

Table of Contents

1	Chapter 1-Requirement Analysis	1
2	Chapter 2-Conceptual Design	6
3	Chapter 3- Implementation	8
4	Chapter 4- Transactions	29
5	Chapter 5- Data Base Tuning.....	40

List of Figures

Figure 2-1: ER Diagram of the E-Waste Management System	6
Figure 2-2 : UML Class Diagram of the E-Waste Management System.....	7

1 Chapter 1-Requirement Analysis

Introduction

In today's rapidly advancing world, efficient waste management systems have become imperative for sustainable development and environmental preservation. Waste management not only ensures the cleanliness of our surroundings but also plays a crucial role in resource conservation and pollution reduction. As we strive towards creating a greener and healthier planet, the need for effective waste management practices is more pronounced than ever.

This mini-project endeavors to address the complexities inherent in managing and optimizing waste management systems. With waste generation on the rise due to population growth and urbanization, it is crucial to develop robust strategies for waste collection, disposal, and recycling. The database system proposed here aims to streamline these processes, offering a comprehensive solution for managing waste-related operations in Sri Lanka.

By delving into the organizational and operational intricacies of waste management departments, this project seeks to design and implement a database management system tailored to the specific needs of waste management authorities in Sri Lanka. Through careful analysis of functional and data requirements, the system aims to enhance the efficiency and effectiveness of waste management processes, from waste collection and segregation to recycling and disposal.

This project aims to develop a sophisticated database management system tailored to the needs of waste management in Sri Lanka. By providing a user-friendly interface and robust functionality, the system empowers administrative staff and stakeholders to make informed decisions, optimize resource allocation, and improve overall waste management outcomes. Through the utilization of data and technology, this project seeks to address the challenges associated with waste management, ultimately contributing to a cleaner, healthier, and more sustainable future for generations to come.

Functional Requirements

1. User authentication and authorization: The system should allow authorized users, such as administrative staff and waste management personnel, to log in securely with appropriate access levels.
2. Waste collection management: The system should facilitate the scheduling, tracking, and management of waste collection activities, including routes, pickup times, and types of waste to be collected.
3. Waste segregation and classification : The system should support the classification and segregation of different types of waste, such as recyclables, organic waste, and hazardous materials, to ensure proper handling and disposal.
4. Inventory Management: The system should maintain an inventory of waste management resources, including bins, containers, vehicles, and equipment, and track their availability, usage, and maintenance schedules.
5. Customer Management: The system should enable the registration and management of customers, such as households, businesses, and industrial facilities, including contact information, service subscriptions, and billing details.
6. Compliance and regulatory requirements: The system should help ensure compliance with local regulations and environmental standards governing waste management practices, including documentation and reporting obligations.

Data Requirements

Data requirements specify the specific data elements and attributes that the waste management database system needs to manage. Here are the data requirements for the waste management project.

List of the entities

1. User
2. WasteBin
3. WasteType
4. WasteCollector
5. WasteCollection
6. RegulatoryAgency
7. ViolationRecord
8. LandFill
9. WasteItem
10. ProcessingFacility
11. ProceedWaste

Entity	Attributes
User	Name
	Address
	Contact_Number
	User_ID
	User_Type
	User_Status
WasteBin	Bin_ID
	User_ID
	Location
	Capacity
	Bin_type
	Current_Fill_Level
WasteType	Type_ID
	Type_Name
WasteCollector	Name
	Collector_ID
	Contact_Number
WasteCollection	Collector_ID
	Bin_ID
	Collection_ID
	Collection_Date
RegulatoryAgency	Agency_ID
	Name
	Contact_Information
ViolationRecord	
	Agency_ID
	Description
LandFill	Date
	LandFill_ID
	Agency_ID
	Type_ID
	Capacity
WasteItem	Location
	Type_ID
	Collection_ID
	LandFill_ID
ProcessingFacility	Date
	Facility_ID
	Facility_Type
	Location
	Type_ID
	Proceed_ID

ProceedWaste	Facility_ID
	Data_Proceed
	Quantity

2 Chapter 2-Conceptual Design

ER Diagram

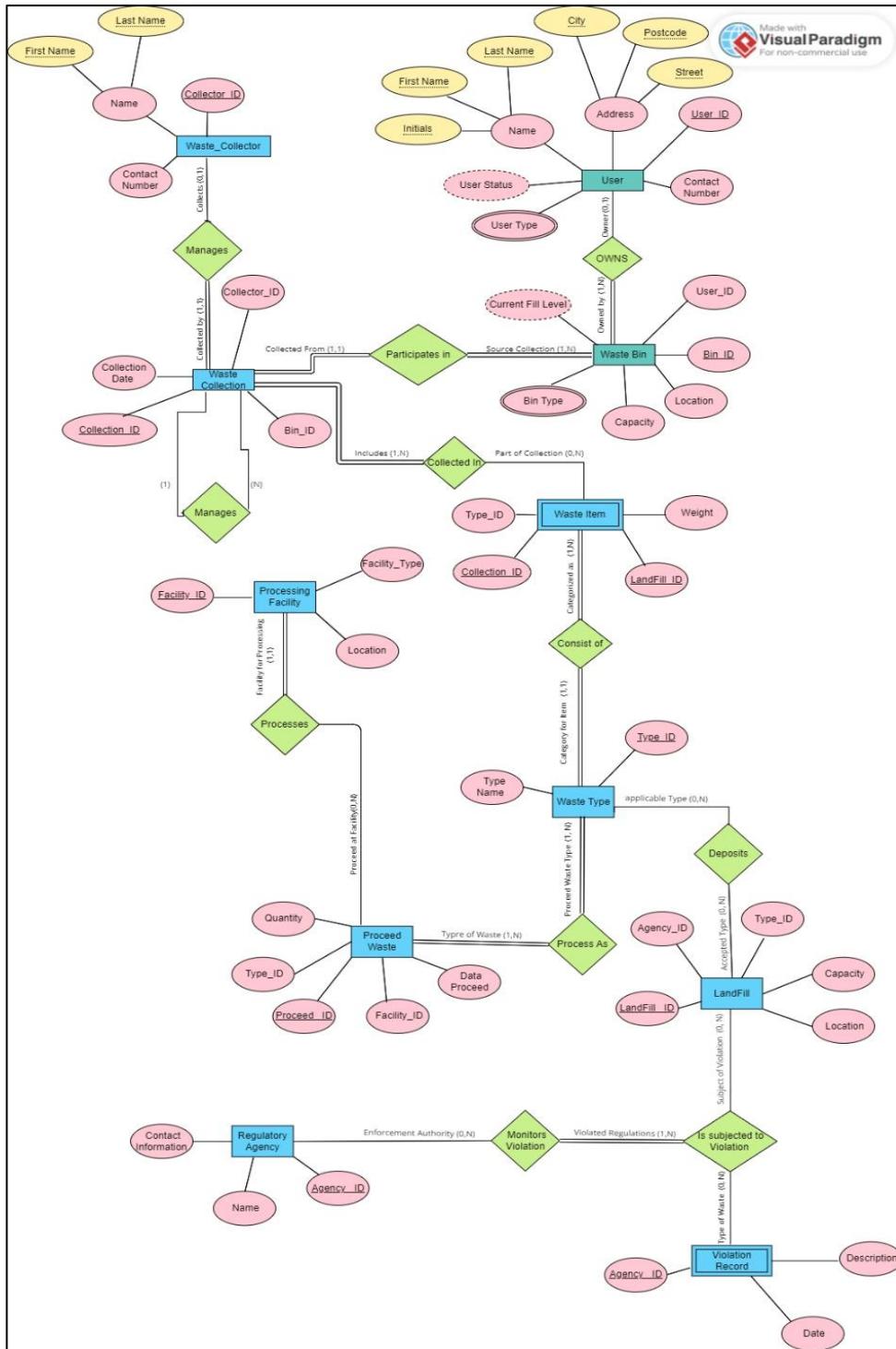


Figure 2-1: ER Diagram of the E-Waste Management System

UML Diagram

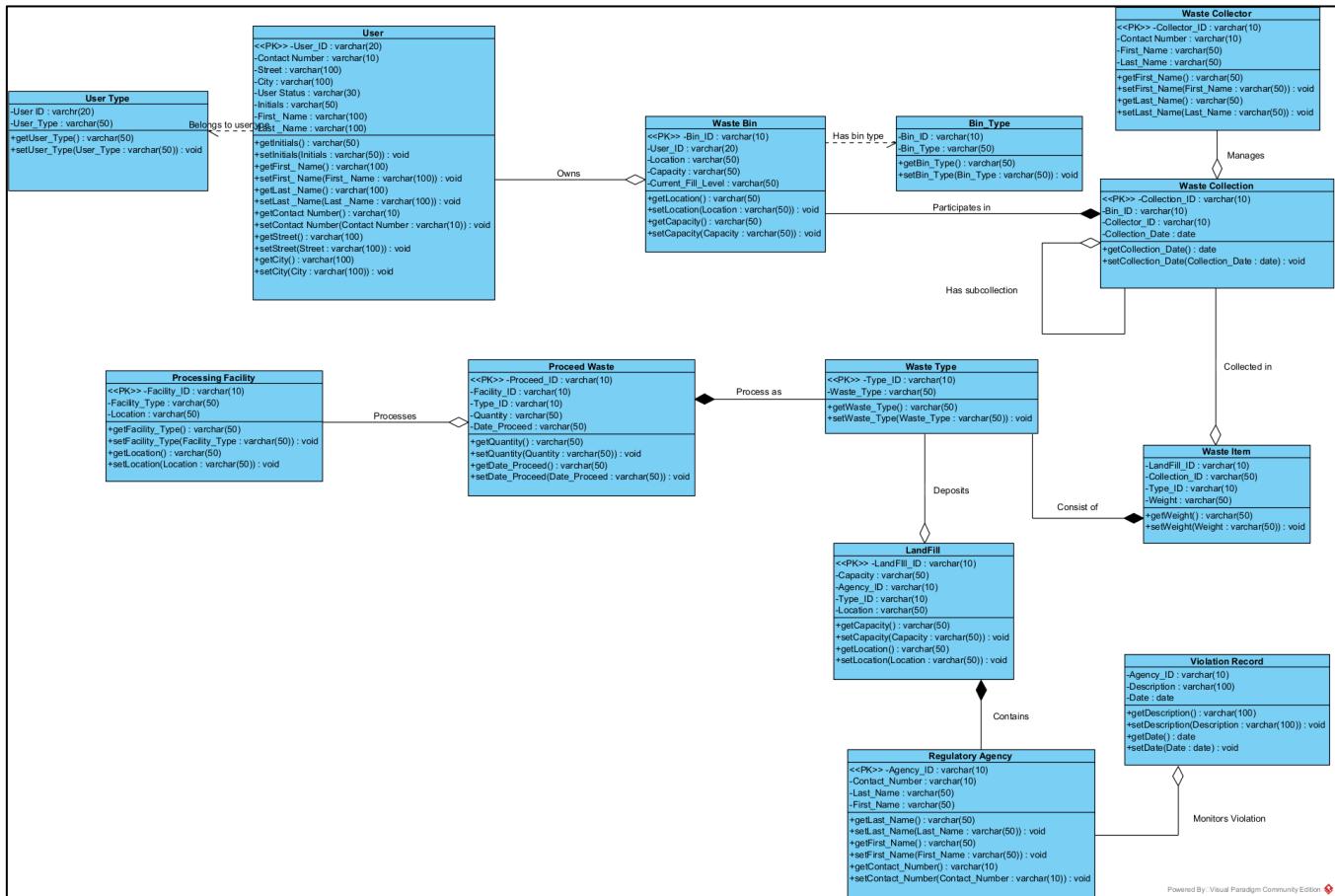


Figure 2-2 : UML Class Diagram of the E-Waste Management System

3 Chapter 3- Implementation

Creating and Use Data Base

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

- Navigator:** Shows the schema `miniproject` containing tables like `binstype`, `user`, `userstype`, `wastebin`, `wasteCollection`, `wastecollector`, and `wastetype`.
- SQL Editor (FOE*):** Displays the SQL code for creating the database and tables. The code includes:
 - Creating the database: `create database MiniProject;`
 - Selecting the database: `use MiniProject;`
 - Creating the `User` table with columns: `User_ID`, `Initials`, `First_Name`, `Last_Name`, `Street`, `City`, `Contact_No`, `User_State`, and a primary key `User_ID`.
 - Creating the `UserType` table with columns: `User_ID` and `User_Type`, and a primary key `(User_ID, User_Type)`. It includes a foreign key constraint `CONSTRAINT FK_UserType_User_ID FOREIGN KEY (User_ID) REFERENCES User (User_ID)` and a cascade delete rule `ON DELETE cascade ON UPDATE CASCADE`.
 - Creating the `WasteBin` table.
- Output:** Shows the execution log with entries for creating the database, selecting it, and creating the `User` and `UserType` tables. The log includes time, action, message, and duration.

Creating Tables

User Table

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

- Navigator:** Shows the schema `miniproject` containing tables like `binstype`, `user`, `userstype`, `wastebin`, `wasteCollection`, `wastecollector`, and `wastetype`.
- SQL Editor (FOE*):** Displays the SQL code for creating the database and tables. The code includes:
 - Creating the database: `create database MiniProject;`
 - Selecting the database: `use MiniProject;`
 - Creating the `User` table with columns: `User_ID`, `Initials`, `First_Name`, `Last_Name`, `Street`, `City`, `Contact_No`, `User_State`, and a primary key `User_ID`.
 - Creating the `UserType` table with columns: `User_ID` and `User_Type`, and a primary key `(User_ID, User_Type)`. It includes a foreign key constraint `CONSTRAINT FK_UserType_User_ID FOREIGN KEY (User_ID) REFERENCES User (User_ID)` and a cascade delete rule `ON DELETE cascade ON UPDATE CASCADE`.
 - Creating the `WasteBin` table.
- Output:** Shows the execution log with entries for creating the database, selecting it, and creating the `User` and `UserType` tables. The log includes time, action, message, and duration.

UserType Table

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

```
13  );
14  );
15  );
16 *+ create table UserType(
17     User_ID varchar(20) not null,
18     User_Type varchar(50) not null,
19     primary key (User_ID,User_Type),
20     CONSTRAINT FK_UserType_User_ID FOREIGN KEY (User_ID) REFERENCES User (User_ID)
21     ON DELETE cascade ON UPDATE CASCADE
22 );
23
24
25 *+ Create table WasteBin(
26     Bin_ID varchar(10) not null ,
27     User_ID varchar(20),
28     Location varchar(50),
29     Capacity varchar(50),
30     Current_FillLevel varchar(50),
31     primary key (Bin_ID),
32     CONSTRAINT FK_User_ID FOREIGN KEY (User_ID) REFERENCES User (User_ID)
```

The output pane shows two successful operations:

Action	Time	Message	Duration / Fetch
create table UserType(58 11:19:17	User_ID varchar(20) not null, User_Type varchar(50) not null, primary key (User_ID,User_Type), 0 row(s) affected	0.031 sec
Create table WasteBin(59 11:20:06	Bin_ID varchar(10) not null , User_ID varchar(20), Location varchar(50)... 0 row(s) affected	0.032 sec

WasteBin Table

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

```
19 );
20 );
21 );
22 );
23
24
25 *+ Create table WasteBin(
26     Bin_ID varchar(10) not null ,
27     User_ID varchar(20),
28     Location varchar(50),
29     Capacity varchar(50),
30     Current_FillLevel varchar(50),
31     primary key (Bin_ID),
32     CONSTRAINT FK_User_ID FOREIGN KEY (User_ID) REFERENCES User (User_ID)
33     ON DELETE CASCADE ON UPDATE CASCADE
34 );
35
36 *+ create table BinType(
37     Bin_ID varchar(10) not null,
38     Bin_Type varchar(50) not null,
39     primary key(Bin_ID,Bin_Type),
40     CONSTRAINT FKBinType_Bin_ID FOREIGN KEY (Bin_ID) REFERENCES WasteBin (Bin_ID)
```

The output pane shows two successful operations:

Action	Time	Message	Duration / Fetch
create table UserType(58 11:19:17	User_ID varchar(20) not null, User_Type varchar(50) not null, primary key (User_ID,User_Type), 0 row(s) affected	0.031 sec
Create table WasteBin(59 11:20:06	Bin_ID varchar(10) not null , User_ID varchar(20), Location varchar(50)... 0 row(s) affected	0.032 sec

BinType Table

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

```
28     Location varchar(50),
29     Capacity varchar(50),
30     Current_Filllevel varchar(50),
31     primary key (Bin_ID),
32     CONSTRAINT FK_User_ID FOREIGN KEY (User_ID) REFERENCES User (User_ID)
33     ON DELETE CASCADE ON UPDATE CASCADE
34   );
35
36 • create table BinType(
37     Bin_ID varchar(10) not null,
38     Bin_Type varchar(50) not null,
39     primary key(Bin_ID,Bin_Type),
40     CONSTRAINT FKBinType_Bin_ID FOREIGN KEY (Bin_ID) REFERENCES WasteBin (Bin_ID)
41     ON DELETE cascade ON UPDATE CASCADE
42   );
43
44
45 • create Table WasteType(
46     Type_ID varchar(10) not null ,
47     Wsate_Type varchar(50),
48
49
50 • create Table WasteCollector(
51     Collector_ID varchar(10) not null ,
52     First_Name varchar(50),
53     Last_Name varchar(50),
54     ContactNo varchar(50),
55     primary key (Collector_ID)
56
57
58 • CREATE TABLE WasteCollection (
59     Collection_ID varchar(10) NOT NULL,
60     Bin_ID varchar(10),
61     Collector_ID varchar(10),
62     CollectionDate,
```

The right panel displays a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

The output pane shows the results of the execution:

#	Time	Action	Message	Duration / Fetch
59	11:20:06	Create table WasteBin(Bin_ID varchar(10) not null , User_ID varchar(20), Location varchar(50), ...)	0 row(s) affected	0.032 sec
60	11:22:40	create table BinType(Bin_ID varchar(10) not null, Bin_Type varchar(50) not null, primary key(Bin_ID,Bin_Type), ...)	0 row(s) affected	0.016 sec

WasteType Table

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

```
43
44
45 • create Table WasteType(
46     Type_ID varchar(10) not null ,
47     Wsate_Type varchar(50),
48     primary key (Type_ID)
49   );
50 • ALTER TABLE WasteType
51     CHANGE COLUMN Wsate_Type Waste_Type varchar(50);
52
53
54 • create table WasteCollector(
55     Collector_ID varchar(10) not null ,
56     First_Name varchar(50),
57     Last_Name varchar(50),
58     ContactNo varchar(50),
59     primary key (Collector_ID)
60
61
62
63 • CREATE TABLE WasteCollection (
64     Collection_ID varchar(10) NOT NULL,
65     Bin_ID varchar(10),
66     Collector_ID varchar(10),
67     CollectionDate,
```

The right panel displays a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

The output pane shows the results of the execution:

#	Time	Action	Message	Duration / Fetch
55	11:19:05	create database MinProject	1row(s) affected	0.000 sec
56	11:19:08	use MinProject	0row(s) affected	0.000 sec
57	11:19:12	create table User(User_ID varchar(20) not null, Initials varchar(50), First_Name varchar(100), ...)	0 row(s) affected	0.015 sec

WasteCollector Table

The screenshot shows the MySQL Workbench interface with the 'Local instance MySQL80' connection selected. In the 'Navigator' pane, the 'Schemas' section shows the 'miniproject' schema with various tables like 'binstype', 'user', 'wastebin', 'wastecollector', etc. The 'SQL' tab is active, displaying the SQL code for creating the 'WasteCollector' table:

```
52
53
54 • create table WasteCollector(
55     Collector_ID varchar(10) not null ,
56     First_Name varchar(50),
57     Last_Name varchar(50),
58     ContactNo varchar(50),
59     primary key (Collector_ID)
60 );
61
62
63 • CREATE TABLE WasteCollection (
64     Collection_ID varchar(10) NOT NULL,
65     Bin_ID varchar(10),
66     Collector_ID varchar(10),
67     CollectionDate date,
68     Parent_Collection_ID varchar(10), -- This will establish the recursive relationship
69     PRIMARY KEY (Collection_ID),
70     CONSTRAINT FK_Bin_ID FOREIGN KEY (Bin_ID) REFERENCES WasteBin (Bin_ID) ON DELETE CASCADE ON UPDATE CASCADE,
71     CONSTRAINT FK_Collector_ID FOREIGN KEY (Collector_ID) REFERENCES WasteCollector (Collector_ID)
72     ON DELETE CASCADE ON UPDATE CASCADE,
73     CONSTRAINT FK_Parent_Collection_ID FOREIGN KEY (Parent_Collection_ID) REFERENCES WasteCollection (Collection_ID)
74     ON DELETE CASCADE ON UPDATE CASCADE
75 );
76
```

The 'Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
55	11:19:05	create database MiniProject	1row(s) affected	0.000 sec
56	11:19:08	use MiniProject	0row(s) affected	0.000 sec
57	11:19:12	create table User(User_ID varchar(20) not null, Initials varchar(50), First_Name varchar(100), ...)	0row(s) affected	0.015 sec

