ASSIGNMENT 2

Problem 1:

a) Suppose the Trips table has only cusid, carid as primary key. Would that be a good idea?How about using cusid, did, getontime as primary key?

Ans: If Trips would have onlt cusid , carid as primary key then we would have loss important information and it would be hard to uniquely identify each trip such as which driver information of particular trip. Although we have relationship between car and driver , many drivers can own one car thus giving only carid it is hard to tell which driver was driving for particular trip. Also it is possible that same customer has booked same car more than once thus in such scenario our primary key constraint does not work. In this scenario getontime or ordertime is necessary to uniquely identify information of one trip. Thus only cusid, carid as primary key is not a good idea.

If we use cusid,did,getontime as primary key then information about which car was used for particular trip will be lost. We have relationship carownership where we can get details of cars but it is possible that one driver can be owner of more than one car. Thus it is hard to tell during particular trip driver used which car owned by him. Thus we need to add carid in primary key.

b) Identify suitable foreign-key relationships for the above tables.

Ans: Table Customer has primary key as cusid and there is no foreign key available.
Table Driver has primary key as did and there is no foreign key available.
Table car has primary key as carid and there is no foreign key available.
Table CarOwnerShip has primary key as did and carid where did is foreign key from Driver table and foreign key carid is from Car table.
Table Trips has primary key as cusid,carid,did,getontime where did and cusid are foreign keys from CarOwnerShip table and foreign key cusid is from Customer table.

c) 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.

mysql> create table customer
(cusid int not null, cusname varchar(45) null, cusphone varchar(45) null, cuscity varchar(45) null, primary key(cusid));
Query OK, 0 rows affected (0.09 sec)

mysql> select * from customer;

```
+-------+--------------+------------+----------+
| cusid | cusname      | cusphone   | cuscity  |
+-------+--------------+------------+----------+
|     1 | Fred         | 9293334444 | Brooklyn |
|     2 | Nick         | 9293435456 | Queens   |
|     3 | Taylor Swift | 9293456789 | New York |
|     4 | Jesse        | 9291234567 | Brooklyn |
+-------+--------------+------------+----------+
```

4 rows in set (0.00 sec)

```
mysql> create table driver
(did int not null, dname varchar(45) null, dphone varchar(45) null, dcity varchar(45) null, primary
key(did));
Query OK, 0 rows affected (0.01 sec)
```

mysql> select * from driver;

```
+-----+-------+------------+----------+
| did | dname | dphone     | dcity    |
+-----+-------+------------+----------+
| 101 | Jay   | 9299878765 | Brooklyn |
| 102 | Tom   | 9293487876 | Queens   |
| 103 | Gray  | 9293230987 | Brooklyn |
| 104 | Mary  | 9294569876 | New York |
| 105 | Frank | 9293236745 | New York |
+-----+-------+------------+----------+
5 rows in set (0.00 sec)
```

```
mysql> create table car
(carid int not null, carbrand varchar(45) null, carsize varchar(45) null, primary key(carid));
Query OK, 0 rows affected (0.01 sec)
```

mysql> select * from car;

```
+-------+----------+---------+
| carid | carbrand | carsize |
+-------+----------+---------+
|   201 | Toyota   | small   |
|   202 | Honda    | medium  |
|   203 | Benz     | medium  |
|   204 | Toyota   | Large   |
|   205 | BMW      | medium  |
|   206 | Ford     | small   |
+-------+----------+---------+
6 rows in set (0.00 sec)
```

```
mysql> create table carownership
(did int not null, carid int not null, primary key(did, carid), foreign key(did) references driver(did), foreign
key (carid) references car(carid));
Query OK, 0 rows affected (0.03 sec)
```

mysql> select * from carownership

```
    -> ;
+-----+-------+
| did | carid |
+-----+-------+
| 101 |   201 |
| 102 |   201 |
| 101 |   202 |
| 102 |   203 |
| 103 |   204 |
| 104 |   205 |
| 104 |   206 |
```

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

mysql> create table trips
(cusid int not null, carid int not null, did int not null, getontime datetime not null,getofftime datetime null,
price float null, distance float null, primary key(cusid, carid, did, getontime), foreign key (cusid)
references customer(cusid), foreign key (carid) references car(carid), foreign key (did) references
driver(did));
Query OK, 0 rows affected (0.02 sec)
```

mysql> select * from trips;

