Name: Rahul Purushottam Gaonkar

Homework-2

Problem 1:

In this problem, you have to write SQL and RA queries for a database modeling the short-term leasing of houses or apartments, in a system somewhat similar to Airbnb. Here are the tables:

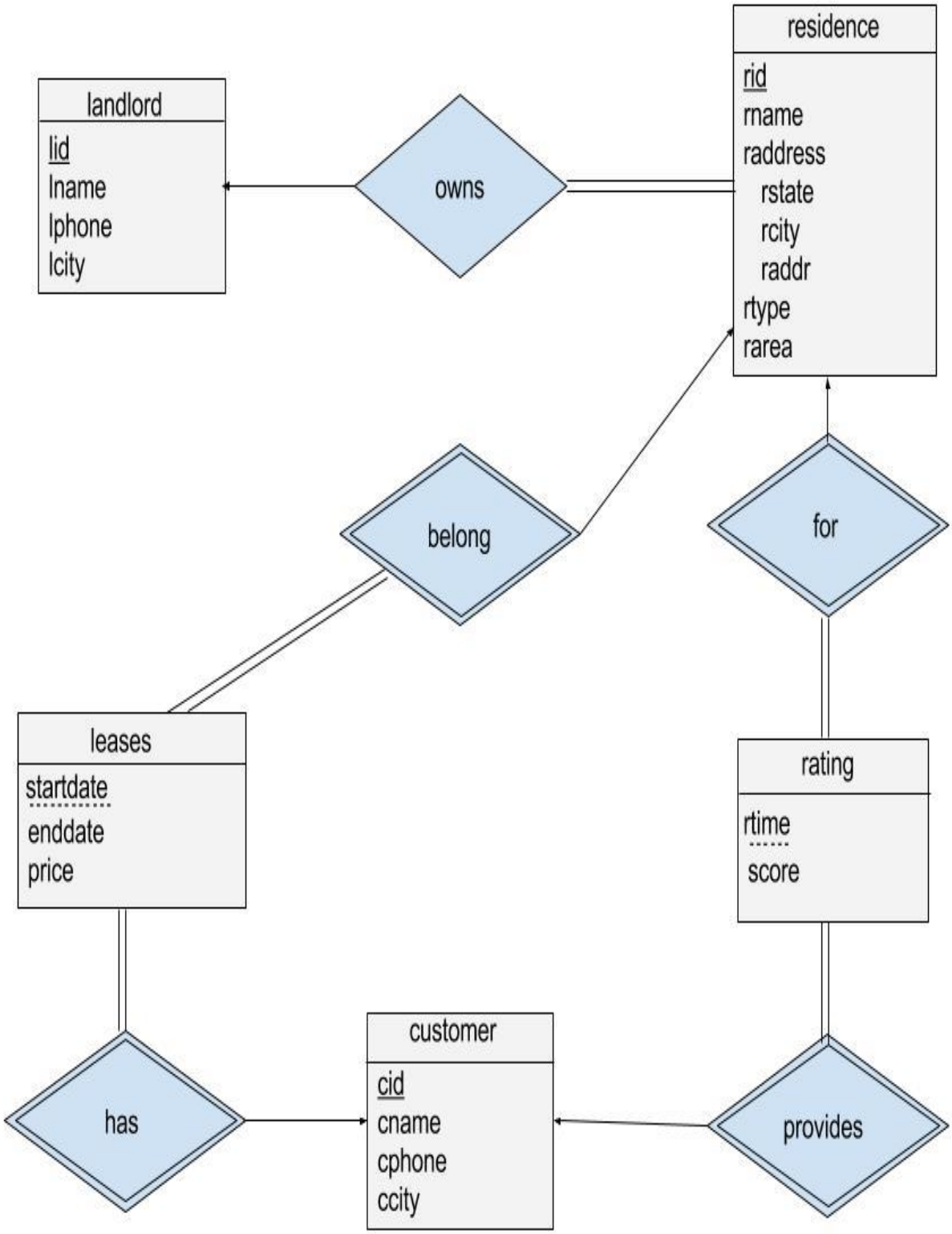
Customer (cid, cname, cphone, ccity);
Landlord (lid, lname, lphone, lcity)
Residence (rid, rname, rstate, rcity, raddr, rtype, rarea, lid);
Leases (cid, rid, startdate, enddate, price);
Rating (cid, rid, rtime, score);

Customers are identified by a cid, and we also store their name, phone number, and the city they live in. Landlords are identified by a lid, and have a name, phone number, and city. One landlord could own multiple houses/apartments, but a house could only have one owner. Residences are identified by a rid, and have an rname, and rstate, rcity, and raddr to store the precise address of the house (e.g. rcity = 'Brooklyn', raddr = '3rd floor, 308 45st, 6 Avenue'), an rtype that stores the type of residence (e.g studio or 2BR-1BA), and rarea indicating the square footage. Leases are identified by cid, rid, lid, and startdate, and we also store enddate and price. The startdate and enddate attributes store both time and date information. When the lease ends, customers could give a rating to this house, where a rating contains cid, rid, rtime, and a score. rtime is a timestamp which indicates the date and time when the customer made the rating; scores are ranging between 1 star (Terrible!) and 5 stars (Awesome!).

(a) Suppose the *Residence* table did not have the attribute rid, can you identify a new primary key? If you can, identify a new primary key; if not, write down the reasons.
Ans: The new primary key for Residence table will be (rname, rstate, rcity, raddr)

(b) Identify suitable foreign-key relationships for the above tables.
Ans: The foreign-key relationships for the above tables are as follows:
Residence.lid references Landlord.lid
Leases.cid references Customer.cid
Leases.rid references Residence.rid
Rating.cid references Customer.cid
Rating.rid references Residence.rid

(c) Draw an ER diagram that is consistent with the above relational schema. Show the cardinalities of the relationships, and identify any weak entities.
Ans:



Weak Entities: leases, rating

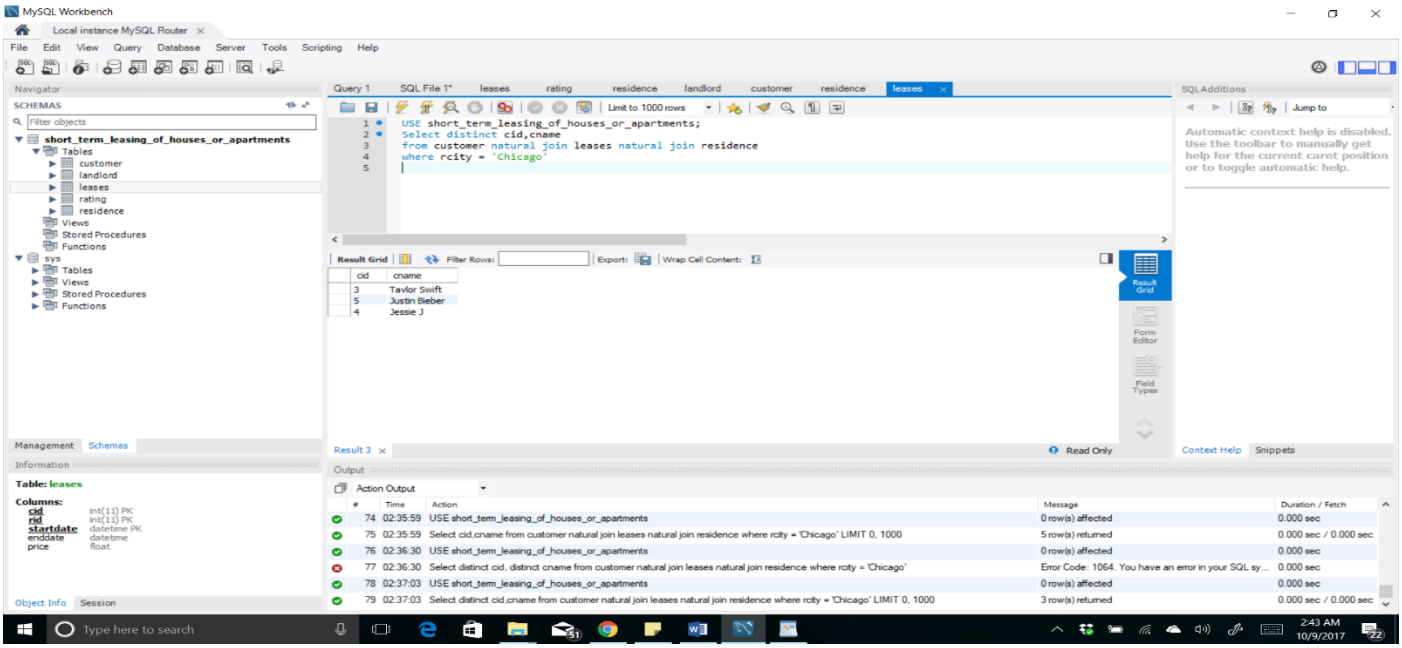
(d) Create the above schema in a database system, choose appropriate attributes types, and define primary keys, foreign keys, and other constraints. Data for this schema will be made available on NYU Classes; please use that data. Load the data into the database using either insert statements or the bulk-loading facilities. You may use any mainstream relational database system.

(e) Write the following SQL queries and execute them in your database. Show the queries and the results:

(i) List the cids and cnames of all customers who have rented residences in Chicago.

Ans:

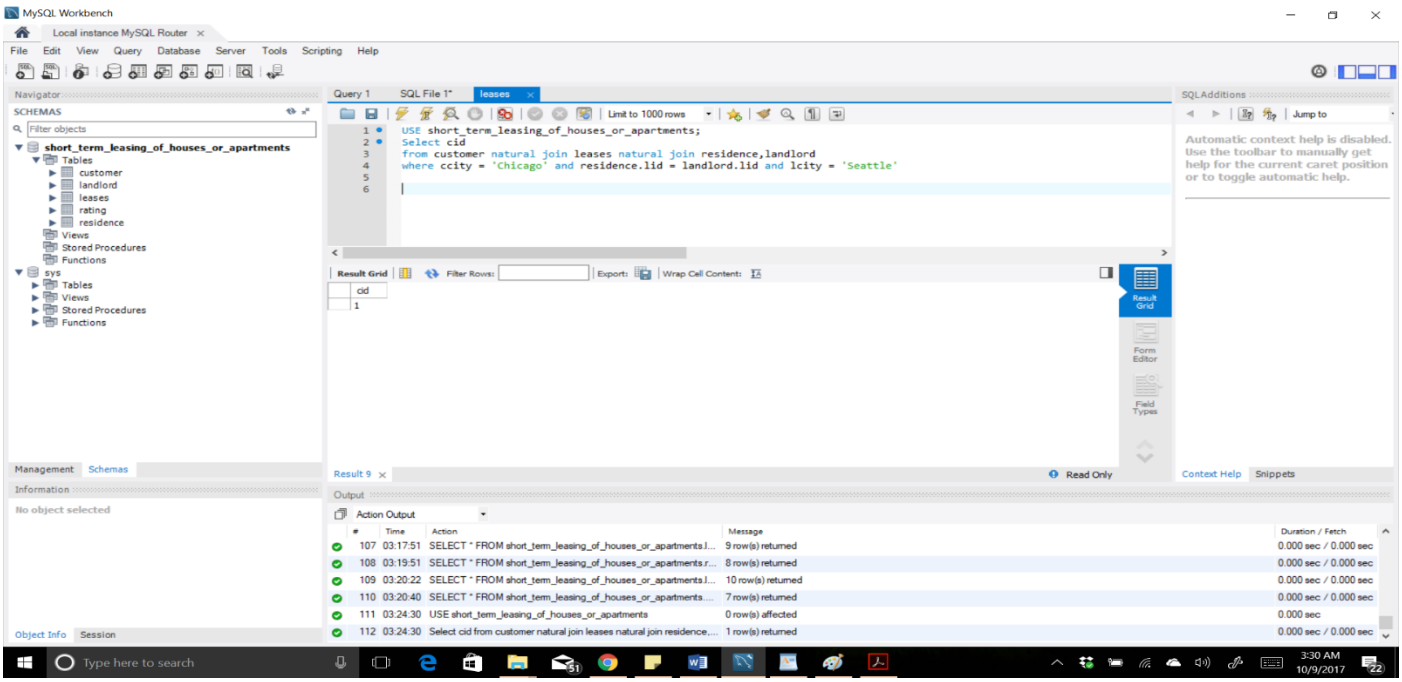
```
USE short_term_leasing_of_houses_or_apartments;
Select distinct cid, cname
from customer natural join leases natural join residence
where rcity = 'Chicago'
```



(ii) List the cids of any customers living in Chicago who had a lease with a landlord living in Seattle.

