

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

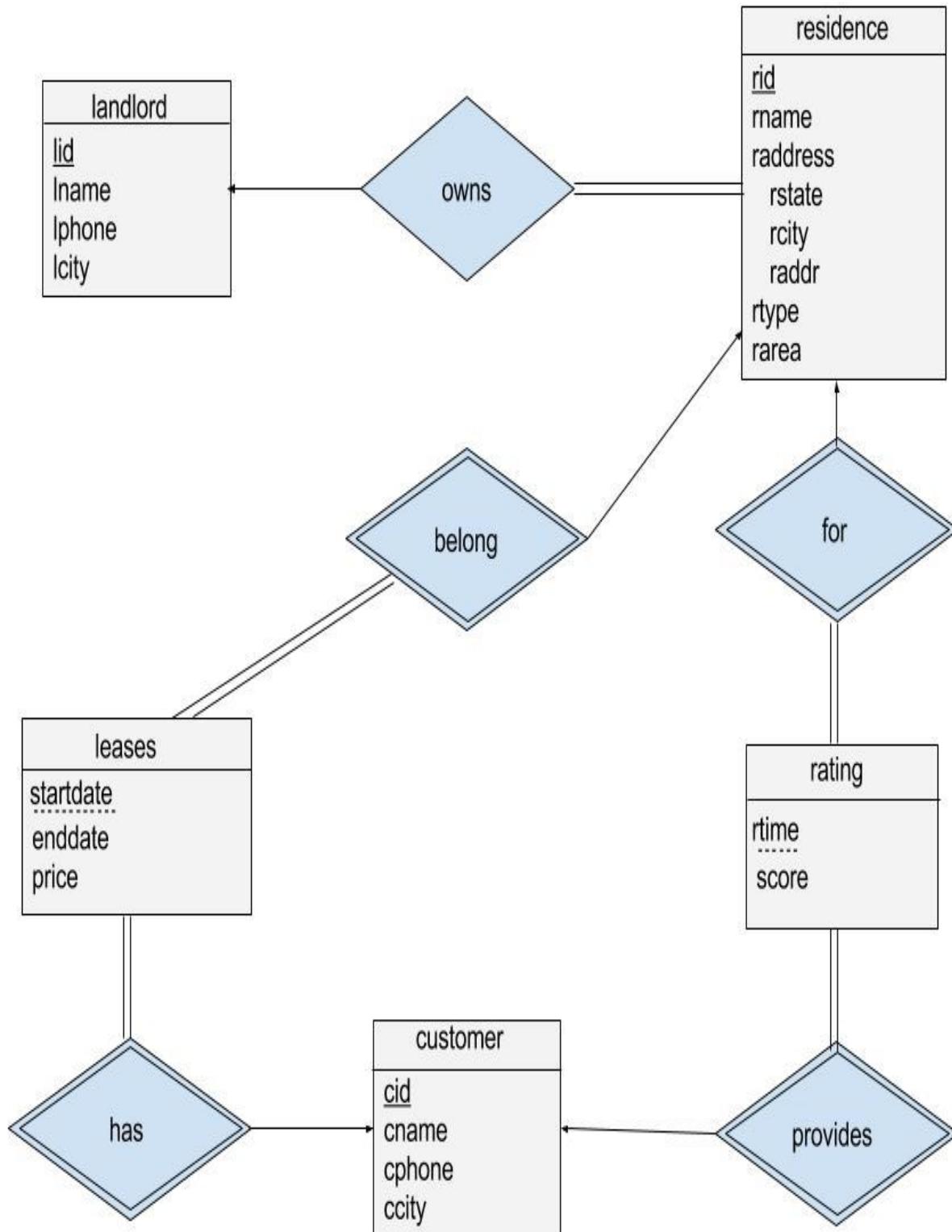
Leases.rid references Residence.rid

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:

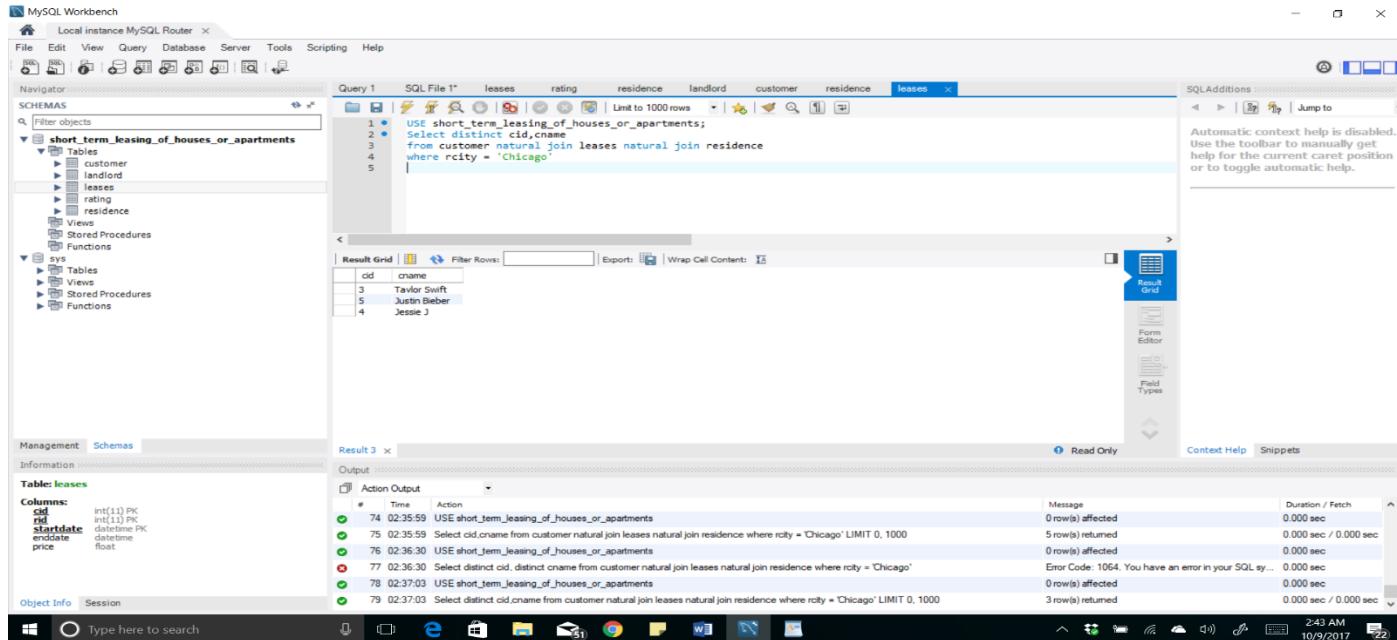
(i) List the cids and cnames of all customers who have rented residences in Chicago.

Ans:

```
USE short_term_leasing_of_houses_or_apartments;
```

```
Select distinct cid, cname
```

```
from customer natural join leases natural join residence  
where rcity = 'Chicago'
```



```
1 USE short_term_leasing_of_houses_or_apartments;
2 Select distinct cid,cname
3 from customer natural join leases natural join residence
4 where rcity = 'Chicago'
5
```

cid	cname
3	Taylor Swift
3	Justin Bieber
4	Jessie J

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

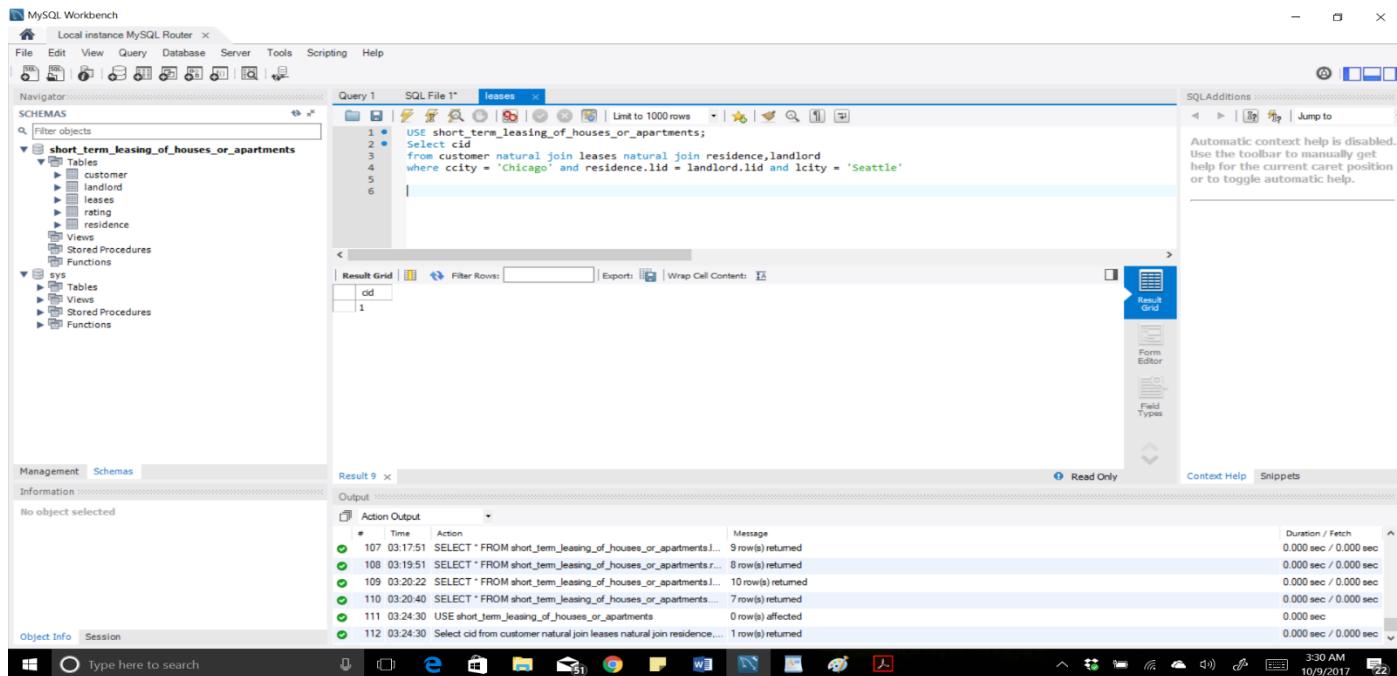
Ans:

```
USE short_term_leasing_of_houses_or_apartments;
```

```
Select cid
```

```
from customer natural join leases natural join residence,landlord
```

```
where ccity = 'Chicago' and residence.lid = landlord.lid and lcity = 'Seattle'
```



```
1 USE short_term_leasing_of_houses_or_apartments;
2 Select cid
3 from customer natural join leases natural join residence,landlord
4 where ccity = 'Chicago' and residence.lid = landlord.lid and lcity = 'Seattle'
5
6
```

