



# **EE4350-Database Systems**

## **Mini Project- Database for a Trip Planner App**

### **Group Members**

EG/2021/ 4667 – Malsha H.C

EG/2021/4675 – Mewan L.A.D.Y

EG/2021/4671 - Manthilaka M.M.P.L

# Contents

Introduction.....	3
1 Chapter 01 – Requirement Analysis .....	4
1.1 Data Requirements .....	4
1.2 Functional Requirements.....	4
2 Chapter 02- Conceptual Design.....	6
3 Chapter 03- Implementation.....	8
3.1 Creating Database .....	8
3.2 Creating tables.....	8
3.3 Insert Data to the Tables.....	10
3.4 Update data in the tables .....	16
3.5 Delete data in the tables .....	20
4 Chapter 04 – Transactions .....	21
4.1 Simple Queries .....	21
4.2 Complex Queries.....	24
5 Chapter 05- Database Tuning .....	31

# **Introduction**

This report outlines the development process of the “Trip Planning Application Database”. The primary objective of the database is to efficiently manage and store the important data related to users who are using the application to plan their trips. The database consists of crucial data such as the user’s password and planned trips, including accommodations, destinations, and activities. MySQL was employed as the database management system.

## **Step 01- Requirement Analysis**

The initial phase was to identify the functional and data requirements of the database. This includes understanding the functionality of the app and finalizing the types of information to be stored. After that, the attributes for each entity were identified.

## **Step 02 – Conceptual Design**

The conceptual design process was done based on the data requirements that were identified during the first step. According to the data and functionality requirements, the ER diagram was created using the visual paradigm tool. This diagram is created focusing on the organization of data and the relationships between them. It helps to understand the entities, their attributes, and how they relate to each other. Notations to show structural constraints and role names in relationships were also included. Then the conceptual schema is used to create a logical design of the database. Creating the UML diagram involved defining classes with careful consideration to the Normalization.

## **Step 03- Implementation**

Implementation of the database was done using MySQL client with 2NF of the representational data model. Schema creation, table creation, and Insert, Update, and Delete operations were also done. In this phase, we populated our database with suitable values.

## **Step 04 – Transactions**

In the Transaction phase, we wrote queries to retrieve data from the database. These queries consist of 7 simple queries and 13 Complex queries based on the requirements stated in the project guidelines.

## **Step 05- Tunning**

We focused on improving efficiency and responsiveness of our SQL queries. For this, first, we identified the slow-running queries and then applied optimization techniques like improving filtering, aggregating, indexing, and optimizing joins. We also used parameterization to prevent SQL injection.

# 1 Chapter 01 – Requirement Analysis

Functional and Data Requirements that were identified for the database are listed below.

## 1.1 Data Requirements

Entity	Attributes
User	User_Id, User Name, First_name, Last_name, Password, Phone Number, Email, Age, Date of Birth, Gender, Profile Picture, Guide_id
Review	Comments, Ratings
Trip	Trip_Id, Start_date, End_date, Description, Status
Accommodation	Acco_Id, Acco_state, Acco_city, Acco_PhoneNumber, Acco_name, Check_out_date, Check_in_date
Transportation	Transportation_Id, Arrival_date, Departure_date, Location, Mode
Expense	Expense_Id, Amount, Notes, Category
Destination	Destination_Id, Latitude, Longitude, Dest_Name, Description
Activity	Name, Time

## 1.2 Functional Requirements

Functional Area	Functional Requirement
User Management	<ol style="list-style-type: none"><li>1. Registration- Users can register by providing the necessary information.</li><li>2. Login- Registered users should be able to log into the system securely using their password.</li><li>3. Profile Management- Users can update their profile information</li><li>4. Role Management – Guides are required to register and get a Guide_id.</li></ol>
Trip Management	<ol style="list-style-type: none"><li>1. Create Trip: Users should have the ability to create new trips by providing trip details.</li><li>2. View Trips- Users should be able to view details of their planned trips.</li></ol>

	<p>3. Edit Trips- Users have access to edit the trip they have planned already.</p> <p>4. Delete Trip- Users can cancel the trip by deleting it.</p>
Accommodation	<p>1. Book Accommodation- Book Accommodations for the trips.</p> <p>2. View Accommodation – Users should be able to view accommodation details.</p>
Transportation	<p>1. Book Transportation- Users should be able to book transportation for planned trips.</p> <p>2. View Transportation- Users should be able to view details of their transportation mode.</p>
Expenses	<p>1. Record Expenses- Users should be able to record trip expenses specifying the amount and other details.</p> <p>2. View Expenses- Users should be able to view the summary of the expenses for their trip.</p>
Destinations	<p>1. Search destinations- Before planning users should be able to search the destinations based on criteria such as name or locations.</p> <p>2. View Destination details- Users should be able to view all the details of the destinations they have chosen.</p>
Activities	<p>1. Choose activities- Users should be able to choose activities that are relevant to their destinations.</p> <p>2. View Activities- Users should be able to view the activities they have chosen</p>
Reviews	<p>1. Submit Reviews- Users have the ability to submit reviews for destinations, accommodations etc.</p> <p>2. View Reviews- Users should have the ability to view the feedback submitted by themselves and other users.</p>
Security and Access Control	<p>1. Authentication- Ensure a secure login system with username and password.</p>

## 2 Chapter 02- Conceptual Design

The conceptual data model is created to represent the overall structure of the database. It provides a valuable opportunity to understand the database very easily. We have created the ER diagram using the visual paradigm tool.

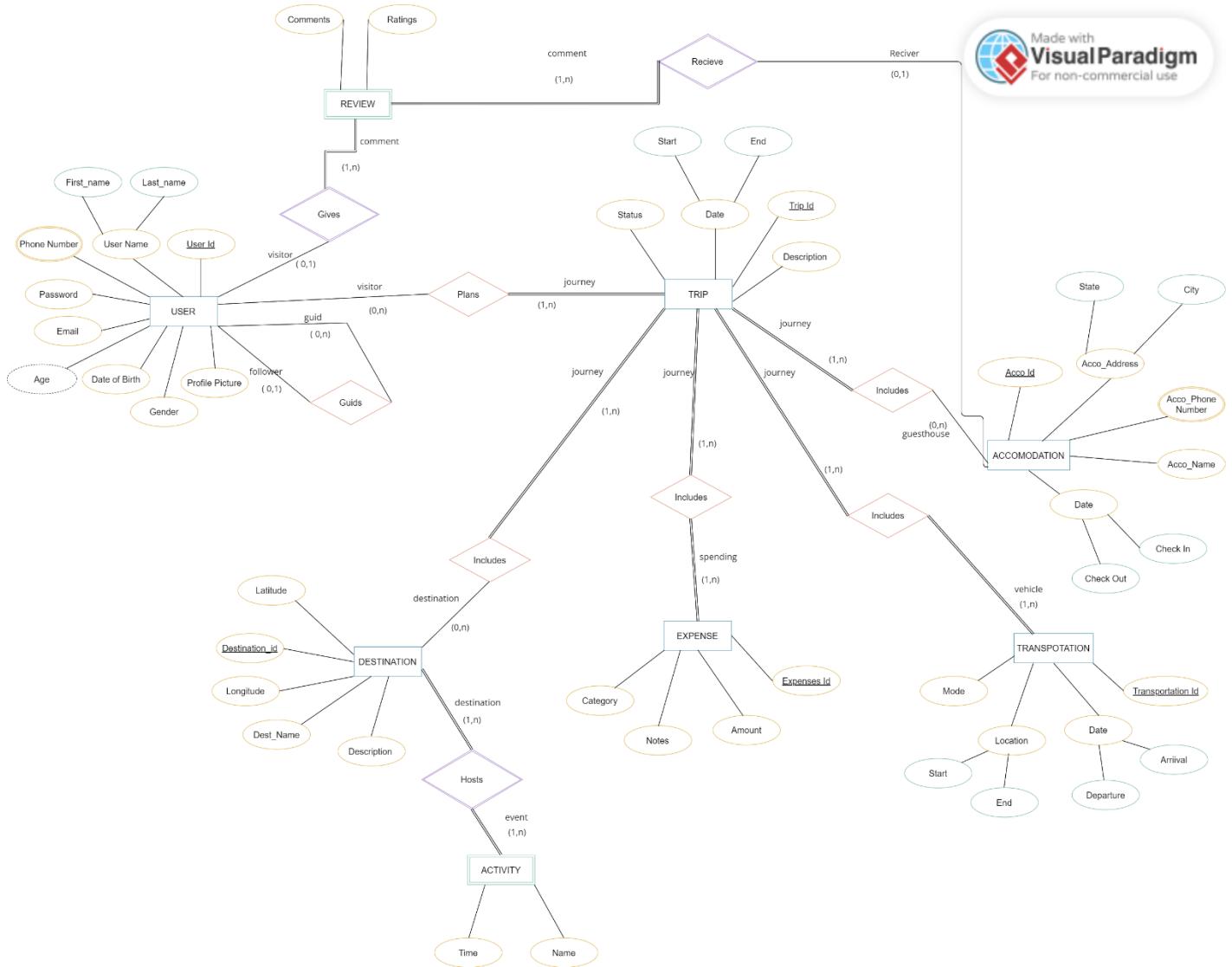


Figure 1: Entity Relationship Diagram

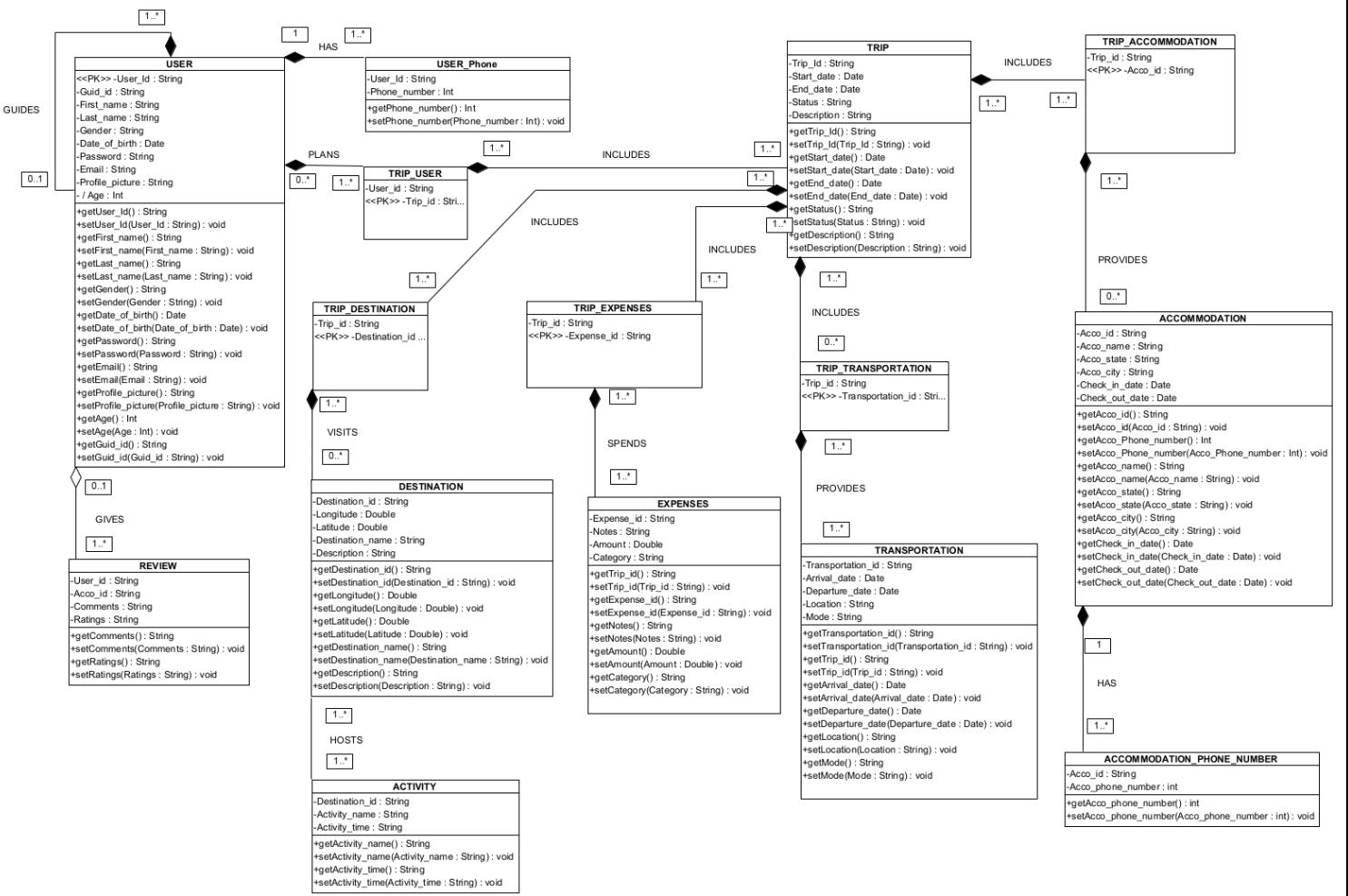
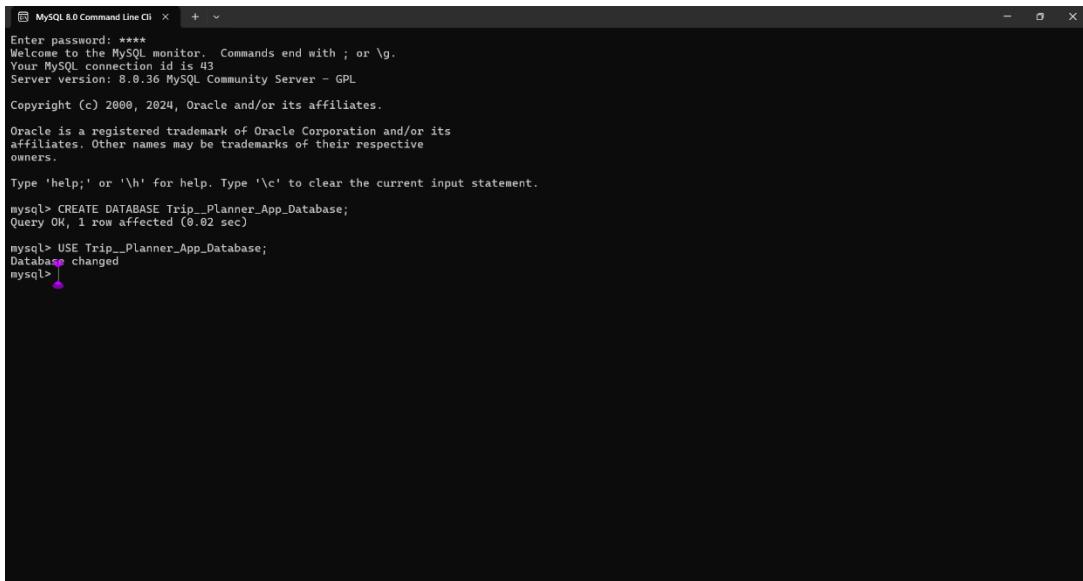


Figure 2: UML Class Diagram

# 3 Chapter 03- Implementation

## 3.1 Creating Database



```
MySQL 8.0 Command Line Cli < + ^

Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 8.0.36 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

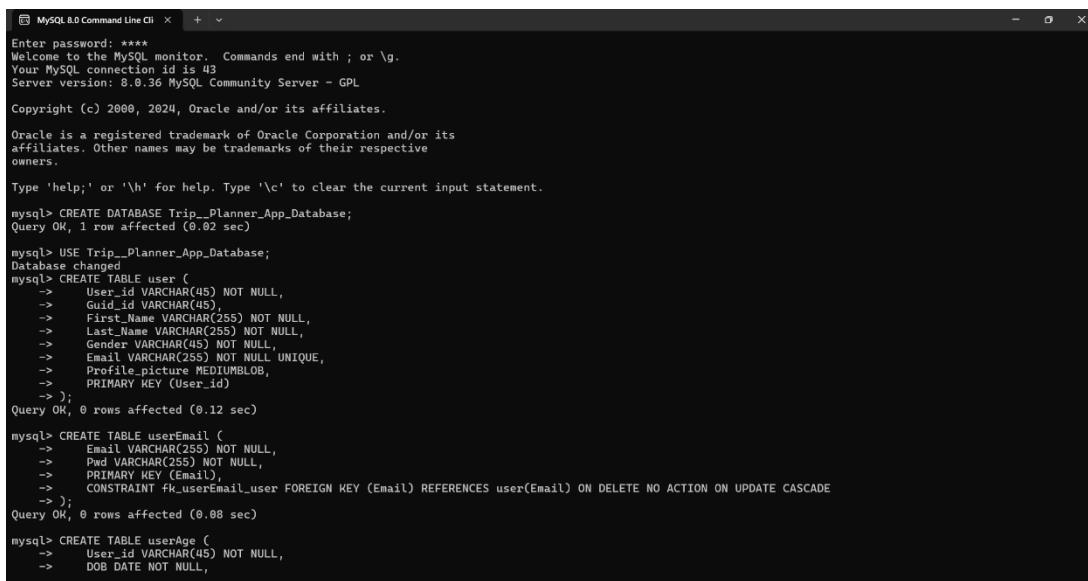
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE Trip_Planner_App_Database;
Query OK, 1 row affected (0.02 sec)

mysql> USE Trip_Planner_App_Database;
Database changed
mysql>
```