Ans:

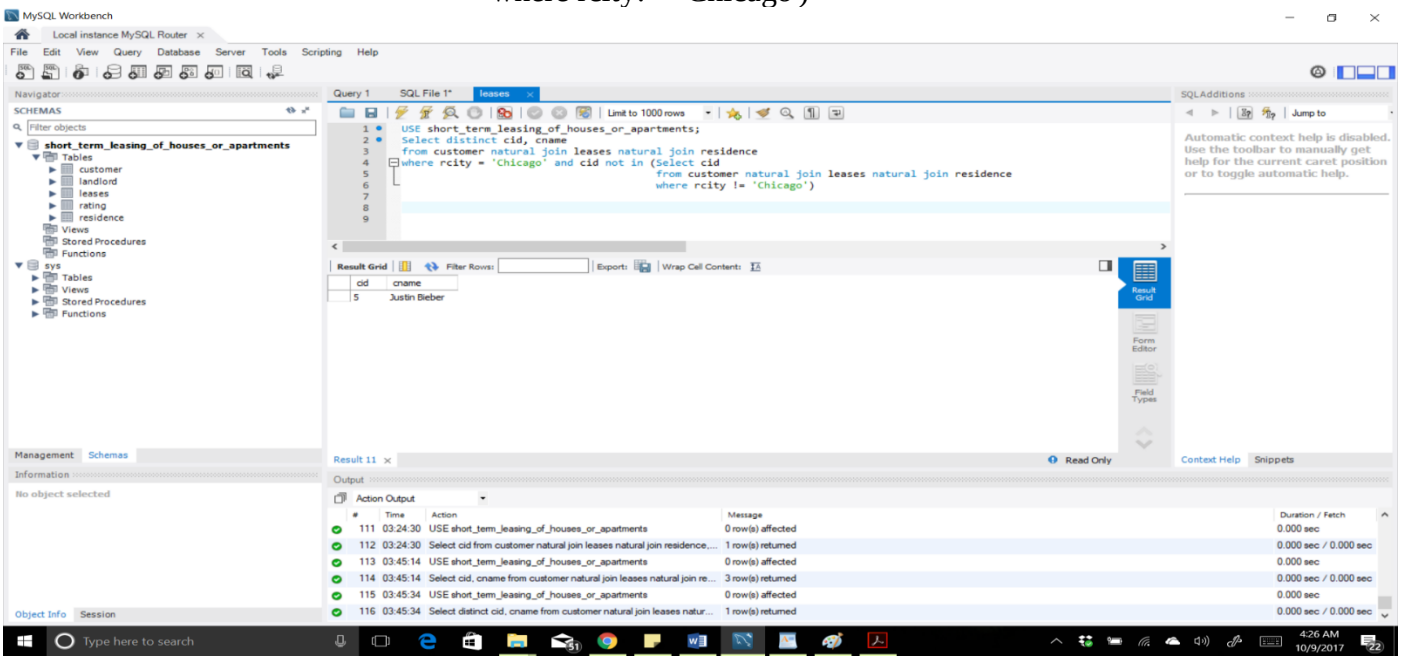
```
USE short_term_leasing_of_houses_or_apartments;
Select cid
from customer natural join leases natural join residence, landlord
where rcity = 'Chicago' and residence.lid = landlord.lid and lcity = 'Seattle'
```



(iii) Output the cid and cname of any customers who have only rented residences in Chicago. (That is, they have rented at least one residence in Chicago, and have not rented anything anywhere else.)

Ans:

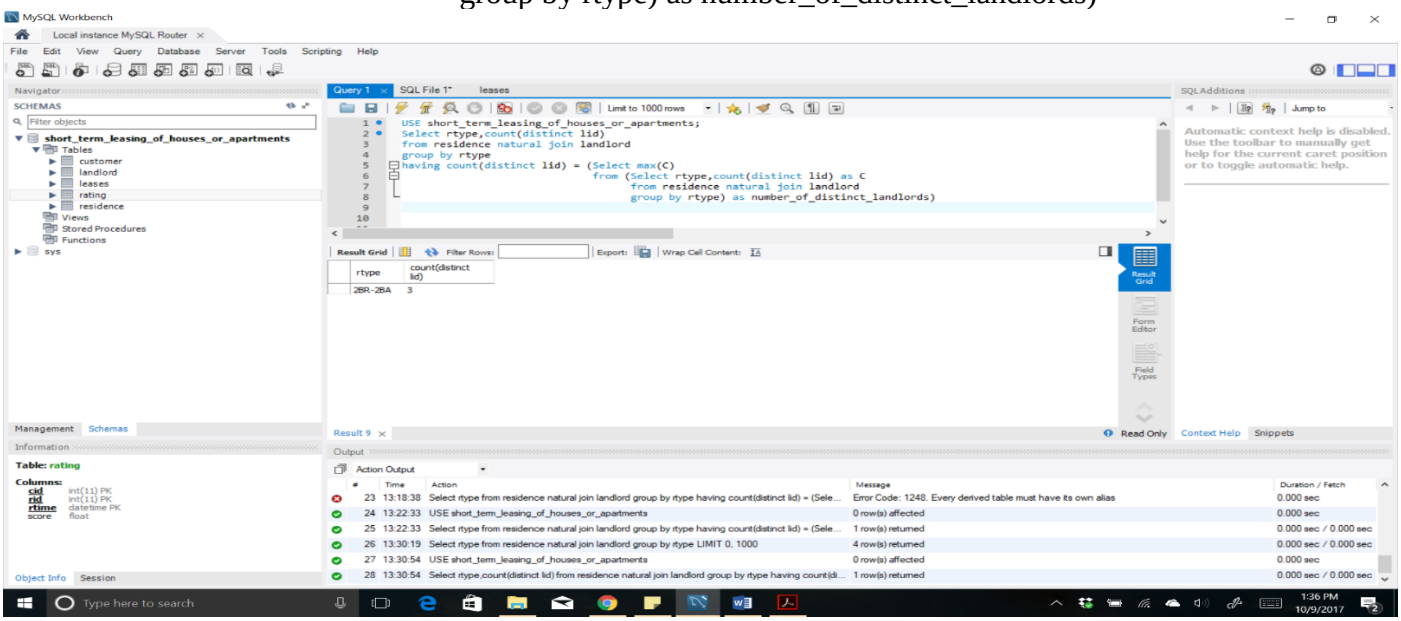
```
USE short_term_leasing_of_houses_or_apartments;
Select distinct cid, cname
from customer natural join leases natural join residence
where rcity = 'Chicago' and cid not in (Select cid
                                        from customer natural join leases natural join residence
                                        where rcity! = 'Chicago')
```



(iv) Output the residence type that is owned by the largest number of distinct landlords.

Ans:

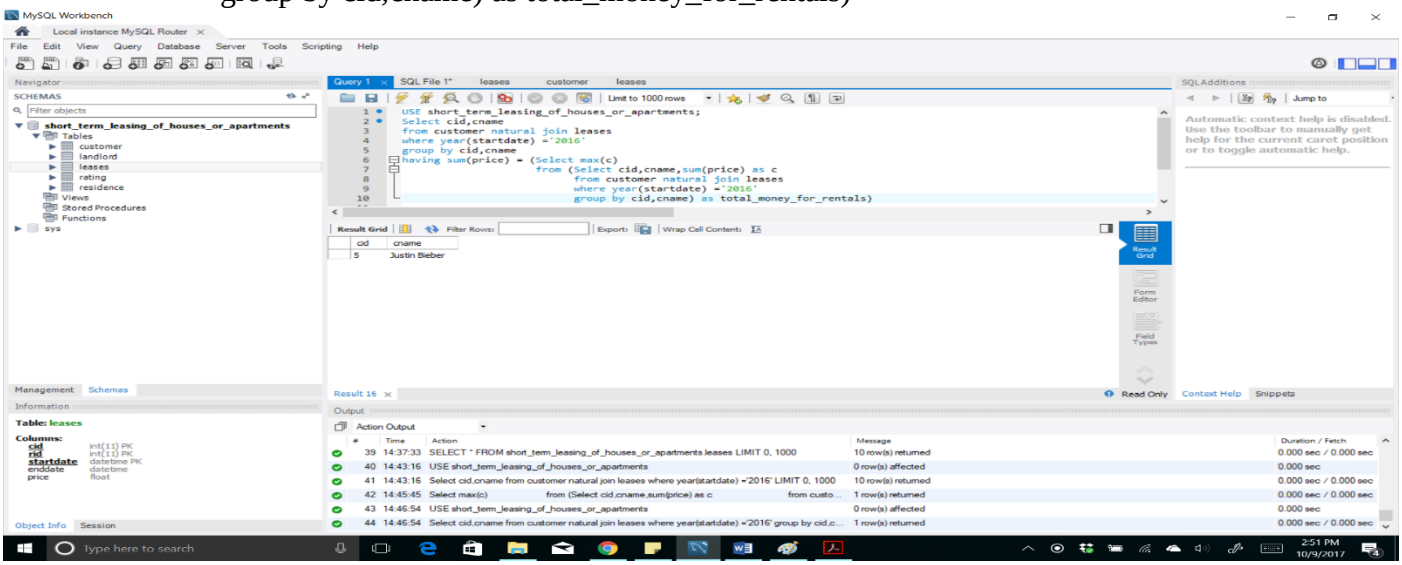
```
USE short_term_leasing_of_houses_or_apartments;
Select rtype, count (distinct lid)
from residence natural join landlord
group by rtype
having count (distinct lid) = (Select max(C)
                               from (Select rtype, count (distinct lid) as C
                                     from residence natural join landlord
                                     group by rtype) as number_of_distinct_landlords)
```



(v) Output the cid and cname of the customer(s) who spent the most money (sum of prices) for rentals with starting dates in 2016.

Ans:

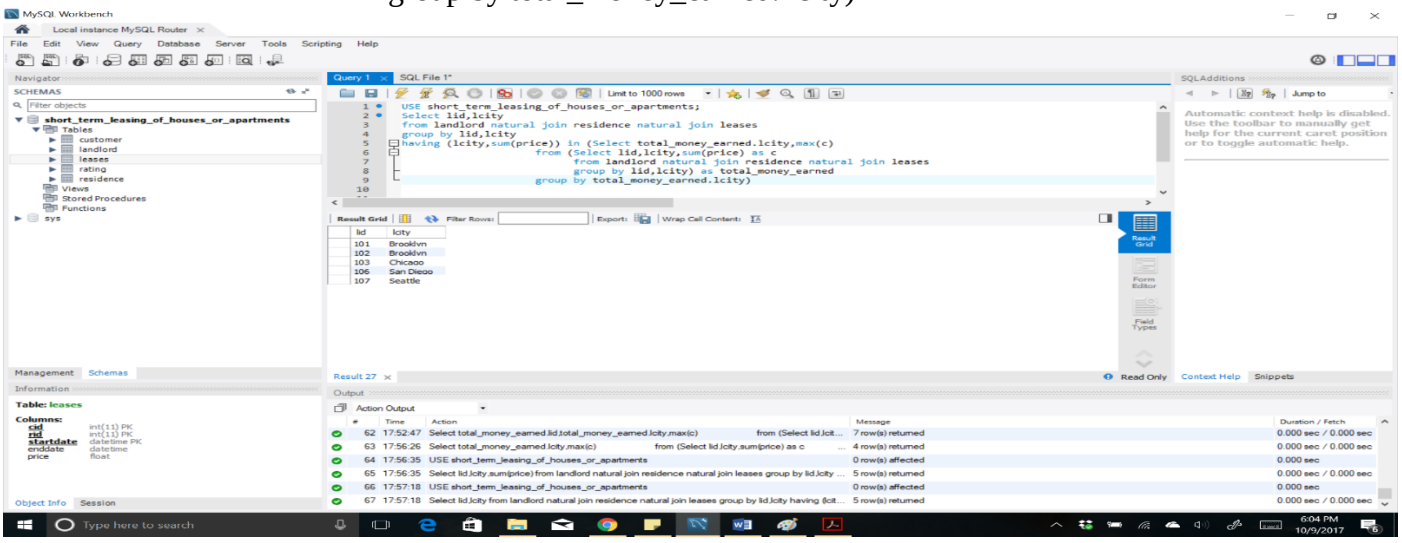
```
USE short_term_leasing_of_houses_or_apartments;
Select cid,cname
from customer natural join leases
where year(startdate) ='2016'
group by cid,cname
having sum(price) = (Select max(c)
                     from (Select cid, cname, sum(price) as c
                           from customer natural join leases
                           where year(startdate) ='2016'
                           group by cid,cname) as total_money_for_rentals)
```



(vi) For each lcity, output the lids of the landlords living in that city who made the most money overall.