```
+-------+-------+-----+---------------------+---------------------+-------+----------+
| cusid | carid | did | getontime           | getofftime          | price | distance |
+-------+-------+-----+---------------------+---------------------+-------+----------+
|     1 |   201 | 101 | 2017-01-01 10:30:00 | 2017-01-01 11:20:00 |    50 |        8 |
|     1 |   203 | 102 | 2017-01-15 15:25:00 | 2017-01-15 15:45:00 |    30 |        4 |
|     2 |   204 | 103 | 2017-02-15 16:00:00 | 2017-02-15 16:35:00 |    45 |        6 |
|     3 |   202 | 101 | 2017-02-16 14:05:00 | 2017-02-16 14:15:00 |    10 |        2 |
|     3 |   204 | 103 | 2017-02-02 17:10:00 | 2017-02-02 17:20:00 |    15 |      3.5 |
|     4 |   205 | 104 | 2017-02-04 19:00:00 | 2017-02-04 19:18:00 |    28 |      3.5 |
|     4 |   206 | 104 | 2017-01-20 16:45:00 | 2017-01-20 17:10:00 |    33 |        5 |
+-------+-------+-----+---------------------+---------------------+-------+----------+
7 rows in set (0.00 sec)
```

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

1. List the cusid and cusname of all customers who made trips in Jan 2017.

```
mysql> select distinct c.cusid, c.cusname
from customer as c natural join trips as t
where month(t.getontime) = '01' and year(t.getontime) = '2017';
+-------+---------+
| cusid | cusname |
+-------+---------+
|     1 | Fred    |
|     4 | Jesse   |
+-------+---------+
2 rows in set (0.00 sec)
```

2. List the cusid of any customer living in Brooklyn who has made a trips with a driver living in Queens .

```
mysql> select distinct c.cusid
from customer c natural join trips t natural join driver d
where cuscity= 'Brooklyn' and dcity= 'Queens';
```

```
+-------+
| cusid |
+-------+
|     1 |
+-------+
1 row in set (0.00 sec)
```

3. Output the cusid and cusname of the customer(s) who took the most expensive trip(s).

```
mysql> Select c.cusid , c.cusname
from customer c natural join trips t
where t.price in
   -> (select max(price) from trips);
+-------+---------+
| cusid | cusname |
+-------+---------+
|     1 | Fred    |
+-------+---------+
1 row in set (0.00 sec)
```

4. Output the car brand that was used in trips by the largest number of distinct customers.

```
mysql> select s.carbrand from
        (select c.carbrand,count(distinct t.cusid) as counter
        from trips t natural join car c group by c.carbrand)as s where s.counter in
        (select max(p.counter)
        from
        (select c.carbrand,count(distinct t.cusid) as counter     from trips t natural join car c group by
        c.carbrand)as p);
+----------+
| carbrand |
+----------+
| Toyota   |
+----------+
1 row in set (0.02 sec)
```

5. Output the did and dname of the driver who earned the most money (sum of prices) during Jan 2017.

```
mysql> select x.did,x.dname
from
        (select d.did,d.dname, sum(t.price) as max_price
        from driver d natural join trips t
        where month(t.getontime)='01' and year(t.getontime)='2017' group by d.did) as x
where x.max_price in
        (select max(y.max_price)
        from
                (select d.did,d.dname, sum(t.price) as max_price
                from driver d natural join trips t
                where month(t.getontime)='01'
```

```
                and year(t.getontime)='2017' group by d.did)
        as y );
+-----+-------+
| did | dname |
+-----+-------+
| 101 | Jay   |
+-----+-------+
```

1 row in set (0.01 sec)

6. For each cuscity, output the cusid of customer living in that city who travelled the longest distance overall.

```
mysql> select c.cusid,c.cuscity
from customer c natural join trips t
where (c.cuscity, t.distance) in
        (select c.cuscity,max(t.distance)
        from customer c natural join trips t
        group by cuscity);
+-------+----------+
| cusid | cuscity  |
+-------+----------+
|     1 | Brooklyn |
|     2 | Queens   |
|     3 | New York |
+-------+----------+
```

3 rows in set (0.00 sec)

7. For each month in 2017, output the customer city that had the largest number of trips during the month.

```
mysql> select month(w.getontime) as month_of_17,s.cuscity,count(*) as no_of_trips
from customer s natural join trips w
group by month_of_17,s.cuscity
having no_of_trips in
        (select max(y.counter)
        from
                (select month(t.getontime) as month,c.cuscity, count(*) as counter
                from trips t natural join customer c
                where year(t.getontime)='2017'
                group by month,c.cuscity)as y
        group by y.month);
+-------------+----------+-------------+
| month_of_17 | cuscity  | no_of_trips |
+-------------+----------+-------------+
|           1 | Brooklyn |           3 |
|           2 | New York |           2 |
+-------------+----------+-------------+
```

2 rows in set (0.00 sec)

8. Output the average duration of trips (getofftime - getontime) for drivers living in Brooklyn, for 2017. The unit of average time should be minutes.

