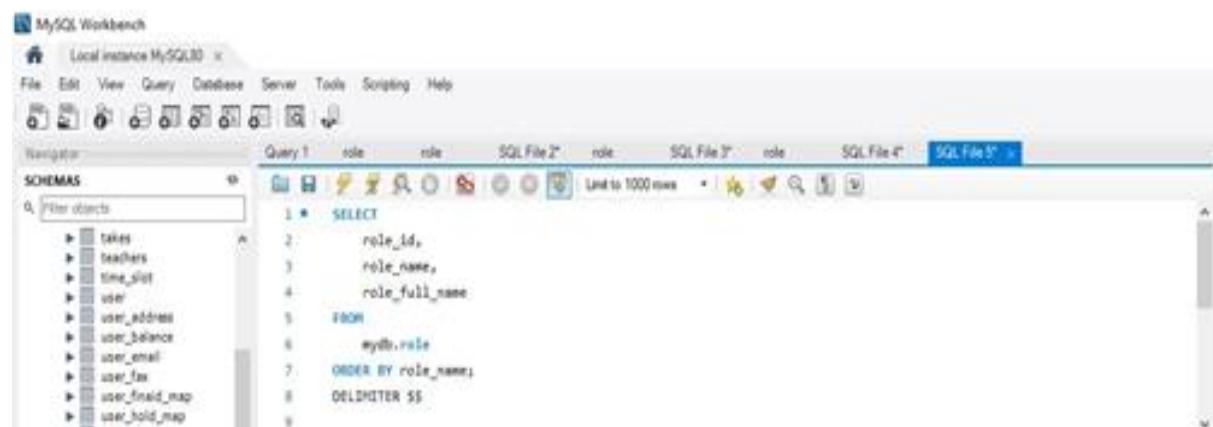
1 • SELECT
2     user_id,
3     username,
4     password
5   FROM
6     mydb.user
7   ORDER BY username
8   DELIMITER ;;
9 • CREATE PROCEDURE getuser()
10 BEGIN
11     SELECT
12         user_id,
13         username,
14         password
15     FROM
16         mydb.user
17     ORDER BY username;
18 END
19 DELIMITER ;
20

```

The screenshot shows the MySQL Workbench interface with the 'Local instance MySQL80' connection selected. In the left sidebar, under 'Schemas', the 'mydb' schema is selected. In the 'Tables' section, the 'user' table is listed. Below it, a results grid window titled 'user' displays the following data:

user_id	username	password
1	ish	123
2	net	123
3	rhuu	12345
4	alan	123
5	era	12345

For ROLE:



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas

Query 1 role role SQL File 2* role SQL File 3* role SQL File 4* SQL File 5*

```
1 * SELECT
  2   role_id,
  3   role_name,
  4   role_full_name
  5   FROM
  6     mydb.role
  7   ORDER BY role_name;
  8 DELIMITER ;;
  9
```



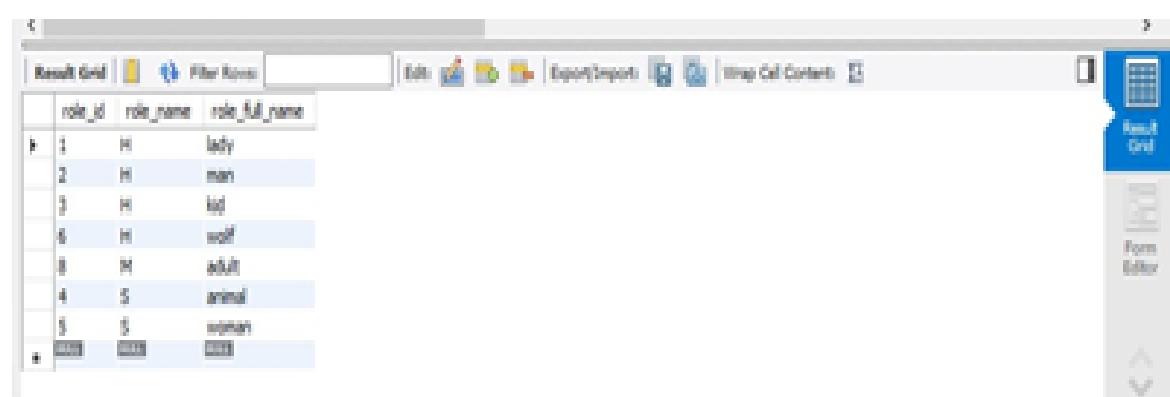
MySQL80 X

File Database Server Tools Scripting Help

Navigator Schemas

Query 1 role role SQL File 2* role SQL File 3* role SQL File 4* SQL File 5*

```
10 * CREATE PROCEDURE getrole()
11 BEGIN
12   SELECT
13     role_id,
14     role_name,
15     role_full_name
16   FROM
17     mydb.role
18   ORDER BY role_name;
```



Result Grid | Filter Rows | Edit | Export/Import | Wrap Cell Content |

	role_id	role_name	role_full_name
1	1	lady	
2	1	man	
3	1	id	
4	1	wolf	
5	2	adult	
6	2	animal	
7	2	asym	
8	2	ccc	

For FINAID_MAIN:

```
SQL File 5* ×
SELECT * FROM mydb.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 ;
```

Result Grid | Filter Rows: [] | Edit: [] | Export/Import: [] | Wrap Cell Content: []

aid_id	name	category
1	help	university
2	help	college
3	success	university
4	success	college
NULL	NULL	NULL

Result Grid | Form Editor | Field Types

```
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;
```

Result Grid | Filter Rows: [] | Export: [] | Wrap Cell Content: []

AID_NAME
Success

Result 14 X

Result Grid | Form Editor | Field Types

Read Only

C. DQL STATEMENTS -DATA QUERY LANGUAGE

C.1. SIMPLE JOINS - JOINING TWO TABLES

- User and role are joined.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** mydb
- Tables:** role
- Columns:** role_id (int PK), role_name (varchar(3)), role_full_name (varchar(15))
- Query Editor:** Contains the SQL query: `SELECT username, password, role_name, role_full_name FROM mydb.user JOIN mydb.role ON mydb.user.user_id = mydb.role.role_id;`
- Result Grid:** Displays the joined data from both tables. The columns are username, password, role_name, and role_full_name. The data includes rows for users like ivana, sara, irem, mert, selcan, and rihane, each associated with a specific role (lady, man, kid, animal) and its full name.

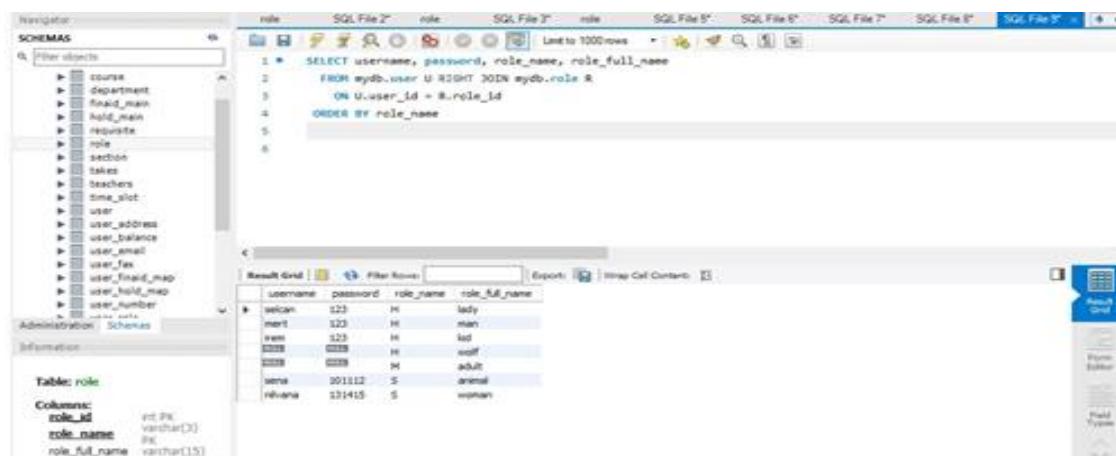
C.2. LEFT/RIGHT INNER/OUTER JOINS

- LEFT JOINS:

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** mydb
- Tables:** role
- Columns:** role_id (int PK), role_name (varchar(3)), role_full_name (varchar(15))
- Query Editor:** Contains the SQL query: `SELECT username, password, role_name, role_full_name FROM mydb.user U LEFT JOIN mydb.role R ON R.role_id = U.user_id; ORDER BY username;`
- Result Grid:** Displays the joined data. The columns are username, password, role_name, and role_full_name. The data includes rows for users like ivana, sara, irem, mert, selcan, and rihane, each associated with a specific role (lady, man, kid, animal) and its full name. Note that the result grid shows all users from the user table, even if they don't have a corresponding role entry.

- **RIGHT JOINS:**



The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query is:

```

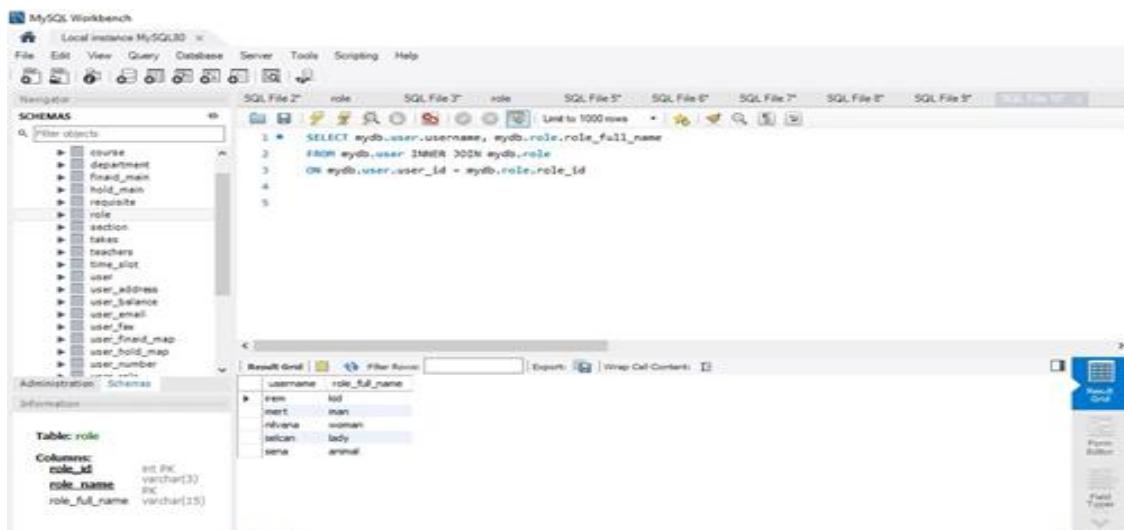
1 * SELECT username, password, role_name, role_full_name
2   FROM mydb.user U RIGHT JOIN mydb.role R
3     ON U.user_id = R.role_id
4
5
6

```

The results grid displays the following data:

username	password	role_name	role_full_name
selen	123	H	lady
mert	123	H	man
irem	123	H	kid
senan	00000	H	staff
nilvana	00000	H	adult
sena	101112	S	animal
nilvana	131415	S	woman

- **INNER JOINS:**



The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query is:

```

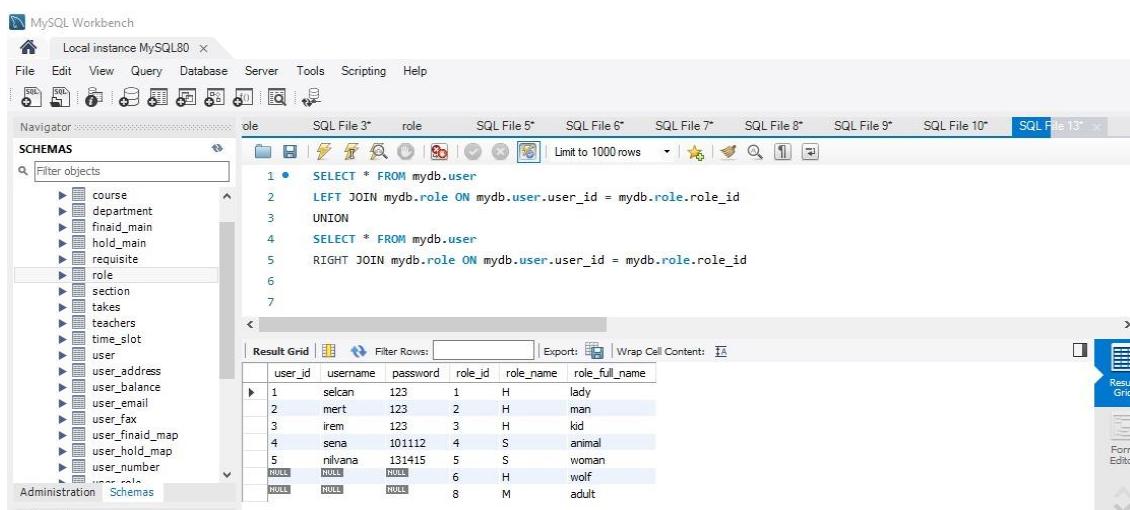
1 * SELECT mydb.user.username, mydb.role.role_full_name
2   FROM mydb.user INNER JOIN mydb.role
3     ON mydb.user.user_id = mydb.role.role_id
4
5

```

The results grid displays the following data:

username	role_full_name
irem	lady
mert	man
nilvana	woman
selcan	lady
sena	animal

- **OUTER JOINS:**



The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query consists of two parts:

```

1 * SELECT * FROM mydb.user
2   LEFT JOIN mydb.role ON mydb.user.user_id = mydb.role.role_id
3
4 UNION
5
6 * SELECT * FROM mydb.user
7   RIGHT JOIN mydb.role ON mydb.user.user_id = mydb.role.role_id
8
9

```

The results grid displays the following data:

user_id	username	password	role_id	role_name	role_full_name
1	selcan	123	1	H	lady
2	mert	123	2	H	man
3	irem	123	3	H	kid
4	sena	101112	4	S	animal
5	nilvana	131415	5	S	woman
NULL	NULL	NULL	6	H	wolf
NULL	NULL	NULL	8	M	adult

C.3. NESTED QUERIES

For ROLE:

The screenshot shows two separate sessions in MySQL Workbench, both targeting the 'mydb' database.

Session 1 (Top):

- SQL Editor:** Contains a series of SQL statements for updating the 'role_salary' column based on 'role_id' and 'role_name', followed by an INSERT statement:


```

1 • ALTER TABLE mydb.role
2     ADD role_salary INT;
3
4 • UPDATE `mydb`.`role` SET `role_salary` = '1500' WHERE (`role_id` = '1') and (`role_name` = 'H');
5 • UPDATE `mydb`.`role` SET `role_salary` = '1200' WHERE (`role_id` = '2') and (`role_name` = 'H');
6 • UPDATE `mydb`.`role` SET `role_salary` = '1400' WHERE (`role_id` = '3') and (`role_name` = 'H');
7 • UPDATE `mydb`.`role` SET `role_salary` = '1600' WHERE (`role_id` = '4') and (`role_name` = 'S');
8 • UPDATE `mydb`.`role` SET `role_salary` = '1800' WHERE (`role_id` = '5') and (`role_name` = 'S');
9 • UPDATE `mydb`.`role` SET `role_salary` = '1750' WHERE (`role_id` = '6') and (`role_name` = 'H');
10 • UPDATE `mydb`.`role` SET `role_salary` = '2000' WHERE (`role_id` = '8') and (`role_name` = 'M');
11 • INSERT INTO `mydb`.`role` ('role_id', `role_name`, `role_full_name`, `role_salary`) VALUES ('9', 'S', 'wife', '1500'
12
            
```
- Result Grid:** Displays the updated data in the 'role' table:

role_id	role_name	role_full_name	role_salary
1	H	lady	1500
2	H	man	1200
3	H	kid	1400
4	S	animal	1600
5	S	woman	1800
6	H	wolf	1750
8	M	adult	2000
9	S	wife	1500
*	NUL	NUL	NUL

Session 2 (Bottom):

- SQL Editor:** Contains a series of SQL statements for updating the 'role_salary' column based on 'role_id' and 'role_name', followed by a SELECT query:


```

9 • UPDATE `mydb`.`role` SET `role_salary` = '1750' WHERE (`role_id` = '6') and (`role_name` = 'H');
10 • UPDATE `mydb`.`role` SET `role_salary` = '2000' WHERE (`role_id` = '8') and (`role_name` = 'M');
11 • INSERT INTO `mydb`.`role` ('role_id', `role_name`, `role_full_name`, `role_salary`) VALUES ('9', 'S', 'wife', '1500'
12
13 • SELECT role_id,role_name,role_full_name,role_salary
14     FROM mydb.role
15     WHERE role_salary > 1500
16
17
18
19
20
            
```
- Result Grid:** Displays the data from the 'role' table where 'role_salary' is greater than 1500:

role_id	role_name	role_full_name	role_salary
4	S	animal	1600
5	S	woman	1800
6	H	wolf	1750
8	M	adult	2000
*	NUL	NUL	NUL

For USER :

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons for file operations, database navigation, and search.
- Navigator:** Shows the schema structure under "mydb".
- SQL Editor:** Contains the following SQL code:

```
1 * ALTER TABLE mydb.user
2   ADD login_attempt INT;
3 *
4 * UPDATE `mydb`.`user` SET `login_attempt` = '3' WHERE (`user_id` = '1') AND (`username` = 'salman');
5 * UPDATE `mydb`.`user` SET `login_attempt` = '4' WHERE (`user_id` = '2') AND (`username` = 'mert');
6 * UPDATE `mydb`.`user` SET `login_attempt` = '3' WHERE (`user_id` = '3') AND (`username` = 'irfan');
7 * UPDATE `mydb`.`user` SET `login_attempt` = '6' WHERE (`user_id` = '3') AND (`username` = 'sena');
8 * UPDATE `mydb`.`user` SET `login_attempt` = '2' WHERE (`user_id` = '5') AND (`username` = 'nilvana');
9 * INSERT INTO `mydb`.`user`(`login_attempt`) VALUES ('4');
```
- Result Grid:** Displays the data from the user table, showing the results of the update operations.

	user_id	username	password	login_attempt
1	1	salman	123	3
2	2	mert	123	4
3	3	irfan	123	3
4	3	sena	101112	6
5	5	nilvana	131415	2
6	nilvana	nilvana	nilvana	nilvana

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons for file operations, database navigation, and search.
- Navigator:** Shows the schema structure under "mydb".
- SQL Editor:** Contains the following SQL code:

```
5 * UPDATE `mydb`.`user` SET `login_attempt` = '4' WHERE (`user_id` = '2') AND (`username` = 'mert');
6 * UPDATE `mydb`.`user` SET `login_attempt` = '3' WHERE (`user_id` = '3') AND (`username` = 'irfan');
7 * UPDATE `mydb`.`user` SET `login_attempt` = '6' WHERE (`user_id` = '3') AND (`username` = 'sena');
8 * UPDATE `mydb`.`user` SET `login_attempt` = '2' WHERE (`user_id` = '5') AND (`username` = 'nilvana');
9 *
10 * SELECT user_id,username,password,login_attempt
11 FROM mydb.user
12 WHERE login_attempt <= 4;
```
- Result Grid:** Displays the data from the user table, showing the results of the query.

	user_id	username	password	login_attempt
1	2	mert	123	4
2	3	irfan	131415	2
3	nilvana	nilvana	nilvana	nilvana

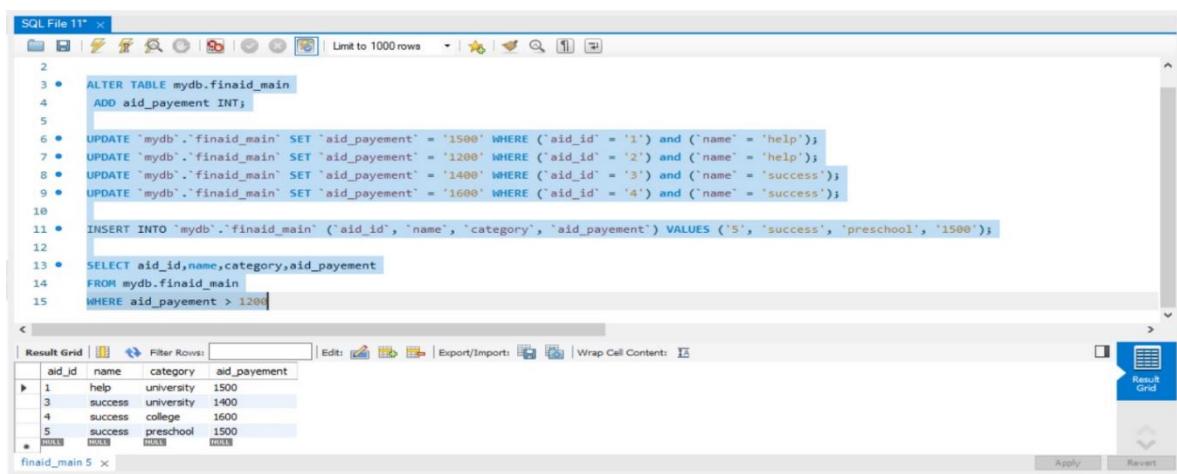
The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons for file operations, database navigation, and search.
- Navigator:** Shows the schema structure under "mydb".
- SQL Editor:** Contains the following SQL code:

```
5 * UPDATE `mydb`.`user` SET `login_attempt` = '4' WHERE (`user_id` = '2') AND (`username` = 'mert');
6 * UPDATE `mydb`.`user` SET `login_attempt` = '3' WHERE (`user_id` = '3') AND (`username` = 'irfan');
7 * UPDATE `mydb`.`user` SET `login_attempt` = '6' WHERE (`user_id` = '3') AND (`username` = 'sena');
8 * UPDATE `mydb`.`user` SET `login_attempt` = '2' WHERE (`user_id` = '5') AND (`username` = 'nilvana');
9 *
10 * SELECT user_id,username,password,login_attempt
11 FROM mydb.user
12 WHERE login_attempt <= 4;
```
- Result Grid:** Displays the data from the user table, showing the results of the query.

	user_id	username	password	login_attempt
1	2	mert	123	4
2	3	irfan	131415	2
3	nilvana	nilvana	nilvana	nilvana

For FINAID_MAIN:



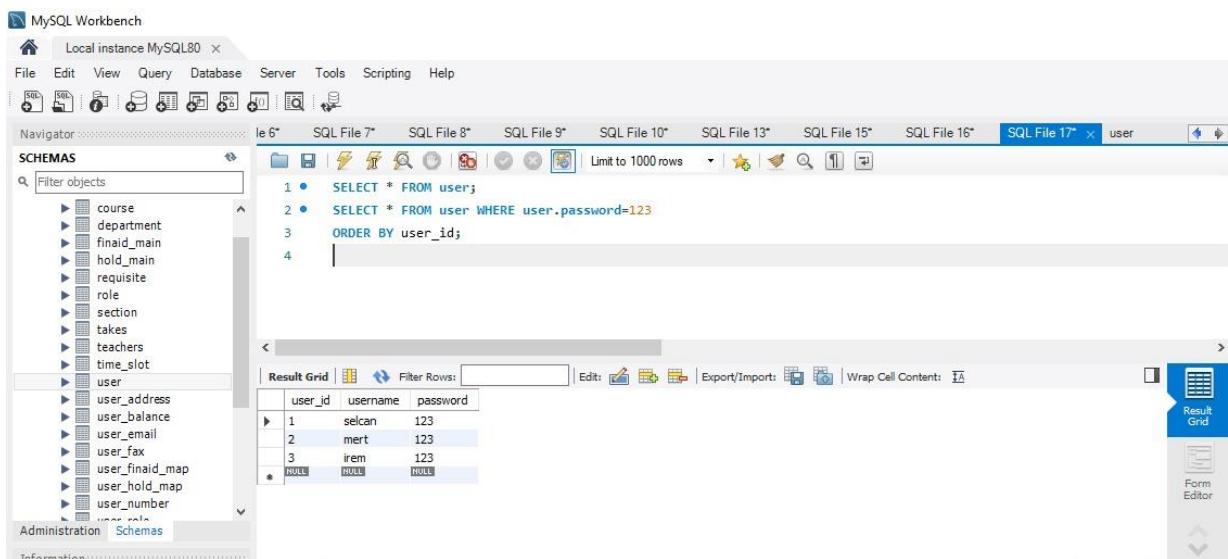
```
SQL File 11* ×
ALTER TABLE mydb.finaid_main
ADD aid_payment INT;
UPDATE `mydb`.`finaid_main` SET `aid_payment` = '1500' WHERE (`aid_id` = '1') and (`name` = 'help');
UPDATE `mydb`.`finaid_main` SET `aid_payment` = '1200' WHERE (`aid_id` = '2') and (`name` = 'help');
UPDATE `mydb`.`finaid_main` SET `aid_payment` = '1400' WHERE (`aid_id` = '3') and (`name` = 'success');
UPDATE `mydb`.`finaid_main` SET `aid_payment` = '1600' WHERE (`aid_id` = '4') and (`name` = 'success');
INSERT INTO `mydb`.`finaid_main` (`aid_id`, `name`, `category`, `aid_payment`) VALUES ('5', 'success', 'preschool', '1500');
SELECT aid_id, name, category, aid_payment
FROM mydb.finaid_main
WHERE aid_payment > 1200;
```

The screenshot shows the MySQL Workbench interface with a SQL editor window titled "SQL File 11*". The code in the editor is used to alter the "finaid_main" table to add a new column "aid_payment" of type INT. It then performs four UPDATE statements to set the value of "aid_payment" for specific rows based on "aid_id" and "name". Finally, it inserts a new row with aid_id 5, name "success", category "preschool", and aid_payment 1500. A SELECT statement is run to retrieve all rows where aid_payment is greater than 1200. The results are displayed in a "Result Grid" table:

aid_id	name	category	aid_payment
1	help	university	1500
3	success	university	1400
4	success	college	1600
5	success	preschool	1500
*	NULL	NULL	NULL

C.4. SORTING SUCH AS ORDER BY STATEMENTS:

For USER:



The screenshot shows the MySQL Workbench interface with a SQL editor window titled "SQL File 17*". The code in the editor is used to select all columns from the "user" table, filter rows where user.password is 123, and sort the results by user_id. The results are displayed in a "Result Grid" table:

user_id	username	password
1	selcan	123
2	mert	123
3	irem	123
*	NULL	NULL

For ROLE:

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL80' is selected. The 'Schemas' tab is open, displaying a tree view of database objects including 'course', 'department', 'finaid_main', 'hold_main', 'requisite', 'role', 'section', 'takes', 'teachers', 'time_slot', and 'user'. A query editor window titled 'SQL File 18*' contains the following SQL code:

```
1 • SELECT * FROM role;
2 • SELECT * FROM role WHERE role_name='S'
3 ORDER BY role_full_name;
4
5
```

The results grid shows the following data:

role_id	role_name	role_full_name
4	S	animal
5	S	woman
*	NULL	NULL

For FINAID_MAIN:

The screenshot shows three separate MySQL Workbench windows for the 'finaid_main' table.

Top Window: Contains the following SQL code:

```
1
2 • SELECT * FROM finaid_main;
3
4 • SELECT * FROM finaid_main WHERE finaid_main.name="help"
5 ORDER BY aid_id;
6
```

Middle Window: Shows the results grid for the first query:

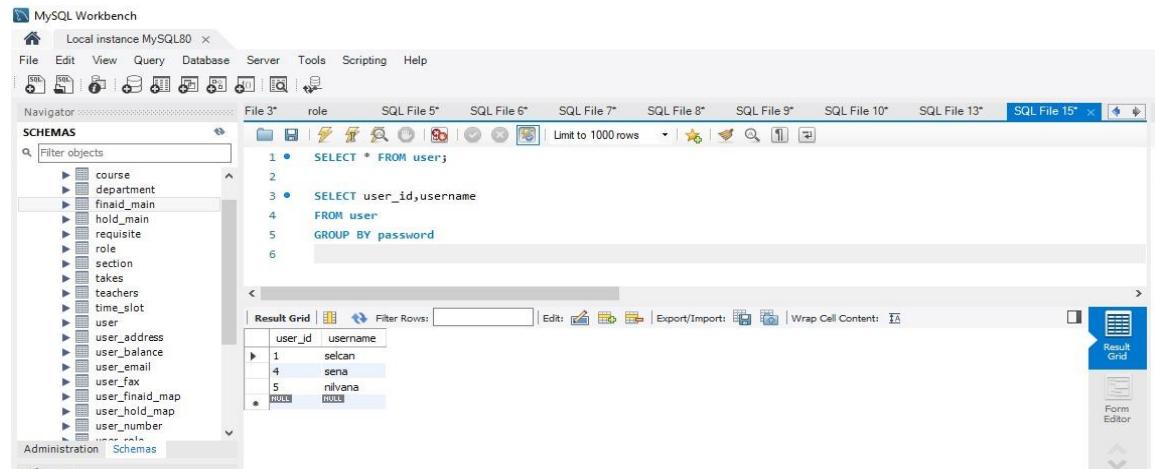
aid_id	name	category
1	help	university
2	help	college
3	success	university
4	success	college
*	NULL	NULL

Bottom Window: Shows the results grid for the second query:

aid_id	name	category
1	help	university
2	help	college
*	NULL	NULL

C.5. GROUP BY AND AGGREGATION FUNCTIONS

For USER:



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas

Filter objects

SCHEMAS

- course
- department
- finaid_main
- hold_main
- requisite
- role
- section
- takes
- teachers
- time_slot
- user
- user_address
- user_balance
- user_email
- user_fax
- user_finaid_map
- user_hold_map
- user_number

Administration Schemas

SQL File 3* role SQL File 5* SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10* SQL File 13* SQL File 15* SQL File 16*

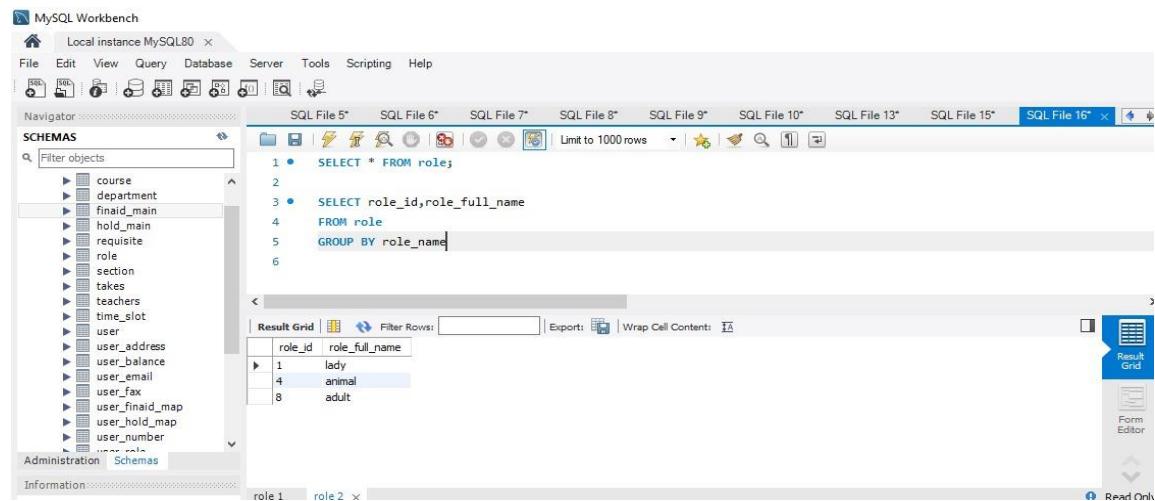
```
1 • SELECT * FROM user;
2
3 • SELECT user_id,username
4   FROM user
5   GROUP BY password
6
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