Ans:

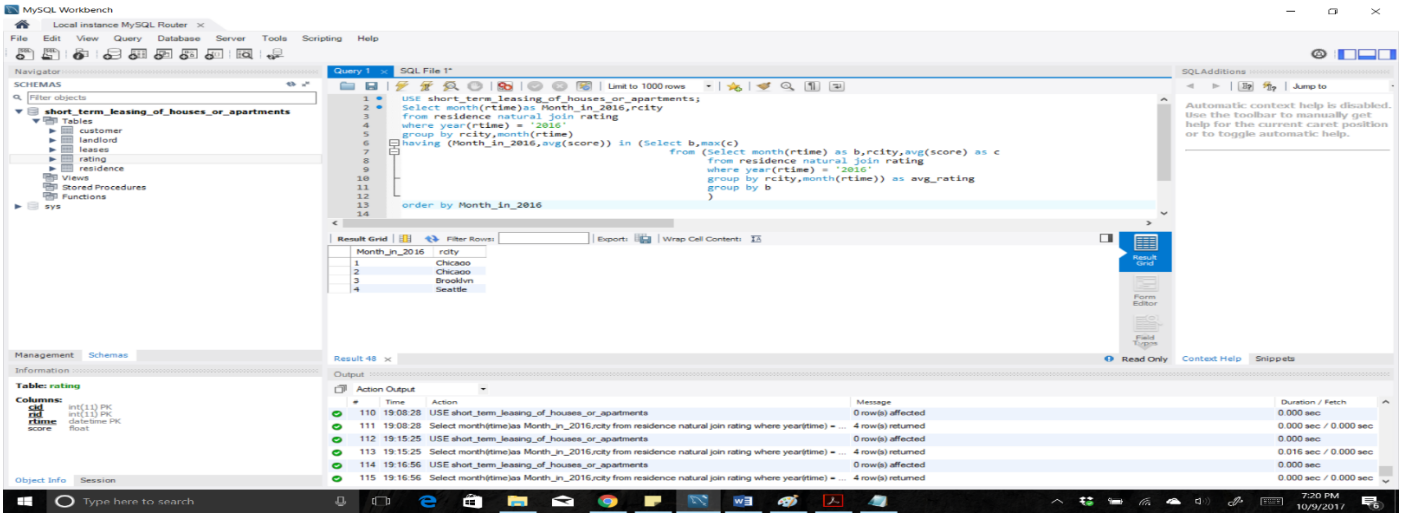
```
USE short_term_leasing_of_houses_or_apartments;
Select lid, lcity
from landlord natural join residence natural join leases
group by lid, lcity
having (lcity, sum(price)) in (Select total_money_earned.lcity,max(c)
                               from (Select lid,lcity,sum(price) as c
                                     from landlord natural join residence natural join leases
                                     group by lid,lcity) as total_money_earned
                               group by total_money_earned. lcity)
```



(vii) For each month in 2016, output the rcity whose houses had the best average rating, based only on ratings that were given during that month.

Ans:

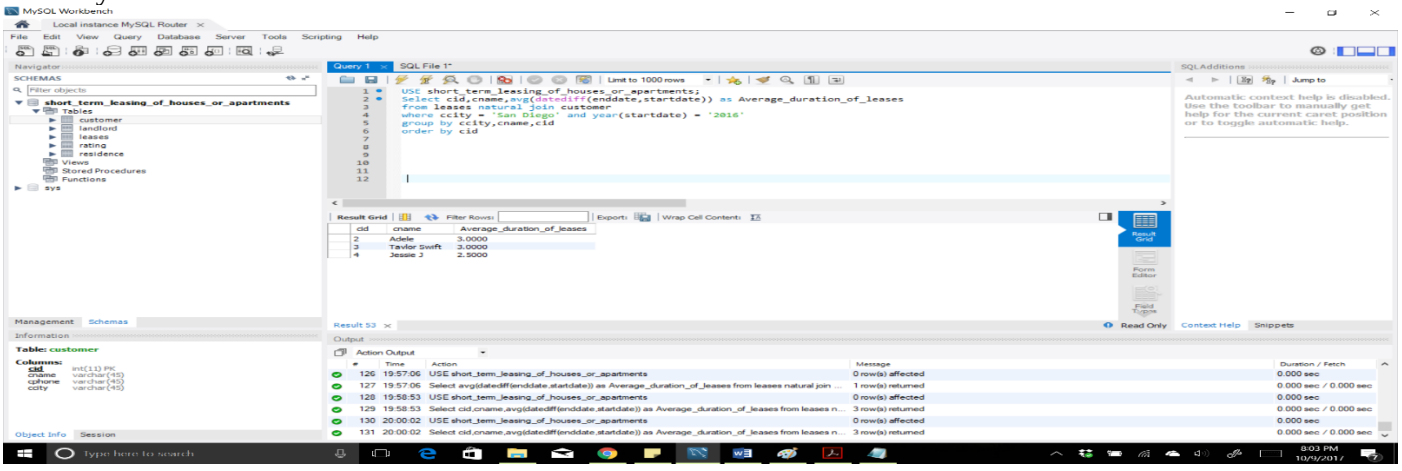
```
USE short_term_leasing_of_houses_or_apartments;
Select month(rtime)as Month_in_2016, rcity
from residence natural join rating
where year(rtime) = '2016'
group by rcity, month(rtime)
having (Month_in_2016, avg(score)) in (Select b,max(c)
                                       from (Select month(rtime) as b,rcity,avg(score) as c
                                             from residence natural join rating
                                             where year(rtime) = '2016'
                                             group by rcity, month(rtime)) as avg_rating
                                       group by b)
order by Month_in_2016
```



(viii) Output the average duration of leases (enddate - startdate) for customers living in San Diego, for leases with start date in 2016. The unit of average time should be days.

Ans:

```
USE short_term_leasing_of_houses_or_apartments;
Select cid, cname, avg (datediff (enddate, startdate)) as Average_duration_of_leases
from leases natural join customer
where ccity = 'San Diego' and year(startdate) = '2016'
group by ccity, cname, cid
order by cid
```



(f) Write expressions in Relational Algebra for queries (iii) to (viii).
Ans:

(iii) $R1 \leftarrow \pi_{cid,cname}(\sigma_{rcity \neq \text{'Chicago'}}(customer \bowtie leases \bowtie residence))$

$R2 \leftarrow \pi_{cid,cname}(\sigma_{rcity = \text{'Chicago'}}(customer \bowtie leases \bowtie residence))$

$\pi_{cid,cname}(R2 - R1)$

(iv) $\rho_{number\_of\_distinct\_landlords\ (rtype,c)}(_{rtype}G_{count(distinct\ lid)}(residence \bowtie landlord))$

$\rho_{R1(b)}(_{}G_{max(c)}(number\_of\_distinct\_landlords))$

$\rho_{R2(rtype,\ b)}(_{rtype}G_{count\ (distinct\ lid)}(residence \bowtie landlord))$

$\pi_{rtype,b}(R1 \bowtie R2)$

(v) $\rho_{total\_money\_for\_rentals\ (cid,cname,c)}(_{cid,cname}G_{sum(price)}(\sigma_{year(startdate)= \text{'2016'}}(customer \bowtie leases)))$

$\rho_{R1(b)}(_{}G_{max(c)}(total\_money\_for\_rentals))$

$\rho_{R2(cid,cname,b)}(_{cid,cname}G_{sum(price)}(\sigma_{year(startdate)= \text{'2016'}}(customer \bowtie leases)))$

$\pi_{cid,cname}(R1 \bowtie R2)$

(vi) $\rho_{total\_money\_earned\ (lid,\ lcity,\ c)}(_{lid,\ lcity}G_{sum(price)}(landlord \bowtie residence \bowtie leases))$

$\rho_{T1\ (a,b)}(_{lcity}G_{max(c)}(total\_money\_earned))$

$\rho_{T2\ (lid,\ lcity,\ c)}(_{lid,\ lcity}G_{sum(price)}(landlord \bowtie residence \bowtie leases))$

$\pi_{lid,\ lcity}(\sigma_{T1.b = T2.c}(T1 \times T2))$

(vii) $\rho_{avg\_rating\ (rcity,\ b,\ c)}(_{rcity,\ month(rtime)}G_{avg(score)}(\sigma_{year(rtime) = \text{'2016'}}(residence \bowtie rating)))$

$\rho_{R1(b,d)}(b\ G_{max(c)(avg\_rating)})$

$\rho_{R2\ (rcity,\ Month\_in\_2016,\ d)}(rcity,\ _{month(rtime)}G_{avg(score)}(\sigma_{year(rtime) = \text{'2016'}}(residence \bowtie rating)))$

$\pi_{rcity,\ Month\_in\_2016}(R1 \bowtie R2)$

(viii)

ρduration_of_lease (ccity, cname, cid, Average_duration_of_lease) (ccity, cname,cid Gavg

(datediff(enddate, startdate)) (σ ccity= ‘San Diego’∧ year(startdate) = ‘2016’ (leases natural join customer)))

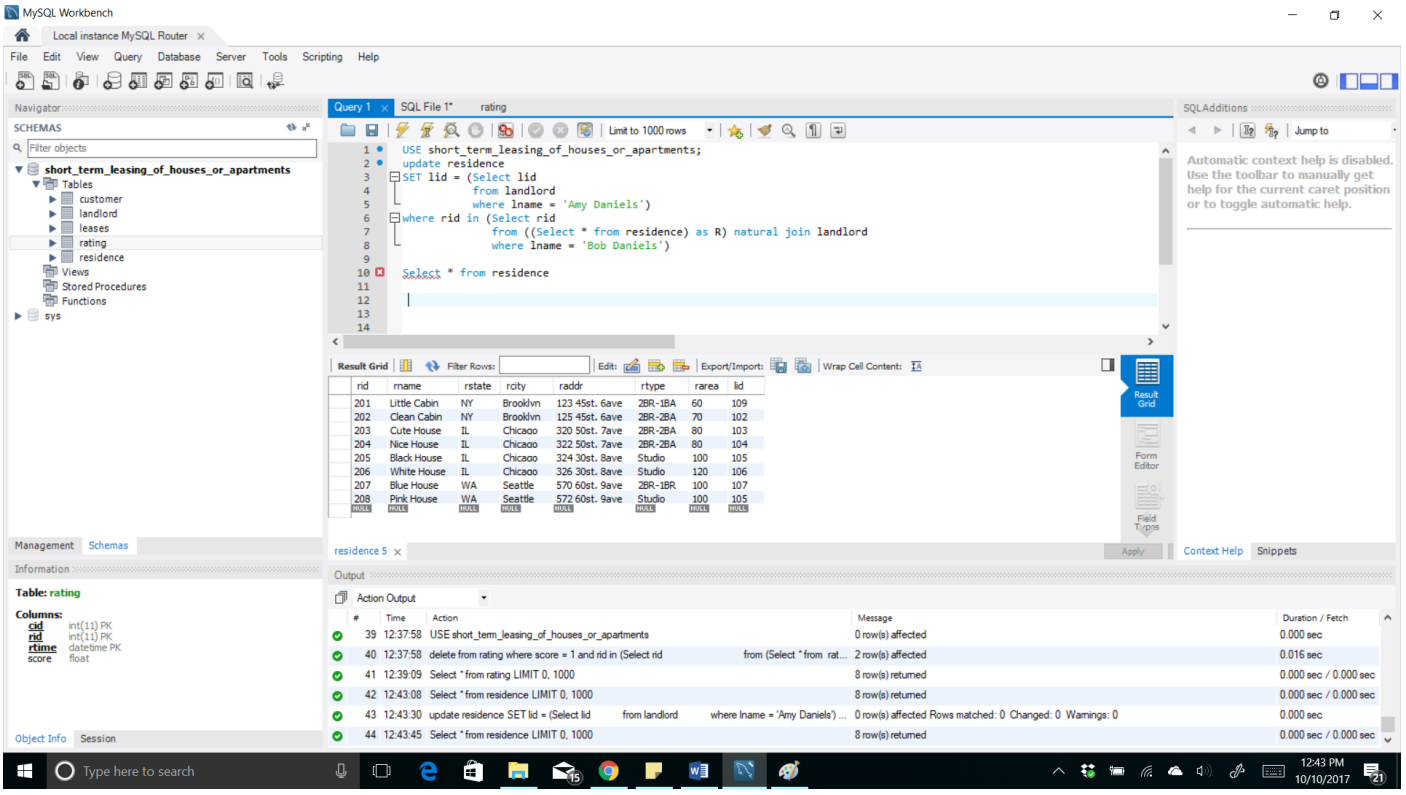
π cid,cname, Average_duration_of_lease (duration_of_lease)

(g) Write SQL statements to perform the following updates to the database:

(i) For every house whose landlord is named “Bob Daniels”, change it to “Amy Daniels”.