WasteCollection Table

The screenshot shows the MySQL Workbench interface with the 'Local instance MySQL80' connection selected. In the 'Navigator' pane, the 'Schemas' section shows the 'miniproject' schema with various tables like 'binstype', 'user', 'wastebin', 'wastecollector', etc. The 'SQL' tab is active, displaying the SQL code for creating the 'WasteCollection' table:

```
58
59
60 );
61
62
63 • CREATE TABLE WasteCollection (
64     Collection_ID varchar(10) NOT NULL,
65     Bin_ID varchar(10),
66     Collector_ID varchar(10),
67     CollectionDate date,
68     Parent_Collection_ID varchar(10), -- This will establish the recursive relationship
69     PRIMARY KEY (Collection_ID),
70     CONSTRAINT FK_Bin_ID FOREIGN KEY (Bin_ID) REFERENCES WasteBin (Bin_ID) ON DELETE CASCADE ON UPDATE CASCADE,
71     CONSTRAINT FK_Collector_ID FOREIGN KEY (Collector_ID) REFERENCES WasteCollector (Collector_ID)
72     ON DELETE CASCADE ON UPDATE CASCADE,
73     CONSTRAINT FK_Parent_Collection_ID FOREIGN KEY (Parent_Collection_ID) REFERENCES WasteCollection (Collection_ID)
74     ON DELETE CASCADE ON UPDATE CASCADE
75 );
76
77
78 • create table RegulatoryAgency(
79     Agency_ID varchar(10) not null,
80     First_Name varchar(50),
81     Last_Name varchar(50),
82     ContactNo varchar(50),
83 );
```

The 'Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
67	11:26:46	DROP TABLE WASTECOLLECTION	0row(s) affected	0.032 sec
68	11:27:16	CREATE TABLE WasteCollection (Collection_ID varchar(10) NOT NULL, Bin_ID varchar(10), Collector_ID varchar(10), CollectionDate date, Parent_Collection_ID varchar(10), -- This will establish the recursive relationship PRIMARY KEY (Collection_ID), CONSTRAINT FK_Bin_ID FOREIGN KEY (Bin_ID) REFERENCES WasteBin (Bin_ID) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT FK_Collector_ID FOREIGN KEY (Collector_ID) REFERENCES WasteCollector (Collector_ID) ON DELETE CASCADE ON UPDATE CASCADE)	0row(s) affected	0.047 sec

RegulatoryAgency Table

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

- Schemas:** The current schema is "miniproject".
- Tables:** The "Tables" section lists tables such as bin, bintype, collector, collectortype, user, usertype, wastebin, wastecollection, wastecollector, and wastetype.
- SQL Editor:** The F8 tab displays the SQL code for creating the "RegulatoryAgency" table. The code includes constraints for foreign keys (FK_Bin_ID, FK_Collector_ID, FK_Parent_Collection_ID) and primary keys (Agency_ID). It also creates a table "ViolationRecord" with columns Data, Agency_ID, and Description, and adds a constraint FK_ViolationRecord_Agency_ID.
- Output:** The Action Output pane shows the results of the operations:
 - Line 55: create database MiniProject. Message: 1 row(s) affected. Duration / Fetch: 0.000 sec.
 - Line 56: use MiniProject. Message: 0 row(s) affected. Duration / Fetch: 0.000 sec.
 - Line 57: create table User. Message: 0 row(s) affected. Duration / Fetch: 0.015 sec.

ViolationRecord Table

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

- Schemas:** The current schema is "miniproject".
- Tables:** The "Tables" section lists tables such as bin, bintype, regulatoryagency, user, usertype, wastebin, wastecollection, wastecollector, and wastetype.
- SQL Editor:** The F8 tab displays the SQL code for creating the "ViolationRecord" table. The code includes constraints for foreign keys (FK_ViolationRecord_Agency_ID) and primary keys (Agency_ID). It also creates a table "Landfill" with columns Landfill_ID, Type_ID, Agency_ID, Capacity, and Location.
- Output:** The Action Output pane shows the results of the operations:
 - Line 71: create table ViolationRecord. Message: 0 row(s) affected. Duration / Fetch: 0.031 sec.
 - Line 72: ALTER TABLE ViolationRecord CHANGE COLUMN Data Date DATE. Message: 0 rows affected, Records: 0, Duplicates: 0, Warnings: 0. Duration / Fetch: 0.047 sec.

LandFill Table

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

```
88 Agency_ID varchar(10),
89 Description varchar(50),
90 CONSTRAINT FK_ViolationRecord_Agency_ID FOREIGN KEY (Agency_ID) REFERENCES RegulatoryAgency (Agency_ID)
91 ON DELETE CASCADE ON UPDATE CASCADE
92 );
93 • ALTER TABLE ViolationRecord CHANGE COLUMN Data Date DATE;
94
95
96
97 • create table LandFill(
98     LandFill_ID varchar(10) not null,
99     Type_ID varchar(10),
100    Agency_ID varchar(10),
101    Capacity varchar(50),
102    Location varchar(50),
103    primary key(LandFill_ID),
104    CONSTRAINT FK_Type_ID FOREIGN KEY (Type_ID) REFERENCES WasteType (Type_ID)
105    ON DELETE CASCADE ON UPDATE CASCADE,
106    CONSTRAINT FK_Agency_ID FOREIGN KEY (Agency_ID) REFERENCES RegulatoryAgency (Agency_ID)
107    ON DELETE CASCADE ON UPDATE CASCADE
108 );
```

The output pane shows two actions:

#	Time	Action	Message	Duration / Fetch
75	11:37:37	DROP TABLE LANDFILL	0 row(s) affected	0.031 sec
76	11:37:45	create table LandFill(LandFill_ID varchar(10) not null, Type_ID varchar(10), Agency_ID varchar(10))	0 row(s) affected	0.047 sec

WasteItem Table

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

```
106 CONSTRAINT FK_Agency_ID FOREIGN KEY (Agency_ID) REFERENCES RegulatoryAgency (Agency_ID)
107 ON DELETE CASCADE ON UPDATE CASCADE
108 );
109
110
111
112
113
114
115 • create table WasteItem(
116     Type_ID varchar(10),
117     LandFill_ID varchar(10),
118     Collection_ID varchar(10),
119     Weight varchar(50),
120     CONSTRAINT FK_WasteItem_Type_ID FOREIGN KEY (Type_ID) REFERENCES WasteType (Type_ID)
121     ON DELETE CASCADE ON UPDATE CASCADE,
122     CONSTRAINT FK_WasteItem_Collection_ID FOREIGN KEY (Collection_ID) REFERENCES WasteCollection (Collection_ID)
123     ON DELETE CASCADE ON UPDATE CASCADE,
124     CONSTRAINT FK_WasteItem_LandFill_ID FOREIGN KEY (LandFill_ID) REFERENCES LandFill (LandFill_ID)
125     ON DELETE CASCADE ON UPDATE CASCADE
126 );
127
128
129
130 • create table ProcessingFacility(
```

The output pane shows two actions:

#	Time	Action	Message	Duration / Fetch
79	11:41:57	DROP TABLE WASTEITEM	Error Code: 1051. Unknown table 'miniproject.wasteitem'	0.000 sec
80	11:42:16	create table WasteItem(Type_ID varchar(10), LandFill_ID varchar(10), Collection_ID varchar(10))	0 row(s) affected	0.031 sec

ProcessingFacility Table

The screenshot shows the MySQL Workbench interface with the 'miniproject' schema selected. The SQL tab displays the creation of the 'ProcessingFacility' table:

```
124 CONSTRAINT FK_WasteItem_LandFill_ID FOREIGN KEY (LandFill_ID) REFERENCES LandFill (LandFill_ID)
125 ON DELETE CASCADE ON UPDATE CASCADE
126 )
127
128
129
130 • create table ProcessingFacility(
131     Facility_ID varchar(10) not null,
132     Facility_Type varchar(50),
133     Location varchar(50),
134     primary key (Facility_ID)
135 );
136
137 • create table ProceedWaste(
138     Proceed_ID varchar(10) not null,
139     Type_ID varchar(10),
140     Facility_ID varchar(10),
141     Quantity varchar(50),
142     Date_Proceed varchar(50),
143     primary key (Proceed_ID),
144     CONSTRAINT FK_ProceedWaste_Type_ID FOREIGN KEY (Type_ID) REFERENCES WasteType (Type_ID)
145     ON DELETE CASCADE ON UPDATE CASCADE,
146     CONSTRAINT FK_ProceedWaste_Facility_ID FOREIGN KEY (Facility_ID) REFERENCES ProcessingFacility (Facility_ID)
147     ON DELETE CASCADE ON UPDATE CASCADE
148 );
```

The Action Output pane shows two successful operations:

#	Time	Action	Message	Duration / Fetch
81	11:42:58	create table ProcessingFacility	Facility_ID varchar(10) not null, Facility_Type varchar(50), Locat... 0 row(s) affected	0.016 sec
82	11:43:41	create table ProceedWaste	Proceed_ID varchar(10) not null, Type_ID varchar(10), Facility_ID var... 0 row(s) affected	0.031 sec

ProceedWaste

The screenshot shows the MySQL Workbench interface with the 'miniproject' schema selected. The SQL tab displays the creation of the 'ProceedWaste' table and the insertion of data into the 'User' table:

```
130 • create table ProcessingFacility(
131     Facility_ID varchar(10) not null,
132     Facility_Type varchar(50),
133     Location varchar(50),
134     primary key (Facility_ID)
135 );
136
137 • create table ProceedWaste(
138     Proceed_ID varchar(10) not null,
139     Type_ID varchar(10),
140     Facility_ID varchar(10),
141     Quantity varchar(50),
142     Date_Proceed varchar(50),
143     primary key (Proceed_ID),
144     CONSTRAINT FK_ProceedWaste_Type_ID FOREIGN KEY (Type_ID) REFERENCES WasteType (Type_ID)
145     ON DELETE CASCADE ON UPDATE CASCADE,
146     CONSTRAINT FK_ProceedWaste_Facility_ID FOREIGN KEY (Facility_ID) REFERENCES ProcessingFacility (Facility_ID)
147     ON DELETE CASCADE ON UPDATE CASCADE
148 );
149
150
151 • insert into User(User_ID, Initials, First_Name, Last_Name, Street, City, Contact_No, User_State)
152 values
153 ('US0001', 'A.B.', 'Namal', 'Abeyasinghe', 'Muthugala Hawatha', 'Colombo', '0776758456', 'Married'),
154 ('US0002', 'P.', 'Kamala', 'Peris', 'Nawala Street', 'Kurunegala', '0773452123', 'Unmarried'),
```

The Action Output pane shows two successful operations:

#	Time	Action	Message	Duration / Fetch
81	11:42:58	create table ProcessingFacility	Facility_ID varchar(10) not null, Facility_Type varchar(50), Locat... 0 row(s) affected	0.016 sec
82	11:43:41	create table ProceedWaste	Proceed_ID varchar(10) not null, Type_ID varchar(10), Facility_ID var... 0 row(s) affected	0.031 sec

Inserting Data in to tables

User Table

The screenshot shows the MySQL Workbench interface with the 'User' table selected in the Navigator. The SQL Editor tab contains the following SQL code:

```
148
149
150
151 • insert into User(User_ID, Initials, First_Name, Last_Name, Street, City, Contact_No, User_State)
152 values
153 ('US0001', 'A.B.', 'Namal', 'Abeyasinghe', 'Muthugala Mawatha', 'Colombo', '0776758456', 'Married'),
154 ('US0002', 'P.', 'Kamala', 'Peris', 'Nawala Street', 'Kurunegala', '0773452123', 'Unmarried'),
155 ('US0003', 'C.D.', 'Kumudu', 'Chandrasekara', 'Mahinda Mawatha', 'Madampe', '0708867564', 'Unmarried'),
156 ('US0004', 'N.', 'Kumara', 'Mannaperuma', 'Rosa Street', 'Kaluthara', '0785673456', 'Married'),
157 ('US0005', 'R.', 'Saman', 'Ranathunge', 'Chandana Street', 'Matara', '0723456734', 'Married'),
158 ('US0006', 'Y.', 'Yoshitha', 'Somathilake', 'Mihindu Mawatha', 'Galle', '0706783456', 'Married')
159 • UPDATE User
160 SET First_Name = 'Amali', Last_Name = 'Perera', Contact_No = '0708889657'
161 WHERE User_ID = 'US0002';
162
163 • UPDATE User
```

The Result Grid shows the inserted data:

User_ID	Initials	First_Name	Last_Name	Street	City	Contact_No	User_State
US0001	A.B.	Namal	Abeyasinghe	Muthugala Mawatha	Colombo	0776758456	Married
US0002	P.	Kamala	Peris	Nawala Street	Kurunegala	0773452123	Unmarried
US0003	C.D.	Kumudu	Chandrasekara	Mahinda Mawatha	Madampe	0708867564	Unmarried
US0004	N.	Kumara	Mannaperuma	Rosa Street	Kaluthara	0785673456	Married
US0005	R.	Saman	Ranathunge	Chandana Street	Matara	0723456734	Married
US0006	Y.	Yoshitha	Somathilake	Mihindu Mawatha	Galle	0706783456	Married

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
83	11:48:18	insert into User(User_ID, Initials, First_Name, Last_Name, Street, City, Contact_No, User_State) values ('US0001', 'A.B.', 'Namal', 'Abeyasinghe', 'Muthugala Mawatha', 'Colombo', '0776758456', 'Married')	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.000 sec
84	11:48:27	select * from User LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

UserType Table

The screenshot shows the MySQL Workbench interface with the 'UserType' table selected in the Navigator. The SQL Editor tab contains the following SQL code:

```
171
172 • insert into UserType(User_ID, User_Type)
173 values
174 ('US0001', 'Domestic'),
175 ('US0002', 'Medical Center'),
176 ('US0006', 'Recycling Center'),
177 ('US0001', 'Commercial Institute'),
178 ('US0004', 'Educational Institute'),
179 ('US0005', 'Medical Center')
180
181 • UPDATE UserType
182 SET User_Type = 'Industrial Waste'
183 WHERE User_ID = 'US0005';
184
185 • UPDATE UserType
```

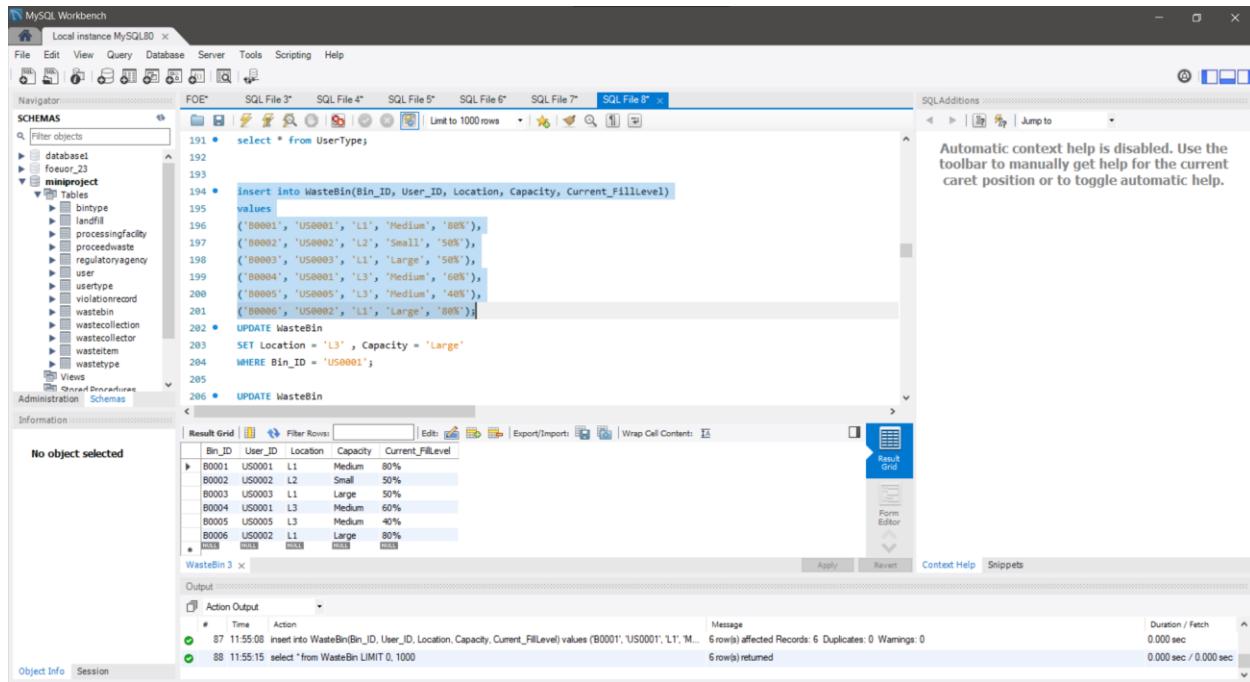
The Result Grid shows the inserted data:

User_ID	User_Type
US0001	Commercial Institute
US0002	Domestic
US0003	Medical Center
US0004	Educational Institute
US0005	Medical Center
US0006	Recycling Center

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
85	11:49:35	insert into UserType(User_ID, User_Type) values ('US0001', 'Domestic'), ('US0002', 'Medical Center'), ('US0006', 'Recycling Center')	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.016 sec
86	11:49:40	select * from UserType LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

WasteBin Table



The screenshot shows the MySQL Workbench interface with the 'WasteBin' table selected in the schema browser. The SQL editor contains the following code:

```

191 • select * from UserTypes
192
193
194 • insert into WasteBin(Bin_ID, User_ID, Location, Capacity, Current_FillLevel)
195 values
196 ('B0001', 'US0001', 'L1', 'Medium', '80%'),
197 ('B0002', 'US0002', 'L2', 'Small', '50%'),
198 ('B0003', 'US0003', 'L1', 'Large', '50%'),
199 ('B0004', 'US0001', 'L3', 'Medium', '60%'),
200 ('B0005', 'US0005', 'L3', 'Medium', '40%'),
201 ('B0006', 'US0002', 'L1', 'Large', '80%')
202 • UPDATE WasteBin
203 SET Location = 'L3', Capacity = 'Large'
204 WHERE Bin_ID = 'US0001';
205
206 • UPDATE WasteBin

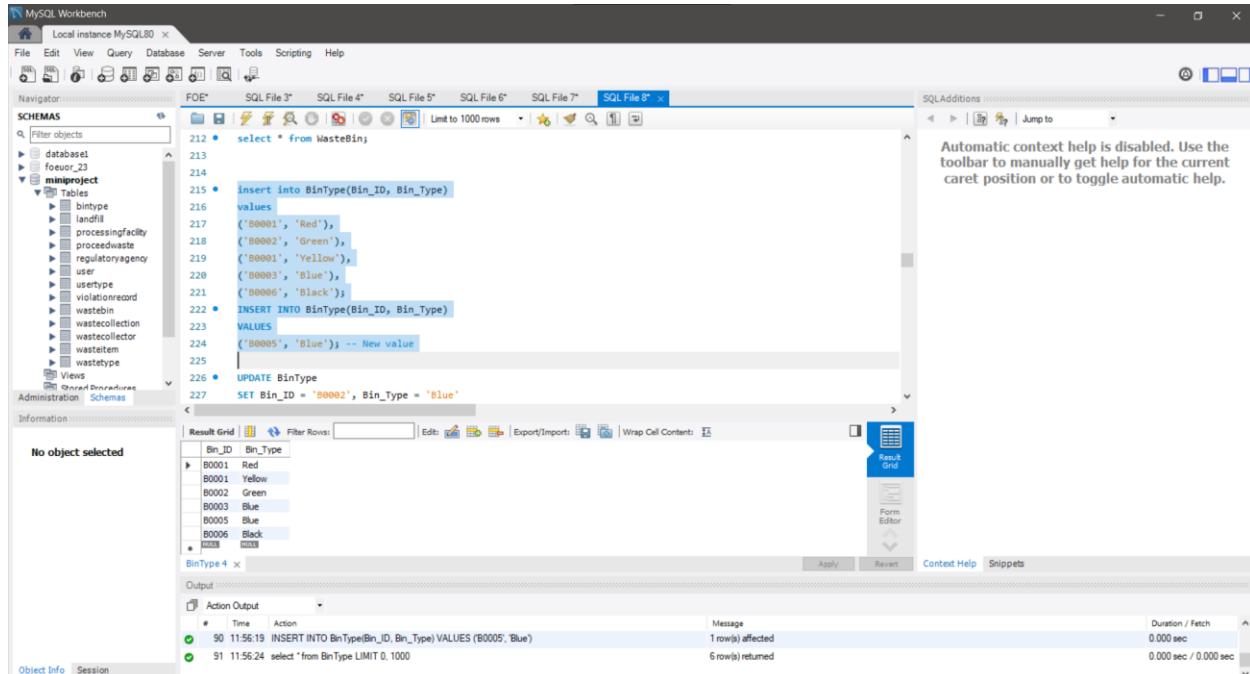
```

The Result Grid shows the following data:

Bin_ID	User_ID	Location	Capacity	Current_FillLevel
B0001	US0001	L1	Medium	80%
B0002	US0002	L2	Small	50%
B0003	US0003	L1	Large	50%
B0004	US0001	L3	Medium	60%
B0005	US0005	L3	Medium	40%
B0006	US0002	L1	Large	80%

The Action Output shows two rows affected by the update operations.