Figure 3: Creating Database

## 3.2 Creating tables



```
MySQL 8.0 Command Line Cli < + ^

Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 8.0.36 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE Trip_Planner_App_Database;
Query OK, 1 row affected (0.02 sec)

mysql> USE Trip_Planner_App_Database;
Database changed
mysql> CREATE TABLE user (
    >     User_id VARCHAR(45) NOT NULL,
    >     Guid_id VARCHAR(45),
    >     First_Name VARCHAR(255) NOT NULL,
    >     Last_Name VARCHAR(255) NOT NULL,
    >     Gender VARCHAR(45) NOT NULL,
    >     Email VARCHAR(255) NOT NULL UNIQUE,
    >     Profile_picture MEDIUMBLOB,
    >     PRIMARY KEY (User_id)
    > );
Query OK, 0 rows affected (0.12 sec)

mysql> CREATE TABLE userEmail (
    >     Email VARCHAR(255) NOT NULL,
    >     Pwd VARCHAR(255) NOT NULL,
    >     PRIMARY KEY (Email),
    >     CONSTRAINT fk_userEmail_user FOREIGN KEY (Email) REFERENCES user(Email) ON DELETE NO ACTION ON UPDATE CASCADE
    > );
Query OK, 0 rows affected (0.88 sec)

mysql> CREATE TABLE userAge (
    >     User_id VARCHAR(45) NOT NULL,
    >     DOB DATE NOT NULL,
```

Figure 4: Creating user, userEmail Tables

```

MySQL> CREATE TABLE userAge (
->     User_id VARCHAR(45) NOT NULL,
->     DOB DATE NOT NULL,
->     Age INT NOT NULL,
->     CONSTRAINT fk_userAge_user FOREIGN KEY (User_id) REFERENCES user(User_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.11 sec)

MySQL> CREATE TABLE userPhone (
->     User_id VARCHAR(45) NOT NULL,
->     Phone_number VARCHAR(45),
->     CONSTRAINT fk_userPhone_user FOREIGN KEY (User_id) REFERENCES user(User_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.10 sec)

MySQL> CREATE TABLE tripUser (
->     User_id VARCHAR(45) NOT NULL,
->     Trip_id VARCHAR(45) NOT NULL,
->     PRIMARY KEY (Trip_id),
->     CONSTRAINT fk_tripUser_user FOREIGN KEY (User_id) REFERENCES user(User_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.10 sec)

MySQL> CREATE TABLE trip (
->     Trip_id VARCHAR(45) NOT NULL,
->     Start_date DATE NOT NULL,
->     End_date DATE NOT NULL,
->     Status VARCHAR(45) NOT NULL,
->     Description VARCHAR(255),
->     -- PRIMARY KEY (Trip_id),
->     CONSTRAINT fk_tripUser_trip FOREIGN KEY (Trip_id) REFERENCES tripUser(Trip_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.09 sec)

MySQL> CREATE TABLE tripAccommodation (
->     Trip_id VARCHAR(45) NOT NULL,
->     Acco_id VARCHAR(45) NOT NULL,
->     PRIMARY KEY (Acco_id),
->     CONSTRAINT fk_tripAccommodation_trip FOREIGN KEY (Trip_id) REFERENCES trip(Trip_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );

```

Figure 5: Creating userAge, userPhone, tripUser, trip, tripAccommodation tables

```

MySQL> CREATE TABLE accommodation (
->     Acco_id VARCHAR(45) NOT NULL,
->     Acco_name VARCHAR(255) NOT NULL,
->     Acco_state VARCHAR(45) NOT NULL,
->     Acco_city VARCHAR(45),
->     PRIMARY KEY (Acco_id),
->     CONSTRAINT fk_tripAccommodation_accommodation FOREIGN KEY (Acco_id) REFERENCES tripAccommodation(Acco_id) ON DELETE NO ACTION ON UPDATE CASCADE
->     -- INDEX idx_accommodation (Acco_id)
-> );
Query OK, 0 rows affected (0.89 sec)

MySQL> CREATE TABLE accomoPhone (
->     Acco_id VARCHAR(45) NOT NULL,
->     Phone_number VARCHAR(45),
->     CONSTRAINT fk_accoPhone_accommodation FOREIGN KEY (Acco_id) REFERENCES accommodation(Acco_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.12 sec)

MySQL> CREATE TABLE review (
->     Review_id VARCHAR(45) NOT NULL,
->     Acco_id VARCHAR(45) NOT NULL,
->     Comments VARCHAR(255),
->     Rating ENUM('0', '1', '2', '3', '4', '5'),
->     CONSTRAINT fk_review_user FOREIGN KEY (User_id) REFERENCES user(User_id),
->     CONSTRAINT fk_review_accommodation FOREIGN KEY (Acco_id) REFERENCES accommodation(Acco_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.11 sec)

MySQL> CREATE TABLE tripDestination (
->     Trip_id VARCHAR(45) NOT NULL,
->     Destination_id VARCHAR(45) NOT NULL,
->     PRIMARY KEY (Destination_id),
->     CONSTRAINT fk_tripDestination_trip FOREIGN KEY (Trip_id) REFERENCES trip(Trip_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.09 sec)

MySQL> CREATE TABLE destination (
->     Destination_id VARCHAR(45) NOT NULL,
->     Longitude VARCHAR(45),
->     Latitude VARCHAR(45),
->     Description VARCHAR(255),
->     CONSTRAINT fk_destination_tripDestination FOREIGN KEY (Destination_id) REFERENCES tripDestination(Destination_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );

```

Figure 6: Creating accommodation, accomoPhone, review, tripDestination tables

```

MySQL> CREATE TABLE destination (
->     Destination_id VARCHAR(45) NOT NULL,
->     Longitude VARCHAR(45),
->     Latitude VARCHAR(45),
->     Destination_name VARCHAR(255) NOT NULL,
->     Description VARCHAR(255),
->     CONSTRAINT fk_destination_tripDestination FOREIGN KEY (Destination_id) REFERENCES tripDestination(Destination_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.88 sec)

MySQL> CREATE TABLE activity (
->     Destination_id VARCHAR(45) NOT NULL,
->     Activity_name VARCHAR(255) NOT NULL,
->     Activity_time TIME,
->     CONSTRAINT fk_activity_destination FOREIGN KEY (Destination_id) REFERENCES destination(Destination_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.07 sec)

MySQL> CREATE TABLE tripExpenses (
->     Expense_id VARCHAR(45) NOT NULL,
->     Trip_id VARCHAR(45) NOT NULL,
->     PRIMARY KEY (Expense_id),
->     CONSTRAINT fk_tripExpenses_trip FOREIGN KEY (Trip_id) REFERENCES trip(Trip_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.07 sec)

MySQL> CREATE TABLE expenses (
->     Expense_id VARCHAR(45) NOT NULL,
->     Note VARCHAR(45),
->     Amount DECIMAL(15, 3) NOT NULL,
->     Category VARCHAR(45) NOT NULL,
->     CONSTRAINT fk_expenses_tripExpenses FOREIGN KEY (Expense_id) REFERENCES tripExpenses(Expense_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.07 sec)

MySQL> CREATE TABLE tripTransportation (
->     Trip_id VARCHAR(45) NOT NULL,
->     Trans_id VARCHAR(45) NOT NULL,
->     PRIMARY KEY (Trans_id),
->     CONSTRAINT fk_tripTransportation_trip FOREIGN KEY (Trip_id) REFERENCES trip(Trip_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );

```

Figure 7: Creating destination, activity, tripExpenses and expenses tables

```

mysql> CREATE TABLE tripExpenses (
->     Expense_id VARCHAR(45) NOT NULL,
->     Trip_id VARCHAR(45) NOT NULL,
->     PRIMARY KEY (Expense_id),
->     CONSTRAINT fk_tripExpenses_trip FOREIGN KEY (Trip_id) REFERENCES trip(Trip_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.07 sec)

mysql> CREATE TABLE expenses (
->     Expense_id VARCHAR(45) NOT NULL,
->     Note VARCHAR(255),
->     Amount DECIMAL(15, 3) NOT NULL,
->     Category VARCHAR(45) NOT NULL,
->     CONSTRAINT fk_expenses_trip FOREIGN KEY (Expense_id) REFERENCES tripExpenses(Expense_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.07 sec)

mysql> CREATE TABLE tripTransportation (
->     Trip_id VARCHAR(45) NOT NULL,
->     Trans_id VARCHAR(45) NOT NULL,
->     PRIMARY KEY (Trans_id),
->     CONSTRAINT fk_tripTransportation_trip FOREIGN KEY (Trip_id) REFERENCES trip(Trip_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.08 sec)

mysql> CREATE TABLE transportation (
->     Trans_id VARCHAR(45) NOT NULL,
->     Arrival_date DATETIME NOT NULL,
->     Departure_date DATETIME NOT NULL,
->     Location VARCHAR(255) NOT NULL,
->     Mode VARCHAR(45) NOT NULL,
->     CONSTRAINT fk_transportation_tripTransportation FOREIGN KEY (Trans_id) REFERENCES tripTransportation(Trans_id) ON DELETE NO ACTION ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.07 sec)

mysql>

```

Figure 8: Creating tripTransportation, transportation table

### 3.3 Insert Data to the Tables

```

mysql> INSERT INTO user (User_id, Guid_id, First_Name, Last_Name, Gender, Email, Profile_picture) VALUES
-> ('user001', 'guid001', 'John', 'Doe', 'Male', 'john.doe@example.com', NULL),
-> ('user002', 'guid002', 'Jane', 'Smith', 'Female', 'jane.smith@example.com', NULL),
-> ('user003', 'guid003', 'Bob', 'Johnson', 'Male', 'bob.johnson@example.com', NULL),
-> ('user004', 'guid004', 'Alice', 'Williams', 'Female', 'alice.williams@example.com', NULL),
-> ('user005', 'guid005', 'Tom', 'Brown', 'Male', 'tom.brown@example.com', NULL),
-> ('user006', 'guid006', 'Sara', 'Davis', 'Female', 'sara.davis@example.com', NULL);
Query OK, 6 rows affected (0.11 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from user
-> ;
+-----+-----+-----+-----+-----+-----+
| User_id | Guid_id | First_Name | Last_Name | Gender | Email           |
+-----+-----+-----+-----+-----+-----+
| user001 | guid001 | John       | Doe      | Male   | john.doe@example.com |
| user002 | guid002 | Jane       | Smith    | Female | jane.smith@example.com |
| user003 | guid003 | Bob        | Johnson  | Male   | bob.johnson@example.com |
| user004 | guid004 | Alice      | Williams | Female | alice.williams@example.com |
| user005 | guid005 | Tom        | Brown    | Male   | tom.brown@example.com |
| user006 | guid006 | Sara       | Davis    | Female | sara.davis@example.com |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>

```

Figure 9: Inserting data into the user table

```

mysql> INSERT INTO userEmail (Email, Pwd) VALUES
-> ('john.doe@example.com', 'password1'),
-> ('jane.smith@example.com', 'password2'),
-> ('bob.johnson@example.com', 'password3'),
-> ('alice.williams@example.com', 'password4'),
-> ('tom.brown@example.com', 'password5'),
-> ('sara.davis@example.com', 'password6');
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from userEmail;
+-----+-----+
| Email          | Pwd      |
+-----+-----+
| alice.williams@example.com | password4 |
| bob.johnson@example.com    | password3 |
| jane.smith@example.com     | password2 |
| john.doe@example.com       | password1 |
| sara.davis@example.com     | password6 |
| tom.brown@example.com     | password5 |
+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 10: Inserting data into the userEmail

```
+-----+-----+
6 rows in set (0.00 sec)

mysql> INSERT INTO userAge (User_id, DOB, Age) VALUES
-> ('user001', '1990-05-15', 33),
-> ('user002', '1985-11-22', 38),
-> ('user003', '1992-03-08', 31),
-> ('user004', '1988-07-01', 35),
-> ('user005', '1995-12-28', 28),
-> ('user006', '1982-09-10', 41);
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from userAge;
+-----+-----+-----+
| User_id | DOB      | Age   |
+-----+-----+-----+
| user001 | 1990-05-15 | 33   |
| user002 | 1985-11-22 | 38   |
| user003 | 1992-03-08 | 31   |
| user004 | 1988-07-01 | 35   |
| user005 | 1995-12-28 | 28   |
| user006 | 1982-09-10 | 41   |
+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 11: Inserting data into the userAge table

```
+-----+-----+
6 rows in set (0.00 sec)

mysql> INSERT INTO userPhone (User_id, Phone_number) VALUES
-> ('user001', '1234567890'),
-> ('user002', '9876543210'),
-> ('user003', '5555555555'),
-> ('user004', '1112223333'),
-> ('user005', '4445556666'),
-> ('user006', '7778889999');
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from userPhone;
+-----+-----+
| User_id | Phone_number |
+-----+-----+
| user001 | 1234567890 |
| user002 | 9876543210 |
| user003 | 5555555555 |
| user004 | 1112223333 |
| user005 | 4445556666 |
| user006 | 7778889999 |
+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 12: Inserting data into the userPhone table

```
+-----+-----+
6 rows in set (0.00 sec)

mysql> INSERT INTO tripUser (User_id, Trip_id) VALUES
-> ('user001', 'trip001'),
-> ('user002', 'trip002'),
-> ('user003', 'trip003'),
-> ('user004', 'trip004'),
-> ('user005', 'trip005'),
-> ('user006', 'trip006');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from tripUser;
+-----+-----+
| User_id | Trip_id  |
+-----+-----+
| user001 | trip001 |
| user002 | trip002 |
| user003 | trip003 |
| user004 | trip004 |
| user005 | trip005 |
| user006 | trip006 |
+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 13: Inserting data into the tripUser table

```

MySQL 8.0 Command Line Cli × + ▾
+-----+-----+
6 rows in set (0.00 sec)

mysql> INSERT INTO trip (Trip_id, Start_date, End_date, Status, Description) VALUES
-> ('trip001', '2023-06-01', '2023-06-15', 'Planned', 'Summer vacation'),
-> ('trip002', '2023-09-15', '2023-09-22', 'Planned', 'Fall getaway'),
-> ('trip003', '2023-12-20', '2024-01-05', 'Planned', 'Winter holidays'),
-> ('trip004', '2024-03-01', '2024-03-10', 'Planned', 'Spring break'),
-> ('trip005', '2024-07-01', '2024-07-15', 'Planned', 'Summer adventure'),
-> ('trip006', '2024-10-01', '2024-10-08', 'Planned', 'Fall foliage tour');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from trip;
+-----+-----+-----+-----+-----+
| Trip_id | Start_date | End_date | Status | Description |
+-----+-----+-----+-----+-----+
| trip001 | 2023-06-01 | 2023-06-15 | Planned | Summer vacation |
| trip002 | 2023-09-15 | 2023-09-22 | Planned | Fall getaway |
| trip003 | 2023-12-20 | 2024-01-05 | Planned | Winter holidays |
| trip004 | 2024-03-01 | 2024-03-10 | Planned | Spring break |
| trip005 | 2024-07-01 | 2024-07-15 | Planned | Summer adventure |
| trip006 | 2024-10-01 | 2024-10-08 | Planned | Fall foliage tour |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 15: Inserting data into the trip table

```

MySQL 8.0 Command Line Cli × + ▾
+-----+-----+
6 rows in set (0.00 sec)

mysql> INSERT INTO tripAccommodation (Trip_id, Acco_id) VALUES
-> ('trip001', 'acco001'),
-> ('trip002', 'acco002'),
-> ('trip003', 'acco003'),
-> ('trip004', 'acco004'),
-> ('trip005', 'acco005'),
-> ('trip006', 'acco006');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from tripAccommodation;
+-----+-----+
| Trip_id | Acco_id |
+-----+-----+
| trip001 | acco001 |
| trip002 | acco002 |
| trip003 | acco003 |
| trip004 | acco004 |
| trip005 | acco005 |
| trip006 | acco006 |
+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 14: Inserting data into the tripAccommodation

```

MySQL 8.0 Command Line Cli × + ▾
+-----+-----+
6 rows in set (0.00 sec)

mysql> INSERT INTO accommodation (Acco_id, Acco_name, Acco_state, Acco_city) VALUES
-> ('acco001', 'Beach Resort', 'Florida', 'Miami'),
-> ('acco002', 'Mountain Lodge', 'Colorado', 'Aspen'),
-> ('acco003', 'City Hotel', 'New York', 'New York City'),
-> ('acco004', 'Countryside Inn', 'Vermont', 'Stowe'),
-> ('acco005', 'Lake Cabin', 'Oregon', 'Bend'),
-> ('acco006', 'Historic B&B', 'Massachusetts', 'Boston');
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from accommodation;
+-----+-----+-----+-----+
| Acco_id | Acco_name | Acco_state | Acco_city |
+-----+-----+-----+-----+
| acco001 | Beach Resort | Florida | Miami |
| acco002 | Mountain Lodge | Colorado | Aspen |
| acco003 | City Hotel | New York | New York City |
| acco004 | Countryside Inn | Vermont | Stowe |
| acco005 | Lake Cabin | Oregon | Bend |
| acco006 | Historic B&B | Massachusetts | Boston |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 16: Inserting data into the accommodation