```
mysql> select d.did,avg(time_to_sec(timediff(t.getofftime,t.getontime))/60) as avg_trip_time from trips t
natural join driver d where d.dcity='Brooklyn' and year(t.getontime)='2017' group by d.did;
+-----+---------------+
| did | avg_trip_time |
+-----+---------------+
| 101 |   30.00000000 |
| 103 |   22.50000000 |
+-----+---------------+
2 rows in set (0.01 sec)
```

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

S: Selection

P: Projection

X: Natural JOIN

G: Group By

R: Rename

3) $P_{c.cusid,c.cusname}(S_{t.price}(G_{max(price)}(trips))((R_c(customer) \text{ X } R_t(trips)))$

4) $P_{s.carbrand}(S_{s.counter=}(G_{max(p.counter)}R_p(_{c.carbrand}G_{count(distinct\ t.cusid)\ as\ counter}(R_t(trips) \text{ X } R_c(car))) R_s(_{c.carbrand}G_{count(distinct\ t.cusid)\ as\ counter}\ (R_t(trips) \text{ X } R_c(car))))$

5) $P_{x.did,x.dname}(S_{x.maxprice=}\ (G_{max(y.maxprice)}\ R_y(_{d.did,d.dname}G_{sum(t.price)\ as\ max\_price}\ (S_{month(t.getontime)='01'\ \wedge\ year(t.getontime)='2017'}(R_d(driver) \text{ X } R_t(trips)))))(R_x(_{d.did,d.dname}G_{sum(t.price)\ as\ max\_price}\ (S_{month(t.getontime)='01'\ \wedge\ year(t.getontime)='2017'}\ (R_d(driver) \text{ X } R_t(trips))))))$

6) $P_{c.cusid,c.cuscity}(S_{(c.cuscity,t.distance)=}(_{c.cuscity}G_{max(t.distance)}(R_c(customer) \text{ X } R_t(trips)))(R_c(customer) \text{ X } R_t(trips)))$

7) $P_{month(w.getontime)\ as\ month\_of\_17\ ,\ s.cuscity}\ G_{count(*)\ as\ no\ of\_trips}(S_{no\_of\_trips=}(G_{max(y.counter)}(R_y(_{month(t.getontime)\ as\ month,c.cuscity}Gcount(*)\ as\ counter\ (S_{year(t.getontime='2017')}(R_t(trips) \text{ X } R_c(customer)))))(R_c(customer) \text{ X } R_w(trips)))$

8) $P_{d.did}G_{(time\_to\_sec(timediff(t.getofftime,t.getontime))/60)\ as\ avg\_trips\_time}\ (S_{d.city='Brooklyn'\ \wedge\ year(t.getontime)='2017'}(R_t(trips) \text{ X } R_d(driver))))$

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

(i). For every car of which the size is small , change it to compact .

```
mysql> select * from car;
+-------+----------+---------+
| carid | carbrand | carsize |
+-------+----------+---------+
|   201 | Toyota   | small   |
|   202 | Honda    | medium  |
|   203 | Benz     | medium  |
|   204 | Toyota   | Large   |
|   205 | BMW      | medium  |
|   206 | Ford     | small   |
+-------+----------+---------+
6 rows in set (0.00 sec)

mysql> update car
set carsize='compact'
where carsize='small';
Query OK, 2 rows affected (0.02 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from car;
+-------+----------+---------+
| carid | carbrand | carsize |
+-------+----------+---------+
|   201 | Toyota   | compact |
|   202 | Honda    | medium  |
|   203 | Benz     | medium  |
|   204 | Toyota   | Large   |
|   205 | BMW      | medium  |
|   206 | Ford     | compact |
+-------+----------+---------+
6 rows in set (0.00 sec)
```

ii) Delete tuples of any drivers who do not own a car from the *Driver* table.

