



Database Systems Mini Project

A report submitted to the

Department of Electrical and Information Engineering
Faculty of Engineering
University of Ruhuna Sri Lanka

On 09th of April 2024

In completing an assignment for the module

EE4350 Data Base systems

By:

Group 47

EG/2021/4696	NISHAKA A.M.I.
EG/2021/4697	NISSANKA N.A.K.D.
EG/2021/4701	PABASARA P.M.G.T.

CONTENTS

REQUIREMENT ANALYSIS	2
Functional Requirements	2
Data Requirements	3
Relations in the ER Diagram	5
CONCEPTUAL DESIGN	7
ER Diagram	7
UML Class Diagram	8
IMPLEMENTATION	9
Table Implementation	9
Inserting Data for Tables	18
Update and Delete	35
TRANSACTIONS	52
Simple Quarries	52
Complex Quarries	56
TUNING	64

REQUIREMENT ANALYSIS

Functional Requirements

The database system for “Farm Management” includes a wide range of features designed specifically to optimize farm management operations. Each feature is carefully integrated to ensure optimal performance and seamless data processing. These are some key components.

- Data Retrieval
- Ability to Alter Database Content
- Completing Data Gaps or Missing Data
- Capability to Edit Existing Data
- Ability to Add Comments to Incomplete or Ambiguous
- Capacity to Provide Data with Enhanced Performance

The Schema comprehensively addresses farm management needs, encompassing employee, Farmer authentication, activities tracking such as Harvesting, Suppliers and Salary management. It enables efficient allocation of tasks for Employees and Farmers, production tracking, ensures secure access, and facilitates inventory and financial management for effective planning.

With its comprehensive functionalities and meticulous design, this database system serves as an invaluable tool for optimizing efficiency and decision-making in farm management.

Data Requirements

The attributes of each entity are shown below

1. Crop

- Crop_ID
- Crop_name
- Variety
- Planting_Date
- Harvest_Date
- Crop_age
- Pests

2. Employee

- Employee_ID
- Employee_name
- Supervisor
- Contact_info
- Email

3. Equipment

- Equipment_ID
- Modal
- Equipment_type
- Purchase_date

4. Field

- Field_ID
- Field_name
- Size

5. Harvest

- Harvest_month
- Quantity

6. Head_farmer

- Farmer_ID
- Farmer_name
- Email_address
- Street
- City
- Province
- Zip_code
- Contact_info

7. Salary

- Salary_month
- Working_hours
- Payment_per_hour
- Monthly_salary

8. Supplier

- Supplier_ID
- Supplier_name
- Product_supplied
- Contact_info

Relations in the ER Diagram

The ER diagram comprises significant entities and relationships, incorporating weak entities and a recursive relationship:

- **One to Many Relationships:**

- HEAD_FARMER and EQUIPMENT relationship (Owner)
- CROP and HARVEST relationship (Harvesting)
- HEAD_FARMER and FIELD relationship (Owner)
- FIELD and EMPLOYEE relationship (Working)
- EMPLOYEE and SALARY relationship (Receives)
- Recursive relationship (Supervise)

- **Many to Many relationships:**

- SUPPLIER and HEAD_FARMER relationship (Ordering)
- FIELD and EQUIPMENT relationship (Utilizing)
- FIELD and CROP relationship (Cultivating)

- **Strong Entities:**

- HEAD_FARMER
- SUPPLIER
- FIELD
- EQUIPMENT
- EMPLOYEE
- CROP

- **Weak Entities:**

- Two weak entities are present, denoted by double-line rectangles.
- "SALARY" is reliant on the "EMPLOYEE" entity and lacks its unique key, categorizing it as weak.
- "HARVEST" also lacks a key and is entirely dependent on the "CROP" entity, thus considered weak.

- **Recursive Relationship:**
 - The database features a recursive relationship, specifically involving supervisor and employees.
 - This relationship demonstrates that a supervisor can oversee a limited number of employees, potentially including one who may also function as an employee, creating a recursive pattern.
- **Composite Attributes**

Address (Head Farmer Attribute)

- **Multi-valued Attributes**

Pests (Crop Attribute)

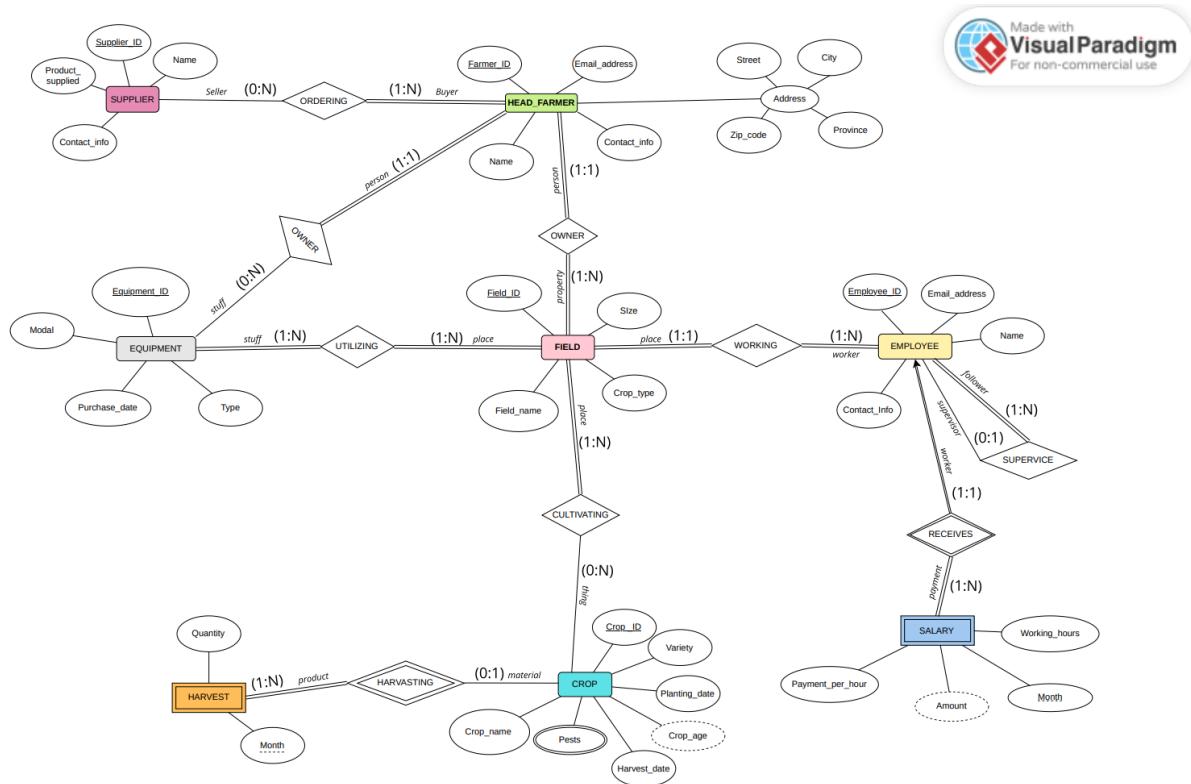
- **Derived Attributes**

Crop_age (Crop Attribute)

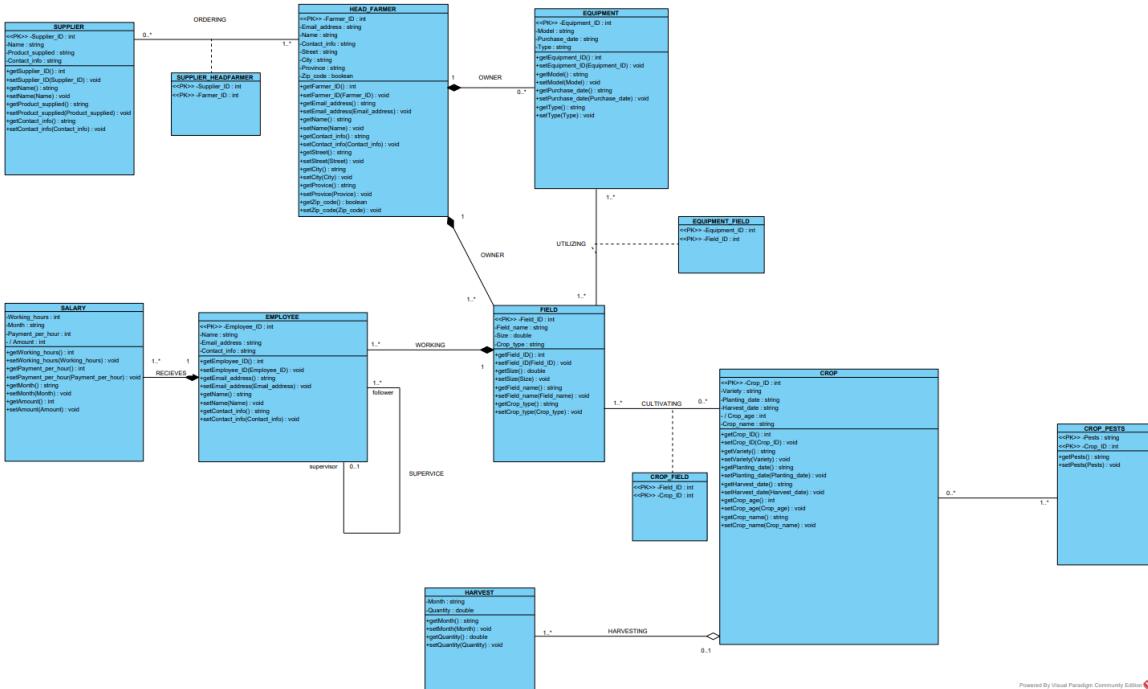
Amount (Salary Attribute)

CONCEPTUAL DESIGN

ER Diagram



UML Class Diagram



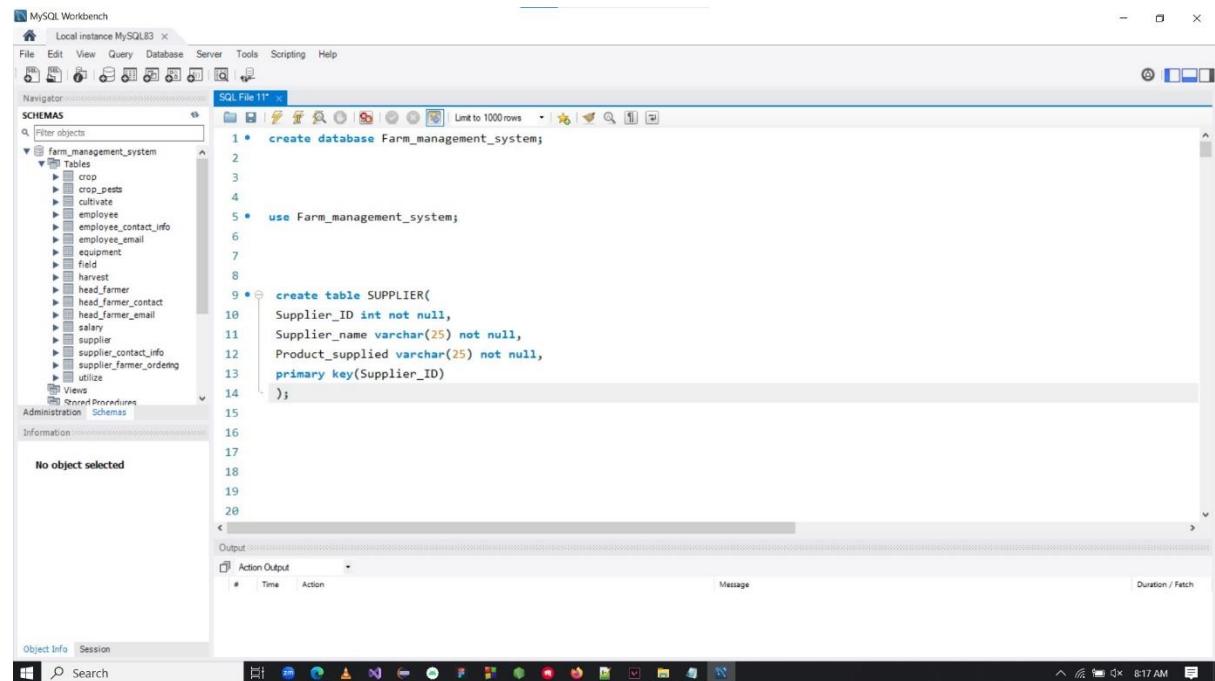
Powered By Visual Paradigm Community Edition

IMPLEMENTATION

All of the implemented database model screenshots are included in this chapter.

Table Implementation

1. Supplier table

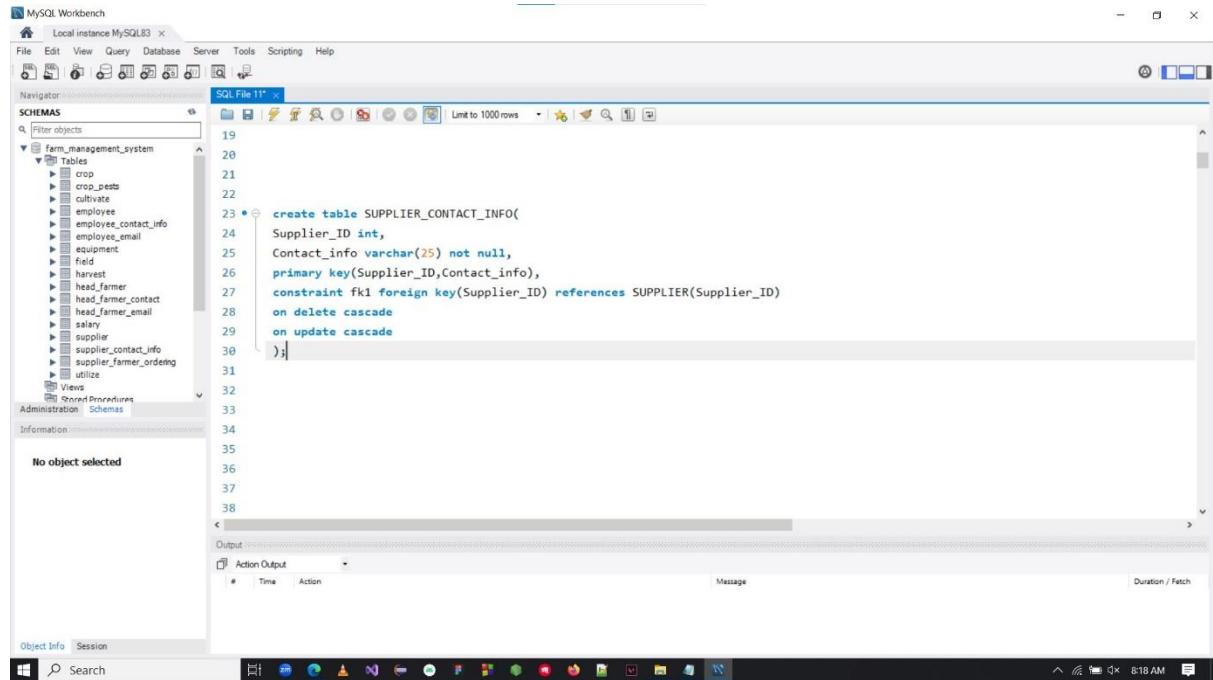


The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Shows the schema "farm_management_system" with tables like crop, crop pests, field, harvest, head_farmer, head_farmer_contact, salary, supplier, etc.
- SQL Editor:** SQL File 11, containing the following SQL code:

```
1 •  create database Farm_management_system;
2
3
4
5 •  use Farm_management_system;
6
7
8
9 •  create table SUPPLIER(
10    Supplier_ID int not null,
11    Supplier_name varchar(25) not null,
12    Product_supplied varchar(25) not null,
13    primary key(Supplier_ID)
14);
```
- Output:** Action Output, showing the results of the query execution.
- Bottom Bar:** Object Info, Session, and a taskbar with various icons.

2. Supplier_contact_info table

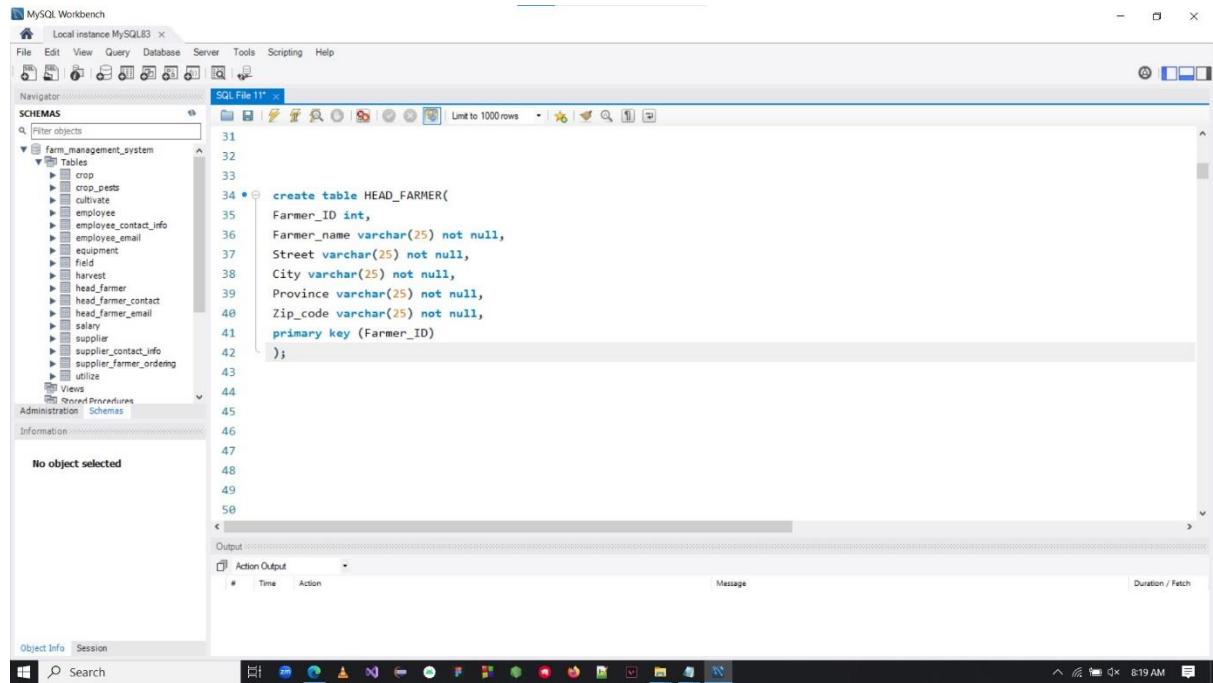


The screenshot shows the MySQL Workbench interface with the SQL Editor tab open. The code pane contains the following SQL script:

```
19
20
21
22
23 •  create table SUPPLIER_CONTACT_INFO(
24     Supplier_ID int,
25     Contact_info varchar(25) not null,
26     primary key(Supplier_ID,Contact_info),
27     constraint fk1 foreign key(Supplier_ID) references SUPPLIER(Supplier_ID)
28     on delete cascade
29     on update cascade
30 );
31
32
33
34
35
36
37
38
```

The code is part of a larger script, indicated by line numbers 19 through 38. The table is named `SUPPLIER_CONTACT_INFO` and has two columns: `Supplier_ID` and `Contact_info`. It includes a primary key constraint on both columns and a foreign key constraint `fk1` referencing the `SUPPLIER` table's `Supplier_ID` column with cascade delete and update rules.

3. Head_farmer table

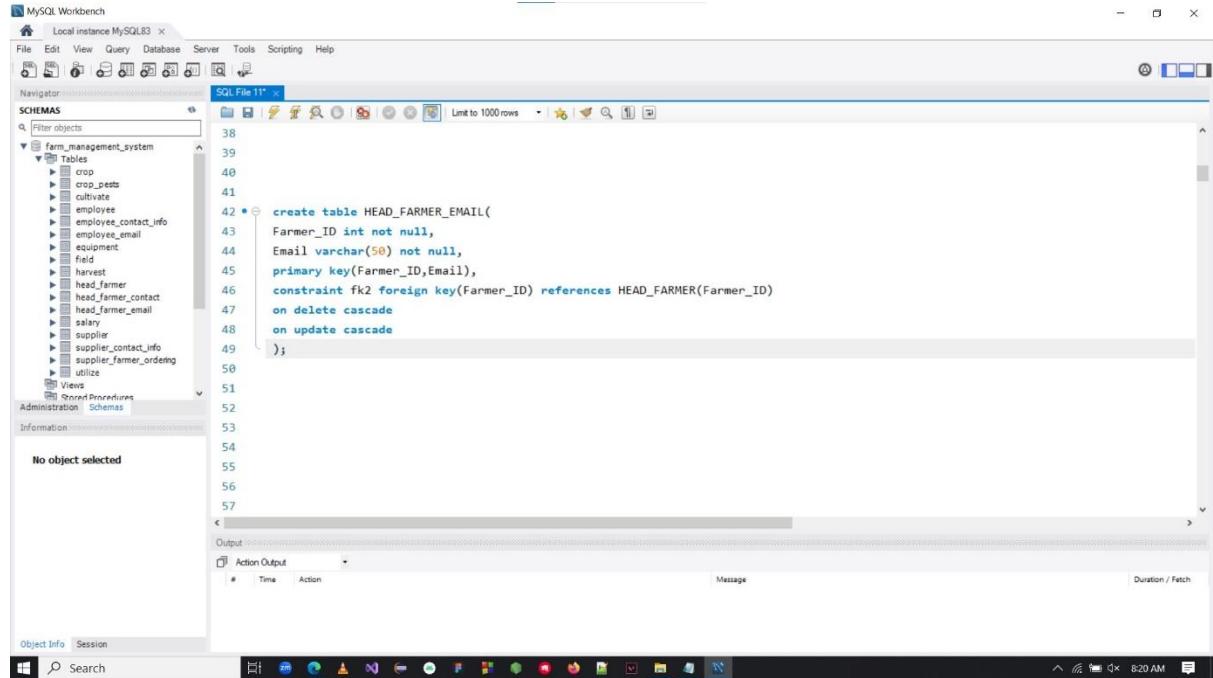


The screenshot shows the MySQL Workbench interface with the SQL Editor tab open. The code pane contains the following SQL script:

```
31
32
33
34 •  create table HEAD_FARMER(
35     Farmer_ID int,
36     Farmer_name varchar(25) not null,
37     Street varchar(25) not null,
38     City varchar(25) not null,
39     Province varchar(25) not null,
40     Zip_code varchar(25) not null,
41     primary key (Farmer_ID)
42 );
43
44
45
46
47
48
49
50
```

The code is part of a larger script, indicated by line numbers 31 through 50. The table is named `HEAD_FARMER` and has one column `Farmer_ID`. It includes a primary key constraint on the `Farmer_ID` column. The table also contains several other columns: `Farmer_name`, `Street`, `City`, `Province`, and `Zip_code`, all defined as `varchar(25)` and marked as not null.

4. Head_farmer_email table

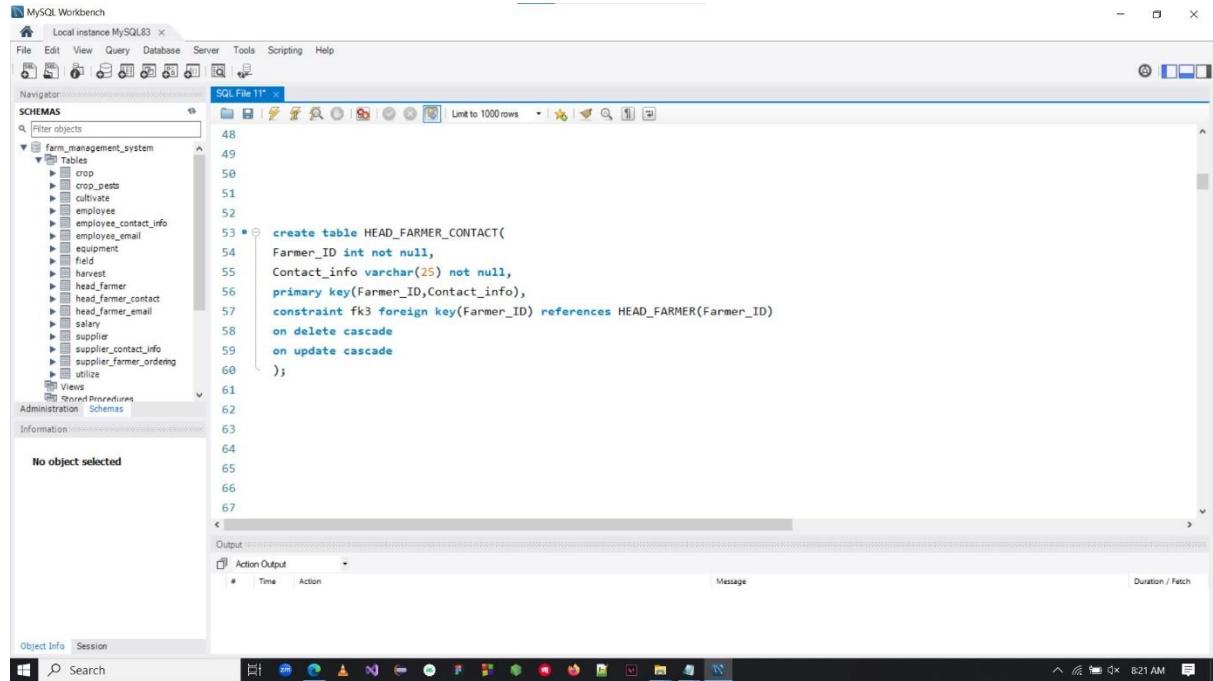


The screenshot shows the MySQL Workbench interface with the SQL Editor tab open. The code pane contains the following SQL script:

```
38
39
40
41
42 • create table HEAD_FARMER_EMAIL(
43     Farmer_ID int not null,
44     Email varchar(50) not null,
45     primary key(Farmer_ID,Email),
46     constraint fk2 foreign key(Farmer_ID) references HEAD_FARMER(Farmer_ID)
47     on delete cascade
48     on update cascade
49 );
50
51
52
53
54
55
56
57
```

The Navigator pane on the left shows the database schema, including the 'farm_management_system' database and its tables: crop, crop_pests, cultivate, employee, employee_contact_info, employee_email, equipment, field, harvest, head_farmer, head_farmer_contact, head_farmer_email, salary, supplier, supplier_contact_info, supplier_farmer_ordering, and utilize.

5. Head_farmer_contact table

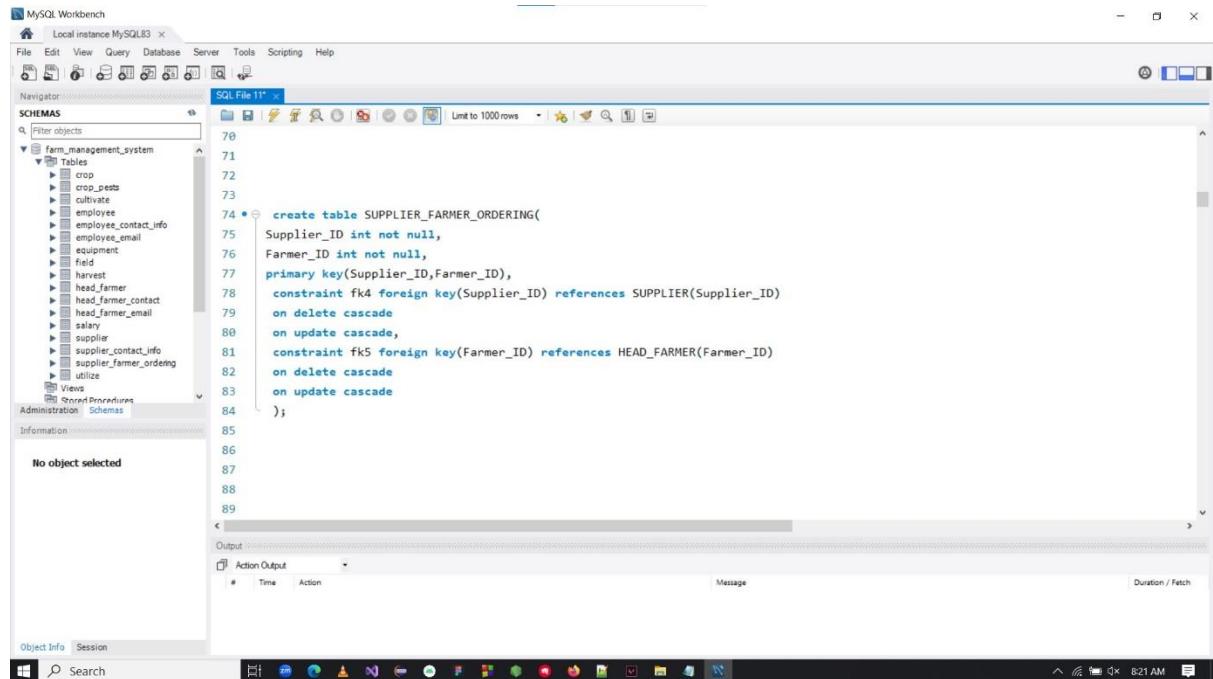


The screenshot shows the MySQL Workbench interface with the SQL Editor tab open. The code pane contains the following SQL script:

```
48
49
50
51
52
53 • create table HEAD_FARMER_CONTACT(
54     Farmer_ID int not null,
55     Contact_info varchar(25) not null,
56     primary key(Farmer_ID,Contact_info),
57     constraint fk3 foreign key(Farmer_ID) references HEAD_FARMER(Farmer_ID)
58     on delete cascade
59     on update cascade
60 );
61
62
63
64
65
66
67
```

The Navigator pane on the left shows the database schema, including the 'farm_management_system' database and its tables: crop, crop_pests, cultivate, employee, employee_contact_info, employee_email, equipment, field, harvest, head_farmer, head_farmer_contact, head_farmer_email, salary, supplier, supplier_contact_info, supplier_farmer_ordering, and utilize.