```
MySQL 8.0 Command Line Cli  +  ×
+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> INSERT INTO accomoPhone (Acco_id, Phone_number) VALUES
    -> ('acco001', '1234567890'),
    -> ('acco002', '9876543210'),
    -> ('acco003', '5555555555'),
    -> ('acco004', '1112223333'),
    -> ('acco005', '4445556666'),
    -> ('acco006', '7778889999');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from accomoPhone;
+-----+-----+
| Acco_id | Phone_number |
+-----+-----+
| acco001 | 1234567890 |
| acco002 | 9876543210 |
| acco003 | 5555555555 |
| acco004 | 1112223333 |
| acco005 | 4445556666 |
| acco006 | 7778889999 |
+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 17: Inserting data into accomoPhone table

```
MySQL 8.0 Command Line Cli  +  ×
+-----+-----+
6 rows in set (0.00 sec)

mysql> INSERT INTO review (User_id, Acco_id, Comments, Rating) VALUES
    -> ('user001', 'acco001', 'Great beachfront location', '5'),
    -> ('user002', 'acco002', 'Beautiful mountain views', '4'),
    -> ('user003', 'acco003', 'Convenient location in the city', '3'),
    -> ('user004', 'acco004', 'Peaceful and cozy', '5'),
    -> ('user005', 'acco005', 'Lovely cabin by the lake', '4'),
    -> ('user006', 'acco006', 'Charming historic building', '3');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from review;
+-----+-----+-----+-----+
| User_id | Acco_id | Comments           | Rating |
+-----+-----+-----+-----+
| user001 | acco001 | Great beachfront location | 5      |
| user002 | acco002 | Beautiful mountain views | 4      |
| user003 | acco003 | Convenient location in the city | 3      |
| user004 | acco004 | Peaceful and cozy | 5      |
| user005 | acco005 | Lovely cabin by the lake | 4      |
| user006 | acco006 | Charming historic building | 3      |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 18: Inserting data into review table

```
MySQL 8.0 Command Line Cli  +  ×
+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> INSERT INTO tripDestination (Trip_id, Destination_id) VALUES
    -> ('trip001', 'dest001'),
    -> ('trip002', 'dest002'),
    -> ('trip003', 'dest003'),
    -> ('trip004', 'dest004'),
    -> ('trip005', 'dest005'),
    -> ('trip006', 'dest006');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from tripDestination;
+-----+-----+
| Trip_id | Destination_id |
+-----+-----+
| trip001 | dest001   |
| trip002 | dest002   |
| trip003 | dest003   |
| trip004 | dest004   |
| trip005 | dest005   |
| trip006 | dest006   |
+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 19: Inserting data into the tripDestination table

```

MySQL 8.0 Command Line Cli  +  -
+-----+
6 rows in set (0.00 sec)

mysql> INSERT INTO destination (Destination_id, Longitude, Latitude, Destination_name, Description) VALUES
-> ('dest001', '-80.1918', '25.7617', 'Miami Beach', 'Sunny beach destination'),
-> ('dest002', '-106.8246', '39.1911', 'Aspen', 'Ski resort in the Rocky Mountains'),
-> ('dest003', '-73.9865', '40.7306', 'New York City', 'Vibrant city with attractions'),
-> ('dest004', '-72.7317', '44.4759', 'Stowe', 'Quaint New England town'),
-> ('dest005', '-121.3153', '44.0582', 'Bend', 'Outdoor adventure hub'),
-> ('dest006', '-71.0589', '42.3601', 'Boston', 'Historic city with rich culture');
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from destination;
+-----+
| Destination_id | Longitude | Latitude | Destination_name | Description |
+-----+
| dest001        | -80.1918 | 25.7617  | Miami Beach     | Sunny beach destination
| dest002        | -106.8246 | 39.1911  | Aspen            | Ski resort in the Rocky Mountains
| dest003        | -73.9865  | 40.7306  | New York City   | Vibrant city with attractions
| dest004        | -72.7317  | 44.4759  | Stowe           | Quaint New England town
| dest005        | -121.3153 | 44.0582  | Bend             | Outdoor adventure hub
| dest006        | -71.0589  | 42.3601  | Boston           | Historic city with rich culture
+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 20: Inserting data into the destination table

```

MySQL 8.0 Command Line Cli  +  -
+-----+
6 rows in set (0.00 sec)

mysql> INSERT INTO activity (Destination_id, Activity_name, Activity_time) VALUES
-> ('dest001', 'Beach activities', '10:00:00'),
-> ('dest002', 'Skiing', '09:00:00'),
-> ('dest003', 'Museum tour', '13:00:00'),
-> ('dest004', 'Hiking', '11:00:00'),
-> ('dest005', 'Kayaking', '14:00:00'),
-> ('dest006', 'Walking tour', '15:00:00');
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from activity;
+-----+
| Destination_id | Activity_name | Activity_time |
+-----+
| dest001        | Beach activities | 10:00:00
| dest002        | Skiing          | 09:00:00
| dest003        | Museum tour     | 13:00:00
| dest004        | Hiking           | 11:00:00
| dest005        | Kayaking         | 14:00:00
| dest006        | Walking tour    | 15:00:00
+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 21: Inserting data into the activity table

```

MySQL 8.0 Command Line Cli  +  -
+-----+
6 rows in set (0.00 sec)

mysql> INSERT INTO tripExpenses (Expense_id, Trip_id) VALUES
-> ('exp001', 'trip001'),
-> ('exp002', 'trip002'),
-> ('exp003', 'trip003'),
-> ('exp004', 'trip004'),
-> ('exp005', 'trip005'),
-> ('exp006', 'trip006');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from tripExpenses;
+-----+
| Expense_id | Trip_id |
+-----+
| exp001     | trip001
| exp002     | trip002
| exp003     | trip003
| exp004     | trip004
| exp005     | trip005
| exp006     | trip006
+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 22: Inserting data into the tripExpenses table

```

MySQL 8.0 Command Line Cli  + 
6 rows in set (0.00 sec)

mysql> INSERT INTO expenses (Expense_id, Note, Amount, Category) VALUES
-> ('exp001', 'Hotel stay', 1000.00, 'Accommodation'),
-> ('exp002', 'Rental car', 500.00, 'Transportation'),
-> ('exp003', 'Broadway show tickets', 200.00, 'Entertainment'),
-> ('exp004', 'Hiking gear rental', 75.00, 'Activities'),
-> ('exp005', 'Kayak rental', 50.00, 'Activities'),
-> ('exp006', 'Tour guide fee', 100.00, 'Activities');
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql>
mysql> select * from expenses;
+-----+-----+-----+
| Expense_id | Note           | Amount | Category      |
+-----+-----+-----+
| exp001    | Hotel stay     | 1000.00 | Accommodation |
| exp002    | Rental car      | 500.00  | Transportation |
| exp003    | Broadway show tickets | 200.00 | Entertainment |
| exp004    | Hiking gear rental | 75.00  | Activities     |
| exp005    | Kayak rental    | 50.00   | Activities     |
| exp006    | Tour guide fee  | 100.00  | Activities     |
+-----+-----+-----+
6 rows in set (0.00 sec)

```

Figure 23: Inserting data into the expenses table

```

MySQL 8.0 Command Line Cli  + 
6 rows in set (0.00 sec)

mysql> INSERT INTO tripTransportation (Trip_id, Trans_id) VALUES
-> ('trip001', 'trans001'),
-> ('trip002', 'trans002'),
-> ('trip003', 'trans003'),
-> ('trip004', 'trans004'),
-> ('trip005', 'trans005'),
-> ('trip006', 'trans006');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from tripTransportation;
+-----+-----+
| Trip_id | Trans_id |
+-----+-----+
| trip001 | trans001 |
| trip002 | trans002 |
| trip003 | trans003 |
| trip004 | trans004 |
| trip005 | trans005 |
| trip006 | trans006 |
+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 24: Inserting data into the tripTransportation table

```

MySQL 8.0 Command Line Cli  + 
6 rows in set (0.00 sec)

mysql> INSERT INTO transportation (Trans_id, Arrival_date, Departure_date, Location, Mode) VALUES
-> ('trans001', '2023-06-01 10:00:00', '2023-06-15 16:00:00', 'Miami, FL', 'Airplane'),
-> ('trans002', '2023-09-15 08:00:00', '2023-09-22 20:00:00', 'Aspen, CO', 'Car'),
-> ('trans003', '2023-12-20 12:00:00', '2024-01-05 18:00:00', 'New York City, NY', 'Train'),
-> ('trans004', '2024-03-01 09:00:00', '2024-03-10 15:00:00', 'Stowe, VT', 'Car'),
-> ('trans005', '2024-07-01 11:00:00', '2024-07-15 17:00:00', 'Bend, OR', 'Airplane'),
-> ('trans006', '2024-10-01 14:00:00', '2024-10-08 11:00:00', 'Boston, MA', 'Bus');
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from transportation;
+-----+-----+-----+-----+
| Trans_id | Arrival_date | Departure_date | Location      | Mode       |
+-----+-----+-----+-----+
| trans001 | 2023-06-01 10:00:00 | 2023-06-15 16:00:00 | Miami, FL | Airplane |
| trans002 | 2023-09-15 08:00:00 | 2023-09-22 20:00:00 | Aspen, CO | Car       |
| trans003 | 2023-12-20 12:00:00 | 2024-01-05 18:00:00 | New York City, NY | Train     |
| trans004 | 2024-03-01 09:00:00 | 2024-03-10 15:00:00 | Stowe, VT | Car       |
| trans005 | 2024-07-01 11:00:00 | 2024-07-15 17:00:00 | Bend, OR | Airplane |
| trans006 | 2024-10-01 14:00:00 | 2024-10-08 11:00:00 | Boston, MA | Bus       |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

```

Figure 25: Inserting data into the transportation table

### 3.4 Update data in the tables

The screenshot shows a MySQL command-line interface window. It starts with a 'select \* from user' query, displaying six rows of user data. Then, two UPDATE statements are executed: one changing the first name of user001 to 'Johnny' and another changing the email of user002. Both queries return 'Query OK'. Finally, a 'select \* from user' command is run again, showing the updated data where user001's first name is now 'Johnny' and user002's email is now 'jane.doe@example.com'.

```

MySQL> select * from user;
+-----+-----+-----+-----+-----+-----+
| User_id | Guid_id | First_Name | Last_Name | Gender | Email           | Profile_picture |
+-----+-----+-----+-----+-----+-----+
| user001 | guid001 | John      | Doe       | Male   | john.doe@example.com | NULL             |
| user002 | guid002 | Jane      | Smith     | Female  | jane.smith@example.com | NULL             |
| user003 | guid003 | Bob       | Johnson   | Male   | bob.johnson@example.com | NULL             |
| user004 | guid004 | Alice     | Williams  | Female  | alice.williams@example.com | NULL             |
| user005 | guid005 | Tom       | Brown    | Male   | tom.brown@example.com | NULL             |
| user006 | guid006 | Sara      | Davis    | Female  | sara.davis@example.com | NULL             |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.03 sec)

MySQL> UPDATE user SET First_Name = 'Johnny', Last_Name = 'Appleseed' WHERE User_id = 'user001';
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MySQL> UPDATE user SET Email = 'jane.doe@example.com' WHERE User_id = 'user002';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MySQL> select * from user;
+-----+-----+-----+-----+-----+-----+
| User_id | Guid_id | First_Name | Last_Name | Gender | Email           | Profile_picture |
+-----+-----+-----+-----+-----+-----+
| user001 | guid001 | Johnny    | Appleseed | Male   | john.doe@example.com | NULL             |
| user002 | guid002 | Jane      | Smith     | Female  | jane.doe@example.com | NULL             |
| user003 | guid003 | Bob       | Johnson   | Male   | bob.johnson@example.com | NULL             |
| user004 | guid004 | Alice     | Williams  | Female  | alice.williams@example.com | NULL             |
| user005 | guid005 | Tom       | Brown    | Male   | tom.brown@example.com | NULL             |
| user006 | guid006 | Sara      | Davis    | Female  | sara.davis@example.com | NULL             |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

MySQL>

```

Figure 26: Before and After updating the user table

The screenshot shows a MySQL command-line interface window. It starts with a 'select \* from userEmail' query, displaying six rows of user email data. Then, two UPDATE statements are executed: one changing the password of user with email 'john.doe@example.com' to 'newpassword1' and another changing the password of user with email 'jane.smith@example.com' to 'newpassword2'. Both queries return 'Query OK'. Finally, a 'select \* from userEmail' command is run again, showing the updated data where the specified users' passwords have been changed.

```

MySQL> select * from userEmail;
+-----+-----+
| Email          | Pwd   |
+-----+-----+
| alice.williams@example.com | password4 |
| bob.johnson@example.com   | password3 |
| jane.doe@example.com      | password2 |
| john.doe@example.com      | password1 |
| sara.davis@example.com    | password6 |
| tom.brown@example.com    | password5 |
+-----+-----+
6 rows in set (0.01 sec)

MySQL> UPDATE userEmail SET Pwd = 'newpassword1' WHERE Email = 'john.doe@example.com';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MySQL> UPDATE userEmail SET Pwd = 'newpassword2' WHERE Email = 'jane.smith@example.com';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0

MySQL> select * from userEmail;
+-----+-----+
| Email          | Pwd   |
+-----+-----+
| alice.williams@example.com | password4 |
| bob.johnson@example.com   | password3 |
| jane.doe@example.com      | password2 |
| john.doe@example.com      | newpassword1 |
| sara.davis@example.com    | password6 |
| tom.brown@example.com    | password5 |
+-----+-----+
6 rows in set (0.00 sec)

MySQL>

```

Figure 27: Before and After updating the userEmail table

The screenshot shows a MySQL command-line interface window. It starts with a 'select \* from userAge' query, displaying six rows of user age data. Then, two UPDATE statements are executed: one changing the age of user001 to 34 and another changing the DOB of user002 to '1984-06-15'. Both queries return 'Query OK'. Finally, a 'select \* from userAge' command is run again, showing the updated data where user001's age is now 34 and user002's DOB is now '1984-06-15'.

```

MySQL> select * from userAge;
+-----+-----+
| User_id | DOB        | Age   |
+-----+-----+
| user001 | 1990-05-15 | 33   |
| user002 | 1985-11-22 | 38   |
| user003 | 1992-03-08 | 1    |
| user004 | 1988-07-01 | 35   |
| user005 | 1995-12-28 | 28   |
| user006 | 1982-09-10 | 41   |
+-----+-----+
6 rows in set (0.01 sec)

MySQL> UPDATE userAge SET Age = 34 WHERE User_id = 'user001';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MySQL> UPDATE userAge SET DOB = '1984-06-15' WHERE User_id = 'user002';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MySQL> select * from userAge;
+-----+-----+
| User_id | DOB        | Age   |
+-----+-----+
| user001 | 1990-05-15 | 34   |
| user002 | 1984-06-15 | 38   |
| user003 | 1992-03-08 | 1    |
| user004 | 1988-07-01 | 35   |
| user005 | 1995-12-28 | 28   |
| user006 | 1982-09-10 | 41   |
+-----+-----+
6 rows in set (0.00 sec)

MySQL>

```

Figure 28: Before and After updating the userAge table

```

mysql> select * from userPhone;
+-----+-----+
| User_id | Phone_number |
+-----+-----+
| user001 | 1234567890 |
| user002 | 9876543210 |
| user003 | 5555555555 |
| user004 | 1112223333 |
| user005 | 4445556666 |
| user006 | 7778889999 |
+-----+-----+
6 rows in set (0.01 sec)

mysql> UPDATE userPhone SET Phone_number = '9998887777' WHERE User_id = 'user001';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE userPhone SET Phone_number = '6667778888' WHERE User_id = 'user002';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from userPhone;
+-----+-----+
| User_id | Phone_number |
+-----+-----+
| user001 | 9998887777 |
| user002 | 6667778888 |
| user003 | 5555555555 |
| user004 | 1112223333 |
| user005 | 4445556666 |
| user006 | 7778889999 |
+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 29: Before and After updating the userPhone table