```
mysql> select * from driver;
+-----+-------+------------+----------+

| did | dname | dphone     | dcity    |
+-----+-------+------------+----------+
| 101 | Jay   | 9299878765 | Brooklyn |
| 102 | Tom   | 9293487876 | Queens   |
| 103 | Gray  | 9293230987 | Brooklyn |
| 104 | Mary  | 9294569876 | New York |
| 105 | Frank | 9293236745 | New York |
+-----+-------+------------+----------+
5 rows in set (0.00 sec)
```

```
mysql> delete from driver
where did not in (select did from carownership);
Query OK, 1 row affected (0.01 sec)

mysql> select * from driver;
+-----+-------+------------+----------+
| did | dname | dphone     | dcity    |
+-----+-------+------------+----------+
| 101 | Jay   | 9299878765 | Brooklyn |
| 102 | Tom   | 9293487876 | Queens   |
| 103 | Gray  | 9293230987 | Brooklyn |
| 104 | Mary  | 9294569876 | New York |
+-----+-------+------------+----------+
4 rows in set (0.00 sec)
```

(iii) For any trip longer than 30 min, add $5 to the price as a special charge. (We admit that this may not make sense in practice, since you cannot retroactively charge people extra money.)

```
mysql> select * from trips;
+-------+-------+-----+---------------------+---------------------+-------+----------+
| cusid | carid | did | getontime           | getofftime          | price | distance |
+-------+-------+-----+---------------------+---------------------+-------+----------+
|     1 |   201 | 101 | 2017-01-01 10:30:00 | 2017-01-01 11:20:00 |    50 |        8 |
|     1 |   203 | 102 | 2017-01-15 15:25:00 | 2017-01-15 15:45:00 |    30 |        4 |
|     2 |   204 | 103 | 2017-02-15 16:00:00 | 2017-02-15 16:35:00 |    45 |        6 |
|     3 |   202 | 101 | 2017-02-16 14:05:00 | 2017-02-16 14:15:00 |    10 |        2 |
|     3 |   204 | 103 | 2017-02-02 17:10:00 | 2017-02-02 17:20:00 |    15 |      3.5 |
|     4 |   205 | 104 | 2017-02-04 19:00:00 | 2017-02-04 19:18:00 |    28 |      3.5 |
|     4 |   206 | 104 | 2017-01-20 16:45:00 | 2017-01-20 17:10:00 |    33 |        5 |
+-------+-------+-----+---------------------+---------------------+-------+----------+
7 rows in set (0.00 sec)

mysql> update trips set price=price+5
where time_to_sec(timediff(getofftime,getontime))/60>30;
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from trips;
+-------+-------+-----+---------------------+---------------------+-------+----------+
| cusid | carid | did | getontime           | getofftime          | price | distance |
+-------+-------+-----+---------------------+---------------------+-------+----------+
|     1 |   201 | 101 | 2017-01-01 10:30:00 | 2017-01-01 11:20:00 |    55 |        8 |
|     1 |   203 | 102 | 2017-01-15 15:25:00 | 2017-01-15 15:45:00 |    30 |        4 |
|     2 |   204 | 103 | 2017-02-15 16:00:00 | 2017-02-15 16:35:00 |    50 |        6 |
|     3 |   202 | 101 | 2017-02-16 14:05:00 | 2017-02-16 14:15:00 |    10 |        2 |
|     3 |   204 | 103 | 2017-02-02 17:10:00 | 2017-02-02 17:20:00 |    15 |      3.5 |
|     4 |   205 | 104 | 2017-02-04 19:00:00 | 2017-02-04 19:18:00 |    28 |      3.5 |
|     4 |   206 | 104 | 2017-01-20 16:45:00 | 2017-01-20 17:10:00 |    33 |        5 |
+-------+-------+-----+---------------------+---------------------+-------+----------+
7 rows in set (0.00 sec)
```

Problems 2:

In this problem, you have to create views and then write queries on the views, and create triggers, for the schemas in Problem 1. Execute everything on your database system, and report the results.

(a) Define a view that contains for each trip the cusid, cusname, cuscity, carid, did, and getontime. Using this view,

```
mysql> create view glancy_view as
select c.cusid,c.cusname,c.cuscity,t.carid,t.did,t.getontime from trips t natural join customer c;
Query OK, 0 rows affected (0.02 sec)
```

mysql> select * from glancy_view;

```
+-------+-------------+----------+-------+-----+---------------------+
| cusid | cusname     | cuscity  | carid | did | getontime           |
+-------+-------------+----------+-------+-----+---------------------+
|     1 | Fred        | Brooklyn |   201 | 101 | 2017-01-01 10:30:00 |
|     1 | Fred        | Brooklyn |   203 | 102 | 2017-01-15 15:25:00 |
|     2 | Nick        | Queens   |   204 | 103 | 2017-02-15 16:00:00 |
|     3 | Taylor Swift | New York |   202 | 101 | 2017-02-16 14:05:00 |
|     3 | Taylor Swift | New York |   204 | 103 | 2017-02-02 17:10:00 |
|     4 | Jesse       | Brooklyn |   205 | 104 | 2017-02-04 19:00:00 |
|     4 | Jesse       | Brooklyn |   206 | 104 | 2017-01-20 16:45:00 |
+-------+-------------+----------+-------+-----+---------------------+
7 rows in set (0.00 sec)
```

(i) For each customer city, output the customer with the most trips.

