Paul Ycay

500709618

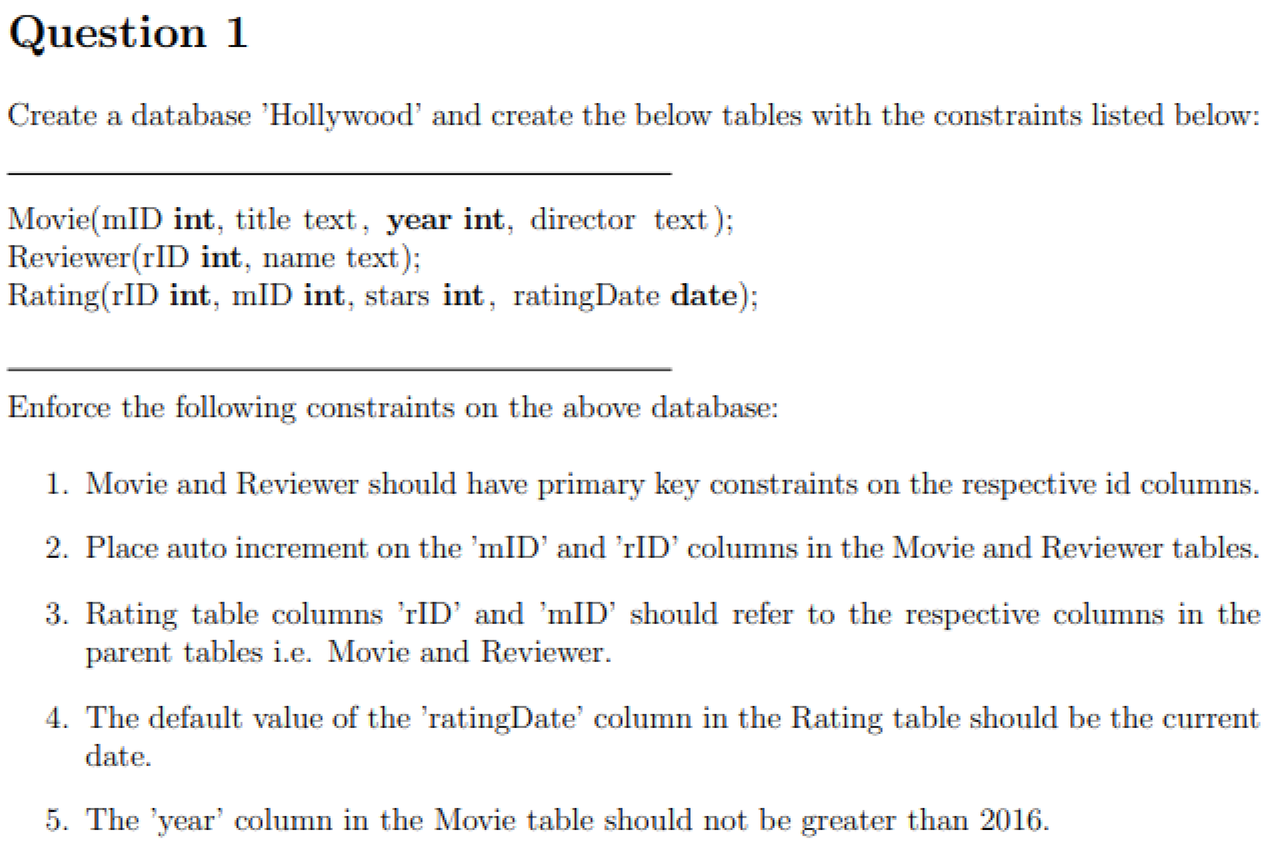
CIND110- Data Organization for Data Analysts

Dr. Tamer Abdou

June 26, 2019

<u>Assignment 2</u>

This assignment was executed using the *MySQL* terminal.