user_id	username
1	selcan
4	sena
5	nivana
*	NULL
	ROLE

Result Grid Form Editor

For ROLE:



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas

Filter objects

SCHEMAS

- course
- department
- finaid_main
- hold_main
- requisite
- role
- section
- takes
- teachers
- time_slot
- user
- user_address
- user_balance
- user_email
- user_fax
- user_finaid_map
- user_hold_map
- user_number

Administration Schemas

Information

SQL File 5* SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10* SQL File 13* SQL File 15* SQL File 16*

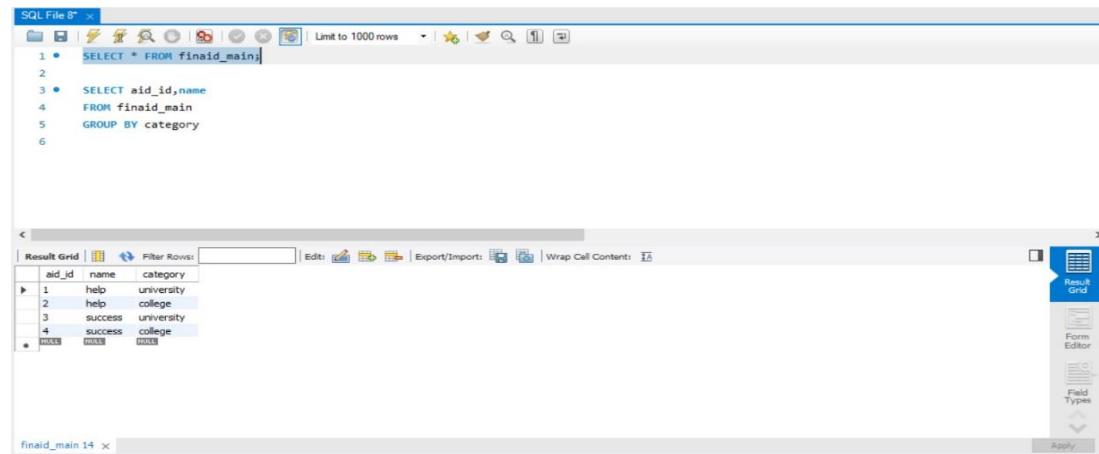
```
1 • SELECT * FROM role;
2
3 • SELECT role_id,role_full_name
4   FROM role
5   GROUP BY role_name
6
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

role_id	role_full_name
1	lady
4	animal
8	adult

Result Grid Form Editor

For FINAID_MAIN:



SQL File 8*

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas

Filter objects

SCHEMAS

- course
- department
- finaid_main
- hold_main
- requisite
- role
- section
- takes
- teachers
- time_slot
- user
- user_address
- user_balance
- user_email
- user_fax
- user_finaid_map
- user_hold_map
- user_number

Administration Schemas

SQL File 8*

```
1 • SELECT * FROM finaid_main;
2
3 • SELECT aid_id,name
4   FROM finaid_main
5   GROUP BY category
6
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

aid_id	name	category
1	help	university
2	help	college
3	success	university
4	success	college
*	NULL	NULL
	NULL	NULL

Result Grid Form Editor Field Types

SQL File 8*

```

1 • SELECT * FROM finaid_main;
2
3 • SELECT aid_id, name
4   FROM finaid_main
5   GROUP BY category
6

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

aid_id	name
1	help
2	help
•	NULL

finaid_main 15 x

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

For USER:

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas SQL File 9* SQL File 10* SQL File 13* SQL File 15* SQL File 16* SQL File 17* SQL File 18* SQL File 19* x user

filter objects

SELECT*FROM user

DELIMITER \$\$

DROP PROCEDURE IF EXISTS `mydb`.`getuserr` \$\$

CREATE PROCEDURE `mydb`.`getuserr`
(getuserr_username VARCHAR(16), OUT getuserr_password VARCHAR(40))
BEGIN
SELECT user.password getuserr_password,user.username getuserr_username
FROM user WHERE user.username =getuserr_username;
END \$\$

DELIMITER ;

SET @getuserr_username="irem";

CALL getuserr(@getuserr_username,@getuser_password)

No object selected

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

getuserr_password	getuserr_username
123	irem

For ROLE:

The screenshot shows the MySQL Workbench interface with the 'SQL File 9*' tab active. The code in the editor is as follows:

```

1 SELECT*FROM role
2
3 DELIMITER $$ 
4
5 • DROP PROCEDURE IF EXISTS `mydb`.`getrole` $$ 
6
7 • CREATE PROCEDURE `mydb`.`getrole`(
8     IN @ROLE_NAME VARCHAR(5), OUT ROLE_FULL_NAME VARCHAR(15))
9 BEGIN
10     SELECT role.role_full_name ROLE_FULL_NAME,role.role_name ROLE_NAME
11     FROM role WHERE role.role_name =@ROLE_NAME;
12 END $$ 
13
14 DELIMITER ;
15
16 • SET @ROLE_NAME= "H";
17 • CALL getrole(@ROLE_NAME,@ROLE_FULL_NAME)

```

The results grid shows the data from the 'role' table:

ROLE_FULL_NAME	ROLE_NAME
lady	H
man	H
kid	H
wolf	H

For FINAID_MAIN:

The screenshot shows the MySQL Workbench interface with the 'SQL File 8*' tab active. The code in the editor is as follows:

```

1 SELECT*FROM finaid_main
2
3 DELIMITER $$ 
4
5 • DROP PROCEDURE IF EXISTS `mydb`.`aid_procedure1` $$ 
6
7 • CREATE PROCEDURE `mydb`.`aid_procedure1`(
8     IN AID_NAME VARCHAR(50), OUT AID_CATEGORY VARCHAR(15))
9 BEGIN
10     SELECT finaid_main.category AID_CATEGORY
11     FROM finaid_main

```

The results grid shows the data from the 'finaid_main' table:

aid_id	name	category
1	help	university
2	help	college
3	success	university
4	success	college
•	NULL	NULL

The screenshot shows the MySQL Workbench interface with the 'SQL File 8*' tab active. The code in the editor is as follows:

```

1 SELECT*FROM finaid_main
2
3 DELIMITER $$ 
4
5 • DROP PROCEDURE IF EXISTS `mydb`.`aid_procedure1` $$ 
6
7 • CREATE PROCEDURE `mydb`.`aid_procedure1`(
8     IN AID_NAME VARCHAR(50), OUT AID_CATEGORY VARCHAR(15))
9 BEGIN
10     SELECT finaid_main.category AID_CATEGORY
11     FROM finaid_main
12     WHERE finaid_main.name =AID_NAME;
13 END $$ 
14
15 DELIMITER ;
16
17 • SET @AID_NAME="success";
18 • CALL aid_procedure1(@AID_NAME,@AID_CATEGORY)

```

Result Grid | Filter Rows: Export: Wrap Cell Content:

AID_CATEGORY

university
college

Result 18 X Read Only

D. DCL STATEMENTS -DATA CONTROL LANGUAGE

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

For FINAID_MAIN:

SQL File 9* | Limit to 1000 rows

```

1 • SELECT * FROM finaid_main;
2
3 ○ /*RollBack is a command in SQL which is used to undo the changes.
4 And after each statement, SAVEPOINT is created so that we can Rollback to that point and whatever changes were after that SAVEPOINT are undone.
5 For ex:
6 In the given SQL statements.
7 Insertion of rows into CUSTOMERS table is done.
8 Then SAVEPOINT Insert_Done is created.
9 After that, Delete statements is written which would have deleted some rows from the table.
10 SAVEPOINT Delete_Done is created.
11 Not UPDATE command updates/modifies one of the rows

```

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

aid_id	name	category
1	help	university
2	help	college
3	success	university
4	success	college
NULL	NULL	NULL

finaid_main 2 X Apply Revert

SQL File 9* | Limit to 1000 rows