BinType Table



The screenshot shows the MySQL Workbench interface with the 'BinType' table selected in the schema browser. The SQL editor contains the following code:

```

212 • select * from WasteBin;
213
214
215 • insert into BinType(Bin_ID, Bin_Type)
216 values
217 ('B0001', 'Red'),
218 ('B0002', 'Green'),
219 ('B0003', 'Yellow'),
220 ('B0004', 'Blue'),
221 ('B0006', 'Black')
222 • INSERT INTO BinType(Bin_ID, Bin_Type)
223 VALUES
224 ('B0005', 'Blue') -- New value
225
226 • UPDATE BinType
227 SET Bin_ID = 'B0002', Bin_Type = 'Blue'

```

The Result Grid shows the following data:

Bin_ID	Bin_Type
B0001	Red
B0002	Yellow
B0003	Green
B0004	Blue
B0005	Blue
B0006	Black

The Action Output shows one row affected by the update operation.

WasteType Table

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (database1, foever_23, miniproject), Tables (bintype, landfill, processingfacility, proceedswaste, regulatoryagency, user, usertype, violationrecord, wastebin, wastecollection, wastecollector, wasteitem, wastetype).
- SQL Editor:** FDE*, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*. The code is as follows:

```
239
240
241
242 • insert into WasteType(Type_ID, Waste_Type)
243   values
244     ('T001', 'Biodegradable'),
245     ('T002', 'Non biodegradable'),
246     ('T003', 'Recyclable'),
247     ('T004', 'Electronic Waste'),
248     ('T005', 'Construction'),
249 • INSERT INTO WasteType(Type_ID, Waste_Type)
250   VALUES
251     ('T006', 'Hazardous Waste'); -- New value
252
253 • UPDATE WasteType
254   SET Waste_Type = 'Chemical'
```

- Result Grid:** Shows the data inserted into the WasteType table:

Type_ID	Waste_Type
T001	Biodegradable
T002	Non biodegradable
T003	Recyclable
T004	Electronic Waste
T005	Construction
T006	Hazardous Waste

- Action Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
93	11:57:08	INSERT INTO WasteType(Type_ID, Waste_Type) VALUES ('T006', 'Hazardous Waste')	1row(s) affected	0.000 sec
94	11:57:13	select * from wasteType LIMIT 0, 1000	6row(s) returned	0.000 sec / 0.000 sec

WasteCollector Table

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (database1, foever_23, miniproject), Tables (bintype, landfill, processingfacility, proceedswaste, regulatoryagency, user, usertype, violationrecord, wastebin, wastecollection, wastecollector, wasteitem, wastetype).
- SQL Editor:** FDE*, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*. The code is as follows:

```
263 • select * from wasteType;
264
265
266 • insert into WasteCollector(Collector_ID, First_Name, Last_Name, ContactNo)
267   values
268     ('C1001', 'Saman', 'Karunaratna', '0417589145'),
269     ('C1002', 'Sanath', 'Liyanage', '0912849751'),
270     ('C1003', 'Aspara', 'Pathirana', '0745684165'),
271     ('C1004', 'Gayani', 'Kularatna', '0784598654'),
272     ('C1005', 'Sumudu', 'Hewawasam', '0764578532'),
273     ('C1006', 'Namal', 'Gamage', '0758643851');
274 • UPDATE WasteCollector
275   SET First_Name = 'Kamal', Last_Name = 'Silva'
276   WHERE Collector_ID = 'C1001';
277
278 • UPDATE WasteCollector
```

- Result Grid:** Shows the data inserted into the WasteCollector table:

Collector_ID	First_Name	Last_Name	ContactNo
C1001	Saman	Karunaratna	0417589145
C1002	Sanath	Liyanage	0912849751
C1003	Aspara	Pathirana	0745684165
C1004	Gayani	Kularatna	0784598654
C1005	Sumudu	Hewawasam	0764578532
C1006	Namal	Gamage	0758643851

- Action Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
95	11:58:15	insert into WasteCollector(Collector_ID, First_Name, Last_Name, ContactNo) values ('C1001', 'Saman', 'Karunar... 6row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.015 sec	0.015 sec
96	11:58:20	select * from wastecollector LIMIT 0, 1000	6row(s) returned	0.000 sec / 0.000 sec

WasteCollection Table

The screenshot shows the MySQL Workbench interface with the 'WasteCollection' table selected. The SQL tab displays the following insert statements:

```
301
302
303
304 • insert into WasteCollection(Collection_ID, Bin_ID, Collector_ID, CollectionDate)
305 values
306 ('C0001', 'B0001', 'C1001', '2024-05-06'),
307 ('C0002', 'B0002', 'C1002', '2024-05-05'),
308 ('C0003', 'B0003', 'C1001', '2024-05-04'),
309 ('C0004', 'B0001', 'C1003', '2024-05-01'),
310 ('C0005', 'B0002', 'C1004', '2024-05-02')
311 • INSERT INTO WasteCollection(Collection_ID, Bin_ID, Collector_ID, CollectionDate)
312 VALUES
313 ('C0006', 'B0004', 'C1003', '2024-05-03'); -- New value
314 ('C0007', 'B0005', 'C1005', '2024-05-07'); -- New value
```

The Result Grid shows the inserted data:

Collection_ID	Bin_ID	Collector_ID	CollectionDate	Parent_Collection_ID
C0001	B0001	C1001	2024-05-06	NULL
C0002	B0002	C1002	2024-05-05	NULL
C0003	B0003	C1001	2024-05-04	NULL
C0004	B0001	C1003	2024-05-01	NULL
C0005	B0002	C1004	2024-05-02	NULL
C0006	B0004	C1003	2024-05-03	NULL
C0007	B0005	C1005	2024-05-07	NULL

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
98	12:00:36	INSERT INTO WasteCollection(Collection_ID, Bin_ID, Collector_ID, CollectionDate) VALUES ('C0006', 'B0004', 'C1003', '2024-05-03')	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
99	12:00:42	select * from wastecollection LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec

RegulatoryAgency Table

The screenshot shows the MySQL Workbench interface with the 'RegulatoryAgency' table selected. The SQL tab displays the following insert statements:

```
327
328 • insert into RegulatoryAgency(Agency_ID, First_Name, Last_Name, ContactNo)
329 values
330 ('A0001', 'Amal', 'Lokuge', '0714589615'),
331 ('A0002', 'Kamal', 'Ranaweera', '0774896584'),
332 ('A0003', 'Nimal', 'Ranathunga', '0764450890'),
333 ('A0004', 'Anura', 'Pathirana', '0742819980'),
334 ('A0005', 'Piyumi', 'Ileperuma', '0715008662')
335 • INSERT INTO RegulatoryAgency(Agency_ID, First_Name, Last_Name, ContactNo)
336 VALUES
337 ('A0006', 'Saman', 'Silva', '0771234567');-- New value
338
339 • UPDATE RegulatoryAgency
340 SET First_Name = 'Nuthu', Last_Name = 'Kumari'
341 WHERE Agency_ID = 1;
```

The Result Grid shows the inserted data:

Agency_ID	First_Name	Last_Name	ContactNo
A0001	Amal	Lokuge	0714589615
A0002	Kamal	Ranaweera	0774896584
A0003	Nimal	Ranathunga	0764450890
A0004	Anura	Pathirana	0742819980
A0005	Piyumi	Ileperuma	0715008662
A0006	Saman	Silva	0771234567

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
101	12:01:38	INSERT INTO RegulatoryAgency(Agency_ID, First_Name, Last_Name, ContactNo) VALUES ('A0006', 'Saman', 'Silva', '0771234567')	1 row(s) affected	0.000 sec
102	12:01:43	select * from RegulatoryAgency LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

ViolationRecord Table

The screenshot shows the MySQL Workbench interface with the 'ViolationRecord' table selected. The SQL editor contains the following code:

```

352
353 • insert into ViolationRecord(Date, Agency_ID, Description)
354     values
355     ('2024-05-08', 'A0001', null),
356     ('2024-05-08', 'A0002', null),
357     ('2024-05-08', 'A0001', null),
358     ('2024-05-07', 'A0003', null),
359     ('2024-05-08', 'A0002', null);
360 • INSERT INTO ViolationRecord(Date, Agency_ID, Description)
361     VALUES
362     ('2024-05-09', 'A0004', 'Improper waste disposal'),
363     ('2024-05-10', 'A0005', 'Unauthorized dumping of hazardous waste');
364 • UPDATE ViolationRecord
365     SET Date = '2023-09-08', Description = 'The restaurant persistently disregarded warnings and continued to violate wa...
    WHERE Agency_ID = 'A0005';
  
```

The Result Grid shows the following data:

Date	Agency_ID	Description
2024-05-08	A0001	null
2024-05-08	A0002	null
2024-05-08	A0001	null
2024-05-07	A0003	null
2024-05-08	A0002	null
2024-05-09	A0004	Improper waste disposal
2024-05-10	A0005	Unauthorized dumping of hazardous waste

The Action Output shows two successful operations:

- 104 12:02:59 INSERT INTO ViolationRecord(Date, Agency_ID, Description) VALUES ('2024-05-09', 'A0004', 'Improper waste disposal') 2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0 Duration / Fetch: 0.016 sec
- 105 12:03:13 select * from ViolationRecord LIMIT 0, 1000 7 row(s) returned 0.000 sec / 0.000 sec

LandFill Table

The screenshot shows the MySQL Workbench interface with the 'LandFill' table selected. The SQL editor contains the following code:

```

376
377
378 • insert into LandFill(LandFill_ID, Type_ID, Agency_ID, Capacity, Location)
379     values
380     ('L001', 'T001', 'A0001', 'Medium', 'land1'),
381     ('L002', 'T002', 'A0002', 'Small', 'land2'),
382     ('L003', 'T004', 'A0001', 'Large', 'land3'),
383     ('L004', 'T003', 'A0003', 'Medium', 'land4'),
384     ('L005', 'T004', 'A0002', 'Large', 'land5');
385 • INSERT INTO LandFill(LandFill_ID, Type_ID, Agency_ID, Capacity, Location)
386     VALUES
387     ('L006', 'T005', 'A0004', 'Medium', 'land6');
388 • UPDATE landFill
  
```

The Result Grid shows the following data:

LandFill_ID	Type_ID	Agency_ID	Capacity	Location
L001	T001	A0001	Medium	land1
L002	T002	A0002	Small	land2
L003	T004	A0001	Large	land3
L004	T003	A0003	Medium	land4
L005	T004	A0002	Large	land5
L006	T005	A0004	Medium	land6

The Action Output shows two successful operations:

- 107 12:04:10 INSERT INTO LandFill(LandFill_ID, Type_ID, Agency_ID, Capacity, Location) VALUES ('L006', 'T005', 'A0004', 'Medium', 'land6') 1 row(s) affected Duration / Fetch: 0.000 sec
- 108 12:04:17 select * from LandFill LIMIT 0, 1000 6 row(s) returned 0.000 sec / 0.000 sec

WasteItem Table

The screenshot shows the MySQL Workbench interface with the 'WasteItem' table being populated. The SQL editor contains the following code:

```
402
403 • insert into WasteItem(Type_ID, LandFill_ID, Collection_ID, Weight)
values
405   ('T001', 'L001', 'C0001', '45kg'),
406   ('T002', 'L002', 'C0002', '30kg'),
407   ('T003', 'L003', 'C0003', '25kg'),
408   ('T004', 'L005', 'C0001', '40kg'),
409   ('T003', 'L005', 'C0001', '35kg'),
410   ('T004', 'L001', 'C0002', '50kg');
411
412 • UPDATE WasteItem
413   SET Weight = '10kg'
414 WHERE Type_ID = 'T004' and LandFill_ID = 'L005';
```

The Result Grid shows the inserted data:

Type_ID	LandFill_ID	Collection_ID	Weight
T001	L001	C0001	45kg
T002	L002	C0002	30kg
T003	L003	C0003	25kg
T004	L005	C0001	40kg
T003	L005	C0001	35kg
T004	L001	C0002	50kg

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
109	12:05:11	Insert into WasteItem(Type_ID, LandFill_ID, Collection_ID, Weight) values ('T001', 'L001', 'C0001', '45kg'), ('T002', 'L002', 'C0002', '30kg'), ('T003', 'L003', 'C0003', '25kg'), ('T004', 'L005', 'C0001', '40kg'), ('T003', 'L005', 'C0001', '35kg'), ('T004', 'L001', 'C0002', '50kg');	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.015 sec
110	12:05:15	select * from WasteItem LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

ProcessingFacility Table

The screenshot shows the MySQL Workbench interface with the 'ProcessingFacility' table being populated. The SQL editor contains the following code:

```
422 • select * from WasteItem;
423
424 • insert into ProcessingFacility(Facility_ID, Facility_Type, Location)
values
425
426   ('F001', 'Recycling Facility', 'L1'),
427   ('F002', 'Composting Facility', 'L2'),
428   ('F003', 'Waste-to-Energy Facility', 'L6'),
429   ('F004', 'Recycling Center', 'L3'),
430   ('F005', 'Incineration Facility', 'L4'),
431   ('F006', 'Biogas Plant', 'L5');
432 • UPDATE ProcessingFacility
433   SET Location = 'L7'
434 WHERE Facility_ID = 'F004' and Facility_Type = 'Recycling Facility';
```

The Result Grid shows the inserted data:

Facility_ID	Facility_Type	Location
F001	Recycling Facility	L1
F002	Composting Facility	L2
F003	Waste-to-Energy Facility	L6
F004	Recycling Center	L3
F005	Incineration Facility	L4
F006	Biogas Plant	L5

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
111	12:05:55	Insert into ProcessingFacility(Facility_ID, Facility_Type, Location) values ('F001', 'Recycling Facility', 'L1'), ('F002', 'Composting Facility', 'L2'), ('F003', 'Waste-to-Energy Facility', 'L6'), ('F004', 'Recycling Center', 'L3'), ('F005', 'Incineration Facility', 'L4'), ('F006', 'Biogas Plant', 'L5');	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.015 sec
112	12:06:04	select * from ProcessingFacility LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

ProceedWaste Table

The screenshot shows the MySQL Workbench interface with the 'miniproject' schema selected. The SQL editor tab contains the following SQL code:

```
451 ('P0006', 'T005', '30%', 'F003', '2024-05-11');
452 • UPDATE ProceedWaste
453 SET Quantity = '10%', Date_Proceed = '2024-02-03'
454 WHERE Proceed_ID = 'P0004';
455
456 • UPDATE ProceedWaste
457 SET Quantity = '20%', Date_Proceed = '2024-04-11'
458 WHERE Proceed_ID = 'P0002';
459
460 • DELETE FROM ProceedWaste WHERE Proceed_ID = 'P0001';
461
462 • select * from ProceedWaste;
463
```

The Result Grid shows the following data:

Proceed_ID	Type_ID	Facility_ID	Quantity	Date_Proceed
P0001	T001	F001	80%	2024-05-09
P0002	T002	F002	50%	2024-05-09
P0003	T004	F003	40%	2024-05-09
P0004	T003	F001	65%	2024-05-11
P0005	T004	F002	50%	2024-05-08
P0006	T005	F003	30%	2024-05-11
*	NULL	NULL	NULL	NULL

The Output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
113	12:06:51	Insert into ProceedWaste(Proceed_ID, Type_ID, Facility_ID, Quantity, Date_Proceed) values ('P0001', 'T001', ..., '2024-05-11')	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.000 sec
114	12:06:57	select * from ProceedWaste LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

Delete and Update Tables

User Table

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

- Navigator:** Shows the schema structure under "miniproject" with tables like bintype, landfill, processingfacility, procedwaste, regulatoryagency, user, usertype, violationrecord, wastebin, wastecollection, wastecollector, wasteitem, and wastetype.
- SQL Editor:** Displays several SQL statements:
 - UPDATE User SET First_Name = 'Amali', Last_Name = 'Perera', Contact_No = '0708896567' WHERE User_ID = 'US0002';
 - UPDATE User SET First_Name = 'Lusi', Last_Name = 'Fedrick', Street = 'Mahinda Mawatha', City = 'Colombo' WHERE User_ID = 'US0004';
 - DELETE FROM User WHERE User_ID = 'US0006';
 - select * from User;
- Result Grid:** Shows the current state of the User table with the following data:

User_ID	Initials	First_Name	Last_Name	Street	City	Contact_No	User_Status
US0001	A.B.	Namal	Abeynighe	Muthugala Mawatha	Colombo	0776758456	Married
US0002	P.	Amali	Perera	Newala Street	Kurunegala	0708896567	Unmarried
US0003	C.D.	Kumudu	Chandrasekara	Mahinda Mawatha	Madapape	0708895564	Unmarried
US0004	K.L.	Lusi	Fedrick	Mahinda Mawatha	Colombo	0795673456	Married
US0005	R.	Saman	Ranathunge	Chandana Street	Matale	0723456734	Married

- Action Output:** Shows the results of the executed queries:
 - 117 12:09:45 DELETE FROM User WHERE User_ID = 'US0006' Message 1 row(s) affected 0.016 sec
 - 118 12:09:48 select * from User LIMIT 0, 1000 Message 5 row(s) returned 0.000 sec / 0.000 sec

UserType Table

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

- Navigator:** Shows the schema structure under "miniproject" with tables like bintype, landfill, processingfacility, procedwaste, regulatoryagency, user, usertype, violationrecord, wastebin, wastecollection, wastecollector, wasteitem, and wastetype.
- SQL Editor:** Displays several SQL statements:
 - UPDATE UserType SET User_Type = 'Industrial Waste' WHERE User_ID = 'US0006';
 - UPDATE Usertype SET User_Type = 'Government Institutes' WHERE User_ID = 'US0005';
 - DELETE FROM Usertype WHERE User_ID = 'US0004';
 - select * from UserType;
- Result Grid:** Shows the current state of the UserType table with the following data:

User_ID	User_Type
US0001	Commercial Institute
US0001	Domestic
US0002	Medical Center
US0005	Government Institutes

- Action Output:** Shows the results of the executed queries:
 - 121 12:10:33 DELETE FROM Usertype WHERE User_ID = 'US0004' Message 1 row(s) affected 0.000 sec
 - 122 12:10:36 select * from UserType LIMIT 0, 1000 Message 4 row(s) returned 0.000 sec / 0.000 sec

WasteBin Table

The screenshot shows the MySQL Workbench interface with the 'WasteBin' table selected. The SQL editor contains the following code:

```
281 ('B0006', 'US0002', 'L1', 'Large', '80%');
282 • UPDATE WasteBin
283     SET Location = 'L3' , Capacity = 'Large'
284     WHERE Bin_ID = 'US0001';
285
286 • UPDATE WasteBin
287     SET Capacity = 'Medium', Current_FillLevel = '30'
288     WHERE Bin_ID = 'B0006';
289
290 • DELETE FROM WasteBin WHERE Bin_ID = 'B0005';
291
292 • select * from WasteBin;
```

The Result Grid shows the following data:

Bin_ID	User_ID	Location	Capacity	Current_FillLevel
B0001	US0001	L1	Medium	80%
B0002	US0002	L2	Small	50%
B0003	US0003	L1	Large	50%
B0004	US0001	L3	Medium	60%
B0006	US0002	L1	Medium	30
NULL	NULL	NULL	NULL	NULL

The Output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
125	12:11:44	DELETE FROM WasteBin WHERE Bin_ID = 'B0005'	1row(s) affected	0.016 sec
126	12:11:48	select * from WasteBin LIMIT 0, 1000	5row(s) returned	0.000 sec / 0.000 sec

BinType Table

The screenshot shows the MySQL Workbench interface with the 'BinType' table selected. The SQL editor contains the following code:

```
225
226 • UPDATE BinType
227     SET Bin_ID = 'B0002' , Bin_Type = 'Blue'
228     WHERE Bin_ID = 'B0003';
229
230 • UPDATE BinType
231     SET Bin_ID = 'B0001' , Bin_Type = 'Black'
232     WHERE Bin_ID = 'B0003';
233
234 • DELETE FROM BinType WHERE Bin_ID = 'B0006';
235
236 • select * from BinType;
```

The Result Grid shows the following data:

Bin_ID	Bin_Type
B0001	Red
B0001	Yellow
B0002	Blue
B0002	Green
NULL	NULL

The Output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
129	12:12:33	DELETE FROM BinType WHERE Bin_ID = 'B0006'	1row(s) affected	0.000 sec
130	12:12:37	select * from BinType LIMIT 0, 1000	4row(s) returned	0.000 sec / 0.000 sec

WasteType Table

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

Schemas: miniproject

SQL Editor: SQL File 8*

Result Grid:

Type_ID	Waste_Type
T001	Chemical
T002	Composite
T003	Recyclable
T004	Electronic Waste
T006	Hazardous Waste
NULL	NULL

Action Output:

#	Time	Action	Message	Duration / Fetch
133	12:13:16	DELETE FROM WasteType WHERE Type_ID = 'T005'	1row(s) affected	0.000 sec
134	12:13:19	select * from wasteType	5row(s) returned	0.000 sec / 0.000 sec