6. Supplier_farmer_ordering table

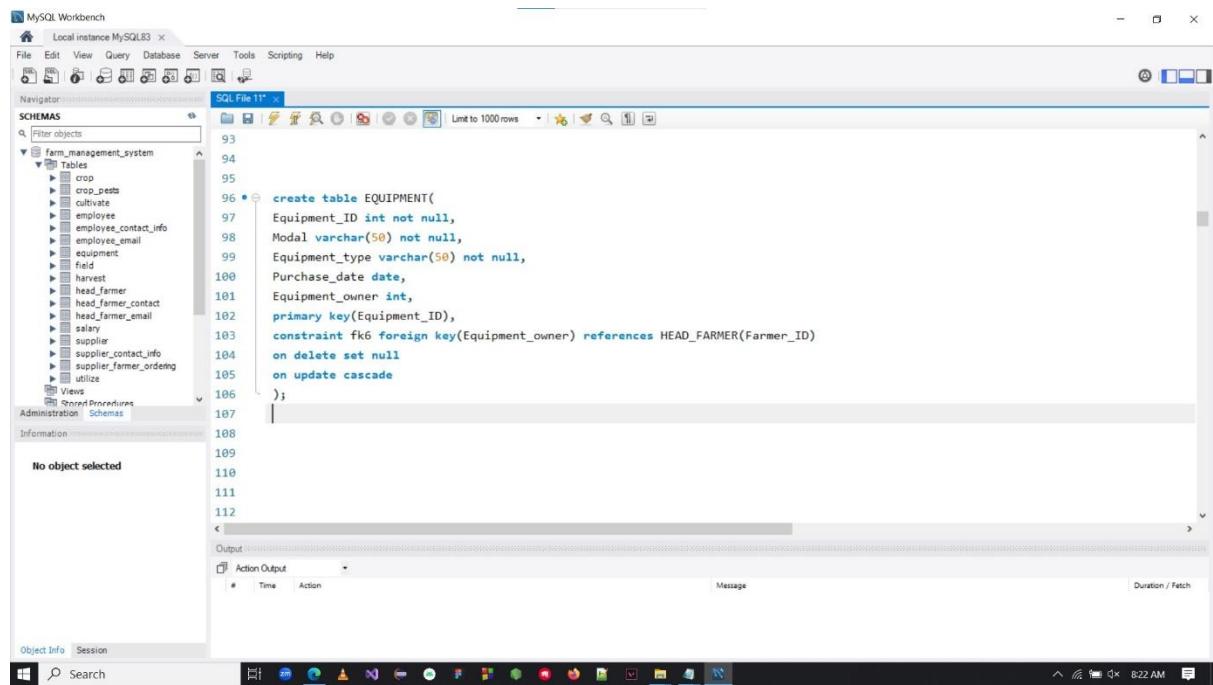


The screenshot shows the MySQL Workbench interface with the SQL tab active. The code pane displays the creation of the `SUPPLIER_FARMER_ORDERING` table:

```
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
```

```
create table SUPPLIER_FARMER_ORDERING(
    Supplier_ID int not null,
    Farmer_ID int not null,
    primary key(Supplier_ID, Farmer_ID),
    constraint fk4 foreign key(Supplier_ID) references SUPPLIER(Supplier_ID)
    on delete cascade
    on update cascade,
    constraint fk5 foreign key(Farmer_ID) references HEAD_FARMER(Farmer_ID)
    on delete cascade
    on update cascade
);
```

7. Equipment table

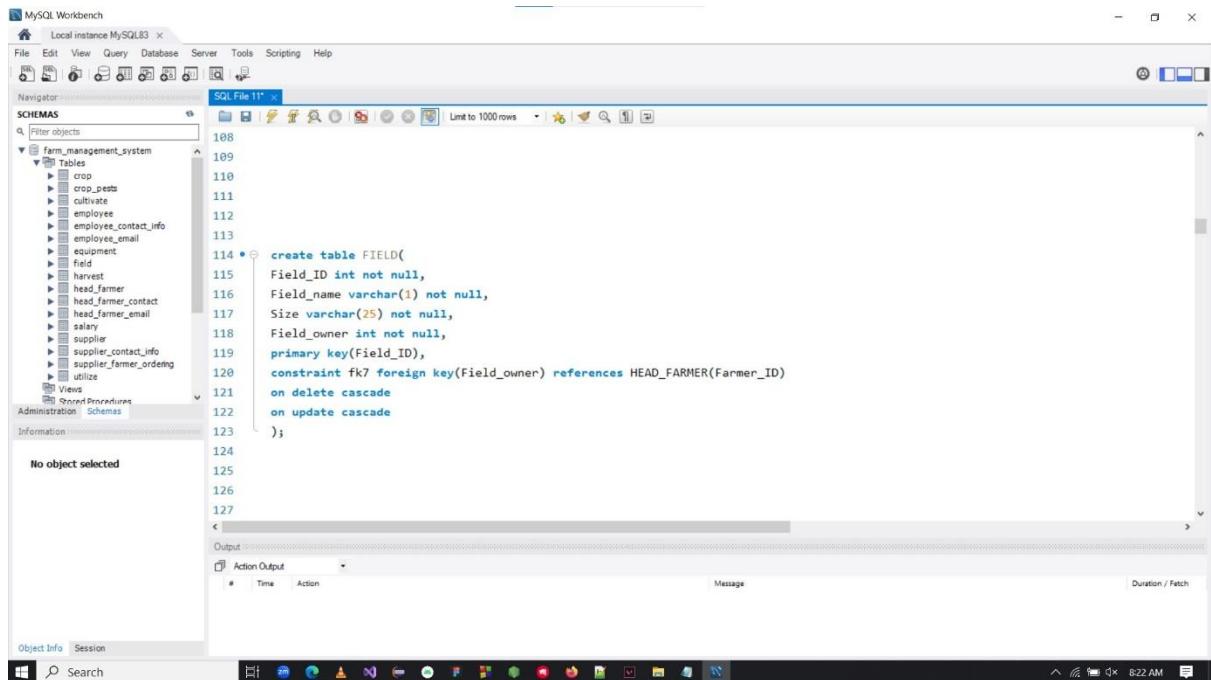


The screenshot shows the MySQL Workbench interface with the SQL tab active. The code pane displays the creation of the `EQUIPMENT` table:

```
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
```

```
create table EQUIPMENT(
    Equipment_ID int not null,
    Model varchar(50) not null,
    Equipment_type varchar(50) not null,
    Purchase_date date,
    Equipment_owner int,
    primary key(Equipment_ID),
    constraint fk6 foreign key(Equipment_owner) references HEAD_FARMER(Farmer_ID)
    on delete set null
    on update cascade
);
```

8. Field table

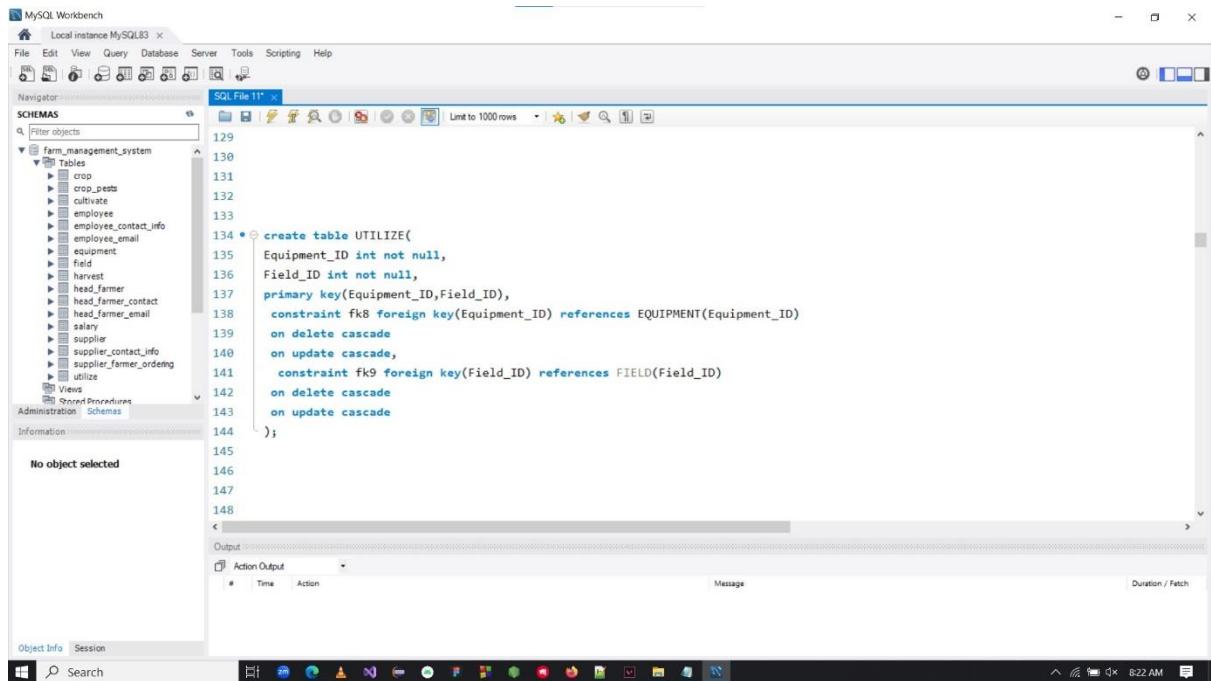


The screenshot shows the MySQL Workbench interface with the SQL editor tab open. The code being run is:

```
108
109
110
111
112
113
114 • create table FIELD(
115     Field_ID int not null,
116     Field_name varchar(1) not null,
117     Size varchar(25) not null,
118     Field_owner int not null,
119     primary key(Field_ID),
120     constraint fk7 foreign key(Field_owner) references HEAD_FARMER(Farmer_ID)
121     on delete cascade
122     on update cascade
123 );
124
125
126
127
```

The code is part of a larger script, indicated by the line numbers 108 to 127. The table is defined with columns for Field_ID, Field_name, Size, and Field_owner, with a primary key on Field_ID and a foreign key constraint fk7 referencing the HEAD_FARMER table.

9. Utilize table



The screenshot shows the MySQL Workbench interface with the SQL editor tab open. The code being run is:

```
129
130
131
132
133
134 • create table UTILIZE(
135     Equipment_ID int not null,
136     Field_ID int not null,
137     primary key(Equipment_ID,Field_ID),
138     constraint fk8 foreign key(Equipment_ID) references EQUIPMENT(Equipment_ID)
139     on delete cascade
140     on update cascade,
141     constraint fk9 foreign key(Field_ID) references FIELD(Field_ID)
142     on delete cascade
143     on update cascade
144 );
```

The code is part of a larger script, indicated by the line numbers 129 to 148. The table UTILIZE is created with two columns: Equipment_ID and Field_ID, both of which are foreign keys. The primary key is a composite key consisting of both columns. Foreign key constraints fk8 and fk9 are defined, linking to the EQUIPMENT and FIELD tables respectively.

10.Employee table

The screenshot shows the MySQL Workbench interface with the SQL tab active. The code editor displays the creation of the EMPLOYEE table:

```
150
151
152
153 • ⚡ create table EMPLOYEE(
154     Employee_ID int not null,
155     Employee_name varchar(50) not null,
156     Supervisor int,
157     Working_field int not null,
158     primary key(Employee_ID),
159     constraint fk10 foreign key(Supervisor) references EMPLOYEE(Employee_ID)
160     on delete set null
161     on update cascade,
162     constraint fk11 foreign key(Working_field) references FIELD(Field_ID)
163     on delete cascade
164     on update cascade
165 );
166
167
168
169
```

The code editor has a vertical scroll bar on the left and a horizontal scroll bar at the bottom. The status bar at the bottom right shows "8:23 AM".

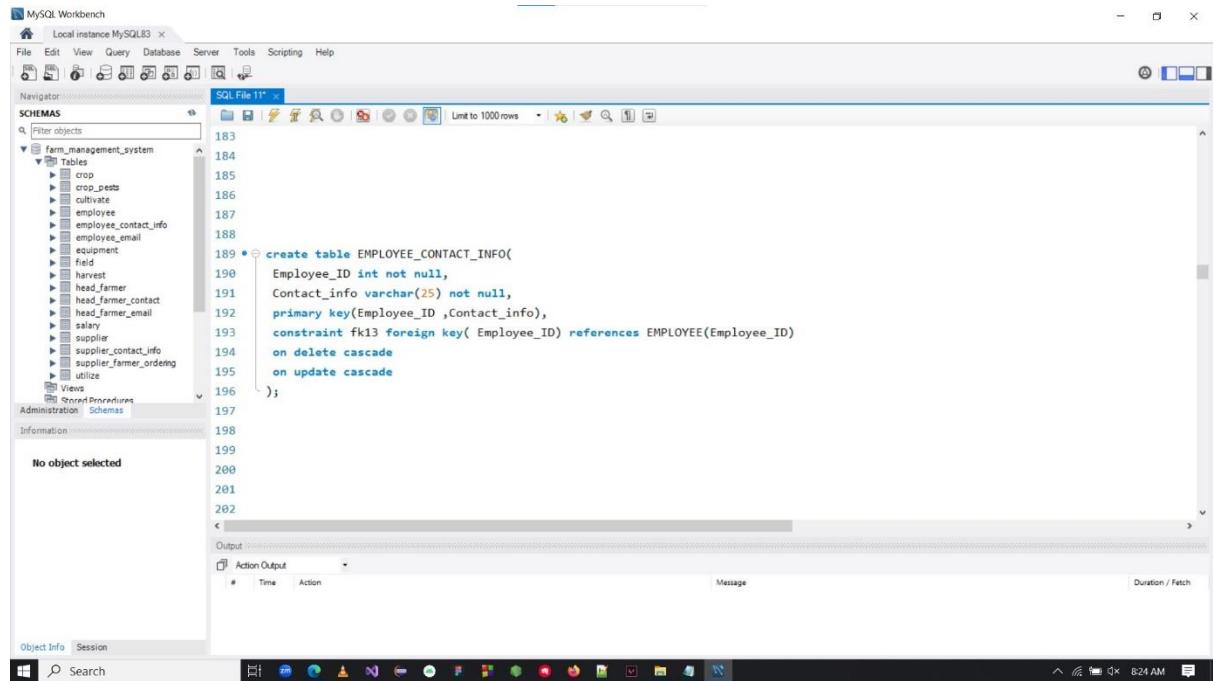
11.Employee_email table

The screenshot shows the MySQL Workbench interface with the SQL tab active. The code editor displays the creation of the EMPLOYEE_EMAIL table:

```
168
169
170 • ⚡ create table EMPLOYEE_EMAIL(
171     Employee_ID int not null,
172     Email varchar(50) not null,
173     primary key(Employee_ID ,Email),
174     constraint fk12 foreign key( Employee_ID ) references EMPLOYEE(Employee_ID)
175     on delete cascade
176     on update cascade
177 );
178
179
180
181
182
183
184
185
186
187
```

The code editor has a vertical scroll bar on the left and a horizontal scroll bar at the bottom. The status bar at the bottom right shows "8:23 AM".

12.Employee_contact_info table

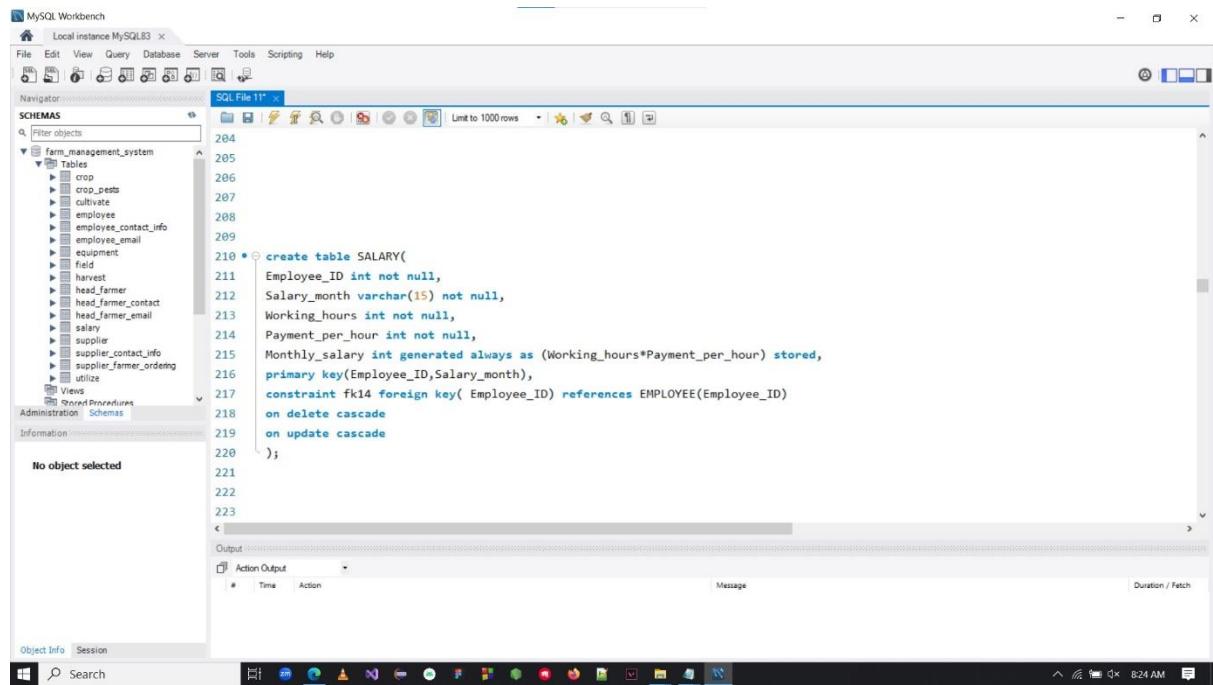


The screenshot shows the MySQL Workbench interface with the SQL editor tab open. The code being run is:

```
183
184
185
186
187
188
189 • ① create table EMPLOYEE_CONTACT_INFO(
190   Employee_ID int not null,
191   Contact_info varchar(25) not null,
192   primary key(Employee_ID ,Contact_info),
193   constraint fk13 foreign key( Employee_ID ) references EMPLOYEE(Employee_ID)
194   on delete cascade
195   on update cascade
196 );
197
198
199
200
201
202
```

The code defines a table named EMPLOYEE_CONTACT_INFO with two columns: Employee_ID (int, not null) and Contact_info (varchar(25), not null). It includes a primary key constraint on both columns and a foreign key constraint named fk13 that references the Employee_ID column in the EMPLOYEE table. The constraint uses cascade operations for delete and update.

13.Salary table



The screenshot shows the MySQL Workbench interface with the SQL editor tab open. The code being run is:

```
204
205
206
207
208
209
210 • ① create table SALARY(
211   Employee_ID int not null,
212   Salary_month varchar(15) not null,
213   Working_hours int not null,
214   Payment_per_hour int not null,
215   Monthly_salary int generated always as (Working_hours*Payment_per_hour) stored,
216   primary key(Employee_ID,Salary_month),
217   constraint fk14 foreign key( Employee_ID ) references EMPLOYEE(Employee_ID)
218   on delete cascade
219   on update cascade
220 );
221
222
223
```

The code defines a table named SALARY with four columns: Employee_ID (int, not null), Salary_month (varchar(15), not null), Working_hours (int, not null), and Payment_per_hour (int, not null). It includes a primary key constraint on both Employee_ID and Salary_month. A generated column named Monthly_salary is defined as the product of Working_hours and Payment_per_hour. A foreign key constraint named fk14 references the Employee_ID column in the EMPLOYEE table. The constraint uses cascade operations for delete and update.

14.Crop table

The screenshot shows the MySQL Workbench interface with the SQL tab selected. The code editor displays the following SQL script:

```
227
228
229
230
231
232
233 • ② create table CROP(
234     Crop_ID int not null,
235     Crop_name varchar(25) not null,
236     Variety varchar(25) not null,
237     Planting_date date,
238     Harvest_date date,
239     Crop_age int as (datediff(Harvest_date,Planting_date)),
240     primary key(Crop_ID)
241 );
242
243
244
245
246
```

The code is part of a larger script, indicated by line numbers 227 through 246. The table structure is defined with columns for Crop_ID (primary key), Crop_name, Variety, Planting_date, Harvest_date, and Crop_age (calculated as the difference between Harvest_date and Planting_date). The primary key is set to Crop_ID.

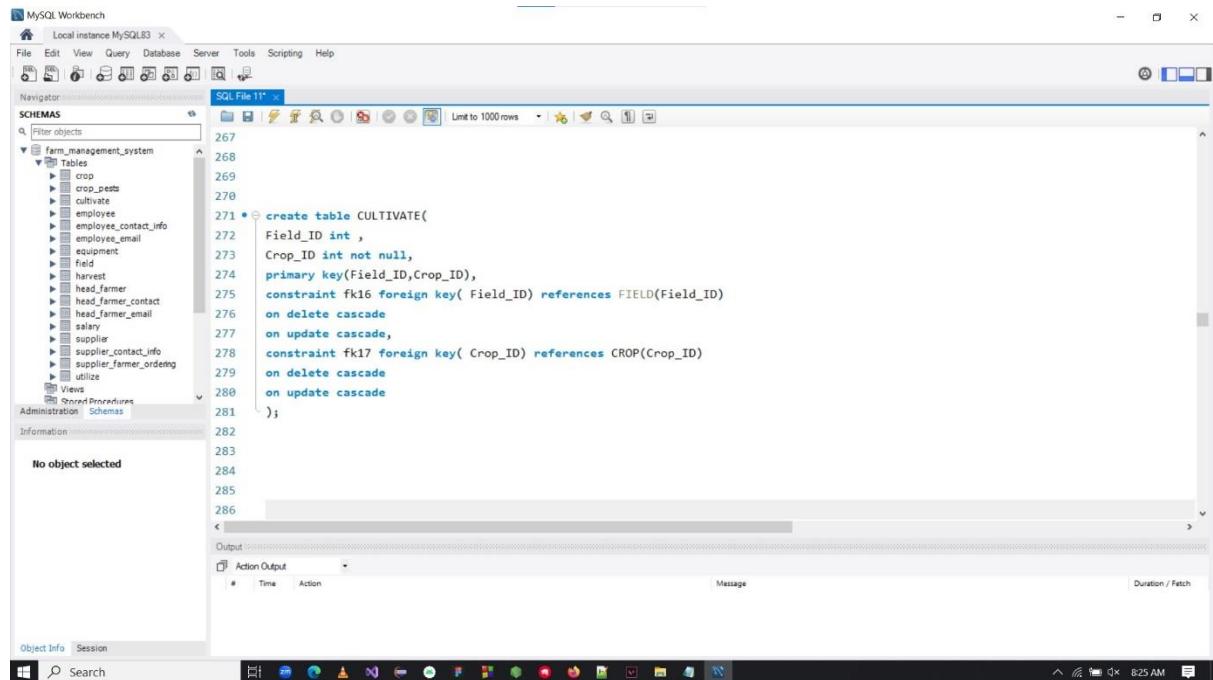
15.Crop_pests table

The screenshot shows the MySQL Workbench interface with the SQL tab selected. The code editor displays the following SQL script:

```
247
248
249
250
251
252
253
254 • ② create table CROP_PESTS(
255     Crop_ID int not null,
256     Pests varchar(25),
257     primary key(Crop_ID,Pests),
258     constraint fk15 foreign key( Crop_ID,Pests) references CROP(Crop_ID)
259     on delete cascade
260     on update cascade
261 );
262
263
264
265
266
```

The code is part of a larger script, indicated by line numbers 247 through 266. The table structure is defined with columns for Crop_ID (primary key) and Pests. It includes a foreign key constraint named fk15 that references the CROP table's Crop_ID column. The table also features cascading delete and update rules.

16.Cultivate table

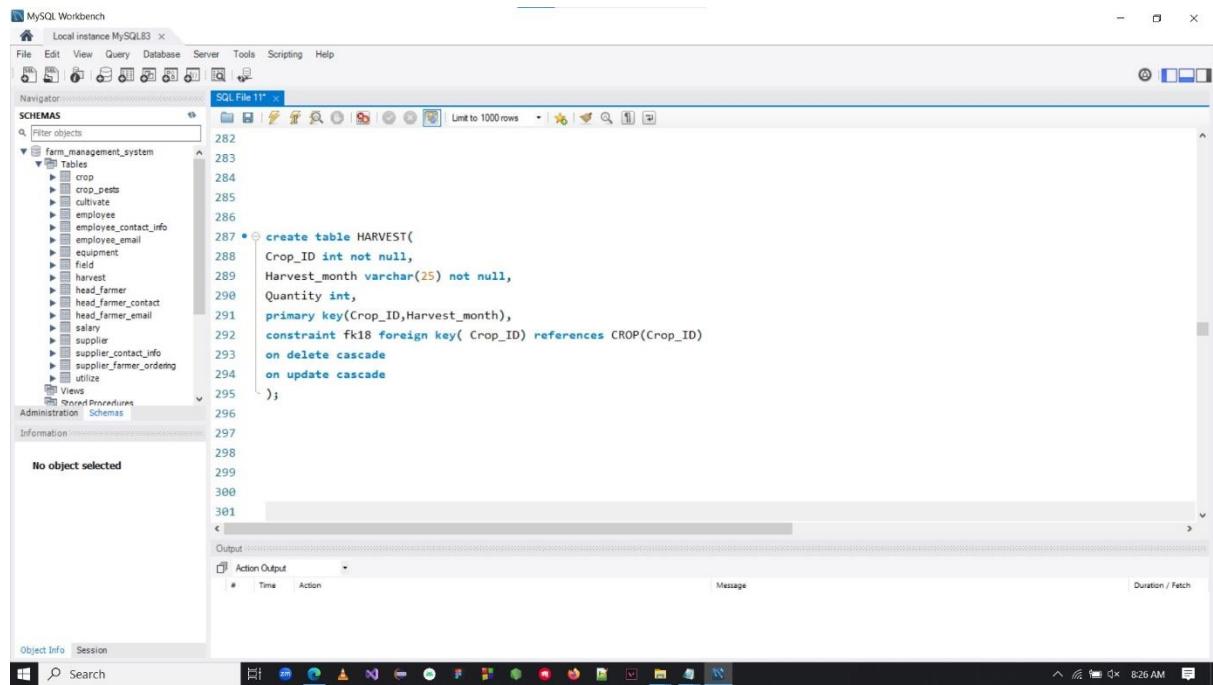


The screenshot shows the MySQL Workbench interface with the SQL Editor tab open. The code being run is:

```
267
268
269
270
271 • ② create table CULTIVATE(
272     Field_ID int ,
273     Crop_ID int not null,
274     primary key(Field_ID,Crop_ID),
275     constraint fk16 foreign key( Field_ID) references FIELD(Field_ID)
276     on delete cascade
277     on update cascade,
278     constraint fk17 foreign key( Crop_ID) references CROP(Crop_ID)
279     on delete cascade
280     on update cascade
281 );
282
283
284
285
286
```

The code is part of a larger script, starting at line 267 and ending at line 281. The table structure includes composite primary keys for Field_ID and Crop_ID, and foreign key constraints linking to the FIELD and CROP tables.

17.Harvest table



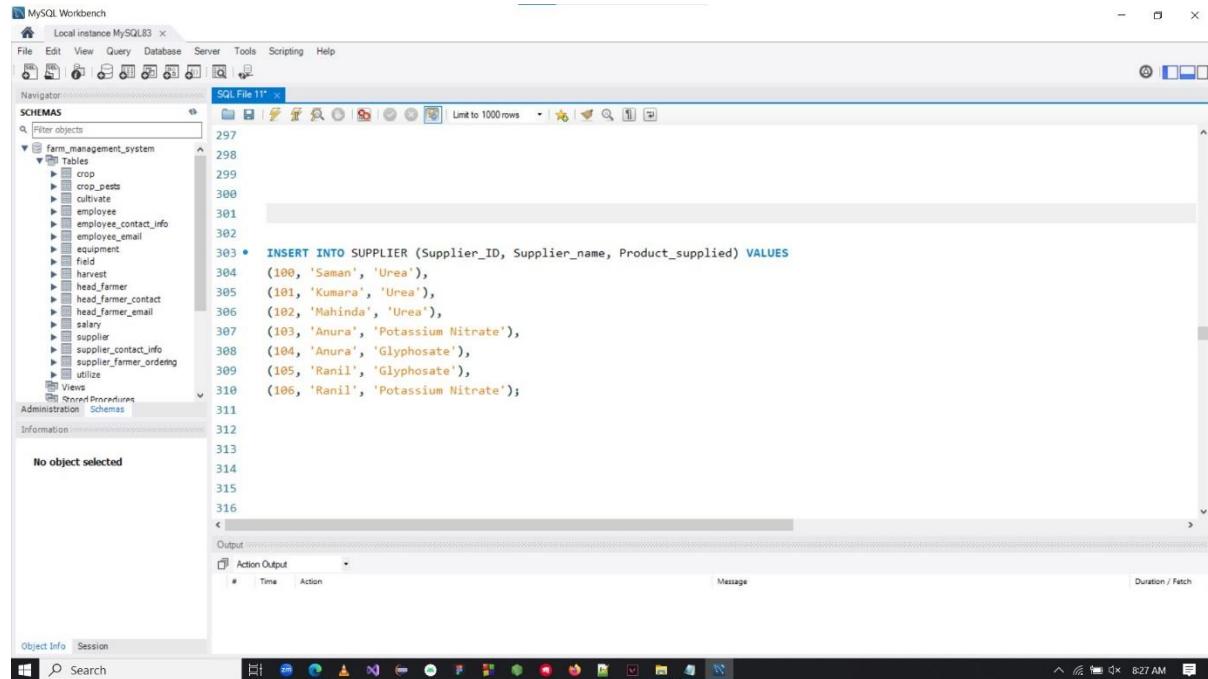
The screenshot shows the MySQL Workbench interface with the SQL Editor tab open. The code being run is:

```
282
283
284
285
286
287 • ② create table HARVEST(
288     Crop_ID int not null,
289     Harvest_month varchar(25) not null,
290     Quantity int,
291     primary key(Crop_ID,Harvest_month),
292     constraint fk18 foreign key( Crop_ID) references CROP(Crop_ID)
293     on delete cascade
294     on update cascade
295 );
296
297
298
299
300
301
```

The code is part of a larger script, starting at line 282 and ending at line 301. The table structure includes composite primary keys for Crop_ID and Harvest_month, and a foreign key constraint linking to the CROP table.

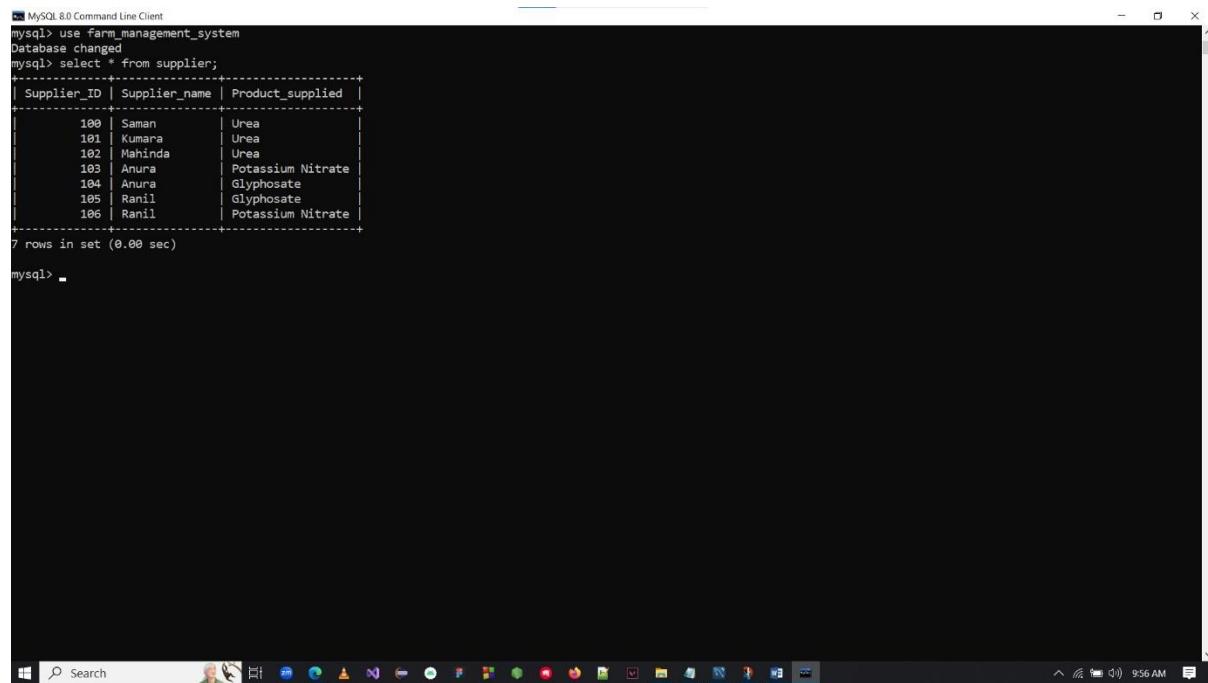
Inserting Data for Tables

1. Insert into Supplier table



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree, with 'farm_management_system' selected, showing various tables like crop, crop_pests, cultivate, employees, etc. The main area is the 'SQL File 11' editor, which contains the following SQL code:

```
INSERT INTO SUPPLIER (Supplier_ID, Supplier_name, Product_supplied) VALUES
(100, 'Saman', 'Urea'),
(101, 'Kumara', 'Urea'),
(102, 'Mahinda', 'Urea'),
(103, 'Anura', 'Potassium Nitrate'),
(104, 'Anura', 'Glyphosate'),
(105, 'Ranil', 'Glyphosate'),
(106, 'Ranil', 'Potassium Nitrate');
```



The screenshot shows the MySQL 8.0 Command Line Client window. The command line shows:

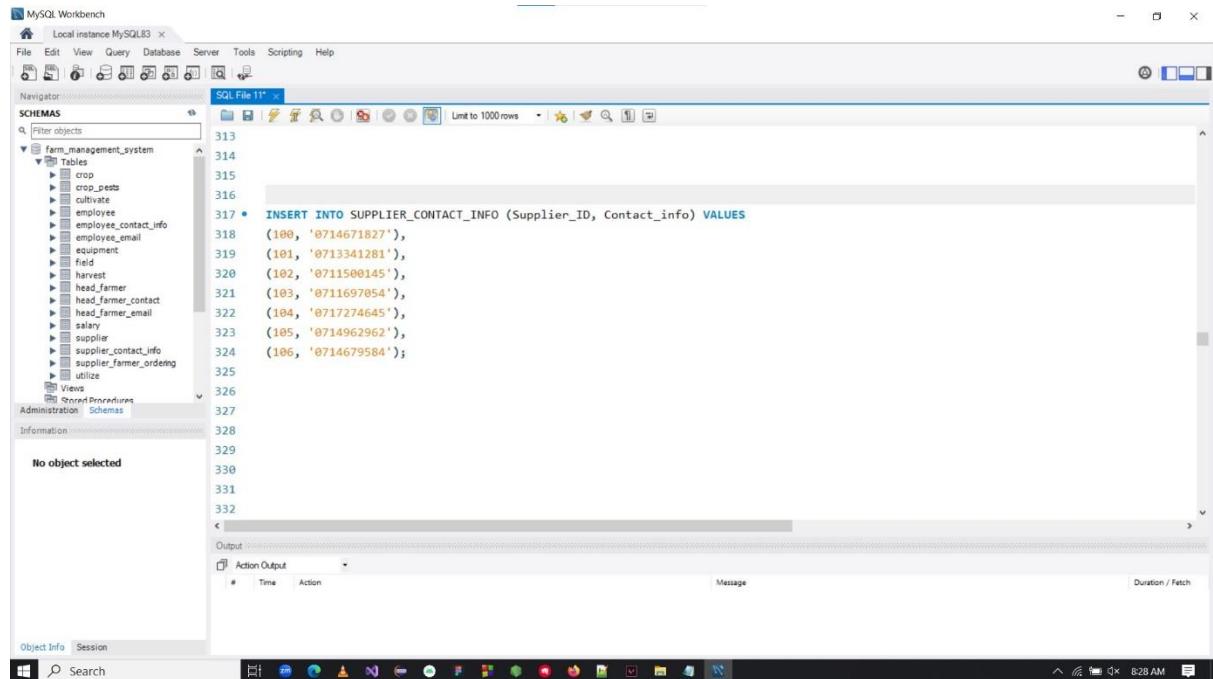
```
mysql> use farm_management_system
Database changed
mysql> select * from supplier;
```