```
mysql> select g.cusid,g.cuscity,count(*) as no_of_trips
from glancy_view g
group by g.cusid, g.cuscity
having no_of_trips in
        (select max(y.counter)
        from (select cusid,cuscity,count(*) as counter
                from glancy_view group by cusid,cuscity)as y
         group by y.cuscity);
+-------+----------+-------------+
| cusid | cuscity  | no_of_trips |
+-------+----------+-------------+
|     1 | Brooklyn |           2 |
|     2 | Queens   |           1 |
|     3 | New York |           2 |
|     4 | Brooklyn |           2 |
+-------+----------+-------------+
4 rows in set (0.01 sec)
```

(ii) Change the name of any customer with name Taylor Swift to Katy Perry .

```
mysql> select * from glancy_view;
+-------+--------------+----------+-------+-----+---------------------+
| cusid | cusname      | cuscity  | carid | did | getontime           |
+-------+--------------+----------+-------+-----+---------------------+
|     1 | Fred         | Brooklyn |   201 | 101 | 2017-01-01 10:30:00 |
|     1 | Fred         | Brooklyn |   203 | 102 | 2017-01-15 15:25:00 |
|     2 | Nick         | Queens   |   204 | 103 | 2017-02-15 16:00:00 |
|     3 | Taylor Swift | New York |   202 | 101 | 2017-02-16 14:05:00 |
|     3 | Taylor Swift | New York |   204 | 103 | 2017-02-02 17:10:00 |
|     4 | Jesse        | Brooklyn |   205 | 104 | 2017-02-04 19:00:00 |
|     4 | Jesse        | Brooklyn |   206 | 104 | 2017-01-20 16:45:00 |
+-------+--------------+----------+-------+-----+---------------------+
7 rows in set (0.00 sec)

mysql> update glancy_view
 set cusname='Katy Perry'
where cusname='Taylor Swift';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from glancy_view;
+-------+------------+----------+-------+-----+---------------------+
| cusid | cusname    | cuscity  | carid | did | getontime           |
+-------+------------+----------+-------+-----+---------------------+
|     1 | Fred       | Brooklyn |   201 | 101 | 2017-01-01 10:30:00 |
|     1 | Fred       | Brooklyn |   203 | 102 | 2017-01-15 15:25:00 |
|     2 | Nick       | Queens   |   204 | 103 | 2017-02-15 16:00:00 |
|     3 | Katy Perry | New York |   202 | 101 | 2017-02-16 14:05:00 |
|     3 | Katy Perry | New York |   204 | 103 | 2017-02-02 17:10:00 |
|     4 | Jesse      | Brooklyn |   205 | 104 | 2017-02-04 19:00:00 |
|     4 | Jesse      | Brooklyn |   206 | 104 | 2017-01-20 16:45:00 |
+-------+------------+----------+-------+-----+---------------------+
7 rows in set (0.00 sec)

mysql> select * from customer;
+-------+------------+------------+----------+
| cusid | cusname    | cusphone   | cuscity  |
+-------+------------+------------+----------+
|     1 | Fred       | 9293334444 | Brooklyn |
|     2 | Nick       | 9293435456 | Queens   |
|     3 | Katy Perry | 9293456789 | New York |
|     4 | Jesse      | 9291234567 | Brooklyn |
+-------+------------+------------+----------+
4 rows in set (0.00 sec)
```

(b) Consider the last query in Problem 1(f). Can you write a trigger that automatically adds the special charge when a trip longer than 30min is inserted?