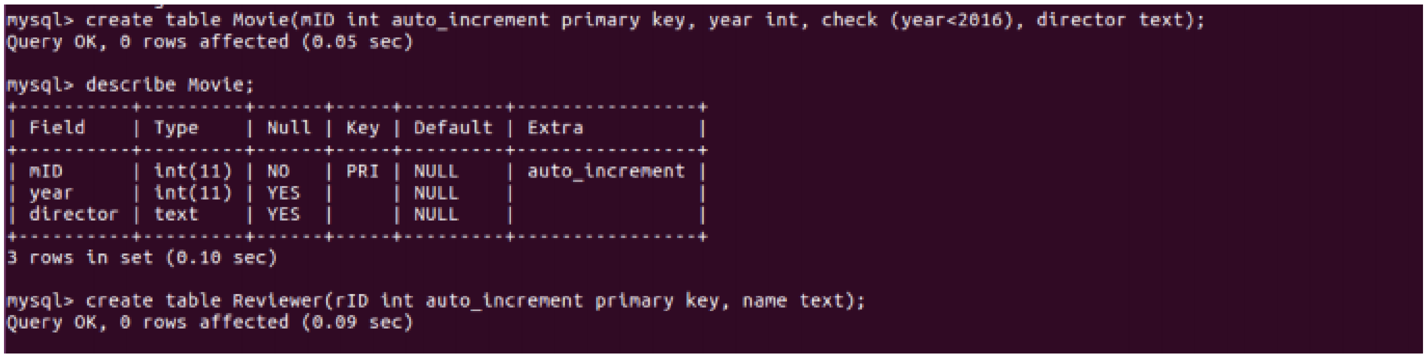
Question 1

Create a database 'Hollywood' and create the below tables with the constraints listed below:

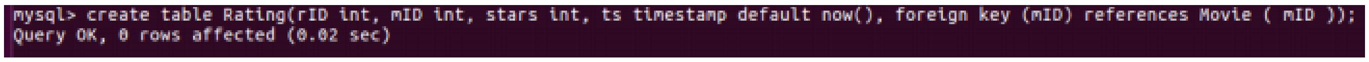
Movie(mID **int**, title text, **year int**, director text);
Reviewer(rID **int**, name text);
Rating(rID **int**, mID **int**, stars **int**, ratingDate **date**);

Enforce the following constraints on the above database:

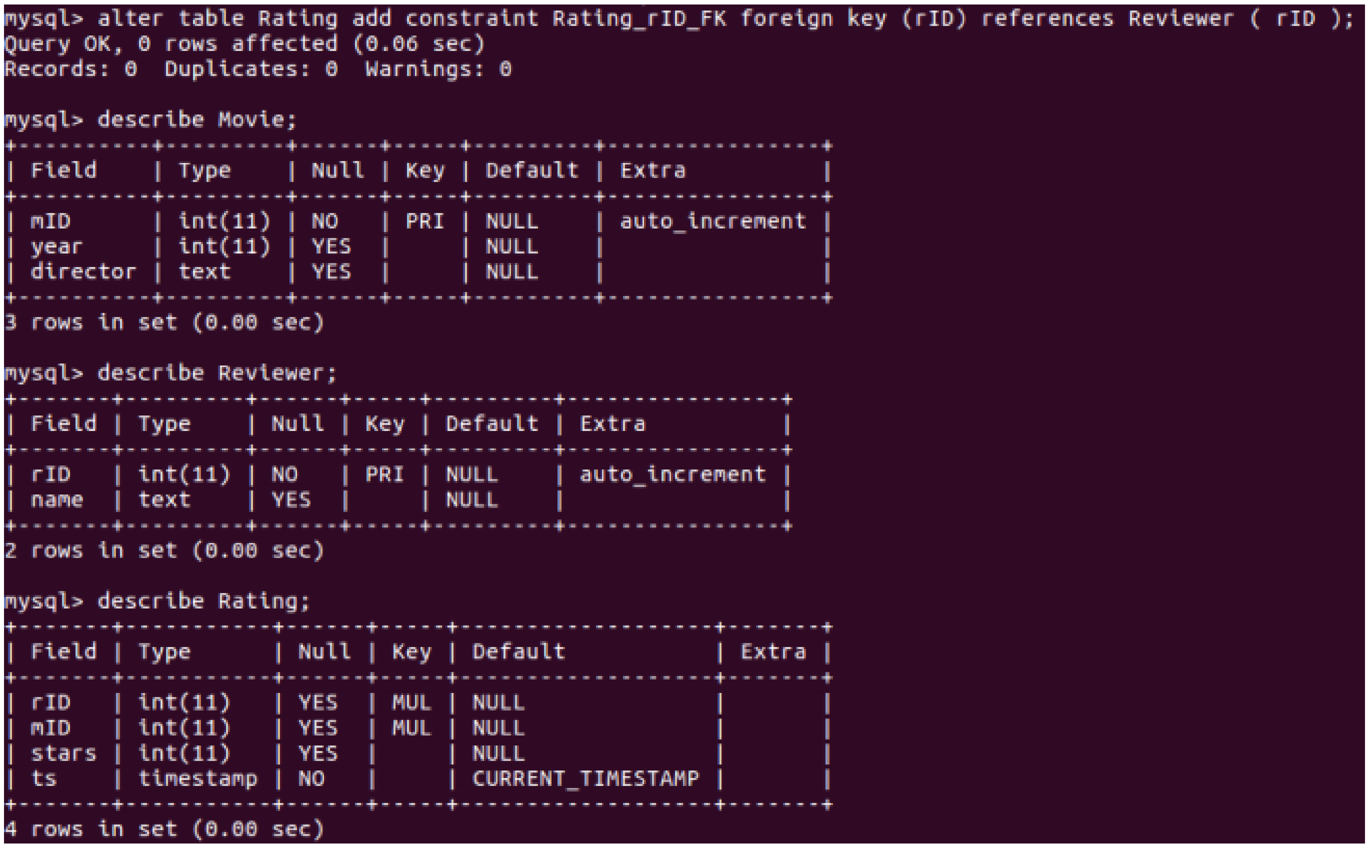
1. Movie and Reviewer should have primary key constraints on the respective id columns.
2. Place auto increment on the 'mID' and 'rID' columns in the Movie and Reviewer tables.
3. Rating table columns 'rID' and 'mID' should refer to the respective columns in the parent tables i.e. Movie and Reviewer.
4. The default value of the 'ratingDate' column in the Rating table should be the current date.
5. The 'year' column in the Movie table should not be greater than 2016.



```
mysql> create table Movie(mID int auto_increment primary key, year int, check (year<2016), director text);
Query OK, 0 rows affected (0.05 sec)

mysql> describe Movie;
+----------+---------+------+-----+---------+----------------+
| Field    | Type    | Null | Key | Default | Extra          |
+----------+---------+------+-----+---------+----------------+
| mID      | int(11) | NO   | PRI | NULL    | auto_increment |
| year     | int(11) | YES  |     | NULL    |                |
| director | text    | YES  |     | NULL    |                |
+----------+---------+------+-----+---------+----------------+
3 rows in set (0.10 sec)

mysql> create table Reviewer(rID int auto_increment primary key, name text);
Query OK, 0 rows affected (0.09 sec)
```



```
mysql> create table Rating(rID int, mID int, stars int, ts timestamp default now(), foreign key (mID) references Movie ( mID ));
Query OK, 0 rows affected (0.02 sec)
```



```
mysql> alter table Rating add constraint Rating_rID_FK foreign key (rID) references Reviewer ( rID );
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe Movie;
+----------+---------+------+-----+---------+----------------+
| Field    | Type    | Null | Key | Default | Extra          |
+----------+---------+------+-----+---------+----------------+
| mID      | int(11) | NO   | PRI | NULL    | auto_increment |
| year     | int(11) | YES  |     | NULL    |                |
| director | text    | YES  |     | NULL    |                |
+----------+---------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> describe Reviewer;
+-------+---------+------+-----+---------+----------------+
| Field | Type    | Null | Key | Default | Extra          |
+-------+---------+------+-----+---------+----------------+
| rID   | int(11) | NO   | PRI | NULL    | auto_increment |
| name  | text    | YES  |     | NULL    |                |
+-------+---------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql> describe Rating;
+-------+-----------+------+-----+-------------------+-------+
| Field | Type      | Null | Key | Default           | Extra |
+-------+-----------+------+-----+-------------------+-------+
| rID   | int(11)   | YES  | MUL | NULL              |       |
| mID   | int(11)   | YES  | MUL | NULL              |       |
| stars | int(11)   | YES  |     | NULL              |       |
| ts    | timestamp | NO   |     | CURRENT_TIMESTAMP |       |
+-------+-----------+------+-----+-------------------+-------+
4 rows in set (0.00 sec)
```