WasteCollector Table

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

Schemas: miniproject

SQL Editor: SQL File 8*

Result Grid:

Collector_ID	First_Name	Last_Name	ContactNo
C1001	Kamal	Silva	0417589145
C1002	Kamani	Perera	0912849751
C1003	Apsara	Pathirana	0745684165
C1004	Gayani	Kularathna	0784598654
C1005	Sunudu	Heinawasana	0764578532
NULL	NULL	NULL	NULL

Action Output:

#	Time	Action	Message	Duration / Fetch
137	12:14:39	DELETE FROM WasteCollector WHERE Collector_ID = 'C1006'	1row(s) affected	0.000 sec
138	12:14:42	select * from wastecollector	5row(s) returned	0.000 sec / 0.000 sec

WasteCollection Table

The screenshot shows the MySQL Workbench interface with the 'WasteCollection' table selected. The SQL editor contains the following code:

```
383 ('C0007', 'B0085', 'C1005', '2024-05-07'); -- New value
384 • UPDATE WasteCollection
385     SET CollectionDate = '2024-03-5'
386     WHERE Collection_ID = 'C0003';
387
388 • UPDATE WasteCollection
389     SET CollectionDate = '2024-03-6'
390     WHERE Collection_ID = 'C0002';
391
392 • DELETE FROM WasteCollection WHERE Collection_ID = 'C0005';
393
394 • select * from wastecollection;
```

The Result Grid shows the following data:

Collection_ID	Bin_ID	Collector_ID	CollectionDate	Parent_Collection_ID
C0001	B0001	C1001	2024-05-06	NULL
C0002	B0002	C1002	2024-03-06	NULL
C0003	B0003	C1001	2024-03-05	NULL
C0004	B0001	C1003	2024-05-01	NULL
C0006	B0004	C1003	2024-05-03	NULL
NULL	NULL	NULL	NULL	NULL

The Output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
141	12:15:58	DELETE FROM WasteCollection WHERE Collection_ID = 'C0005'	1 row(s) affected	0.000 sec
142	12:16:01	select * from wastecollection LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

RegulatoryAgency Table

The screenshot shows the MySQL Workbench interface with the 'RegulatoryAgency' table selected. The SQL editor contains the following code:

```
327
328 • UPDATE RegulatoryAgency
329     SET First_Name = 'Muthu', Last_Name = 'Kumari'
330     WHERE Agency_ID = 'A0004';
331
332 • UPDATE RegulatoryAgency
333     SET First_Name = 'Uresha', Last_Name = 'Herath', ContactNo = '0776857356'
334     WHERE Agency_ID = 'A0002';
335
336 • DELETE FROM RegulatoryAgency WHERE Agency_ID = 'A0001';
337
338 • select * from RegulatoryAgency;
```

The Result Grid shows the following data:

Agency_ID	First_Name	Last_Name	ContactNo
A0002	Uresha	Herath	0776857356
A0003	Nimal	Ranathunga	0764450890
A0004	Muthu	Kumari	0742019980
A0005	Piyumi	Ieperuma	0715000662
A0006	Saman	Silva	0771234567
NULL	NULL	NULL	NULL

The Output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
145	12:16:43	DELETE FROM RegulatoryAgency WHERE Agency_ID = 'A0001'	1 row(s) affected	0.000 sec
146	12:16:46	select * from RegulatoryAgency LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

ViolationRecord Table

The screenshot shows the MySQL Workbench interface with the 'ViolationRecord' table selected. The SQL editor contains the following code:

```
352 ('2024-05-18', 'A0005', 'Unauthorized dumping of hazardous waste');
353 • UPDATE ViolationRecord
354 SET Date = '2023-09-08', Description = 'The restaurant persistently disregarded warnings and continued
355 WHERE Agency_ID = 'A0001';
356
357 • UPDATE ViolationRecord
358 SET Date = '2024-05-07', Description = 'dumping hazardous waste'
359 WHERE Agency_ID = 'A0002';
360
361 • DELETE FROM ViolationRecord WHERE Agency_ID = 'A0003';
362
363 • select * from ViolationRecord;
```

The Result Grid shows the following data:

Date	Agency_ID	Description
2024-05-07	A0002	dumping hazardous waste
2024-05-07	A0002	dumping hazardous waste
2024-05-09	A0004	Improper waste disposal
2024-05-10	A0005	Unauthorized dumping of hazardous waste

The Action Output shows two actions:

#	Time	Action	Message	Duration / Fetch
149	12:17:34	DELETE FROM ViolationRecord WHERE Agency_ID = 'A0003'	1 row(s) affected	0.000 sec
150	12:17:38	select * from ViolationRecord LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

LandFill Table

The screenshot shows the MySQL Workbench interface with the 'LandFill' table selected. The SQL editor contains the following code:

```
376 ('L003', 'T003', 'A0003', 'Small', 'land3');
377 • Execute the statement under the keyboard cursor;
378 SET Capacity = 'Small', Location = 'land3';
379 WHERE LandFill_ID = 'L003';
380
381 • UPDATE LandFill
382 SET Capacity = 'Medium', Location = 'land2';
383 WHERE LandFill_ID = 'L002';
384
385 • DELETE FROM LandFill WHERE LandFill_ID = 'L001';
386
387 • select * from LandFill;
```

The Result Grid shows the following data:

LandFill_ID	Type_ID	Agency_ID	Capacity	Location
L002	T002	A0002	Medium	land2
L004	T003	A0003	Medium	land4
L005	T004	A0002	Large	land5
L003	T003	A0003	Small	land3

The Action Output shows two actions:

#	Time	Action	Message	Duration / Fetch
153	12:18:24	DELETE FROM LandFill WHERE LandFill_ID = 'L001'	0 row(s) affected	0.000 sec
154	12:18:28	select * from LandFill LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

WasteItem Table

The screenshot shows the MySQL Workbench interface with the 'WasteItem' table selected in the schema browser. The SQL editor contains the following code:

```
400
401 • UPDATE Execute the statement under the keyboard cursor
402     SET Weight = '10kg'
403     WHERE Type_ID = 'T004' and LandFill_ID = 'L005';
404
405 • UPDATE WasteItem
406     SET Weight = '60kg'
407     WHERE Type_ID = 'T002' and LandFill_ID = 'L002';
408
409 • DELETE FROM WasteItem WHERE Type_ID = 'T001';
410
411 • select * from WasteItem;
```

The result grid shows the following data:

Type_ID	LandFill_ID	Collection_ID	Weight
T002	L002	C0002	60kg
T004	L005	C0001	10kg
T003	L005	C0001	38kg

The output pane shows the following log entries:

Action	Time	Message	Duration / Fetch
DELETE FROM WasteItem WHERE Type_ID = 'T001'	157 12:20:01	0 row(s) affected	0.000 sec
select * from WasteItem LIMIT 0, 1000	158 12:20:04	3 row(s) returned	0.000 sec / 0.000 sec

ProcessingFacility Table

The screenshot shows the MySQL Workbench interface with the 'ProcessingFacility' table selected in the schema browser. The SQL editor contains the following code:

```
420 ('F006', 'Biogas Plant', 'L5');
421 • UPDATE ProcessingFacility
422     SET Location = 'L7'
423     WHERE Facility_ID = 'F004' and Facility_type = 'Recycling Facility' ;
424
425 • UPDATE ProcessingFacility
426     SET Location = 'L8'
427     WHERE Facility_ID = 'F002';
428
429 • DELETE FROM ProcessingFacility WHERE Facility_ID = 'F005';
430
431 • select * from ProcessingFacility;
```

The result grid shows the following data:

Facility_ID	Facility_Type	Location
F001	Recycling Facility	L1
F002	Composting Facility	L8
F003	Waste-to-Energy Facility	L6
F004	Recycling Center	L3
F005	Biogas Plant	L5
F006	Landfill	NULL

The output pane shows the following log entries:

Action	Time	Message	Duration / Fetch
DELETE FROM ProcessingFacility WHERE Facility_ID = F005	161 12:20:44	1 row(s) affected	0.000 sec
select * from ProcessingFacility LIMIT 0, 1000	162 12:20:48	5 row(s) returned	0.000 sec / 0.000 sec

ProceedWaste Table

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

- Schemas:** miniproject (selected)
- Tables:** bintype, landfill, proceedingfacility, proceedwaste, regulatoryagency, user, usertype, violationrecord, wastebin, wastecollection, wastecollector, wasteitem, wastetype
- SQL Editor:** Contains several SQL statements:

```
440 ('P0006', 'T005', '30%', 'F003', '2024-05-11');
441 • UPDATE ProceedWaste
442 SET Quantity = '10%', Date_Proceed = '2024-02-03'
443 WHERE Proceed_ID = 'P0004';
444
445 • UPDATE ProceedWaste
446 SET Quantity = '20%', Date_Proceed = '2024-04-11'
447 WHERE Proceed_ID = 'P0002';
448
449 • DELETE FROM ProceedWaste WHERE Proceed_ID = 'P0001';
450
451 • select * from ProceedWaste;
452
```
- Result Grid:** Displays the data from the ProceedWaste table.

Proceed_ID	Type_ID	Facility_ID	Quantity	Date_Proceed
P0002	T002	F002	20%	2024-04-11
P0003	T004	F003	40%	2024-05-09
P0004	T003	F001	10%	2024-02-03
P0005	T004	F002	50%	2024-05-08
NULL	NULL	NULL	NULL	NULL
- Output:** Shows the execution history of the queries.

#	Time	Action	Message	Duration / Fetch
165	12:21:41	DELETE FROM ProceedWaste WHERE Proceed_ID = 'P0001'	1row(s) affected	0.000 sec
166	12:21:45	select * from ProceedWaste LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

4 Chapter 4- Transactions

Simple Queries

Select Operation

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard icons for opening files, saving, running queries, and navigating.
- Navigator:** Shows the schema structure under "miniproject" including tables like bintype, landfill, processingfacility, proceedwaste, regulatoryagency, user, usertype, violationrecord, wastebin, wastecollection, wastecollector, wasteitem, and wastetype.
- SQL Editor:** SQL File 8* tab active, containing the following code:

```

448 • DELETE FROM Proceedwaste WHERE Proceed_ID = 'P0001';
449 • select * from Proceedwaste;
450
451 • -- Retrieve the details of all waste collection records including the collection ID, bin ID, collector ID, and collection date.
452
453 • SELECT Collection_ID, Bin_ID, Collector_ID, CollectionDate FROM WasteCollection;
454
455
456
457
458
459
460
461
462
463

```
- Result Grid:** Shows the results of the SELECT query:

Collection_ID	Bin_ID	Collector_ID	CollectorDate
C0001	B0001	C0001	2024-05-06
C0002	B0002	C0002	2024-03-06
C0003	B0003	C0001	2024-03-05
C0004	B0001	C0003	2024-05-01
C0006	B0004	C0003	2024-05-03
...
- Output Tab:** Action Output section shows two rows returned for each query.
- Message Bar:** Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Project Operation

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard icons for opening files, saving, running queries, and navigating.
- Navigator:** Shows the schema structure under "miniproject" including tables like bintype, landfill, processingfacility, proceedwaste, regulatoryagency, user, usertype, violationrecord, wastebin, wastecollection, wastecollector, wasteitem, and wastetype.
- SQL Editor:** SQL File 8* tab active, containing the following code:

```

445 • UPDATE Proceedwaste
446     SET Quantity = '20%', Date_Proceed = '2024-04-11'
447     WHERE Proceed_ID = 'P0002';
448
449 • DELETE FROM Proceedwaste WHERE Proceed_ID = 'P0001';
450
451 • select * from Proceedwaste;
452
453 • -- Retrieve the details of all waste collection records including the collection ID, bin ID, collector ID, and collection date.
454 • SELECT Collection_ID, Bin_ID, Collector_ID, CollectionDate FROM WasteCollection;
455
456 • -- Retrieve the first name, last name, and contact number of all waste collectors.
457 • SELECT First_Name, Last_Name, ContactNo FROM WasteCollector;
458
459
460
461
462
463

```
- Result Grid:** Shows the results of the SELECT query:

First_Name	Last_Name	ContactNo
Kamal	Silva	0417589145
Kamari	Pereira	0912849751
Aparna	Patravara	0745684165
Gayatri	kularatna	074598654
Samudu	Hewawasen	074578532
- Output Tab:** Action Output section shows two rows returned for each query.
- Message Bar:** Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Cartesian Product Operation

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.
- Schemas:** Local instance MySQL80, showing databases database1, feover_23, and miniproject.
- SQL Editor:** FOE*, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*. The query executed is:

```
-- Retrieve the first name, last name, and contact number of all waste collectors;
SELECT First_Name, Last_Name, ContactNo FROM WasteCollector;
```
- Result Grid:** Shows the results of the query. The columns are User_ID, Initials, First_Name, Last_Name, Street, City, Contact_No, User_Status, Landfill_ID, Type_ID, Agency_ID. The data includes rows for US0001 to US0005 and their respective details.
- Output Panel:** Action Output shows two actions:
 - CREATE VIEW User_Details AS SELECT User_ID, First_Name, Last_Name, Street, City FROM User
 - SELECT * FROM User, Landfill LIMIT 0, 1000Message: Error Code: 1050. Table 'User_Details' already exists. Duration / Fetch: 0.000 sec / 0.000 sec / 0.000 sec.

Creating a User View

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.
- Schemas:** Local instance MySQL80, showing databases database1, feover_23, and miniproject.
- SQL Editor:** FOE*, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*. The query executed is:

```
-- Create a view named "User_Details" that contains the user ID, first name, last name, street, and city of all users;
CREATE VIEW User_Details AS
SELECT User_ID, First_Name, Last_Name, Street, City FROM User;
```
- Result Grid:** Shows the results of the query. The columns are User_ID, First_Name, Last_Name, Street, City. The data includes rows for US0001 to US0005 and their respective details.
- Output Panel:** Action Output shows two actions:
 - CREATE VIEW User_Details AS SELECT User_ID, First_Name, Last_Name, Street, City FROM User
 - SELECT * FROM User_DetailsMessage: 0 row(s) affected. Duration / Fetch: 0.000 sec / 0.000 sec / 0.000 sec.

Renaming Operation

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

- Schemas:** Local instance MySQL80, miniproject.
- SQL Editor:** SQL File 8* contains the following code:

```
466 • SELECT * FROM User_Details;
467
468
469 -- Rename the "Location" column in the wasteBin table to "Bin_Location".
470 • select Bin_ID, Location as Bin_Location from wasteBin;
```
- Result Grid:** Shows the results of the query, displaying 5 rows of data with columns Bin_ID and Bin_Location.
- Output Panel:** Action Output shows two log entries for the executed queries.
- Help:** A message states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Aggregation Function

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

- Schemas:** Local instance MySQL80, miniproject.
- SQL Editor:** SQL File 8* contains the following code:

```
470 • select Bin_ID, Location as Bin_Location from wasteBin;
471
472
473 -- Calculate the count of records
474 • SELECT COUNT(*) AS Total_Count FROM wasteItem;
```
- Result Grid:** Shows the result of the COUNT(*) query, displaying 1 row with column Total_Count.
- Output Panel:** Action Output shows multiple log entries for various queries, including the COUNT(*) query and an ALTER TABLE command to rename a column.
- Help:** A message states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Use Of LIKE keyword

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Local instance MySQL80, showing databases1, foour_23, and miniproject. miniproject contains tables: binstype, landfill, proceedingfacility, processwaste, regulatoryagency, user, usertype, violationrecord, wastebin, wastecollection, wastecollector, wasteite, wastetype.
- SQL Editor:** SQL File 8* (active), containing the following code:

```
474 • SELECT COUNT(*) AS Total_Count FROM Wasteitem;
475
476 -- Retrieve the first_name, last_name, and contact number of waste collectors whose last names start with 'k'.
477 • SELECT First_Name, Last_Name, ContactNo FROM WasteCollector WHERE First_Name LIKE '%Sa%';
478
```
- Result Grid:** Shows the result of the query: Apsara Pathirana 0745684165.
- Output Tab:** Action Output, displaying the execution log with entries from 189 to 199, showing various SQL statements and their execution times.

Complex Queries

Union

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

- Schema:** miniproject
- SQL Editor:** SQL File 8*
- Query:**

```
477 • SELECT First_Name, Last_Name, ContactNo FROM WasteCollector WHERE First_Name LIKE '%Se%'  
478  
479  
480  
481 -- Complex Queries  
482  
483 • SELECT Waste_Type FROM WasteType WHERE Type_Id='T006'  
484 UNION  
485 SELECT Bin_Type FROM BinType;
```
- Result Grid:** Shows the results of the UNION query.
- Output:** Action Output table showing the execution details of the query.

Intersection

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

- Schema:** miniproject
- SQL Editor:** SQL File 8*
- Query:**

```
483 • SELECT Waste_Type FROM WasteType WHERE Type_Id='T006'  
484 UNION  
485 SELECT Bin_Type FROM BinType;  
486  
487  
488  
489 □ SELECT L.Type_ID FROM LandFill AS L INTERSECT SELECT W.Type_ID FROM WasteType AS W;
```
- Result Grid:** Shows the results of the INTERSECT query.
- Output:** Action Output table showing the execution details of the query.

Set Difference

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Local instance MySQL80, miniproject.
- SQL Editor:** SQL File 8* (active). The query is:


```

488
489 SELECT L.Type_ID FROM LandFill as L INTERSECT SELECT W.Type_ID FROM WasteType as W;
490
491 -- Finding collectors who have collected from all bins
492 • SELECT Collector_ID
493   FROM WasteCollector
494   WHERE Collector_ID NOT IN (
495     SELECT Collector_ID
496       FROM WasteCollection
497       WHERE Collector_ID IS NOT NULL
498   );
499
500
      
```
- Result Grid:** Shows results for the first part of the query:

Collector_ID
C1004
C1005
...
- Output Window:** Action Output shows three log entries:

Time	Action	Message	Duration / Fetch
7 17:58:20	SELECT Type_ID FROM LandFill INTERSECT SELECT Type_ID FROM WasteType	3 row(s) returned	0.000 sec / 0.000 sec
8 17:59:51	SELECT L.Type_ID FROM LandFill as L INTERSECT SELECT W.Type_ID FROM WasteType as W	3 row(s) returned	0.000 sec / 0.000 sec
9 18:04:25	SELECT Collector_ID FROM WasteCollector WHERE Collector_ID NOT IN (SELECT Collector_ID FRO...	2 row(s) returned	0.000 sec / 0.000 sec

Division

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Local instance MySQL80, miniproject.
- SQL Editor:** SQL File 8* (active). The query is:


```

596 FROM WasteCollector c
597 WHERE c.Collector_ID NOT IN (SELECT Collector_ID FROM WasteCollection);
598
599 -- division
600 • SELECT Facility_ID FROM ProcessingFacility
601 WHERE NOT EXISTS (
602   SELECT * FROM ProceedWaste
603   WHERE ProcessingFacility.Facility_ID = ProceedWaste.Facility_ID
604 );
605
      
```
- Result Grid:** Shows results for the first part of the query:

Facility_ID
F004
F006
...
- Output Window:** Action Output shows four log entries:

Time	Action	Message	Duration / Fetch
7 21:22:59	SELECT c.Collector_ID, c.First_Name, c.Last_Name FROM WasteCollector c WHERE c.Collector_ID NOT IN (SELECT Facility_ID FROM ProcessingFacility)	2 row(s) returned	0.016 sec / 0.000 sec
8 21:35:26	SELECT Collector_ID FROM WasteCollector WHERE Collector_ID NOT IN (SELECT DISTINCT wc.Collector_ID FROM WasteCollector wc WHERE wc.Collector_ID NOT IN (SELECT Facility_ID FROM ProcessingFacility))	0 row(s) returned	0.000 sec / 0.000 sec
9 21:35:48	SELECT c.Collector_ID, c.First_Name, c.Last_Name FROM WasteCollector c WHERE c.Collector_ID NOT IN (SELECT Facility_ID FROM ProcessingFacility)	2 row(s) returned	0.000 sec / 0.000 sec
10 21:38:33	SELECT Facility_ID FROM ProcessingFacility WHERE NOT EXISTS (SELECT * FROM ProceedWaste WHERE ProceedWaste.Facility_ID = ProcessingFacility.Facility_ID)	2 row(s) returned	0.016 sec / 0.000 sec

Inner join using User Views

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

```
494 FROM WasteCollector
495 WHERE Collector_ID NOT IN (
496     SELECT Collector_ID
497     FROM WasteCollection
498     WHERE Collector_ID IS NOT NULL
499 )
500
501 -- inner join
502 -- Joining WasteCollection and WasteCollector to get collector details for each collection
503 • SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wcl.First_Name, wcl.Last_Name
504 FROM WasteCollection wc
505 INNER JOIN WasteCollector wcl ON wc.Collector_ID = wcl.Collector_ID
506
```

The result grid shows the following data:

Collection_ID	Bin_ID	CollectionDate	First_Name	Last_Name
C0001	B0001	2024-05-06	Kamali	Silva
C0002	B0002	2024-03-06	Kamani	Perera
C0003	B0003	2024-03-05	Kamali	Silva
C0004	B0001	2024-05-01	Apsara	Pathirana
C0006	B0004	2024-05-03	Apsara	Pathirana