```

mysql> select * from trip;
+-----+-----+-----+-----+-----+
| Trip_id | Start_date | End_date | Status | Description |
+-----+-----+-----+-----+-----+
| trip001 | 2023-06-01 | 2023-06-15 | Planned | Summer vacation |
| trip002 | 2023-09-15 | 2023-09-22 | Planned | Fall getaways |
| trip003 | 2023-12-20 | 2024-01-05 | Planned | Winter holidays |
| trip004 | 2024-03-01 | 2024-03-10 | Planned | Spring break |
| trip005 | 2024-07-01 | 2024-07-15 | Planned | Summer adventure |
| trip006 | 2024-10-01 | 2024-10-08 | Planned | Fall foliage tour |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> UPDATE trip SET Status = 'In Progress' WHERE Trip_id = 'trip001';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE trip SET Description = 'Family vacation' WHERE Trip_id = 'trip002';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from trip;
+-----+-----+-----+-----+-----+
| Trip_id | Start_date | End_date | Status | Description |
+-----+-----+-----+-----+-----+
| trip001 | 2023-06-01 | 2023-06-15 | In Progress | Summer vacation |
| trip002 | 2023-09-15 | 2023-09-22 | Planned | Family vacation |
| trip003 | 2023-12-20 | 2024-01-05 | Planned | Winter holidays |
| trip004 | 2024-03-01 | 2024-03-10 | Planned | Spring break |
| trip005 | 2024-07-01 | 2024-07-15 | Planned | Summer adventure |
| trip006 | 2024-10-01 | 2024-10-08 | Planned | Fall foliage tour |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 30: Before and After updating the trip table

```

mysql> select * from accommodation;
+-----+-----+-----+-----+
| Acco_id | Acco_name | Acco_state | Acco_city |
+-----+-----+-----+-----+
| acco001 | Beach Resort | Florida | Miami |
| acco002 | Mountain Lodge | Colorado | Aspen |
| acco003 | City Hotel | New York | New York City |
| acco004 | Countryside Inn | Vermont | Stowe |
| acco005 | Lake Cabin | Oregon | Bend |
| acco006 | Historic B&B | Massachusetts | Boston |
+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> UPDATE accommodation SET Acco_city = 'Tampa' WHERE Acco_id = 'acco001';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE accommodation SET Acco_name = 'Mountain Resort' WHERE Acco_id = 'acco002';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from accommodation;
+-----+-----+-----+-----+
| Acco_id | Acco_name | Acco_state | Acco_city |
+-----+-----+-----+-----+
| acco001 | Beach Resort | Florida | Tampa |
| acco002 | Mountain Resort | Colorado | Aspen |
| acco003 | City Hotel | New York | New York City |
| acco004 | Countryside Inn | Vermont | Stowe |
| acco005 | Lake Cabin | Oregon | Bend |
| acco006 | Historic B&B | Massachusetts | Boston |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 31: Before and After updating the accommodation table

```

MySQL 8.0 Command Line Cli + 
| acco006 | Historic B&B | Massachusetts | Boston |
6 rows in set (0.00 sec)

mysql> select * from accomoPhone;
+-----+-----+
| Acco_id | Phone_number |
+-----+-----+
| acco001 | 1234567890 |
| acco002 | 9876543210 |
| acco003 | 123455555555 |
| acco004 | 1112223333 |
| acco005 | 444455566666 |
| acco006 | 777788899999 |
+-----+-----+
6 rows in set (0.03 sec)

mysql> UPDATE accomoPhone SET Phone_number = '5555551234' WHERE Acco_id = 'acco001';
Query OK, 1 row affected (0.04 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE accomoPhone SET Phone_number = '9876541230' WHERE Acco_id = 'acco002';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from accomoPhone;
+-----+-----+
| Acco_id | Phone_number |
+-----+-----+
| acco001 | 5555551234 |
| acco002 | 9876541230 |
| acco003 | 5555555555 |
| acco004 | 1112223333 |
| acco005 | 444455566666 |
| acco006 | 777788899999 |
+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 32: Before and After updating the accomoPhone table

```

MySQL 8.0 Command Line Cli + 
| acco006 | 777888999999 |
6 rows in set (0.00 sec)

mysql> select * from review;
+-----+-----+-----+-----+
| User_id | Acco_id | Comments | Rating |
+-----+-----+-----+-----+
| user001 | acco001 | Great beachfront location | 5 |
| user002 | acco002 | Beautiful mountain views | 4 |
| user003 | acco003 | Convenient location in the city | 5 |
| user004 | acco004 | Peaceful and cozy | 5 |
| user005 | acco005 | Lovely cabin by the lake | 4 |
| user006 | acco006 | Charming historic building | 3 |
+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> UPDATE review SET Comments = 'Excellent service' WHERE User_id = 'user001' AND Acco_id = 'acco001';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE review SET Rating = '4' WHERE User_id = 'user002' AND Acco_id = 'acco002';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1 Changed: 0 Warnings: 0

mysql> select * from review;
+-----+-----+-----+-----+
| User_id | Acco_id | Comments | Rating |
+-----+-----+-----+-----+
| user001 | acco001 | Excellent service | 5 |
| user002 | acco002 | Beautiful mountain views | 4 |
| user003 | acco003 | Convenient location in the city | 5 |
| user004 | acco004 | Peaceful and cozy | 5 |
| user005 | acco005 | Lovely cabin by the lake | 4 |
| user006 | acco006 | Charming historic building | 3 |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 33: Before and After updating the review table

```

MySQL 8.0 Command Line Cli + 
6 rows in set (0.00 sec)

mysql> select * from destination;
+-----+-----+-----+-----+
| Destination_id | Longitude | Latitude | Destination_name | Description |
+-----+-----+-----+-----+
| dest001 | -88.1918 | 25.7617 | Miami Beach | Sunny beach destination |
| dest002 | -106.8246 | 39.1911 | Aspen | Ski resort in the Rocky Mountains |
| dest003 | -73.9865 | 40.7388 | New York City | Vibrant city with attractions |
| dest004 | -72.0007 | 40.6769 | Stowe | Quaint New England town |
| dest005 | -121.3153 | 40.0582 | Bend | Outdoor adventure hub |
| dest006 | -71.0589 | 42.3601 | Boston | Historic city with rich culture |
+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> UPDATE destination SET Destination_name = 'South Beach' WHERE Destination_id = 'dest001';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE destination SET Description = 'World-class ski resort' WHERE Destination_id = 'dest002';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from destination;
+-----+-----+-----+-----+
| Destination_id | Longitude | Latitude | Destination_name | Description |
+-----+-----+-----+-----+
| dest001 | -88.1918 | 25.7617 | South Beach | Sunny beach destination |
| dest002 | -106.8246 | 39.1911 | Aspen | World-class ski resort |
| dest003 | -73.9865 | 40.7388 | New York City | Vibrant city with attractions |
| dest004 | -72.0007 | 40.6769 | Stowe | Quaint New England town |
| dest005 | -121.3153 | 40.0582 | Bend | Outdoor adventure hub |
| dest006 | -71.0589 | 42.3601 | Boston | Historic city with rich culture |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 34: Before and After updating the destination table

MySQL 8.0 Command Line Cli

```

mysql> select * from activity;
+-----+-----+-----+
| Destination_id | Activity_name | Activity_time |
+-----+-----+-----+
| dest001 | Beach activities | 10:00:00 |
| dest002 | Skiing | 09:00:00 |
| dest003 | Museum tour | 13:00:00 |
| dest004 | Hiking | 11:00:00 |
| dest005 | Kayaking | 14:00:00 |
| dest006 | Walking tour | 15:00:00 |
+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> UPDATE activity SET Activity_name = 'Surfing lessons' WHERE Destination_id = 'dest001';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE activity SET Activity_time = '10:00:00' WHERE Destination_id = 'dest002';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from activity;
+-----+-----+-----+
| Destination_id | Activity_name | Activity_time |
+-----+-----+-----+
| dest001 | Surfing lessons | 10:00:00 |
| dest002 | Skiing | 10:00:00 |
| dest003 | Museum tour | 13:00:00 |
| dest004 | Hiking | 11:00:00 |
| dest005 | Kayaking | 14:00:00 |
| dest006 | Walking tour | 15:00:00 |
+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 35: Before and After updating the activity table

MySQL 8.0 Command Line Cli

```

mysql> select * from expenses;
+-----+-----+-----+-----+
| Expense_id | Note | Amount | Category |
+-----+-----+-----+-----+
| exp001 | Hotel stay | 1000.000 | Accommodation |
| exp002 | Rental car | 500.000 | Transportation |
| exp003 | Broadway show tickets | 200.000 | Entertainment |
| exp004 | Hiking gear rental | 75.000 | Activities |
| exp005 | Kayak rental | 50.000 | Activities |
| exp006 | Tour guide fee | 100.000 | Activities |
+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> UPDATE expenses SET Amount = 1200.00 WHERE Expense_id = 'exp001';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE expenses SET Category = 'Food' WHERE Expense_id = 'exp002';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from expenses;
+-----+-----+-----+-----+
| Expense_id | Note | Amount | Category |
+-----+-----+-----+-----+
| exp001 | Hotel stay | 1200.000 | Accommodation |
| exp002 | Rental car | 500.000 | Food |
| exp003 | Broadway show tickets | 200.000 | Entertainment |
| exp004 | Hiking gear rental | 75.000 | Activities |
| exp005 | Kayak rental | 50.000 | Activities |
| exp006 | Tour guide fee | 100.000 | Activities |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 36: Before and After updating the accommodation table

MySQL 8.0 Command Line Cli

```

mysql> select * from transportation;
+-----+-----+-----+-----+-----+
| Trans_id | Arrival_date | Departure_date | Location | Mode |
+-----+-----+-----+-----+-----+
| trans001 | 2023-06-01 10:00:00 | 2023-06-15 16:00:00 | Miami, FL | Airplane |
| trans002 | 2023-09-15 08:00:00 | 2023-09-22 20:00:00 | Aspen, CO | Car |
| trans003 | 2024-01-01 12:00:00 | 2024-01-08 08:00:00 | New York City, NY | Train |
| trans004 | 2024-03-01 09:00:00 | 2024-03-10 15:00:00 | Stowe, VT | Car |
| trans005 | 2024-07-01 11:00:00 | 2024-07-15 17:00:00 | Bend, OR | Airplane |
| trans006 | 2024-10-01 14:00:00 | 2024-10-08 11:00:00 | Boston, MA | Bus |
+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> UPDATE transportation SET Mode = 'Cruise' WHERE Trans_id = 'trans001';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE transportation SET Location = 'Denver, CO' WHERE Trans_id = 'trans002';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

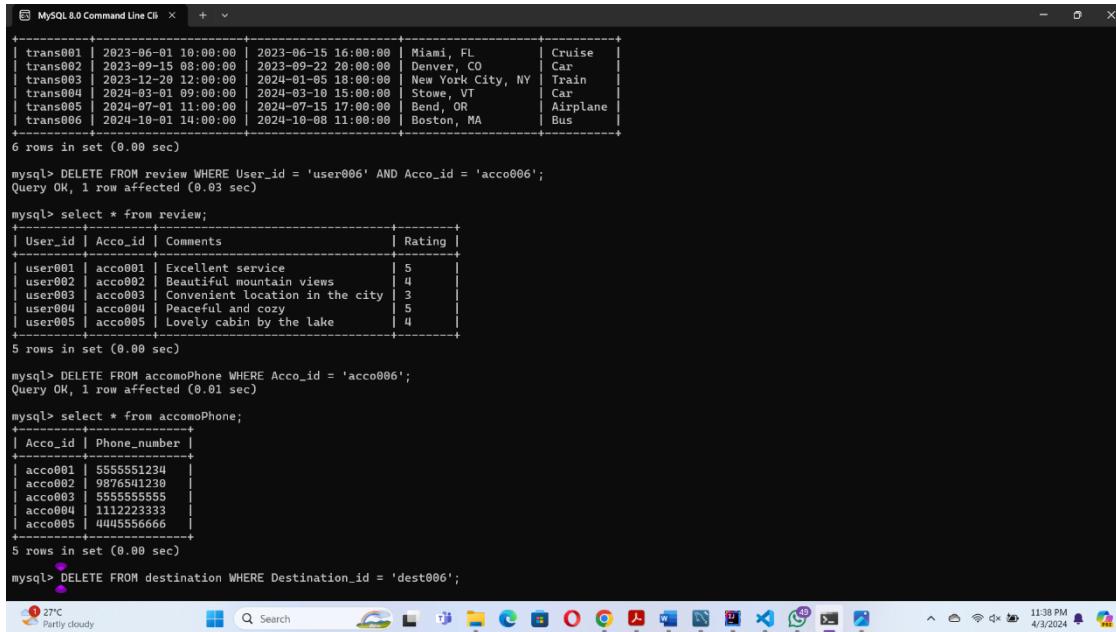
mysql> select * from transportation;
+-----+-----+-----+-----+-----+
| Trans_id | Arrival_date | Departure_date | Location | Mode |
+-----+-----+-----+-----+-----+
| trans001 | 2023-06-01 10:00:00 | 2023-06-15 16:00:00 | Miami, FL | Cruise |
| trans002 | 2023-09-15 08:00:00 | 2023-09-22 20:00:00 | Denver, CO | Car |
| trans003 | 2023-12-20 12:00:00 | 2024-01-05 18:00:00 | New York City, NY | Train |
| trans004 | 2024-03-01 09:00:00 | 2024-03-10 15:00:00 | Stowe, VT | Car |
| trans005 | 2024-07-01 11:00:00 | 2024-07-15 17:00:00 | Bend, OR | Airplane |
| trans006 | 2024-10-01 14:00:00 | 2024-10-08 11:00:00 | Boston, MA | Bus |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 37: Before and After updating the transportation table

### 3.5 Delete data in the tables



The screenshot shows the MySQL 8.0 Command Line Client interface. The command line displays the following SQL queries and their results:

```

mysql> select * from review;
+-----+-----+-----+-----+
| trans_id | start_date | end_date | location | mode |
+-----+-----+-----+-----+
| trans001 | 2023-06-01 10:00:00 | 2023-06-15 16:00:00 | Miami, FL | Cruise |
| trans002 | 2023-09-15 08:00:00 | 2023-09-22 20:00:00 | Denver, CO | Car |
| trans003 | 2023-12-20 12:00:00 | 2024-01-05 18:00:00 | New York City, NY | Train |
| trans004 | 2024-03-01 09:00:00 | 2024-03-10 15:00:00 | Stowe, VT | Car |
| trans005 | 2024-07-01 11:00:00 | 2024-07-15 17:00:00 | Bend, OR | Airplane |
| trans006 | 2024-10-01 14:00:00 | 2024-10-08 11:00:00 | Boston, MA | Bus |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> DELETE FROM review WHERE User_id = 'user006' AND Acco_id = 'acco006';
Query OK, 1 row affected (0.03 sec)

mysql> select * from review;
+-----+-----+-----+-----+
| User_id | Acco_id | Comments | Rating |
+-----+-----+-----+-----+
| user001 | acco001 | Excellent service | 5 |
| user002 | acco002 | Beautiful mountain views | 4 |
| user003 | acco003 | Convenient location in the city | 3 |
| user004 | acco004 | Peaceful and cozy | 5 |
| user005 | acco005 | Lovely cabin by the lake | 4 |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

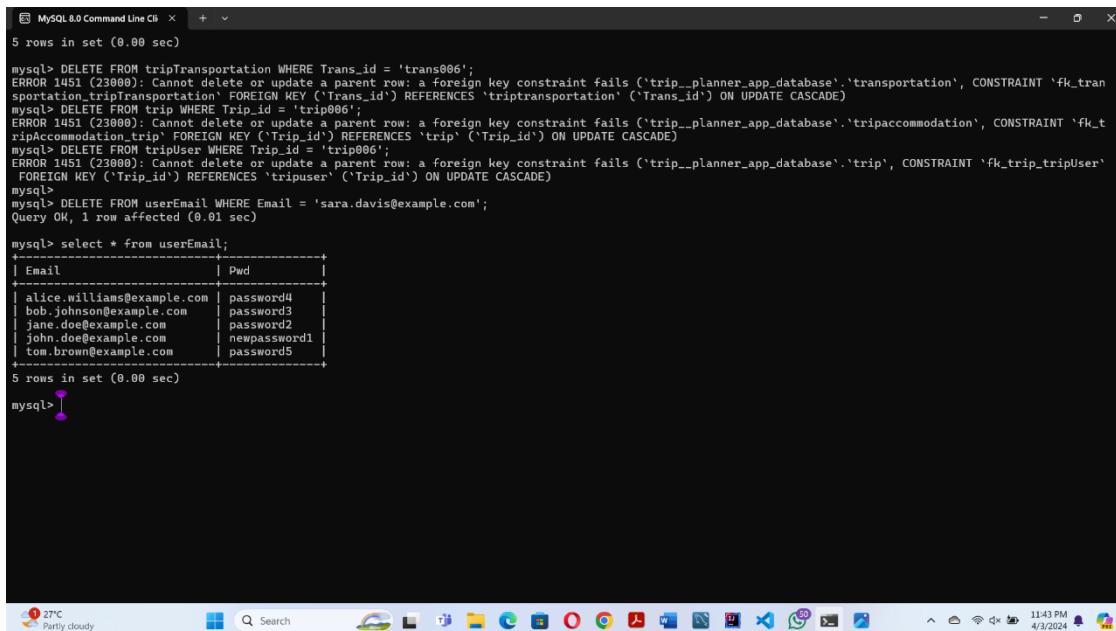
mysql> DELETE FROM accomoPhone WHERE Acco_id = 'acco006';
Query OK, 1 row affected (0.01 sec)

mysql> select * from accomoPhone;
+-----+-----+
| Acco_id | Phone_number |
+-----+-----+
| acco001 | 55555551234 |
| acco002 | 9876541230 |
| acco003 | 5555555555 |
| acco004 | 1112223333 |
| acco005 | 4445556666 |
+-----+-----+
5 rows in set (0.00 sec)

mysql> DELETE FROM destination WHERE Destination_id = 'dest006';
Query OK, 1 row affected (0.01 sec)

```

Figure 38: Deleting data from review and accomoPhone table



The screenshot shows the MySQL 8.0 Command Line Client interface. The command line displays the following SQL queries and their results:

```

mysql> select * from userEmail;
+-----+-----+
| Email | Pwd |
+-----+-----+
| alice.williams@example.com | password4 |
| bob.johnson@example.com | password3 |
| jane.doe@example.com | password2 |
| john.doe@example.com | newpassword1 |
| tom.brown@example.com | password5 |
+-----+-----+
5 rows in set (0.00 sec)

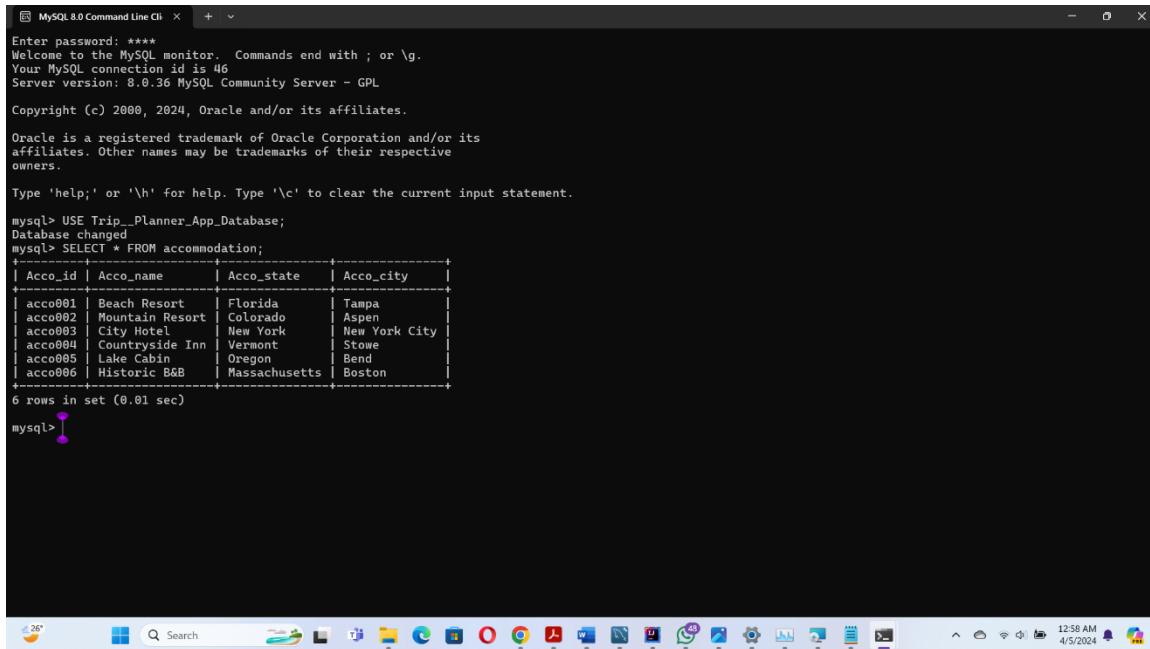
mysql>

```

Figure 39: Deleting data from userEmail table

## 4 Chapter 04 – Transactions

### 4.1 Simple Queries



MySQL 8.0 Command Line Cli X + -

```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 46
Server version: 8.0.36 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

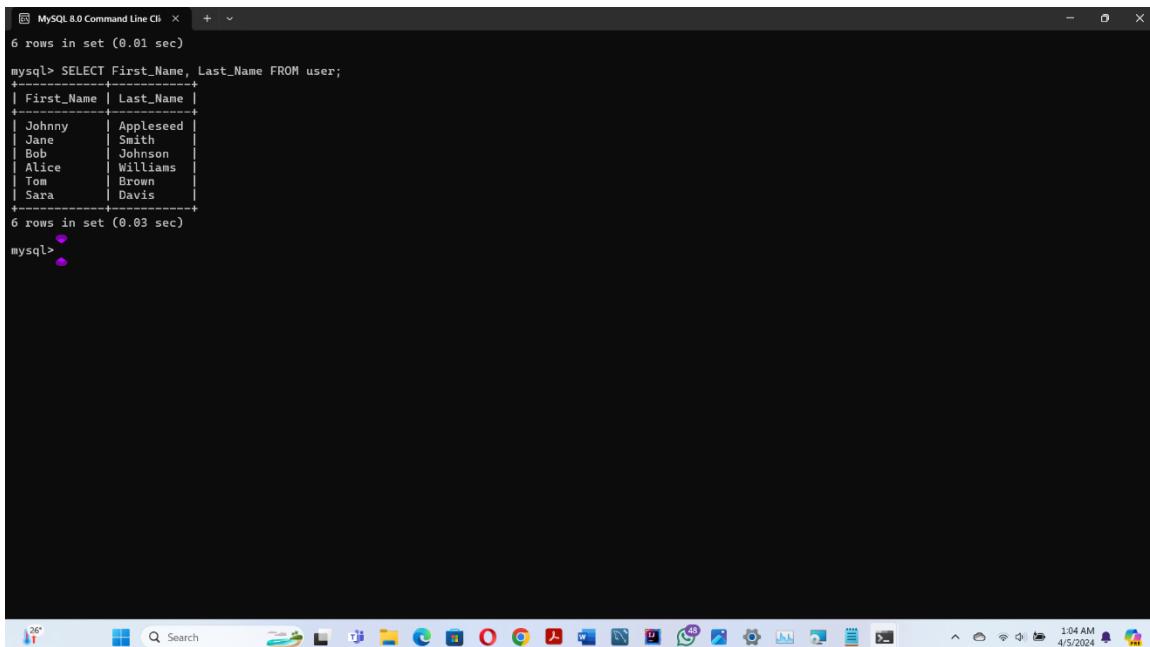
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE Trip_Planner_App_Database;
Database changed
mysql> SELECT * FROM accommodation;
+-----+-----+-----+
| Acco_id | Acco_name      | Acco_state   | Acco_city    |
+-----+-----+-----+
| acco001 | Beach Resort    | Florida      | Tampa        |
| acco002 | Mountain Resort | Colorado     | Aspen         |
| acco003 | City Hotel      | New York    | New York City|
| acco004 | Countryside Inn | Vermont     | Stowe        |
| acco005 | Lake Cabin      | Oregon      | Bend          |
| acco006 | Historic B&B   | Massachusetts | Boston        |
+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>
```

The screenshot shows the MySQL 8.0 Command Line Client window. It displays a simple SELECT query on the 'accommodation' table. The output shows six rows of data with columns: Acco\_id, Acco\_name, Acco\_state, and Acco\_city. The data includes various types of accommodations like Beach Resorts, Mountain Resorts, and City Hotels located in states like Florida, Colorado, New York, Vermont, Oregon, and Massachusetts.

Figure 40: Simple Query demonstrating Select Operation



MySQL 8.0 Command Line Cli X + -

```
6 rows in set (0.01 sec)

mysql> SELECT First_Name, Last_Name FROM user;
+-----+-----+
| First_Name | Last_Name |
+-----+-----+
| Johnny     | Appleseed |
| Jane       | Smith      |
| Bob        | Johnson    |
| Alice      | Williams   |
| Tom        | Brown      |
| Sara       | Davis      |
+-----+-----+
6 rows in set (0.03 sec)

mysql>
```

The screenshot shows the MySQL 8.0 Command Line Client window. It displays a simple SELECT query on the 'user' table. The output shows six rows of data with columns: First\_Name and Last\_Name. The data includes names like Johnny Appleseed, Jane Smith, Bob Johnson, Alice Williams, Tom Brown, and Sara Davis.

Figure 41: Simple Query demonstrating Select Operation

MySQL 8.0 Command Line Cli

```
mysql> SELECT u.First_Name, a.Acco_name FROM user u, accommodation a;
```

First_Name	Acco_name
Sara	Beach Resort
Tom	Beach Resort
Alice	Beach Resort
Bob	Beach Resort
Jane	Beach Resort
Johnny	Beach Resort
Sara	Mountain Resort
Tom	Mountain Resort
Alice	Mountain Resort
Bob	Mountain Resort
Jane	Mountain Resort
Johnny	Mountain Resort
Sara	City Hotel
Tom	City Hotel
Alice	City Hotel
Bob	City Hotel
Jane	City Hotel
Johnny	City Hotel
Sara	Countryside Inn
Tom	Countryside Inn
Alice	Countryside Inn
Bob	Countryside Inn
Jane	Countryside Inn
Johnny	Countryside Inn
Sara	Lake Cabin
Tom	Lake Cabin
Alice	Lake Cabin
Bob	Lake Cabin
Jane	Lake Cabin
Johnny	Lake Cabin
Sara	Historic B&B
Tom	Historic B&B
Alice	Historic B&B
Bob	Historic B&B
Jane	Historic B&B
Johnny	Historic B&B

1:05 AM 4/5/2024

Figure 42: Simple Query demonstrating select operation

MySQL 8.0 Command Line Cli

```
mysql> CREATE VIEW user_profile AS SELECT User_id, First_Name, Last_Name, Email FROM user;
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> SELECT * FROM user_profile;
```

User_id	First_Name	Last_Name	Email
user001	Johnny	Applesseed	john.doe@example.com
user002	Jane	Smith	jane.doe@example.com
user003	Bob	Johnson	bob.johnson@example.com
user004	Alice	Williams	alice.williams@example.com
user005	Tom	Brown	tom.brown@example.com
user006	Sara	Davis	sara.davis@example.com

6 rows in set (0.06 sec)

mysql>

1:10 AM 4/5/2024

Figure 43: Simple Query to create a User View

```
MySQL 8.0 Command Line Cli + ~
+-----+
| user004 | Alice      | Williams   | alice.williams@example.com |
| user005 | Tom        | Brown      | tom.brown@example.com       |
| user006 | Sara        | Davis      | sara.davis@example.com     |
+-----+
6 rows in set (0.06 sec)

mysql> SELECT u.First_Name AS FirstName, u.Last_Name AS LastName FROM user u;
+-----+-----+
| FirstName | LastName |
+-----+-----+
| Johnny    | Appleseed |
| Jane      | Smith      |
| Bob       | Johnson    |
| Alice     | Williams   |
| Tom       | Brown      |
| Sara      | Davis      |
+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 44: Simple query to retrieve data from a table (from user table)

```
MySQL 8.0 Command Line Cli + ~
+-----+
| FirstName | LastName |
+-----+-----+
| Johnny    | Appleseed |
| Jane      | Smith      |
| Bob       | Johnson    |
| Alice     | Williams   |
| Tom       | Brown      |
| Sara      | Davis      |
+-----+-----+
6 rows in set (0.00 sec)

mysql> SELECT AVG(Amount) AS AverageExpense FROM expenses;
+-----+
| AverageExpense |
+-----+
| 405.00000000 |
+-----+
1 row in set (0.02 sec)

mysql>
```

Figure 45: Simple Query demonstrating the use of an aggregation function(Average)

```

MySQL 8.0 Command Line Cli + ~
| 405.0000000 |
+-----+
1 row in set (0.02 sec)

mysql> SELECT * FROM user WHERE Email LIKE '%example.com';
+-----+
| User_id | Guid_id | First_Name | Last_Name | Gender | Email
| Profile_picture |
+-----+
| user001 | guid001 | Johnny    | Appleseed | Male   | john.doe@example.com | NULL
| user002 | guid002 | Jane      | Smith     | Female | jane.doe@example.com | NULL
| user003 | guid003 | Bob       | Johnson   | Male   | bob.johnson@example.com | NULL
| user004 | guid004 | Alice     | Williams  | Female | alice.williams@example.com | NULL
| user005 | guid005 | Tom       | Brown    | Male   | tom.brown@example.com | NULL
| user006 | guid006 | Sara      | Davis    | Female | sara.davis@example.com | NULL
+-----+
6 rows in set (0.01 sec)

mysql> ^

```

Figure 46: Simple Query to Demonstrate the use of Like keyword

## 4.2 Complex Queries

```

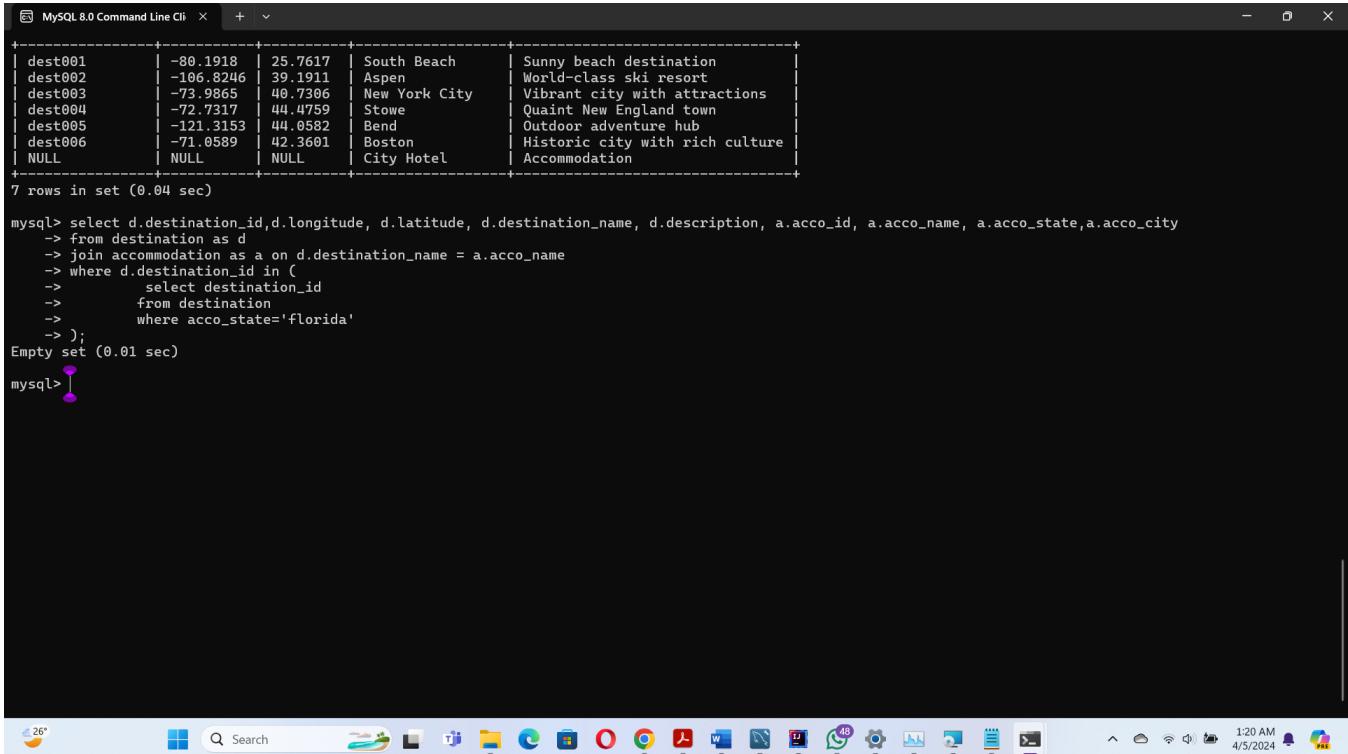
MySQL 8.0 Command Line Cli + ~
| user006 | guid006 | Sara      | Davis    | Female | sara.davis@example.com | NULL |
+-----+
6 rows in set (0.01 sec)

mysql> select destination_id, longitude, latitude,destination_name, description
-> from destination
-> union
-> select null as destination_id, null as longitude, null as latitude, acco_name as destination_name, 'Accommodation' as description
-> from accommodation
-> where acco_city = 'New York City';
+-----+
| destination_id | longitude | latitude | destination_name | description |
+-----+
| dest001        | -80.1918  | 25.7617  | South Beach      | Sunny beach destination
| dest002        | -106.8246 | 39.1911  | Aspen            | World-class ski resort
| dest003        | -73.9865  | 40.7306  | New York City    | Vibrant city with attractions
| dest004        | -72.7317  | 44.4759  | Stowe           | Quaint New England town
| dest005        | -121.3153 | 44.0582  | Bend             | Outdoor adventure hub
| dest006        | -71.0589  | 42.3601  | Boston          | Historic city with rich culture
| NULL           | NULL      | NULL      | City Hotel      | Accommodation
+-----+
7 rows in set (0.04 sec)

mysql> ^