```
mysql> delimiter //
mysql> CREATE TRIGGER special_charge BEFORE INSERT ON trips
   -> FOR EACH ROW
   ->    BEGIN
   ->       IF (time_to_sec(timediff(NEW.getofftime,NEW.getontime))/60>30) THEN
   ->          SET NEW.price = NEW.price+5;
   ->      END IF;
   ->   END;//
Query OK, 0 rows affected (0.04 sec)

mysql> delimiter ;

mysql> select * from trips;
+-------+-------+-----+---------------------+---------------------+-------+----------+
| cusid | carid | did | getontime           | getofftime          | price | distance |
+-------+-------+-----+---------------------+---------------------+-------+----------+
|     1 |   201 | 101 | 2017-01-01 10:30:00 | 2017-01-01 11:20:00 |    55 |        8 |
|     1 |   203 | 102 | 2017-01-15 15:25:00 | 2017-01-15 15:45:00 |    30 |        4 |
|     2 |   204 | 103 | 2017-02-15 16:00:00 | 2017-02-15 16:35:00 |    50 |        6 |
|     3 |   202 | 101 | 2017-02-16 14:05:00 | 2017-02-16 14:15:00 |    10 |        2 |
|     3 |   204 | 103 | 2017-02-02 17:10:00 | 2017-02-02 17:20:00 |    15 |      3.5 |
|     4 |   205 | 104 | 2017-02-04 19:00:00 | 2017-02-04 19:18:00 |    28 |      3.5 |
|     4 |   206 | 104 | 2017-01-20 16:45:00 | 2017-01-20 17:10:00 |    33 |        5 |
+-------+-------+-----+---------------------+---------------------+-------+----------+
7 rows in set (0.01 sec)