cid
1

(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')

```

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')

```

cid	cname
5	Justin Bieber

Action Output

#	Time	Action	Message	Duration / Fetch
111	03:24:30	USE short_term_leasing_of_houses_or_apartments	0 row(s) affected	0.000 sec
112	03:24:30	Select cid from customer natural join leases natural join residence...	1 row(s) returned	0.000 sec / 0.000 sec
113	03:45:14	USE short_term_leasing_of_houses_or_apartments	0 row(s) affected	0.000 sec
114	03:45:14	Select cid, cname from customer natural join leases natural join re...	3 row(s) returned	0.000 sec / 0.000 sec
115	03:45:34	USE short_term_leasing_of_houses_or_apartments	0 row(s) affected	0.000 sec
116	03:45:34	Select distinct cid, cname from customer natural join leases natur...	1 row(s) returned	0.000 sec / 0.000 sec

(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)

```

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)

```

rtype	count(distinct lid)
2BR-2BA	3

Action Output

#	Time	Action	Message	Duration / Fetch
23	13:18:38	Select rtype from residence natural join landlord group by rtype having count(distinct lid) = (Select max(C)...	Error Code: 1248. Every derived table must have its own alias	0.000 sec
24	13:22:33	USE short_term_leasing_of_houses_or_apartments	0 row(s) affected	0.000 sec
25	13:22:33	Select rtype from residence natural join landlord group by rtype having count(distinct lid) = (Select max(C)...	1 row(s) returned	0.000 sec / 0.000 sec
26	13:30:19	Select rtype from residence natural join landlord group by rtype LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
27	13:30:54	USE short_term_leasing_of_houses_or_apartments	0 row(s) affected	0.000 sec
28	13:30:54	Select type, count(distinct lid) from residence natural join landlord group by rtype having count(d...	1 row(s) returned	0.000 sec / 0.000 sec

- (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)

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
  Filter objects
  short_term_leasing_of_houses_or_apartments
    Tables
      customer
      landlord
      leases
      rating
      residence
      Views
      Stored Procedures
      Functions
      sys
Management Schemas
Information
Table: leases
Columns:
  id int(11) PK
  price float
  startdate date
  enddate date
  price float
Object Info Session
Result 16 x
Action Output
Time Action Message Duration / Fetch
39 14:37:33 SELECT * FROM short_term_leasing_of_houses_or_apartments LIMIT 0, 1000 10 rows(s) returned 0.000 sec / 0.000 sec
40 14:43:16 USE short_term_leasing_of_houses_or_apartments 0 rows(s) affected 0.000 sec
41 14:43:16 Select cid,cname from customer natural join leases where year(startdate) ='2016' LIMIT 0, 1000 10 rows(s) returned 0.000 sec / 0.000 sec
42 14:45:45 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 1 row(s) returned 0.000 sec / 0.000 sec
43 14:46:54 USE short_term_leasing_of_houses_or_apartments 0 rows(s) affected 0.000 sec
44 14:46:54 Select cid,cname from customer natural join leases where year(startdate) ='2016' group by cid.c... 1 row(s) returned 0.000 sec / 0.000 sec

```

- (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)

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
  Filter objects
  short_term_leasing_of_houses_or_apartments
    Tables
      customer
      landlord
      leases
      rating
      residence
      Views
      Stored Procedures
      Functions
      sys
Management Schemas
Information
Table: leases
Columns:
  id int(11) PK
  price float
  startdate date
  enddate date
  price float
Object Info Session
Result 27 x
Action Output
Time Action Message Duration / Fetch
62 17:52:47 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 7 rows(s) returned 0.000 sec / 0.000 sec
63 17:56:26 Select total_money_earned.lcity,max(c) from (Select lid,lcity,sum(price) as c ... 4 rows(s) returned 0.000 sec / 0.000 sec
64 17:56:35 USE short_term_leasing_of_houses_or_apartments 0 rows(s) affected 0.000 sec
65 17:56:35 Select lid,lcity,sum(price) from landlord natural join residence natural join leases group by lid,lcity ... 5 rows(s) returned 0.000 sec / 0.000 sec
66 17:57:18 USE short_term_leasing_of_houses_or_apartments 0 rows(s) affected 0.000 sec
67 17:57:18 Select lid,lcity from landlord natural join residence natural join leases group by lid,lcity having (lid,lcity) ... 5 rows(s) returned 0.000 sec / 0.000 sec