```

Figure 47: Complex Query 1 (Basic set operation Union)



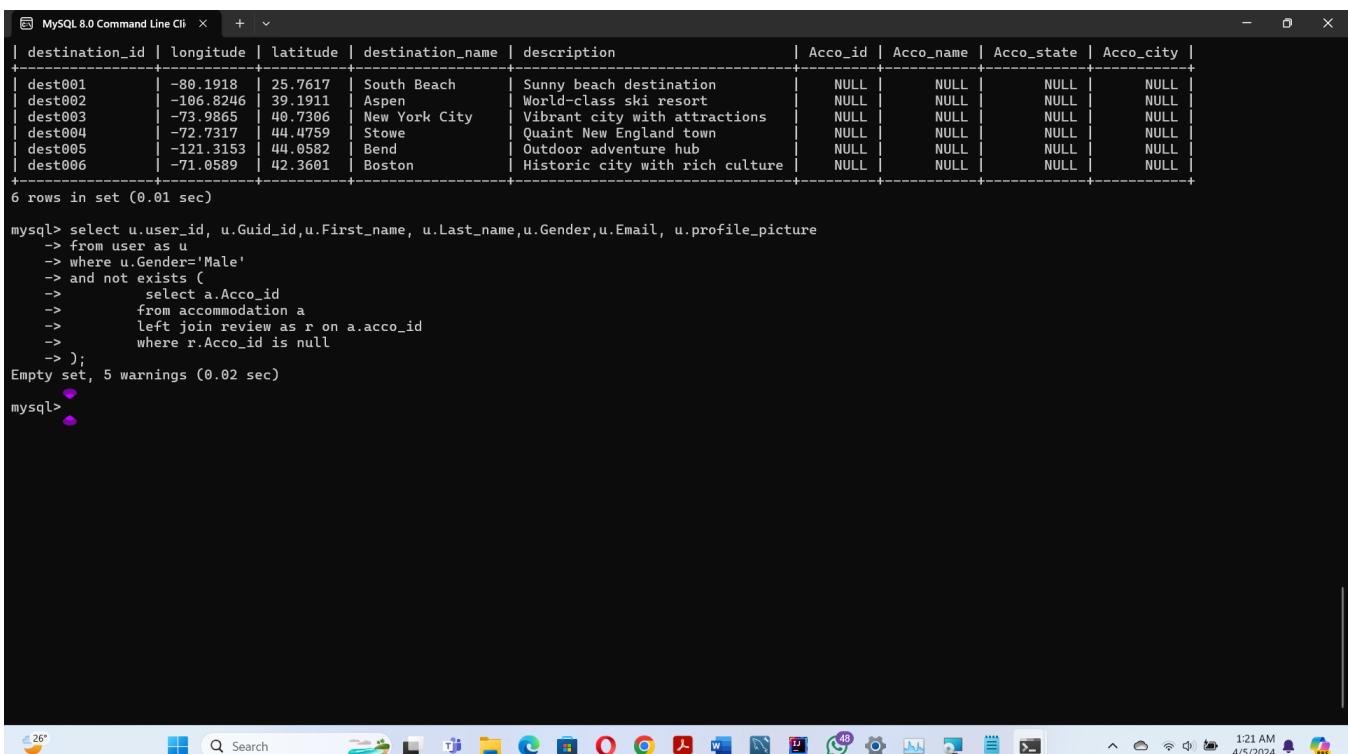
MySQL 8.0 Command Line Cli

```
+-----+-----+-----+-----+-----+
| dest001 | -80.1918 | 25.7617 | South Beach | Sunny beach destination
| dest002 | -106.8246 | 39.1911 | Aspen | World-class ski resort
| dest003 | -73.9865 | 40.7306 | New York City | Vibrant city with attractions
| dest004 | -72.7317 | 44.4759 | Stowe | Quaint New England town
| dest005 | -121.3153 | 44.0582 | Bend | Outdoor adventure hub
| dest006 | -71.0589 | 42.3601 | Boston | Historic city with rich culture
| NULL    | NULL       | NULL     | City Hotel | Accommodation
+-----+-----+-----+-----+-----+
7 rows in set (0.04 sec)

mysql> select d.destination_id,d.longitude, d.latitude, d.destination_name, d.description, a.acco_id, a.acco_name, a.acco_state,a.acco_city
-> from destination as d
-> join accommodation as a on d.destination_name = a.acco_name
-> where d.destination_id in (
->      select destination_id
->      from destination
->      where acco_state='florida'
-> );
Empty set (0.01 sec)

mysql>
```

Figure 48: Complex Query 2 (Basic set operation Intersection)



MySQL 8.0 Command Line Cli

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| destination_id | longitude | latitude | destination_name | description | Acco_id | Acco_name | Acco_state | Acco_city |
+-----+-----+-----+-----+-----+-----+-----+-----+
| dest001 | -80.1918 | 25.7617 | South Beach | Sunny beach destination | NULL | NULL | NULL | NULL |
| dest002 | -106.8246 | 39.1911 | Aspen | World-class ski resort | NULL | NULL | NULL | NULL |
| dest003 | -73.9865 | 40.7306 | New York City | Vibrant city with attractions | NULL | NULL | NULL | NULL |
| dest004 | -72.7317 | 44.4759 | Stowe | Quaint New England town | NULL | NULL | NULL | NULL |
| dest005 | -121.3153 | 44.0582 | Bend | Outdoor adventure hub | NULL | NULL | NULL | NULL |
| dest006 | -71.0589 | 42.3601 | Boston | Historic city with rich culture | NULL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> select u.user_id, u.Guid_id,u.First_name, u.Last_name,u.Gender,u.Email, u.profile_picture
-> from user as u
-> where u.Gender='Male'
-> and not exists (
->      select a.Acco_id
->      from accommodation a
->      left join review as r on a.acco_id
->      where r.Acco_id is null
-> );
Empty set, 5 warnings (0.02 sec)

mysql>
```

Figure 49: Complex Query 3 (Basic set operation division)

```

MySQL 8.0 Command Line Cli  x  +  v
-> where d.destination_id in (
->     select destination_id
->         from destination
->         where acco_state='florida'
-> );
Empty set (0.01 sec)

mysql> select destination_id,longitude,latitude,destination_name,description, null as Acco_id, null as Acco_name, null as Acco_state, null as Acco_city
-> from destination
-> where destination_name not in(
->     select Acco_name
->         from accommodation
->         where Acco_city in('Miami','orlando')
-> );
+-----+-----+-----+-----+-----+-----+
| destination_id | longitude | latitude | destination_name | description | Acco_id | Acco_name | Acco_state | Acco_city |
+-----+-----+-----+-----+-----+-----+
| dest001 | -89.1918 | 25.7617 | South Beach | Sunny beach destination | NULL | NULL | NULL | NULL |
| dest002 | -106.8246 | 39.1911 | Aspen | World-class ski resort | NULL | NULL | NULL | NULL |
| dest003 | -73.9865 | 40.7306 | New York City | Vibrant city with attractions | NULL | NULL | NULL | NULL |
| dest004 | -72.7317 | 44.4759 | Stowe | Quaint New England town | NULL | NULL | NULL | NULL |
| dest005 | -121.3153 | 44.0582 | Bend | Outdoor adventure hub | NULL | NULL | NULL | NULL |
| dest006 | -71.0589 | 42.3601 | Boston | Historic city with rich culture | NULL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>

```

Figure 50: Complex Query 4 (Basic set operation set difference)

```

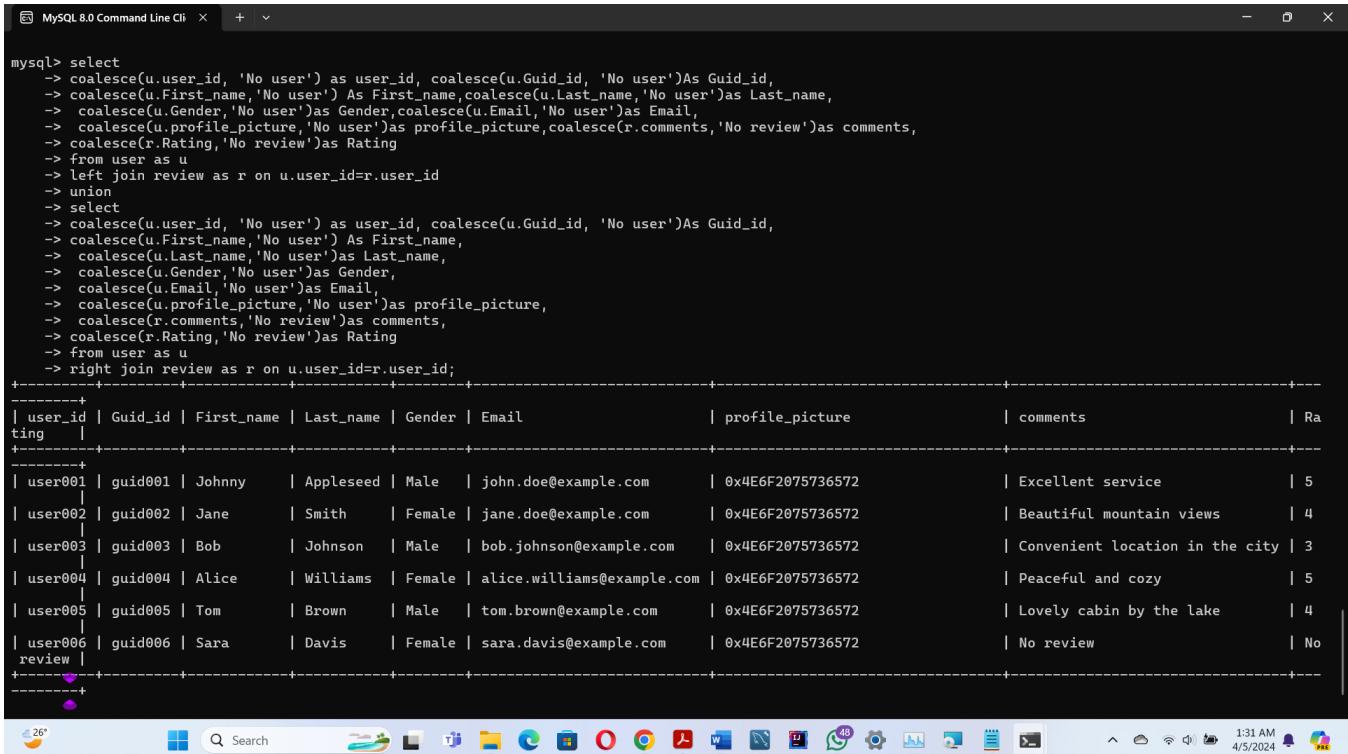
MySQL 8.0 Command Line Cli  x  +  v
Empty set, 5 warnings (0.02 sec)

mysql> select u.user_id, u.guid_id,u.First_name,u.Last_name,u.gender,u.Email, u.profile_picture,tu.trip_id,t.start_date,t.end_date,t.status,t.description
-> from user as u
-> inner join tripuser as tu on u.user_id=tu.user_id
-> inner join trip as t on tu.trip_id=t.trip_id
-> where t.start_date>='2024.01.01' and t.end_date<='2024.12.31';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | guid_id | First_name | Last_name | gender | Email | profile_picture | trip_id | start_date | end_date | status | description |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user004 | guid004 | Alice | Williams | Female | alice.williams@example.com | NULL | trip004 | 2024-03-01 | 2024-03-10 | Planned | Spring break |
| user005 | guid005 | Tom | Brown | Male | tom.brown@example.com | NULL | trip005 | 2024-07-01 | 2024-07-15 | Planned | Summer adventure |
| user006 | guid006 | Sara | Davis | Female | sara.davis@example.com | NULL | trip006 | 2024-10-01 | 2024-10-08 | Planned | Fall foliage tour |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set, 2 warnings (0.02 sec)

mysql>

```

Figure 51: Complex Query 5 (inner join)

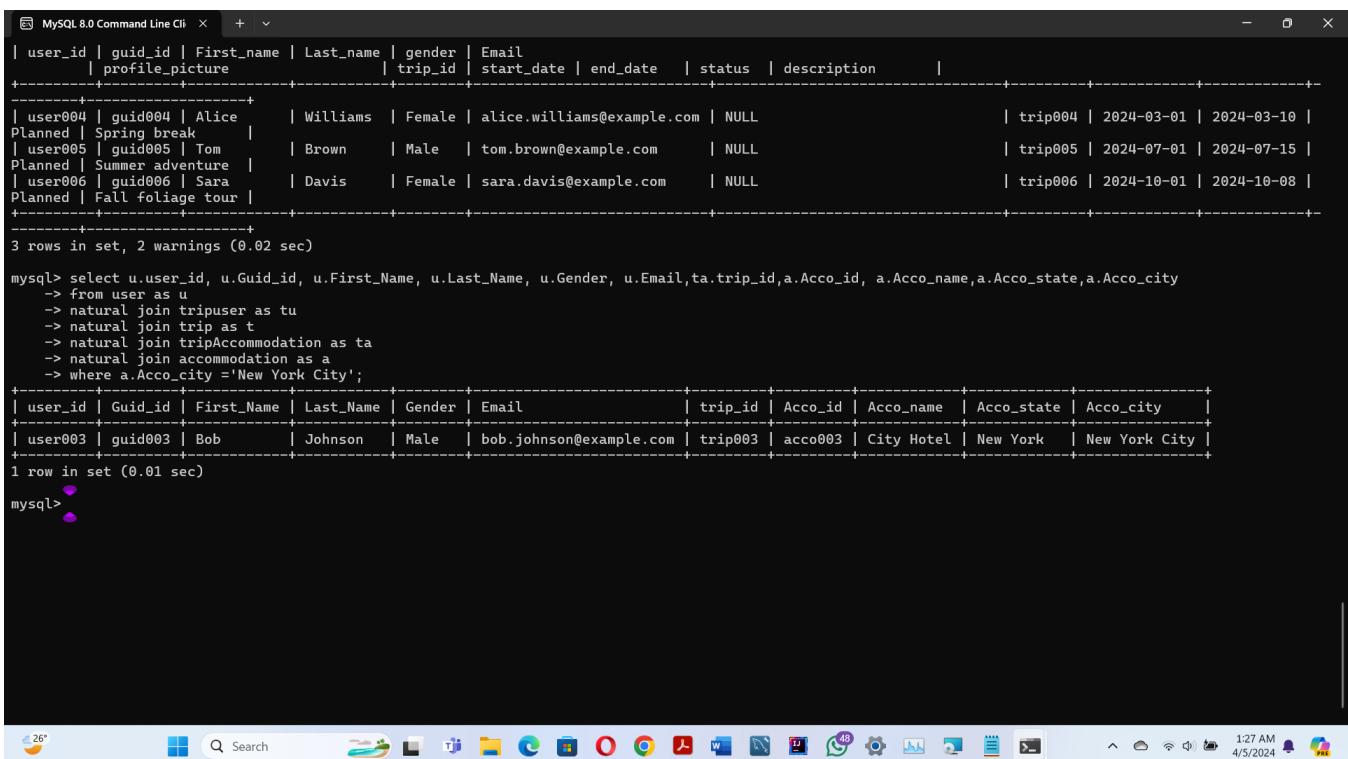