Ans:

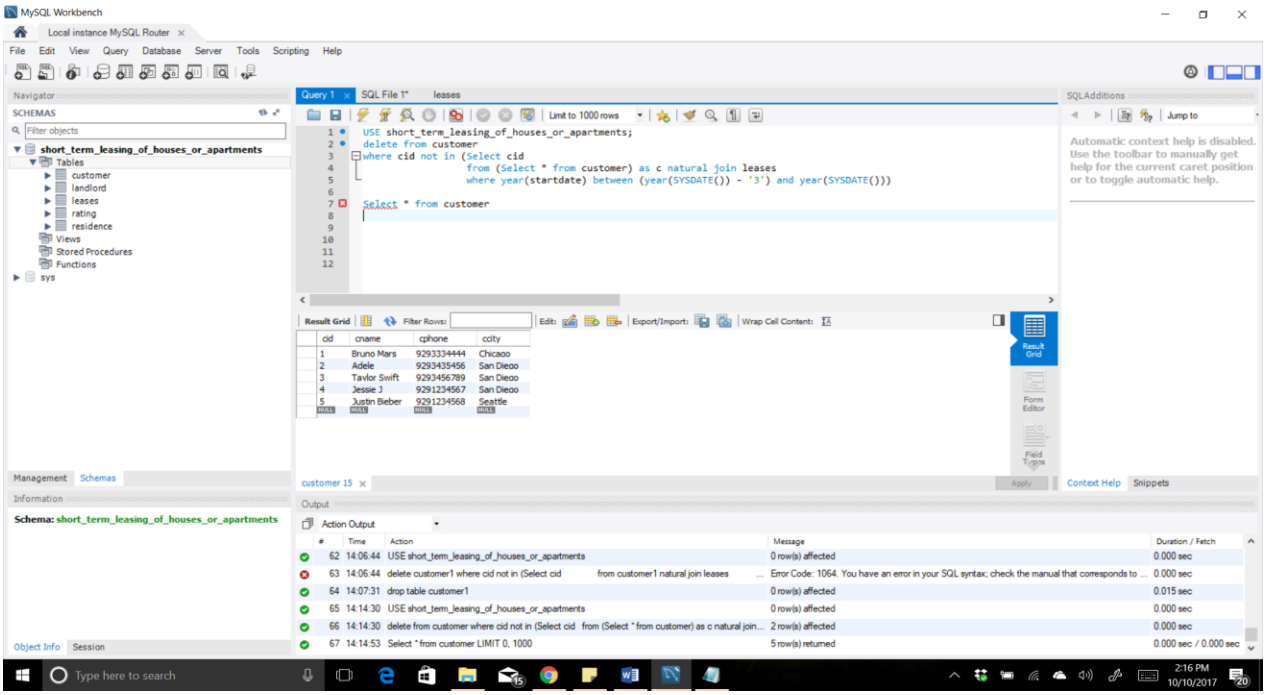
```
USE short_term_leasing_of_houses_or_apartments;
update residence
SET lid = (Select lid
        from landlord
        where lname = 'Amy Daniels')
where rid in (Select rid
          from ((Select * from residence) as R) natural join landlord
          where lname = 'Bob Daniels')
```



(ii) Delete tuples of any customer who has not made a lease in the last 3 years from the *Customer* table.

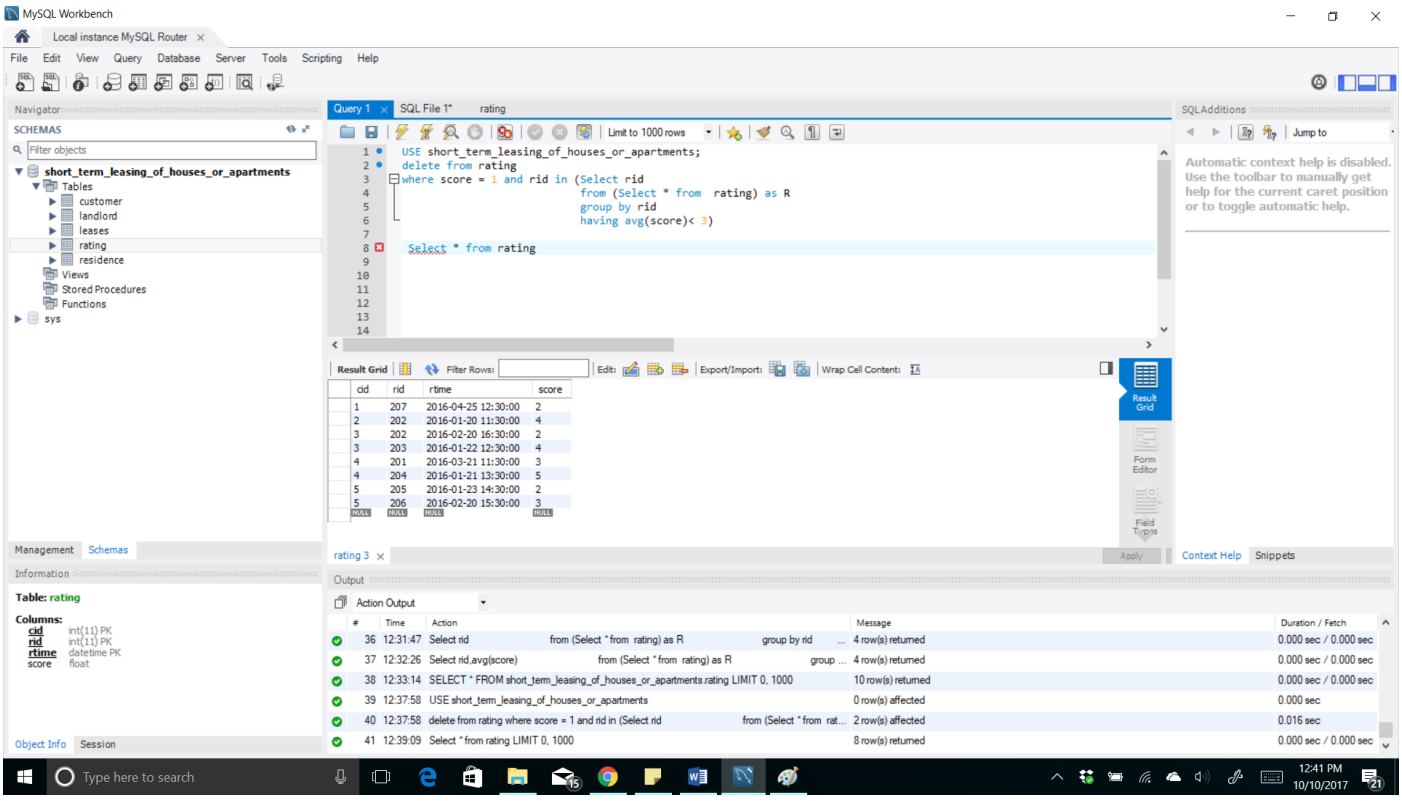
Ans:

```
USE short_term_leasing_of_houses_or_apartments;
delete from customer
where cid not in (Select cid
                  from (Select * from customer) as c natural join leases
                  where year(startdate) between (year (SYSDATE ()) - '3') and year(SYSDATE ()))
```



(iii) For any house with average rating below 3 stars, delete all of its rating records with rating 1 star.
Ans:

```
USE short_term_leasing_of_houses_or_apartments;
delete from rating
where score = 1 and rid in (Select rid
                            from (Select * from rating) as R
                            group by rid
                            having avg(score)< 3)
```

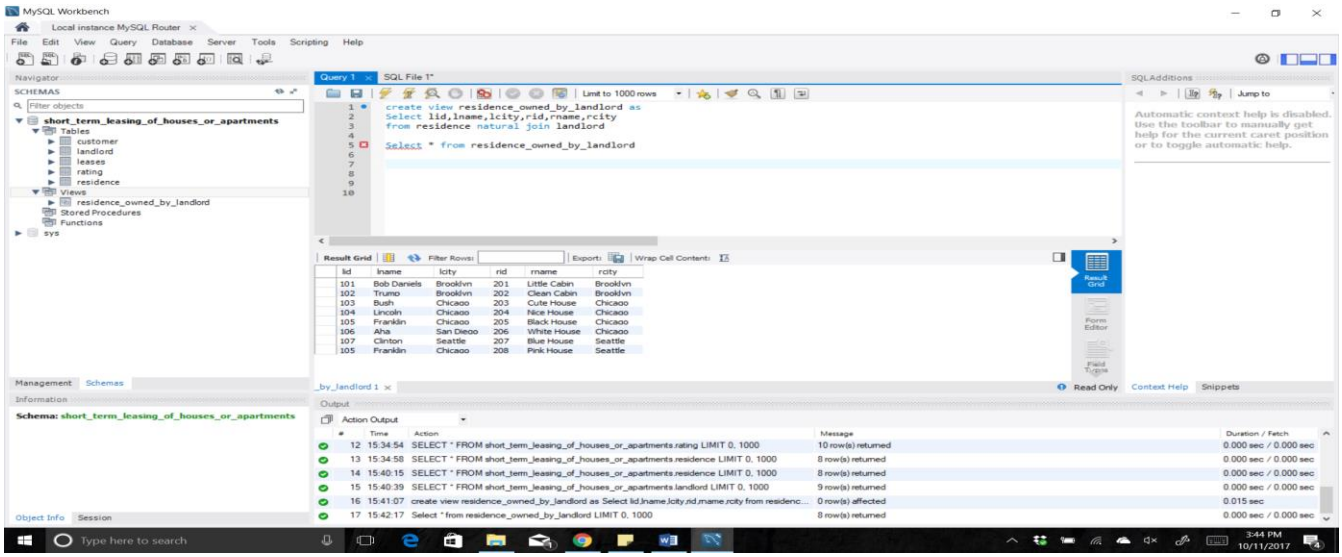


Problem 2:

In this problem, you have to create views and then write queries on the views, and create triggers, for the schema in Problem 1. Execute everything on your database system, and report the results.
(a) Define a view that contains for each residence the lid, lname, lcity, rid, rname, and rcity. Using this view, try to answer the following queries:

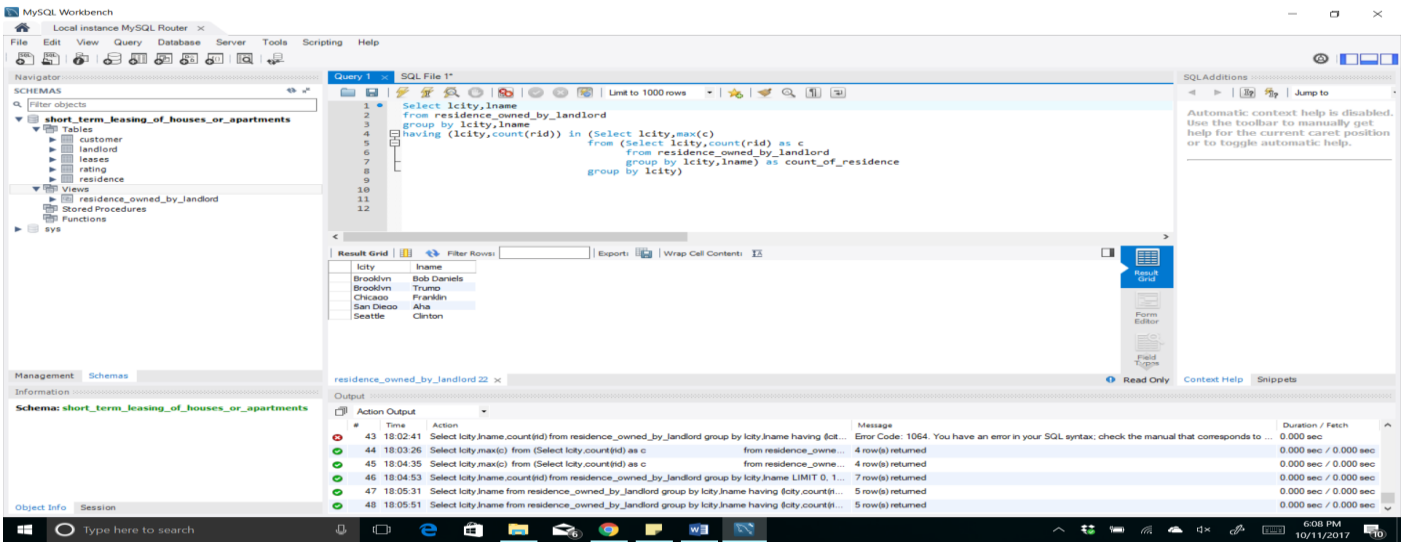
Query to create the view:

```
create view residence_owned_by_landlord as
Select lid, lname, lcity, rid, rname, rcity
from residence natural join landlord
```