The output shows the data inserted into the Supplier table:

Supplier_ID	Supplier_name	Product_supplied
100	Saman	Urea
101	Kumara	Urea
102	Mahinda	Urea
103	Anura	Potassium Nitrate
104	Anura	Glyphosate
105	Ranil	Glyphosate
106	Ranil	Potassium Nitrate

7 rows in set (0.00 sec)

2. Insert into Supplier_contact_info table



The screenshot shows the MySQL 8.0 Command Line Client window. The command entered is:

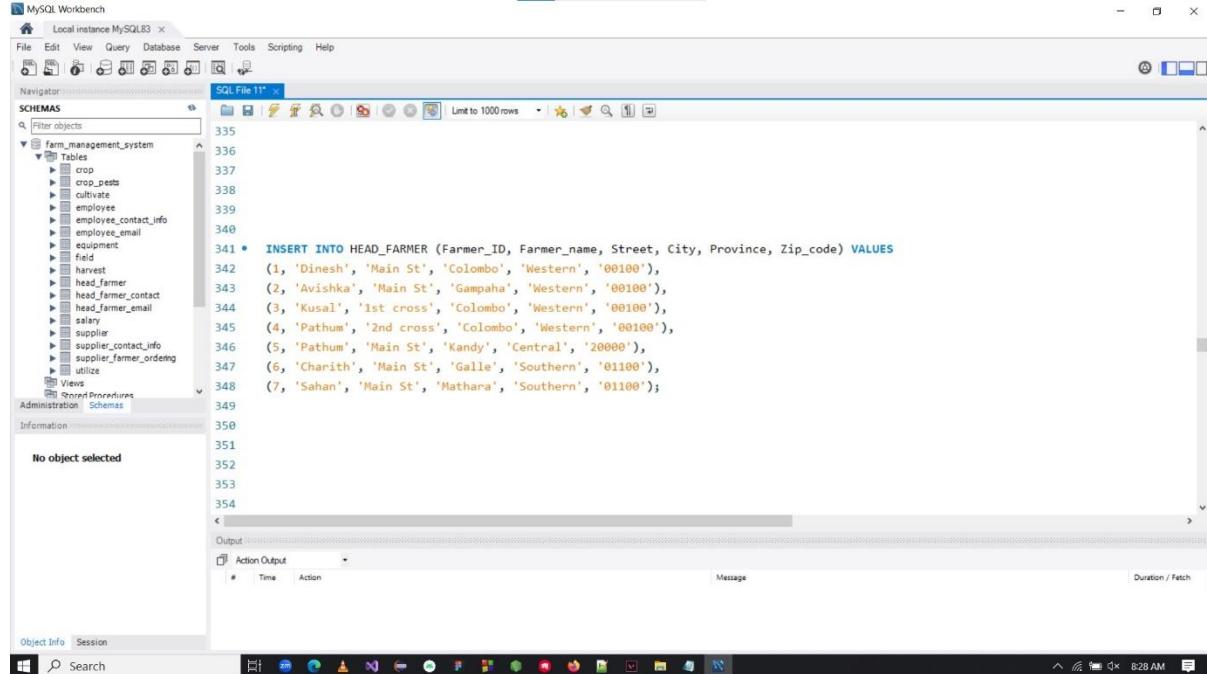
```
mysql> select * from supplier_contact_info;
```

The resulting table output is:

Supplier_ID	Contact_info
100	0714671827
101	0713341281
102	0711500145
103	0711697054
104	0717274645
105	0714962962
106	0714679584

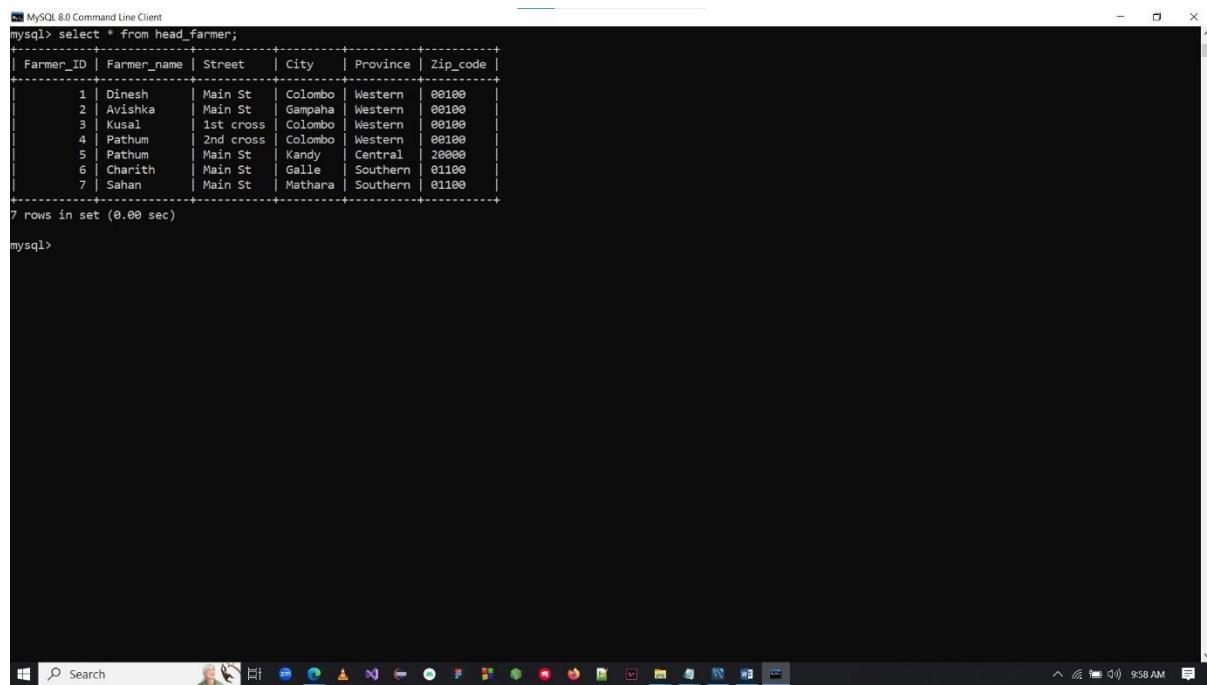
The status bar at the bottom shows the date and time as 9:57 AM.

3. Insert into Head_famer table



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, which includes the 'farm_management_system' schema with various tables like 'crop', 'crop_peds', 'cultivate', 'employee', etc. The main area is titled 'SQL File 11' and contains the following SQL code:

```
335
336
337
338
339
340
341 • INSERT INTO HEAD_FARMER (Farmer_ID, Farmer_name, Street, City, Province, Zip_code) VALUES
342 (1, 'Dinesh', 'Main St', 'Colombo', 'Western', '00100'),
343 (2, 'Avishka', 'Main St', 'Gampaha', 'Western', '00100'),
344 (3, 'Kusal', '1st cross', 'Colombo', 'Western', '00100'),
345 (4, 'Pathum', '2nd cross', 'Colombo', 'Western', '00100'),
346 (5, 'Pathum', 'Main St', 'Kandy', 'Central', '20000'),
347 (6, 'Charith', 'Main St', 'Galle', 'Southern', '01100'),
348 (7, 'Sahan', 'Main St', 'Mathara', 'Southern', '01100');
349
350
351
352
353
354
```

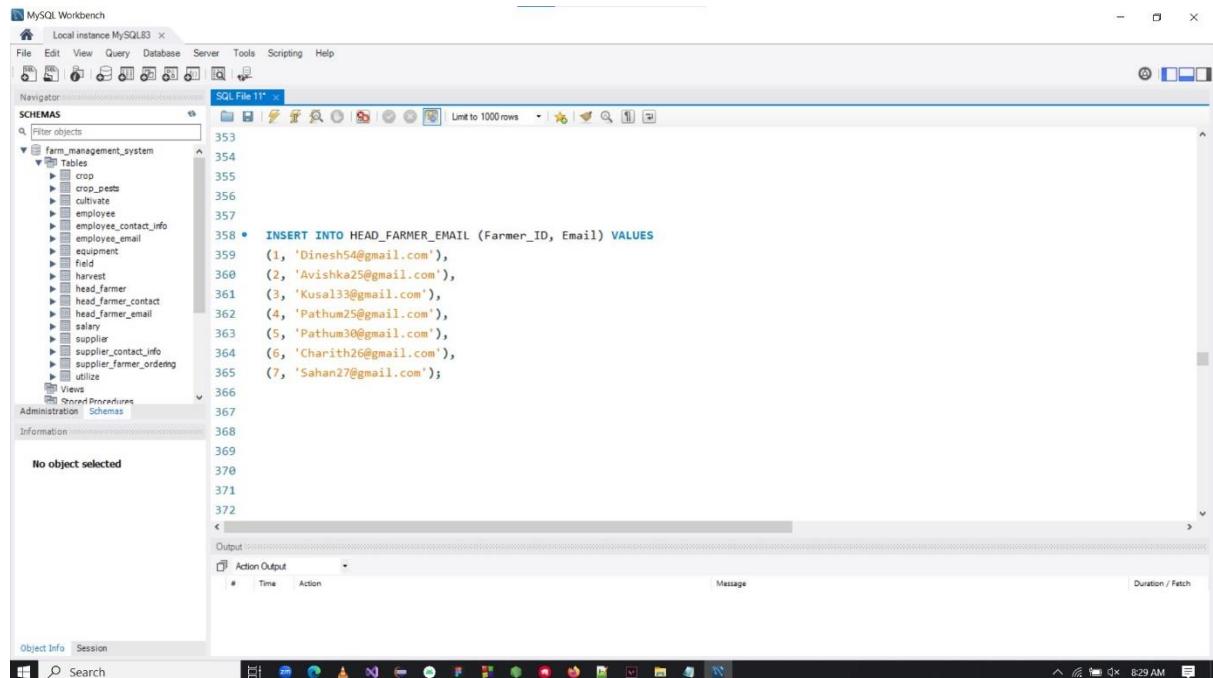


The screenshot shows the MySQL 8.0 Command Line Client window. The command 'select * from head_farmer;' has been run, and the output is displayed as a table:

Farmer_ID	Farmer_name	Street	City	Province	Zip_code
1	Dinesh	Main St	Colombo	Western	00100
2	Avishka	Main St	Gampaha	Western	00100
3	Kusal	1st cross	Colombo	Western	00100
4	Pathum	2nd cross	Colombo	Western	00100
5	Pathum	Main St	Kandy	Central	20000
6	Charith	Main St	Galle	Southern	01100
7	Sahan	Main St	Mathara	Southern	01100

7 rows in set (0.00 sec)

4. Insert into Head_farmer_email table

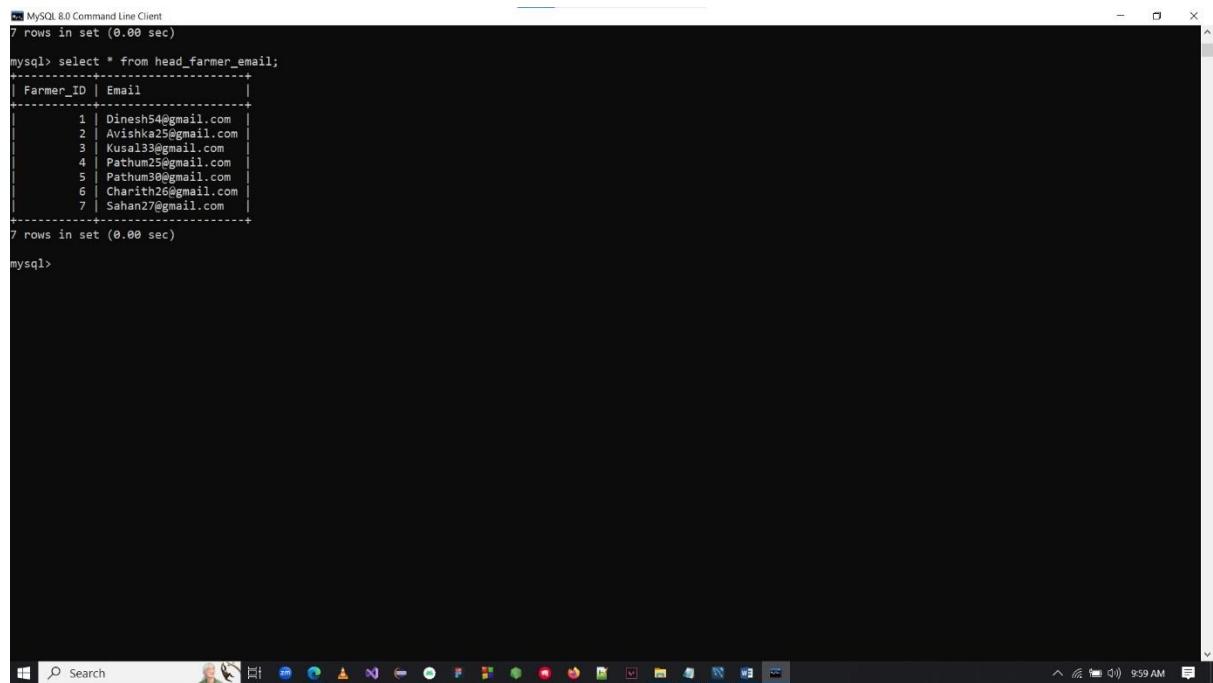


The screenshot shows the MySQL Workbench interface. The SQL Editor pane contains the following SQL code:

```
INSERT INTO HEAD_FARMER_EMAIL (Farmer_ID, Email) VALUES
(1, 'Dinesh54@gmail.com'),
(2, 'Avishka25@gmail.com'),
(3, 'Kusal33@gmail.com'),
(4, 'Pathum25@gmail.com'),
(5, 'Pathum30@gmail.com'),
(6, 'Charith26@gmail.com'),
(7, 'Sahan27@gmail.com');
```

The Results pane below shows the output of the query:

Farmer_ID	Email
1	Dinesh54@gmail.com
2	Avishka25@gmail.com
3	Kusal33@gmail.com
4	Pathum25@gmail.com
5	Pathum30@gmail.com
6	Charith26@gmail.com
7	Sahan27@gmail.com



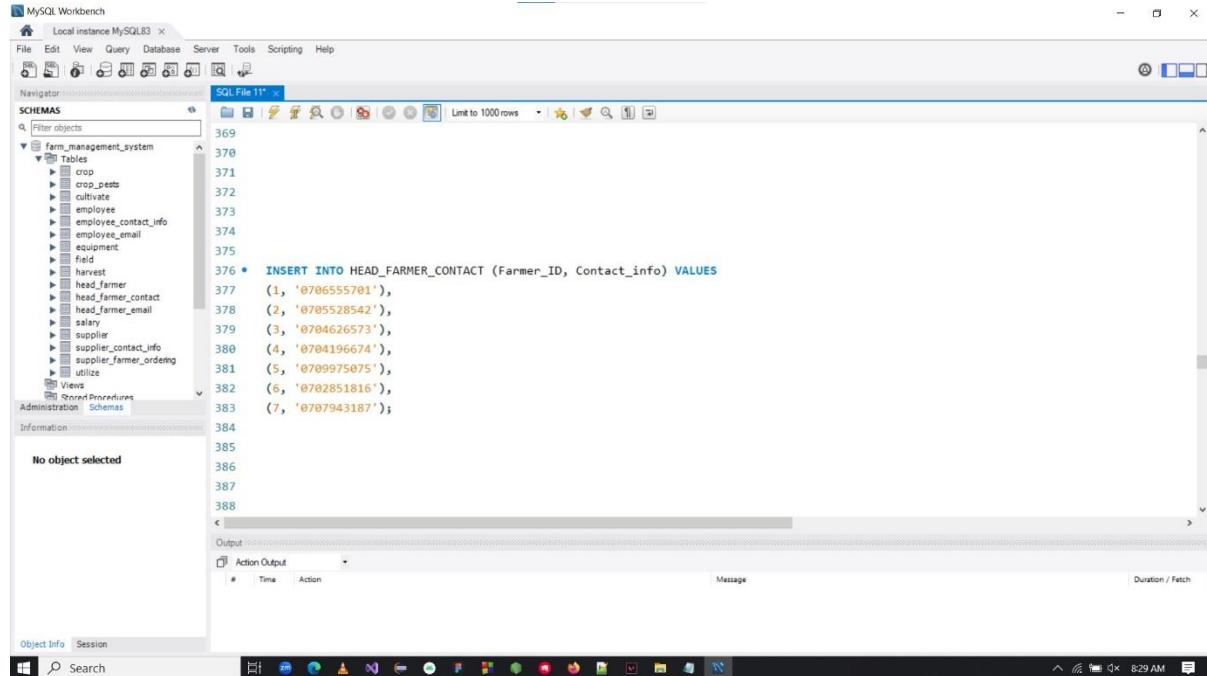
The screenshot shows the MySQL 8.0 Command Line Client interface. The command line shows the following output:

```
7 rows in set (0.00 sec)

mysql> select * from head_farmer_email;
+-----+-----+
| Farmer_ID | Email      |
+-----+-----+
| 1 | Dinesh54@gmail.com |
| 2 | Avishka25@gmail.com |
| 3 | Kusal33@gmail.com |
| 4 | Pathum25@gmail.com |
| 5 | Pathum30@gmail.com |
| 6 | Charith26@gmail.com |
| 7 | Sahan27@gmail.com |
+-----+-----+
7 rows in set (0.00 sec)

mysql>
```

5. Insert into Head_farmer_contact table

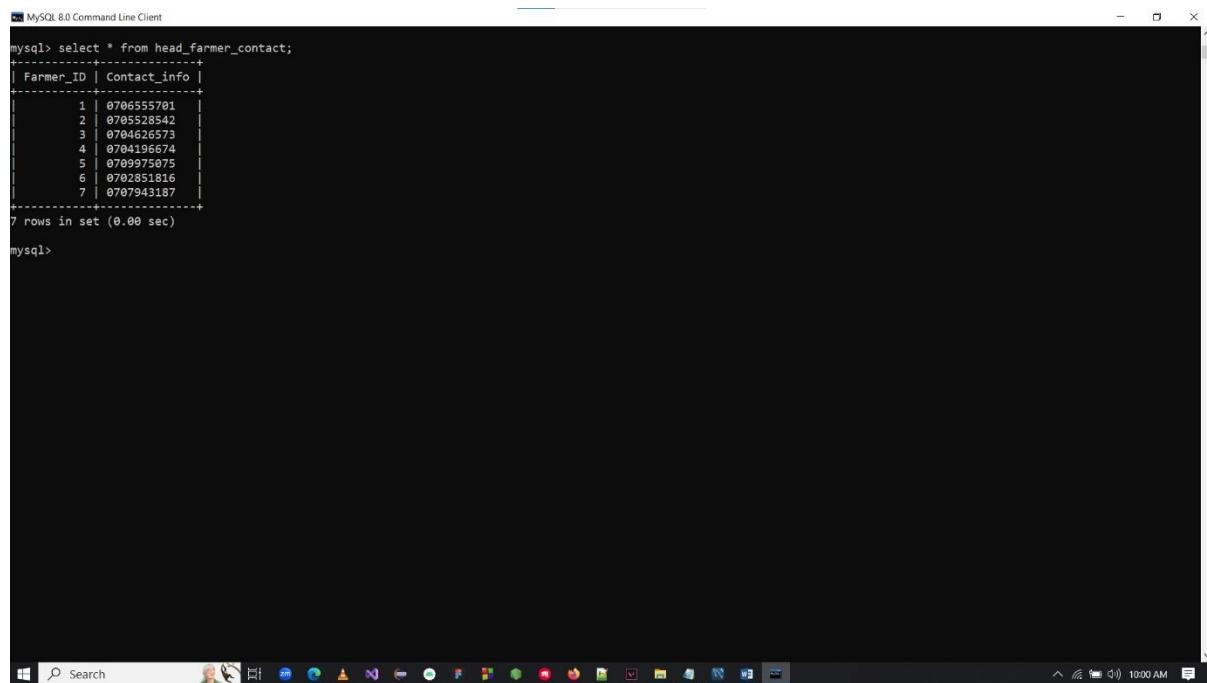


The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, there is a script titled "SQL File 11" containing the following SQL code:

```
INSERT INTO HEAD_FARMER_CONTACT (Farmer_ID, Contact_info) VALUES
(1, '0706555701'),
(2, '0705528542'),
(3, '0784626573'),
(4, '0704196674'),
(5, '0789975075'),
(6, '0702851816'),
(7, '0707943187');
```

The Results tab shows the output of the query:

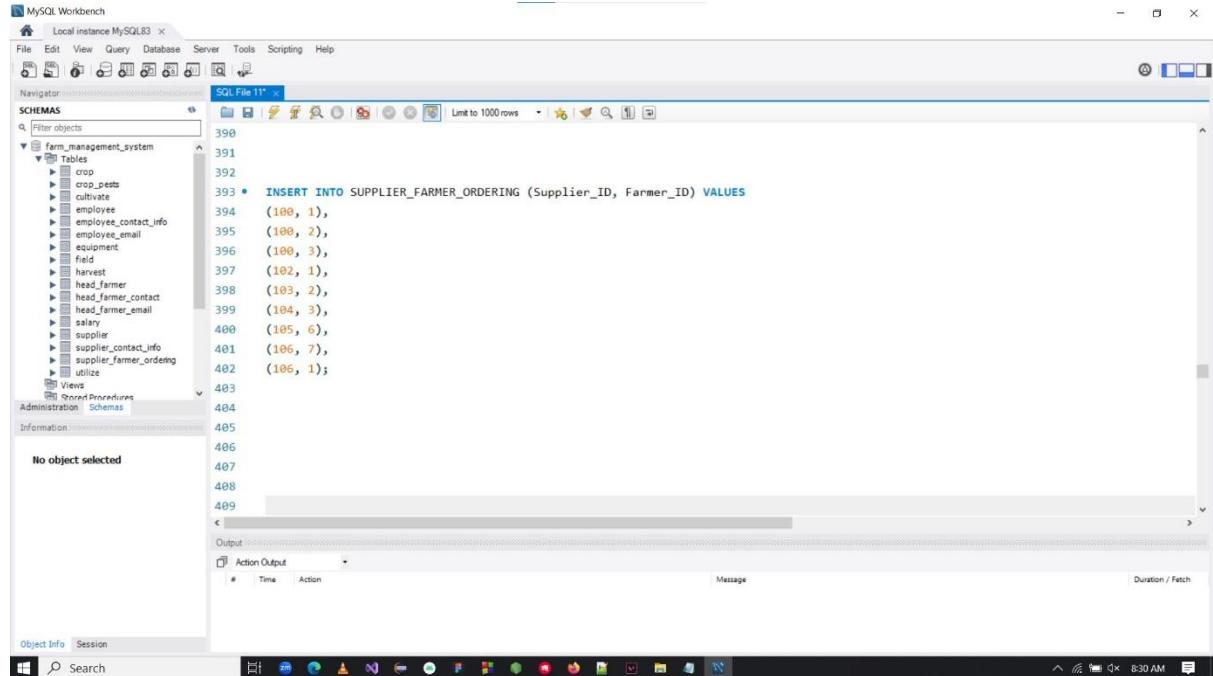
Farmer_ID	Contact_info
1	0706555701
2	0705528542
3	0784626573
4	0704196674
5	0789975075
6	0702851816
7	0707943187



The screenshot shows the MySQL 8.0 Command Line Client interface. The command line displays the following SQL query and its result:

```
mysql> select * from head_farmer_contact;
+-----+-----+
| Farmer_ID | Contact_info |
+-----+-----+
| 1 | 0706555701 |
| 2 | 0705528542 |
| 3 | 0784626573 |
| 4 | 0704196674 |
| 5 | 0789975075 |
| 6 | 0702851816 |
| 7 | 0707943187 |
+-----+-----+
7 rows in set (0.00 sec)
```

6. Insert into Supplier_farmer_ordering table

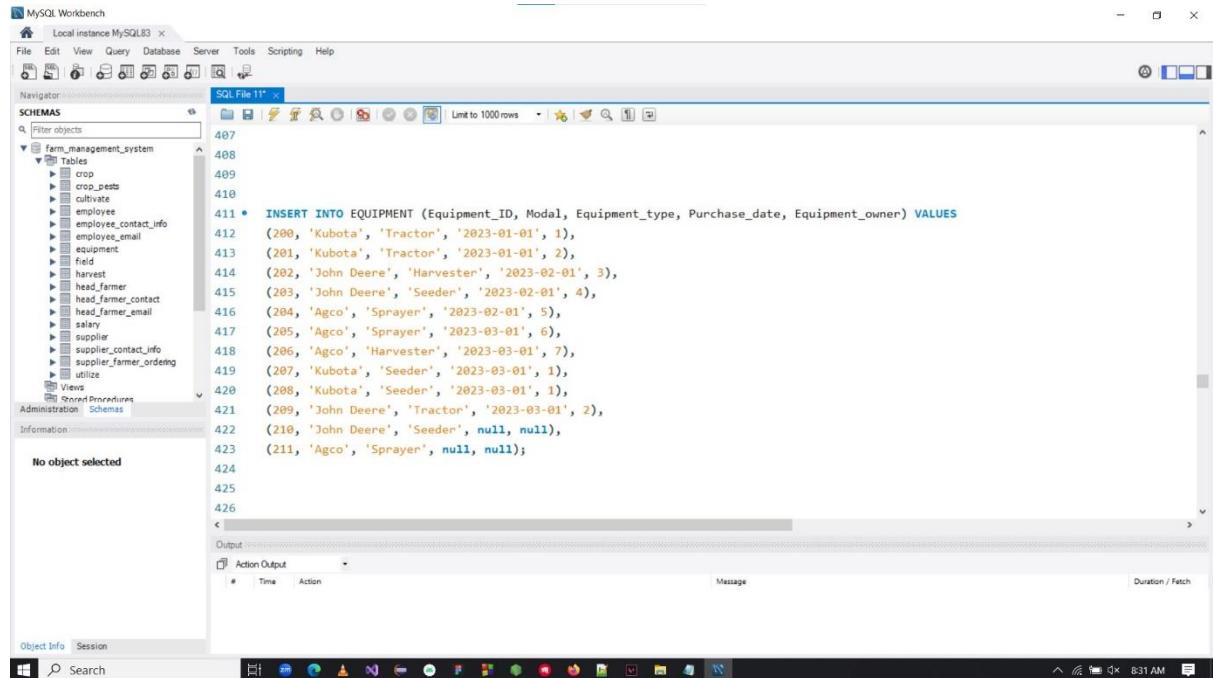


The screenshot shows the MySQL 8.0 Command Line Client window. The command 'select * from supplier_farmer_ordering;' was run, resulting in the following output:

```
+-----+-----+
| Supplier_ID | Farmer_ID |
+-----+-----+
|      100    |       1    |
|      102    |       1    |
|      106    |       1    |
|      100    |       2    |
|      103    |       2    |
|      100    |       3    |
|      104    |       3    |
|      105    |       6    |
|      106    |       7    |
+-----+-----+
```

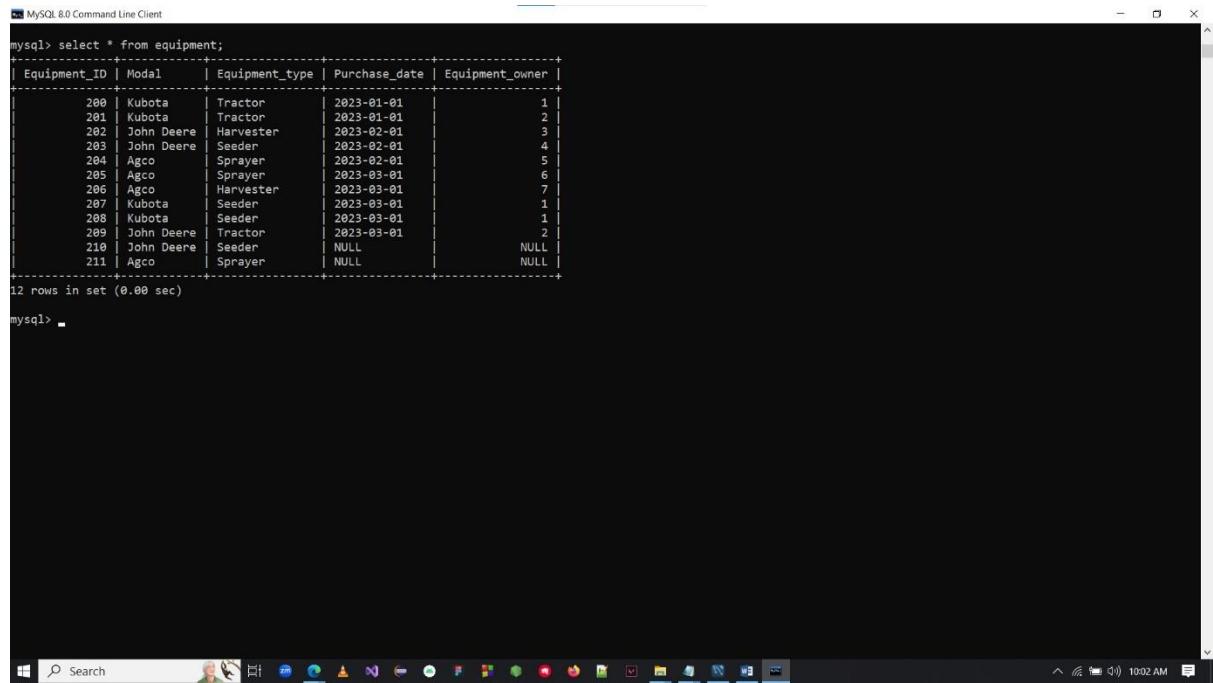
The status bar at the bottom shows the date and time as 10:01 AM.

7. Insert into Equipment table



The screenshot shows the MySQL Workbench interface. The SQL Editor tab is active, displaying a script named "SQL File 11". The script contains an `INSERT INTO` statement for the `EQUIPMENT` table, inserting 12 rows of data. The data includes equipment IDs from 200 to 211, modal names like Kubota, John Deere, and Agco, equipment types like Tractor, Harvester, Sprayer, and Seeder, purchase dates ranging from 2023-01-01 to 2023-03-01, and equipment owners numbered 1 through 7.

```
411 • INSERT INTO EQUIPMENT (Equipment_ID, Modal, Equipment_type, Purchase_date, Equipment_owner) VALUES
412 (200, 'Kubota', 'Tractor', '2023-01-01', 1),
413 (201, 'Kubota', 'Tractor', '2023-01-01', 2),
414 (202, 'John Deere', 'Harvester', '2023-02-01', 3),
415 (203, 'John Deere', 'Seeder', '2023-02-01', 4),
416 (204, 'Agco', 'Sprayer', '2023-02-01', 5),
417 (205, 'Agco', 'Sprayer', '2023-03-01', 6),
418 (206, 'Agco', 'Harvester', '2023-03-01', 7),
419 (207, 'Kubota', 'Seeder', '2023-03-01', 1),
420 (208, 'Kubota', 'Seeder', '2023-03-01', 1),
421 (209, 'John Deere', 'Tractor', '2023-03-01', 2),
422 (210, 'John Deere', 'Seeder', null, null),
423 (211, 'Agco', 'Sprayer', null, null);
```

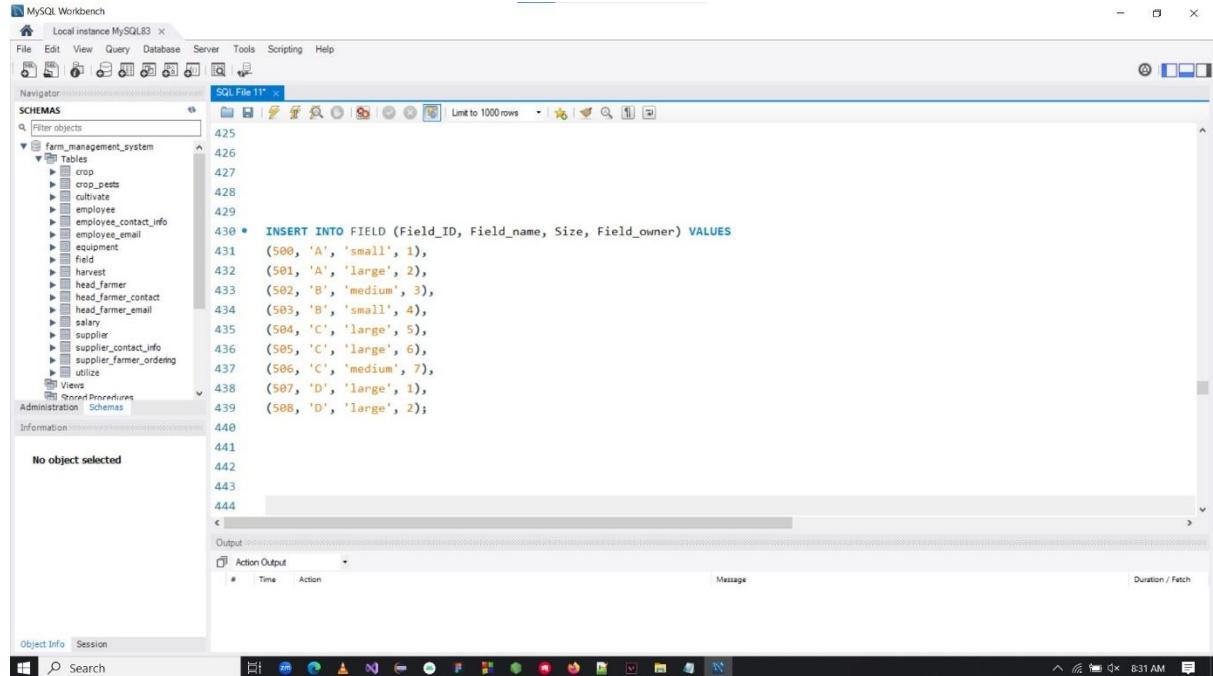


The screenshot shows the MySQL 8.0 Command Line Client window. The command `select * from equipment;` has been run, and the results are displayed in a table. The table has four columns: Equipment_ID, Modal, Equipment_type, and Purchase_date. The Equipment_owner column is present but appears to be empty or NULL. The data matches the insertions made in the previous step.