```

(vii) For each month in 2016, output the city 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(runtime) as Month_in_2016, rcity
```

```
from residence natural join rating
```

```
where year(runtime) = '2016'
```

```
group by rcity, month(runtime)
```

```
having (Month_in_2016, avg(score)) in (Select b,max(c)
```

```
from (Select month(runtime) as b,rcity,avg(score) as c
```

```
from residence natural join rating
```

```
where year(runtime) = '2016'
```

```
group by rcity, month(runtime)) as avg_rating
```

```
group by b)
```

```
order by Month_in_2016
```

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

- Schemas:** The current schema is `short_term_leasing_of_houses_or_apartments`.
- Table:** The table `rating` is selected, showing columns: `cid`, `ccid`, `ctime`, `rating`, and `score`.
- Query Editor:** The SQL query is displayed, which retrieves the city with the best average rating for each month in 2016, based on ratings from that month.
- Result Grid:** The result shows four rows of data:

Month_in_2016	rcity
1	Baltimore
2	Chicago
3	Brooklyn
4	Seattle
- Action Output:** Shows the execution log with 11 entries, all indicating 0 rows affected.

(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
```

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

- Schemas:** The current schema is `short_term_leasing_of_houses_or_apartments`.
- Table:** The table `customer` is selected, showing columns: `cid`, `cphone`, `ccity`, and `cname`.
- Query Editor:** The SQL query is displayed, which retrieves the average duration of leases for customers in San Diego starting in 2016.
- Result Grid:** The result shows four rows of data:

cid	cname	Average_duration_of_leases
1	Ariana Grande	3.0000
2	Taylor Swift	2.0000
3	Jessie J	2.5000
- Action Output:** Shows the execution log with 13 entries, all indicating 0 rows affected.

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

Ans:

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

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

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

(iv) \mathbf{P} number_of_distinct_landlords (rtype, c) (rtype \mathbf{G} count(distinct lid) (residence \bowtie landlord))

$\mathbf{P}_{R1(b)} (\mathbf{G} \max(c) (\text{number_of_distinct_landlords}))$

$\mathbf{P}_{R2(rtype, b)} (\text{rtype} \mathbf{G} \text{count}(\text{distinct lid}) (\text{residence} \bowtie \text{landlord}))$

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

(v) \mathbf{P} total_money_for_rentals (cid, cname, c) (cid, cname \mathbf{G} sum(price) ($\sigma_{\text{year(startdate)} = '2016'}$ (customer \bowtie leases)))

$\mathbf{P}_{R1(b)} (\mathbf{G} \max(c) (\text{total_money_for_rentals}))$

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

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

(vi) \mathbf{P} total_money_earned (lid, lcity, c) (lid, lcity \mathbf{G} sum(price) (landlord \bowtie residence \bowtie leases))

$\mathbf{P}_{T1(a,b)} (lcity \mathbf{G} \max(c) (\text{total_money_earned}))$

$\mathbf{P}_{T2(lid, lcity, c)} (lid, lcity \mathbf{G} \text{sum(price)} (\text{landlord} \bowtie \text{residence} \bowtie \text{leases}))$

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

(vii) \mathbf{P} avg_rating (rcity, b, c) (rcity, month(runtime) \mathbf{G} avg(score) ($\sigma_{\text{year(runtime)} = '2016'}$ (residence \bowtie rating)))

$\mathbf{P}_{R1(b,d)} (b \mathbf{G} \max(c)(\text{avg_rating}))$

$\mathbf{P}_{R2(rcity, Month_in_2016, d)} (rcity, month(runtime) \mathbf{G} \text{avg(score)} (\sigma_{\text{year(runtime)} = '2016'} (\text{residence} \bowtie \text{rating})))$

$\pi_{rcity, Month_in_2016} (\mathbf{R1} \bowtie \mathbf{R2})$

(viii)

$\text{P}_{\text{duration_of_lease}}(\text{ccity}, \text{cname}, \text{cid}, \text{Average_duration_of_lease})$ (ccity, cname, cid) G_{avg}

(datediff(enddate, startdate)) ($\sigma_{\text{ccity} = \text{'San Diego'} \wedge \text{year(startdate)} = \text{'2016'}}$ (leases natural join customer)))

$\pi_{\text{cid}, \text{cname}, \text{Average_duration_of_lease}}(\text{duration_of_lease})$

(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')