The output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
8	17:59:51	SELECT L.Type_ID FROM LandFill as L INTERSECT SELECT W.Type_ID FROM WasteType as W	3 row(s) returned	0.000 sec / 0.000 sec
9	18:04:25	SELECT Collector_ID FROM WasteCollector WHERE Collector_ID NOT IN (2 row(s) returned	0.000 sec / 0.000 sec
10	18:06:23	SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wcl.First_Name, wcl.Last_Name FROM WasteCollector wc INNER JOIN WasteCollector wcl ON wc.Collector_ID = wcl.Collector_ID	5 row(s) returned	0.000 sec / 0.000 sec

Natural Join using User Views

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

```
507
508 -- natural join
509 • SELECT *
510 NATURAL JOIN WasteCollector as WC where collector_ID='C1002';
511
512
513
514
515
516
517
518
```

The result grid shows the following data:

Bin_ID	User_ID	Location	Capacity	Current_FillLevel	Collector_ID	First_Name	Last_Name	ContactNo
B0001	U50001	L1	Medium	80%	C1002	Kamani	Perera	0912849751
B0002	U50002	L2	Small	50%	C1002	Kamani	Perera	0912849751
B0003	U50003	L1	Large	50%	C1002	Kamani	Perera	0912849751
B0004	U50001	L3	Medium	60%	C1002	Kamani	Perera	0912849751
B0006	U50002	L1	Medium	30	C1002	Kamani	Perera	0912849751

The output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
10	18:06:23	SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wcl.First_Name, wcl.Last_Name FROM WasteCollector wc INNER JOIN WasteCollector wcl ON wc.Collector_ID = wcl.Collector_ID	5 row(s) returned	0.000 sec / 0.000 sec
11	18:10:25	SELECT * FROM WasteBin NATURAL JOIN WasteCollector LIMIT 0, 1000	25 row(s) returned	0.000 sec / 0.000 sec
12	18:13:42	SELECT * FROM WasteBin as WB NATURAL JOIN WasteCollector as WC LIMIT 0, 1000	25 row(s) returned	0.000 sec / 0.000 sec
13	18:17:05	SELECT * FROM WasteBin as WB NATURAL JOIN WasteCollector as WC where collector_ID='C1002' LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Left Outer Join

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

- Schemas:** miniproject
- SQL Editor:** SQL File 8*
- SQL:**

```
517
518 • CREATE VIEW UserView AS
519   SELECT User_ID, Contact_No
520   FROM User
521 • CREATE VIEW UserTypeView AS
522   SELECT User_ID, User_Type
523   FROM UserType
524 •   SELECT u.User_ID, u.Contact_No, ut.User_Type
525   FROM UserView u
526   LEFT OUTER JOIN UserTypeView ut ON u.User_ID = ut.User_ID;
527
528
```
- Result Grid:** Shows the results of the query, displaying columns: User_ID, Contact_No, User_Type. The data includes:

User_ID	Contact_No	User_Type
US0001	0776758456	Commercial Institute
US0001	0776758456	Domestic
US0002	070896567	Medical Center
US0003	0708867564	■■■■■
US0004	078567456	■■■■■
US0005	0723456734	Government Institutes
- Action Output:**

#	Time	Action	Message	Duration / Fetch
1	16 18:33:11	SELECT u.User_ID, u.Contact_No, ut.User_Type FROM User AS u LEFT OUTER JOIN UserTypeView AS ut ON u.User_ID = ut.User_ID;	6 row(s) returned	0.000 sec / 0.000 sec
2	17 18:34:42	CREATE VIEW UserView AS SELECT User_ID, Contact_No FROM User	0 row(s) affected	0.016 sec
3	18 18:34:46	CREATE VIEW UserTypeView AS SELECT User_ID, User_Type FROM UserType	0 row(s) affected	0.015 sec
4	19 18:35:03	SELECT u.User_ID, u.Contact_No, ut.User_Type FROM UserView u LEFT OUTER JOIN UserTypeView ut ON u.User_ID = ut.User_ID;	6 row(s) returned	0.000 sec / 0.000 sec

Right Outer Join

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

- Schemas:** miniproject
- SQL Editor:** SQL File 8*
- SQL:**

```
530 -- Right Outer Join
531 • CREATE VIEW UserView AS
532   SELECT User_ID, Contact_No
533   FROM User
534 • CREATE VIEW UserTypeView AS
535   SELECT User_ID, User_Type
536   FROM UserType
537 •   SELECT u.User_ID, u.Contact_No, ut.User_Type
538   FROM UserView u
539   RIGHT OUTER JOIN UserTypeView ut ON ut.User_ID = u.User_ID;
540
541
```
- Result Grid:** Shows the results of the query, displaying columns: User_ID, Contact_No, User_Type. The data includes:

User_ID	Contact_No	User_Type
US0001	0776758456	Commercial Institute
US0001	0776758456	Domestic
US0002	070896567	Medical Center
US0005	0723456734	Government Institutes
- Action Output:**

#	Time	Action	Message	Duration / Fetch
1	20 18:37:35	CREATE VIEW UserView AS SELECT User_ID, Contact_No FROM User	Error Code: 1050. Table 'UserView' already exists	0.000 sec
2	21 18:37:43	SELECT u.User_ID, u.Contact_No, ut.User_Type FROM UserTypeView AS ut RIGHT OUTER JOIN UserView u ON ut.User_ID = u.User_ID;	6 row(s) returned	0.000 sec / 0.000 sec
3	22 18:38:12	SELECT u.User_ID, u.Contact_No, ut.User_Type FROM UserView AS u RIGHT OUTER JOIN UserTypeView AS ut ON u.User_ID = ut.User_ID;	Error Code: 1054. Unknown column 'u.Contact_No' in field list	0.000 sec
4	23 18:39:47	SELECT u.User_ID, u.Contact_No, ut.User_Type FROM UserView u RIGHT OUTER JOIN UserTypeView AS ut ON u.User_ID = ut.User_ID;	4 row(s) returned	0.000 sec / 0.000 sec

Full Outer Join

The screenshot shows the MySQL Workbench interface with a query editor window. The schema is set to 'miniproject'. The query is:

```
541
542 •  SELECT u.User_ID, u.Contact_No, ut.User_Type
543   FROM UserTypeView AS ut
544   LEFT OUTER JOIN UserView AS u ON ut.User_ID = u.User_ID
545
546 UNION
547
548 SELECT u.User_ID, u.Contact_No, ut.User_Type
549   FROM UserTypeView AS ut
550   RIGHT OUTER JOIN UserView AS u ON ut.User_ID = u.User_ID;
```

The result grid shows the following data:

User_ID	Contact_No	User_Type
US0001	077675456	Commercial Institute
US0001	077675456	Domestic
US0002	0708896567	Medical Center
US0005	0723456734	Government Institutes
US0003	0708867564	Industrial
US0004	0785673456	Residential

The output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	18:43:18	SELECT u.User_ID, u.Contact_No, ut.User_Type FROM UserTypeView AS ut LEFT OUTER JOIN UserView AS u ON ut.User_ID = u.User_ID;	6 row(s) returned	0.000 sec / 0.000 sec

Outer Union Relational Algebraic Operation

The screenshot shows the MySQL Workbench interface with a query editor window. The schema is set to 'miniproject'. The query is:

```
584
585
586 •  (SELECT b.Bin_ID, b.Location, wc.Collection_ID, wc.CollectionDate
587   FROM WasteBin b
588   LEFT JOIN WasteCollection wc ON b.Bin_ID = wc.Bin_ID)
589 UNION ALL
590 (SELECT b.Bin_ID, b.Location, NULL AS Collection_ID, NULL AS CollectionDate
591   FROM WasteBin b
592 WHERE b.Bin_ID NOT IN (SELECT Bin_ID FROM WasteCollection));
```

The result grid shows the following data:

Bin_ID	Location	Collection_ID	CollectionDate
B0001	L1	C0001	2024-05-06
B0001	L1	C0004	2024-05-01
B0002	L2	C0002	2024-03-06
B0003	L1	C0003	2024-03-05
B0004	L3	C0006	2024-05-03
B0006	L1	NULL	NULL
B0006	L1	NULL	NULL

The output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
3	21:00:17	SELECT Type_ID, Weight FROM WasteItem	UNION SELECT Type_ID, Quantity AS Weight FROM Proc...	7row(s) returned 0.000 sec / 0.000 sec
4	21:04:04	SELECT Type_ID, SourceTable, Quantity FROM (SELECT Type_ID, 'WasteItem' AS SourceTable, Weight...	7row(s) returned 0.006 sec / 0.000 sec
5	21:07:50	SELECT Type_ID, Landfill_ID, Collection_ID, Weight, WasteItem AS SourceTable FROM WasteItem UNION...	7row(s) returned 0.016 sec / 0.000 sec	
6	21:19:00	(SELECT b.Bin_ID, b.Location, wc.Collection_ID, wc.CollectionDate FROM WasteBin b LEFT JOIN WasteC...	7row(s) returned 0.000 sec / 0.000 sec	

Nested Query with Union

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

```
568
569 •   SELECT Type_ID, SourceTable, Quantity
570   FROM (
571     SELECT Type_ID, 'WasteItem' AS SourceTable, Weight AS Quantity
572       FROM WasteItem
573     UNION
574     SELECT Type_ID, 'ProceedWaste' AS SourceTable, Quantity AS Quantity
575       FROM ProceedWaste
576   ) AS CombinedData;
577
```

The result grid shows the following data:

Type_ID	SourceTable	Quantity
T002	WasteItem	60kg
T004	WasteItem	10kg
T003	WasteItem	38kg
T002	ProceedWaste	20%
T004	ProceedWaste	40%
T003	ProceedWaste	10%
T004	ProceedWaste	50%

The output pane shows the following log entries:

Action	Time	Action	Message	Duration / Fetch
1	20:57:54	SELECT Type_ID, Weight FROM (SELECT Type_ID, Weight FROM WasteItem UNION SELECT Type_ID, Weight FROM ProceedWaste) AS CombinedData;	7 row(s) returned	0.000 sec / 0.000 sec
2	20:59:52	SELECT Type_ID, Weight, Quantity FROM (SELECT Type_ID, Weight FROM WasteItem UNION SELECT Type_ID, Weight AS Weight FROM ProceedWaste) AS CombinedData;	Error Code: 1054. Unknown column 'Quantity' in field list'	0.000 sec
3	21:00:17	SELECT Type_ID, Weight FROM WasteItem UNION SELECT Type_ID, Quantity AS Weight FROM ProceedWaste	7 row(s) returned	0.000 sec / 0.000 sec
4	21:04:04	SELECT Type_ID, SourceTable, Quantity FROM (SELECT Type_ID, 'WasteItem' AS SourceTable, Weight AS Quantity FROM WasteItem UNION SELECT Type_ID, 'ProceedWaste' AS SourceTable, Quantity AS Quantity FROM ProceedWaste) AS CombinedData;	7 row(s) returned	0.000 sec / 0.000 sec

Nested Query With Join

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

```
556 -- Using a nested query with join to retrieve waste collection details along with collector
557 •   SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wcl.First_Name, wcl.Last_Name
558   FROM (
559     SELECT *
560       FROM WasteCollection
561      WHERE CollectionDate = '2024-05-06'
562   ) AS wc
563   JOIN WasteCollector wcl ON wc.Collector_ID = wcl.Collector_ID;
```

The result grid shows the following data:

Collection_ID	Bin_ID	CollectionDate	First_Name	Last_Name
C0001	80001	2024-05-06	Kamal	Silva

The output pane shows the following log entries:

Action	Time	Action	Message	Duration / Fetch
6	18:50:04	SELECT WC.Collection_ID, WC.Bin_ID, WC.CollectionDate, WC.First_Name, WC.Last_Name FROM WasteCollection AS WC	Error Code: 1054. Unknown column 'WC' in field list'	0.000 sec
7	18:52:34	SELECT WC.Collection_ID, WC.Bin_ID, WC.CollectionDate, WC.First_Name, WC.Last_Name FROM WasteCollection AS WC	Error Code: 1054. Unknown column 'WC.First_Name' in field list'	0.000 sec
8	18:55:51	SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wcl.First_Name, wcl.Last_Name FROM (SELECT * FROM WasteCollection) AS wc JOIN WasteCollector wcl ON wc.Collector_ID = wcl.Collector_ID	0 row(s) returned	0.015 sec / 0.000 sec
9	18:56:02	SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wcl.First_Name, wcl.Last_Name FROM (SELECT * FROM WasteCollection) AS wc JOIN WasteCollector wcl ON wc.Collector_ID = wcl.Collector_ID	1 row(s) returned	0.000 sec / 0.000 sec

Nested Query with Projection

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbars:** FDE*, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*.
- Navigator:** Shows the schema `miniproject` containing tables like `binstype`, `landfill`, `procedwaste`, `processingfacility`, `regulatoryagency`, `user`, `userstype`, `violationrecord`, `wastebin`, `wastecollection`, `wastecollector`, `wasteitem`, and `wastetype`.
- SQL Editor:** Displays the following SQL code:

```
-- Retrieve only the first names of users who have bins.  
SELECT First_Name FROM User WHERE User_ID IN  
(SELECT DISTINCT User_ID FROM WasteBin);
```
- Result Grid:** Shows the results of the query:

First_Name
Namal
Amali
Kumudu
- Action Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
7	18:52:34	SELECT WC.Collection_ID, WC.CollectionDate, WC.First_Name, WC.Last_Name FROM WasteCollection AS...	Error Code: 1054. Unknown column 'WC.First_Name' in field list'	0.000 sec
8	18:55:51	SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wc.First_Name, wc.Last_Name FROM (SELECT...)	0 row(s) returned	0.015 sec / 0.000 sec
9	18:56:02	SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wc.First_Name, wc.Last_Name FROM (SELECT...)	1 row(s) returned	0.000 sec / 0.000 sec
10	18:59:59	SELECT First_Name FROM User WHERE User_ID IN (SELECT DISTINCT User_ID FROM WasteBin) LIMIT ...	3 row(s) returned	0.016 sec / 0.000 sec

5 Chapter 5- Data Base Tuning

Union

Before tuning

```

MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas Administration Information
Schema: miniproject
FOE* SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 8*
686 -- union
687 -- before tuning
688 EXPLAIN SELECT Waste_Type FROM WasteType WHERE Type_Id='T006'
689 UNION
690 SELECT Bin_Type FROM BinType;
691
692 -- create index
693 CREATE INDEX idx_Type_Id ON WasteType (Type_Id);
694 CREATE INDEX idx_Bin_Type ON BinType (Bin_Type);
695
696 -- after tuning
697 EXPLAIN SELECT Waste_Type
698 FROM WasteType
699 WHERE Type_Id = 'T006';
700 UNION
701 SELECT Bin_Type
702 FROM BinType;
703

```

Result Grid

ID	Select_Type	Table	Partitions	Type	Possible_Keys	Key	Key_Len	Ref	Rows	Filtered	Extra
1	PRIMARY	WasteType		const	PRIMARY, idx_Type_Id	PRIMARY	42	const	1	100.00	Using index
2	UNION	BinType		index		idx_Bin_Type	5	const	202	100.00	Using temporary
3	UNION RESULT	<union1,2>		ALL							Using temporary

Result 9

#	Time	Action	Message	Duration / Fetch
1	14:59:39	EXPLAIN SELECT Waste_Type FROM WasteType WHERE Type_Id = 'T006' UNION SELECT Bin_Type FR...	3 row(s) returned	0.000 sec / 0.000 sec
2	15:28:52	explain SELECT Collector_ID FROM WasteCollector WHERE Collector_ID NOT IN (...)	2 row(s) returned	0.016 sec / 0.000 sec
3	15:31:13	explain SELECT wc.Collector_ID FROM WasteCollector wc LEFT JOIN WasteCollection wl ON wc.Collector_...	2 row(s) returned	0.000 sec / 0.000 sec
4	15:32:33	EXPLAIN SELECT Waste_Type FROM WasteType where Type_Id='T006' UNION SELECT Bin_Type FROM ...	3 row(s) returned	0.000 sec / 0.000 sec

Create index WasteType

```

MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas Administration Information
Schema: miniproject
FOE* SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 8*
686 -- union
687 -- before tuning
688 EXPLAIN SELECT Waste_Type FROM WasteType WHERE Type_Id='T006'
689 UNION
690 SELECT Bin_Type FROM BinType;
691
692 -- create index
693 CREATE INDEX idx_Type_Id ON WasteType (Type_Id);
694 CREATE INDEX idx_Bin_Type ON BinType (Bin_Type);
695
696 -- show indexes from wastetype;
697
698 -- after tuning
699 EXPLAIN SELECT Waste_Type
700 FROM WasteType
701 WHERE Type_Id = 'T006';
702 UNION
703 SELECT Bin_Type
704 FROM BinType;
705

```

Result Grid

Table	Non_unique	Key_name	Seq_in_Index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
wastetype	0	PRIMARY	1	Type_ID	A	5				BTREE		YES		
wastetype	1	idx_Type_Id	1	Type_ID	A	5				BTREE		YES		

Result 17

#	Time	Action	Message	Duration / Fetch
19	01:24:01	EXPLAIN SELECT Waste_Type FROM WasteType WHERE Type_Id='T006' UNION SELECT Bin_Type FROM ...	3 row(s) returned	0.016 sec / 0.000 sec
20	01:24:09	CREATE INDEX idx_Type_Id ON WasteType (Type_Id)	0 rows affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec
21	01:24:13	CREATE INDEX idx_Bin_Type ON BinType (Bin_Type)	0 rows affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
22	01:24:17	show indexes from wastetype	2 row(s) returned	0.000 sec / 0.000 sec

Create index in BinType

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbars:** Standard, Navigator, SQL, Database, Server, Tools, Scripting, Help.
- SQL Editor:** F0-E*, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*. The query window contains the following code:


```

687 -- union
688 -- before tuning
689 EXPLAIN SELECT Waste_Type FROM WasteType WHERE Type_Id='T006'
690 UNION
691
692 SELECT Bin_Type FROM BinType;
693
694 -- create index
695 CREATE INDEX idx_Type_Id ON WasteType (Type_Id);
696 CREATE INDEX idx_Bin_Type ON BinType (Bin_Type);
697
698 -- show indexes from wastetype;
699 SHOW INDEXES FROM wastetype;
700
701 -- after tuning
    
```
- Result Grid:** Shows the results of the EXPLAIN command for the first query. It lists three rows for the BinType table:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Visible	Expression
bintype	0	PRIMARY	1	Bin_Id	A	3				BTREE		YES	
bintype	0	PRIMARY	2	Bin_Type	A	5				BTREE		YES	
bintype	1	idx_Bin_Type	1	Bin_Type	A	4				BTREE		YES	
- Action Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
20	01:24:09	CREATE INDEX idx_Type_Id ON WasteType (Type_Id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec
21	01:24:13	CREATE INDEX idx_Bin_Type ON BinType (Bin_Type)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
22	01:24:17	show indexes from wastetype	2 row(s) returned	0.000 sec / 0.000 sec
23	01:33:07	show indexes from BinType	3 row(s) returned	0.015 sec / 0.000 sec

After Tuning

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbars:** Standard, Navigator, SQL, Database, Server, Tools, Scripting, Help.
- SQL Editor:** F0-E*, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*. The query window contains the following code:


```

688 EXPLAIN SELECT Waste_Type FROM WasteType WHERE Type_Id='T006'
689 UNION
690
691 SELECT Bin_Type FROM BinType;
692
693
694 -- create index
695 CREATE INDEX idx_Type_Id ON WasteType (Type_Id);
696 CREATE INDEX idx_Bin_Type ON BinType (Bin_Type);
697
698 -- after tuning
699 EXPLAIN SELECT Waste_Type
700 FROM WasteType
701 WHERE Type_Id = 'T006';
702 UNION
703
704 SELECT Bin_Type
705 FROM BinType;
706
707
708 -- set difference
    
```
- Result Grid:** Shows the results of the EXPLAIN command for the second query. It lists three rows:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	WasteType		const	PRIMARY, idx_Type_Id	PRIMARY	42	const	1	100.00	
2	UNION	BinType		index		idx_Bin_Type	202	const	5	100.00	Using index
3	UNION RESULT	<union1,2>		ALL			505	const	505	100.00	Using temporary
- Action Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
5	15:32:33	EXPLAIN SELECT Waste_Type FROM WasteType WHERE Type_Id='T006' UNION SELECT Bin_Type FROM ...	3 row(s) returned	0.000 sec / 0.000 sec
6	15:33:44	CREATE INDEX idx_Type_Id ON WasteType (Type_Id)	Error Code: 1061. Duplicate key name 'idx_Type_Id'	0.016 sec
7	15:33:50	CREATE INDEX idx_Bin_Type ON BinType (Bin_Type)	Error Code: 1061. Duplicate key name 'idx_Bin_Type'	0.000 sec
8	15:34:04	EXPLAIN SELECT Waste_Type FROM WasteType WHERE Type_Id = 'T006' UNION SELECT Bin_Type FR...	3 row(s) returned	0.000 sec / 0.000 sec