Equipment_ID	Modal	Equipment_type	Purchase_date	Equipment_owner
200	Kubota	Tractor	2023-01-01	1
201	Kubota	Tractor	2023-01-01	2
202	John Deere	Harvester	2023-02-01	3
203	John Deere	Seeder	2023-02-01	4
204	Agco	Sprayer	2023-02-01	5
205	Agco	Sprayer	2023-03-01	6
206	Agco	Harvester	2023-03-01	7
207	Kubota	Seeder	2023-03-01	1
208	Kubota	Seeder	2023-03-01	1
209	John Deere	Tractor	2023-03-01	2
210	John Deere	Seeder	NULL	NULL
211	Agco	Sprayer	NULL	NULL

12 rows in set (0.00 sec)

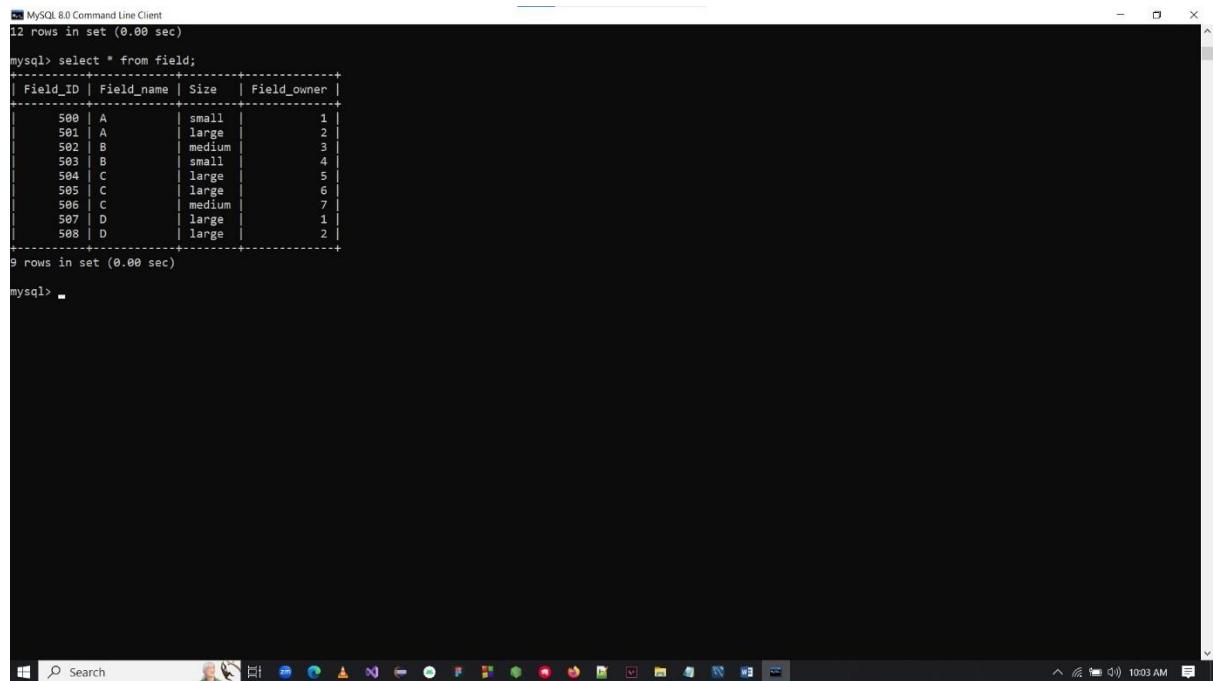
8. Insert into Field table



The screenshot shows the MySQL Workbench interface. The SQL Editor tab is active, displaying the following SQL code:

```
INSERT INTO FIELD (Field_ID, Field_name, Size, Field_owner) VALUES
(500, 'A', 'small', 1),
(501, 'A', 'large', 2),
(502, 'B', 'medium', 3),
(503, 'B', 'small', 4),
(504, 'C', 'large', 5),
(505, 'C', 'large', 6),
(506, 'C', 'medium', 7),
(507, 'D', 'large', 1),
(508, 'D', 'large', 2);
```

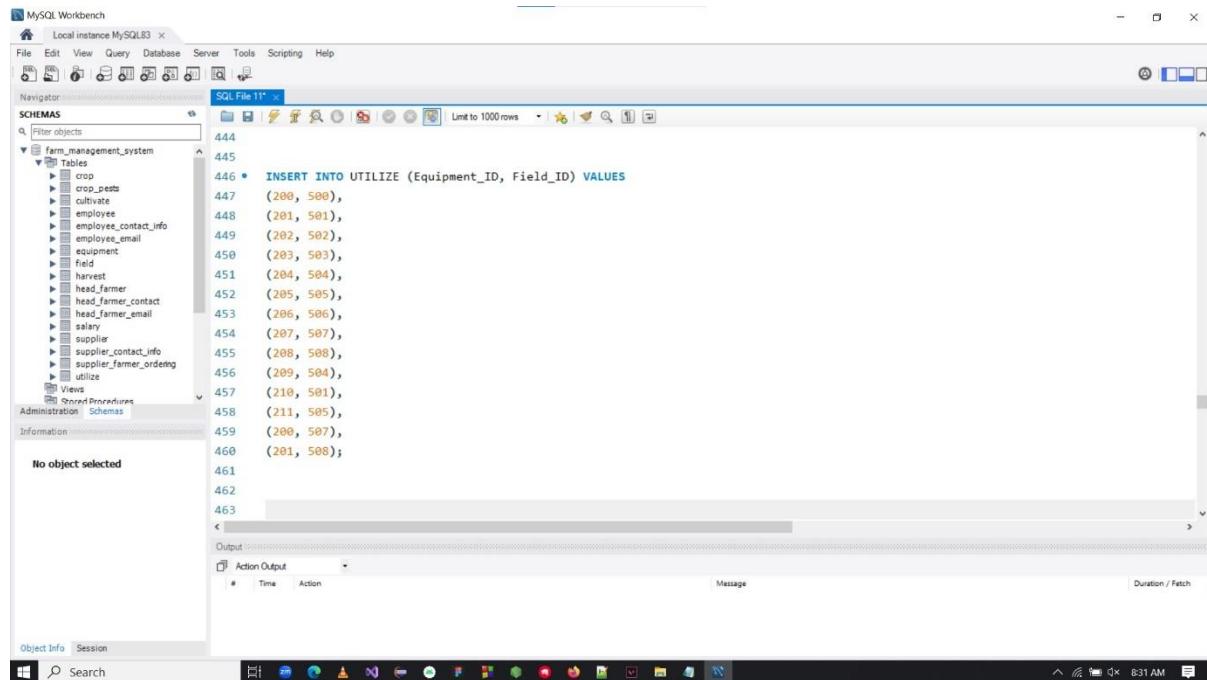
The Navigator pane on the left shows the schema structure of the 'farm_management_system' database, including tables like crop, crop_peds, cultivate, employee, etc.



The screenshot shows the MySQL 8.0 Command Line Client window. The command `select * from field;` has been run, and the results are displayed in a table:

Field_ID	Field_name	Size	Field_owner
500	A	small	1
501	A	large	2
502	B	medium	3
503	B	small	4
504	C	large	5
505	C	large	6
506	C	medium	7
507	D	large	1
508	D	large	2

9. Insert into Utilize table

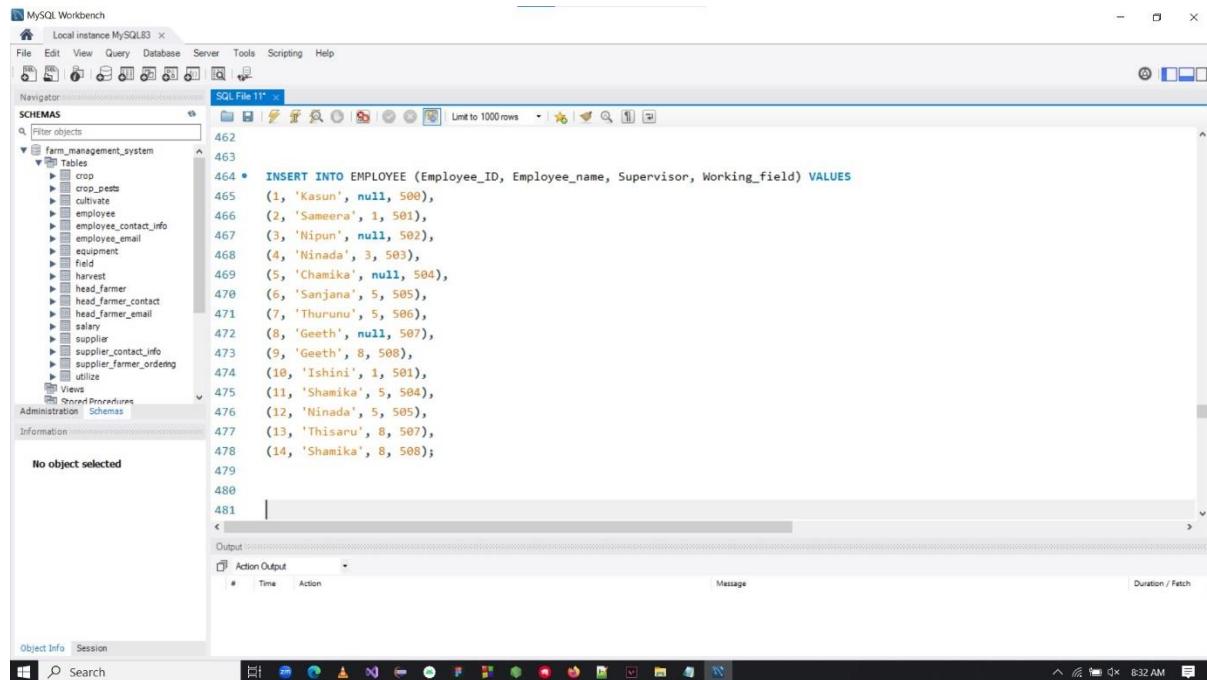


```
9 rows in set (0.00 sec)

mysql> select * from utilize;
+-----+-----+
| Equipment_ID | Field_ID |
+-----+-----+
| 200 | 500 |
| 201 | 501 |
| 210 | 501 |
| 202 | 502 |
| 203 | 503 |
| 204 | 504 |
| 209 | 504 |
| 205 | 505 |
| 211 | 505 |
| 206 | 506 |
| 200 | 507 |
| 207 | 507 |
| 201 | 508 |
| 208 | 508 |
+-----+-----+
14 rows in set (0.00 sec)

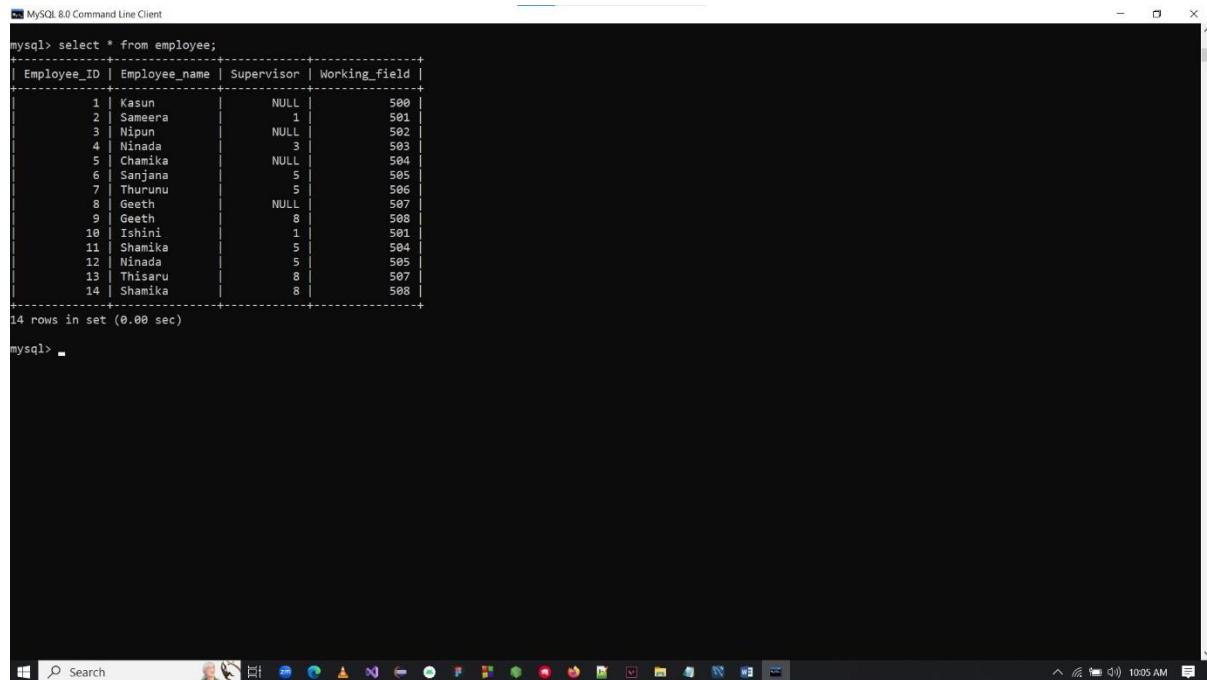
mysql>
```

10.Insert into Employee table



The screenshot shows the MySQL Workbench interface. The left pane displays the Navigator with the 'farm_management_system' schema selected. The right pane shows the SQL Editor with the following SQL code:

```
462
463
464 •  INSERT INTO EMPLOYEE (Employee_ID, Employee_name, Supervisor, Working_field) VALUES
465 (1, 'Kasun', null, 500),
466 (2, 'Sameera', 1, 501),
467 (3, 'Nipun', null, 502),
468 (4, 'Ninada', 3, 503),
469 (5, 'Chamika', null, 504),
470 (6, 'Sanjana', 5, 505),
471 (7, 'Thurunu', 5, 506),
472 (8, 'Geeth', null, 507),
473 (9, 'Geeth', 8, 508),
474 (10, 'Ishini', 1, 501),
475 (11, 'Shamika', 5, 504),
476 (12, 'Ninada', 5, 505),
477 (13, 'Thisaru', 8, 507),
478 (14, 'Shamika', 8, 508);
479
480
481
```

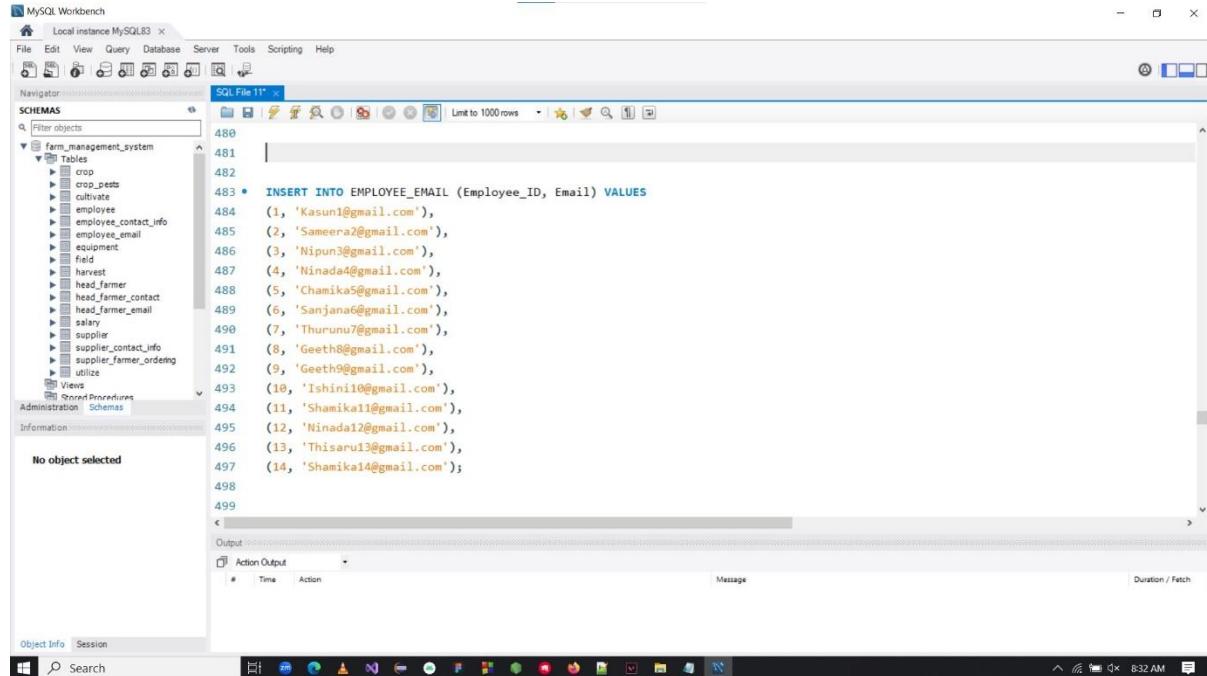


The screenshot shows the MySQL 8.0 Command Line Client window. The command `select * from employee;` has been run, and the results are displayed in a table:

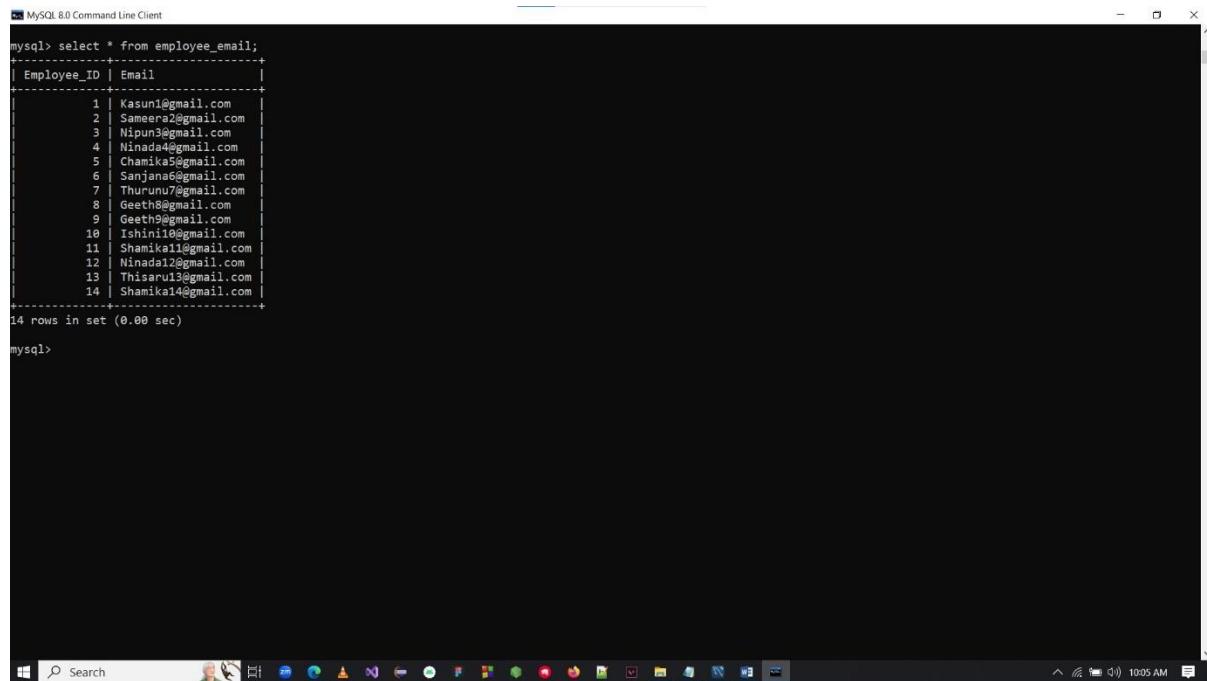
Employee_ID	Employee_name	Supervisor	Working_field
1	Kasun	NULL	500
2	Sameera	1	501
3	Nipun	NULL	502
4	Ninada	3	503
5	Chamika	NULL	504
6	Sanjana	5	505
7	Thurunu	5	506
8	Geeth	NULL	507
9	Geeth	8	508
10	Ishini	1	501
11	Shamika	5	504
12	Ninada	5	505
13	Thisaru	8	507
14	Shamika	8	508

14 rows in set (0.00 sec)

11.Insert into Employee_email table



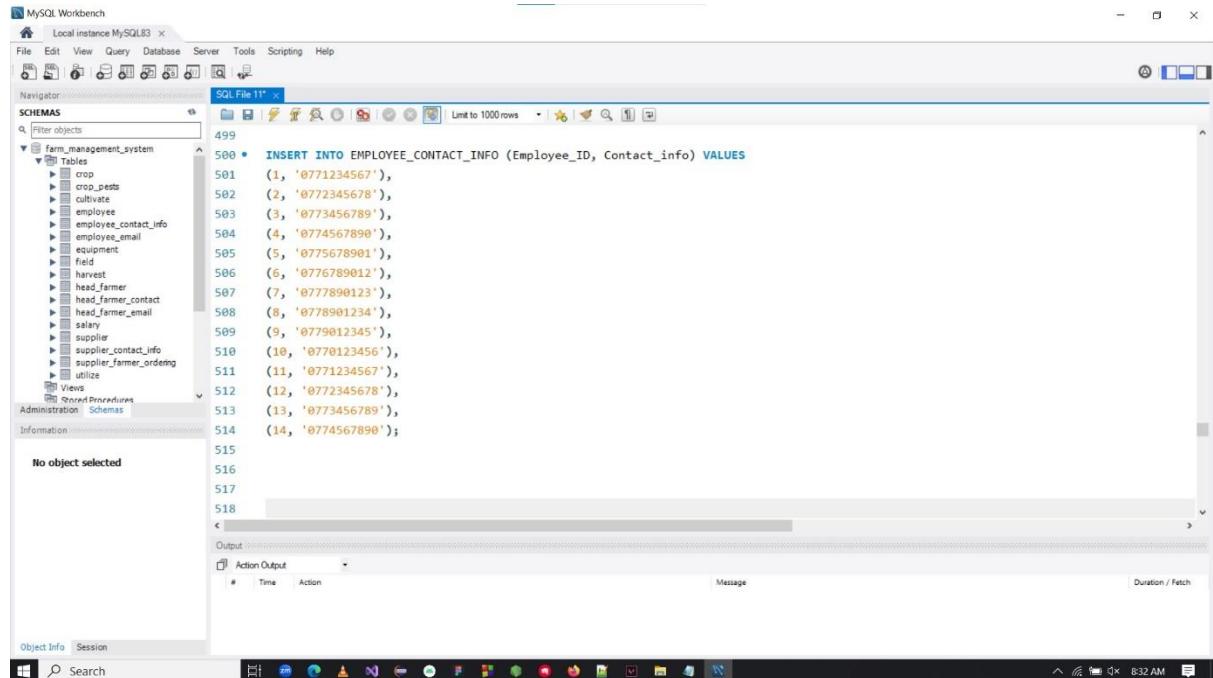
```
MySQL Workbench
Local instance MySQL83
File Edit View Query Database Server Tools Scripting Help
SQL File 11
schemas
farm_management_system
Tables
crop
crop_peds
cultivate
employee
employee_contact_info
employee_email
equipment
field
harvest
head_farmer
head_farmer_contact
head_farmer_email
salary
supplier
supplier_contact_info
supplier_farmer_ordering
utilize
Views
Stored Procedures
Administration Schemas
Information
No object selected
Object Info Session
Output
Action Output
# Time Action
Message Duration / Fetch
481
482
483 • INSERT INTO EMPLOYEE_EMAIL (Employee_ID, Email) VALUES
484 (1, 'Kasun1@gmail.com'),
485 (2, 'Sameera2@gmail.com'),
486 (3, 'Nipun3@gmail.com'),
487 (4, 'Ninada4@gmail.com'),
488 (5, 'Chamika5@gmail.com'),
489 (6, 'Sanjana6@gmail.com'),
490 (7, 'Thurunuru7@gmail.com'),
491 (8, 'Geeth8@gmail.com'),
492 (9, 'Geeth9@gmail.com'),
493 (10, 'Ishini10@gmail.com'),
494 (11, 'Shamikai11@gmail.com'),
495 (12, 'Ninada12@gmail.com'),
496 (13, 'Thisaru13@gmail.com'),
497 (14, 'Shamikai14@gmail.com');
498
499
```



```
mysql> select * from employee_email;
+-----+-----+
| Employee_ID | Email      |
+-----+-----+
| 1 | Kasun1@gmail.com |
| 2 | Sameera2@gmail.com |
| 3 | Nipun3@gmail.com |
| 4 | Ninada4@gmail.com |
| 5 | Chamika5@gmail.com |
| 6 | Sanjana6@gmail.com |
| 7 | Thurunuru7@gmail.com |
| 8 | Geeth8@gmail.com |
| 9 | Geeth9@gmail.com |
| 10 | Ishini10@gmail.com |
| 11 | Shamikai11@gmail.com |
| 12 | Ninada12@gmail.com |
| 13 | Thisaru13@gmail.com |
| 14 | Shamikai14@gmail.com |
+-----+-----+
14 rows in set (0.00 sec)

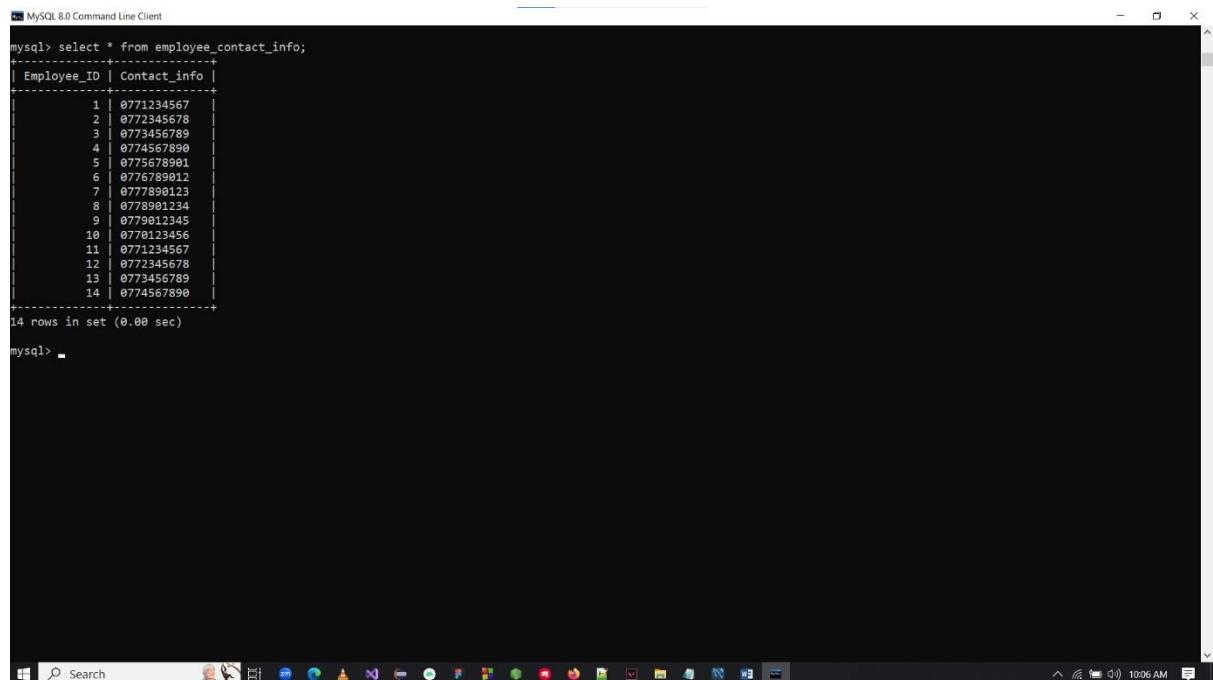
mysql>
```

12. Insert into Employee_contact_info table



The screenshot shows the MySQL Workbench interface. The left pane displays the 'Navigator' with the 'SCHEMAS' section expanded, showing the 'farm_management_system' schema with various tables like 'crop', 'employee', and 'employee_contact_info'. The right pane contains the 'SQL File 11' editor with the following SQL code:

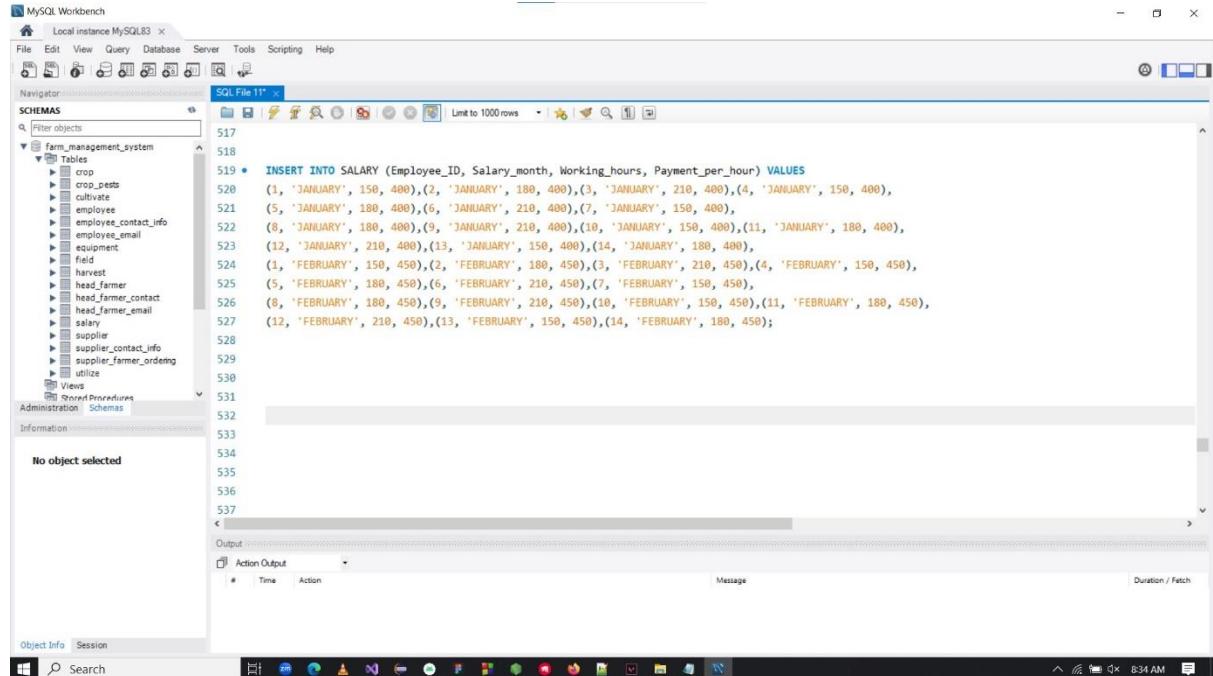
```
499
500 • INSERT INTO EMPLOYEE_CONTACT_INFO (Employee_ID, Contact_info) VALUES
501 (1, '0771234567'),
502 (2, '0772345678'),
503 (3, '0773456789'),
504 (4, '0774567890'),
505 (5, '0775678901'),
506 (6, '0776789012'),
507 (7, '0777890123'),
508 (8, '0778901234'),
509 (9, '07798012345'),
510 (10, '0770123456'),
511 (11, '0771234567'),
512 (12, '0772345678'),
513 (13, '0773456789'),
514 (14, '0774567890');
515
516
517
518
```



The screenshot shows the MySQL 8.0 Command Line Client window. The command prompt shows the following query and its results:

```
mysql> select * from employee_contact_info;
+-----+-----+
| Employee_ID | Contact_info |
+-----+-----+
| 1 | 0771234567 |
| 2 | 0772345678 |
| 3 | 0773456789 |
| 4 | 0774567890 |
| 5 | 0775678901 |
| 6 | 0776789012 |
| 7 | 0777890123 |
| 8 | 0778901234 |
| 9 | 0779012345 |
| 10 | 0770123456 |
| 11 | 0771234567 |
| 12 | 0772345678 |
| 13 | 0773456789 |
| 14 | 0774567890 |
+-----+-----+
14 rows in set (0.00 sec)
```