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

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

```
1 USE short_term_leasing_of_houses_or_apartments;
2 update residence
3 SET lid = (Select lid
4     from landlord
5     where lname = 'Amy Daniels')
6 WHERE rid IN (Select rid
7     from ((Select * from residence) AS R) natural join landlord
8     where lname = 'Bob Daniels')
9
10 SELECT * FROM residence
11
12
13
14
```
- Result Grid:** Displays the updated data for the 'residence' table. The columns are: rid, rname, rstate, roty, raddr, rtype, rarea, lid. The data includes rows for various houses like Little Cabin, Clean Cabin, etc., with their new landlord set to 'Amy Daniels'.
- Action Output:** Shows the log of actions taken by the query:

#	Time	Action	Message	Duration / Fetch
39	12:37:58	USE short_term_leasing_of_houses_or_apartments	0 row(s) affected	0.000 sec
40	12:37:58	DELETE FROM rating WHERE score = 1 AND rid IN (SELECT rid FROM ((SELECT * FROM residence) AS R) NATURAL JOIN landlord WHERE lname = 'Bob Daniels')	2 row(s) affected	0.016 sec
41	12:39:09	SELECT * FROM rating LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
42	12:43:08	SELECT * FROM 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')	8 row(s) returned	0.000 sec / 0.000 sec
43	12:43:30	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')	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
44	12:43:45	SELECT * FROM residence LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec

(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 ()))

```

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()))
Select * from customer

```

id	cname	cphone	city
1	Bruno Mars	9293334444	Chicago
2	Adele	9293456789	San Diego
3	Taylor Swift	9293456789	San Diego
4	Jessica Biel	9291234567	San Diego
5	Justin Bieber	9291234568	Seattle

Management Schemas

Schema: short_term_leasing_of_houses_or_apartments

Action Output

- # Time Action Message Duration / Fetch
 - 62 14:06:44 USE short_term_leasing_of_houses_or_apartments 0 row(s) affected 0.000 sec
 - 63 14:06:44 delete customer1 where cid not in (Select cid from customer1 natural join leases) ... Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to ... 0.000 sec
 - 64 14:07:31 drop table customer1 0 row(s) affected 0.015 sec
 - 65 14:14:30 USE short_term_leasing_of_houses_or_apartments 0 row(s) affected 0.000 sec
 - 66 14:14:30 delete from customer where cid not in (Select * from customer) as c natural join... 2 row(s) affected 0.000 sec
 - 67 14:14:53 Select * from customer LIMIT 0, 1000 5 row(s) returned 0.000 sec / 0.000 sec

(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)

```

```

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)
Select * from rating

```

id	rid	rtime	score
1	207	2016-04-25 12:30:00	2
2	202	2016-01-20 11:30:00	4
3	202	2016-02-20 16:30:00	2
3	203	2016-01-22 12:30:00	4
4	203	2016-03-21 11:30:00	3
4	204	2016-01-21 13:30:00	5
5	205	2016-01-23 14:30:00	2
5	206	2016-02-20 15:30:00	3

Management Schemas

Table: rating

Columns:

- cid int(11) PK
- rid int(11) PK
- rtime datetime PK
- score float

Action Output

- # Time Action Message Duration / Fetch
 - 36 12:31:47 Select rid from (Select * from rating) as R group by rid ... 4 row(s) returned 0.000 sec / 0.000 sec
 - 37 12:32:26 Select rid.avg(score) from (Select * from rating) as R group ... 4 row(s) returned 0.000 sec / 0.000 sec
 - 38 12:33:14 SELECT * FROM short_term_leasing_of_houses_or_apartments.rating LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
 - 39 12:37:58 USE short_term_leasing_of_houses_or_apartments 0 row(s) affected 0.000 sec
 - 40 12:37:58 delete from rating where score = 1 and rid in (Select rid from (Select * from rating) as R group by rid having avg(score)< 3) 2 row(s) affected 0.016 sec
 - 41 12:39:09 Select * from rating LIMIT 0, 1000 8 row(s) returned 0.000 sec / 0.000 sec

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
```

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the schema 'short_term_leasing_of_houses_or_apartments', there is a 'Views' folder containing a single view named 'residence_owned_by_landlord'. The main Query Editor window displays the SQL code for creating this view:

```
1 create view residence_owned_by_landlord as
2 select lid, lname, lcity, rid, rname, rcity
3   from residence natural join landlord
4
5 Select * from residence_owned_by_landlord
```

Below the code, the Result Grid shows the data returned by the query:

lid	lname	lcity	rid	rname	rcity
101	Bob Daniels	Brooklyn	201	Little Cabin	Brooklyn
102	Trumo	Brooklyn	202	Clean Cabin	Brooklyn
103	Trump	Chicago	203	Cheer House	Chicago
104	Lincoln	Chicago	204	Nice House	Chicago
105	Franklin	Chicago	205	Black House	Chicago
106	Abe	San Diego	206	White House	Chicago
107	Clinton	Seattle	207	Blue House	Seattle
105	Franklin	Chicago	208	Pink House	Seattle

The Output pane at the bottom shows the execution history of the query:

Action	Time	Action	Message	Duration / Fetch
12	15:34:54	SELECT * FROM short_term_leasing_of_houses_or_apartments.rating LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
13	15:34:58	SELECT * FROM short_term_leasing_of_houses_or_apartments.residence LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
14	15:40:15	SELECT * FROM short_term_leasing_of_houses_or_apartments.residence LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
15	15:40:39	SELECT * FROM short_term_leasing_of_houses_or_apartments.landlord LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
16	15:41:07	create view residence_owned_by_landlord as Select lid,lname,lcity,rid,rname,rcity from residence natural join landlord	0 rows(a) affected	0.015 sec
17	15:42:17	Select * from residence_owned_by_landlord LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec

(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)
```

```
from (Select lcity, count(rid) as c
```

```
from residence_owned_by_landlord
```

```
group by lcity, lname) as count_of_residence
```

```
group by lcity)
```

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the schema 'short_term_leasing_of_houses_or_apartments', there is a 'Views' folder containing a view named 'residence_owned_by_landlord'. The main Query Editor window displays the SQL code for creating this view:

```
1 Select lcity, lname
2   from residence_owned_by_landlord
3   group by lcity, lname
4   having (lcity, count(rid)) in (Select lcity, max(c)
5       from (Select lcity, count(rid) as c
6           from residence_owned_by_landlord
7           group by lcity, lname) as count_of_residence
8       group by lcity)
```

The Output pane at the bottom shows the execution history of the query, starting with an error message:

Action	Time	Action	Message	Duration / Fetch
43	18:02:41	Select lcity, count(rid) from residence_owned_by_landlord group by lcity, name having (lcity, count(rid)) in (Select lcity, max(c) from (Select lcity, count(rid) as c from residence_owned_by_landlord group by lcity, name) as count_of_residence group by lcity)	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to ...	0.000 sec / 0.000 sec
44	18:03:26	Select lcity, count(rid) as c	4 row(s) returned	0.000 sec / 0.000 sec
45	18:04:35	Select lcity, count(c) from (Select lcity, count(rid) as c from residence_owned_by_landlord group by lcity, name) as count_of_residence group by lcity	4 row(s) returned	0.000 sec / 0.000 sec
46	18:04:53	Select lcity, count(rid) from residence_owned_by_landlord group by lcity, name having lcity, count(rid) in (Select lcity, max(c) from (Select lcity, count(rid) as c from residence_owned_by_landlord group by lcity, name) as count_of_residence group by lcity)	7 row(s) returned	0.000 sec / 0.000 sec
47	18:05:31	Select lcity, name from residence_owned_by_landlord group by lcity, name having lcity, count(rid) in (Select lcity, max(c) from (Select lcity, count(rid) as c from residence_owned_by_landlord group by lcity, name) as count_of_residence group by lcity)	5 row(s) returned	0.000 sec / 0.000 sec
48	18:05:51	Select lcity, name from residence_owned_by_landlord group by lcity, name having lcity, count(rid) in (Select lcity, max(c) from (Select lcity, count(rid) as c from residence_owned_by_landlord group by lcity, name) as count_of_residence group by lcity)	5 row(s) returned	0.000 sec / 0.000 sec

(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'
```

The screenshot shows the MySQL Workbench interface. In the top-left pane, the 'Navigator' shows the schema 'short_term_leasing_of_houses_or_apartments' with tables like customer, landlord, leases, rating, and residence. In the top-right pane, 'Query 1' contains the SQL code for updating the residence name. Below it, the 'Result Grid' displays the updated data. The bottom pane shows the 'Output' window with the log of executed statements and their results.

id	Iname	icity	rid	rname	rcity
101	Bob Daniels	Brooklyn	201	Bio Cabin	Brooklyn
102	Trump	Brooklyn	202	Clean Cabin	Brooklyn
103	Bush	Chicago	203	Cute House	Chicago
104	Lincoln	Chicago	204	Dark Cabin	Chicago
105	Franklin	Chicago	205	Black House	Chicago
106	Aha	San Diego	206	White House	Chicago
107	Clinton	Seattle	207	Blue House	Seattle
105	Franklin	Chicago	208	Pink House	Seattle

Action Output

#	Time	Action	Message	Duration / Fetch
49	18:10:01	Select * from residence_owned_by_landlord LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
50	18:12:49	update residence_owned_by_landlord set rname = 'Big Cabin' where rname = 'Little Cabin'	Error Code: 1175. You are using safe update mode and you tried to update a table without a W...	0.000 sec
51	18:14:18	Select * from residence_owned_by_landlord LIMIT 0, 1000	8 row(s) returned	0.016 sec / 0.000 sec
52	18:14:28	update residence_owned_by_landlord set rname = 'Big Cabin' where rname = 'Little Cabin'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
53	18:14:37	Select * from residence_owned_by_landlord LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
54	18:15:13	Select * from residence_owned_by_landlord LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec

(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:

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (happypack), Tables (customer, customer_credit_card, customer_phone, driver, driver_phone, orders, orders_details, product, product_price, store), Views, Stored Procedures, Functions.
- Query Editor:** SQL File 1*, SQL File 5*, SQL File 7*, residence, rating, leases, SQL File 10*. The current tab is SQL File 10*. The code is:

```
1 INSERT INTO RATING VALUE ('1','201','2016-04-26 12:30:00','3')
2
3 delete from rating
4 where cid = '1'
5
6 update rating
7 set score = '5'
8 where cid = '1'
```
- Output Window:** Shows the execution log with rows 29 through 34. Rows 29, 30, 31, and 32 are successful. Row 32 fails with Error Code: 1442. Rows 33 and 34 also fail with Error Code: 1442.