```

mysql> select
    -> coalesce(u.user_id, 'No user') as user_id, coalesce(u.Guid_id, 'No user')As Guid_id,
    -> coalesce(u.First_name,'No user') As First_name,coalesce(u.Last_name,'No user')as Last_name,
    -> coalesce(u.Gender,'No user')as Gender,coalesce(u.Email,'No user')as Email,
    -> coalesce(u.profile_picture,'No user')as profile_picture,coalesce(r.comments,'No review')as comments,
    -> coalesce(r.Rating,'No review')as Rating
    -> from user as u
    -> left join review as r on u.user_id=r.user_id
    -> union
    -> select
    -> coalesce(u.user_id, 'No user') as user_id, coalesce(u.Guid_id, 'No user')As Guid_id,
    -> coalesce(u.First_name,'No user') As First_name,
    -> coalesce(u.Last_name,'No user')as Last_name,
    -> coalesce(u.Gender,'No user')as Gender,
    -> coalesce(u.Email,'No user')as Email,
    -> coalesce(u.profile_picture,'No user')as profile_picture,
    -> coalesce(r.comments,'No review')as comments,
    -> coalesce(r.Rating,'No review')as Rating
    -> from user as u
    -> right join review as r on u.user_id=r.user_id;
+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | Guid_id | First_name | Last_name | Gender | Email           | profile_picture | comments          | Ra
ting |
+-----+-----+-----+-----+-----+-----+-----+-----+
| user001 | guid001 | Johnny     | Appleseed  | Male   | john.doe@example.com | 0x4E6F2075736572 | Excellent service | 5
| user002 | guid002 | Jane       | Smith      | Female  | jane.doe@example.com | 0x4E6F2075736572 | Beautiful mountain views | 4
| user003 | guid003 | Bob        | Johnson    | Male   | bob.johnson@example.com | 0x4E6F2075736572 | Convenient location in the city | 3
| user004 | guid004 | Alice      | Williams   | Female  | alice.williams@example.com | 0x4E6F2075736572 | Peaceful and cozy | 5
| user005 | guid005 | Tom        | Brown      | Male   | tom.brown@example.com | 0x4E6F2075736572 | Lovely cabin by the lake | 4
| user006 | guid006 | Sara       | Davis      | Female  | sara.davis@example.com | 0x4E6F2075736572 | No review          | No
review |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 52: Complex Query 6 (full outer join)



```

mysql> select
    -> user_id | guid_id | First_name | Last_name | gender | Email
    -> | profile_picture | trip_id | start_date | end_date | status | description
    -> +-----+-----+-----+-----+-----+-----+-----+-----+
    -> | user004 | guid004 | Alice      | Williams   | Female  | alice.williams@example.com | NULL             | trip004 | 2024-03-01 | 2024-03-10 | Planned | Spring break |
    -> | user005 | guid005 | Tom        | Brown      | Male   | tom.brown@example.com | NULL             | trip005 | 2024-07-01 | 2024-07-15 | Planned | Summer adventure |
    -> | user006 | guid006 | Sara       | Davis      | Female  | sara.davis@example.com | NULL             | trip006 | 2024-10-01 | 2024-10-08 | Planned | Fall foliage tour |
    -> +-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set, 2 warnings (0.02 sec)

mysql> select u.user_id, u.Guid_id, u.First_Name, u.Last_Name, u.Gender, u.Email,ta.trip_id,a.Acco_id, a.Acco_name,a.Acco_state,a.Acco_city
    -> from user as u
    -> natural join tripuser as tu
    -> natural join trip as t
    -> natural join tripAccommodation as ta
    -> natural join accommodation as a
    -> where a.Acco_city ='New York City';
+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | Guid_id | First_Name | Last_Name | Gender | Email           | trip_id | Acco_id | Acco_name | Acco_state | Acco_city |
| user003 | guid003 | Bob        | Johnson   | Male   | bob.johnson@example.com | trip003 | acco003 | City Hotel | New York | New York City |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql>

```

Figure 53: Complex Query 7 (natural join)

```
MySQL 8.0 Command Line Cli x + v
-----+
3 rows in set, 2 warnings (0.02 sec)

mysql> select u.user_id, u.Guid_id, u.First_Name, u.Last_Name, u.Gender, u.Email, ta.trip_id, a.Acco_id, a.Acco_name, a.Acco_state, a.Acco_city
-> from user as u
-> natural join tripuser as tu
-> natural join trip as t
-> natural join tripAccommodation as ta
-> natural join accommodation as a
-> where a.Acco_city ='New York City';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | Guid_id | First_Name | Last_Name | Gender | Email      | trip_id | Acco_id | Acco_name | Acco_state | Acco_city   |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user003 | guid003 | Bob        | Johnson    | Male    | bob.johnson@example.com | trip003 | acco003 | City Hotel | New York   | New York City |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 54: Complex Query 8

```
MySQL 8.0 Command Line Cli x + v
-----+
6 rows in set (0.01 sec)

mysql> select Acco_id,Acco_name, Acco_state,Acco_city
-> from accommodation
-> where Acco_state='Florida'
-> union
-> select Acco_id,Acco_name,Acco_state,Acco_city
-> from accommodation
-> where Acco_state='colorado';
+-----+-----+-----+-----+
| Acco_id | Acco_name     | Acco_state | Acco_city |
+-----+-----+-----+-----+
| acco001 | Beach Resort  | Florida    | Tampa     |
| acco002 | Mountain Resort | Colorado   | Aspen     |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Figure 55: Complex Query 9 (outer join)

MySQL 8.0 Command Line Cli

```
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select u.user_id,u.First_name,u.Last_name,u.Email,up.Phone_number
-> from user as u
-> join userEmail as ue on u.Email=ue.Email
-> join userphone as up on u.User_id =up.User_id
-> where exists(
->     select 1
->         from review as r
->         where r.user_id=u.user_id
-> );
+-----+-----+-----+-----+
| user_id | First_name | Last_name | Email      | Phone_number |
+-----+-----+-----+-----+
| user001 | Johnny     | Appleseed | john.doe@example.com | 9998877777    |
| user002 | Jane       | Smith     | jane.doe@example.com | 6667778888    |
| user003 | Bob        | Johnson   | bob.johnson@example.com | 5555555555    |
| user004 | Alice      | Williams  | alice.williams@example.com | 1112223333    |
| user005 | Tom        | Brown     | tom.brown@example.com | 4445556666    |
+-----+-----+-----+-----+
5 rows in set (0.01 sec)

mysql>
```

Figure 56: nested query with subquery

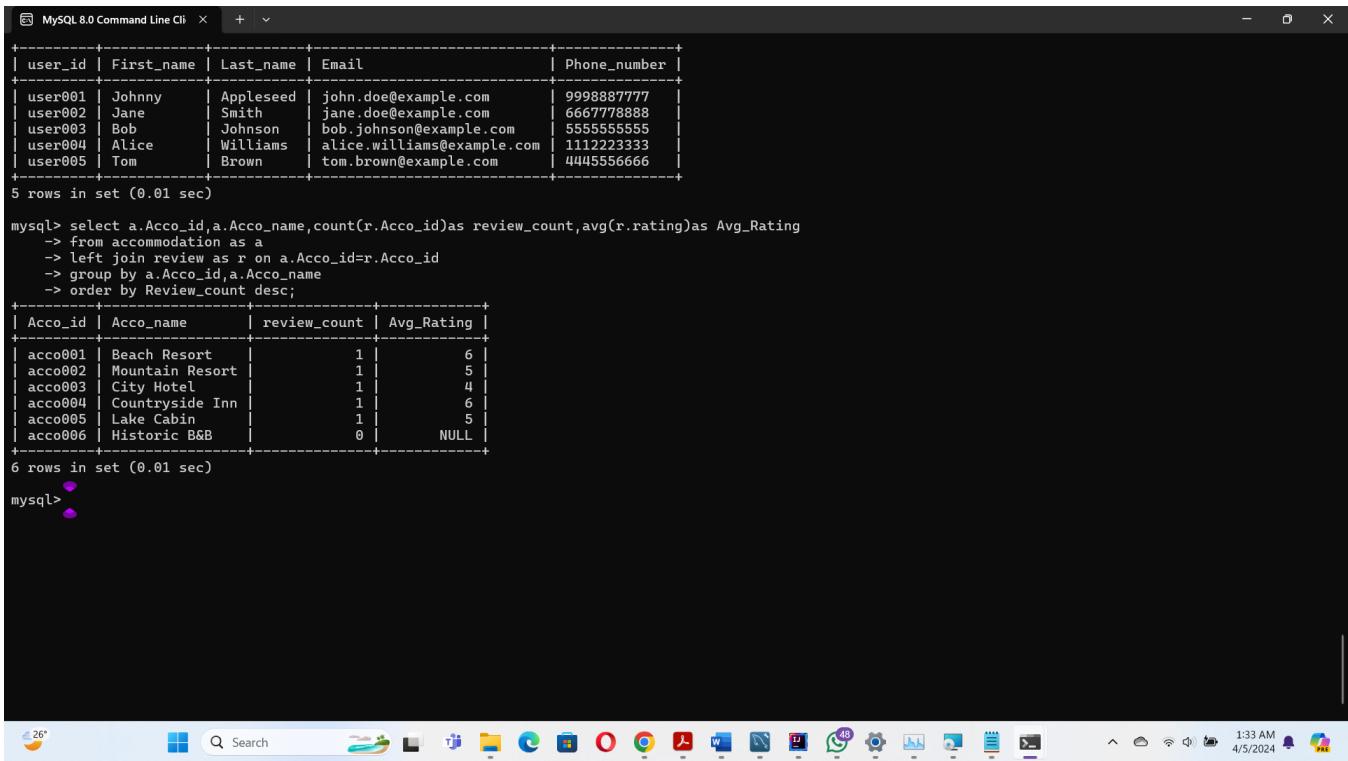
MySQL 8.0 Command Line Cli

```
+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> select u.user_id,u.First_name,u.Last_name,count(tu.Trip_id)as Trip_count
-> from user as u
-> left join tripUser as tu on u.user_id=tu.user_id
-> group by u.user_id,u.First_name,u.last_name
-> having Trip_count>0;
+-----+-----+-----+
| user_id | First_name | Last_name | Trip_count |
+-----+-----+-----+-----+
| user001 | Johnny     | Appleseed | 1          |
| user002 | Jane       | Smith     | 1          |
| user003 | Bob        | Johnson   | 1          |
| user004 | Alice      | Williams  | 1          |
| user005 | Tom        | Brown     | 1          |
| user006 | Sara        | Davis     | 1          |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 57: nested query with subquery and conditional count



MySQL 8.0 Command Line Cli

```

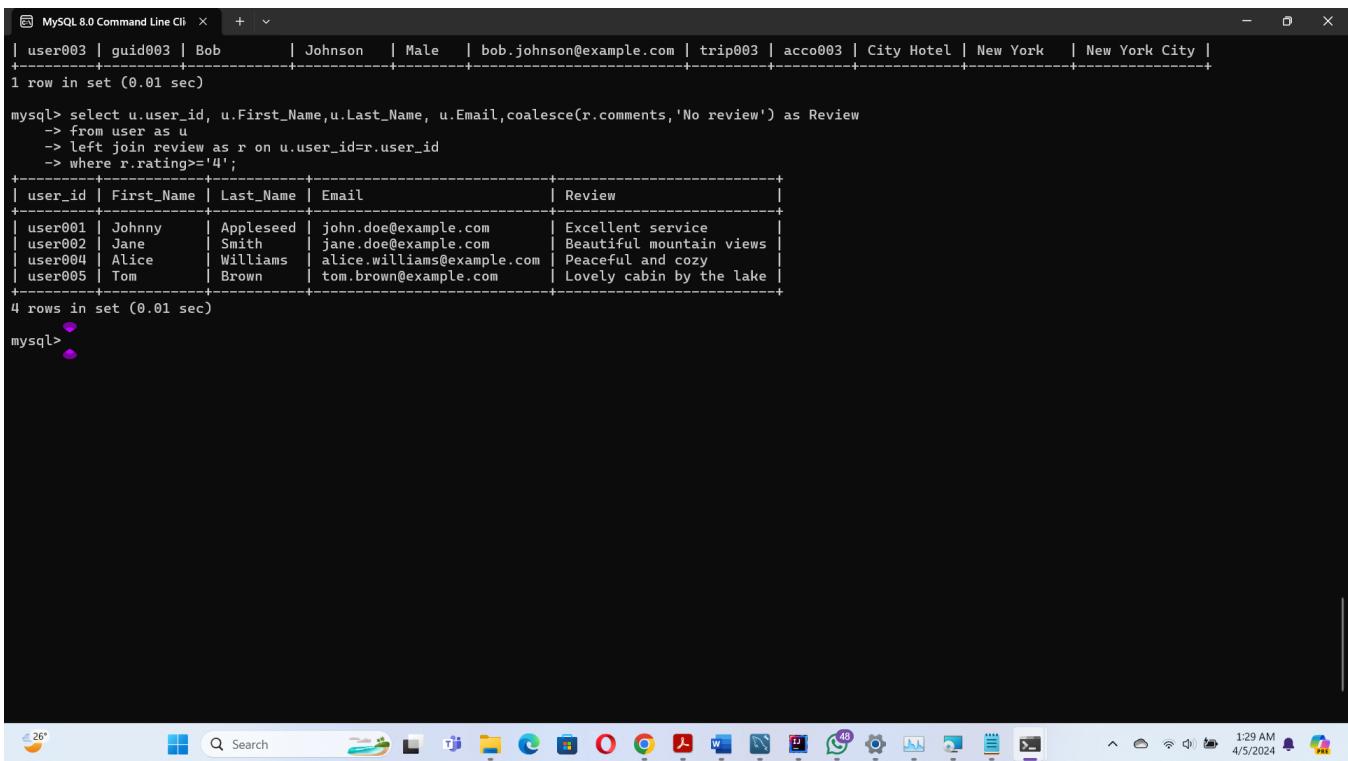
+-----+-----+-----+-----+
| user_id | First_name | Last_name | Email           | Phone_number |
+-----+-----+-----+-----+
| user001 | Johnny     | Appleseed | john.doe@example.com | 9998887777 |
| user002 | Jane       | Smith     | jane.doe@example.com | 6667778888 |
| user003 | Bob        | Johnson   | bob.johnson@example.com | 5555555555 |
| user004 | Alice      | Williams  | alice.williams@example.com | 1112223333 |
| user005 | Tom        | Brown     | tom.brown@example.com | 4445556666 |
+-----+-----+-----+-----+
5 rows in set (0.01 sec)

mysql> select a.Acco_id,a.Acco_name,count(r.Acco_id)as review_count,avg(r.rating)as Avg_Rating
    -> from accommodation as a
    -> left join review as r on a.Acco_id=r.Acco_id
    -> group by a.Acco_id,a.Acco_name
    -> order by Review_count desc;
+-----+-----+-----+-----+
| Acco_id | Acco_name | review_count | Avg_Rating |
+-----+-----+-----+-----+
| acco001 | Beach Resort | 1 | 6 |
| acco002 | Mountain Resort | 1 | 5 |
| acco003 | City Hotel | 1 | 4 |
| acco004 | Countryside Inn | 1 | 6 |
| acco005 | Lake Cabin | 1 | 5 |
| acco006 | Historic B&B | 0 | NULL |
+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>

```

Figure 58: nested query with aggregate function



MySQL 8.0 Command Line Cli

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| user003 | guid003 | Bob     | Johnson | Male   | bob.johnson@example.com | trip003 | acco003 | City Hotel | New York | New York City |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql> select u.user_id, u.First_Name,u.Last_Name, u.Email,coalesce(r.comments,'No review') as Review
    -> from user as u
    -> left join review as r on u.user_id=r.user_id
    -> where r.rating>='4';
+-----+-----+-----+-----+-----+
| user_id | First_Name | Last_Name | Email           | Review          |
+-----+-----+-----+-----+-----+
| user001 | Johnny     | Appleseed | john.doe@example.com | Excellent service |
| user002 | Jane       | Smith     | jane.doe@example.com | Beautiful mountain views |
| user004 | Alice      | Williams  | alice.williams@example.com | Peaceful and cozy |
| user005 | Tom        | Brown     | tom.brown@example.com | Lovely cabin by the lake |
+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>

```

Figure 59: complex query – left outer join

## 5 Chapter 05- Database Tuning

```

mysql> Select MySQL 8.0 Commandline Client
Your MySQL connection id is 12
Server version: 8.0.36 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or 'h' for help. Type 'c' to clear the current input statement.

mysql> use Trip_Planner_App_Database_01;
Database changed
mysql> EXPLAIN
-> SELECT destination_id, longitude, latitude, destination_name, description
-> FROM destination
-> UNION
-> SELECT NULL AS destination_id, NULL AS longitude, NULL AS latitude, acco_name AS destination_name, 'Accommodation' AS description
-> FROM accommodation
-> WHERE acco_city = 'New York City';
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | destination | NULL | ALL | NULL | NULL | NULL |
| 2 | UNION | accommodation | NULL | ALL | NULL | NULL | NULL |
| 3 | UNION RESULT | union0_2 | NULL | ALL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)

mysql> EXPLAIN
-> SELECT destination_id, longitude, latitude, destination_name, description
-> FROM (
->     SELECT destination_id, longitude, latitude, destination_name, description
->     FROM destination
->     UNION ALL
->     SELECT NULL AS destination_id, NULL AS longitude, NULL AS latitude, acco_name AS destination_name, 'Accommodation' AS description
->     FROM accommodation
->     WHERE acco_city = 'New York City'
-> ) AS combined_data;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | <derived2> | NULL | ALL | NULL | NULL | NULL |
| 2 | DERIVED | destination | NULL | ALL | NULL | NULL | NULL |
| 3 | UNION | accommodation | NULL | ALL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)

mysql>

```

The screenshot shows the MySQL Command Line Client interface. The command `EXPLAIN` is used to analyze two different ways of writing a UNION query. Both queries result in three rows being returned, with one warning about using UNION with NULL values.