13.Insert into Salary table

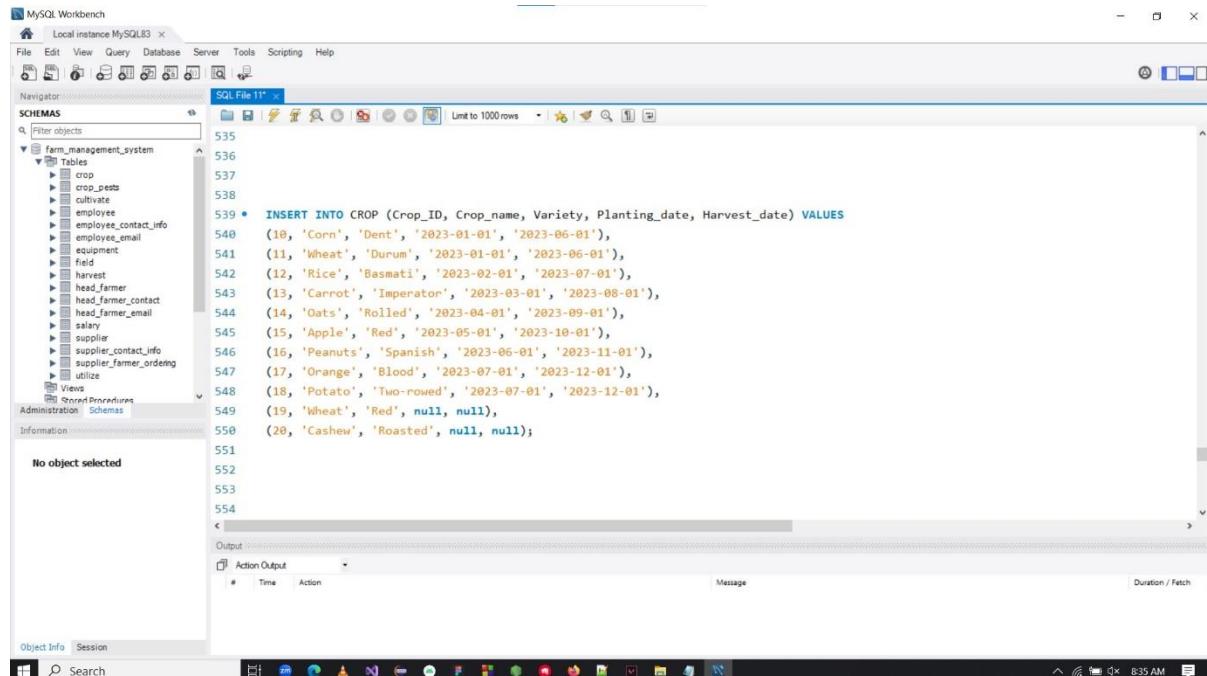


```
MySQL 8.0 Command Line Client
14 rows in set (0.00 sec)

mysql> select * from salary;
+-----+-----+-----+-----+-----+
| Employee_ID | Salary_month | Working_hours | Payment_per_hour | Monthly_salary |
+-----+-----+-----+-----+-----+
| 1 | FEBRUARY | 150 | 450 | 67500 |
| 1 | JANUARY | 150 | 400 | 60000 |
| 2 | FEBRUARY | 180 | 450 | 81000 |
| 2 | JANUARY | 180 | 400 | 72000 |
| 3 | FEBRUARY | 210 | 450 | 94500 |
| 3 | JANUARY | 210 | 400 | 84000 |
| 4 | FEBRUARY | 150 | 450 | 67500 |
| 4 | JANUARY | 150 | 400 | 60000 |
| 5 | FEBRUARY | 180 | 450 | 81000 |
| 5 | JANUARY | 180 | 400 | 72000 |
| 6 | FEBRUARY | 210 | 450 | 94500 |
| 6 | JANUARY | 210 | 400 | 84000 |
| 7 | FEBRUARY | 150 | 450 | 67500 |
| 7 | JANUARY | 150 | 400 | 60000 |
| 8 | FEBRUARY | 180 | 450 | 81000 |
| 8 | JANUARY | 180 | 400 | 72000 |
| 9 | FEBRUARY | 210 | 450 | 94500 |
| 9 | JANUARY | 210 | 400 | 84000 |
| 10 | FEBRUARY | 150 | 450 | 67500 |
| 10 | JANUARY | 150 | 400 | 60000 |
| 11 | FEBRUARY | 180 | 450 | 81000 |
| 11 | JANUARY | 180 | 400 | 72000 |
| 12 | FEBRUARY | 210 | 450 | 94500 |
| 12 | JANUARY | 210 | 400 | 84000 |
| 13 | FEBRUARY | 150 | 450 | 67500 |
| 13 | JANUARY | 150 | 400 | 60000 |
| 14 | FEBRUARY | 180 | 450 | 81000 |
| 14 | JANUARY | 180 | 400 | 72000 |
+-----+-----+-----+-----+-----+
28 rows in set (0.00 sec)

mysql>
```

14. Insert into Crop table

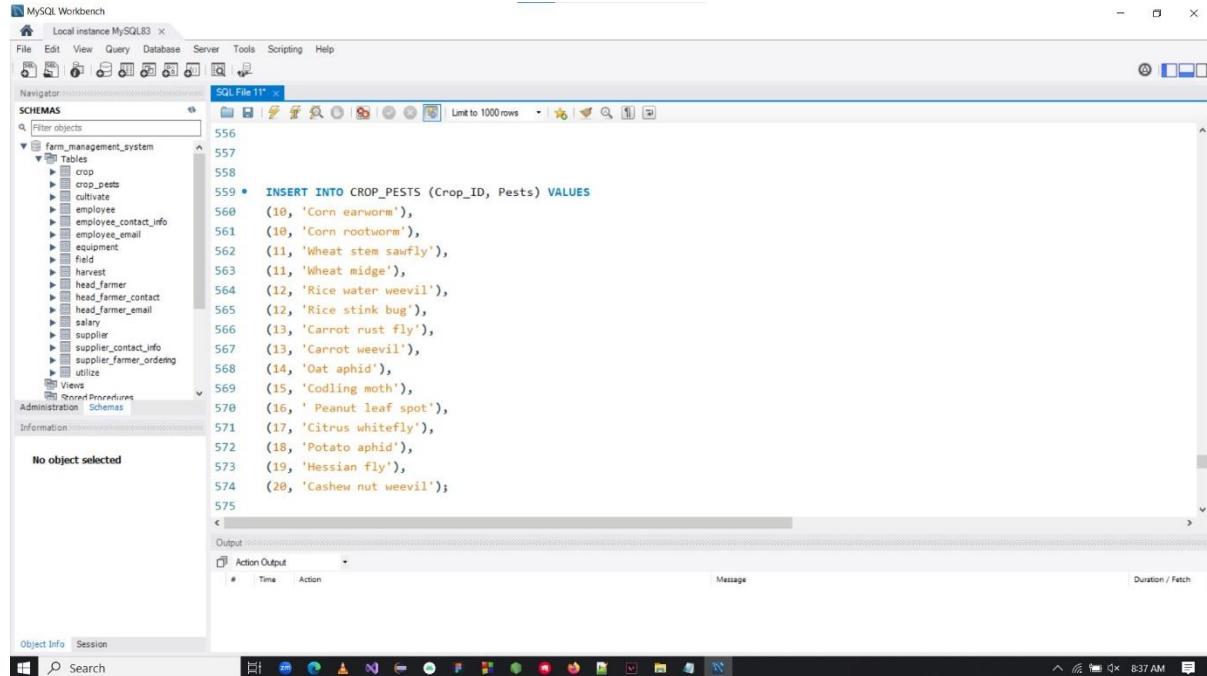


The screenshot shows the MySQL Command Line Client window. The command 'select * from crop;' has been run, and the results are displayed in a table:

Crop_ID	Crop_name	Variety	Planting_date	Harvest_date	Crop_age
10	Corn	Dent	2023-01-01	2023-06-01	151
11	Wheat	Durum	2023-01-01	2023-06-01	151
12	Rice	Basmati	2023-02-01	2023-07-01	150
13	Carrot	Imperator	2023-03-01	2023-08-01	153
14	Oats	Rolled	2023-04-01	2023-09-01	153
15	Apple	Red	2023-05-01	2023-10-01	153
16	Peanuts	Spanish	2023-06-01	2023-11-01	153
17	Orange	Blood	2023-07-01	2023-12-01	153
18	Potato	Two-rowed	2023-07-01	2023-12-01	153
19	Wheat	Red	NULL	NULL	NULL
20	Casew	Roasted	NULL	NULL	NULL

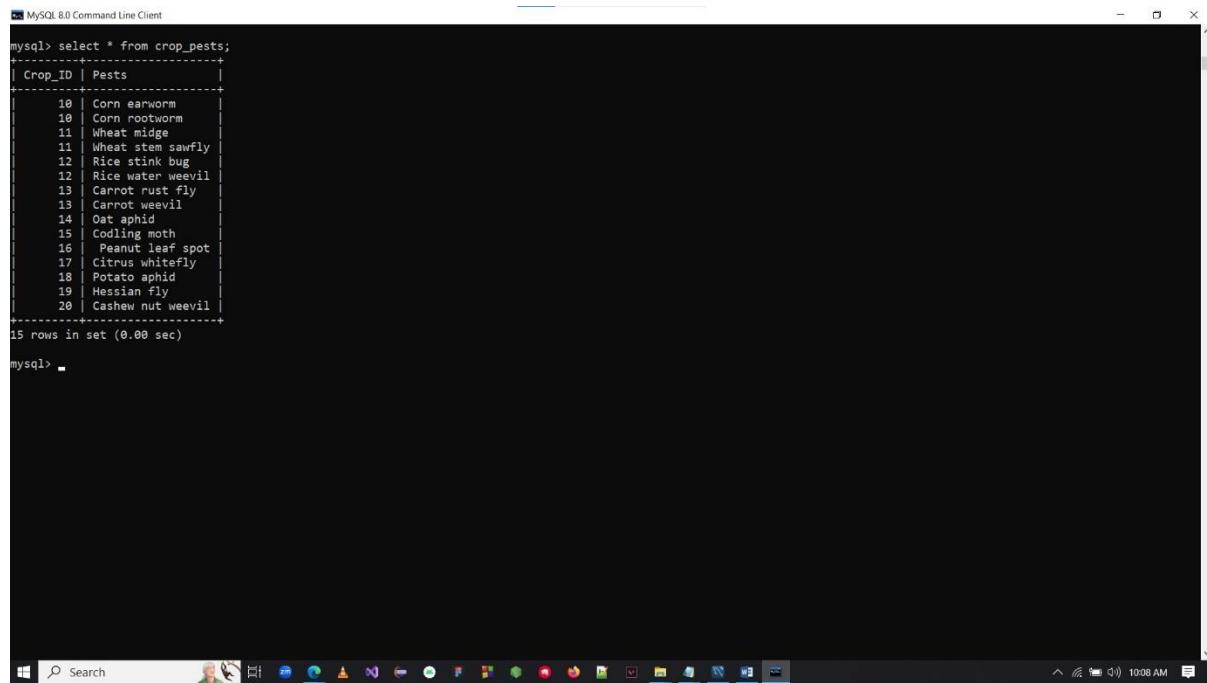
The status bar at the bottom indicates '10:08 AM'.

15.Insert into Crop_pests table



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, which includes the 'farm_management_system' schema with various tables like crop, crop_pests, cultivate, employee, etc. The main area is titled 'SQL File 11' and contains the following SQL code:

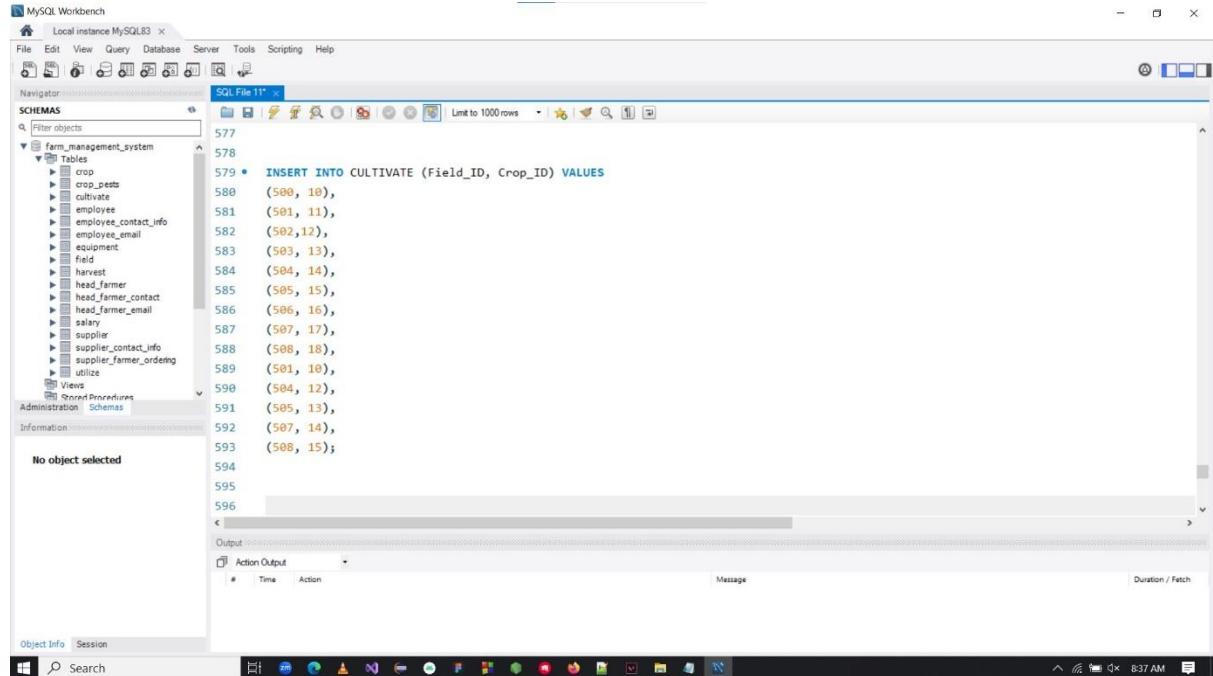
```
556
557
558
559 • INSERT INTO CROP_PESTS (Crop_ID, Pests) VALUES
560 (10, 'Corn earworm'),
561 (10, 'Corn rootworm'),
562 (11, 'Wheat stem sawfly'),
563 (11, 'Wheat midge'),
564 (12, 'Rice water weevil'),
565 (12, 'Rice stink bug'),
566 (13, 'Carrot rust fly'),
567 (13, 'Carrot weevil'),
568 (14, 'Oat aphid'),
569 (15, 'Coding moth'),
570 (16, 'Peanut leaf spot'),
571 (17, 'Citrus whitefly'),
572 (18, 'Potato aphid'),
573 (19, 'Hessian fly'),
574 (20, 'Caseweb nut weevil');
575
```



The screenshot shows the MySQL 8.0 Command Line Client window. The command line displays the following SQL query and its results:

```
mysql> select * from crop_pests;
+-----+-----+
| Crop_ID | Pests      |
+-----+-----+
|    10   | Corn earworm |
|    10   | Corn rootworm |
|    11   | Wheat midge   |
|    11   | Wheat stem sawfly |
|    12   | Rice stink bug |
|    12   | Rice water weevil |
|    13   | Carrot rust fly |
|    13   | Carrot weevil  |
|    14   | Oat aphid     |
|    15   | Coding moth   |
|    16   | Peanut leaf spot |
|    17   | Citrus whitefly |
|    18   | Potato aphid   |
|    19   | Hessian fly   |
|    20   | Caseweb nut weevil |
+-----+-----+
15 rows in set (0.00 sec)
```

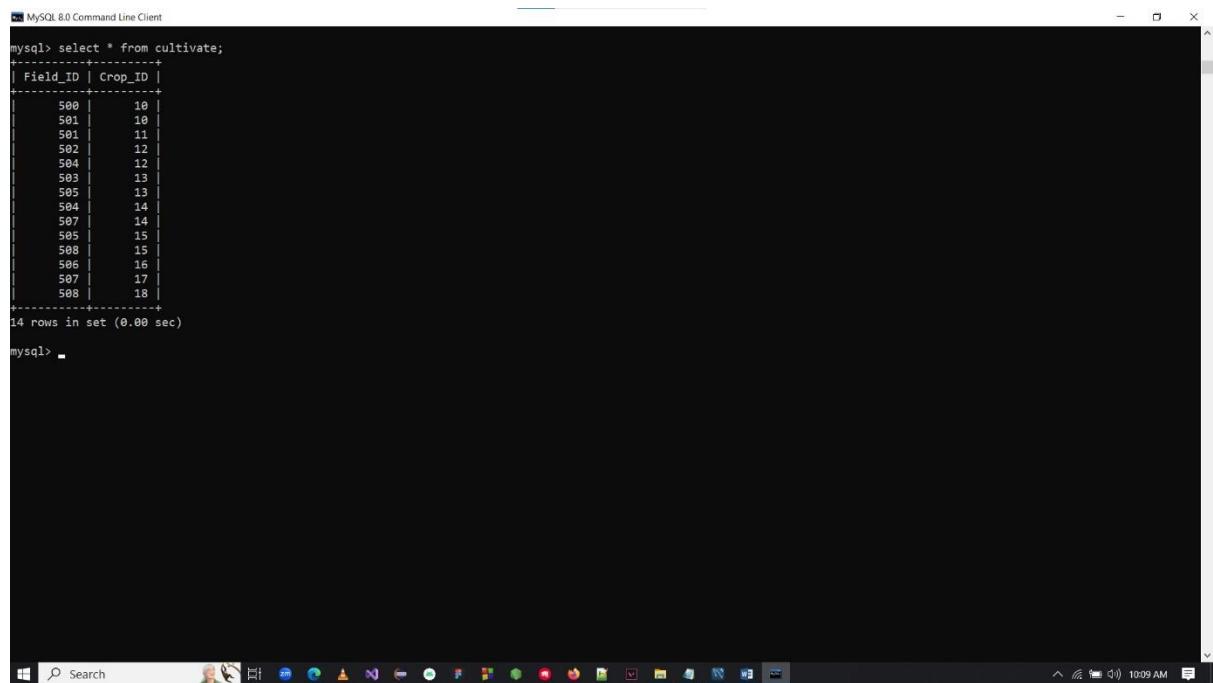
16.Insert into Cultivate table



The screenshot shows the MySQL Workbench interface. The left sidebar displays the schema 'farm_management_system' with various tables listed. The main area is titled 'SQL File 11' and contains the following SQL code:

```
577 • INSERT INTO CULTIVATE (Field_ID, Crop_ID) VALUES
578 (500, 10),
579 (501, 10),
580 (501, 11),
581 (502, 12),
582 (502, 12),
583 (503, 13),
584 (504, 14),
585 (505, 15),
586 (506, 16),
587 (507, 17),
588 (508, 18),
589 (501, 18),
590 (504, 12),
591 (505, 13),
592 (507, 14),
593 (508, 15);
594
595
596
```

The status bar at the bottom shows the date and time as 8:37 AM.

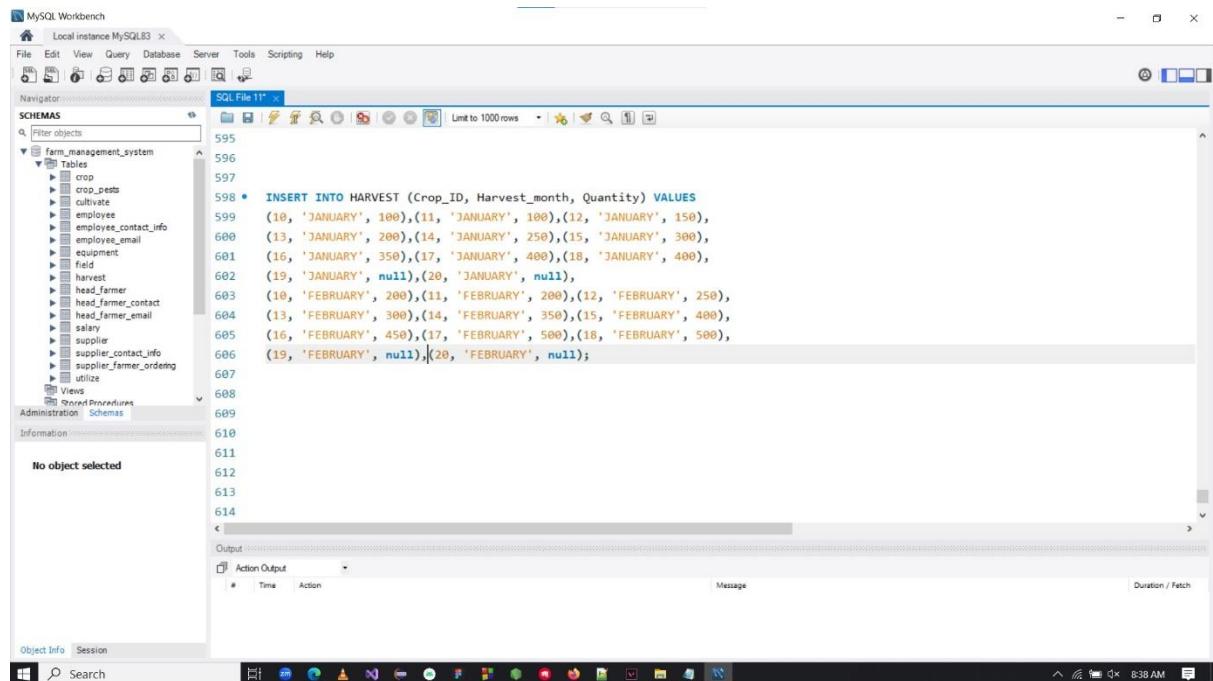


The screenshot shows the MySQL 8.0 Command Line Client window. The command prompt is at the top, followed by the output of a SELECT query:

```
mysql> select * from cultivate;
+-----+-----+
| Field_ID | Crop_ID |
+-----+-----+
| 500 | 10 |
| 501 | 10 |
| 501 | 11 |
| 502 | 12 |
| 504 | 12 |
| 503 | 13 |
| 505 | 13 |
| 504 | 14 |
| 507 | 14 |
| 505 | 15 |
| 508 | 15 |
| 506 | 16 |
| 507 | 17 |
| 508 | 18 |
+-----+-----+
14 rows in set (0.00 sec)
```

The status bar at the bottom shows the date and time as 10:09 AM.

17.Insert into Harvest table



The screenshot shows the MySQL Command Line Client window. The command 'select * from harvest;' has been run, resulting in the following output:

```
14 rows in set (0.00 sec)

mysql> select * from harvest;
+-----+-----+-----+
| Crop_ID | Harvest_month | Quantity |
+-----+-----+-----+
|    10 | FEBRUARY     |    200   |
|    10 | JANUARY      |    100   |
|    11 | FEBRUARY     |    200   |
|    11 | JANUARY      |    100   |
|    12 | FEBRUARY     |    250   |
|    12 | JANUARY      |    150   |
|    13 | FEBRUARY     |    300   |
|    13 | JANUARY      |    200   |
|    14 | FEBRUARY     |    350   |
|    14 | JANUARY      |    250   |
|    15 | FEBRUARY     |    400   |
|    15 | JANUARY      |    300   |
|    16 | FEBRUARY     |    450   |
|    16 | JANUARY      |    350   |
|    17 | FEBRUARY     |    500   |
|    17 | JANUARY      |    400   |
|    18 | FEBRUARY     |    500   |
|    18 | JANUARY      |    400   |
|    19 | FEBRUARY     |    NULL   |
|    19 | JANUARY      |    NULL   |
|    20 | FEBRUARY     |    NULL   |
|    20 | JANUARY      |    NULL   |
+-----+-----+-----+
22 rows in set (0.00 sec)
```

Update and Delete

1. Crop

The screenshot shows the MySQL Workbench interface. In the center, there is a SQL editor window titled "SQL File 11" containing the following SQL code:

```
782
783 • UPDATE CROP
784     SET Planting_date = '2023-02-15'
785     WHERE Crop_ID = 12;
786
787 • UPDATE CROP
788     SET Harvest_date = '2023-07-15'
789     WHERE Crop_ID = 13;
790
791 • DELETE FROM CROP
792     WHERE Crop_ID = 14;
793
794
795
796
797
798
799
```

Below the SQL editor, the "Output" pane displays the results of the executed statements:

#	Time	Action	Message	Duration / Fetch
11	11:03:07	UPDATE CROP SET Planting_date = '2023-02-15' WHERE Crop_ID = 12	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
12	11:03:10	UPDATE CROP SET Harvest_date = '2023-07-15' WHERE Crop_ID = 13	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
13	11:03:14	DELETE FROM CROP WHERE Crop_ID = 14	1 row(s) affected	0.000 sec

The screenshot shows the MySQL Command Line Client interface. The command "select * from crop;" has been run, resulting in the following output:

Crop_ID	Crop_name	Variety	Planting_date	Harvest_date	Crop_age
10	Corn	Dent	2023-01-01	2023-06-01	151
11	Wheat	Durum	2023-01-01	2023-06-01	151
12	Rice	Basmati	2023-02-15	2023-07-01	136
13	Carrot	Imperator	2023-03-01	2023-07-15	136
15	Apple	Red	2023-05-01	2023-10-01	153
16	Peanuts	Spanish	2023-06-01	2023-11-01	153
17	Orange	Blood	2023-07-01	2023-12-01	153
18	Potato	Two-rowed	2023-07-01	2023-12-01	153
19	Wheat	Red	NULL	NULL	NULL
20	Cashew	Roasted	NULL	NULL	NULL

2. Crop_pests

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

- Schemas:** farm_management_system
- Tables:** crop, crop_pests, cultivate, employee, employee_contact_info, employee_email, equipment, field, harvest, head_farmer, head_farmer_contact, head_farmer_email, salary, supplier, supplier_contact_info, supplier_farmer_ordering, utility.
- SQL File 11:** Contains the following SQL code:

```
794
795 • UPDATE CROP_PESTS
796   SET Pests = 'Corn borer'
797   WHERE Crop_ID = 10 AND Pests = 'Corn earworm';
798
799 • UPDATE CROP_PESTS
800   SET Pests = 'Rice blast'
801   WHERE Crop_ID = 12 AND Pests = 'Rice stink bug';
802
803 • DELETE FROM CROP_PESTS
804   WHERE Crop_ID = 15;
805
806
807
808
809
810
811
```

- Action Output:** Shows the results of the executed queries:

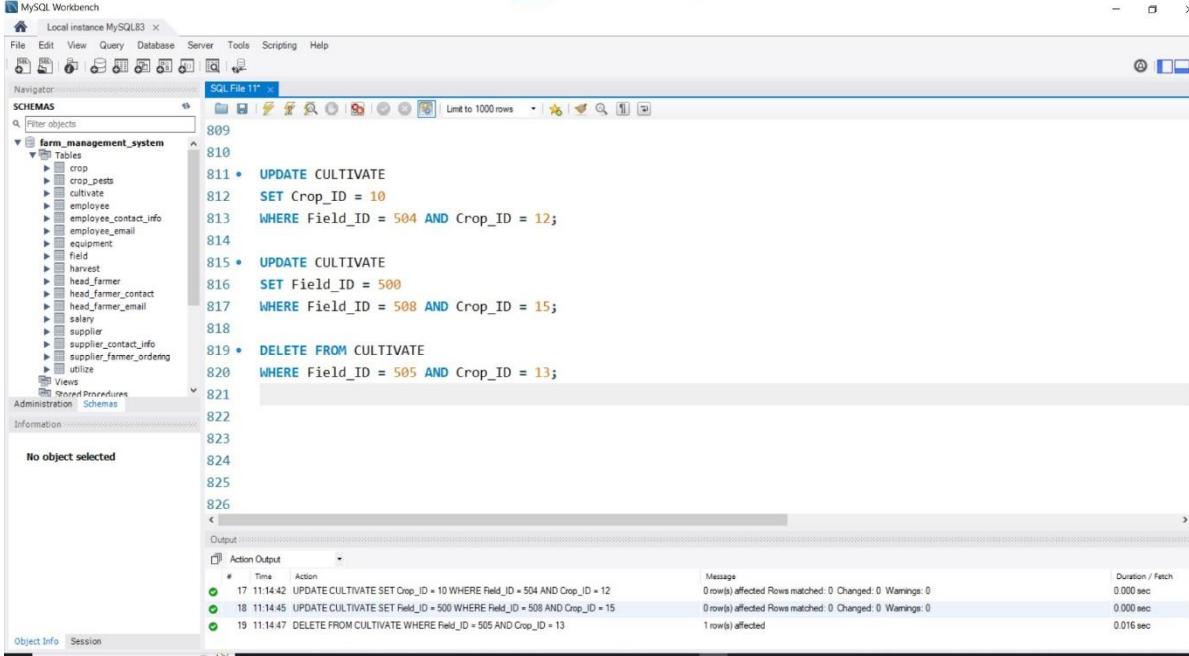
Action	Time	Message	Duration / Fetch
14 11:10:13 UPDATE CROP_PESTS SET Pests = 'Corn borer' WHERE Crop_ID = 10 AND Pests = 'Corn earworm'		1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
15 11:10:16 UPDATE CROP_PESTS SET Pests = 'Rice blast' WHERE Crop_ID = 12 AND Pests = 'Rice stink bug'		1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
16 11:10:20 DELETE FROM CROP_PESTS WHERE Crop_ID = 15		1 row(s) affected	0.047 sec

The screenshot shows the MySQL Command Line Client output:

```
mysql> select * from crop_pests;
+-----+-----+
| Crop_ID | Pests |
+-----+-----+
| 10     | Corn borer          |
| 10     | Corn rootworm       |
| 11     | Wheat midge          |
| 11     | Wheat stem sawfly    |
| 12     | Rice blast           |
| 12     | Rice water weevil    |
| 13     | Carrot rust fly      |
| 13     | Carrot weevil         |
| 16     | Peanut leaf spot     |
| 17     | Citrus whitefly       |
| 18     | Potato aphid          |
| 19     | Hessian fly           |
| 20     | Cashew nut weevil    |
+-----+-----+
13 rows in set (0.00 sec)

mysql>
```

3. Cultivate

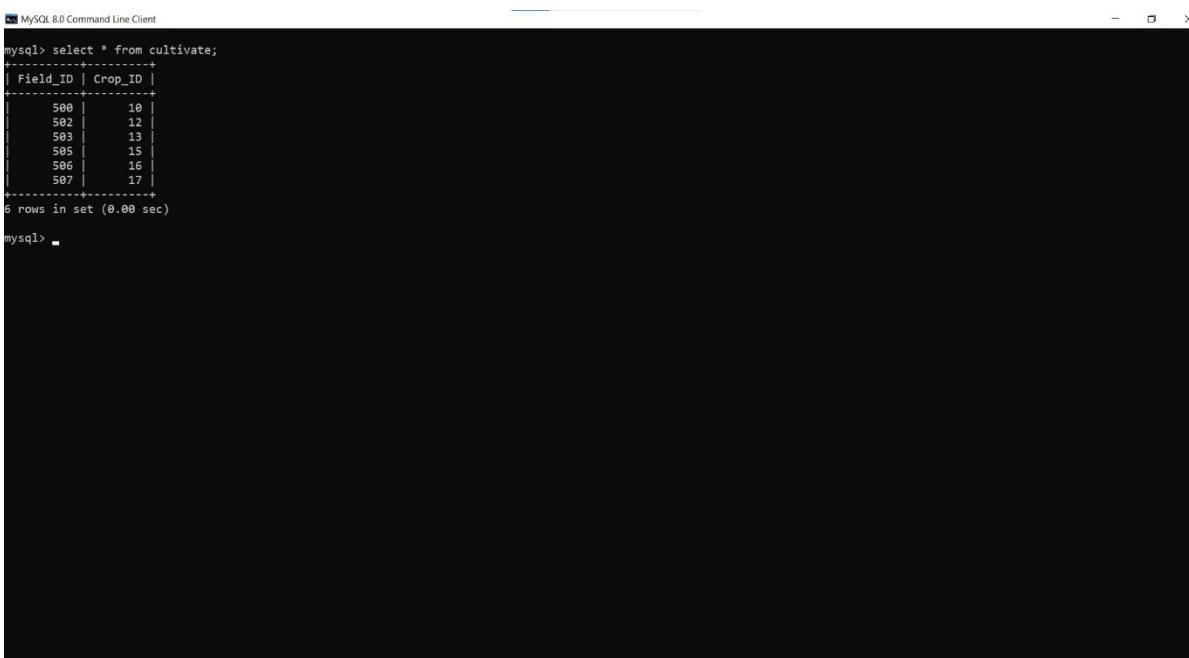


The screenshot shows the MySQL Workbench interface with a script editor containing SQL code and an output window showing execution results.

```
809
810 • UPDATE CULTIVATE
811 SET Crop_ID = 10
812 WHERE Field_ID = 504 AND Crop_ID = 12;
813
814 • UPDATE CULTIVATE
815 SET Field_ID = 500
816 WHERE Field_ID = 508 AND Crop_ID = 15;
817
818 • DELETE FROM CULTIVATE
819 WHERE Field_ID = 505 AND Crop_ID = 13;
820
821
822
823
824
825
826
```

Output:

#	Time	Action	Message	Duration / Fetch
17	11:14:42	UPDATE CULTIVATE SET Crop_ID = 10 WHERE Field_ID = 504 AND Crop_ID = 12	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
18	11:14:45	UPDATE CULTIVATE SET Field_ID = 500 WHERE Field_ID = 508 AND Crop_ID = 15	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
19	11:14:47	DELETE FROM CULTIVATE WHERE Field_ID = 505 AND Crop_ID = 13	1 row(s) affected	0.016 sec



```
mysql> select * from cultivate;
+-----+-----+
| Field_ID | Crop_ID |
+-----+-----+
| 500 | 10 |
| 502 | 12 |
| 503 | 13 |
| 505 | 15 |
| 506 | 16 |
| 507 | 17 |
+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

4. Employee

The screenshot shows the MySQL Workbench interface. In the Navigator pane, the schema 'farm_management_system' is selected, displaying tables like crop, employee, and salary. The SQL Editor pane contains the following code:

```
729 • UPDATE EMPLOYEE
730 SET Employee_name = 'Sandun'
731 WHERE Employee_ID = 2;
732
733 • UPDATE EMPLOYEE
734 SET Working_field = 506
735 WHERE Employee_ID = 12;
736
737
738 • DELETE FROM EMPLOYEE
739 WHERE Employee_ID = 11;
740
741
742
743
744
745
746
```

The Output pane shows the results of the last query:

#	Action	Time	Message	Duration / Fetch
1	DELETE FROM EMPLOYEE WHERE Employee_ID = 11;	10:53:05	0 row(s) affected	0.000 sec

The screenshot shows the MySQL 8.0 Command Line Client. The command `select * from employee;` was run, resulting in the following output:

```
mysql> select * from employee;
+-----+-----+-----+-----+
| Employee_ID | Employee_name | Supervisor | Working_field |
+-----+-----+-----+-----+
| 1 | Kasun | NULL | 500 |
| 3 | Nipun | NULL | 502 |
| 4 | Ninada | 3 | 503 |
| 6 | Sanjana | NULL | 505 |
| 7 | Thurunu | NULL | 506 |
| 8 | Geeth | NULL | 507 |
| 12 | Ninada | NULL | 506 |
| 13 | Thisaru | 8 | 507 |
+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

mysql>

5. Employee_contact_info

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' section, the 'farm_management_system' schema is selected, displaying various tables like crop, field, harvest, head_farmer, head_farmer_contact, head_farmer_email, salary, supplier, supplier_contact_info, supplier_farmer_ordering, and utilize. The main pane contains a SQL editor titled 'SQL File 1' with the following code:

```
758
759 • UPDATE EMPLOYEE_CONTACT_INFO
760 SET Contact_info = '0772345670'
761 WHERE Employee_ID = 2;
762
763 • UPDATE EMPLOYEE_CONTACT_INFO
764 SET Contact_info = '0773456780'
765 WHERE Employee_ID = 3;
766
767 • DELETE FROM EMPLOYEE_CONTACT_INFO
768 WHERE Employee_ID = 8;
769
770
771
772
773
774
```

Below the SQL editor is an 'Output' panel showing the results of the executed statements:

#	Time	Action	Message	Duration / Fetch
5	10:57:17	UPDATE EMPLOYEE_CONTACT_INFO SET Contact_info = '0772345670' WHERE Employee_ID = 2;	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
6	10:57:20	UPDATE EMPLOYEE_CONTACT_INFO SET Contact_info = '0773456780' WHERE Employee_ID = 3;	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
7	10:57:23	DELETE FROM EMPLOYEE_CONTACT_INFO WHERE Employee_ID = 8;	1 row(s) affected	0.016 sec

The screenshot shows the MySQL 8.0 Command Line Client. The command prompt is at the top, followed by the output of a SELECT query:

```
+-----+
| Employee_ID | Contact_info |
+-----+
| 1           | 0771234567 |
| 3           | 0773456788 |
| 4           | 0774567890 |
| 6           | 0776789812 |
| 7           | 0777890123 |
| 12          | 0772345678 |
| 13          | 0773456789 |
+-----+
```

Below the table, the message '7 rows in set (0.00 sec)' is displayed. The MySQL prompt 'mysql>' appears at the bottom.