#	Time	Action	Message	Duration / Fetch
29	04:53:44	create trigger check_average2 after update on rating for each row BEGIN delete from rating	... 0 row(s) affected	0.000 sec
30	04:56:50	SELECT * FROM short_term_leasing_of_houses_or_apartments.rating LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
31	04:58:34	SELECT * FROM short_term_leasing_of_houses_or_apartments.leases LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
32	05:00:17	INSERT INTO RATING VALUE ('1','201','2016-04-26 12:30:00','3')	Error Code: 1442. Can't update table 'rating' in stored function/trigger because it is already used...	0.016 sec
33	05:04:45	delete from rating where cid = '1'	Error Code: 1442. Can't update table 'rating' in stored function/trigger because it is already used...	0.015 sec
34	05:05:54	update rating set score = '5' where cid = '1'	Error Code: 1442. Can't update table 'rating' in stored function/trigger because it is already used...	0.016 sec
- System Bar:** Type here to search, Taskbar icons, Date/Time (5:16 AM, 10/15/2017).

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.

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

- Schemas:** The schema `short_term_leasing_of_houses_or_apartments` is selected, showing tables: `customer`, `landlord`, `leases`, `rating`, and `residence`.
- Tables:** The `rating` table is selected in the Query Editor.
- Query Editor:** A query `1 • Select * from rating` is running, showing the following data:| cid | rid | rtime | score |
| --- | --- | --- | --- |
| 1 | 201 | 2017-10-04 12:30:00 | 1 |
| 1 | 201 | 2017-10-05 12:30:00 | 1 |
| 1 | 201 | 2017-10-06 12:30:00 | 1 |
| 1 | 201 | 2017-10-07 12:30:00 | 1 |
| 1 | 201 | 2017-10-14 10:30:00 | 1 |
| 1 | 207 | 2016-04-25 12:30:00 | 2 |
| 2 | 202 | 2016-01-20 11:30:00 | 4 |
| 3 | 202 | 2016-02-20 16:30:00 | 2 |
| 3 | 203 | 2016-01-22 12:30:00 | 4 |
| 4 | 201 | 2016-03-21 11:30:00 | 3 |
| 4 | 204 | 2016-01-21 13:30:00 | 5 |
- Output:** The Action History shows the following operations:| # | Time | Action | Message | Duration / Fetch |
| --- | --- | --- | --- | --- |
| 4 | 13:40:06 | INSERT INTO RATING VALUE ('1','201','2017-10-4 12:30:00','1') | 1 row(s) affected | 0.000 sec |
| 5 | 13:41:15 | INSERT INTO RATING VALUE ('1','201','2017-10-5 12:30:00','1') | 1 row(s) affected | 0.000 sec |
| 6 | 13:41:31 | INSERT INTO RATING VALUE ('1','201','2017-10-6 12:30:00','1') | 1 row(s) affected | 0.000 sec |
| 7 | 13:41:49 | INSERT INTO RATING VALUE ('1','201','2017-10-7 12:30:00','1') | 1 row(s) affected | 0.000 sec |
| 8 | 13:42:32 | SELECT * FROM short_term_leasing_of_houses_or_apartments.rating LIMIT 0, 1000 | 14 row(s) returned | 0.000 sec / 0.000 sec |
| 9 | 13:43:07 | Select * from rating LIMIT 0, 1000 | 14 row(s) returned | 0.000 sec / 0.000 sec |

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.

The screenshot shows the MySQL Workbench interface. In the top-left, the Navigator pane displays the database schema with the 'rating' table selected. The top-right shows the SQL pane with a failed INSERT INTO RATING query. The bottom-right shows the Action Output pane listing the history of operations, including the successful insertion of five rows and the failure of the sixth row due to a trigger.