Figure 60: Tuned and explained query of basic set operation union

```

mysql> Explain
3 rows in set, 1 warning (0.00 sec)

mysql> EXPLAIN
-> SELECT destination_id, longitude, latitude, destination_name, description,
->       NULL AS Acco_id, NULL AS Acco_name, NULL AS Acco_state, NULL AS Acco_city
->   FROM destination
->   WHERE destination_name NOT IN (
->       SELECT Acco_name
->       FROM accommodation
->       WHERE Acco_city IN ('Miami', 'Orlando')
->   );
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | destination | NULL | ALL | NULL | NULL | NULL |
| 2 | SIMPLE | <subquery2> | NULL | eq_ref | <auto_distinct_key> | <auto_distinct_key> | trip_planner_app_database_01.destination.Destination_name |
| 2 | MATERIALIZED | accommodation | NULL | ALL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)

mysql> EXPLAIN
-> SELECT d.destination_id, d.longitude, d.latitude, d.destination_name, d.description,
->       NULL AS Acco_id, NULL AS Acco_name, NULL AS Acco_state, NULL AS Acco_city
->   FROM destination d
->   LEFT JOIN accommodation a ON d.destination_name = a.Acco_name AND a.Acco_city IN ('Miami', 'Orlando')
->   WHERE a.Acco_name IS NULL;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | d | NULL | ALL | NULL | NULL | NULL |
| 1 | SIMPLE | a | NULL | ALL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql>

```

The screenshot shows the MySQL Command Line Client interface. The command `EXPLAIN` is used to analyze two different ways of writing a SET DIFFERENCE query. Both queries result in two rows being returned, with one warning about using NOT IN with subqueries.

Figure 61: Tuned and explained query of basic set operation set difference

```

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

mysql> EXPLAIN
--> SELECT
-->   u.user_id, u.Guid_id, u.First_Name, u.Last_Name, u.Gender, u.Email,
-->   ta.trip_id,
-->   a.Acco_id, a.Acco_name,a.Acco_state,a.Acco_city
-->   FROM `user` AS u
-->   NATURAL JOIN tripuser AS tu
-->   NATURAL JOIN trip AS ta
-->   NATURAL JOIN tripaccommodation AS ta
-->   NATURAL JOIN accommodation AS a
-->   WHERE
-->     a.Acco_city = 'New York City';
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1  | SIMPLE      | a     | NULL       | ALL  | PRIMARY       | NULL | NULL    | NULL  |
| 1  | SIMPLE      | ta    | NULL       | eq_ref| PRIMARY       | PRIMARY| 182    | trip_planner_app_database_01.a.Acco_id |
| 1  | SIMPLE      | tu    | NULL       | eq_ref| PRIMARY       | PRIMARY| 182    | trip_planner_app_database_01.ta.Trip_id  |
| 1  | SIMPLE      | u     | NULL       | eq_ref| PRIMARY       | PRIMARY| 182    | trip_planner_app_database_01.tu.User_id   |
| 1  | SIMPLE      | t     | NULL       | ref  | fk_trip_tripuser | fk_trip_tripuser | 182    | trip_planner_app_database_01.ta.Trip_id   |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set, 1 warning (0.00 sec)

mysql> EXPLAIN
--> SELECT
-->   u.user_id,
-->   u.Guid_id,
-->   u.First_Name,
-->   u.Last_Name,
-->   u.Gender,
-->   u.Email,
-->   ta.trip_id,
-->   a.Acco_id,
-->   a.Acco_name,
-->   a.Acco_state,
-->   a.Acco_city
-->   FROM `user` AS u
-->   JOIN tripuser AS tu ON u.user_id = tu.user_id
-->   JOIN trip AS t ON tu.trip_id = t.trip_id
-->   JOIN tripAccommodation AS ta ON t.trip_id = ta.trip_id
-->   JOIN accommodation AS a ON ta.Acco_id = a.Acco_id
-->   WHERE
-->     a.Acco_city = 'New York City';
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1  | SIMPLE      | a     | NULL       | ALL  | PRIMARY       | NULL | NULL    | NULL  |
| 1  | SIMPLE      | ta    | NULL       | eq_ref| PRIMARY       | PRIMARY| 182    | trip_planner_app_database_01.a.Acco_id |
| 1  | SIMPLE      | tu    | NULL       | eq_ref| PRIMARY       | PRIMARY| 182    | trip_planner_app_database_01.ta.Trip_id  |
| 1  | SIMPLE      | u     | NULL       | eq_ref| PRIMARY       | PRIMARY| 182    | trip_planner_app_database_01.tu.User_id   |
| 1  | SIMPLE      | t     | NULL       | ref  | fk_trip_tripuser | fk_trip_tripuser | 182    | trip_planner_app_database_01.ta.Trip_id   |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set, 1 warning (0.00 sec)

mysql>

```

Figure 62: Tuned and explained query of natural join

```

MySQL 8.0 Command Line Client
2 rows in set, 1 warning (0.00 sec)

mysql> EXPLAIN
--> SELECT
-->   u.user_id, u.First_Name, u.Last_Name, u.Email, COALESCE(r.comments,'No review') as Review
-->   FROM `user` AS u
-->   LEFT JOIN
-->     review AS r ON u.user_id = r.user_id AND r.rating >= '4';
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1  | SIMPLE      | u     | NULL       | ALL  | NULL          | NULL | NULL    | NULL  |
| 1  | SIMPLE      | r     | NULL       | ref  | fk_review_user | fk_review_user | 182    | trip_planner_app_database_01.u.User_id |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql> EXPLAIN
--> SELECT
-->   u.user_id, u.First_Name, u.Last_Name, u.Email, COALESCE(r.Review, 'No review') AS Review
-->   FROM `user` AS u
-->   LEFT JOIN
-->     (SELECT user_id, comments AS Review FROM review WHERE rating >= '4') AS r ON u.user_id = r.user_id;
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1  | SIMPLE      | u     | NULL       | ALL  | NULL          | NULL | NULL    | NULL  |
| 1  | SIMPLE      | review | NULL      | ref  | fk_review_user | fk_review_user | 182    | trip_planner_app_database_01.u.User_id |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql>

```

Figure 63: Tuned and explained query of left outer join

```

MySQL 8.0 Command Line Client
mysql> EXPLAIN
--> SELECT
--> r.user_id, r.Acco_id, r.comments, r.Rating, a.Acco_name, a.Acco_state, a.Acco_city
--> FROM
--> review AS r
--> RIGHT JOIN
--> a_accommodation AS a ON r.Acco_id = a.Acco_id
--> WHERE
--> a.Acco_state = 'Colorado';
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | a | NULL | ALL | NULL | NULL | NULL |
| 1 | SIMPLE | r | NULL | ref | fk_review_accommodation | fk_review_accommodation | 182 | trip_planner_app_database_01.a.acco_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql> EXPLAIN
--> SELECT
--> r.user_id, r.Acco_id, r.comments, r.Rating, a.Acco_name, a.Acco_state, a.Acco_city
--> FROM
--> a_accommodation AS a (SELECT * FROM accommodation WHERE Acco_state = 'Colorado') AS a
--> LEFT JOIN
--> review AS r ON a.Acco_id = r.Acco_id;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | accommodation | 1 | ALL | NULL | NULL | NULL |
| 1 | SIMPLE | r | NULL | ref | fk_review_accommodation | fk_review_accommodation | 182 | trip_planner_app_database_01.accommodation.acco_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql>

```

Figure 64: Tuned and explained query of right outer join

```

MySQL 8.0 Command Line Client
1 row in set, 1 warning (0.00 sec)

mysql> EXPLAIN
--> SELECT
--> COALESCE(u.user_id, 'No user') AS user_id,
--> COALESCE(u.Guild_id, 'No user') AS Guild_id,
--> COALESCE(u.First_name, 'No user') AS First_name,
--> COALESCE(u.Last_name, 'No user') AS Last_name,
--> COALESCE(u.Gender, 'No user') AS gender,
--> COALESCE(u.Email, 'No user') AS Email,
--> COALESCE(u.profile_picture, 'No user') AS profile_picture,
--> COALESCE(r.comments, 'No review') AS comments,
--> COALESCE(r.rating, 'No rating') AS Rating
--> FROM user AS u
--> LEFT JOIN review AS r ON u.user_id = r.user_id
--> UNION
--> SELECT
--> COALESCE(u.user_id, 'No user') AS user_id,
--> COALESCE(u.Guild_id, 'No user') AS Guild_id,
--> COALESCE(u.First_name, 'No user') AS First_name,
--> COALESCE(u.Last_name, 'No user') AS Last_name,
--> COALESCE(u.Gender, 'No user') AS gender,
--> COALESCE(u.Email, 'No user') AS Email,
--> COALESCE(u.profile_picture, 'No user') AS profile_picture,
--> COALESCE(r.comments, 'No review') AS comments,
--> COALESCE(r.rating, 'No rating') AS Rating
--> FROM user AS u
--> RIGHT JOIN review AS r ON u.user_id = r.user_id
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | u | NULL | ALL | NULL | NULL | NULL | |
| 1 | PRIMARY | r | NULL | ref | fk_review_user | fk_review_user | 182 | trip_planner_app_database_01.u.user_id |
| 2 | UNION | u | NULL | ALL | NULL | NULL | NULL |
| 2 | UNION | r | NULL | ref | PRIMARY | PRIMARY | 182 | trip_planner_app_database_01.r.user_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)

mysql> EXPLAIN
--> SELECT
--> COALESCE(u.user_id, 'No user') AS user_id,
--> COALESCE(u.Guild_id, 'No user') AS Guild_id,
--> COALESCE(u.First_name, 'No user') AS First_name,
--> COALESCE(u.Last_name, 'No user') AS Last_name,
--> COALESCE(u.Gender, 'No user') AS gender,
--> COALESCE(u.Email, 'No user') AS Email,
--> COALESCE(u.profile_picture, 'No user') AS profile_picture,
--> COALESCE(r.comments, 'No review') AS comments,
--> COALESCE(r.rating, 'No rating') AS Rating
--> FROM user AS u
--> LEFT JOIN review AS r ON u.user_id = r.user_id
--> UNION
--> ALL
--> SELECT
--> 'No user',
--> 'No user'
--> FROM review AS r
--> WHERE r.user_id = (
-->   SELECT user_id FROM user WHERE user_id = r.user_id
--> );
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | u | NULL | ALL | NULL | NULL | NULL | |
| 1 | PRIMARY | r | NULL | ref | fk_review_user | fk_review_user | 182 | trip_planner_app_database_01.u.user_id |
| 2 | UNION | r | NULL | ALL | NULL | NULL | NULL |
| 2 | UNION | r | NULL | ref | PRIMARY | PRIMARY | 182 | trip_planner_app_database_01.r.user_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 2 warnings (0.00 sec)

mysql>

```

Figure 65: Tuned and explained query of full outer join

```

MySQL 8.0 Command Line Client

mysql> EXPLAIN
--> SELECT Acco_id, Acco_name, Acco_state, Acco_city
--> FROM accommodation
--> WHERE Acco_state = 'Florida'
--> UNION
--> SELECT Acco_id, Acco_name, Acco_state, Acco_city
--> FROM accommodation
--> WHERE Acco_state = 'Colorado';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | accommodation | NULL | ALL | NULL | NULL | NULL | NULL |
| 2 | UNION | accommodation | NULL | ALL | NULL | NULL | NULL | NULL |
| 3 | UNION RESULT | union1,> | NULL | ALL | NULL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)

mysql> EXPLAIN
--> SELECT Acco_id, Acco_name, Acco_state, Acco_city
--> FROM accommodation
--> WHERE Acco_state IN ('Florida', 'Colorado');
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | accommodation | NULL | ALL | NULL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)

mysql>

```

Figure 66: Tuned and explained query of outer join

```

MySQL 8.0 Command Line Client

mysql> EXPLAIN
--> SELECT u.user_id, u.First_name, u.Last_name, u.Email, up.Phone_number
--> FROM user AS u
--> JOIN userEmail AS ue ON u.Email = ue.Email
--> JOIN userPhone AS up ON u.user_id = up.User_id
--> WHERE EXISTS (
-->   SELECT 1
-->   FROM review AS r
-->   WHERE r.user_id = u.user_id
--> );
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | r | NULL | index | fk_review_user | fk_review_user | 182 | NULL |
| 1 | SIMPLE | u | NULL | eq_ref | PRIMARY,Email | PRIMARY | 182 | ue.Email |
| 1 | SIMPLE | ue | NULL | eq_ref | PRIMARY | PRIMARY | 182 | trip_planner_app_database_01.r.User_id |
| 1 | SIMPLE | up | NULL | ref | fk_userPhone_user | fk_userPhone_user | 182 | trip_planner_app_database_01.r.User_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set, 2 warnings (0.00 sec)

mysql> EXPLAIN
--> SELECT u.user_id, u.First_name, u.Last_name, u.Email, up.Phone_number
--> FROM user AS u
--> INNER JOIN review AS r ON u.user_id = r.user_id
--> INNER JOIN userEmail AS ue ON u.Email = ue.Email
--> INNER JOIN userPhone AS up ON u.user_id = up.User_id
--> GROUP BY user_id, u.First_name, u.Last_name, u.Email, up.Phone_number;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | u | NULL | ALL | PRIMARY,Email | NULL | NULL | NULL |
| 1 | SIMPLE | r | NULL | ref | fk_review_user | fk_review_user | 182 | trip_planner_app_database_01.u.User_id |
| 1 | SIMPLE | ue | NULL | eq_ref | PRIMARY | PRIMARY | 182 | trip_planner_app_database_01.u.Email |
| 1 | SIMPLE | up | NULL | ref | fk_userPhone_user | fk_userPhone_user | 182 | trip_planner_app_database_01.u.User_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set, 1 warning (0.00 sec)

mysql>

```

Figure 67: Tuned and explained query of nested query with subquery

```

MySQL> EXPLAIN
--> SELECT a.Acco_id, a.Acco_name, COUNT(r.Acco_id) AS review_count, AVG(r.rating) AS Avg_Rating
--> FROM accommodation AS a
--> LEFT JOIN review AS r ON a.Acco_id = r.Acco_id
--> GROUP BY a.Acco_id, a.Acco_name
--> ORDER BY review_count DESC;
+----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | a | NULL | ALL | NULL | NULL | NULL | NULL |
| 1 | SIMPLE | r | NULL | ref | fk_review_accommodation | fk_review_accommodation | 182 | trip_planner_app_database_01.a.Acco_id |
+----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

MySQL> EXPLAIN
--> SELECT a.Acco_id, a.Acco_name, COUNT(r.Acco_id) AS review_count, AVG(r.rating) AS Avg_Rating
--> FROM accommodation AS a
--> LEFT JOIN review AS r ON a.Acco_id = r.Acco_id
--> GROUP BY a.Acco_id, a.Acco_name
--> ORDER BY review_count DESC;
+----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | a | NULL | ALL | NULL | NULL | NULL | NULL |
| 1 | SIMPLE | r | NULL | ref | fk_review_accommodation | fk_review_accommodation | 182 | trip_planner_app_database_01.a.Acco_id |
+----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql>

```

Figure 68: Tuned and explained query of nested query with aggregate function

```

MySQL> EXPLAIN
--> SELECT u.user_id, u.First_name, u.Last_name, COUNT(tu.Trip_id) AS Trip_count
--> FROM user AS u
--> LEFT JOIN tripUser AS tu ON u.user_id = tu.user_id
--> GROUP BY u.user_id, u.First_name, u.Last_name
--> HAVING Trip_count > 0;
+----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | tu | NULL | ALL | NULL | NULL | NULL | NULL |
| 1 | SIMPLE | u | NULL | ref | fk_tripUser_user | fk_tripUser_user | 182 | trip_planner_app_database_01.u.user_id |
+----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

MySQL> EXPLAIN
--> SELECT u.user_id, u.First_name, u.last_name, COUNT(tu.Trip_id) AS Trip_count
--> FROM user AS u
--> LEFT JOIN tripUser AS tu ON u.user_id = tu.user_id
--> WHERE tu.user_id IS NOT NULL -- Only consider users who have at least one trip
--> GROUP BY u.user_id, u.First_name, u.last_name;
+----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | tu | NULL | index | fk_tripUser_user | fk_tripUser_user | 182 | NULL |
| 1 | SIMPLE | u | NULL | eq_ref | PRIMARY | PRIMARY | 182 | trip_planner_app_database_01.tu.user_id |
+----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql>

```

Figure 69: Tuned and explained query of nested query with subquery and conditional count