```

1 • SELECT * FROM finaid_main;
2 ○ /*RollBack is a command in SQL which is used to undo the changes.
3 And after each statement, SAVEPOINT is created so that we can Rollback to that point and whatever changes were after that SAVEPOINT are undone.
4 For ex:
5 In the given SQL statements.
6 Insertion of rows into CUSTOMERS table is done.
7 Then SAVEPOINT Insert_Done is created.
8 After that, Delete statements is written which would have deleted some rows from the table.
9 SAVEPOINT Delete_Done is created.
10 Not UPDATE command updates/modifies one of the rows.
11 Since After UPDATE command changes are not committed, therefore it can be undone.
12 Now to undo this last query i.e., UPDATE statement:
13 The state of the data can be roll backed to the SAVEPOINT Delete_Done to ensures that UPDATE is not done and the system restores the previous state.*/
14
15 • UPDATE finaid_main SET aid_id=aid_id-1 where category="university";
16
17 • SELECT * FROM finaid_main
18

```

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

aid_id	name	category
0	help	university
2	help	college
2	success	university
4	success	college
NULL	NULL	NULL

finaid_main 9 X Apply Revert

For USER:

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas Administration Information

Table: role

Columns:

- role_id int PK
- role_name varchar(3) PK
- role_full_name varchar(15)

Result Grid:

user_id	username	password
1	selcon	123
2	mert	123
3	irem	123
3	sena	101112
5	nivalna	131415
*	NULL	NULL

For ROLE:

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas Administration Information

Table: role

Columns:

- role_id int PK
- role_name varchar(3) PK
- role_full_name varchar(15)

Result Grid:

role_id	role_name	role_full_name
1	H	lady
2	H	man
3	H	kid
4	S	animal
5	S	woman
6	H	wolf
8	M	adult
*	NULL	NULL

D.2. SECURITY FUNCTIONS (GRANT/REVOKE)

- GRANT

SQL File 10*

File Edit View Query Database Server Tools Scripting Help

Result Grid:

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...

- REVOKE

The screenshot shows the MySQL Workbench interface with a SQL editor window titled "SQL File 10". The SQL code entered is:

```
1 • SHOW GRANTS
2
3 • REVOKE SELECT ON finaid_main FROM root@localhost;
4
5 • SHOW GRANTS
6
7
```

Below the editor, the "Result Grid" tab is selected, displaying the grants for the user "root@localhost". The output is:

```
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...
```

The interface includes standard MySQL Workbench navigation and search tools at the top, and a sidebar on the right with icons for "Result Grid", "Form Editor", and "Read Only" mode.

The screenshot shows the MySQL Workbench interface with a SQL editor window titled "SQL File 10". The SQL code entered is:

```
1
2 • REVOKE SELECT ON finaid_main FROM root@localhost;
3
4 • SHOW GRANTS
5
6
7
```

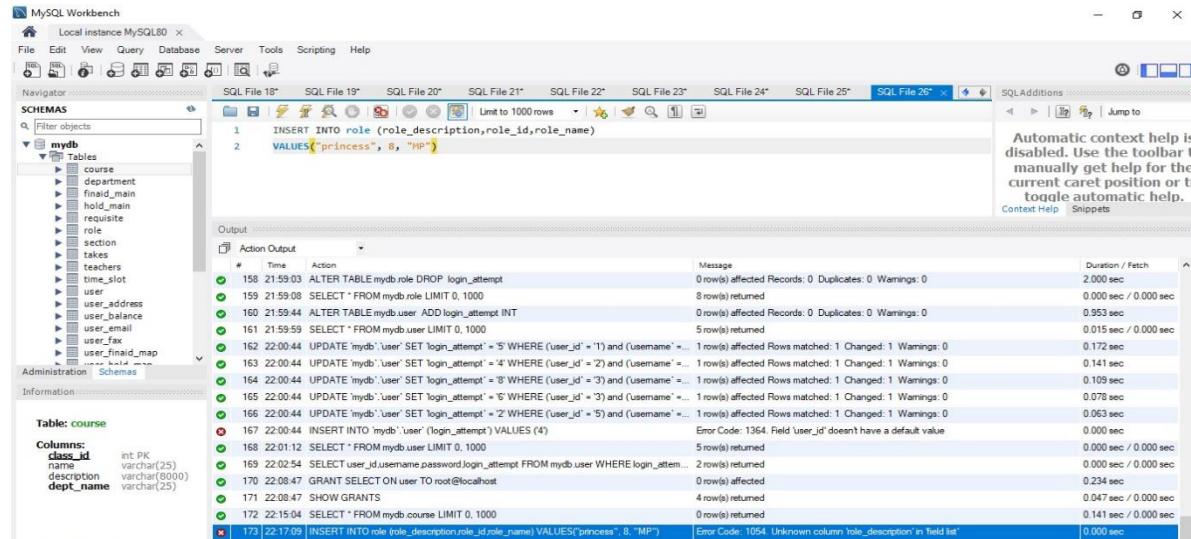
Below the editor, the "Result Grid" tab is selected, displaying the grants for the user "root@localhost". The output is:

```
Grants for root@localhost
▶ GRANT SELECT, INSERT, UPDATE, DELETE, CR...
GRANT APPLICATION_PASSWORD_ADMIN,AU...
GRANT PROXY ON '@' TO 'root'@localhost WI...
```

The interface includes standard MySQL Workbench navigation and search tools at the top, and a sidebar on the right with icons for "Result Grid", "Form Editor", and "Read Only" mode.

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'.

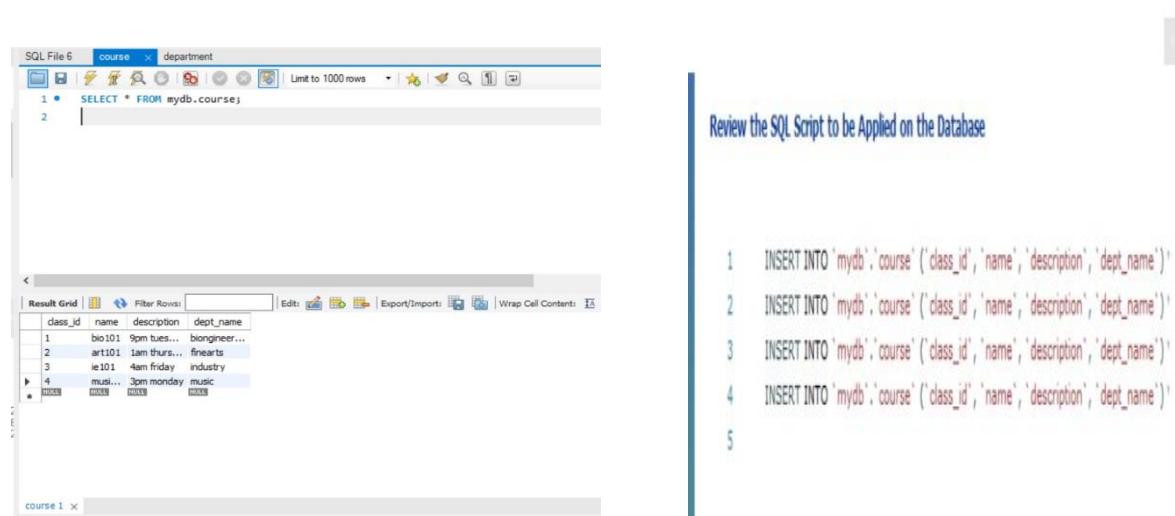


The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, there is a single line of SQL code:

```
1 INSERT INTO role (role_description,role_id,role_name)
2 VALUES('princess', 8, 'MP')
```

The second line contains a syntax error: 'Unknown column 'role_description' in 'field list''. The output window shows the execution of various database operations, including table creation, data insertion, and grants, followed by the error message at the bottom.

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



The screenshot shows the MySQL Workbench interface. The SQL Editor tab contains a query:

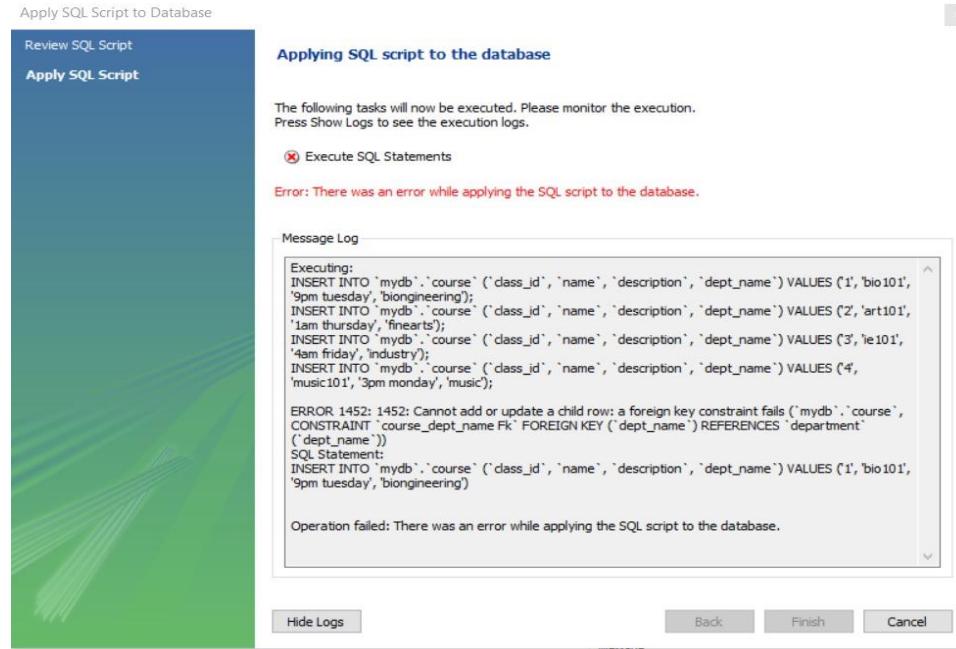
```
1 SELECT * FROM mydb.course;
```

The Result Grid shows the following data:

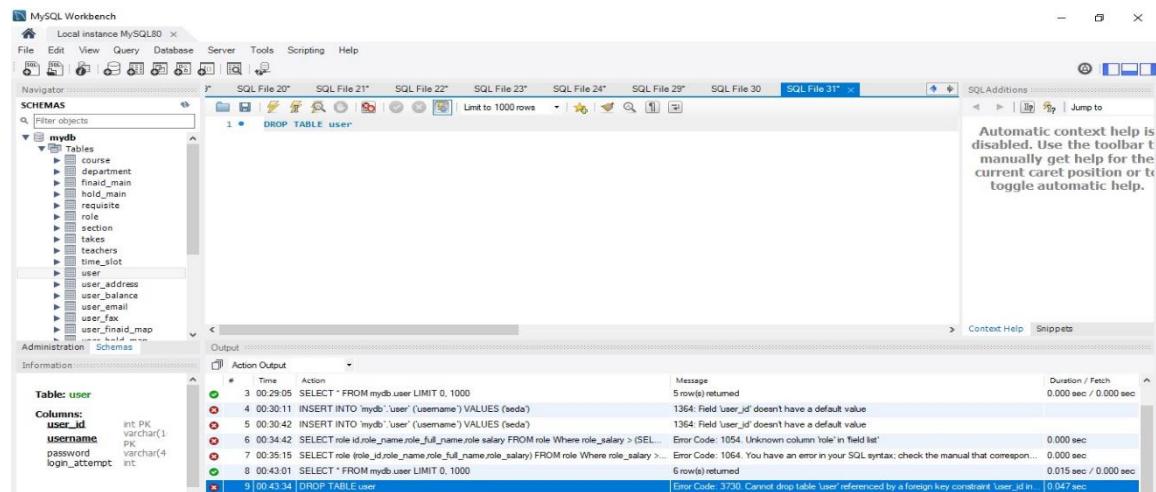
class_id	name	description	dept_name
1	bio101	9pm tues..	biogineer...
2	art101	1am thurs..	firearts
3	ie101	4pm friday	industry
4	mus101	3pm monday	music

A vertical bar on the right side of the interface has the text 'Review the SQL Script to be Applied on the Database'. Below this, a numbered list of SQL statements is shown, with the fourth statement highlighted in red:

- 1 `INSERT INTO 'mydb'.'course' ('class_id', 'name', 'description', 'dept_name')`
- 2 `INSERT INTO 'mydb'.'course' ('class_id', 'name', 'description', 'dept_name')`
- 3 `INSERT INTO 'mydb'.'course' ('class_id', 'name', 'description', 'dept_name')`
- 4 `INSERT INTO 'mydb'.'course' ('class_id', 'name', 'description', 'dept_name')`**
- 5



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 have 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)"

#	Time	Action	Message	Duration / Fetch
1	00:27:51	UPDATE takes SET user_id = 5 where user_id = 1	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.015 sec
2	00:28:21	SELECT * FROM mydb.takes LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
3	00:29:05	SELECT * FROM mydb.user LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
4	00:30:11	INSERT INTO mydb.`user`(`username`) VALUES ('seda')	1364: Field 'user_id' doesn't have a default value	
5	00:30:42	INSERT INTO mydb.`user`(`username`) VALUES ('seda')	1364: Field 'user_id' doesn't have a default value	
6	00:34:42	SELECT role_id,role_name,role_full_name,role_salary FROM role Where role_salary > (SELECT AVG(role_salary) FROM role)	Error Code: 1054. Unknown column 'role_id' in 'field list'	0.000 sec
7	00:35:15	SELECT role_id,role_name,role_full_name,role_salary FROM role Where role_salary > (SELECT AVG(role_salary) FROM role)	Error Code: 1064. 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 '(SELECT AVG(role_salary) FROM role)' at line 1	0.000 sec

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.

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas
mydb Tables
course department final_main hold_main requisite role section taken teachers time_slot user user_address user_balance user_email user_fax user_final_map user_hold_main
SQL File 20* SQL File 21* SQL File 22* SQL File 23* SQL File 24* SQL File 25* SQL File 30 SQL File 31* SQL File 32*
1 • DELETE FROM user
2 WHERE username= 'All'
3
4
Output
Action Output
# Time Action Message Duration / Fetch
5 00:30:42 INSERT INTO `mydb`.`user` ('username') VALUES ('seda')
1364: Field 'user_id' doesn't have a default value 0.000 sec
6 00:34:42 SELECT role_id,role_name,role_full_name,role_salary FROM role Where role_salary > (SEL... Error Code: 1054: Unknown column 'role' in 'field list' 0.000 sec
7 00:35:15 SELECT role_id,role_name,role_full_name,role_salary FROM role Where role_salary > ... Error Code: 1064: You have an error in your SQL syntax; check the manual that corresponds ... 0.000 sec
8 00:43:01 SELECT * FROM mydb.user LIMIT 0, 1000 6 row(s) returned 0.015 sec / 0.000 sec
9 00:43:34 DROP TABLE user Error Code: 3730: Cannot drop table 'user' referenced by a foreign key constraint 'user_id in ... 0.047 sec
10 00:43:44 SELECT * FROM mydb.user LIMIT 0, 1000 6 row(s) returned 0.000 sec / 0.000 sec
11 00:49:50 DELETE FROM user WHERE username= "All" 0 row(s) affected 0.000 sec
12 00:50:43 [DELETE FROM user WHERE username= 'All'] Error Code: 1175: 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' 0.031 sec

```

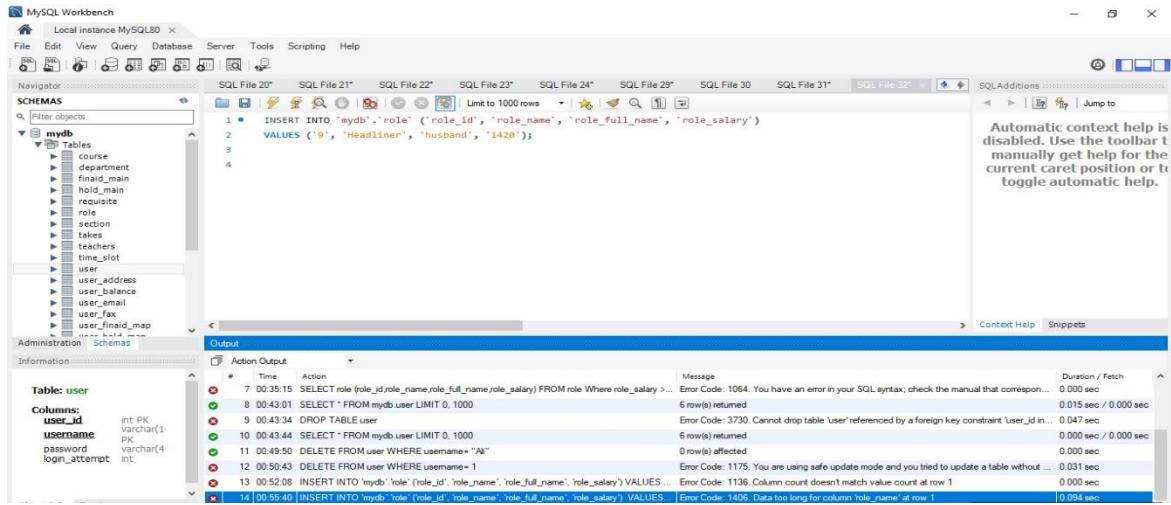
7) This script will not execute. Because, it gives error as "Column count doesn't match value count at row 1."

```

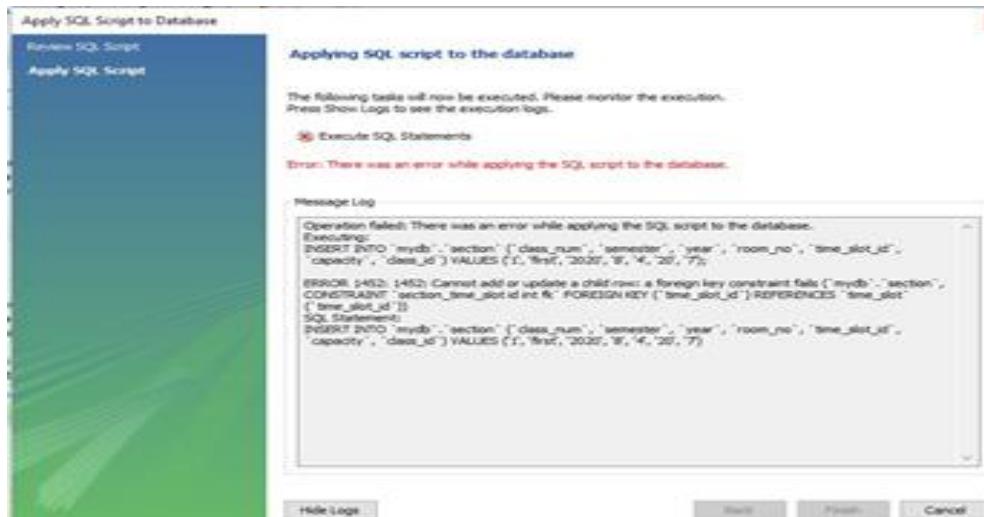
MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas
mydb Tables
course department final_main hold_main requisite role section taken teachers time_slot user user_address user_balance user_email user_fax user_final_map user_hold_main
SQL File 20* SQL File 21* SQL File 22* SQL File 23* SQL File 24* SQL File 25* SQL File 30 SQL File 31* SQL File 32*
1 • INSERT INTO `mydb`.`role` ('role_id', 'role_name', 'role_full_name', 'role_salary') VALUES ('9', 'S', 'wife', '1500', '9')
2
3
4
Output
Action Output
# Time Action Message Duration / Fetch
6 00:34:42 SELECT role_id,role_name,role_full_name,role_salary FROM role Where role_salary > (SEL... Error Code: 1054: Unknown column 'role' in 'field list' 0.000 sec
7 00:35:15 SELECT role_id,role_name,role_full_name,role_salary FROM role Where role_salary > ... Error Code: 1064: You have an error in your SQL syntax; check the manual that corresponds ... 0.000 sec
8 00:43:01 SELECT * FROM mydb.user LIMIT 0, 1000 6 row(s) returned 0.015 sec / 0.000 sec
9 00:43:34 DROP TABLE user Error Code: 3730: Cannot drop table 'user' referenced by a foreign key constraint 'user_id in ... 0.047 sec
10 00:43:44 SELECT * FROM mydb.user LIMIT 0, 1000 6 row(s) returned 0.000 sec / 0.000 sec
11 00:49:50 DELETE FROM user WHERE username= "All" 0 row(s) affected 0.000 sec
12 00:50:43 [DELETE FROM user WHERE username= 'All'] Error Code: 1175: 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' 0.031 sec
13 00:52:08 [INSERT INTO mydb.role ('role_id', 'role_name', 'role_full_name', 'role_salary') VALUES ('9', 'S', 'wife', '1500')] Error Code: 1136: Column count doesn't match value count at row 1 0.000 sec