Question 2

Execute the following script in *MySQL* terminal

```
DROP DATABASE IF EXISTS cind110A2Script1;

CREATE SCHEMA cind110A2Script1;

USE cind110A2Script1;

CREATE TABLE hiking (
 trail  CHAR (50),
area CHAR (50),
distance FLOAT,
est_time FLOAT);

SHOW TABLES;

SHOW COLUMNS FROM hiking;

INSERT INTO hiking VALUES
('Cedar Creek Falls','Upper San Diego',4.5,2.5);

INSERT INTO hiking(trail, area) VALUES
('East Mesa Loop', 'Cuyamaca Mountains');

SELECT * FROM hiking;

SET SQL_SAFE_UPDATES = 0;

UPDATE hiking
SET distance = 10.5, est_time = 5.5
WHERE trail = 'East Mesa Loop';

USE cind110A2Script1;

DELETE FROM hiking WHERE trail = 'Cedar Creek Falls';

SELECT * FROM hiking;
```

1. Write the SQL statements to insert the following values into the hiking table:

trail	area	distance	est_time
East Mesa Loop	Cuyamaca Mountains	10.50	5.50
Oak Canyon	NULL	3.00	NULL

2. Write the SQL statements to update the entry for the 'Oak Canyon' trail. Set the area to 'Mission Trails Regional Park' and the estimated time (est_time) to 2 hours. Your table should then look like the following:

trail	area	distance	est_time
East Mesa Loop	Cuyamaca Mountains	10.50	5.50
Oak Canyon	Mission Trails Regional Park	3.00	2.00

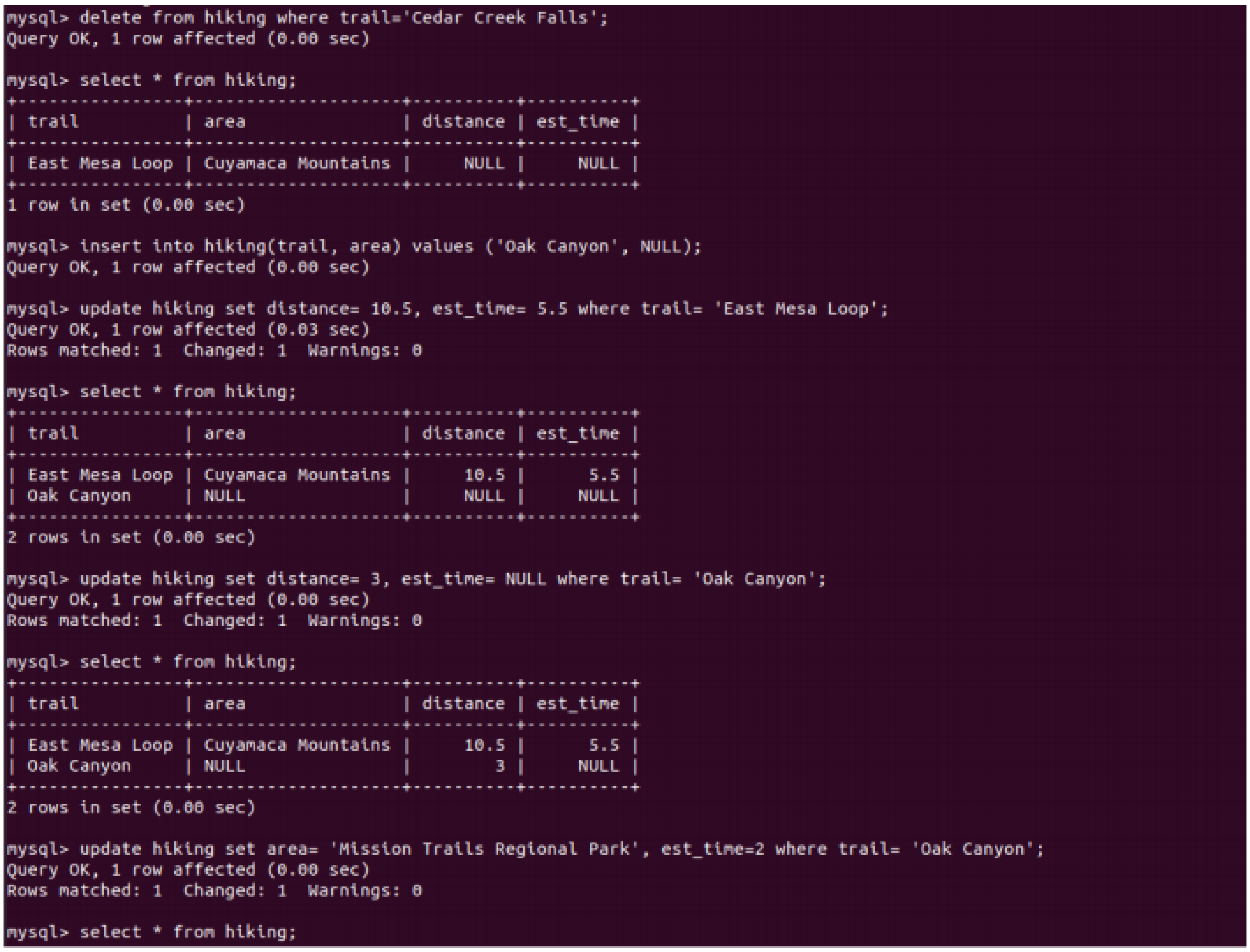
3. Write the SQL statement to delete trails with a distance greater than 5 miles.