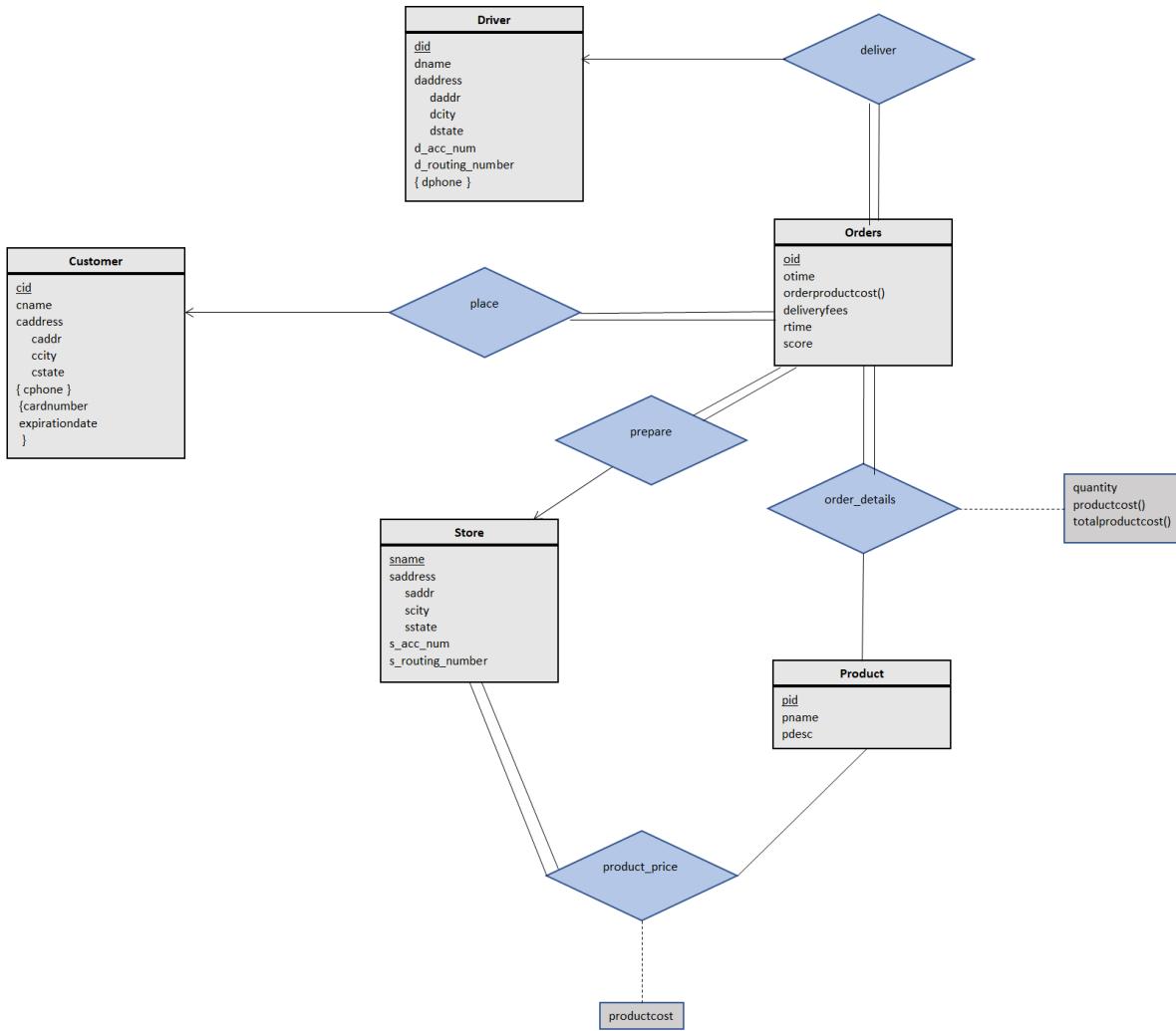
#	Time	Action	Message	Duration / Fetch
5	13:41:15	INSERT INTO RATING VALUE ('1','201','2017-10-5 12:30:00','1')	1 row(s) affected	0.000 sec
6	13:41:31	INSERT INTO RATING VALUE ('1','201','2017-10-6 12:30:00','1')	1 row(s) affected	0.000 sec
7	13:41:49	INSERT INTO RATING VALUE ('1','201','2017-10-7 12:30:00','1')	1 row(s) affected	0.000 sec
8	13:42:32	SELECT * FROM short_term_leasing_of_houses_or_apartments.rating LIMIT 0, 1000	14 row(s) returned	0.000 sec / 0.000 sec
9	13:43:07	Select * from rating LIMIT 0, 1000	14 row(s) returned	0.000 sec / 0.000 sec
10	13:44:01	INSERT INTO RATING VALUE ('1','201','2017-10-8 12:30:00','1')	Error Code: 1644. Error : The customer has already given five 1 star ratings in the past 2 weeks....	0.000 sec

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 delivery 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**, ottime, 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,'4678469256803  
452','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,'+13124560  
791'),(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'

did	dname	total_amount_of_delivery_fees
4	Marish	146
5	Harish	80

- (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')

```

1. Select distinct did
   from Driver natural join Orders natural join Rating
   where did not in (Select distinct did
                      from Driver natural join Orders natural join Rating
                      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

```

1. 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)

```

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas, Tables, Views, Functions, Stored Procedures, Triggers.
- Query Editor:** Shows the complex SQL query for finding customers who spent the most in 2016.
- Result Grid:** Displays the result of the query, showing one row for customer 'Anoop'.
- SQL Additions:** A panel on the right with help and context information.
- Log:** Shows the execution log with various events and their times.
- Object Info:** Shows the structure of the 'customer' table.
- Session:** Shows the current session details.