```

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 varchar 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 constraint 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**

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject - main.py
pythonProject main.py
Project pythonProject C:
  1 import mysql.connector
  2
  3 cnx = mysql.connector.connect(user='root', password='1234',
  4                               host='127.0.0.1',
  5                               database="pydb")
  6
  7 cursor = cnx.cursor()
  8
  9 query = ("SELECT aid_id, name FROM finaid_main")
 10 cursor.execute(query)
 11
 12 myresult = cursor.fetchall()
 13
 14 for x in myresult:
 15     print(x)
 16
 17 cursor.close()
 18
 19 cnx.close()
```

Run: main

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

- **UPDATE**

The screenshot shows a PyCharm interface with a top navigation bar and two main panes. The top pane is a 'console' window titled 'main.py' with the following SQL code:

```
1 update finaid_main SET aid_payment=2000 where aid_id =3;
2
3 select aid_id, name, category, aid_payment
4 from finaid_main
```

The bottom pane is an 'Output' window titled 'mydb.finaid_main' showing the results of the query:

	aid_id	name	category	aid_payment
1	1	help	university	1500
2	2	help	college	1200
3	3	success	university	2000
4	4	success	college	1600
5	5	success	preschool	1500

- CREATE

```

1 import mysql.connector
2
3 cnx = mysql.connector.connect(user='root', password='1234',
4                               host='127.0.0.1',
5                               database="mydb")
6 cursor = cnx.cursor()
7
8 #Dropping EMPLOYEE table if already exists.
9 cursor.execute("DROP TABLE IF EXISTS STUDENT")
10
11 #Creating table as per requirement
12 sql ='''CREATE TABLE STUDENT(
13     FIRST_NAME CHAR(20) NOT NULL,
14     LAST_NAME CHAR(20) NOT NULL,
15     GRADE INT NOT NULL
16 )'''
17 cursor.execute(sql)
18
19 cnx.close()
20
21
22

```

The screenshot shows the MySQL Workbench interface. On the left, the Schema browser displays the 'mydb' schema with tables like 'student', 'takes', and 'user'. In the center, a query editor window titled 'student' contains the SQL command:

```
SELECT * FROM mydb.student;
```

Below the query editor is a 'Result Grid' showing the following data:

FIRST_NAME	LAST_NAME	GRADE

- DELETE

The screenshot shows the PyCharm interface with the 'main.py' file open. The code includes a section for deleting data from the 'finaid_main' table:

```

16     (8, 'help', 'college', 1600)
17 ]
18 cursor.executemany(sql, val)
19
20
21 cnx.commit()
22
23 print(cursor.rowcount, "was inserted.")
24 """
25
26 query = "DELETE FROM finaid_main WHERE aid_payment = '2000';"
27
28 cursor.execute(query)
29
30 cnx.commit()
31
32 print(cursor.rowcount, "record(s) deleted")
33
34 cnx.close()
35
36 cursor.close()
37

```

The 'Run' tab at the bottom shows the output of the script:

```
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

1 • SELECT * FROM mydb.finaid_main;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

aid_id name category aid_payment

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
*	NULL	NULL	NULL

aid_id = 3 is deleted

finaid_main 1 × Apply Revert C

- **INSERT**

thonProject main.py

```

1 import mysql.connector
2
3 cnx = mysql.connector.connect(user='root', password='1234',
4                               host='127.0.0.1',
5                               database='mydb')
6
7 cursor = cnx.cursor()
8
9 sql = "INSERT INTO finaid_main(aid_id, name, category, aid.payment) VALUES (%s, %s, %s, %s)"
10 val = [
11     (6, 'help', 'primary school', 1200),
12     (7, 'success', 'middle school', 1400),
13     (8, 'help', ' college', 1600)
14 ]
15
16 cursor.executemany(sql, val)
17
18 cnx.commit()
19
20 print(cursor.rowcount, "was inserted.")
21
22

```

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

user Administration - Users and Privil... finaid_main finaid_main finaid_main

1 • SELECT * FROM mydb.finaid_main;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

aid_id name category aid_payment

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
*	NULL	NULL	NULL

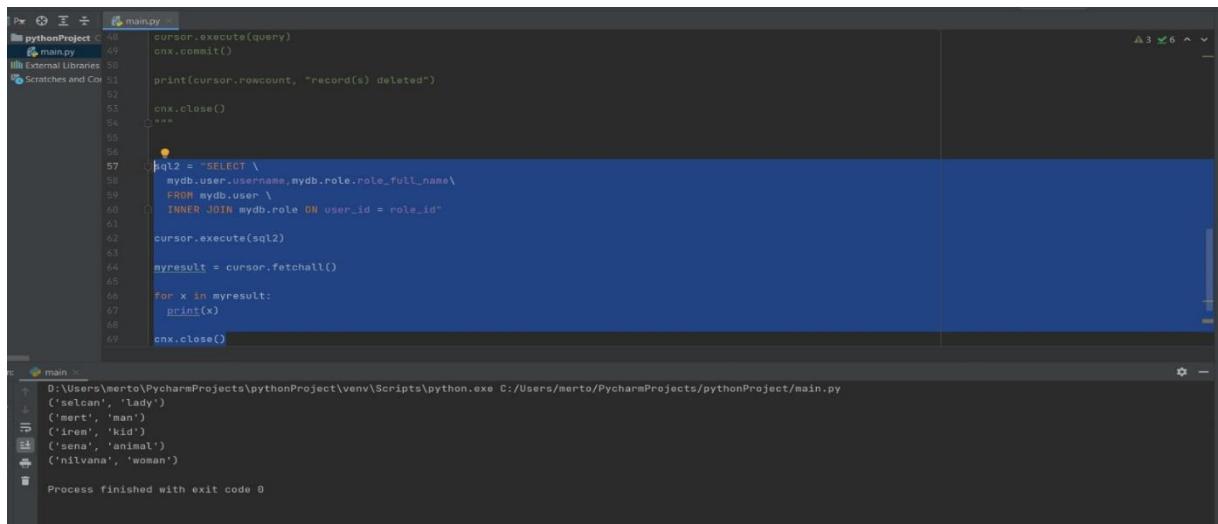
Result Grid

Form Editor

Field Types

Query Stats

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

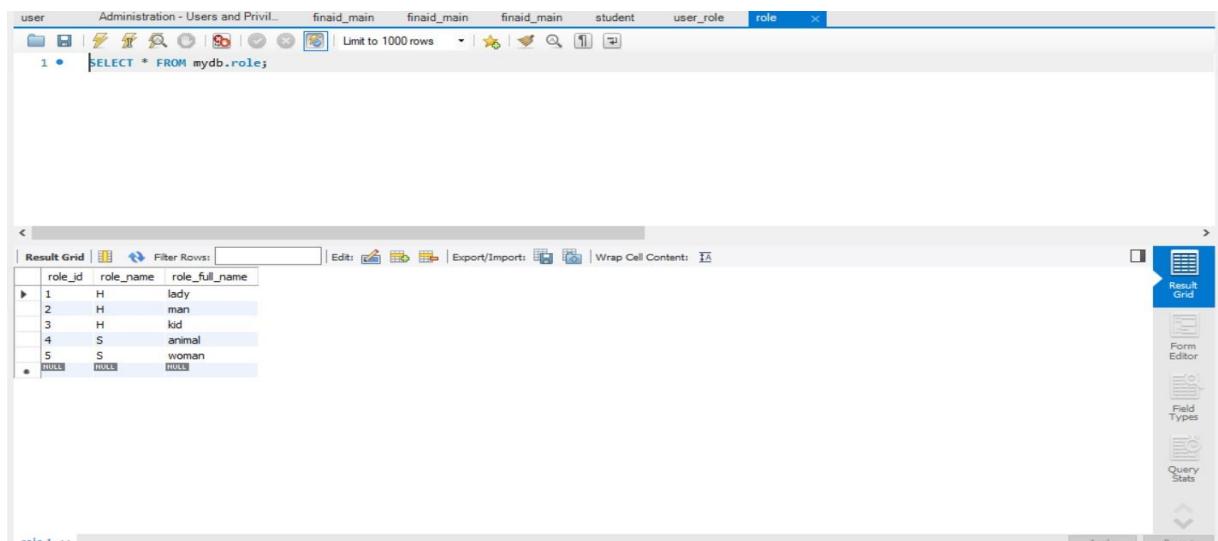


```

48     cursor.execute(query)
49
50     cnx.commit()
51
52     print(cursor.rowcount, "record(s) deleted")
53
54     """
55
56
57     sql2 = "SELECT \
58         mydb.user.username,mydb.role.role_full_name\
59     FROM mydb.user \
60     INNER JOIN mydb.role ON user_id = role_id"
61
62     cursor.execute(sql2)
63
64     myresult = cursor.fetchall()
65
66     for x in myresult:
67         print(x)
68
69     cnx.close()

```

The screenshot shows a PyCharm interface with a code editor and a terminal window. The code editor contains Python code that performs a database query. The terminal window shows the command run and the output of the query results.