4. Write the SQL statement to create a table called 'rating'. This table rates the difficulty of a hiking trail. It will have two columns: the trail name, 'trail' and the difficulty, 'difficulty'. The tail name is a string of no more than 50 characters and the difficulty is an integer (INT).

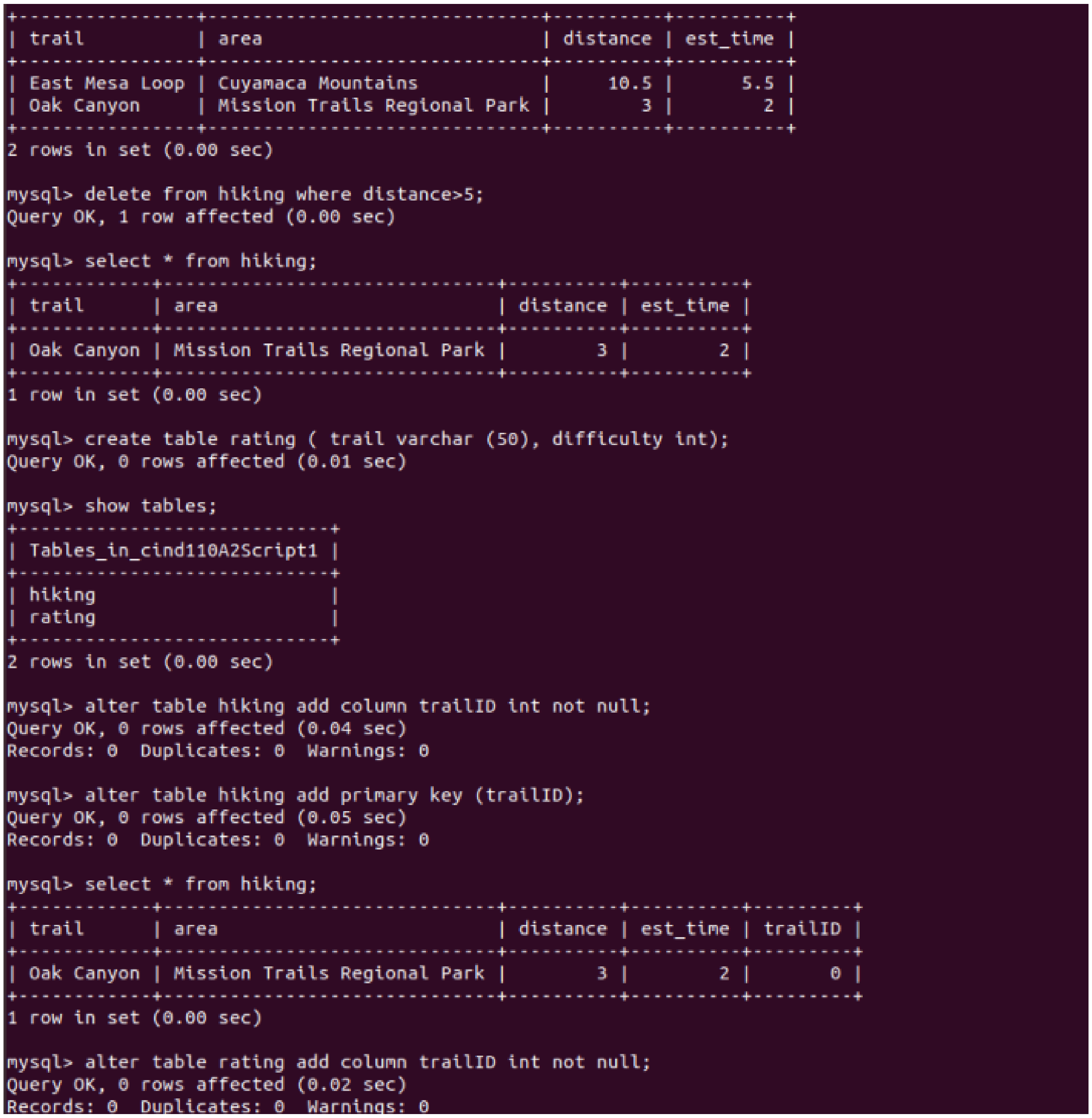
5. Write the command to add another column to the hiking table called 'trailID' with Primary key constraint.