Set Difference

Before Tuning

```

SELECT u.User_ID, u.Contact_No, ut.user_Type
FROM userview AS u
LEFT OUTER JOIN userTypeview AS ut ON u.User_ID = ut.User_ID;
EXPLAIN SELECT c.Collector_ID, c.First_Name, c.Last_Name
FROM WasteCollector c
WHERE c.Collector_ID NOT IN (SELECT Collector_ID FROM WasteCollection);

```

ID	Select_Type	Table	Partitions	Type	Possible_Keys	Key	Key_Len	Ref	Rows	Filtered	Extra
1	PRIMARY	c	ALL	ALL					5	100.00	Using where
2	SUBQUERY	WasteCollection	ALL	Index	idx_Collector_ID_wc	idx_Collector_ID_wc	43		5	100.00	Using index

Action Output:

#	Time	Action	Message	Duration / Fetch
26	01:48:48	Explain SELECT Collector_ID FROM WasteCollector WHERE Collector_ID NOT IN (SELECT Collector_ID ... 2 row(s) returned	0.015 sec / 0.000 sec
27	01:51:34	EXPLAIN SELECT wc.Collector_ID FROM WasteCollector wc LEFT JOIN WasteCollection wd ON wc.Collector_ID = wd.Collector_ID	2 row(s) returned	0.000 sec / 0.000 sec
28	01:57:08	SELECT c.Collector_ID, c.First_Name, c.Last_Name FROM WasteCollector c WHERE c.Collector_ID NOT IN (2 row(s) returned	0.000 sec / 0.000 sec
29	01:57:29	EXPLAIN SELECT c.Collector_ID, c.First_Name, c.Last_Name FROM WasteCollector c WHERE c.Collector_ID NOT IN (2 row(s) returned	0.000 sec / 0.000 sec

After tuning

```

SELECT c.Collector_ID NOT IN (SELECT collector_ID FROM WasteCollection);
-- after tuning
EXPLAIN SELECT c.Collector_ID, c.First_Name, c.Last_Name
FROM WasteCollector c
LEFT JOIN WasteCollection wc ON c.Collector_ID = wc.Collector_ID
WHERE wc.Collector_ID IS NULL;

```

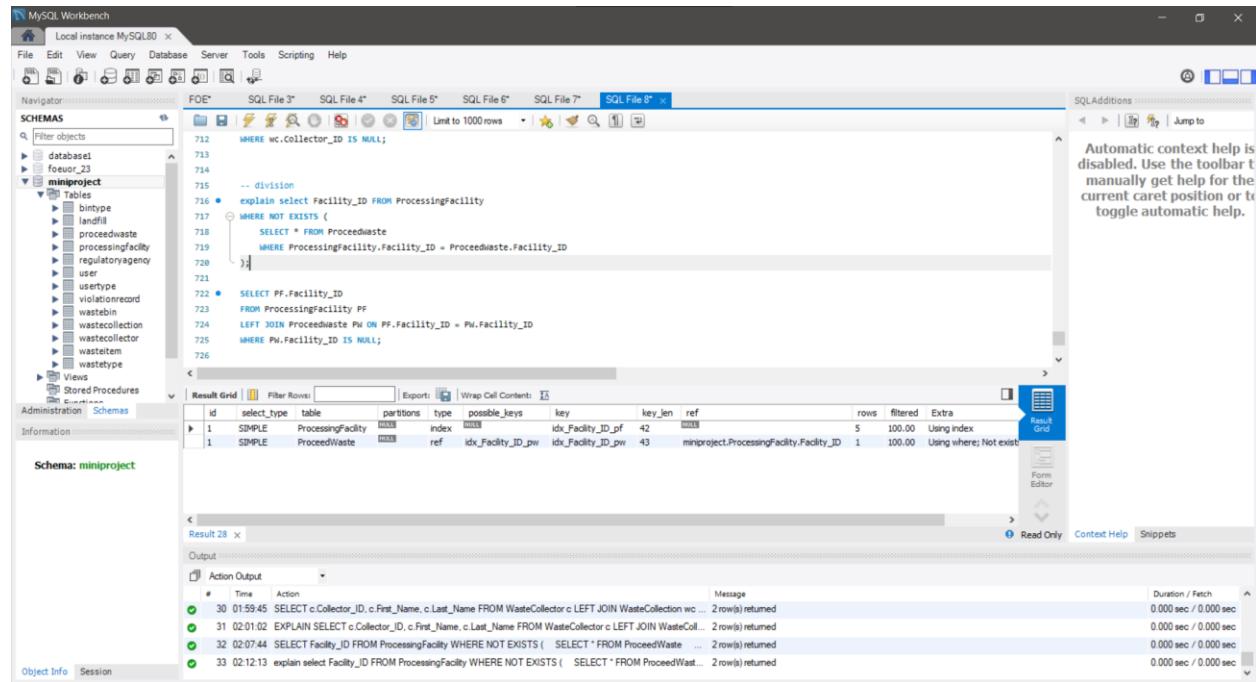
ID	Select_Type	Table	Partitions	Type	Possible_Keys	Key	Key_Len	Ref	Rows	Filtered	Extra
1	SIMPLE	c	ALL	ALL					5	100.00	Using where; Using index
1	SIMPLE	wc	ALL	ref	idx_Collector_ID_wc	idx_Collector_ID_wc	43	miniproject.c.Collector_ID	1	100.00	Using where; Using index

Action Output:

#	Time	Action	Message	Duration / Fetch
28	01:57:08	SELECT c.Collector_ID, c.First_Name, c.Last_Name FROM WasteCollector c WHERE c.Collector_ID NOT IN (2 row(s) returned	0.000 sec / 0.000 sec
29	01:57:29	EXPLAIN SELECT c.Collector_ID, c.First_Name, c.Last_Name FROM WasteCollector c WHERE c.Collector_ID NOT IN (2 row(s) returned	0.000 sec / 0.000 sec
30	01:59:45	SELECT c.Collector_ID, c.First_Name, c.Last_Name FROM WasteCollector c LEFT JOIN WasteCollection wc ...	2 row(s) returned	0.000 sec / 0.000 sec
31	02:01:02	EXPLAIN SELECT c.Collector_ID, c.First_Name, c.Last_Name FROM WasteCollector c LEFT JOIN WasteCollection wc ...	2 row(s) returned	0.000 sec / 0.000 sec

Division

Before Tunning



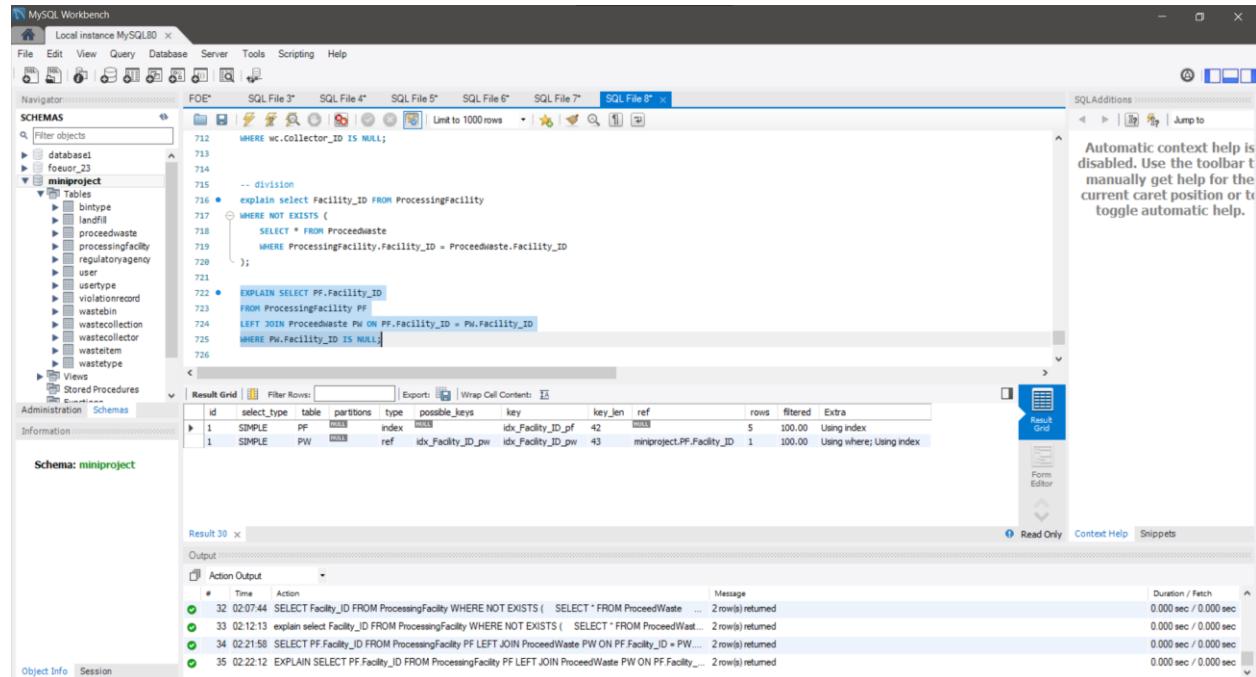
```
1 WHERE wc.Collector_ID IS NULL;
2
3 -- division
4 16 explain select Facility_ID FROM ProcessingFacility
5 WHERE NOT EXISTS (
6     SELECT * FROM Proceedwaste
7     WHERE ProcessingFacility.Facility_ID = Proceedwaste.Facility_ID
8 );
9
10
11 22 SELECT PF.Facility_ID
12 FROM ProcessingFacility PF
13 LEFT JOIN Proceedwaste PW ON PF.Facility_ID = PW.Facility_ID
14 WHERE PW.Facility_ID IS NULL;
15
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	ProcessingFacility		index	idx_Facility_ID_pf	42	NULL		5	100.00	Using index
1	SIMPLE	Proceedwaste		ref	idx_Facility_ID_pw	43	miniproject.ProcessingFacility.Facility_ID	1	100.00	Using where; Not exists	

Action Output

#	Time	Action	Message	Duration / Fetch
30	01:59:45	SELECT c.Collector_ID, c.First_Name, c.Last_Name FROM WasteCollector c LEFT JOIN WasteCollection wc ...	2 row(s) returned	0.000 sec / 0.000 sec
31	02:01:02	EXPLAIN SELECT c.Collector_ID, c.First_Name, c.Last_Name FROM WasteCollector c LEFT JOIN WasteCollection ...	2 row(s) returned	0.000 sec / 0.000 sec
32	02:07:44	SELECT Facility_ID FROM ProcessingFacility WHERE NOT EXISTS (2 row(s) returned	0.000 sec / 0.000 sec
33	02:12:13	explain select Facility_ID FROM ProcessingFacility WHERE NOT EXISTS (2 row(s) returned	0.000 sec / 0.000 sec

After Tunning



```
1 WHERE wc.Collector_ID IS NULL;
2
3 -- division
4 16 explain select Facility_ID FROM ProcessingFacility
5 WHERE NOT EXISTS (
6     SELECT * FROM Proceedwaste
7     WHERE ProcessingFacility.Facility_ID = Proceedwaste.Facility_ID
8 );
9
10
11 22 EXPLAIN SELECT PF.Facility_ID
12 FROM ProcessingFacility PF
13 LEFT JOIN Proceedwaste PW ON PF.Facility_ID = PW.Facility_ID
14 WHERE PW.Facility_ID IS NULL;
15
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	PF		index	idx_Facility_ID_pf	42	NULL		1	100.00	Using index
1	SIMPLE	PW		ref	idx_Facility_ID_pw	43	miniproject.PF.Facility_ID	1	100.00	Using where; Using index	

Action Output

#	Time	Action	Message	Duration / Fetch
32	02:07:44	SELECT Facility_ID FROM ProcessingFacility WHERE NOT EXISTS (2 row(s) returned	0.000 sec / 0.000 sec
33	02:12:13	explain select Facility_ID FROM ProcessingFacility WHERE NOT EXISTS (2 row(s) returned	0.000 sec / 0.000 sec
34	02:21:58	SELECT PF.Facility_ID FROM ProcessingFacility PF LEFT JOIN Proceedwaste PW ON PF.Facility_ID = PW...	2 row(s) returned	0.000 sec / 0.000 sec
35	02:22:12	EXPLAIN SELECT PF.Facility_ID FROM ProcessingFacility PF LEFT JOIN Proceedwaste PW ON PF.Facility...	2 row(s) returned	0.000 sec / 0.000 sec

Inner Join

Before Tuning

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Local instance MySQL80, miniproject.
- Tables:** Shows tables like bintype, landfill, proceedwaste, processingfacility, regulatoryagency, user, usertype, violationrecord, wastebin, wastecollection, wastecollector, wasteitem, wastetype.
- SQL Editor:** Contains the following SQL code:


```

727 -- Execute the selected portion of the script or everything, if there is no selection
728 -- before tuning
729 explain select wc.collection_ID, wc.Bin_ID, wc.CollectionDate, wcl.First_Name, wcl.Last_Name
730   FROM WasteCollection wc
731     INNER JOIN WasteCollector wcl ON wc.collector_ID = wcl.collector_ID;
732
733 -- Create indexes
734 CREATE INDEX idx_WasteCollection_Collector_ID ON WasteCollection (Collector_ID);
735 CREATE INDEX idx_WasteCollector_Collector_ID ON WasteCollector (Collector_ID);
736
737 -- after tuning
738 Explain SELECT wc.collection_ID, wc.Bin_ID, wc.CollectionDate, wcl.First_Name, wcl.Last_Name
739   FROM WasteCollection wc
740     INNER JOIN WasteCollector wcl ON wc.collector_ID = wcl.collector_ID;
    
```
- Result Grid:** Shows the execution plan for the EXPLAIN command, indicating a PRIMARY key for the wastecollection table and a PRIMARY key for the wastecollector table.
- Action Output:** Shows the history of actions taken, including the creation of indexes and the execution of the EXPLAIN command.

Create index in WasteCollection

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Local instance MySQL80, miniproject.
- Tables:** Shows tables like bintype, landfill, proceedwaste, processingfacility, regulatoryagency, user, usertype, violationrecord, wastebin, wastecollection, wastecollector, wasteitem, wastetype.
- SQL Editor:** Contains the following SQL code:


```

729 -- before tuning
730 explain select wc.collection_ID, wc.Bin_ID, wc.CollectionDate, wcl.First_Name, wcl.Last_Name
731   FROM WasteCollection wc
732     INNER JOIN WasteCollector wcl ON wc.collector_ID = wcl.collector_ID;
733
734 -- Create indexes
735 CREATE INDEX idx_WasteCollection_Collector_ID ON WasteCollection (Collector_ID);
736 CREATE INDEX idx_WasteCollector_Collector_ID ON WasteCollector (Collector_ID);
737
738 -- show indexes
739 show indexes from wastecollection;
740 show indexes from wastecollector;
741
742 -- after tuning
    
```
- Result Grid:** Shows the results of the SHOW INDEXES command for the wastecollection table, listing four indexes: PRIMARY, FK_Bin_ID, idx_Parent_Collector_ID, and idx_Collector_ID_wc.
- Action Output:** Shows the history of actions taken, including the creation of indexes and the execution of the SHOW INDEXES command.

Create index in WasteCollector

```

-- Create indexes
CREATE INDEX idx_WasteCollection_Collector_ID ON WasteCollection (Collector_ID);
CREATE INDEX idx_WasteCollector_Collector_ID ON WasteCollector (Collector_ID);

-- show indexes
show indexes from wastecollection;
show indexes from wastecollector;

-- after tuning
Explain SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wc1.First_Name, wc1.Last_Name
FROM wastecollection wc
INNER JOIN WasteCollector wc1 ON wc.Collector_ID = wc1.Collector_ID;
    
```

Result Grid:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Express
wastecollector	0	PRIMARY	1	Collector_ID	A	5				BTREE			YES	
wastecollector	1	idx_Collector_ID_wd	1	Collector_ID	A	5				BTREE			YES	

Action Output:

#	Time	Action	Message	Duration / Fetch
45	02:39:19	Drop INDEX idx_WasteCollection_Collector_ID ON WasteCollection	0 row(s) affected; 0 Duplicates: 0 Warnings: 0	0.032 sec
46	02:39:44	CREATE INDEX idx_WasteCollector_Collector_ID ON WasteCollector (Collector_ID)	0 row(s) affected, 1 warning(s): 1831 Duplicate index idx_WasteCollection_Collector_ID defined on the table 'miniproj...'.	0.047 sec
47	02:40:48	show indexes from wastecollection	5 row(s) returned	0.015 sec / 0.000 sec
48	02:42:38	show indexes from wastecollector	2 row(s) returned	0.000 sec / 0.000 sec

After Tuning

```

-- Create indexes
CREATE INDEX idx_WasteCollection_Collector_ID ON WasteCollection (Collector_ID);
CREATE INDEX idx_WasteCollector_Collector_ID ON WasteCollector (Collector_ID);

-- show indexes
show indexes from wastecollection;
show indexes from wastecollector;

-- after tuning
Explain SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wc1.First_Name, wc1.Last_Name
FROM wastecollection wc
INNER JOIN WasteCollector wc1 ON wc.Collector_ID = wc1.Collector_ID;
    
```

Result Grid:

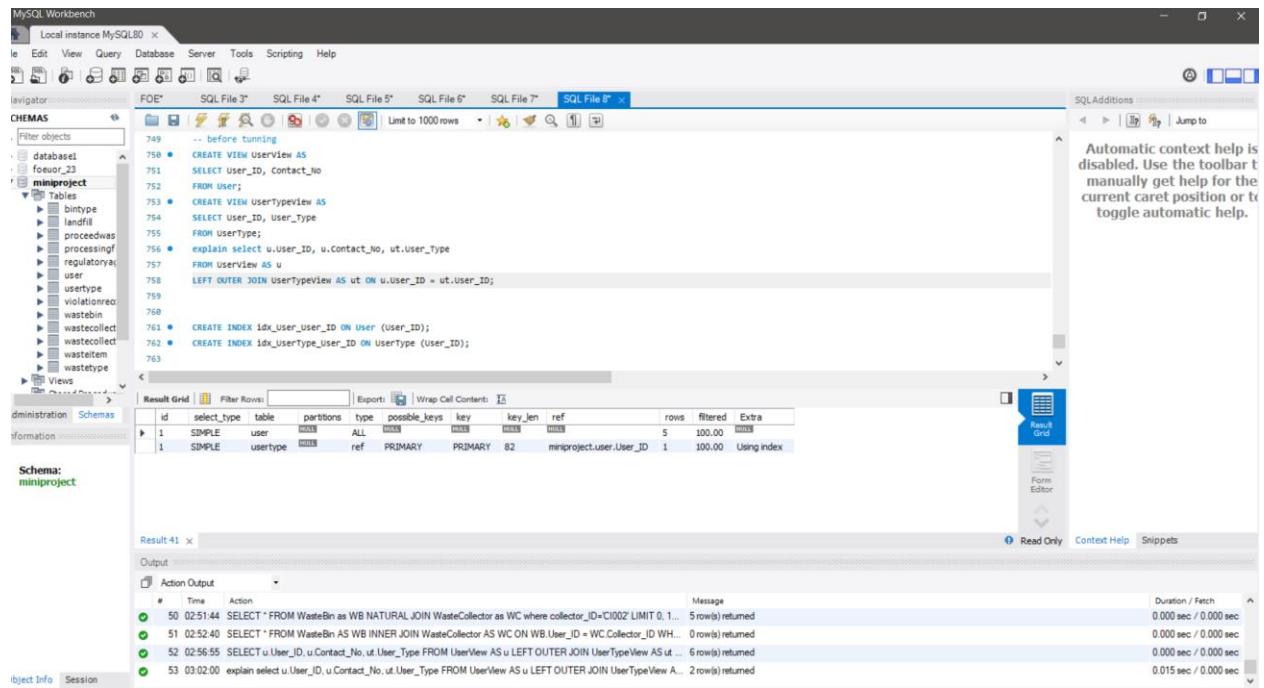
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	wc		ALL	idx_Collector_ID_wc, idx_WasteCollection_Colle...	idx_Collector_ID_wc	42	miniproj.wc.Collector_ID	5	100.00	Using where
1	SIMPLE	wd		eq_ref	PRIMARY, idx_Collector_ID_wd	PRIMARY	42	miniproj.wc.Collector_ID	1	100.00	

Action Output:

#	Time	Action	Message	Duration / Fetch
46	02:39:44	CREATE INDEX idx_WasteCollection_Collector_ID ON WasteCollection (Collector_ID)	0 row(s) affected, 1 warning(s): 1831 Duplicate index idx_WasteCollection_Collector_ID defined on the table 'miniproj...'.	0.047 sec
47	02:40:48	show indexes from wastecollection	5 row(s) returned	0.015 sec / 0.000 sec
48	02:42:38	show indexes from wastecollector	2 row(s) returned	0.000 sec / 0.000 sec
49	02:44:54	Explain SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wc1.First_Name, wc1.Last_Name FROM WasteC...	2 row(s) returned	0.000 sec / 0.000 sec