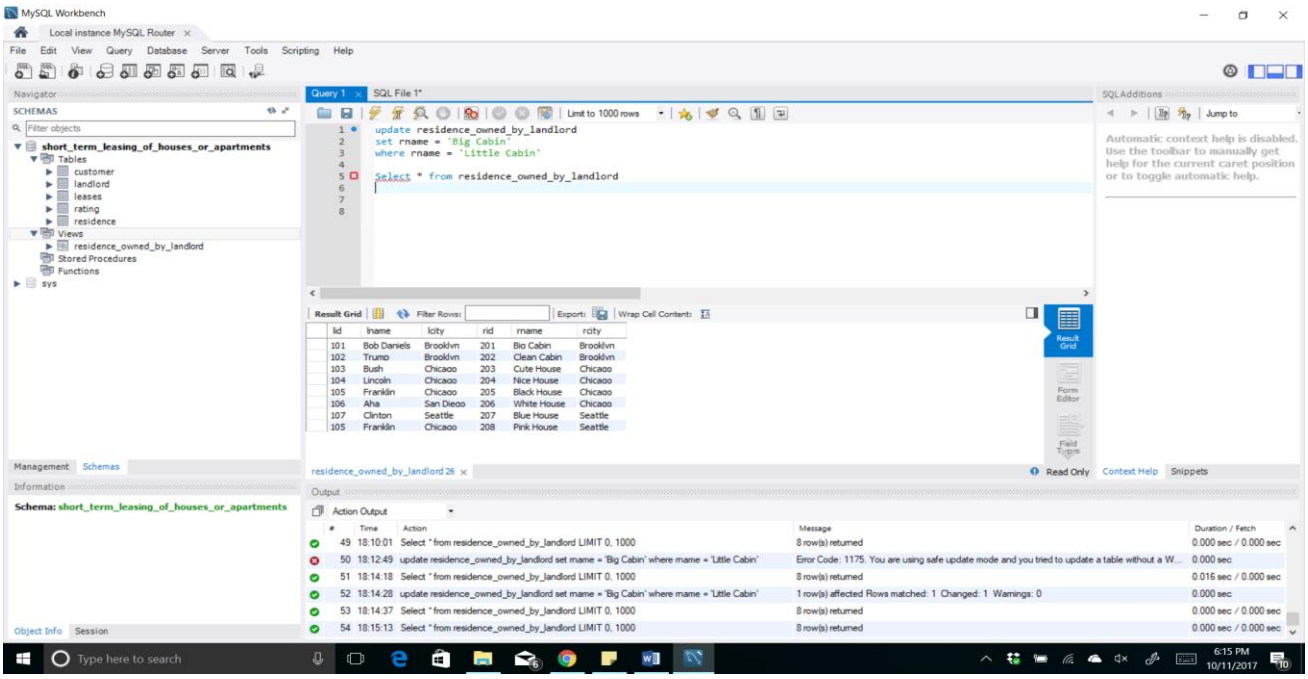
(i) For each lcity, output the lname that owns the most residences.
Ans:

```
Select lcity,lname
from residence_owned_by_landlord
group by lcity,lname
having (lcity,count(rid)) in (Select lcity,max(c)
                                from (Select lcity,count(rid) as c
                                from residence_owned_by_landlord
                                group by lcity,lname) as count_of_residence
                                group by lcity)
```



(ii) Change the name of any residence with name “Little Cabin” to “Big Cabin”.
Ans:

```
update residence_owned_by_landlord
set rname = 'Big Cabin'
where rname = 'Little Cabin'
```



(b) Consider the last query in Problem 1(g). Can you write a trigger that automatically deletes ratings with rating 1 star whenever a residence’s average rating drops below 3 stars? If yes, show the trigger. If no, explain why not.
Ans: We can’t write a trigger for the above SQL query.
Solution: The average rating may change when a tuple is inserted, deleted or updated in the ratings table. So, in the above case, we must create three triggers to handle all the three operations i.e. insert, delete, update. The triggers to be created are as shown below.

For insert operation:

```
DELIMITER \\
create trigger check_average after insert on rating
for each row
BEGIN
delete from rating
    where score = 1 and rid in (Select rid
                                from (Select * from rating) as R
                                group by rid
                                having avg(score)< 3);
END \\
```

For delete operation:

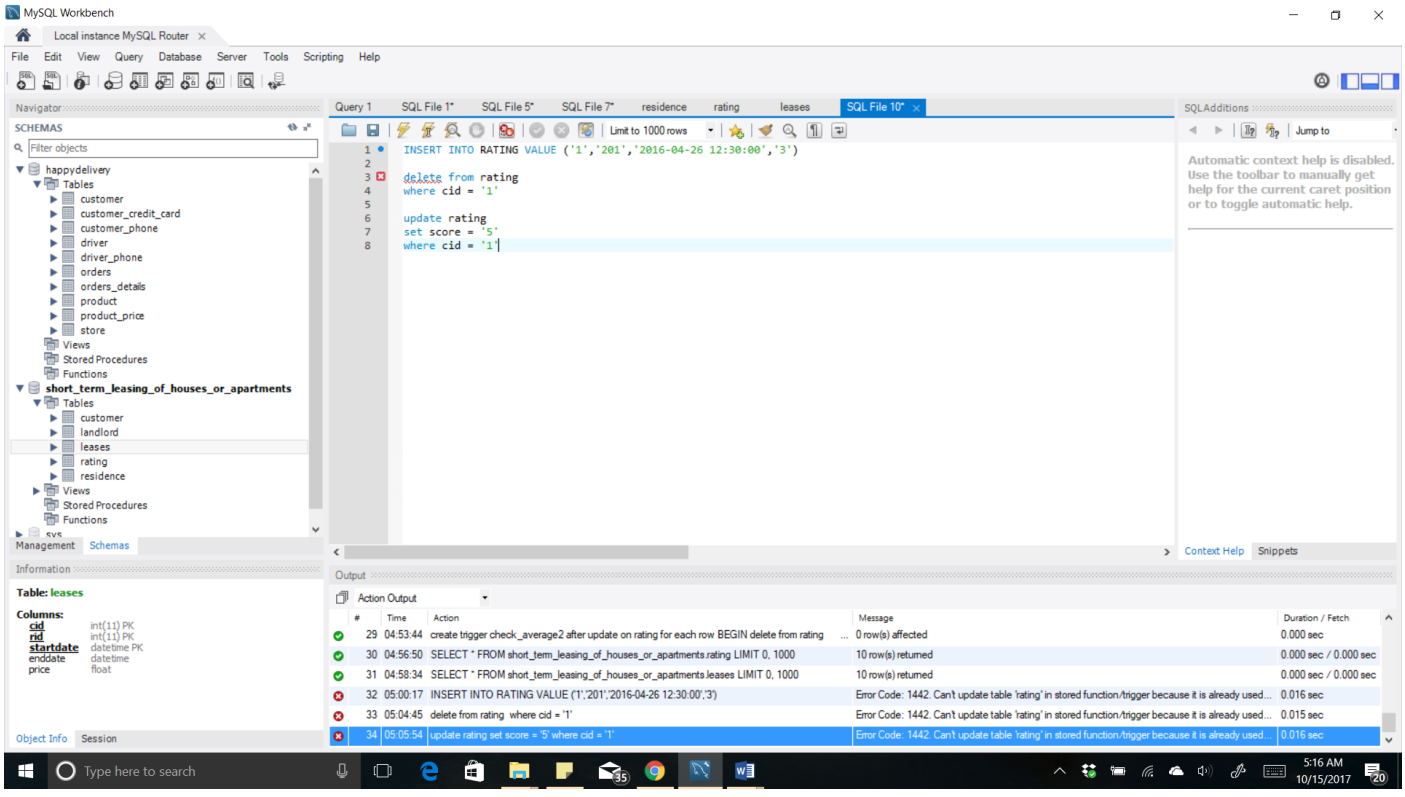
```
DELIMITER \\
create trigger check_average1 after delete on rating
for each row
BEGIN
```

```
delete from rating
    where score = 1 and rid in (Select rid
                                from (Select * from rating) as R
                                group by rid
                                having avg(score)< 3);
END \\
```

For update operation:

```
DELIMITER \\
create trigger check_average2 after update on rating
for each row
BEGIN
delete from rating
    where score = 1 and rid in (Select rid
                                from (Select * from rating) as R
                                group by rid
                                having avg(score)< 3);
END \\
```

SQL Queries after executing the triggers:



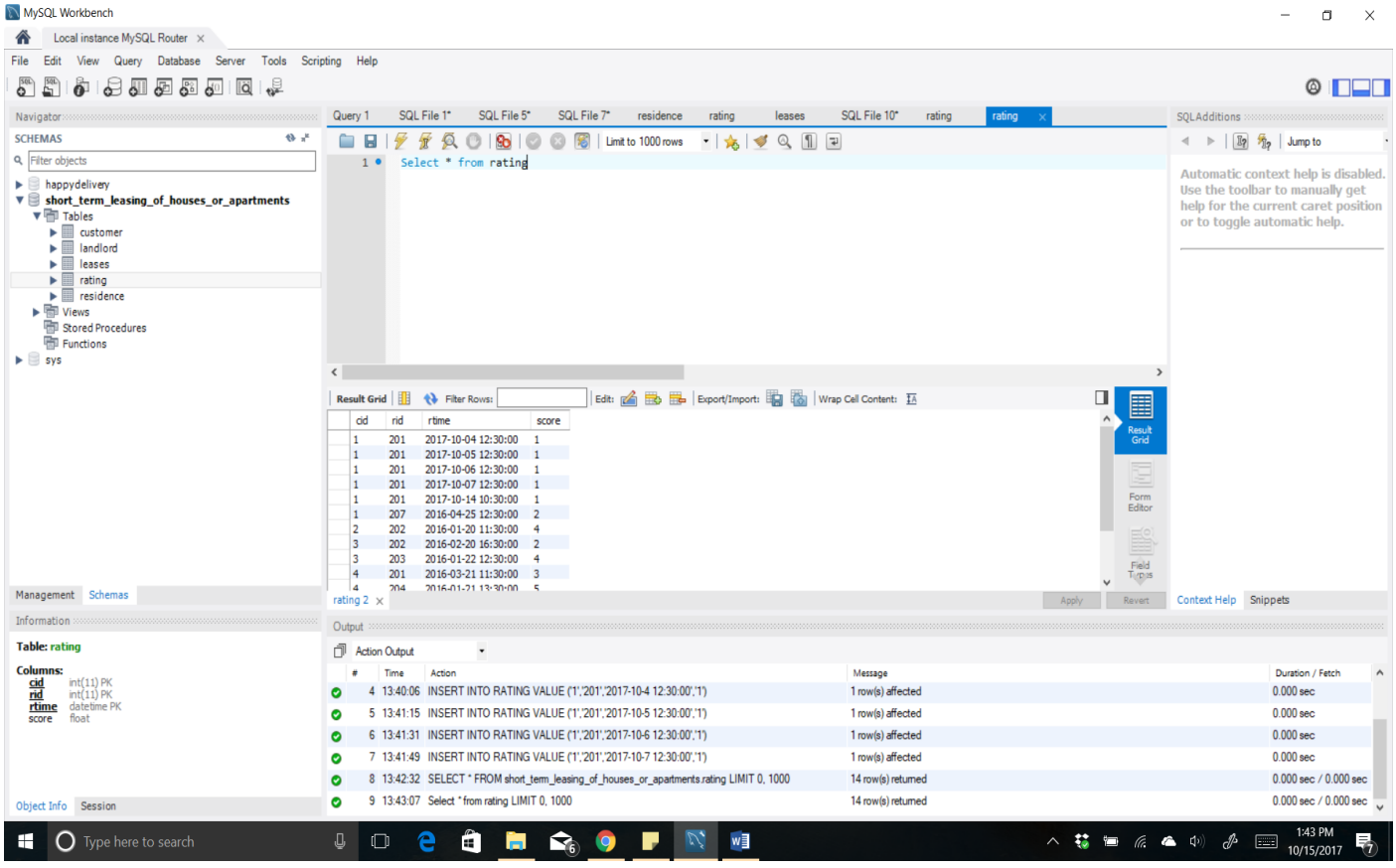
As we can see from the above screenshot, MySQL is not allowing me to insert, delete or update any value in the rating table and throwing an error which states that we can't insert delete or update the values in the same table which invoked the trigger i.e. rating table in this case.