6. Add another column called 'trailID' in the 'rating' table, which should be the foreign key with the table referring to the hiking table.

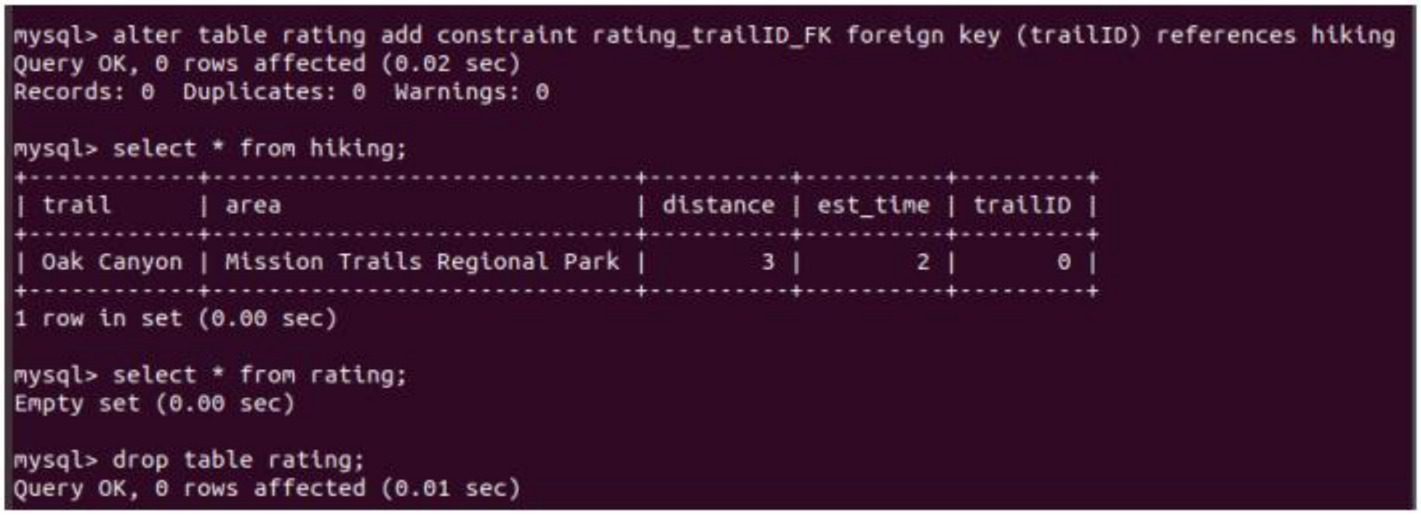
7. What is the command to delete the rating table?



```
mysql> delete from hiking where trail='Cedar Creek Falls';
Query OK, 1 row affected (0.00 sec)

mysql> select * from hiking;
+----------------+--------------------+----------+----------+
| trail          | area               | distance | est_time |
+----------------+--------------------+----------+----------+
| East Mesa Loop | Cuyamaca Mountains |     NULL |     NULL |
+----------------+--------------------+----------+----------+
1 row in set (0.00 sec)

mysql> insert into hiking(trail, area) values ('Oak Canyon', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> update hiking set distance= 10.5, est_time= 5.5 where trail= 'East Mesa Loop';
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from hiking;
+----------------+--------------------+----------+----------+
| trail          | area               | distance | est_time |
+----------------+--------------------+----------+----------+
| East Mesa Loop | Cuyamaca Mountains |     10.5 |      5.5 |
| Oak Canyon     | NULL               |     NULL |     NULL |
+----------------+--------------------+----------+----------+
2 rows in set (0.00 sec)

mysql> update hiking set distance= 3, est_time= NULL where trail= 'Oak Canyon';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from hiking;
+----------------+--------------------+----------+----------+
| trail          | area               | distance | est_time |
+----------------+--------------------+----------+----------+
| East Mesa Loop | Cuyamaca Mountains |     10.5 |      5.5 |
| Oak Canyon     | NULL               |        3 |     NULL |
+----------------+--------------------+----------+----------+
2 rows in set (0.00 sec)

mysql> update hiking set area= 'Mission Trails Regional Park', est_time=2 where trail= 'Oak Canyon';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from hiking;
```



```
+----------------+------------------------------+----------+----------+
| trail          | area                         | distance | est_time |
+----------------+------------------------------+----------+----------+
| East Mesa Loop | Cuyamaca Mountains           |     10.5 |      5.5 |
| Oak Canyon     | Mission Trails Regional Park |        3 |        2 |
+----------------+------------------------------+----------+----------+
2 rows in set (0.00 sec)

mysql> delete from hiking where distance>5;
Query OK, 1 row affected (0.00 sec)

mysql> select * from hiking;
+------------+------------------------------+----------+----------+
| trail      | area                         | distance | est_time |
+------------+------------------------------+----------+----------+
| Oak Canyon | Mission Trails Regional Park |        3 |        2 |
+------------+------------------------------+----------+----------+
1 row in set (0.00 sec)

mysql> create table rating ( trail varchar (50), difficulty int);
Query OK, 0 rows affected (0.01 sec)

mysql> show tables;
+----------------------------+
| Tables_in_cind110A2Script1 |
+----------------------------+
| hiking                     |
| rating                     |
+----------------------------+
2 rows in set (0.00 sec)

mysql> alter table hiking add column trailID int not null;
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table hiking add primary key (trailID);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from hiking;
+------------+------------------------------+----------+----------+---------+
| trail      | area                         | distance | est_time | trailID |
+------------+------------------------------+----------+----------+---------+
| Oak Canyon | Mission Trails Regional Park |        3 |        2 |       0 |
+------------+------------------------------+----------+----------+---------+
1 row in set (0.00 sec)

mysql> alter table rating add column trailID int not null;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