6. Employee_email

The screenshot shows the MySQL Workbench interface. In the Navigator pane, the schema 'farm_management_system' is selected, displaying various tables like crop, crop_pests, cultivate, employee, etc. The SQL Editor pane contains the following SQL code:

```
742
743
744 • UPDATE EMPLOYEE_EMAIL
745 SET Email = 'Samiya2'
746 WHERE Employee_ID = 2;
747
748 • UPDATE EMPLOYEE_EMAIL
749 SET Email = 'NipunDilshan3'
750 WHERE Employee_ID = 3;
751
752
753 • DELETE FROM EMPLOYEE_EMAIL
754 WHERE Employee_ID = 7;
755
756
757
758
759
```

The Output pane shows the results of the executed statements:

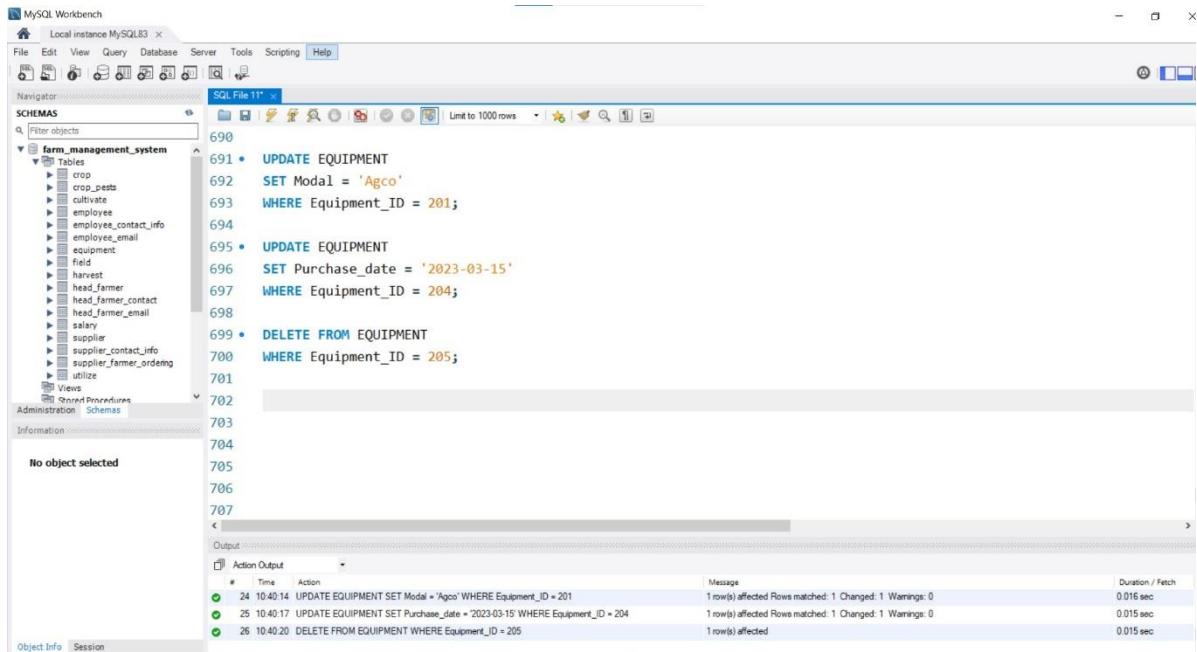
#	Action	Time	Message	Duration / Fetch
2	UPDATE EMPLOYEE_EMAIL	10:54:56	SET Email = 'Samiya2' WHERE Employee_ID = 2 0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
3	UPDATE EMPLOYEE_EMAIL	10:54:59	SET Email = 'NipunDilshan3' WHERE Employee_ID = 3 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
4	DELETE FROM EMPLOYEE_EMAIL	10:55:01	WHERE Employee_ID = 7 1 row(s) affected	0.016 sec

The screenshot shows the MySQL Command Line Client window. The command prompt is at the top, followed by the output of a SELECT query:

```
mysql> select * from employee_email;
+-----+-----+
| Employee_ID | Email |
+-----+-----+
| 1 | Kasuni@gmail.com |
| 3 | NipunDilshan3 |
| 4 | Nianda4@gmail.com |
| 6 | Sanjana6@gmail.com |
| 8 | Geeth8@gmail.com |
| 12 | Nianda12@gmail.com |
| 13 | Thisaru13@gmail.com |
+-----+-----+
7 rows in set (0.00 sec)
```

mysql>

7. Equipment

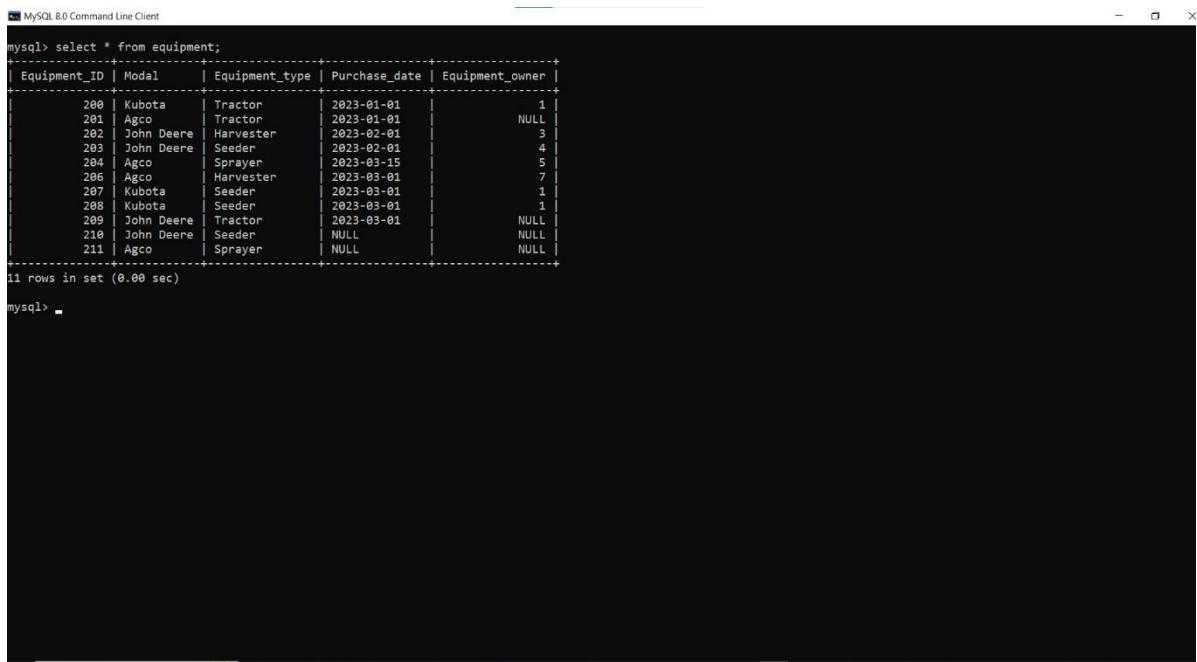


The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Shows the schema `farm_management_system` with tables: crop, crop_pests, cultivate, employee, employee_contact_info, employee_email, equipment, field, harvest, head_farmer, head_farmer_contact, head_farmer_email, salary, supplier, supplier_contact_info, supplier_farmer_ordering, utilize.
- SQL Editor:** Contains the following SQL code:

```
690 • UPDATE EQUIPMENT
691 SET Modal = 'Agco'
692 WHERE Equipment_ID = 201;
693
694 • UPDATE EQUIPMENT
695 SET Purchase_date = '2023-03-15'
696 WHERE Equipment_ID = 204;
697
698 • DELETE FROM EQUIPMENT
699 WHERE Equipment_ID = 205;
700
701
702
703
704
705
706
707
```
- Output Window:** Shows the results of the executed queries:

#	Time	Action	Message	Duration / Fetch
24	10:40:14	UPDATE EQUIPMENT SET Modal = 'Agco' WHERE Equipment_ID = 201	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
25	10:40:17	UPDATE EQUIPMENT SET Purchase_date = '2023-03-15' WHERE Equipment_ID = 204	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
26	10:40:20	DELETE FROM EQUIPMENT WHERE Equipment_ID = 205	1 row(s) affected	0.015 sec



The screenshot shows the MySQL Command Line Client with the following details:

- SQL Prompt:** mysql>
- Query:** `select * from equipment;`
- Result:** A table showing equipment data:

Equipment_ID	Modal	Equipment_type	Purchase_date	Equipment_owner
200	Kubota	Tractor	2023-01-01	1
201	Agco	Tractor	2023-01-01	NULL
202	John Deere	Harvester	2023-02-01	3
203	John Deere	Seeder	2023-02-01	4
204	Agco	Sprayer	2023-03-15	5
206	Agco	Harvester	2023-03-01	7
207	Kubota	Seeder	2023-03-01	1
208	Kubota	Seeder	2023-03-01	1
209	John Deere	Tractor	2023-03-01	NULL
210	John Deere	Seeder	NULL	NULL
211	Agco	Sprayer	NULL	NULL

11 rows in set (0.00 sec)

8. Field

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'SCHEMAS' section, the 'farm_management_system' schema is selected, displaying its tables: crop, crop_pest, cultivate, employee, employee_contact_info, employee_email, equipment, field, harvest, head_farmer, head_farmer_contact, head_farmer_email, salary, supplier, supplier_contact_info, supplier_farmer_ordering, and utility. The main pane contains the following SQL code:

```
703
704
705 • UPDATE FIELD
706 SET Size = 'Small'
707 WHERE Field_ID = 502;
708
709 • UPDATE FIELD
710 SET Size = 'Medium'
711 WHERE Field_ID = 503;
712
713
714 • DELETE FROM FIELD
715 WHERE Field_ID = 504;
716
717
718
719
720
```

The 'Output' tab at the bottom shows the results of the executed queries:

#	Time	Action	Message	Duration / Fetch
27	10:42:22	UPDATE FIELD SET Size = 'Small' WHERE Field_ID = 502	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
28	10:42:26	UPDATE FIELD SET Size = 'Medium' WHERE Field_ID = 503	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
29	10:42:29	DELETE FROM FIELD WHERE Field_ID = 504	1 row(s) affected	0.015 sec

The screenshot shows the MySQL Command Line Client window. The command entered is:

```
mysql> select * from field;
```

The output displays the following table:

Field_ID	Field_name	Size	Field_owner
500	A	small	1
502	B	Small	3
503	B	Medium	4
505	C	large	6
506	C	medium	7
507	D	large	1

At the bottom, the message '6 rows in set (0.00 sec)' is displayed.

9. Harvest

The screenshot shows the MySQL Workbench interface with a query editor window titled "SQL File 11". The code consists of three statements:

```
821
822
823 • UPDATE HARVEST
824     SET Quantity = 250
825     WHERE Crop_ID = 10 AND Harvest_month = 'JANUARY';
826
827 • UPDATE HARVEST
828     SET Quantity = 250
829     WHERE Crop_ID = 11 AND Harvest_month = 'FEBRUARY';
830
831 • DELETE FROM HARVEST
832     WHERE Crop_ID = 12;
```

The "Output" pane shows the results of the executed statements:

Action	Time	Message	Duration / Fetch
20 11:17:28	UPDATE HARVEST SET Quantity = 250 WHERE Crop_ID = 10 AND Harvest_month = 'JANUARY'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
21 11:17:31	UPDATE HARVEST SET Quantity = 250 WHERE Crop_ID = 11 AND Harvest_month = 'FEBRUARY'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
22 11:17:34	DELETE FROM HARVEST WHERE Crop_ID = 12	2 row(s) affected	0.016 sec

The screenshot shows the MySQL Command Line Client output. A "select * from harvest;" query was run, resulting in 18 rows of data:

Crop_ID	Harvest_month	Quantity
10	FEBRUARY	200
10	JANUARY	250
11	FEBRUARY	250
11	JANUARY	100
13	FEBRUARY	300
13	JANUARY	200
15	FEBRUARY	400
15	JANUARY	300
16	FEBRUARY	450
16	JANUARY	350
17	FEBRUARY	500
17	JANUARY	400
18	FEBRUARY	500
18	JANUARY	400
19	FEBRUARY	NULL
19	JANUARY	NULL
20	FEBRUARY	NULL
20	JANUARY	NULL

10.Head_farmer

The screenshot shows the MySQL Workbench interface with a SQL file named "SQL File 11". The code consists of several UPDATE and DELETE statements targeting the "HEAD_FARMER" table in the "farm_management_system" schema. The queries are numbered 637 through 653. The output pane shows the results of these queries, indicating successful execution with 1 row affected and 0 warnings.

```
636
637 • UPDATE HEAD_FARMER
638 SET Zip_code = '00101'
639 WHERE Farmer_ID = 4;
640
641 • UPDATE HEAD_FARMER
642 SET Zip_code = '01101'
643 WHERE Farmer_ID = 6;
644
645 • DELETE FROM HEAD_FARMER
646 WHERE Farmer_ID = 2;
647
648
649
650
651
652
653
```

#	Time	Action	Message	Duration / Fetch
12	10:24:00	UPDATE HEAD_FARMER SET Zip_code = '00101' WHERE Farmer_ID = 4	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
13	10:24:04	UPDATE HEAD_FARMER SET Zip_code = '01101' WHERE Farmer_ID = 6	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
14	10:24:07	DELETE FROM HEAD_FARMER WHERE Farmer_ID = 2	1 row(s) affected	0.015 sec

The screenshot shows the MySQL Command Line Client interface. A SELECT query is run against the "head_farmer" table, displaying the results in a tabular format. The output shows 6 rows in the set, each containing Farmer_ID, Farmer_name, Street, City, Province, and Zip_code. The data includes entries for Dinesh, Kusal, Pathum, Charith, and Sahan.

```
mysql> select * from head_farmer;
+-----+-----+-----+-----+-----+-----+
| Farmer_ID | Farmer_name | Street | City | Province | Zip_code |
+-----+-----+-----+-----+-----+-----+
| 1 | Dinesh | Main St | Colombo | Western | 00100 |
| 3 | Kusal | 1st cross | Colombo | Western | 00100 |
| 4 | Pathum | 2nd cross | Colombo | Western | 00101 |
| 5 | Pathum | Main St | Kandy | Central | 20000 |
| 6 | Charith | Main St | Galle | Southern | 01101 |
| 7 | Sahan | Main St | Mathara | Southern | 01100 |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

11. Head_farmer_contact

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Shows the schema `farm_management_system` with tables: crop, crop_pests, cultivate, employee, employee_contact_info, employee_email, equipment, field, harvest, head_farmer, head_farmer_contact, head_farmer_email, salary, supplier, supplier_contact_info, supplier_farmer_ordering, utilize.
- SQL Editor:** Contains the following SQL code:

```
661
662
663
664 • UPDATE HEAD_FARMER_CONTACT
665     SET Contact_info = '0706564704'
666     WHERE Farmer_ID = 1;
667
668 • UPDATE HEAD_FARMER_CONTACT
669     SET Contact_info = '0704196666'
670     WHERE Farmer_ID = 4;
671
672
673 • DELETE FROM HEAD_FARMER_CONTACT
674     WHERE Farmer_ID = 7;
675
676
677
678
```
- Output Window:** Shows the results of the executed statements:

#	Time	Action	Message	Duration / Fetch
18	10:30:28	UPDATE HEAD_FARMER_CONTACT SET Contact_info = '0706564704' WHERE Farmer_ID = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.04 sec
19	10:30:31	UPDATE HEAD_FARMER_CONTACT SET Contact_info = '0704196666' WHERE Farmer_ID = 4	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
20	10:30:34	DELETE FROM HEAD_FARMER_CONTACT WHERE Farmer_ID = 7	1 row(s) affected	0.016 sec

The screenshot shows the MySQL 8.0 Command Line Client with the following details:

- Output:** The command `select * from head_farmer_contact;` was run, resulting in the following output:

```
5 rows in set (0.00 sec)

mysql> select * from head_farmer_contact;
+-----+-----+
| Farmer_ID | Contact_info |
+-----+-----+
| 1 | 0706564704 |
| 3 | 0704196666 |
| 4 | 0704196666 |
| 5 | 0709975075 |
| 6 | 0702851816 |
+-----+-----+
5 rows in set (0.00 sec)
```

12.Head_farmer_email

The screenshot shows the MySQL Workbench interface. In the Navigator pane, the schema 'farm_management_system' is selected, displaying tables like crop, crop_pests, cultivate, employee, etc. The SQL Editor pane contains the following SQL code:

```
648 • UPDATE HEAD_FARMER_EMAIL  
649 SET Email = 'Dinesh70@gmail.com'  
650 WHERE Farmer_ID = 1;  
653  
654 • UPDATE HEAD_FARMER_EMAIL  
655 SET Email = 'Pathum26@gmail.com'  
656 WHERE Farmer_ID = 4;  
657  
658 • DELETE FROM HEAD_FARMER_EMAIL  
659 WHERE Farmer_ID = 6;  
660  
661  
662  
663  
664  
665
```

The Output pane shows the results of the executed statements:

#	Action	Time	Message	Duration / Fetch
15	UPDATE HEAD_FARMER_EMAIL	10:26:47	SET Email = 'Dinesh70@gmail.com' WHERE Farmer_ID = 1 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
16	UPDATE HEAD_FARMER_EMAIL	10:26:50	SET Email = 'Pathum26@gmail.com' WHERE Farmer_ID = 4 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
17	DELETE FROM HEAD_FARMER_EMAIL	10:26:53	WHERE Farmer_ID = 6 1 row(s) affected	0.016 sec

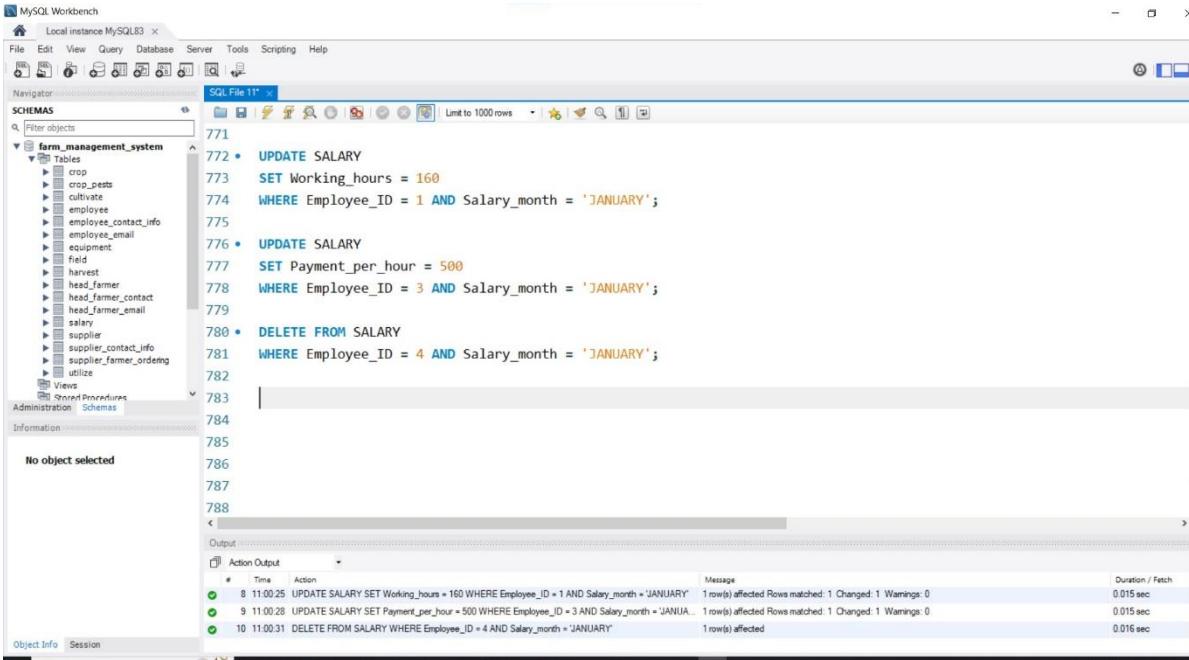
The screenshot shows the MySQL Command Line Client window. The command `select * from head_farmer_email;` was run, resulting in the following output:

```
6 rows in set (0.00 sec)

mysql> select * from head_farmer_email;
+-----+-----+
| Farmer_ID | Email      |
+-----+-----+
| 1 | Dinesh70@gmail.com |
| 3 | Kusal13@gmail.com |
| 4 | Pathum26@gmail.com |
| 5 | Pathum30@gmail.com |
| 7 | Sahan27@gmail.com |
+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

13.Salary

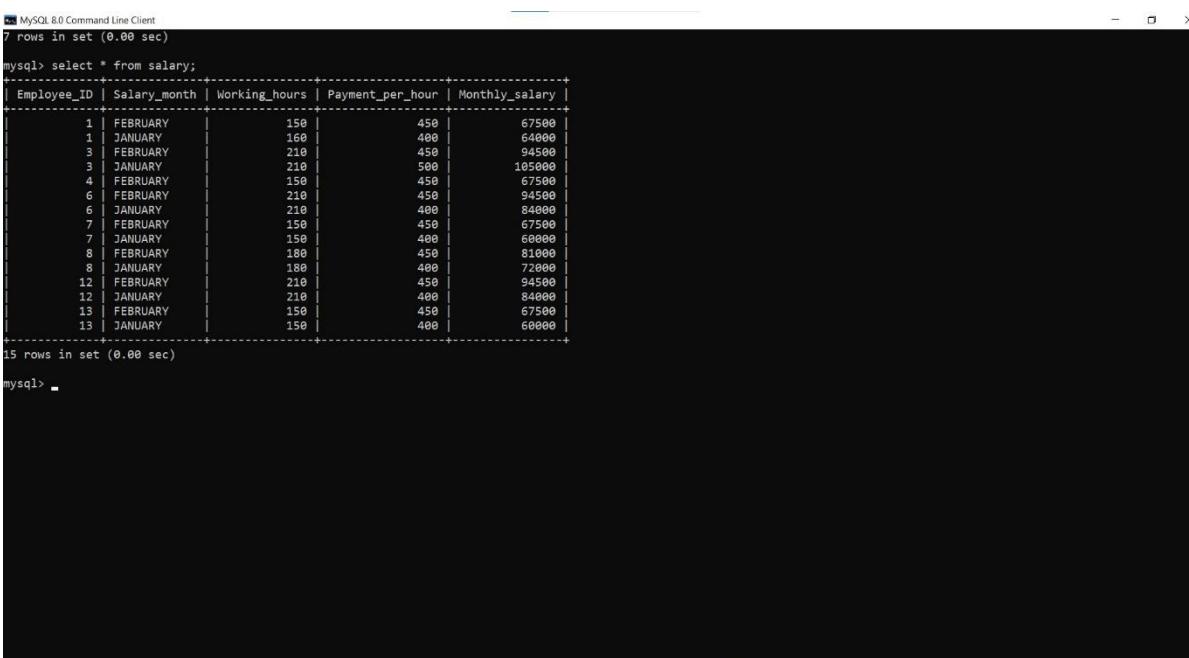


The screenshot shows the MySQL Workbench interface with a SQL editor tab titled "SQL File 11". The code entered is:

```
771 • UPDATE SALARY
772 SET Working_hours = 160
773 WHERE Employee_ID = 1 AND Salary_month = 'JANUARY';
774
775 • UPDATE SALARY
776 SET Payment_per_hour = 500
777 WHERE Employee_ID = 3 AND Salary_month = 'JANUARY';
778
779 • DELETE FROM SALARY
780 WHERE Employee_ID = 4 AND Salary_month = 'JANUARY';
781
782
783
784
785
786
787
788
```

The "Output" pane shows the results of the executed statements:

#	Time	Action	Message	Duration / Fetch
8	11:00:25	UPDATE SALARY SET Working_hours = 160 WHERE Employee_ID = 1 AND Salary_month = 'JANUARY'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
9	11:00:28	UPDATE SALARY SET Payment_per_hour = 500 WHERE Employee_ID = 3 AND Salary_month = 'JANUARY'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
10	11:00:31	DELETE FROM SALARY WHERE Employee_ID = 4 AND Salary_month = 'JANUARY'	1 row(s) affected	0.016 sec



The screenshot shows the MySQL Command Line Client window. The command entered is:

```
mysql> select * from salary;
```

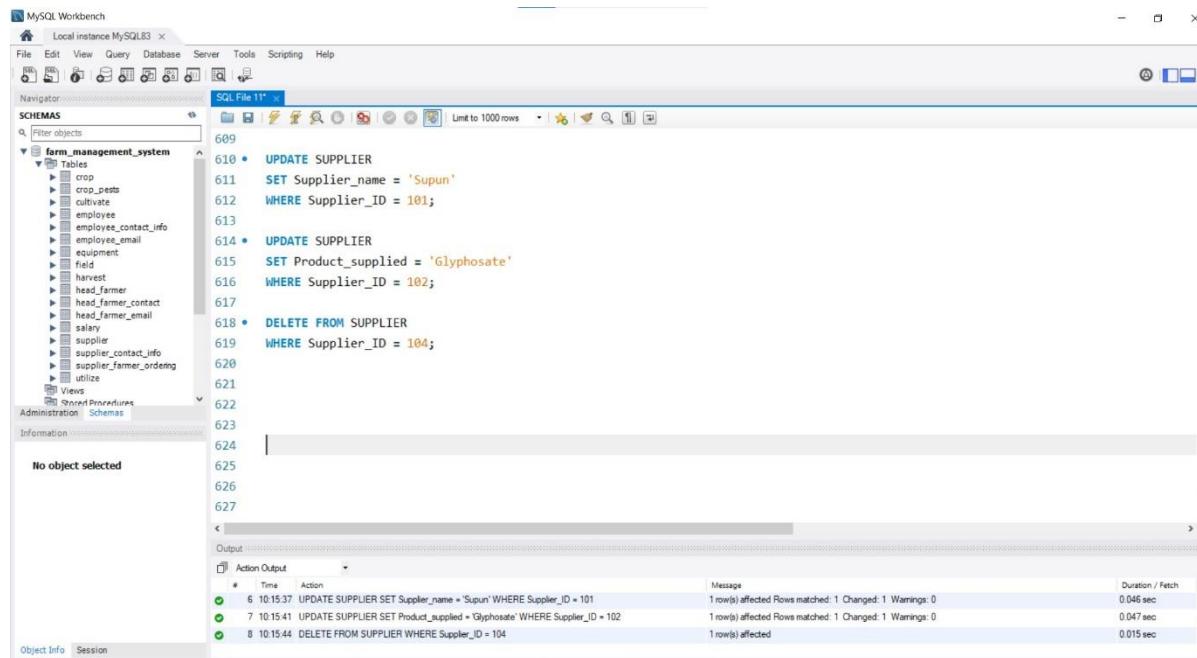
The output displays the salary data:

Employee_ID	Salary_month	Working_hours	Payment_per_hour	Monthly_salary
1	FEBRUARY	150	450	67500
1	JANUARY	160	400	64000
3	FEBRUARY	210	450	94500
3	JANUARY	210	500	105000
4	FEBRUARY	150	450	67500
6	FEBRUARY	210	450	94500
6	JANUARY	210	400	84000
7	FEBRUARY	150	450	67500
7	JANUARY	150	400	60000
8	FEBRUARY	180	450	81000
8	JANUARY	180	400	72000
12	FEBRUARY	210	450	94500
12	JANUARY	210	400	84000
13	FEBRUARY	150	450	67500
13	JANUARY	150	400	60000

15 rows in set (0.00 sec)

```
mysql>
```

14. Supplier

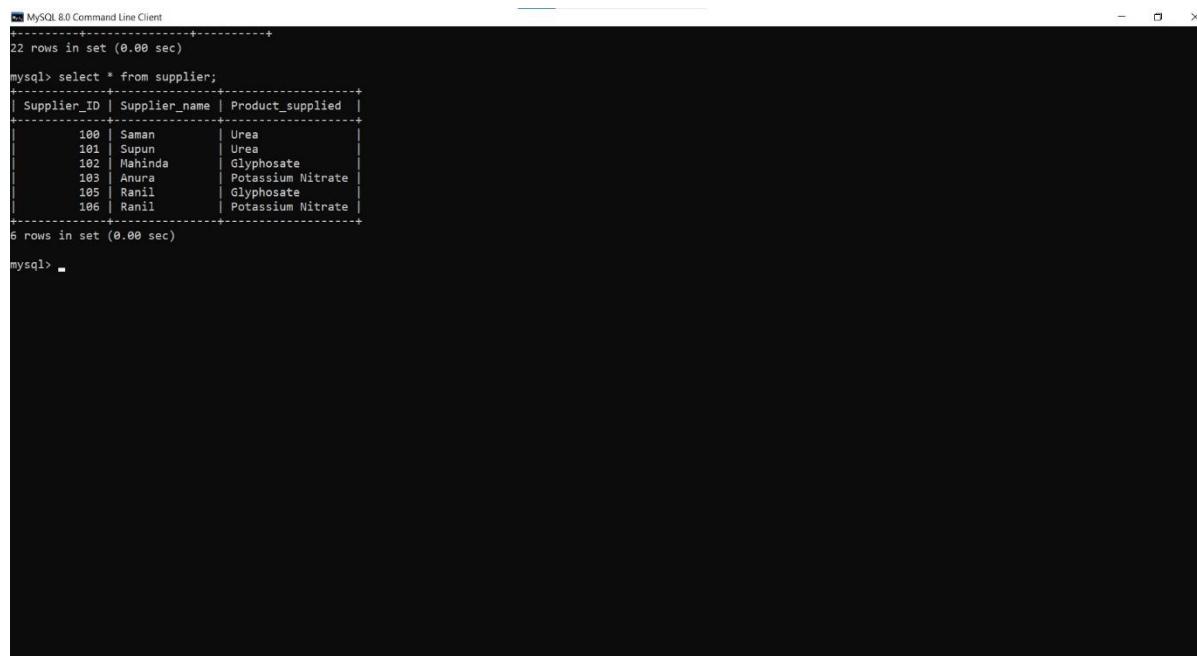


The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard database management tools.
- Navigator:** Shows the schema `farm_management_system` with tables like `crop`, `crop_pests`, `cultivate`, `employee`, `employee_contact_info`, `employee_email`, `equipment`, `field`, `harvest`, `head_farmer`, `head_farmer_contact`, `head_farmer_email`, `salary`, `supplier`, `supplier_contact_info`, `supplier_farmer_ordering`, and `utilize`.
- SQL Editor:** Contains the following SQL code:

```
609 • UPDATE SUPPLIER
610   SET Supplier_name = 'Supun'
611   WHERE Supplier_ID = 101;
613
614 • UPDATE SUPPLIER
615   SET Product_supplied = 'Glyphosate'
616   WHERE Supplier_ID = 102;
617
618 • DELETE FROM SUPPLIER
619   WHERE Supplier_ID = 104;
620
621
622
623
624
625
626
627
```
- Output Window:** Shows the results of the executed queries:

#	Time	Action	Message	Duration / Fetch
6	10:15:37	UPDATE SUPPLIER SET Supplier_name = 'Supun' WHERE Supplier_ID = 101	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.04 sec
7	10:15:41	UPDATE SUPPLIER SET Product_supplied = 'Glyphosate' WHERE Supplier_ID = 102	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.047 sec
8	10:15:44	DELETE FROM SUPPLIER WHERE Supplier_ID = 104	1 row(s) affected	0.015 sec
- Bottom Navigation:** Object Info, Session.