(c) Write a trigger that prevents additional ratings for any customer who has already done five 1-star ratings in the past 2 weeks.

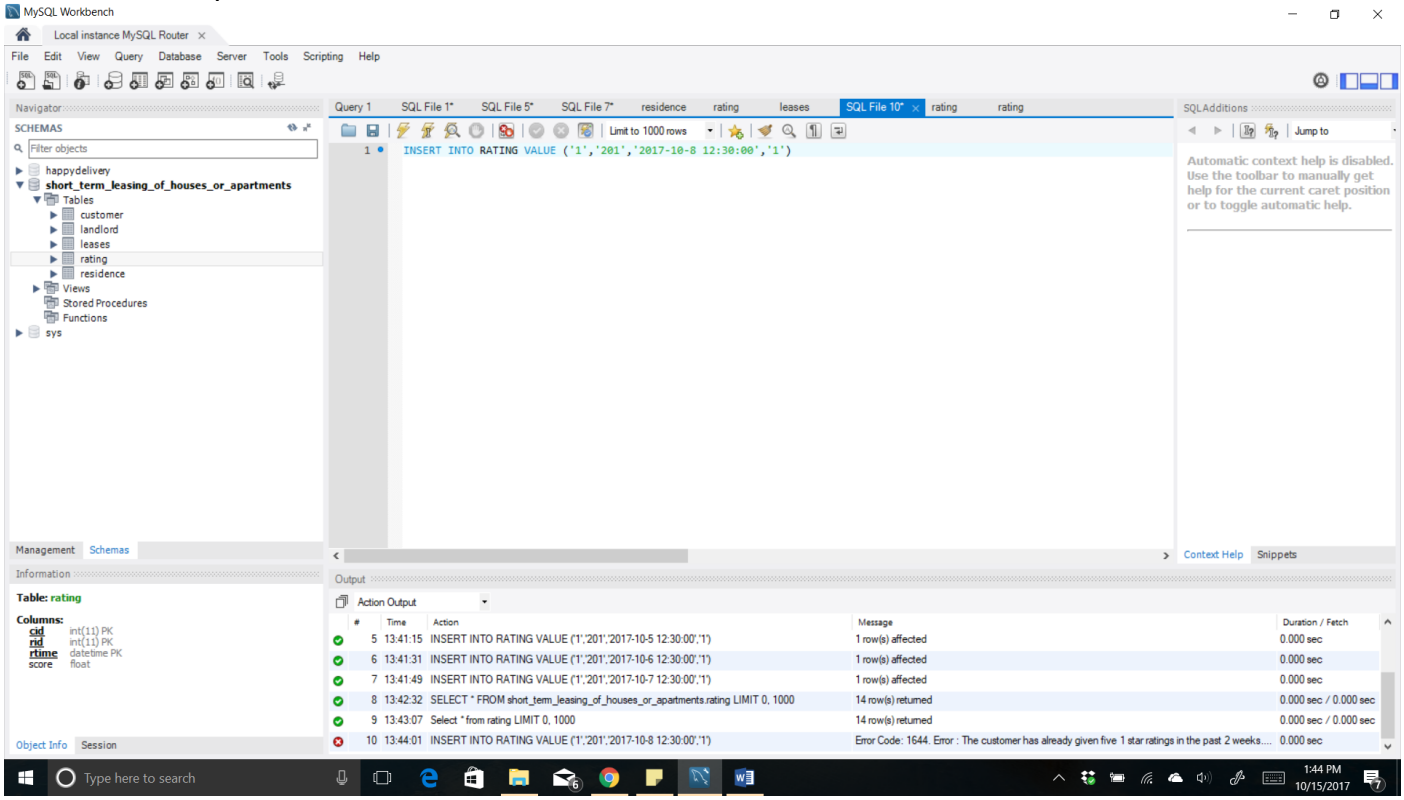
Ans: **Trigger executed**:

```
DELIMITER \\
create trigger additional_entry before insert on rating
for each row
BEGIN
  IF(new.cid in (Select cid
          from (Select * from rating) as R
          where score = '1' and DATEDIFF(SYSDATE(),rtime) <='14'
          group by cid
          having count(*) = '5'))
  THEN
  signal sqlstate '45000' set message_text = 'Error: The customer has already given five 1 star ratings in the past 2 weeks.Cannot accept this entry';
END IF;
END \\
```

Output of the rating table before the trigger prevents additional entries into the rating table i.e when five 1-star entries in the past 2 weeks of cid = '1' are inserted.



When we try to insert the sixth 1-star entry of cid = ‘1’ the error message defined in the trigger is displayed and the insert is prevented.

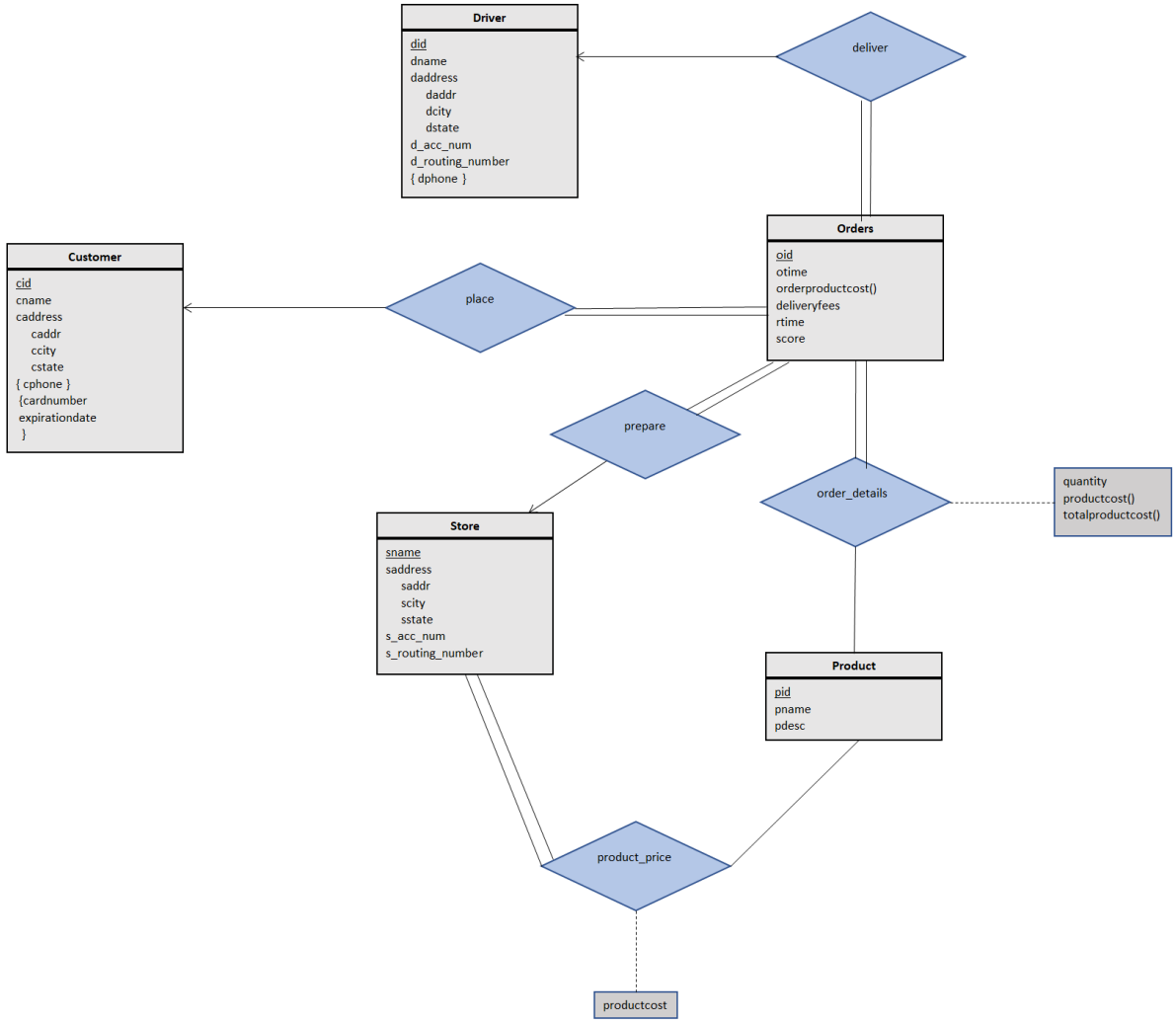


Problem 3:

In this problem, you are asked to design the relational backend for a company called HappyDelivery. The idea is that sometimes people want to buy products from a convenience store without leaving their home, but not every convenience store provides delivery service. HappyDelivery is making this possible, by maintaining product and price information for many participating chains of stores. Thus, a customer can visit their site, select some products for purchase, and pay the bill by using a credit card. After the order is successfully placed, the convenience store will prepare the order for pickup. When the order is ready, HappyDelivery will send a driver to pick up the products and deliver them to the customer. After the delivery is done, HappyDelivery credits the delivery fee to the driver’s account and adds the purchase price to the convenience store’s account. (HappyDelivery makes profits by getting referral fees from the stores, but you do not have to model this part).

HappyDelivery assigns a unique ID to each driver, and the database also stores their real name, address and phone number. Customers are identified by a unique customer ID, and they also need to provide their name, phone number, address, and credit card number. Each credit card consists of a card number and an expiration date. A customer may have more than one credit card. An order may involve several items, for example three cans of coke and one donut, but all items must be from the same store (otherwise you need to place two orders). For each order, we need to specify which products were bought, the customer, the store, the driver who delivered, the cost of the products, the delivery fee, and which credit card was used. Each store (i.e., convenience store chain) has a unique name. Products are identified by product IDs that are globally unique – that is, two items in different stores have the same product ID if and only if they are in fact the same product -- and have a short description. Of course, different stores may charge different prices for the same product. After the deliver is done, the customer can rate the service on the order from one to five stars.

(a) Design an ER diagram that can model the above scenario. Identify suitable keys and the cardinalities of the relationships. Also identify any weak entities. Discuss any assumptions that you are making in your design.



Assumptions made:

1. A single order is delivered by a single driver only.
2. Driver d_acc_num and d_routing_number details and Store s_acc_num and s_routing_number details are given in Driver and Store tables which would help in transferring the money received on order delivery completion by happydelivery to the respective store and driver's account.
3. productcost of relationship set product_price stores the current price of the product of a store which will keep changing depending upon the discount received or due to inflation.
4. the productcost attribute of the relationship set order_details is a derived attribute which fetches the productcost attribute value from the product_price relationship set. The reason for maintaining productcost attribute in two different relationship sets is to maintain the integrity of data of past orders in case the productcost of any product changes in future.
5. The totalproductcost attribute of the relationship set order_details is also a derived attribute which calculates the total price of any particular product in the order_details relationship set i.e quantity * productcost.
6. The orderproductcost attribute of the orders entity stores the total cost of all the product for a order i.e sum of totalproductcost attribute of all the products in a particular order.
7. The rating for the order can be given only once and cannot be changed in future. The rating is stored in the score attribute of orders entity. The otime and rtime are the time when the order was placed and the time when the rating was given respectively.

(b) Convert your ER diagram into a relational schema. Show primary keys and foreign key constraints.
Ans: Relational Schema:
Driver (**did**, dname, daddr, dcity, dstate,d_acc_num,d_routing_number)
Driver_phone (**dphone**, did)
Customer (**cid**, cname, caddr, ccity, cstate)
Customer_phone (**cphone,** cid)
Customer_credit_card (**cardnumber**, cid, expirationdate)
Orders_details (**oid, pid**, quantity, productcost,totalproductcost)
Orders (**oid**, otime, cid, sname, did, orderproductcost, deliveryfees, cardnumber, rtime, score)
Store (**sname**, saddr, scity, sstate, s_acc_num, s_routing_number)
Product (**pid**, pname, pdesc)
Product_price (**pid, sname,** productcost)

Table	Primary Key
Driver	did
Driver_phone	dphone
Customer	cid
Customer_phone	cphone
Customer_credit_card	cardnumber
Order_details	oid, pid
Orders	oid
Store	sname
Product	pid
Product_price	pid, sname

Foreign Keys:
Driver_phone.did references Driver.did
Customer_phone.cid references Customer.cid
Customer_credit_card.cid references Customer.cid
Orders_details.oid references Orders.oid
Orders_details.pid references Product.pid
(Orders.cid, Orders.cardnumber) references (Customer_credit_card.cid, Customer_credit_card.cardnumber)
Orders.sname references Store.sname
Orders.did references Driver.did
Product_price.pid references Product.pid
Product_price.sname references Product.sname

(c) Write statements in SQL for the following queries. Note that if your schema does not allow you to answer a query, you may have to go back and change your design.