mysql> insert into trips values ('4', '205', '104', '2017-02-03 19:00:00', '2017-02-03 19:58:00', '30', '3.5');
Query OK, 1 row affected (0.02 sec)

mysql> select * from trips;
+-------+-------+-----+---------------------+---------------------+-------+----------+
| cusid | carid | did | getontime           | getofftime          | price | distance |
+-------+-------+-----+---------------------+---------------------+-------+----------+
|     1 |   201 | 101 | 2017-01-01 10:30:00 | 2017-01-01 11:20:00 |    55 |        8 |
|     1 |   203 | 102 | 2017-01-15 15:25:00 | 2017-01-15 15:45:00 |    30 |        4 |
|     2 |   204 | 103 | 2017-02-15 16:00:00 | 2017-02-15 16:35:00 |    50 |        6 |
|     3 |   202 | 101 | 2017-02-16 14:05:00 | 2017-02-16 14:15:00 |    10 |        2 |
|     3 |   204 | 103 | 2017-02-02 17:10:00 | 2017-02-02 17:20:00 |    15 |      3.5 |
|     4 |   205 | 104 | 2017-02-03 19:00:00 | 2017-02-03 19:58:00 |    35 |      3.5 |
|     4 |   205 | 104 | 2017-02-04 19:00:00 | 2017-02-04 19:18:00 |    28 |      3.5 |
|     4 |   206 | 104 | 2017-01-20 16:45:00 | 2017-01-20 17:10:00 |    33 |        5 |
+-------+-------+-----+---------------------+---------------------+-------+----------+
8 rows in set (0.00 sec)
```

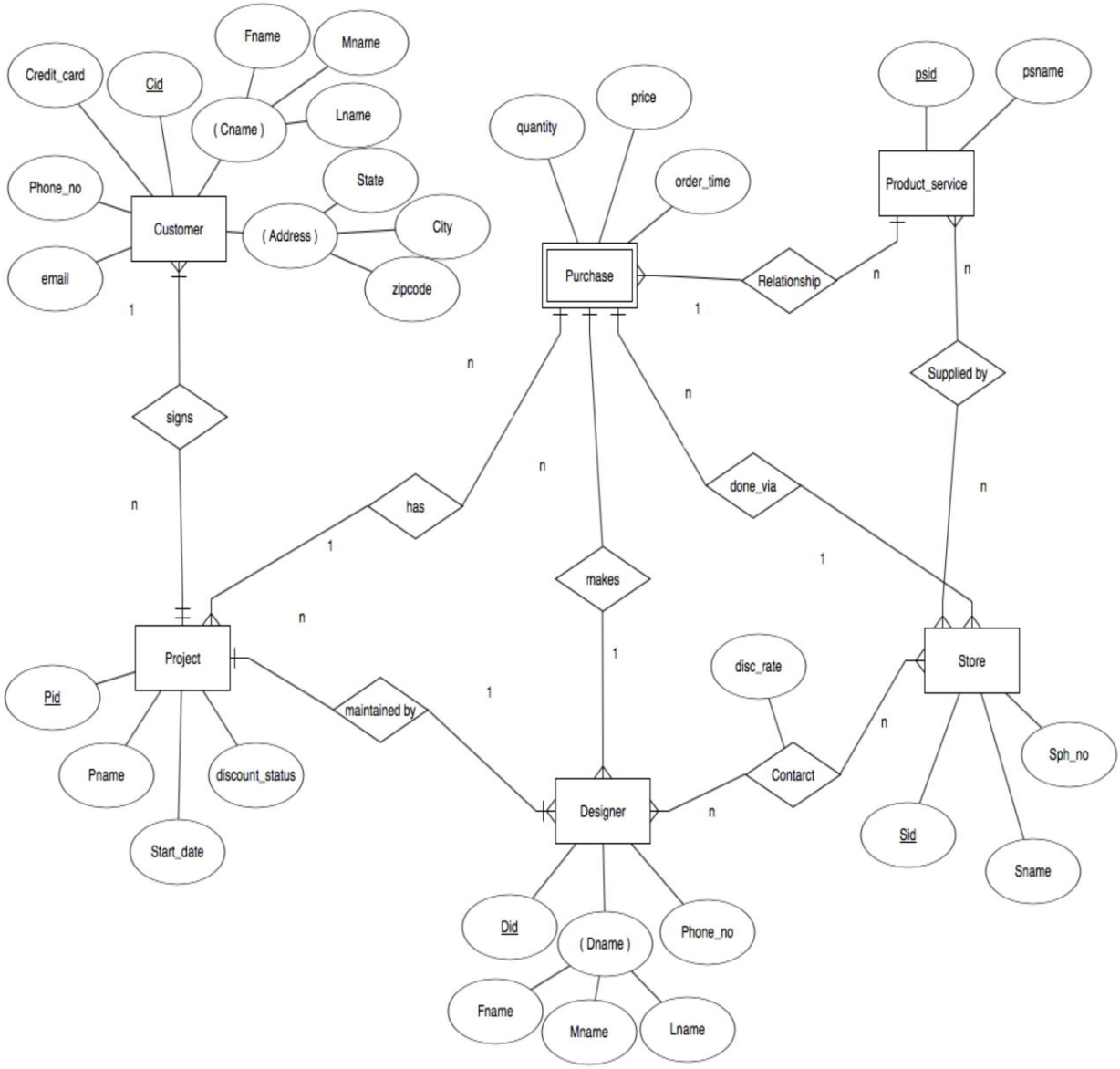
(c) Write a trigger that disables additional trips if a driver has already done 5 trips in the last 60 minutes.