The screenshot shows the MySQL Command Line Client with the following details:

- Output:** Displays the results of the `select * from supplier;` query:

```
+-----+-----+-----+
| Supplier_ID | Supplier_name | Product_supplied |
+-----+-----+-----+
|      100 | Saman        | Urea             |
|      101 | Supun        | Urea             |
|      102 | Mahinda      | Glyphosate       |
|      103 | Anura        | Potassium Nitrate|
|      105 | Ranil         | Glyphosate       |
|      106 | Ranil         | Potassium Nitrate|
+-----+-----+-----+
```
- Message:** Shows "6 rows in set (0.00 sec)".
- Prompt:** mysql> ■

15. Supplier_contact_info

The screenshot shows the MySQL Workbench interface. The left pane displays the Navigator with the schema 'farm_management_system' selected, showing various tables like crop, crop_pests, cultivate, employee, etc. The central pane contains a SQL editor window titled 'SQL File 11*' with the following code:

```
620
621
622
623
624 • UPDATE SUPPLIER_CONTACT_INFO
625 SET Contact_info = '0714431825'
626 WHERE Supplier_ID = 101;
627
628 • UPDATE SUPPLIER_CONTACT_INFO
629 SET Contact_info = '0717051699'
630 WHERE Supplier_ID = 103;
631
632 • DELETE FROM SUPPLIER_CONTACT_INFO
633 WHERE Supplier_ID = 105;
634
635
636
637
638
```

The right pane shows the 'Output' tab with the results of the executed statements:

Action	Time	Message	Duration / Fetch
9 10:19:50 UPDATE SUPPLIER_CONTACT_INFO SET Contact_info = '0714431825' WHERE Supplier_ID = 101		1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.032 sec
10 10:19:53 UPDATE SUPPLIER_CONTACT_INFO SET Contact_info = '0717051699' WHERE Supplier_ID = 103		1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
11 10:19:56 DELETE FROM SUPPLIER_CONTACT_INFO WHERE Supplier_ID = 105		1 row(s) affected	0.016 sec

The screenshot shows the MySQL Command Line Client window. The command 'select * from supplier_contact_info;' was run, resulting in the following output:

```
mysql> select * from supplier_contact_info;
+-----+-----+
| Supplier_ID | Contact_info |
+-----+-----+
| 100 | 0714671827 |
| 101 | 0714431825 |
| 102 | 0711500145 |
| 103 | 0717051699 |
| 106 | 0714679584 |
+-----+-----+
5 rows in set (0.00 sec)
```

16. Supplier_farmer_ordering

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Shows the schema `farm_management_system` with tables: crop, crop_pests, cultivate, employee, employee_contact_info, employee_email, equipment, field, harvest, head_farmer, head_farmer_contact, head_farmer_email, salary, supplier, supplier_contact_info, supplier_farmer_ordering, utilize.
- SQL Editor:** Contains the following SQL code:

```
678 • UPDATE SUPPLIER_FARMER_ORDERING
679   SET Farmer_ID = 5
680 WHERE Supplier_ID = 105;
681
682
683 • UPDATE SUPPLIER_FARMER_ORDERING
684   SET Farmer_ID = 4
685 WHERE Supplier_ID = 102;
686
687
688 • DELETE FROM SUPPLIER_FARMER_ORDERING
689 WHERE Supplier_ID = 106;
690
691
692
693
694
```
- Output Window:** Shows the execution results for each query:

Action	Time	Message	Duration / Fetch
21	10:38:06	UPDATE SUPPLIER_FARMER_ORDERING SET Farmer_ID = 5 WHERE Supplier_ID = 105 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
22	10:38:12	UPDATE SUPPLIER_FARMER_ORDERING SET Farmer_ID = 4 WHERE Supplier_ID = 102 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
23	10:38:24	DELETE FROM SUPPLIER_FARMER_ORDERING WHERE Supplier_ID = 106 2 row(s) affected	0.016 sec

The screenshot shows the MySQL Command Line Client with the following output:

```
5 rows in set (0.00 sec)

mysql> select * from supplier_farmer_ordering;
+-----+-----+
| Supplier_ID | Farmer_ID |
+-----+-----+
|      100    |      1     |
|      100    |      3     |
|      102    |      4     |
|      105    |      5     |
+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

17.Utilize

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL83' is selected. The 'Navigator' pane on the left shows the 'farm_management_system' schema with various tables like crop, crop_pests, cultivate, employee, etc. The central 'SQL File 11*' tab contains the following SQL code:

```
718 • UPDATE UTILIZE
719   SET Field_ID = 505
720 WHERE Equipment_ID = 210;
721
722 • UPDATE UTILIZE
723   SET Equipment_ID = 200
724 WHERE Field_ID = 508;
725
726 • DELETE FROM UTILIZE
727 WHERE Equipment_ID = 209;
728 |
729
730
731
732
733
734
```

The 'Output' pane at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
30	10:46:17	UPDATE UTILIZE SET Field_ID = 505 WHERE Equipment_ID = 210	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
31	10:46:20	UPDATE UTILIZE SET Equipment_ID = 200 WHERE Field_ID = 508	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
32	10:46:33	DELETE FROM UTILIZE WHERE Equipment_ID = 209	0 row(s) affected	0.000 sec

The screenshot shows the MySQL 8.0 Command Line Client window. The command 'select * from utilize;' is run, resulting in the following output:

```
6 rows in set (0.00 sec)

mysql> select * from utilize;
+-----+-----+
| Equipment_ID | Field_ID |
+-----+-----+
|      200 |     500 |
|      202 |     502 |
|      203 |     503 |
|      211 |     505 |
|      206 |     506 |
|      200 |     507 |
|      207 |     507 |
+-----+-----+
7 rows in set (0.00 sec)

mysql>
```

TRANSACTIONS

Simple Quarries

1. Like

The screenshot shows the MySQL Workbench interface. The left pane displays the database schema for 'farm_management_system' with tables like crop, crop_pests, cultivate, employee, etc. The right pane shows the results of a query:

```
-- Retrieve Equipment_ID,Model,Equipment_type records from the EQUIPMENT table where Equipment_type contains 'er'
SELECT Equipment_ID,Model,Equipment_type FROM equipment WHERE Equipment_type LIKE '%er%';
```

The results grid shows the following data:

Equipment_ID	Model	Equipment_type
202	John Deere	Harvester
203	John Deere	Seeder
204	Agco	Sprayer
206	Agco	Harvester
207	Kubota	Seeder
208	Kubota	Seeder
210	John Deere	Seeder
211	Agco	Sprayer

The status bar at the bottom indicates the query was executed at 12:26:11 and returned 8 rows.

2. Aggregation function

The screenshot shows the MySQL Workbench interface. The left pane displays the database schema for 'farm_management_system' with tables like crop, crop_pests, cultivate, employee, etc. The right pane shows the results of a query:

```
-- Calculate the average Quantity from the HARVEST table
SELECT AVG(Quantity) AS AvgQuantity FROM HARVEST;
```

The results grid shows the following data:

AvgQuantity
328.5714

The status bar at the bottom indicates the query was executed at 12:17:48 and returned 1 row.

3. User view

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

- Navigator:** Shows the schema `farm_management_system` with tables like `crop`, `crop_pests`, `cultivate`, `employee`, etc.
- Database assignment:** Displays the SQL code for creating a view:

```
609 -- Create a view named userview that combines Farmer_ID and Farmer_name from HEAD_FARMER table
610 • CREATE VIEW userview AS
611   SELECT Farmer_ID,Farmer_name
612   FROM HEAD_FARMER;
613 • SELECT * FROM userview;
```
- Result Grid:** Shows the results of the query:

Farmer_ID	Farmer_name
1	Dresh
3	Kusal
4	Pathum
5	Pathum
6	Charith
7	Sahan
- Object Info:** Shows the table `supplier_contact_info` with columns `Supplier_ID` and `Contact_info`.
- Action Output:** Shows the execution log:

Time	Action	Message	Duration / Fetch
1 12:17:48	CREATE VIEW userview AS SELECT Farmer_ID,Farmer_name FROM HEAD_FARMER	0 row(s) affected	0.016 sec
2 12:17:48	SELECT * FROM userview LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

4. Rename

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

- Navigator:** Shows the schema `farm_management_system` with tables like `crop`, `crop_pests`, `cultivate`, `employee`, etc.
- Database assignment:** Displays the SQL code for creating a view:

```
615
616 -- Retrieve Supplier_ID as ID and Supplier_name as Name from SUPPLIER table
617 • SELECT Supplier_ID AS ID, Supplier_name AS Name FROM SUPPLIER;
```
- Result Grid:** Shows the results of the query:

ID	Name
100	Saman
101	Supun
102	Mahinda
103	Anura
105	Rani
106	Rani
- Object Info:** Shows the table `supplier_contact_info` with columns `Supplier_ID` and `Contact_info`.
- Action Output:** Shows the execution log:

Time	Action	Message	Duration / Fetch
29 12:09:18	CREATE VIEW userview AS SELECT Farmer_ID,Farmer_name FROM HEAD_FARMER	0 row(s) affected	0.047 sec
30 12:09:28	SELECT * FROM userview LIMIT 0, 1000	6 row(s) returned	0.047 sec / 0.000 sec
31 12:13:42	SELECT Supplier_ID AS ID, Supplier_name AS Name FROM SUPPLIER LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

5. Cartesian product

The screenshot shows the MySQL Workbench interface with a query editor window. The code entered is:

```
-- Retrieve all columns from supplier_farmer_ordering and supplier_contact_info tables, combining every row from both tables
SELECT * FROM supplier_farmer_ordering, supplier_contact_info;
```

The result grid displays the combined data from both tables. The columns are Supplier_ID, Farmer_ID, Supplier_ID, and Contact_info. The data shows multiple rows where each row from the first table is paired with each row from the second table.

Supplier_ID	Farmer_ID	Supplier_ID	Contact_info
105	5	100	0714671827
102	4	100	0714671827
100	3	100	0714671827
100	1	100	0714671827
105	5	101	0714431825
102	4	101	0714431825
100	3	101	0714431825
100	1	101	0714431825
105	5	102	0711500145
102	4	102	0711500145
100	3	102	0711500145
100	1	102	0711500145
105	5	103	0717051699
102	4	103	0717051699
100	3	103	0717051699
100	1	103	0717051699
105	5	106	0714679584
102	4	106	0714679584
100	3	106	0714679584
100	1	106	0714679584

6. Select

The screenshot shows the MySQL Workbench interface with a query editor window. The code entered is:

```
-- Retrieve all records from the SUPPLIER table where Supplier_ID = 100;
SELECT * FROM SUPPLIER where Supplier_ID = 100;
```

The result grid displays the data for Supplier_ID 100. The columns are Supplier_ID, Supplier_name, and Product_supplied. The data shows one row for Supplier_ID 100, named Saman, supplying Urea.

Supplier_ID	Supplier_name	Product_supplied
100	Saman	Urea

7. Project

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar has sections for Navigator (SCHEMAS, Views, Stored Procedures, Functions), Administration, and Schemas. The main area displays a query editor titled "Database assignment" with the following content:

```
-- Retrieve only Supplier_ID and Supplier_name columns from the SUPPLIER table
SELECT Supplier_ID, Supplier_name FROM SUPPLIER;
```

The result grid shows the following data:

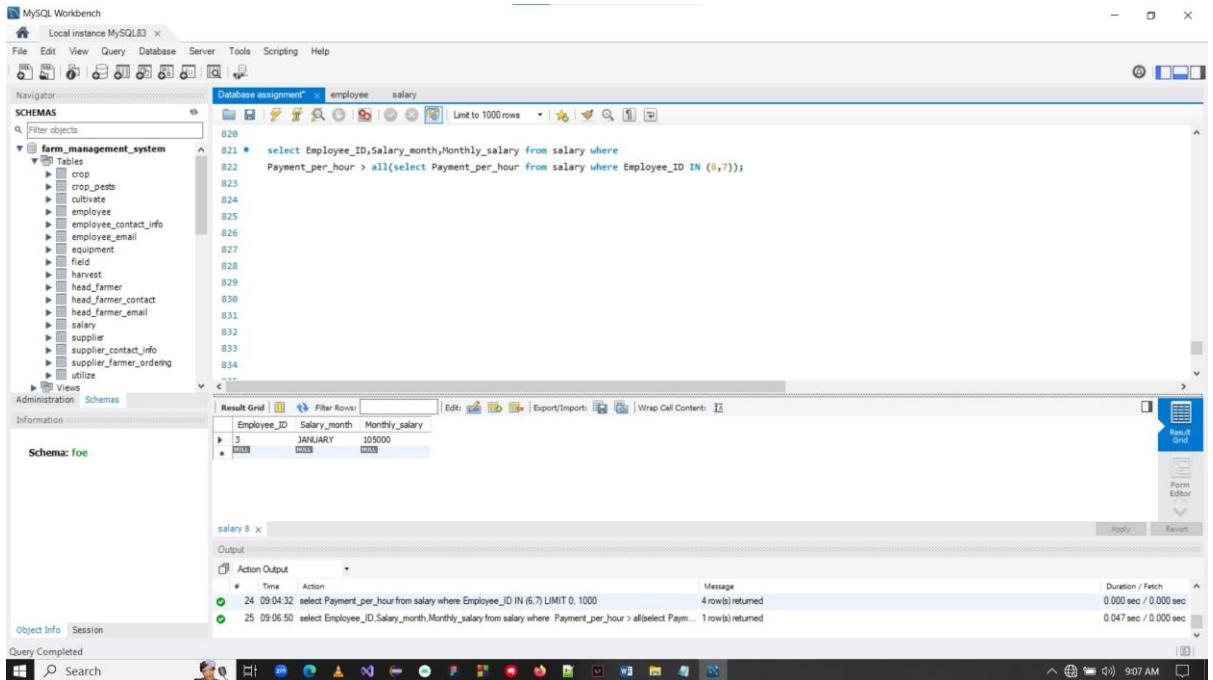
Supplier_ID	Supplier_name
100	Saman
101	Supun
102	Mahinda
103	Anura
105	Ranil
106	Ranil
107	Ranil

The bottom section, "Action Output", lists the following log entries:

Time	Action	Message	Duration / Fetch
22 11:17:34	DELETE FROM HARVEST WHERE Crop_ID = 12	2 row(s) affected	0.016 sec
23 11:51:29	SELECT * FROM SUPPLIER LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
24 11:55:38	SELECT Supplier_ID, Supplier_name FROM SUPPLIER LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

Complex Quarries

1. Nested 1



The screenshot shows the MySQL Workbench interface with a query editor window. The query being run is:

```
821 * select Employee_ID,Salary_month,Monthly_salary from salary where
822 Payment_per_hour > all(select Payment_per_hour from salary where Employee_ID IN (6,7));
```

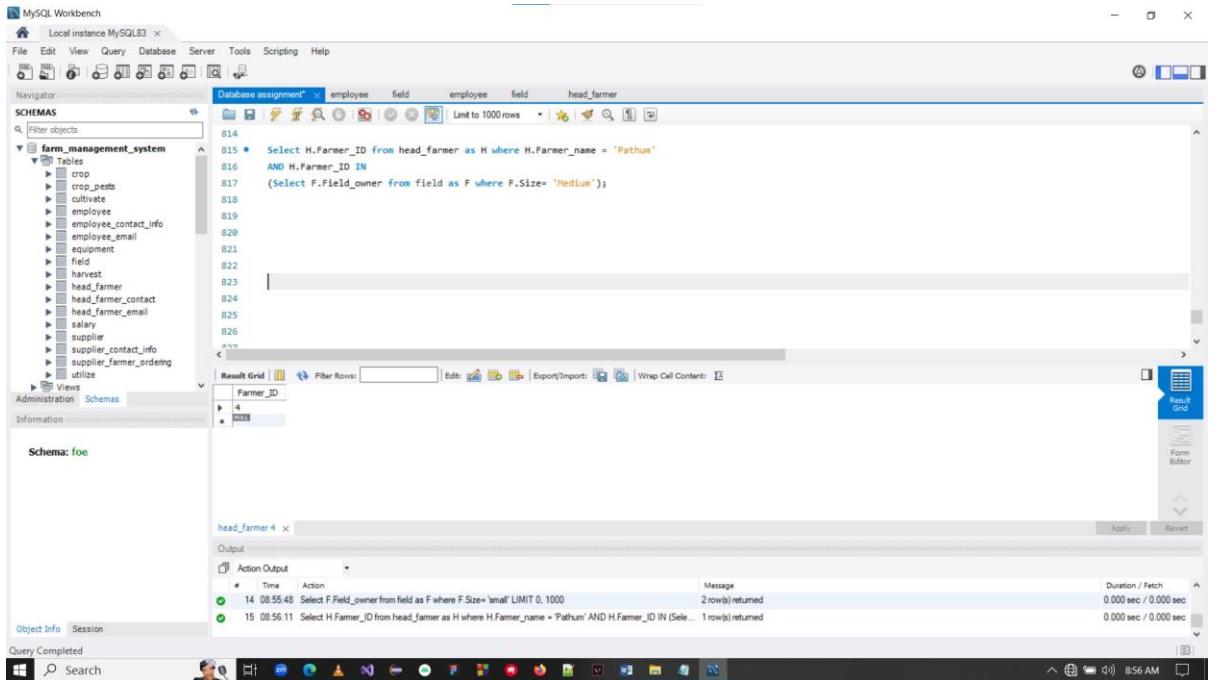
The result grid shows one row of data:

Employee_ID	Salary_month	Monthly_salary
3	JANUARY	10500

The output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
24	09:04:32	select Payment_per_hour from salary where Employee_ID IN (6,7) LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
25	09:06:50	select Employee_ID,Salary_month,Monthly_salary from salary where Payment_per_hour > all(select Paym... 1 row(s) returned		0.047 sec / 0.000 sec

2. Nested 2



The screenshot shows the MySQL Workbench interface with a query editor window. The query being run is:

```
814 * Select H.Farmer_ID from head_farmer as H where H.Farmer_name = 'Pathum'
815 AND H.Farmer_ID IN
816 (Select F.Field_owner from field as F where F.Size= 'Medium');
```

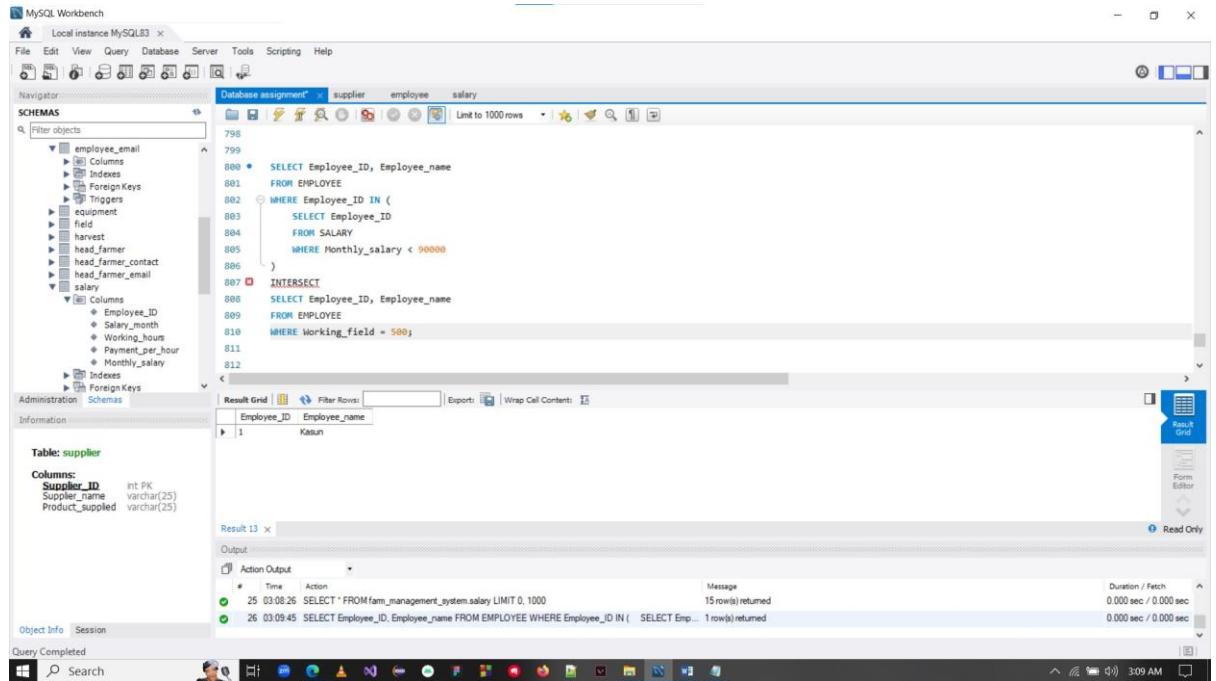
The result grid shows one row of data:

Farmer_ID
4

The output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
14	08:55:48	Select H.Farmer_ID from head_farmer as H where H.Farmer_name = 'Pathum' AND H.Farmer_ID IN (Sel... 2 row(s) returned		0.000 sec / 0.000 sec
15	08:56:11	Select F.Field_owner from field as F where F.Size= 'Medium' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

3. Nested 3



The screenshot shows the MySQL Workbench interface with a query editor window open. The query being run is:

```
798
799
800 •  SELECT Employee_ID, Employee_name
801   FROM EMPLOYEE
802  WHERE Employee_ID IN (
803    SELECT Employee_ID
804      FROM SALARY
805     WHERE Monthly_salary < 90000
806  )
807 □ INTERSECT
808   SELECT Employee_ID, Employee_name
809   FROM EMPLOYEE
810  WHERE Working_field = 500;
811
812
```

The results grid shows one row:

Employee_ID	Employee_name
1	Kasun

The "Result 13" pane shows the history of actions taken:

Action	Time	Message	Duration / Fetch
SELECT * FROM fam_management_system.salary LIMIT 0, 1000	25 03:08:26	15 row(s) returned	0.000 sec / 0.000 sec
SELECT Employee_ID, Employee_name FROM EMPLOYEE WHERE Employee_ID IN (26 03:09:45	1 row(s) returned	0.000 sec / 0.000 sec

4. Full outer join

MySQL Workbench

Local instance MySQL83

File Edit View Query Database Server Tools Scripting Help

Navigator: Database assignment

SCHEMAS: emp, emp, equi, field, harv, head, head, head, salary

Result Grid: Employee_ID, Employee_name, Supervisor, Working_field, Employee_ID, Salary_month, Working_hours, Payment_per_hour, Monthly_salary

```

768
769 • CREATE VIEW userview9 AS
770   SELECT * FROM employee;
771 • CREATE VIEW userview10 AS
772   SELECT * FROM salary;
773
774 • (select * from
    userview9 left outer join userview10
    ON userview9.Employee_ID = userview10.Employee_ID)
  union
  (select * from
    userview9 right outer join userview10
    ON userview9.Employee_ID = userview10.Employee_ID);
775
776
777
778
779
780
781
782
783

```

Result Grid: Employee_ID, Employee_name, Supervisor, Working_field, Employee_ID, Salary_month, Working_hours, Payment_per_hour, Monthly_salary

Employee_ID	Employee_name	Supervisor	Working_field	Employee_ID	Salary_month	Working_hours	Payment_per_hour	Monthly_salary	
1	Kasun	500	1	FEBRUARY	150	450	67500		
1	Kasun	500	1	JANUARY	160	400	64000		
3	Npun	502	3	FEBRUARY	210	450	94500		
3	Npun	502	3	JANUARY	210	500	105000		
4	Ninada	3	503	4	FEBRUARY	150	450	67500	
6	Sarjana	505	6	FEBRUARY	210	450	94500		
6	Sarjana	505	6	JANUARY	210	400	84000		
7	Thurunu	506	7	FEBRUARY	150	450	67500		
7	Thurunu	506	7	JANUARY	150	400	60000		
8	Geeth	507	8	FEBRUARY	180	450	81000		
8	Geeth	507	8	JANUARY	180	400	72000		
12	Ninada	506	12	FEBRUARY	210	450	94500		
12	Ninada	506	12	JANUARY	210	400	84000		
13	Thisaru	8	507	13	FEBRUARY	150	450	67500	
13	Thisaru	8	507	13	JANUARY	150	400	60000	

Result 1:

Action Output: 6 01:54:51 CREATE VIEW userview10 AS SELECT * FROM salary 0 rows/affected Duration / Fetch: 0.016 sec

Query Completed

MySQL Workbench

Local instance MySQL83

File Edit View Query Database Server Tools Scripting Help

Navigator: Database assignment

SCHEMAS: emp, emp, equi, field, harv, head, head, head, salary

Result Grid: Employee_ID, Employee_name, Supervisor, Working_field, Employee_ID, Salary_month, Working_hours, Payment_per_hour, Monthly_salary

```

768
769 • CREATE VIEW userview9 AS
770   SELECT * FROM employee;
771 • CREATE VIEW userview10 AS
772   SELECT * FROM salary;
773
774 • (select * from
    userview9 left outer join userview10
    ON userview9.Employee_ID = userview10.Employee_ID)
  union
  (select * from
    userview9 right outer join userview10
    ON userview9.Employee_ID = userview10.Employee_ID);
775
776
777
778
779
780
781
782
783

```

Result Grid: Employee_ID, Employee_name, Supervisor, Working_field, Employee_ID, Salary_month, Working_hours, Payment_per_hour, Monthly_salary

Employee_ID	Employee_name	Supervisor	Working_field	Employee_ID	Salary_month	Working_hours	Payment_per_hour	Monthly_salary	
1	Kasun	500	1	FEBRUARY	150	450	67500		
1	Kasun	500	1	JANUARY	160	400	64000		
3	Npun	502	3	FEBRUARY	210	450	94500		
3	Npun	502	3	JANUARY	210	500	105000		
4	Ninada	3	503	4	FEBRUARY	150	450	67500	
6	Sarjana	505	6	FEBRUARY	210	450	94500		
6	Sarjana	505	6	JANUARY	210	400	84000		
7	Thurunu	506	7	FEBRUARY	150	450	67500		
7	Thurunu	506	7	JANUARY	150	400	60000		

Result 1:

Action Output: 6 01:54:51 CREATE VIEW userview10 AS SELECT * FROM salary 0 rows/affected Duration / Fetch: 0.016 sec

Query Completed

5. Right outer join

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

- Database:** assignment
- Tables:** equipment, supplier_farmer_ordering, crop, cultivate
- Code Editor:**

```

756 • CREATE VIEW userview7 AS
757     SELECT * FROM cultivate;
758
759 • CREATE VIEW userview8 AS
760     SELECT * FROM crop;
761
762
763 • select * from
764     userview7 right outer join userview8
765     ON userview7.Crop_ID = userview8.Crop_ID;
766
767
    
```
- Result Grid:** Displays the results of the right outer join query. The columns are Field_ID, Crop_ID, Crop_ID, Crop_name, Variety, Planting_date, Harvest_date, and Crop_Age. The data includes rows for various crops like Corn, Wheat, Durum, Rice, Beans, Carrot, Tomato, Apple, Peanuts, Orange, Potato, and Wheat.
- Action Output:**
 - Time: 21:33:08
 - Action: select * from userview7 right outer join userview8 ON userview7.Crop_ID = userview8.Crop_ID LIMIT 0, 1000
 - Message: 10 row(s) returned
 - Duration / Fetch: 0.000 sec / 0.000 sec

6. Left outer join

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

- Database:** assignment
- Tables:** equipment, supplier_farmer_ordering, head_farmer
- Code Editor:**

```

740
741 • CREATE VIEW userview5 AS
742     SELECT * FROM supplier_farmer_ordering;
743
744 • CREATE VIEW userview6 AS
745     SELECT * FROM head_farmer;
746
747
748 • select * from
749     userview5 left outer join userview6
750     ON userview5.Farmer_ID = userview6.Farmer_ID;
    
```
- Result Grid:** Displays the results of the left outer join query. The columns are Supplier_ID, Farmer_ID, Farmer_ID, Farmer_name, Street, City, Province, and Zip_code. The data includes rows for farmers Dinesh, Kusal, Pathum, and Pathum.
- Action Output:**
 - Time: 21:08:56
 - Action: CREATE VIEW userview5 AS SELECT * FROM head_farmer
 - Message: 0 row(s) affected
 - Time: 21:08:59
 - Action: select * from userview5 left outer join userview6 ON userview5.Farmer_ID = userview6.Farmer_ID LIMIT 0, 1000
 - Message: 4 row(s) returned
 - Duration / Fetch: 0.031 sec / 0.000 sec / 0.000 sec

7. Division

The screenshot shows the MySQL Workbench interface with a query editor window titled "Database assignment" containing the following SQL code:

```

723
724
725 •   SELECT Equipment_owner
726   FROM equipment AS e1
727   GROUP BY Equipment_owner
728   HAVING NOT EXISTS (
729     SELECT * FROM (
730       SELECT Equipment_ID FROM equipment
731       WHERE Equipment_ID IN (200, 207, 208)
732     ) As e2
733     WHERE NOT EXISTS (
734       SELECT * FROM equipment AS e3
735       WHERE e3.Equipment_owner = e1.Equipment_owner
736       AND e3.Equipment_ID = e2.Equipment_ID
737     )
738   );
739
740
    
```

The "Result Grid" pane shows the output of the query:

Equipment_owner
1

The "Action Output" pane displays the execution log:

- 10 20:35:35 SELECT Equipment_owner FROM equipment e1 GROUP BY Equipment_owner HAVING NOT EXISTS (...) 1 row(s) returned 0.000 sec / 0.000 sec
- 11 20:38:42 SELECT Equipment_owner FROM equipment AS e1 GROUP BY Equipment_owner HAVING NOT EXIST... 1 row(s) returned 0.000 sec / 0.000 sec

8. Natural join

The screenshot shows the MySQL Workbench interface with a query editor window titled "Database assignment" containing the following SQL code:

```

702 • CREATE VIEW userview3 AS SELECT Field_ID,Field_Name,Size FROM FIELD;
703 • CREATE VIEW userview4 AS SELECT * FROM CULTIVATE;
704
705
706 • SELECT * FROM
707 userview3
708 NATURAL JOIN userview4;
709
710
    
```

The "Result Grid" pane shows the output of the query:

Field_ID	Field_Name	Size	Crop_ID
500	A	small	10
502	B	Small	12
503	B	Medium	13
505	C	large	15
506	C	medium	16
507	D	large	17

The "Action Output" pane displays the execution log:

- 24 08:46:24 SELECT * FROM userview3 NATURAL JOIN userview4 LIMIT 0, 1000 6 row(s) returned 0.000 sec / 0.000 sec
- 25 08:47:13 SELECT * FROM farm_management_system.field LIMIT 0, 1000 6 row(s) returned 0.000 sec / 0.000 sec

9. Inner join

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

- File menu:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (selected), Filter objects, Fields, Columns, Indexes, Foreign Keys, Triggers, Field, etc.
- Database assignment:** A code editor window containing a query that creates two views and performs an inner join between them.
- Result Grid:** A table showing the results of the query. The columns are Head_farmer, Field_ID, and Equipment_model. The data includes rows for Kubota, John Deere, and Agco.
- Action Output:** A log of SQL statements and their execution times.
- Object Info:** Information about the selected column 'Equipment_owner'.
- Session:** Session information.