(i) For each driver, output her ID, name, and the total amount of delivery fees she has earned from doing deliveries for a convenience store named "Tom's Store".

Ans:
```
Select   did, dname, sum(deliveryfees) as total_amount_of_delivery_fees
    from driver natural join orders
    group by did, dname, sname
    having sname = 'Tom's Store'
```

(ii) Output the ID of any driver who has never received a rating of five stars.

Ans:
```
Select distinct did
    from Driver natural join Orders
    where did not in (Select distinct did
                      from Driver natural join Orders
                      where score = '5')
```

(iii) Output the name and ID of any customer who has ordered at least four times from the same store.

Ans:
```
Select cid, cname
    from Customer natural join Orders
```

```
group by cid, cname, sname
having count (*) >= 4
```

(iv) Output the name and ID of the customer(s) who spent the most money during 2016. Include both the product prices and the delivery fees in your calculation.

Ans: **Implemented in this schema by using triggers and calculating and storing values dynamically in the database system.**

Query for creating the trigger:

```
DELIMITER //
create trigger product_value before insert on orders_details
for each row
begin
set NEW.productcost = (Select productcost
                       from product_price natural join orders
              where pid = new.pid and oid = new.oid);
set NEW.totalproductcost = NEW.quantity * NEW.productcost;
update orders
set orderproductcost = orderproductcost + NEW.totalproductcost
where oid = new.oid;
END//
DELIMITER;
```

Explanation of the trigger:

The trigger product_value basically calculates values for productcost, totalproductcost, orderproductcost.

productcost: this attribute fetches value from the productcost attribute of the product_price table which is the price of the individual product.

The reason for using two separate tables to store the same value is that incase the price of the product changes in future due to some discount or inflation it will only be changed in the productcost attribute of the product_price table and not the orders_details table. This will maintain the integrity of data of the orders done in past.

totalproductcost: this attribute stores the total cost of a single product i.e the cost of the individual product multiplied by the quantity of that product that is to be purchased.

orderproductcost: this attribute stores the total product cost of the order i.e the sum of the totalproductcost of all the products being purchased in a single order.

Note: orderproductcost attribute is set to default value 0 or should not be null because if it is null the below statement in the trigger will just set null values.

```
set orderproductcost = orderproductcost + NEW.totalproductcost
```

Query for solving the problem:

```
Select cid,cname
from Customer natural join Orders
where year(otime) = '2016'
group by cid,cname
having sum(deliveryfees + orderproductcost) = (Select max(c)
                                               from (Select sum(deliveryfees + orderproductcost) as c
                                                     from Customer natural join Orders
                                                     where year(otime) = '2016'
                                                     group by cid,cname) as total_money_spent)
```

(d) Create the tables in your database system, and insert some sample data (maybe 5-10 tuples per table). Choose an interesting and meaningful data set. Then execute the queries from part (c). Submit your

sample data, and screenshots or logs of the queries and outputs.

Dataset:

```
CREATE DATABASE  IF NOT EXISTS `happydelivery`
USE `happydelivery`;

CREATE TABLE `customer` (
  `cid` int(11) NOT NULL,
  `cname` varchar(45) DEFAULT NULL,
  `caddr` varchar(45) DEFAULT NULL,
  `ccity` varchar(45) DEFAULT NULL,
  `cstate` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`cid`)
)

INSERT INTO `customer` VALUES (1,'Seo','373,92nd Street,Apt A-54,Brooklyn,NY-11209','Brooklyn','New York'),(2,'Rahul','373,92nd Street,Apt A-22,Brooklyn,NY-11209','Brooklyn','New York'),(3,'Rishi','373,92nd Street,Apt A-31,Brooklyn,NY-11209','Brooklyn','New York'),(4,'Bob','121 N. LaSalle Street Chicago, Illinois 60602','Chicago','Illinois'),(5,'Anoop','131 N. LaSalle Street Chicago, Illinois 60602','Chicago','Illinois');

CREATE TABLE `customer_credit_card` (
  `cid` int(11) DEFAULT NULL,
  `cardnumber` varchar(45) NOT NULL,
  `expirationdate` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`cardnumber`),
  KEY `cid` (`cid`),
  CONSTRAINT `customer_credit_card_ibfk_1` FOREIGN KEY (`cid`) REFERENCES `customer` (`cid`)
)

INSERT  INTO `customer_credit_card` VALUES (1,'1234567890123456','10/18'),
(5,'1928373645012345','05/19'),(2,'3456752458941267','05/18'),(1,'4567890123456789','09/19'),(2,'4678469256803452','11/19'),(2,'7902574582693571','04/19'),(3,'9087654321234567','09/19'),(4,'9273651586253468','06/18');

CREATE TABLE `customer_phone` (
  `cid` int(11) DEFAULT NULL,
  `cphone` varchar(45) NOT NULL,
  PRIMARY KEY (`cphone`),
  KEY `cid` (`cid`),
  CONSTRAINT `cid` FOREIGN KEY (`cid`) REFERENCES `customer` (`cid`) ON DELETE NO ACTION ON UPDATE NO ACTION
)

INSERT INTO `customer_phone` VALUES
(1,'+12018850791'),(1,'+12018860961'),(2,'+12017090821'),(3,'+12018900731'),(4,'+13128384567'),(5,'+13124560791'),(5,'+13124768901');




CREATE TABLE `driver` (
  `did` int(11) NOT NULL,
  `dname` varchar(45) DEFAULT NULL,
  `daddr` varchar(45) DEFAULT NULL,
  `dcity` varchar(45) DEFAULT NULL,
  `dstate` varchar(45) DEFAULT NULL,
```

```
  `d_acc_num` varchar(45) DEFAULT NULL,
  `d_routing_number` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`did`)
)

INSERT INTO `driver` VALUES (1,'Mike','375,92nd Street,Apt B-50,Brooklyn,NY-11209','Brooklyn','New
York','902345678','123456789'),(2,'John','375,92nd Street,Apt B-30,Brooklyn,NY-11209','Brooklyn','New
York','345678567','567456902'),(3,'Justin','375,92nd Street,Apt B-20,Brooklyn,NY-11209','Brooklyn','New
York','124563478','664764256'),(4,'Manish','141 N. LaSalle Street,Chicago, Illinois
60602','Chicago','Illinois','473658290','125983560'),(5,'Harish','151 N. LaSalle Street,Chicago, Illinois
60602','Chicago','Illinois','346789236','336545688');

CREATE TABLE `driver_phone` (
  `did` int(11) DEFAULT NULL,
  `dphone` varchar(45) NOT NULL,
  PRIMARY KEY (`dphone`),
  KEY `did` (`did`),
  CONSTRAINT `did` FOREIGN KEY (`did`) REFERENCES `driver` (`did`) ON DELETE NO ACTION ON
UPDATE NO ACTION
)

INSERT INTO `driver_phone` VALUES
(1,'+12015469021'),(1,'+12016750921'),(2,'+12013456571'),(3,'+12014560927'),(4,'+13120675681'),(4,'+13126789
034'),(5,'+13127808945');

CREATE TABLE `orders` (
  `oid` int(11) NOT NULL,
  `otime` datetime DEFAULT NULL,
  `cid` int(11) DEFAULT NULL,
  `sname` varchar(45) DEFAULT NULL,
  `did` int(11) DEFAULT NULL,
  `deliveryfees` int(11) DEFAULT NULL,
  `cardnumber` varchar(45) DEFAULT NULL,
  `orderproductcost` int(11) NOT NULL DEFAULT '0',
  `rtime` datetime DEFAULT NULL,
  `score` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `cid` (`cid`,`cardnumber`),
  KEY `did` (`did`),
  KEY `sname` (`sname`),
  CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`cid`, `cardnumber`) REFERENCES `customer_credit_card`
(`cid`, `cardnumber`),
  CONSTRAINT `orders_ibfk_2` FOREIGN KEY (`did`) REFERENCES `driver` (`did`),
  CONSTRAINT `orders_ibfk_3` FOREIGN KEY (`sname`) REFERENCES `store` (`sname`)
)

INSERT INTO `orders` VALUES (1,'2015-01-01 00:00:00',5,'Tom’s Store',4,60,'1928373645012345',140,'2015-
01-02 00:00:00',3),(2,'2015-01-02 00:00:00',4,'Tom’s Store',4,40,'9273651586253468',75,'2015-01-03
00:00:00',2),(3,'2016-01-02 00:00:00',5,'Tom’s Store',5,20,'1928373645012345',40,'2016-01-03
00:00:00',4),(4,'2016-02-02 13:25:00',2,'Delli\'s Store',2,30,'3456752458941267',251,'2016-02-03
13:25:00',4),(5,'2017-02-02 13:25:00',2,'Delli\'s Store',2,40,'3456752458941267',150,'2017-02-03
13:25:00',5),(6,'2017-02-03 13:25:00',1,'Patel\'s Store',1,20,'4567890123456789',140,'2017-02-04
```

13:25:00',5),(7,'2016-02-04 13:25:00',5,'Tom’s Store',5,25,'1928373645012345',525,'2016-02-05 13:25:00',5),(8,'2016-02-05 13:25:00',5,'Tom’s Store',5,35,'1928373645012345',60,'2016-02-06 13:25:00',4),(9,'2016-02-06 13:25:00',5,'Tom’s Store',4,46,'1928373645012345',260,'2016-02-07 13:25:00',2),(10,'2016-03-06 13:25:00',1,'Patel\\'s Store',1,70,'1234567890123456',70,'2016-03-07 13:25:00',3);

```
CREATE TABLE `orders_details` (
  `oid` int(11) NOT NULL,
  `pid` int(11) NOT NULL,
  `quantity` int(11) DEFAULT NULL,
  `productcost` int(11) DEFAULT NULL,
  `totalproductcost` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`,`pid`),
  KEY `pid` (`pid`),
  CONSTRAINT `orders_details_ibfk_1` FOREIGN KEY (`oid`) REFERENCES `orders` (`oid`),
  CONSTRAINT `orders_details_ibfk_2` FOREIGN KEY (`pid`) REFERENCES `product` (`pid`)
)
```

INSERT INTO `orders_details` VALUES (1,1,1,60,60),(1,2,2,40,80),(2,3,3,25,75),(3,2,1,40,40),(4,1,1,65,65),(4,2,3,42,126),(4,3,3,20,60),(5,1,2,65,130),(5,3,1,20,20),(6,2,4,35,140),(7,1,6,60,360),(7,2,1,40,40),(7,3,5,25,125),(8,1,1,60,60),(9,1,1,60,60),(9,2,5,40,200),(10,2,2,35,70);

Trigger executed:

```
DELIMITER ;;
/*!50003 CREATE*/ /*!50017 DEFINER=`root`@`localhost`*/ /*!50003 trigger product_value before insert on orders_details
for each row
begin
set NEW.productcost = (Select productcost
                       from product_price natural join orders
            where pid = new.pid and oid = new.oid);
set NEW.totalproductcost = NEW.quantity * NEW.productcost;
update orders
set orderproductcost = orderproductcost + NEW.totalproductcost
where oid = new.oid;
END */;;
DELIMITER ;
```

```
CREATE TABLE `product` (
  `pid` int(11) NOT NULL,
  `pname` varchar(45) DEFAULT NULL,
  `pdesc` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`pid`)
)
```