Left Outer Join

Before Tunning



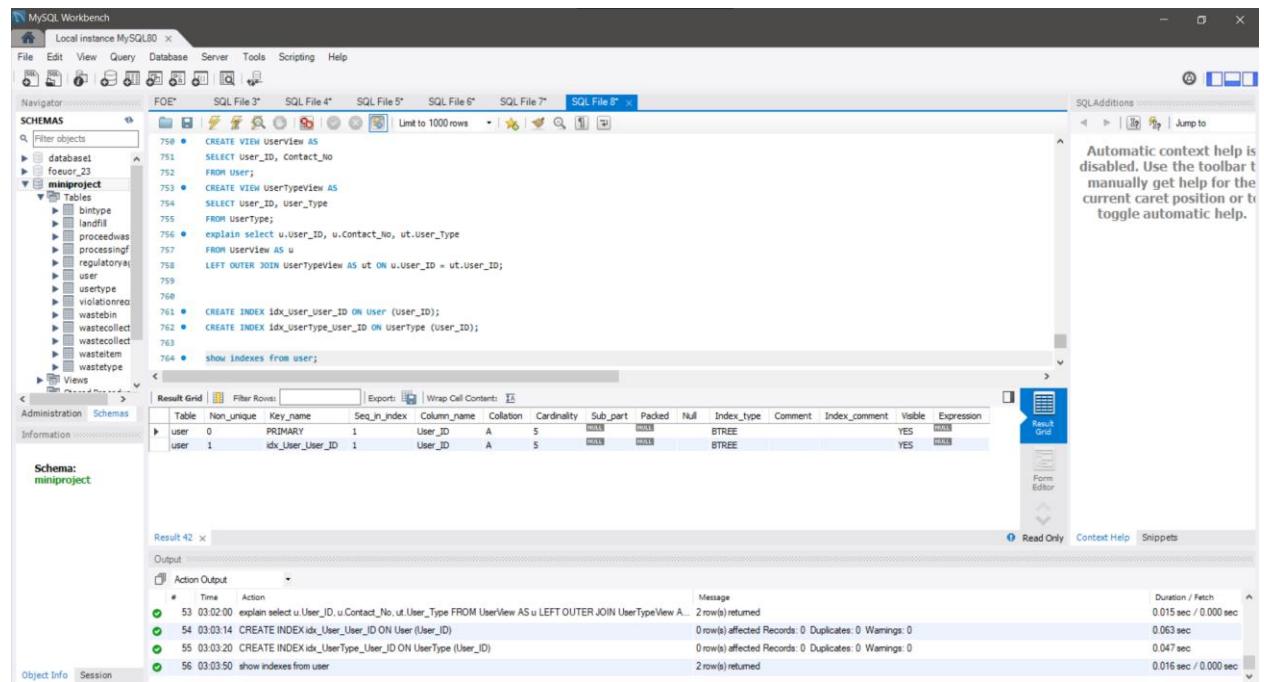
The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** miniproject
- Tables:** user, userType
- SQL Editor:** FOE*


```

749 -- before tuning
750 • CREATE VIEW UserView AS
751     SELECT User_ID, Contact_No
752     FROM User;
753 • CREATE VIEW UserTypeView AS
754     SELECT User_ID, User_Type
755     FROM UserType;
756 • explain select u.User_ID, u.Contact_No, ut.User_Type
757     FROM UserView AS u
758     LEFT OUTER JOIN UserTypeView AS ut ON u.User_ID = ut.User_ID;
759
760 • CREATE INDEX idx_User_User_ID ON User (User_ID);
761 • CREATE INDEX idx_UserType_User_ID ON UserType (User_ID);
762
763
    
```
- Result Grid:** Shows the execution of the EXPLAIN command for the query.
- Output:** Shows the execution log for the session.

Create index in User



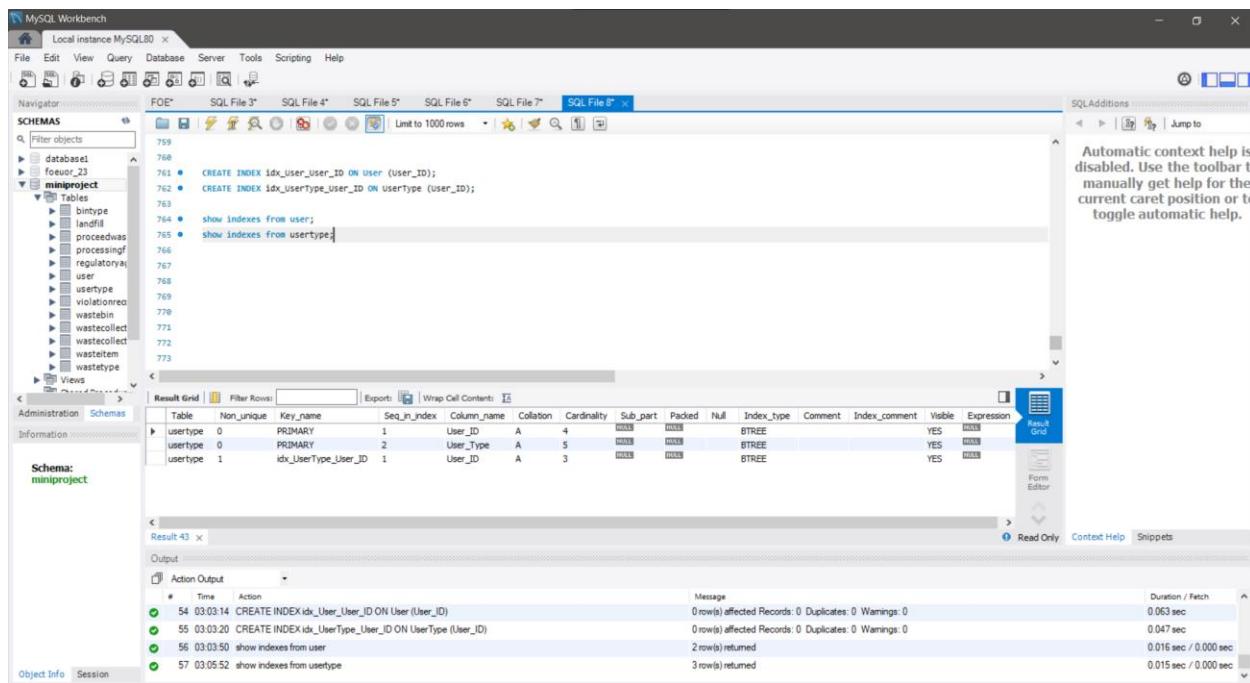
The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** miniproject
- Tables:** user
- SQL Editor:** FOE*


```

759 • CREATE VIEW UserView AS
760     SELECT User_ID, Contact_No
761     FROM User;
762 • CREATE VIEW UserTypeView AS
763     SELECT User_ID, User_Type
764     FROM UserType;
765 • explain select u.User_ID, u.Contact_No, ut.User_Type
766     FROM UserView AS u
767     LEFT OUTER JOIN UserTypeView AS ut ON u.User_ID = ut.User_ID;
768
769 • CREATE INDEX idx_User_User_ID ON User (User_ID);
770 • CREATE INDEX idx_UserType_User_ID ON UserType (User_ID);
771
772 • show indexes from user;
    
```
- Result Grid:** Shows the execution of the SHOW INDEXES command for the user table.
- Output:** Shows the execution log for the session.

Create index in UserType



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

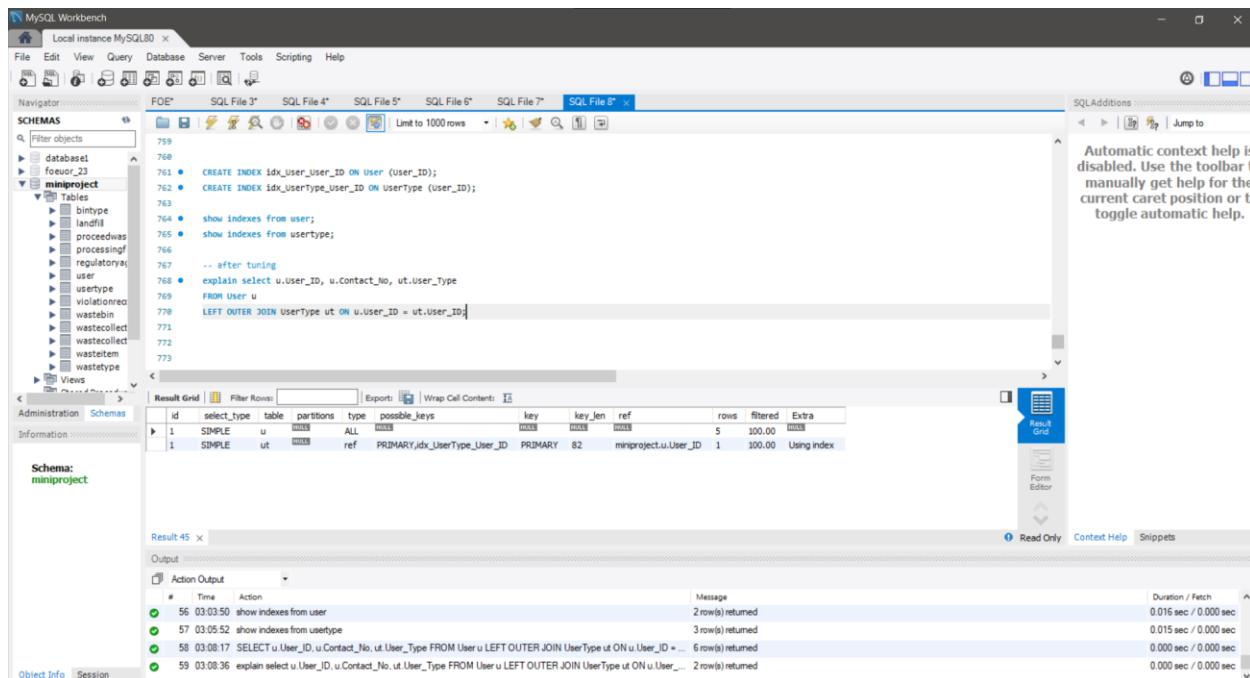
- Schemas:** miniproject
- Tables:** user, userType
- SQL Editor:** Contains the following SQL statements:


```

CREATE INDEX idx_User_User_ID ON User (User_ID);
CREATE INDEX idx_UserType_User_ID ON UserType (User_ID);
show indexes from user;
show indexes from userType;
      
```
- Result Grid:** Displays the results of the SHOW INDEXES command for both tables.
- Action Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
54	03:03:14	CREATE INDEX idx_User_User_ID ON User (User_ID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.063 sec
55	03:03:20	CREATE INDEX idx_UserType_User_ID ON UserType (User_ID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec
56	03:03:50	show indexes from user	2 row(s) returned	0.016 sec / 0.000 sec
57	03:05:52	show indexes from userType	3 row(s) returned	0.015 sec / 0.000 sec

After Tuning



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

- Schemas:** miniproject
- Tables:** user, userType
- SQL Editor:** Contains the following SQL statements:


```

CREATE INDEX idx_User_User_ID ON User (User_ID);
CREATE INDEX idx_UserType_User_ID ON UserType (User_ID);
show indexes from user;
show indexes from userType;
-- after tuning
explain select u.user_ID, u.Contact_No, ut.user_Type
FROM User u
LEFT OUTER JOIN UserType ut ON u.user_ID = ut.User_ID;
      
```
- Result Grid:** Displays the results of the EXPLAIN command, showing the execution plan.
- Action Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
56	03:03:50	show indexes from user	2 row(s) returned	0.015 sec / 0.000 sec
57	03:05:52	show indexes from userType	3 row(s) returned	0.015 sec / 0.000 sec
58	03:08:17	SELECT u.user_ID, u.Contact_No, ut.user_Type FROM User u LEFT OUTER JOIN UserType ut ON u.user_ID = ut.User_ID;	6 row(s) returned	0.000 sec / 0.000 sec
59	03:08:36	explain select u.user_ID, u.Contact_No, ut.user_Type FROM User u LEFT OUTER JOIN UserType ut ON u.user_ID = ut.User_ID;	2 row(s) returned	0.000 sec / 0.000 sec

Right Outer Join

Before Tunning

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Home, New, Open, Save, Import, Export, Run, Stop, Refresh, Help.
- Schemas:** Local instance MySQL80, miniproject.
- SQL Editor:** SQL File 8*, contains the following code:

```
772 • drop INDEX idx_User_User_ID ON User;
773 • drop
774
775 -- before tuning
776 • CREATE VIEW UserView AS
777     SELECT User_ID, Contact_No
778     FROM User;
779 • CREATE VIEW UserTypeView AS
780     SELECT User_ID, User_Type
781     FROM UserType;
782 • explain SELECT u.User_ID, u.Contact_No, ut.User_Type
783     FROM UserView AS u
784     RIGHT OUTER JOIN UserTypeView AS ut ON ut.User_ID = u.User_ID;
```
- Result Grid:** Shows the execution results of the EXPLAIN command.
- Action Output:** Shows the actions taken: dropping indexes and running the EXPLAIN command.

Show index in User

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Home, New, Open, Save, Import, Export, Run, Stop, Refresh, Help.
- Schemas:** Local instance MySQL80, miniproject.
- SQL Editor:** SQL File 8*, contains the following code:

```
784 • RIGHT OUTER JOIN UserTypeView AS ut ON ut.User_ID = u.User_ID;
785
786 • CREATE INDEX idx_User_User_ID ON User (User_ID);
787 • CREATE INDEX idx_UserType_User_ID ON UserType (User_ID);
788
789 • show indexes from user;
790
791
792
793
794
795
796
797
```
- Result Grid:** Shows the indexes for the User table.
- Action Output:** Shows the actions taken: creating indexes and displaying indexes.

Create index in UserType

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons for database management.
- Schemas:** Navigator pane showing the schema structure. The `miniproject` schema is selected, containing tables like `user`, `userrole`, `wastecollect`, etc.
- SQL Editor:** SQL File 8* tab active, displaying the following SQL code:


```

784 RIGHT OUTER JOIN UserTypeView AS ut ON ut.User_ID = u.User_ID;
785
786 • CREATE INDEX idx_User_User_ID ON User (User_ID);
787 • CREATE INDEX idx_UserType_User_ID ON UserType (User_ID);
788
789 • show indexes from user;
790 • show indexes from usertype;
    
```
- Result Grid:** Shows the results of the `SHOW INDEXES` command for the `usertype` table, listing three indexes:

Table	Non_unique	Key_name	Seq_in_Index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Visible	Expression
usertype	0	PRIMARY	1	User_ID	A	4				BTREE		YES	NO
usertype	0	PRIMARY	2	User_Type	A	5				BTREE		YES	NO
usertype	1	idx_UserType_User_ID	1	User_ID	A	3				BTREE		YES	NO
- Action Output:** Displays the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
4	08:54:56	CREATE INDEX idx_User_User_ID ON User (User_ID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec
5	08:55:01	CREATE INDEX idx_UserType_User_ID ON UserType (User_ID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
6	08:55:17	show indexes from user	2 row(s) returned	0.000 sec / 0.000 sec
7	08:57:33	show indexes from usertype	3 row(s) returned	0.015 sec / 0.000 sec

After Tuning

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons for database management.
- Schemas:** Navigator pane showing the schema structure. The `miniproject` schema is selected, containing tables like `user`, `userrole`, `wastecollect`, etc.
- SQL Editor:** SQL File 8* tab active, displaying the following SQL code:


```

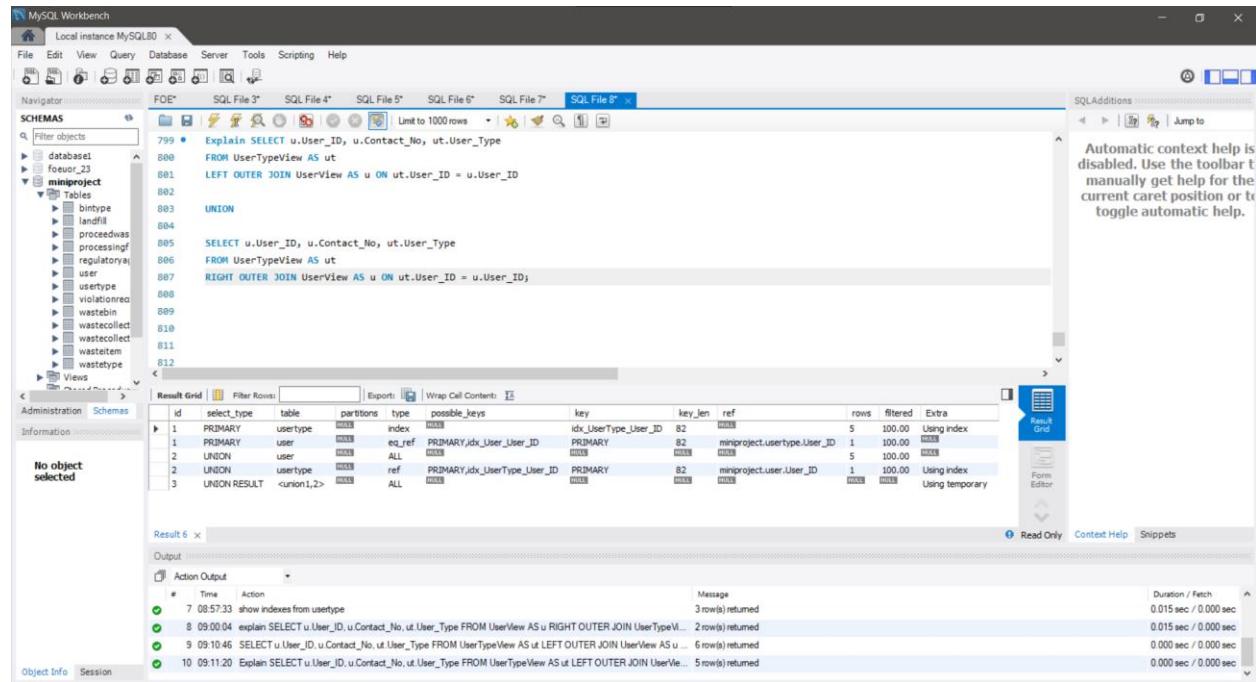
784 RIGHT OUTER JOIN UserTypeView AS ut ON ut.User_ID = u.User_ID;
785
786 • CREATE INDEX idx_User_User_ID ON User (User_ID);
787 • CREATE INDEX idx_UserType_User_ID ON UserType (User_ID);
788
789 • show indexes from user;
790 • show indexes from usertype;
791
792 • explain SELECT u.User_ID, u.Contact_No, ut.User_Type
    FROM UserView AS u
    RIGHT OUTER JOIN UserTypeView AS ut ON ut.User_ID = u.User_ID;
    
```
- Result Grid:** Shows the execution plan for the EXPLAIN command, detailing the query execution steps:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	user		index	idx_UserType_User_ID	82	NULL		5	100.00	Using index
1	SIMPLE	user		eq_ref	PRIMARY	82	miniproject.usertype.User_ID	1	100.00	NULL	
- Action Output:** Displays the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
5	08:55:01	CREATE INDEX idx_UserType_User_ID ON UserType (User_ID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
6	08:55:17	show indexes from user	2 row(s) returned	0.000 sec / 0.000 sec
7	08:57:33	show indexes from usertype	3 row(s) returned	0.015 sec / 0.000 sec
8	09:00:04	explain SELECT u.User_ID, u.Contact_No, ut.User_Type FROM UserView AS u RIGHT OUTER JOIN UserTypeV...	2 row(s) returned	0.015 sec / 0.000 sec