```

CREATE VIEW userview1 AS SELECT Field_ID,Field_owner FROM FIELD;
CREATE VIEW userview2 AS SELECT Model as Equipment_model,Equipment_owner as Head_farmer FROM EQUIPMENT;
SELECT userview2.Head_farmer, userview1.Field_ID,userview2.Equipment_model
FROM
userview1
INNER JOIN userview2
ON userview2.Head_farmer = userview1.Field_owner;
    
```

10. Set difference

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

- File menu:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (selected), Filter objects, Fields, Columns, Indexes, Foreign Keys, Triggers, Field, etc.
- Database assignment:** A code editor window containing a query that uses a self-join and an except clause to find suppliers who supply 'Glyphosate' but not 'Ranil'.
- Result Grid:** A table showing the results of the query. The columns are Supplier_ID, Supplier_name, and Product_supplied. The data shows one row for Mahinda.
- Action Output:** A log of SQL statements and their execution times.
- Object Info:** Information about the selected schema 'farm_management_system'.
- Session:** Session information.

```

SELECT * FROM SUPPLIER where Product_supplied='Glyphosate'
EXCEPT
SELECT * FROM SUPPLIER where Supplier_name='Ranil';
    
```

11. Intersect

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

- Navigator:** Shows the schema `farm_management_system` with tables like `crop`, `crop_pests`, `cultivate`, etc.
- Query Editor:** Displays the following SQL code:

```
657
658
659
660
661
662
663 •  SELECT * FROM SUPPLIER where Supplier_name='Ranil'
664 □  Intersect
665 SELECT * FROM SUPPLIER where Product_supplied='Glyphosate';
```
- Result Grid:** Shows the result of the query:

Supplier_ID	Supplier_name	Product_supplied
105	Ranil	Glyphosate
- Action Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
4	20:58:05	SELECT * FROM SUPPLIER where Supplier_name='Ranil' Union SELECT * FROM SUPPLIER where Pro... 3 row(s) returned		0.000 sec / 0.000 sec
5	20:58:33	SELECT * FROM SUPPLIER where Supplier_name='Ranil' Union SELECT * FROM SUPPLIER where Pro... 3 row(s) returned		0.000 sec / 0.000 sec
6	21:00:27	SELECT * FROM SUPPLIER where Supplier_name='Ranil' Intersect SELECT * FROM SUPPLIER where ... 1 row(s) returned		0.000 sec / 0.000 sec

12. Union

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

- Navigator:** Shows the schema `farm_management_system` with tables like `crop`, `crop_pests`, `cultivate`, etc.
- Query Editor:** Displays the following SQL code:

```
647
648
649
650
651
652 •  SELECT * FROM SUPPLIER where Supplier_name='Ranil'
653 Union
654 SELECT * FROM SUPPLIER where Product_supplied='Glyphosate';
```
- Result Grid:** Shows the result of the query:

Supplier_ID	Supplier_name	Product_supplied
105	Ranil	Glyphosate
106	Ranil	Potassium Nitrate
102	Mahinda	Glyphosate
- Action Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
3	20:57:12	SELECT * FROM SUPPLIER where Product_supplied='Glyphosate' LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
4	20:58:05	SELECT * FROM SUPPLIER where Supplier_name='Ranil' Union SELECT * FROM SUPPLIER where Pro... 3 row(s) returned		0.000 sec / 0.000 sec
5	20:58:33	SELECT * FROM SUPPLIER where Supplier_name='Ranil' Union SELECT * FROM SUPPLIER where Pro... 3 row(s) returned		0.000 sec / 0.000 sec

13. Outer Union

MySQL Workbench - Local instance MySQL83

```

File Edit View Query Database Server Tools Scripting Help
Navigator Database assignment SQL File 3*
SCHEMAS Filter objects
farm_management_system Tables
crop crop_pests cultivate employee employee_contact_info employee_email equipment field harvest head_farmer head_farmer_contact head_farmer_email salary supplier supplier_contact_info supplier_farmer_ordering utilize
Views Administration Schemas Information
No object selected
Result Grid Filter Rows Export Wrap Cell Content
Employee_ID Email Contact_info
1 Kasun1@gmail.com 0771234567
3 NipurDilshan3 0773456780
4 Nhada4@gmail.com 0774567890
6 Sanjanas@gmail.com 0776789012
8 Geeth8@gmail.com
Result 6 x
Output Action Output
# Time Action Message Duration / Fetch
34 21:24:42 Create view userview11 AS SELECT COALESCE(ee.Employee_ID, ec.Employee_ID) AS Employee_ID, ee.Email, ec.Contact_info 0 row(s) affected 0.016 sec
35 21:25:31 select * from userview11 UNION select * from userview12 8 row(s) returned 0.000 sec / 0.000 sec
36 21:26:12 select * from userview11 UNION select * from userview12 8 row(s) returned 0.000 sec / 0.000 sec
Object Info Session

```

MySQL Workbench - Local instance MySQL83

```

File Edit View Query Database Server Tools Scripting Help
Navigator Database assignment SQL File 3*
SCHEMAS Filter objects
farm_management_system Tables
crop crop_pests cultivate employee employee_contact_info employee_email equipment field harvest head_farmer head_farmer_contact head_farmer_email salary supplier supplier_contact_info supplier_farmer_ordering utilize
Views Administration Schemas Information
No object selected
Result Grid Filter Rows Export Wrap Cell Content
Employee_ID Email Contact_info
1 Kasun1@gmail.com 0771234567
3 NipurDilshan3 0773456780
4 Nhada4@gmail.com 0774567890
6 Sanjanas@gmail.com 0776789012
8 Geeth8@gmail.com
12 Nhada12@gmail.com 0772345678
13 Thiseeru13@gmail.com 0773456799
7
Result 6 x
Output Action Output
# Time Action Message Duration / Fetch
34 21:24:42 Create view userview12 AS SELECT COALESCE(ee.Employee_ID, ec.Employee_ID) AS Employee_ID, ee.Email, ec.Contact_info 0 row(s) affected 0.016 sec
35 21:25:31 select * from userview11 UNION select * from userview12 8 row(s) returned 0.000 sec / 0.000 sec
36 21:26:12 select * from userview11 UNION select * from userview12 8 row(s) returned 0.000 sec / 0.000 sec
Object Info Session

```

TUNING

Query optimization involves assessing the efficiency of a query by comparing data access before and after implementing an appropriate index. If the number of accessed rows decreases post-index establishment, it indicates successful tuning. All screenshots in the provided list meet this criteria. Each challenging query undergoes optimization through a series of steps for clarity. These steps involve removing existing externally built indexes, creating suitable indexes, and using commands such as EXPLAIN and SHOW INDEX to analyze data access and index structures before and after optimization.

➤ 1. Union

- Before

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, the following code is displayed:

```
1 use farm_management_system;
2 
3 EXPLAIN(SELECT * FROM SUPPLIER where Supplier_name='Ranil'
4 Union
5 SELECT * FROM SUPPLIER where Product_supplied='Glyphosate');
```

In the Result Grid tab, the EXPLAIN output is shown:

ID	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	SUPPLIER	ALL	ALL	ALL	ALL	ALL	const	7	14.29	Using where
2	UNION	SUPPLIER	ALL	ALL	ALL	ALL	ALL	const	7	14.29	Using where
3	UNION RESULT	cunion1,2>	ALL	ALL	ALL	ALL	ALL	const	3	100.00	Using temporary

The Action Output panel shows the execution of the EXPLAIN command:

- Action: 14:43:38 use farm_management_system
- Action: 14:44:12 EXPLAINSELECT * FROM SUPPLIER where Supplier_name='Ranil' Union SELECT * FROM SUPPLIER ... 3 row(s) returned

- After

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, the following code is displayed, showing the creation of indexes followed by the EXPLAIN output:

```
1 use farm_management_system;
2 
3 EXPLAIN(SELECT * FROM SUPPLIER where Supplier_name='Ranil'
4 Union
5 SELECT * FROM SUPPLIER where Product_supplied='Glyphosate');
6 
7 CREATE INDEX idx_supplier_name ON SUPPLIER (Supplier_name);
8 CREATE INDEX idx_product_supplied ON SUPPLIER (Product_supplied);
9 
10 EXPLAIN(SELECT * FROM SUPPLIER where Supplier_name='Ranil');
11 Union
12 
13 SELECT * FROM SUPPLIER where Product_supplied='Glyphosate');
```

In the Result Grid tab, the EXPLAIN output is shown:

ID	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	SUPPLIER	ALL	ref	idx_supplier_name	idx_supplier_name	202	const	2	100.00	Using where
2	UNION	SUPPLIER	ALL	ref	idx_product_supplied	idx_product_supplied	202	const	2	100.00	Using where
3	UNION RESULT	cunion1,2>	ALL	ALL	ALL	ALL	ALL	const	3	100.00	Using temporary

The Action Output panel shows the creation of indexes and the execution of the EXPLAIN command:

- Action: 14:46:38 CREATE INDEX idx_product_supplied ON SUPPLIER (Product_supplied)
- Action: 14:46:47 EXPLAINSELECT * FROM SUPPLIER where Supplier_name='Ranil' Union SELECT * FROM SUPPLIER ... 3 row(s) returned

➤ 2. Intersect

- Before

```

use farm_management_system;
DROP INDEX idx_supplier_name ON SUPPLIER ;
DROP INDEX idx_product_supplied ON SUPPLIER ;
explain
(SELECT * FROM SUPPLIER where Supplier_name='Ranil'
Intersect
SELECT * FROM SUPPLIER where Product_supplied='Glyphosate');

```

ID	Select Type	Table	Partitions	Type	Possible Keys	Key	Key Len	Ref	Rows	Filtered	Extra
1	PRIMARY	SUPPLIER	ALL	ref	idx_supplier_name	idx_supplier_name	102	const	2	100.00	
2	INTERSECT	SUPPLIER	ALL	ref	idx_product_supplied	idx_product_supplied	102	const	2	100.00	
3	INTERSECT RESULT	<intersect1,2>	ALL	ref	idx_supplier_name	idx_supplier_name	102	const	3	100.00	Using temporary

Action Output:

#	Time	Action	Message	Duration / Fetch
15	15:00:35	DROP INDEX idx_supplier_name ON SUPPLIER	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
16	15:00:35	DROP INDEX idx_product_supplied ON SUPPLIER	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
17	15:01:15	explain (SELECT * FROM SUPPLIER where Supplier_name='Ranil' Intersect SELECT * FROM SUPPLIE...	3 row(s) returned	0.000 sec / 0.000 sec

- After

```

use farm_management_system;
DROP INDEX idx_supplier_name ON SUPPLIER ;
DROP INDEX idx_product_supplied ON SUPPLIER ;
explain
(SELECT * FROM SUPPLIER where Supplier_name='Ranil'
Intersect
SELECT * FROM SUPPLIER where Product_supplied='Glyphosate');

```

ID	Select Type	Table	Partitions	Type	Possible Keys	Key	Key Len	Ref	Rows	Filtered	Extra
1	PRIMARY	SUPPLIER	ALL	ref	idx_supplier_name	idx_supplier_name	102	const	2	100.00	
2	INTERSECT	SUPPLIER	ALL	ref	idx_product_supplied	idx_product_supplied	102	const	2	100.00	
3	INTERSECT RESULT	<intersect1,2>	ALL	ref	idx_supplier_name	idx_supplier_name	102	const	3	100.00	Using temporary

Action Output:

#	Time	Action	Message	Duration / Fetch
19	15:03:01	CREATE INDEX idx_product_supplied ON SUPPLIER (Product_supplied)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.032 sec
20	15:03:09	explain (SELECT * FROM SUPPLIER where Supplier_name='Ranil' Intersect SELECT * FROM SUPPLIE...	3 row(s) returned	0.000 sec / 0.000 sec

➤ 3. Inner join

- Before

```

1 • use farm_management_system;
2 • Explain(
3   SELECT userview2.Head_farmer, userview1.Field_ID,userview2.Equipment_model
4   FROM
5     userview1
6   INNER JOIN userview2
7   ON userview2.Head_farmer = userview1.Field_owner);
8
9
10
11
12

```

ID	Select Type	Table	Partitions	Type	Possible Keys	Key	Key Len	Ref	Rows	Filtered	Extra
1	SIMPLE	equipment		ALL	#<key>	#<key>	#<key>		11	100.00	Using where
1	SIMPLE	field		ref	#<key>	#<key>	4	farm_management_system.equipment.Equipme...	1	100.00	Using index

Result 8 x

Action Output

#	Time	Action	Message	Duration / Fetch
20	15:03:09	explain (SELECT * FROM SUPPLIER where Supplier_name='Rani' Intersect SELECT * FROM SUPPLIE...)	3 row(s) returned	0.000 sec / 0.000 sec
21	15:13:43	Explain(SELECT userview2.Head_farmer, userview1.Field_ID,userview2.Equipment_model FROM us...)	2 row(s) returned	0.000 sec / 0.000 sec

- After

```

1 • use farm_management_system;
2 • Explain(
3   SELECT userview2.Head_farmer, userview1.Field_ID,userview2.Equipment_model
4   FROM
5     userview1
6   INNER JOIN userview2
7   ON userview2.Head_farmer = userview1.Field_owner);
8
9
10
11
12 • CREATE INDEX idx_field_owner ON FIELD (Field_owner);
13
14 • Explain(
15   SELECT userview2.Head_farmer, userview1.Field_ID,userview2.Equipment_model
16   FROM
17     userview1
18   INNER JOIN userview2
19   ON userview2.Head_farmer = userview1.Field_owner);
20
21

```

ID	Select Type	Table	Partitions	Type	Possible Keys	Key	Key Len	Ref	Rows	Filtered	Extra
1	SIMPLE	field		index	idx_field_owner	idx_field_owner	4	#<key>	7	100.00	Using index
1	SIMPLE	equipment		ref	#<key>	#<key>	5	farm_management_system.field.Field_owner	1	100.00	

Result 10 x

Action Output

#	Time	Action	Message	Duration / Fetch
1	15:19:26	CREATE INDEXidx_field_owner ON FIELD (Field_owner)	Error Code: 1061. Duplicate key name 'idx_field_owner'	0.000 sec
2	15:19:33	Explain(SELECT userview2.Head_farmer, userview1.Field_ID,userview2.Equipment_model FROM us...)	2 row(s) returned	0.000 sec / 0.000 sec

➤ 4. Nested 1

- Before

The screenshot shows the MySQL Workbench interface with the 'farm_management_system' database selected. In the SQL editor, the following query is displayed:

```

4 ● explain
5   (SELECT Employee_ID, Employee_name
6     FROM EMPLOYEE
7    WHERE Employee_ID IN (
8      SELECT Employee_ID
9        FROM SALARY
10       WHERE Monthly_salary < 30000
11    )
12   INTERSECT
13   (SELECT Employee_ID, Employee_name
14     FROM EMPLOYEE
15    WHERE Working_field = 500);

```

The 'Result Grid' pane shows the execution plan with the following details:

ID	Select Type	Table	Partitions	Type	Possible Keys	Key	Key Len	Ref	Rows	Filtered	Extra
1	PRIMARY	EMPLOYEE		ALL	PRIMARY	<auto_distinct_key>	4	farm_management_system.EMPLOYEE.Employee...	8	100.00	
1	PRIMARY	<subquery2>		eq_ref	<auto_distinct_key>	<auto_distinct_key>	4	farm_management_system.EMPLOYEE.Employee...	1	100.00	
2	MATERIALIZED	SALARY		ALL	PRIMARY		4	farm_management_system.SALARY.Monthly_...	16	33.33	Using where
3	INTERSECT	EMPLOYEE		ref	f11	f11	4	const	1	100.00	
4	INTERSECT RESULT	<intersect1>		ALL			4				Using temporary

The 'Output' pane shows the following log entries:

- 34 17:59:48 CREATE INDEX idx_employee_id_salary ON salary (Employee_ID)
- 35 17:59:51 explain(select * from userview9 left outer join userview10 ON userview9.Employee_ID = userview10.Employee_ID)
- 36 18:00:09 DROP INDEX idx_employee_id_salary ON salary
- 37 18:00:44 explain (SELECT Employee_ID, Employee_name FROM EMPLOYEE WHERE Employee_ID IN (

- After

The screenshot shows the MySQL Workbench interface with the 'farm_management_system' database selected. In the SQL editor, the following query is displayed:

```

15   WHERE Working_field = 500;
16
17   ● CREATE INDEX idx_monthly_salary_salary ON salary (Monthly_salary);
18   ● CREATE INDEX idx_working_field_employee ON employee (Working_field);
19
20   ● explain
21   (SELECT Employee_ID, Employee_name
22     FROM EMPLOYEE
23    WHERE Employee_ID IN (
24      SELECT Employee_ID
25        FROM SALARY
26       WHERE Monthly_salary < 30000
27    )
28   INTERSECT
29   (SELECT Employee_ID, Employee_name
30     FROM EMPLOYEE
31    WHERE Working_field = 500);
32

```

The 'Result Grid' pane shows the execution plan with the following details:

ID	Select Type	Table	Partitions	Type	Possible Keys	Key	Key Len	Ref	Rows	Filtered	Extra
1	PRIMARY	EMPLOYEE		ALL	PRIMARY	<auto_distinct_key>	4	farm_management_system.EMPLOYEE.Employee...	8	100.00	
1	PRIMARY	<subquery2>		eq_ref	<auto_distinct_key>	<auto_distinct_key>	4	farm_management_system.EMPLOYEE.Employee...	1	100.00	
2	MATERIALIZED	SALARY		range	PRIMARY	idx_monthly_salary_salary	5	farm_management_system.SALARY.Monthly_...	11	100.00	Using where; User...
3	INTERSECT	EMPLOYEE		ref	idx_working_field_employee	idx_working_field_employee	4	const	1	100.00	
4	INTERSECT RESULT	<intersect1>		ALL			4				Using temporary

The 'Output' pane shows the following log entries:

- 37 18:01:44 explain (SELECT Employee_ID, Employee_name FROM EMPLOYEE WHERE Employee_ID IN (
- 38 18:01:58 CREATE INDEX idx_monthly_salary_salary ON salary (Monthly_salary)

➤ 5. Nested 2

- Before

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

- Navigator:** Shows the schema `farm_management_system` with tables: crop, crop_pests, cultivate, employee, employee_contact_info, employee_email, equipment, field, harvest, head_farmer, head_farmer_contact, head_farmer_email, salary, supplier, supplier_contact_info, supplier_farmer_ordering, utilize.
- SQL Editor:** Contains the following SQL code:


```
1 ● use farm_management_system;
2
3
4 ● explain(
5   Select H.Farmer_ID from head_farmer as H where H.Farmer_name = 'Pathum'
6   AND H.Farmer_ID IN
7   (Select F.Field_owner from field as F where F.Size= 'Medium'));
```
- Result Grid:** Displays the execution plan for the query. The first row shows a simple table scan on table `H` with a key of `ALL` and a ref of `NULL`. The second row shows a simple table scan on table `F` with a key of `ref` and a ref of `idx_field_owner`.
- Action Output:** Shows two log entries:
 - Line 40: `explain(SELECT Employee_ID, Employee_NAME FROM EMPLOYEE WHERE Employee_ID IN (...))` returned 5 row(s).
 - Line 41: `explain(Select H.Farmer_ID from head_farmer as H where H.Farmer_name = 'Pathum' AND H.Farmer_ID ...)` returned 2 row(s).

- After

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

- Navigator:** Shows the schema `farm_management_system` with the same set of tables as the 'Before' state.
- SQL Editor:** Contains the following SQL code, including index creation statements:


```
1 ● use farm_management_system;
2
3
4 ● explain(
5   Select H.Farmer_ID from head_farmer as H where H.Farmer_name = 'Pathum'
6   AND H.Farmer_ID IN
7   (Select F.Field_owner from field as F where F.Size= 'Medium'));
```

 - CREATE INDEX idx_farmer_name_head_farmer ON head_farmer (Farmer_name);
 - CREATE INDEX idx_size_field ON field (Size);
 - explain(Select H.Farmer_ID from head_farmer as H where H.Farmer_name = 'Pathum' AND H.Farmer_ID IN (Select F.Field_owner from field as F where F.Size= 'Medium'));
- Result Grid:** Displays the execution plan for the query. The first row shows a simple table scan on table `H` with a key of `ref` and a ref of `PRIMARY, idx_farmer_name_head_farmer`. The second row shows a simple table scan on table `F` with a key of `ref` and a ref of `idx_field_owner, idx_size_field`.
- Action Output:** Shows two log entries:
 - Line 43: `CREATE INDEX idx_size_field ON field (Size)` affected Records: 0 Duplicates: 0 Warnings: 0 Duration / Fetch: 0.031 sec / 0.000 sec
 - Line 44: `explain(Select H.Farmer_ID from head_farmer as H where H.Farmer_name = 'Pathum' AND H.Farmer_ID ...)` returned 2 row(s). Duration / Fetch: 0.000 sec / 0.000 sec

➤ 6. Natural join

- Already Indexed

This is already tuned because of primary key.

The screenshot shows the MySQL Workbench interface. In the top-left pane, the 'Schemas' tree is expanded to show the 'farm_management_system' database, which contains tables like 'crop', 'cultivate', 'employee', etc. The 'Tables' section is selected. In the center, the 'SQL Editor' tab is active, displaying the following SQL code:

```
explain(
SELECT * FROM
userview3
NATURAL JOIN userview4);
```

Below the editor, the 'Result Grid' shows the execution plan for the EXPLAIN command:

ID	Select_Type	Table	Partitions	Type	Possible_Keys	Key	Key_Len	Ref	Rows	Filtered	Extra
1	SIMPLE	cultivate	NULL	index	PRIMARY	f17	4	NULL	6	100.00	Using index
1	SIMPLE	field	NULL	eq_ref	PRIMARY	PRIMARY	4	farm_management_system.cultivate.Field_ID	1	100.00	NULL

The 'Output' pane at the bottom shows the log of actions and their times:

#	Time	Action	Message	Duration / Fetch
15	18:48:02	explain (SELECT s.Employee_ID, s.Salary_month, s.Monthly_salary FROM salary s WHERE s.Payment...)	4 row(s) returned	0.016 sec / 0.000 sec
16	18:50:49	explain(SELECT userview2.Head_farmer, userview1.Field_ID userview2.Equipment_model FROM userv...)	2 row(s) returned	0.000 sec / 0.000 sec
17	18:55:14	explain(SELECT * FROM userview3 NATURAL JOIN userview4)	2 row(s) returned	0.000 sec / 0.000 sec

- 7. Left outer join
- Already indexed

This is already tuned because of primary key.

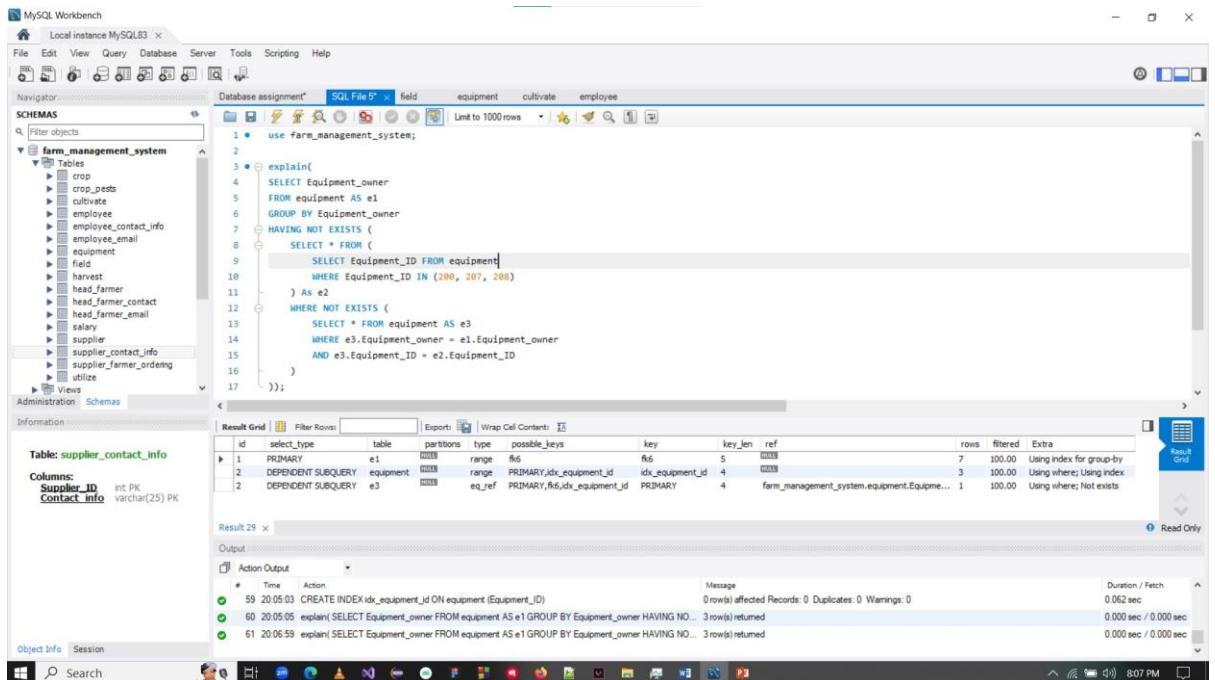
```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator Database assignment* SQL File* field equipment
SCHEMAS
Q Filter objects
farm_management_system
Tables
crop
crop_pests
cultivate
employee
employee_contact_info
employee_email
equipment
field
harvest
head_farmer
head_farmer_contact
head_farmer_email
salary
supplier
supplier_contact_info
supplier_farmer_ordering
utilize
Views
Administration Schemas
Information
Table: supplier_contact_info
Columns:
Supplier_ID int PK
Contact_Info varchar(25) PK
Result Grid Filter Rows: Export: Wrap Cell Content:
id select_type table partitions type possible_keys key key_len ref rows filtered Extra
1 SIMPLE supplier_farmer_ordering index idx_farmer_id_supplier_farmer_ordering 4 Using index
1 SIMPLE head_farmer eq_ref PRIMARY PRIMARY 4 farm_management_system.supplier_farmer_ordering
Read Only
Result 21 x
Output
Action Output
# Time Action Message Duration / Fetch
41 19:54:43 SELECT s.Employee_ID, s.Salary_month, s.Monthly_salary FROM salary s WHERE s.Payment_per_hour ... 1 row(s) returned 0.000 sec / 0.000 sec
42 19:54:56 explain( SELECT s.Employee_ID, s.Salary_month, s.Monthly_salary FROM salary s WHERE s.Payment_per_h... 2 row(s) returned 0.000 sec / 0.000 sec
43 19:57:01 explain(select * from userview5 left outer join userview6 ON userview5.Farmer_ID = userview6.Farmer_ID) 2 row(s) returned 0.000 sec / 0.000 sec
Object Info Session

```

► 8. Division

Already indexed.



The screenshot shows the MySQL Workbench interface with a complex SQL query being executed. The query involves multiple joins and subqueries, including an EXPLAIN statement to analyze the execution plan. The results grid shows the execution statistics for each part of the query, such as rows scanned, filtered, and extra information like index usage. The bottom pane displays the log of actions taken during the query execution.

```
use farm_management_system;
explain(
  SELECT Equipment_owner
  FROM equipment AS e1
  GROUP BY Equipment_owner
  HAVING NOT EXISTS (
    SELECT * FROM (
      SELECT Equipment_ID FROM equipment
      WHERE Equipment_ID IN (200, 207, 208)
    ) As e2
    WHERE NOT EXISTS (
      SELECT * FROM equipment AS e3
      WHERE e3.Equipment_owner = e1.Equipment_owner
      AND e3.Equipment_ID = e2.Equipment_ID
    )
  );
);

Result 29 x
Output
Action Output
# Time Action Message Duration / Fetch
59 20:05:03 CREATE INDEX idx_equipment_id ON equipment (Equipment_ID) 0 rows affected Records: 0 Duplicates: 0 Warnings: 0 0.062 sec
60 20:05:05 explain(SELECT Equipment_owner FROM equipment AS e1 GROUP BY Equipment_owner HAVING NO... 3 row(s) returned 0.000 sec / 0.000 sec
61 20:06:59 explain(SELECT Equipment_owner FROM equipment AS e1 GROUP BY Equipment_owner HAVING NO... 3 row(s) returned 0.000 sec / 0.000 sec
```

➤ 9. Set difference

- Before

```

use farm_management_system;
explain(SELECT * FROM SUPPLIER where Product_supplied='Glyphosate' EXCEPT SELECT * FROM SUPPLIER where Supplier_name='Ranii');

```

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	SUPPLIER			ref	idx_product_supplied1	idx_product_supplied1	102	const	2	100.00	
2	EXCEPT	SUPPLIER			ref	idx_supplier_name	idx_supplier_name	102	const	2	100.00	
3	EXCEPT RESULT	<except1,2>			ALL							Using temporary

- After

```

use farm_management_system;
explain(SELECT * FROM SUPPLIER where Product_supplied='Glyphosate' EXCEPT SELECT * FROM SUPPLIER where Supplier_name='Ranii');

```

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	s1			ref	idx_product_supplied1	idx_product_supplied1	102	const	2	100.00	
1	SIMPLE	s2			eq_ref	PRIMARY, idx_supplier_name	PRIMARY	4	farm_management_system.s1.Supplier_ID	1	100.00	Using where; Not exists

➤ 10. Nested 3

- Before

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Local instance MySQL83, Schemas (farm_management_system), Tables (crop, crop_pests, cultivate, employee, employee_contact_info, employee_email, equipment, field, harvest, head_farmer, head_farmer_contact, head_farmer_email, salary, supplier, supplier_contact_info, supplier_farmer_ordering, utilize).
- Database assignment:** SQL File 5*
- SQL Editor:**

```

15
16
17
18
19
20 • explain(select Employee_ID,Salary_month,Monthly_salary from salary where
21     Payment_per_hour > all(select Payment_per_hour from salary where Employee_ID IN (6,7)));
22
23
24
25
26
27
28
29
30
31

```
- Result Grid:**

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	salary	ALL	ALL			4	4	16	66.67	Using where
2	SUBQUERY	salary	range	PRIMARY, idx_employee_id_salary	PRIMARY	4	4	4	4	100.00	Using where
- Action Output:**

#	Time	Action	Message	Duration / Fetch
29	20:47:53	explain(select Employee_ID,Salary_month,Monthly_salary from salary where Payment_per_hour > all(select Payment_per_hour from salary where Employee_ID IN (6,7)))	2 row(s) returned	0.000 sec / 0.000 sec
30	20:47:58	explain(SELECT s1.Employee_ID, s1.Salary_month, s1.Monthly_salary FROM salary s1 JOIN (SELECT MAX(Payment_per_hour) AS max_payment FROM salary WHERE Employee_ID IN (6, 7)) s2 ON s1.Payment_per_hour > s2.max_payment)	3 row(s) returned	0.000 sec / 0.000 sec
31	20:48:10	explain(select Employee_ID,Salary_month,Monthly_salary from salary where Payment_per_hour > all(select Payment_per_hour from salary where Employee_ID IN (6,7)))	2 row(s) returned	0.000 sec / 0.000 sec

- After

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Local instance MySQL83, Schemas (farm_management_system), Tables (crop, crop_pests, cultivate, employee, employee_contact_info, employee_email, equipment, field, harvest, head_farmer, head_farmer_contact, head_farmer_email, salary, supplier, supplier_contact_info, supplier_farmer_ordering, utilize).
- Database assignment:** SQL File 5*
- SQL Editor:**

```

1 • use Farm_management_system;
2
3 • explain(
4     SELECT s1.Employee_ID, s1.Salary_month, s1.Monthly_salary
5     FROM salary s1
6     JOIN (
7         SELECT MAX(Payment_per_hour) AS max_payment
8         FROM salary
9         WHERE Employee_ID IN (6, 7)
10    ) s2 ON s1.Payment_per_hour > s2.max_payment;
11
12
13
14
15
16
17
18

```
- Result Grid:**

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	system	range			1	1	1	100.00	Using index condition
2	PRIMARY	s1	range	idx_payment_per_hour	idx_payment_per_hour	4	4	1	1	100.00	Using where
3	DERIVED	salary	range	PRIMARY, idx_employee_id_salary	PRIMARY	4	4	4	4	100.00	Using where
- Action Output:**

#	Time	Action	Message	Duration / Fetch
30	20:47:58	explain(SELECT s1.Employee_ID, s1.Salary_month, s1.Monthly_salary FROM salary s1 JOIN (SELECT MAX(Payment_per_hour) AS max_payment FROM salary WHERE Employee_ID IN (6, 7)) s2 ON s1.Payment_per_hour > s2.max_payment)	3 row(s) returned	0.000 sec / 0.000 sec
31	20:48:10	explain(select Employee_ID,Salary_month,Monthly_salary from salary where Payment_per_hour > all(select Payment_per_hour from salary where Employee_ID IN (6,7)))	2 row(s) returned	0.000 sec / 0.000 sec
32	20:50:48	explain(SELECT s1.Employee_ID, s1.Salary_month, s1.Monthly_salary FROM salary s1 JOIN (SELECT MAX(Payment_per_hour) AS max_payment FROM salary WHERE Employee_ID IN (6, 7)) s2 ON s1.Payment_per_hour > s2.max_payment)	3 row(s) returned	0.000 sec / 0.000 sec