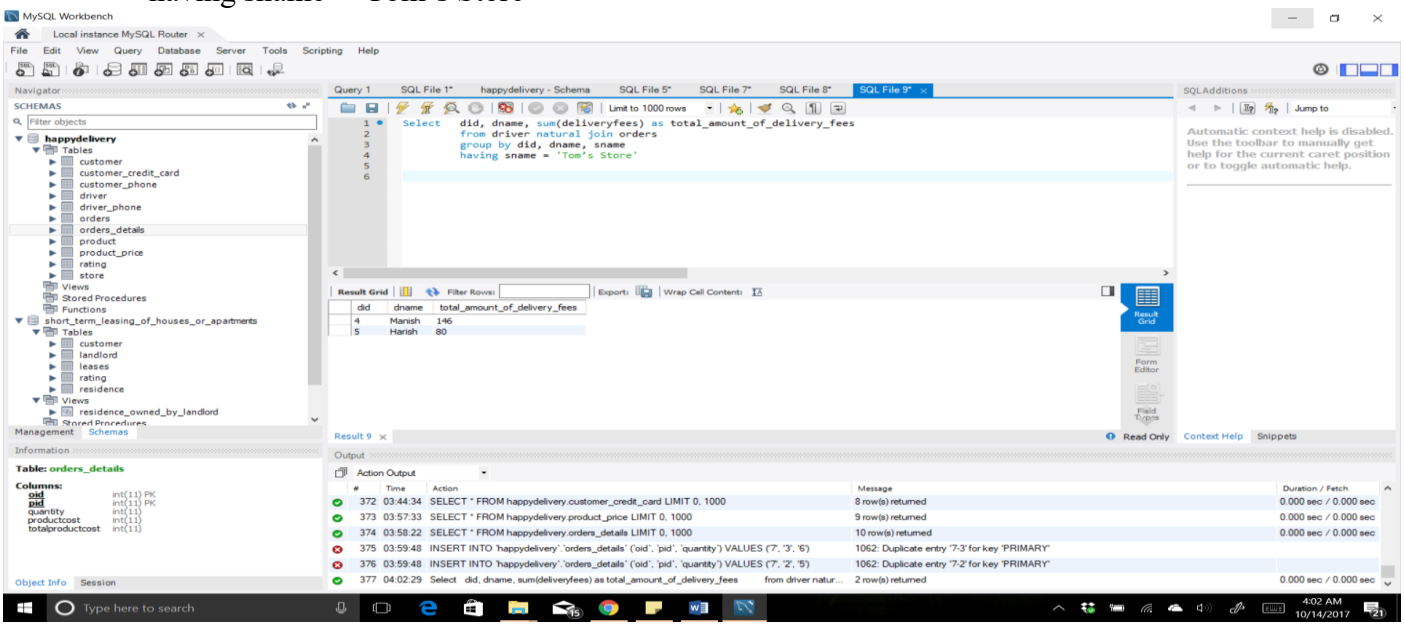
INSERT INTO `product` VALUES (1,'Dining Table','A sylish portable table used for dining '),(2,'Table Fan','A sleek electronic device used for cooling '),(3,'Clothes Organiser','Used to organise and store clothes');

```
CREATE TABLE `product_price` (
  `pid` int(11) NOT NULL,
  `sname` varchar(45) NOT NULL,
  `productcost` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`pid`,`sname`),
  KEY `sname_idx` (`sname`),
```

```
  CONSTRAINT `pid` FOREIGN KEY (`pid`) REFERENCES `product` (`pid`) ON DELETE NO ACTION ON
UPDATE NO ACTION,
  CONSTRAINT `sname` FOREIGN KEY (`sname`) REFERENCES `store` (`sname`) ON DELETE NO ACTION
ON UPDATE NO ACTION
)

INSERT INTO `product_price` VALUES (1,'Delli\'s Store','65'),(1,'Patel\'s Store','55'),(1,'Tom's
Store','60'),(2,'Delli\'s Store','42'),(2,'Patel\'s Store','35'),(2,'Tom's Store','40'),(3,'Delli\'s Store','20'),(3,'Patel\'s
Store','20'),(3,'Tom's Store','25');

CREATE TABLE `store` (
  `sname` varchar(45) NOT NULL,
  `saddr` varchar(45) DEFAULT NULL,
  `scity` varchar(45) DEFAULT NULL,
  `sstate` varchar(45) DEFAULT NULL,
  `s_acc_num` varchar(45) DEFAULT NULL,
  `s_routing_number` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`sname`)
)

INSERT INTO `store` VALUES ('Delli\'s Store','89 Montague St, Brooklyn, NY 11201','Brooklyn','New
York','563864286','264783478'),('Good Luck Store','301 65th St, Brooklyn, NY 11220','Brooklyn','New
York','456789456','374569367'),('Patel\'s Store','630 Caton Ave, Brooklyn, NY 11218','Brooklyn','New
York','457890347','375169457'),('Popeye\'s Store','800 S Wells St, Chicago, IL
60607','Chicago','Illinois','378578934','428036789'),('Tom's Store','6958 N Western Ave, Chicago, IL
60645','Chicago','Illinois','234567894','268469458');
```

(i) For each driver, output her ID, name, and the total amount of delivery fees she has earned from doing deliveries for a convenience store named "Tom's Store".

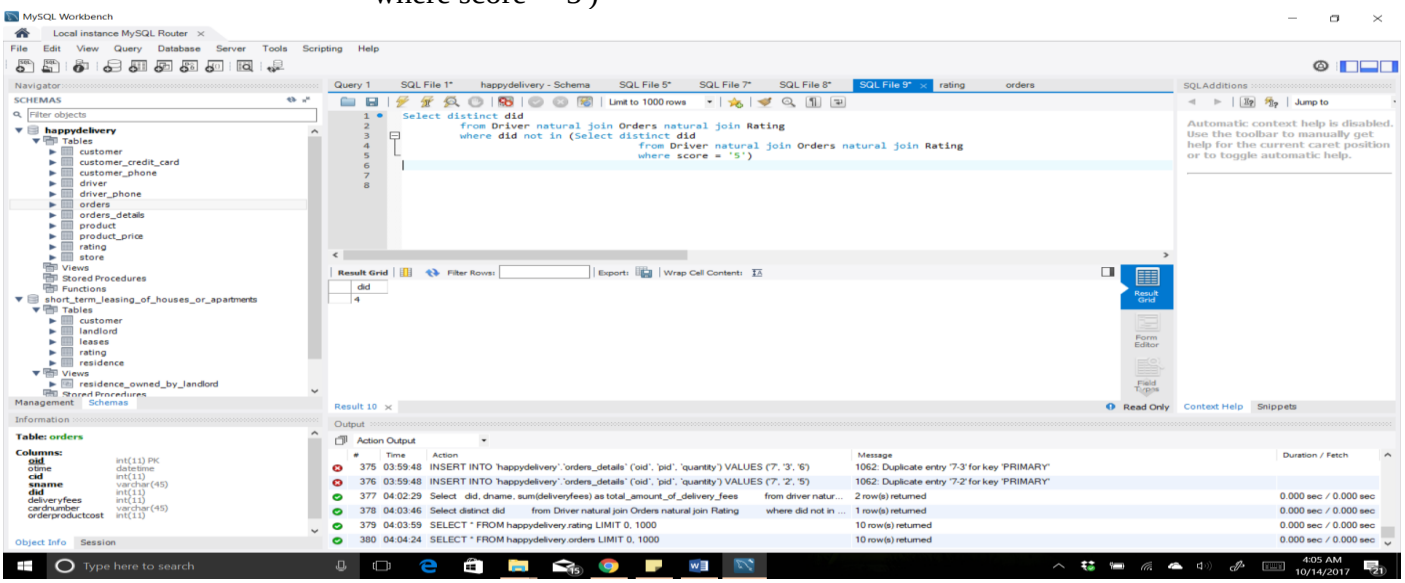
Ans:
```
Select   did, dname, sum(deliveryfees) as total_amount_of_delivery_fees
        from driver natural join orders
        group by did, dname, sname
        having sname = 'Tom's Store'
```



(ii) Output the ID of any driver who has never received a rating of five stars.

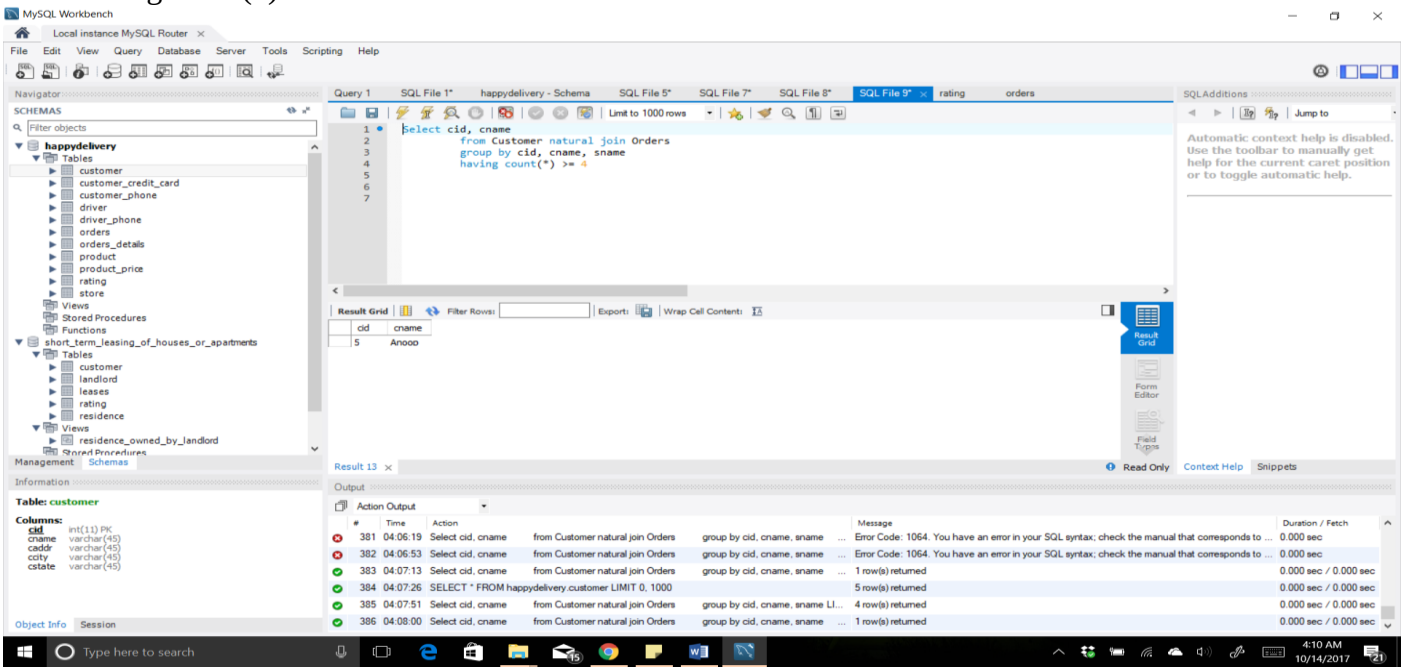
Ans:
```
Select distinct did
        from Driver natural join Orders
        where did not in (Select distinct did
                          from Driver natural join Orders
```

where score = '5')



(iii) Output the name and ID of any customer who has ordered at least four times from the same store.

Ans: Select cid, cname
from Customer natural join Orders
group by cid, cname, sname
having count(*) >= 4



(iv) Output the name and ID of the customer(s) who spent the most money during 2016. Include both the product prices and the delivery fees in your calculation.

Ans: **Implemented in this schema by using triggers and calculating and storing values dynamically in the database system.**

Query for creating the trigger:

```
DELIMITER //
create trigger product_value before insert on orders_details
for each row
begin
set NEW.productcost = (Select productcost
                       from product_price natural join orders
```

```
        where pid = new.pid and oid = new.oid);
set NEW.totalproductcost = NEW.quantity * NEW.productcost;
update orders
set orderproductcost = orderproductcost + NEW.totalproductcost
where oid = new.oid;
END//
DELIMITER;
```

Explanation of the trigger:

The trigger product_value basically calculates values for productcost, totalproductcost, orderproductcost.

productcost: this attribute fetches value from the productcost attribute of the product_price table which is the price of the individual product.

The reason for using two separate tables to store the same value is that incase the price of the product changes in future due to some discount or inflation it will only be changed in the productcost attribute of the product_price table and not the orders_details table. This will maintain the integrity of data of the orders done in past.

totalproductcost: this attribute stores the total cost of a single product i.e the cost of the individual product multiplied by the quantity of that product that is to be purchased.

orderproductcost: this attribute stores the total product cost of the order i.e the sum of the totalproductcost of all the products being purchased in a single order.

Note: orderproductcost attribute is set to default value 0 or should not be null because if it is null the below statement in the trigger will just set null values.

```
set orderproductcost = orderproductcost + NEW.totalproductcost
```

Query for solving the problem:

```
Select cid,cname
from Customer natural join Orders
where year(otime) = '2016'
group by cid,cname
having sum(deliveryfees + orderproductcost) = (Select max(c)
                                               from (Select sum(deliveryfees + orderproductcost) as c
                                                     from Customer natural join Orders
                                                     where year(otime) = '2016'
                                                     group by cid,cname) as total_money_spent)
```