Full outer join

Before Tunning



The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```

799 • Explain SELECT u.User_ID, u.Contact_No, ut.User_Type
800   FROM UserTypeView AS ut
801   LEFT OUTER JOIN UserView AS u ON ut.User_ID = u.User_ID
802
803 UNION
804
805   SELECT u.User_ID, u.Contact_No, ut.User_Type
806   FROM UserTypeView AS ut
807   RIGHT OUTER JOIN UserView AS u ON ut.User_ID = u.User_ID
808
809
810
811
812
  
```

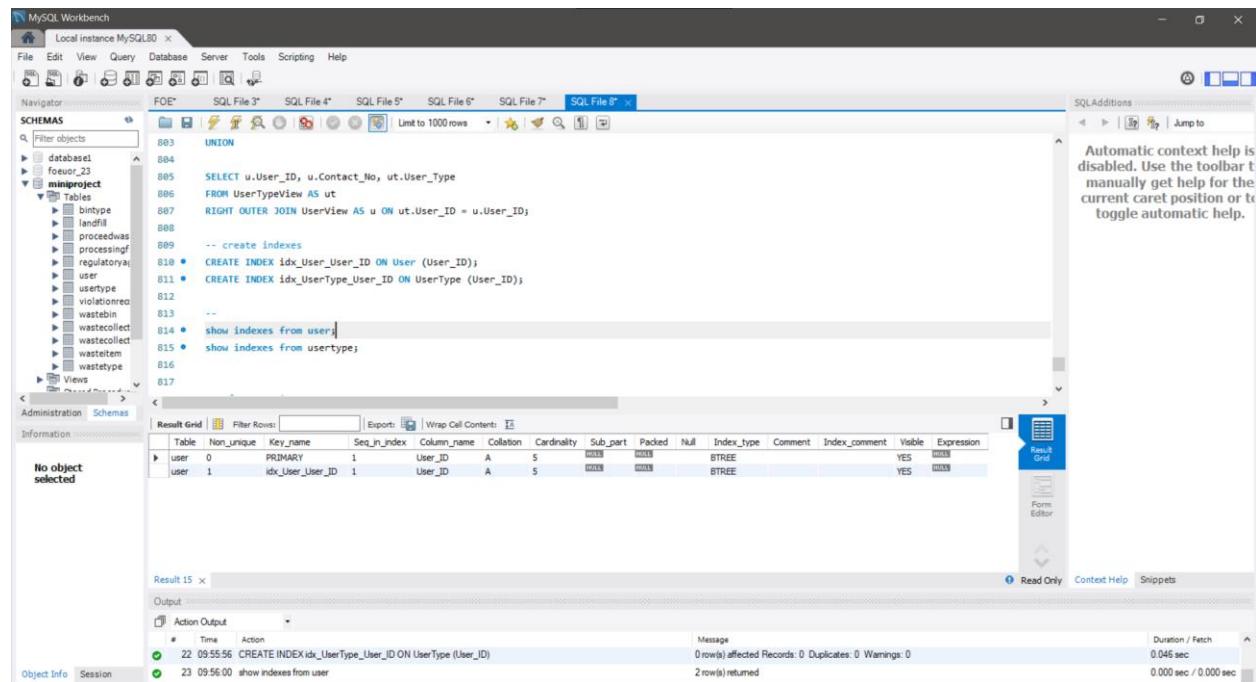
The Result Grid shows the execution plan for the EXPLAIN command, detailing the use of indexes and temporary tables.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	user		index	idx_UserType_User_ID	idx_UserType_User_ID	82		5	100.00	Using index
1	PRIMARY	user		eq_ref	PRIMARY, idx_UserType_User_ID	PRIMARY	82	miniproject.usertype.User_ID	1	100.00	
2	UNION	user		ALL		user	82		5	100.00	
2	UNION	user		ref	PRIMARY, idx_UserType_User_ID	PRIMARY	82	miniproject.user.User_ID	1	100.00	Using index
3	UNION RESULT	<union1,2>		ALL		user	82				Using temporary

The Action Output shows the execution log:

- 7 08:57:33 show indexes from user
- 8 09:00:04 explain SELECT u.User_ID, u.Contact_No, ut.User_Type FROM UserView AS ut RIGHT OUTER JOIN UserTypeView AS ut
- 9 09:10:46 SELECT u.User_ID, u.Contact_No, ut.User_Type FROM UserTypeView AS ut LEFT OUTER JOIN UserView AS u
- 10 09:11:20 Explain SELECT u.User_ID, u.Contact_No, ut.User_Type FROM UserTypeView AS ut LEFT OUTER JOIN UserView AS u

Create index in User



The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```

803 UNION
804
805   SELECT u.User_ID, u.Contact_No, ut.User_Type
806   FROM UserTypeView AS ut
807   RIGHT OUTER JOIN UserView AS u ON ut.User_ID = u.User_ID;
808
809 -- create indexes
810 • CREATE INDEX idx_User_User_ID ON User (User_ID);
811 • CREATE INDEX idx_UserType_User_ID ON UserType (User_ID);
812
813 --
814 • show indexes from user;
815 • show indexes from usertypes;
816
817
  
```

The Result Grid shows the creation of two indexes on the User table.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
user	0	PRIMARY	1	User_ID	A	5				BTREE			YES	
user	1	idx_User_User_ID	1	User_ID	A	5				BTREE			YES	

The Action Output shows the execution log:

- 22 09:55:56 CREATE INDEX idx_UserType_User_ID ON UserType (User_ID)
- 23 09:56:00 show indexes from user

Create index in UserType

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

- Schemas:** Local instance MySQL80, miniproject
- SQL Editor:** Contains the following SQL code:


```

803 UNION
804
805 SELECT u.User_ID, u.Contact_No, ut.User_Type
806 FROM UserTypeView AS ut
807 RIGHT OUTER JOIN UserView AS u ON ut.User_ID = u.User_ID;
808
809 -- create indexes
810 • CREATE INDEX idx_User_User_ID ON User (User_ID);
811 • CREATE INDEX idx_UserType_User_ID ON UserType (User_ID);
812
813 --
814 • show indexes from user;
815 • show indexes from userType;
816
817
      
```
- Result Grid:** Shows the index definitions for the `user` and `userType` tables.
- Action Output:** Displays two log entries:
 - 23 09:56:00 show indexes from user
 - 24 09:59:07 show indexes from userType

After tuning

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

- Schemas:** Local instance MySQL80, miniproject
- SQL Editor:** Contains the following SQL code:


```

815 • show indexes from userType;
816
817
818 -- after tuning
819 • Explain SELECT u.User_ID, u.Contact_No, ut.User_Type
820 FROM UserTypeView AS ut
821 LEFT OUTER JOIN UserView AS u ON ut.User_ID = u.User_ID;
822
823 UNION
824
825 SELECT u.User_ID, u.Contact_No, ut.User_Type
826 FROM UserTypeView AS ut
827 RIGHT OUTER JOIN UserView AS u ON ut.User_ID = u.User_ID;
828
829
      
```
- Result Grid:** Shows the execution plan for the tuned query, including the number of rows, filtered rows, and extra information.
- Action Output:** Displays two log entries:
 - 24 09:59:07 show indexes from userType
 - 25 09:59:55 Explain SELECT u.User_ID, u.Contact_No, ut.User_Type FROM UserTypeView AS ut LEFT OUTER JOIN UserView AS u ON ut.User_ID = u.User_ID;

Nested Query with inner Join

Before Tunning

```

MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator FOE* SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 8* 
SCHEMAS Filter objects
databasel
fouur_23
miniproject
Tables
binstype
landfill
procedwas
processingf
regulatorys
user
usertype
violationreq
wastebin
wastecollect
wasteitem
wastetype
Views
Administration Schemas
Information No object selected
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
1 id select_type table partitions type possible_keys key key_len ref rows filtered Extra
1 SIMPLE WasteCollection ALL idx_Collector_ID_wc_idx_WasteCollection_Colle... key1 42 miniproject.WasteCollection.Collector_ID 1 100.00 Using where
1 SIMPLE wd eq_ref PRIMARY_idx_Collector_ID_wd PRIMARY 42 miniproject.WasteCollection.Collector_ID 1 100.00
Result 24 x
Output Action Output
# Time Action Message Duration / Fetch
31 10:25:11 SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wc.First_Name, wc.Last_Name FROM WasteCollection... 1 row(s) returned 0.000 sec / 0.000 sec
32 10:26:32 explain SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wc.First_Name, wc.Last_Name FROM ( SEL... 2 row(s) returned 0.000 sec / 0.000 sec
Object Info Session

```

The screenshot shows the MySQL Workbench interface with a complex SQL query in the editor. The query includes a nested query, an inner join, and multiple EXPLAIN statements. The results grid shows one row from the main query and two rows from the EXPLAIN statements. The status bar at the bottom indicates the execution time.

Create index in WasteCollection

```

MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator FOE* SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 8* 
SCHEMAS Filter objects
databasel
fouur_23
miniproject
Tables
binstype
landfill
procedwas
processingf
regulatorys
user
usertype
violationreq
wastebin
wastecollect
wasteitem
wastetype
Views
Administration Schemas
Information No object selected
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
Table Non_Unique Key_name Seq_in_Index Column_name Collation Cardinality Sub_part Packed Null Index_type Comment Index_comment
wastecollection 0 PRIMARY 1 Collection_ID A 5 NULL NULL BTREE
wastecollection 1 FK_Bin_ID 1 Bin_ID A 4 NULL NULL YES BTREE
wastecollection 1 FK_Parent_Collection_ID 1 Parent_Collection_ID A 1 NULL NULL YES BTREE
wastecollection 1 idx_Collector_ID_wc 1 Collector_ID A 3 NULL NULL YES BTREE
wastecollection 1 idx_WasteCollection_Collector_ID 1 Collector_ID A 3 NULL NULL YES BTREE
wastecollection 1 idx_WasteCollection_CollectionDate 1 CollectionDate A 5 NULL NULL YES BTREE
Result 25 x
Output Action Output
# Time Action Message Duration / Fetch
35 10:30:28 CREATE INDEX idx_WasteCollector_Collector_ID ON WasteCollector (Collector_ID)
0 row(s) affected. 1 warning(s): 1831 Duplicate index idx_WasteCollector_Collector_ID defined on the table 'miniproject.WasteCollector'.
0.047 sec
36 10:30:53 show indexes from wastecollection
6 row(s) returned
0.016 sec / 0.000 sec
Object Info Session

```

The screenshot shows the MySQL Workbench interface with a query to create indexes on the WasteCollection table. The results grid shows the newly created indexes. The status bar at the bottom indicates the execution time.

Create index in Waste Collection

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

- SQL Editor:** Contains the following SQL code:


```

831 • FROM (
832   SELECT *
833     FROM WasteCollection
834   WHERE CollectionDate = '2024-05-06'
835 ) AS wc
836   inner JOIN WasteCollector wc1 ON wc.Collector_ID = wc1.Collector_ID;
837
838 • CREATE INDEX idx_WasteCollection_CollectionDate ON WasteCollection (CollectionDate);
839 • CREATE INDEX idx_WasteCollection_Collector_ID ON WasteCollection (Collector_ID);
840 • CREATE INDEX idx_WasteCollector_Collector_ID ON WasteCollector (Collector_ID);
841
842 • show indexes from wastecollection;
843 • show indexes from wastecollector;
844
845 • Explain SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wc1.First_Name, wc1.Last_Name
846   FROM WasteCollection wc
      
```
- Result Grid:** Shows the results of the EXPLAIN command for the query. It includes columns: Table, Non_unique, Key_name, Seq_in_index, Column_name, Collation, Cardinality, Sub_part, Packed, Null, Index_type, Comment, Index_comment, and Visible. The data is as follows:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible
wastecollector	0	PRIMARY	1	Collector_ID	A	5				BTREE		YES	
wastecollector	1	idx_Collector_ID_wid	1	Collector_ID	A	5				BTREE		YES	
wastecollector	1	idx_WasteCollector_Collector_ID	1	Collector_ID	A	5				BTREE		YES	
- Output:** Shows the execution history with two entries:

#	Time	Action	Message	Duration / Fetch
36	10:30:53	show indexes from wastecollection	6 row(s) returned	0.016 sec / 0.000 sec
37	10:32:25	show indexes from wastecollector	3 row(s) returned	0.000 sec / 0.000 sec

Aftr Tunning

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

- SQL Editor:** Contains the following SQL code:


```

841 • CREATE INDEX idx_WasteCollector_Collector_ID ON WasteCollector (Collector_ID);
842 • show indexes from wastecollection;
843 • show indexes from wastecollector;
844
845 • Explain SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wc1.First_Name, wc1.Last_Name
846   FROM WasteCollection wc
847   inner JOIN WasteCollector wc1 ON wc.Collector_ID = wc1.Collector_ID
848   WHERE wc.CollectionDate = '2024-05-06';
      
```
- Result Grid:** Shows the results of the EXPLAIN command for the query. It includes columns: id, select_type, table, partitions, type, possible_keys, key, key_len, ref, rows, filtered, Extra. The data is as follows:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	wc	NULL	ref	idx_Collector_ID_wc_idx_WasteCollection_Colle...	idx_WasteCollection_CollectionDate	4	const	1	100.00	Using where
1	SIMPLE	wc1	NULL	eq_ref	PRIMARY, idx_Collector_ID_wc_idx_WasteCollecto...	PRIMARY	42	miniproject.wc.Collector_ID	1	100.00	NULL
- Output:** Shows the execution history with two entries:

#	Time	Action	Message	Duration / Fetch
37	10:32:25	show indexes from wastecollection	3 row(s) returned	0.000 sec / 0.000 sec
38	10:33:21	Explain SELECT wc.Collection_ID, wc.Bin_ID, wc.CollectionDate, wc1.First_Name, wc1.Last_Name FROM WasteC...	2 row(s) returned	0.000 sec / 0.000 sec

Nested Query with Projection

Before Tunning

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** FOE*, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*.
- SQL Editor:** Contains the following SQL code:

```
849
850     Execute the statement under the keyboard cursor
851
852 • Explain SELECT First_Name FROM User WHERE User_ID IN
853     (SELECT DISTINCT User_ID FROM WasteBin);
854
855
856 • Explain SELECT DISTINCT u.First_Name
857     FROM User u
858     inner JOIN WasteBin wb ON u.User_ID = wb.User_ID;
859
860
861
862
863
864
```
- Result Grid:** Shows the execution plan for the EXPLAIN command. The output is:

ID	Select_Type	Table	Partitions	Type	Possible_Keys	Key	Key_Len	Ref	Rows	Filtered	Extra
1	SIMPLE	WasteBin		index	FK_User_ID	FK_User_ID	83	const	5	60.00	Using where; Using index; LooseScan
1	SIMPLE	User		eq_ref	PRIMARY, idx_User_User_ID	PRIMARY	82	miniproject.WasteBin.User_ID	1	100.00	
- Output Tab:** Action Output shows two log entries:

#	Time	Action	Message	Duration / Fetch
45	10:55:24	Explain SELECT DISTINCT u.First_Name FROM User u JOIN WasteBin wb ON u.User_ID = wb.User_ID	2 row(s) returned	0.000 sec / 0.000 sec
46	11:04:12	Explain SELECT First_Name FROM User WHERE User_ID IN (SELECT DISTINCT User_ID FROM WasteBin)	2 row(s) returned	0.000 sec / 0.000 sec

After Tuning

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** FOE*, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*.
- SQL Editor:** Contains the same SQL code as the 'Before Tuning' section.
- Result Grid:** Shows the execution plan for the EXPLAIN command. The output is:

ID	Select_Type	Table	Partitions	Type	Possible_Keys	Key	Key_Len	Ref	Rows	Filtered	Extra
1	SIMPLE	wb	HASH	index	FK_User_ID	FK_User_ID	83	const	5	100.00	Using where; Using index; Using temporary
1	SIMPLE	u	HASH	eq_ref	PRIMARY, idx_User_User_ID	PRIMARY	82	miniproject.wb.User_ID	1	100.00	
- Output Tab:** Action Output shows two log entries:

#	Time	Action	Message	Duration / Fetch
47	11:04:49	SELECT DISTINCT u.First_Name FROM User u inner JOIN WasteBin wb ON u.User_ID = wb.User_ID LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
48	11:04:57	Explain SELECT DISTINCT u.First_Name FROM User u inner JOIN WasteBin wb ON u.User_ID = wb.User_ID	2 row(s) returned	0.000 sec / 0.000 sec

Nested Query with Union

Before Tunning

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

```
861 -- nested query in union
862
863 Explain SELECT Type_ID, SourceTable, Quantity
864   FROM (
865     SELECT Type_ID, 'WasteItem' AS SourceTable, Weight AS Quantity
866       FROM WasteItem
867     UNION
868     SELECT Type_ID, 'ProceedWaste' AS SourceTable, Quantity AS Quantity
869       FROM ProceedWaste
870   ) AS CombinedData;
```

The Explain output shows the execution plan for the query:

ID	Select_Type	Table	Partitions	Type	Possible_Keys	Key	Key_Len	Ref	Rows	Filtered	Extra
1	PRIMARY	<derived2>		ALL	none	none	none	none	7	100.00	
2	DERIVED	WasteItem		ALL	none	none	none	none	3	100.00	
3	UNION	ProceedWaste		ALL	none	none	none	none	4	100.00	
4	UNION RESULT	<union2,3>		ALL	none	none	none	none	Using temporary		

The Result Grid shows the final combined data:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>		ALL	none	none	none	none	7	100.00	
2	DERIVED	WasteItem		ALL	none	none	none	none	3	100.00	
3	UNION	ProceedWaste		ALL	none	none	none	none	4	100.00	
4	UNION RESULT	<union2,3>		ALL	none	none	none	none	Using temporary		

The Output pane shows the actions taken:

#	Time	Action	Message	Duration / Fetch
48	11:04:57	Explain SELECT DISTINCT u.First_Name FROM User u inner JOIN WasteBin wb ON u.User_ID = wb.User_ID	2 row(s) returned	0.000 sec / 0.000 sec
49	11:29:46	Explain SELECT Type_ID, SourceTable, Quantity FROM (SELECT Type_ID, 'WasteItem' AS SourceTable, Weight AS Q... 4 row(s) returned	0.016 sec / 0.000 sec

Create Index WasteItem

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

```
861 -- nested query in union
862
863 Explain SELECT Type_ID, SourceTable, Quantity
864   FROM (
865     SELECT Type_ID, 'WasteItem' AS SourceTable, Weight AS Quantity
866       FROM WasteItem
867     UNION
868     SELECT Type_ID, 'ProceedWaste' AS SourceTable, Quantity AS Quantity
869       FROM ProceedWaste
870   ) AS CombinedData;
872
873 CREATE INDEX idx_WasteItem_Type_ID ON WasteItem (Type_ID);
874 CREATE INDEX idx_ProceedWaste_Type_ID ON ProceedWaste (Type_ID);
875
876 show indexes from WasteItem;
```

The Result Grid shows the indexes created:

Table	Non_unique	Key_name	Seq_in_Index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
wasteitem	1	FK_WasteItem_Collection_ID	1	Collection_ID	A	2	NULL	NULL	YES	BTREE			YES	NULL
wasteitem	1	FK_WasteItem_Landfill_ID	1	Landfill_ID	A	2	NULL	NULL	YES	BTREE			YES	NULL
wasteitem	1	idx_WasteItem_Type_ID	1	Type_ID	A	3	NULL	NULL	YES	BTREE			YES	NULL

The Output pane shows the results of the index creation:

#	Time	Action	Message	Duration / Fetch
51	11:31:30	CREATE INDEX idx_ProceedWaste_Type_ID ON ProceedWaste (Type_ID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.062 sec
52	11:32:20	show indexes from WasteItem	3 row(s) returned	0.000 sec / 0.000 sec

Create Index in ProceedWaste

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

- SQL Editor:** Displays the SQL code for creating indexes. The code includes:


```

864 • Explain SELECT Type_ID, SourceTable, Quantity
865   FROM (
866     SELECT Type_ID, 'WasteItem' AS SourceTable, Weight AS Quantity
867     FROM WasteItem
868     UNION
869     SELECT Type_ID, 'ProceedWaste' AS SourceTable, Quantity AS Quantity
870     FROM ProceedWaste
871   ) AS CombinedData;
872
873 • CREATE INDEX idx_WasteItem_Type_ID ON WasteItem (Type_ID);
874 • CREATE INDEX idx_ProceedWaste_Type_ID ON ProceedWaste (Type_ID);
875
876 • show indexes from WasteItem;
877 • show indexes from proceedwaste;
878
879
      
```
- Result Grid:** Shows the results of the 'show indexes' command for the 'proceedwaste' table. The output is:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
proceedwaste	0	PRIMARY	1	ProcedWaste	A	4				BTREE			YES	NULL
proceedwaste	1	idx_Facility_ID_pw	1	Facility_ID	A	3				BTREE			YES	NULL
proceedwaste	1	idx_ProceedWaste_Type_ID	1	Type_ID	A	3				BTREE			YES	NULL
- Output Panel:** Shows the execution log with two entries:

#	Time	Action	Message	Duration / Fetch
52	11:32:20	show indexes from WasteItem	3 row(s) returned	0.000 sec / 0.000 sec
53	11:34:33	show indexes from proceedwaste	3 row(s) returned	0.000 sec / 0.000 sec

After Tuning

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

- SQL Editor:** Displays the tuned query. The code includes:


```

876 • show indexes from WasteItem;
877 • show indexes from proceedwaste;
878
879 -- after tuning
880 • Explain SELECT Type_ID, SourceTable, Quantity
881   FROM (
882     SELECT Type_ID, 'WasteItem' AS SourceTable, Weight AS Quantity
883     FROM WasteItem
884     UNION ALL
885     SELECT Type_ID, 'ProceedWaste' AS SourceTable, Quantity AS Quantity
886     FROM ProceedWaste
887   ) AS CombinedData;
888
889
890
891
      
```
- Result Grid:** Shows the results of the 'Explain' command for the tuned query. The output is:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>		ALL	NULL	NULL	NULL	NULL	7	100.00	NULL
2	DERIVED	WasteItem		ALL	NULL	NULL	NULL	NULL	3	100.00	NULL
3	UNION	ProceedWaste		ALL	NULL	NULL	NULL	NULL	4	100.00	NULL
- Output Panel:** Shows the execution log with two entries:

#	Time	Action	Message	Duration / Fetch
54	11:37:48	SELECT Type_ID, SourceTable, Quantity FROM (SELECT Type_ID, 'WasteItem' AS SourceTable, Weight AS Quantity ...	7 row(s) returned	0.000 sec / 0.000 sec
55	11:38:01	Explain SELECT Type_ID, SourceTable, Quantity FROM (SELECT Type_ID, 'WasteItem' AS SourceTable, Weight AS Q ...	3 row(s) returned	0.000 sec / 0.000 sec