Administration - Users and Privileges

Limit to 1000 rows

Result Grid

role_id	role_name	role_full_name
1	H	lady
2	H	man
3	H	kid
4	S	animal
5	S	women
NULL	NULL	NULL

Role

Result Grid

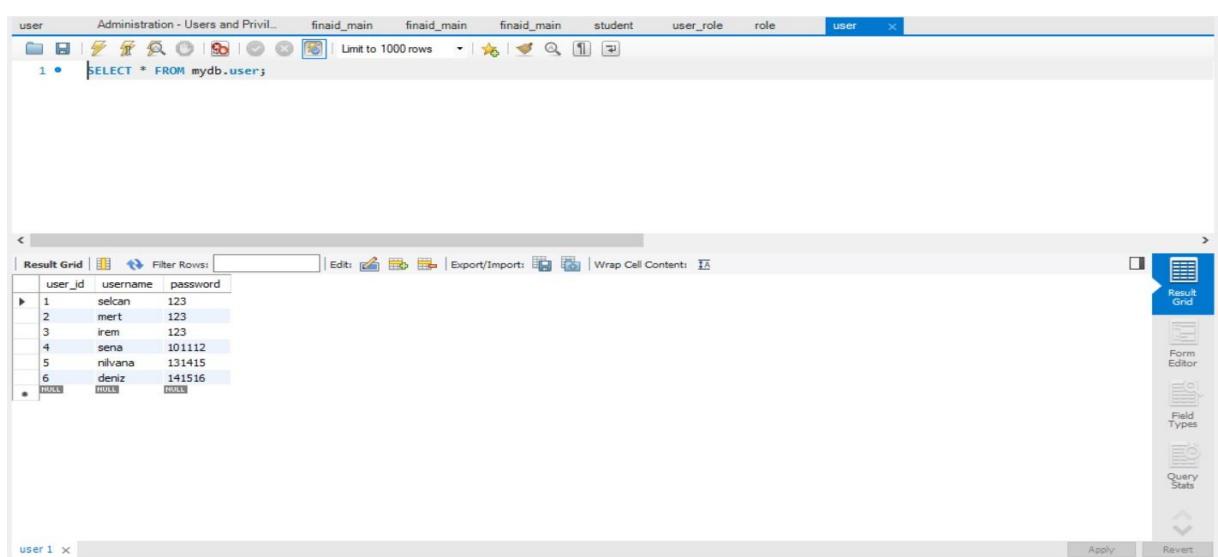
Form Editor

Field Types

Query Stats

Apply Revert

This screenshot shows the MySQL Workbench interface with the 'roles' table selected. The table contains five rows with columns: role_id, role_name, and role_full_name. The 'role' tab is selected at the top.



Administration - Users and Privileges

Limit to 1000 rows

Result Grid

user_id	username	password
1	selcan	123
2	mert	123
3	irem	123
4	sena	101112
5	nilvana	131415
6	deniz	141516
NULL	NULL	NULL

User

Result Grid

Form Editor

Field Types

Query Stats

Apply Revert

This screenshot shows the MySQL Workbench interface with the 'users' table selected. The table contains six rows with columns: user_id, username, and password. The 'user' tab is selected at the top.

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

- DATA MANIPULATION & STORED PROCEDURE

Screenshot of MySQL Workbench showing a query results grid and a Python script in PyCharm.

MySQL Workbench Results Grid:

role_id	role_name	role_full_name
1	H	lady
2	H	man
3	H	kid
4	S	animal
5	S	woman
6	S	man

PyCharm Script (main.py):

```

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

cursor = cnx.cursor()

#data manipulation
sql3 = "UPDATE role SET role_name = 'T' WHERE role_full_name = 'animal'"

cursor.execute(sql3)

cnx.commit()

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

cnx.close()

```

PyCharm Run Output:

```

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

Process finished with exit code 0

```

Screenshot of MySQL Workbench showing a query results grid with handwritten annotations.

MySQL Workbench Results Grid:

role_id	role_name	role_full_name
1	H	lady
2	H	man
3	H	kid
4	T	animal
5	S	woman
6	S	man

Handwritten Annotations:

Handwritten arrows and numbers are present in the bottom left corner of the results grid area, indicating specific rows or columns.

```

pythonProject pythonProject
  main.py
  deneme.py
  deneme.py
  External Libraries
  Scratches and Con
  Deneme

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

```

• INSERT

```

pythonProject main.py
  main.py
  deneme.py
  External Libraries
  Scratches and Con
  Deneme

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

```

1 • SELECT * FROM mydb.finaid_main;

aid_id	name	category	aid_payment
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

 The 'Result Grid' tab is selected on the right side of the interface.

- UPDATE

The screenshot shows the PyCharm IDE with two panes. The top pane is a console window displaying the following SQL code:

```

1 update finaid_main SET aid_payment=2000 where aid_id = 3;
2
3 select aid_id, name, category, aid_payment
4 from finaid_main
5

```

The bottom pane is a database browser window titled "mydb.finaid_main" showing the table structure and data. The table has columns: aid_id, name, category, and aid_payment. The data is as follows:

aid_id	name	category	aid_payment
1	help	university	1500
2	help	college	1200
3	success	university	2000
4	success	college	1600
5	success	preschool	1500

- DELETE

The screenshot shows the PyCharm IDE with a code editor and a terminal window. The code editor contains Python code that includes a DELETE statement:

```

16     (8, 'help', 'college', 1600)
17 ]
18 cursor.executemany(sql, val)
19
20 cnx.commit()
21
22 print(cursor.rowcount, "was inserted.")
23 """
24
25 query = "DELETE FROM finaid_main WHERE aid_payment = '2000';"
26
27 cursor.execute(query)
28 cnx.commit()
29
30 print(cursor.rowcount, "record(s) deleted")
31
32 cnx.close()
33
34 cursor.close()
35
36
37

```

The terminal window shows the output of the script execution:

```

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

```

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query editor contains:

```

1 • SELECT * FROM mydb.finaid_main;

```

The results grid shows the table "finaid_main" with the following data:

aid_id	name	category	aid_payment
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
NULL	NULL	NULL	NULL

A handwritten note in blue ink on the right side of the results grid says: "aid_id = 3 is deleted".

• FOR

```

main.py >
1 import mysql.connector
2
3 cnx = mysql.connector.connect(user='root', password='1234',
4                               host='127.0.0.1',
5                               database='mydb')
6
7 cursor = cnx.cursor()
8
9
10 # Using the cursor as iterator
11 cursor.execute("SELECT * FROM mydb.user")
12 for row in cursor:
13     print(row)
14
15
16 cnx.close()
17
18 """
19 import mysql.connector
20
21 cnx = mysql.connector.connect(user='root', password='1234',
22                               host='127.0.0.1',
23

```

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

```

tion - Users and Privil... finaid_main finaid_main student user_role role user course department requisite role user

1 • `SELECT * FROM mydb.user;`

user_id	username	password
1	selcan	123
2	mert	123
3	irem	123
4	sena	101112
5	nilvana	131415
6	deniz	141516
NULL	NULL	NULL

user 1 x

```

File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject - main.py
pythonProject main.py >
1 import mysql.connector
2
3 cnx = mysql.connector.connect(user='root', password='1234',
4                               host='127.0.0.1',
5                               database='mydb')
6
7 cursor = cnx.cursor()
8
9 query = ("SELECT sid_id, name FROM finaid_main")
10 cursor.execute(query)
11
12 myresult = cursor.fetchall()
13
14 for(x) in myresult:
15     print(x)
16
17 cursor.close()
18
19 cnx.close()

```

Run: main >

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

```

```

main.py
1 cursor.execute(query)
2 cnx.commit()
3 print(cursor.rowcount, "record(s) deleted")
4
5 """
6
7 sql2 = "SELECT \
8     mydb.user.username,mydb.role.role_full_name\
9     FROM mydb.user \
10    INNER JOIN mydb.role ON user_id = role_id"
11
12 cursor.execute(sql2)
13 myresult = cursor.fetchall()
14
15 for x in myresult:
16     print(x)
17
18 cnx.close()

```

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

username	role_full_name
'selcan'	'lady'
('mert')	('man')
('irem')	('kid')
('sema')	('animal')
('nilvana')	('woman')

Process finished with exit code 0

• WHILE

```

main.py
1 import mysql.connector
2
3 cnx = mysql.connector.connect(user='root',password='1234',
4                               host='127.0.0.1',
5                               database="mydb")
6
7 cursor = cnx.cursor()
8
9 cursor.execute("SELECT * FROM role")
10 row = cursor.fetchone()
11 while row is not None:
12     print(row)
13     row = cursor.fetchone()
14
15 cnx.close()
16
17 """
18 #CREATE TABLE
19 #Dropping EMPLOYEE table if already exists.
20 cursor.execute("DROP TABLE IF EXISTS STUDENT")
21
22

```

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

(1, 'H')	'Lady'
(2, 'M')	'man'
(3, 'K')	'kid'
(4, 'S')	'animal'
(5, 'W')	'woman'

Process finished with exit code 0

Administration - Users and Privileges

Limit to 1000 rows

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
NULL	NULL	NULL

- IF/THEN

```

1  cursor = cnx.cursor()
2
3  allpayments = "SELECT aid_id,aid_payment FROM finaid_main"
4  cursor.execute(allpayments)
5  myresult4=cursor.fetchall()
6  print(myresult4)
7
8  #IF STATEMENT
9  averageQuery= "SELECT ROUND(AVG(aid_payment)) FROM finaid_main"
10 cursor.execute(averageQuery)
11 myresult2 = cursor.fetchall()
12 print(myresult2)
13
14 insertQuery = " Select aid_id,aid_payment, " \
15             "IF(aid_payment>1471,'MORE_than_avg','LESS_than_avg') from finaid_main;"
16
17 cursor.execute(insertQuery)
18 myresult5 = cursor.fetchall()
19 print(myresult5)
20
21
22
23
24
25
26

```

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, 'MORE_than_avg')]
Process finished with exit code 0

```

1  import pymysql
2  from werkzeug.security import generate_password_hash, check_password_hash
3
4  try:
5      conn = pymysql.connect(host='localhost', database='mydb', user='root', password='1234')
6      cur = conn.cursor()
7      _hashed_password = generate_password_hash('secret')
8      cur.callproc('sp_createUser', ('Suumitra Roy', 'contact@roytuts.com', _hashed_password))
9      data = cur.fetchall()
10     if len(data) == 0:
11         conn.commit()
12         print("User information saved successfully !!")
13     else:
14         print("error: ", str(data[0]))
15     except Exception as e:
16         print(e)
17     finally:
18         cur.close()
19         conn.close()

```

finally

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.