```
mysql> alter table rating add constraint rating_trailID_FK foreign key (trailID) references hiking
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from hiking;
+------------+------------------------------+----------+----------+---------+
| trail      | area                         | distance | est_time | trailID |
+------------+------------------------------+----------+----------+---------+
| Oak Canyon | Mission Trails Regional Park |        3 |        2 |       0 |
+------------+------------------------------+----------+----------+---------+
1 row in set (0.00 sec)

mysql> select * from rating;
Empty set (0.00 sec)

mysql> drop table rating;
Query OK, 0 rows affected (0.01 sec)
```

Question 3

Consider the following tables for Customer, Salesman and Order entities.

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	5002
3001	Brad Guzan	London		5005
3004	Fabian Johns	Paris	300	5006
3007	Brad Davis	New York	200	5001
3009	Geoff Camero	Berlin	100	5003
3008	Julian Green	London	300	5002
3003	Jozy Altidor	Moscow	200	5007

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5003	Lauson Hen		0.12
5007	Paul Adam	Rome	0.13

Order_No	Purch_Amt	Ord_Date	Customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003
70012	250.45	2012-06-27	3008	5002
70011	75.29	2012-08-17	3003	5007
70013	3045.6	2012-04-25	3002	5001

1. Write an SQL statement to prepare a list with salesman name, customer name and their cities for the salesmen and customer who belong to same city.
2. Write an SQL statement to make a list with order no, purchase amount, customer name and their cities for the orders where order amount is between 500 and 2000.
3. Write an SQL statement to find out which salesmen are working for which customer.
4. Write an SQL statement to find the list of customers who appointed a salesman for their jobs whose commission is more than 12%.
5. Write an SQL statement to find the list of customers who appointed a salesman for their jobs who does not live in same city where the customer lives, and gets a commission above 12%.
6. Write an SQL statement to find the details of an order i.e. order number, order date, amount of order, which customer gives the order and which salesman works for that customer and how much commission he gets for an order.
7. Write an SQL statement to make a join within the tables salesman, customer and orders such that the same column of each table will appear once and only the related rows will be returned.

1. select Salesman.s_name, Customer.c_name, customer.city from Customer inner join Salesman on Customer.city=Salesman.city;

2. select Order.ord_no, Order.purch_amt, Customer.cust_name, Customer.city from Order inner join Customer on Order.Customer_id=Customer.customer_id where Order.purch_amt between 500 and 2000;

3. select Customer.cust_name, Salesman.name from Customer, Salesman where Salesman.salesman_id=Customer.salesman_id;

4. select a.cust_name, a.city, b.name, b.commission from Customer as a inner join Salesman as b on a.salesman_id=b.salesman_id where b.commission>0.12;

5. select a.cust_name, a.city, b.name, b.commission from Customer as a inner join Salesman as b on a.salesman_id=b.salesman_id where b.commission>0.12 and not a.city=b.city;

6. select o.ord_no, o.ord_date, o.purch_amt, c.cust_name, s.name as "Salesman Name", s.commission from Orders as o, Customer as c, Salesman as s where o.customer_id=c.customer_id and c.salesman_id=s.salesman_id;

   alternate version: select a.ord_no, a.ord_date, a.purch_amt, b.cust_name as Customer, b.grade, c.name as Salesman, c.commission from Orders a inner join Customer b on a.customer_id=b.customer_id inner join Salesman c on a.salesman_id=c.salesman_id;

7. select * from Orders natural join Customer natural join salesman;

   a. *alternate*: select o.ord_no, o.purch_amt, o.ord_date, o.customer_id, o.salesman_id, c.cust_name, c.city, s.city, c.grade, s.name, s.commission from Orders as o, Customer as c, Salesman as s where o.customer_id=c.customer_id and o.salesman_id=s.salesman_id and c.city=s.city;

Question 4

Consider the following relations for a database that keeps track of student enrollment in courses and books adopted for each course.

STUDENT(Ssn, Name, Major, Bdate)
COURSE(Course#, Cname, Dept)
ENROLL(Ssn, Course#, Quarter, Grade)
BOOK_ADOPTION(Course#, Quarter, Book_isbn)
TEXT(Book_isbn, Book_title, Publisher, Author)

Having that a relation can have zero or more foreign keys and each foreign key can refer to different referenced relations. Specify the foreign keys for this schema.

The foreign keys for this schema are Course# in the tables ENROLL and BOOK_ADOPTION, which references the primary key Course# in table COURSE. Ssn in table ENROLL is a foreign key referencing the primary key Ssn in the table STUDENT. The foreign key Book_isbn in table BOOK_ADOPTION references the primary key Book_isbn in table TEXT.