```
mysql> delimiter //
mysql> CREATE TRIGGER Max_trips BEFORE INSERT ON trips
   -> FOR EACH ROW
   ->     BEGIN
   ->         IF ((select count(*) from trips where did=NEW.did and (timestampdiff(minute,getontime,
NEW.getontime)) <= 60)>=5) THEN
   ->
Display all 821 possibilities? (y or n)
   ->  SIGNAL SQLSTATE '45000' set MESSAGE_TEXT = 'Driver already has 5 trips';
   ->       END IF;
   ->    END;
   -> //
Query OK, 0 rows affected (0.02 sec)

mysql> delimiter ;


mysql> INSERT INTO `Trips` (`cusid`, `carid`, `did`, `getontime`, `getofftime`, `price`, `distance`)
VALUES ('3', '204', '103', '2017-03-04 23:19:00', '2017-03-04 23:20:00', '5', '1');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO `Trips` (`cusid`, `carid`, `did`, `getontime`, `getofftime`, `price`, `distance`)
VALUES ('3', '204', '103', '2017-03-04 23:20:00', '2017-03-04 23:21:00', '5', '1');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO `Trips` (`cusid`, `carid`, `did`, `getontime`, `getofftime`, `price`, `distance`)
VALUES ('3', '204', '103', '2017-03-04 23:21:00', '2017-03-04 23:22:00', '5', '1');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO `Trips` (`cusid`, `carid`, `did`, `getontime`, `getofftime`, `price`, `distance`)
VALUES ('3', '204', '103', '2017-03-04 23:24:00', '2017-03-04 23:25:00', '5', '1');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO `Trips` (`cusid`, `carid`, `did`, `getontime`, `getofftime`, `price`, `distance`)
VALUES ('3', '204', '103', '2017-03-04 23:26:00', '2017-03-04 23:27:00', '5', '1');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO `Trips` (`cusid`, `carid`, `did`, `getontime`, `getofftime`, `price`, `distance`)
VALUES ('3', '204', '103', '2017-03-04 23:28:00', '2017-03-04 23:29:00', '5', '1');
ERROR 1644 (45000): Driver already has 5 trips
```

Problem 3

(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.

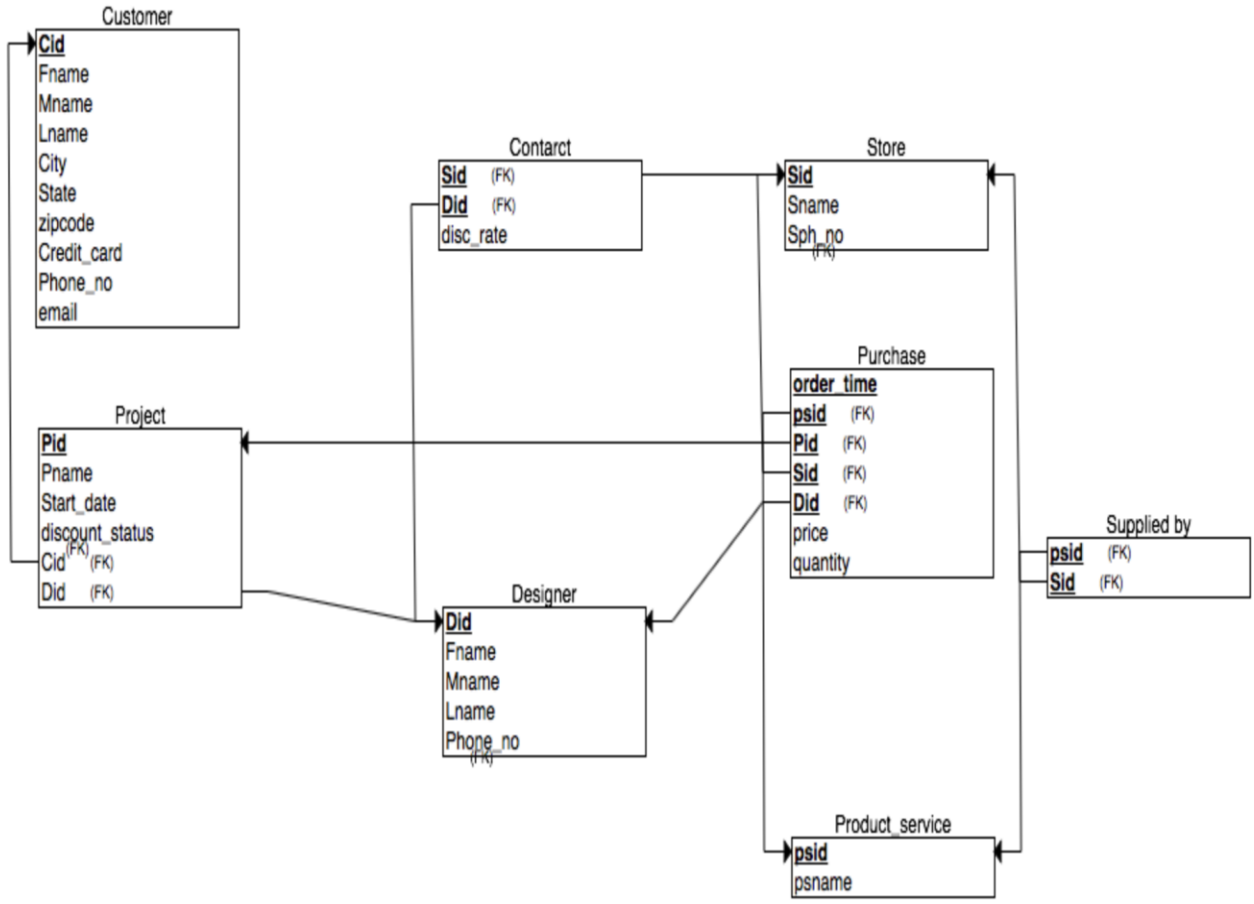
ER diagram:



Assumption: The discount given by stores is the only amount earned by designer.

(b) Convert your ER diagram into a relational schema. Show primary keys and foreign key constrains.

Relational Schema with constrains



(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 designer, output their ID and the total amount of commission they earned from orders during 2016.

```
select pr.did,sum((p.price*p.quantity)*c.disc_rate/100) as total_commission
from purchase p natural join contract c natural join project pr
where pr.disc_status='no' and year(p.order_time)='2016'
 group by pr.did ;
```

(ii) For product ID 481787 and project 82347, output the lowest price and the name of the store offering that price. Take into account any discounts and whether they are passed on to the customer for this project!

```
create view min_prices as
        select t.sname, min(t.price) as min_price
        from  ( select p.price as price ,s.sname
                from purchase p natural join store s natural join project pr
                where p.pid='82347' and p.psid='481787' and pr.disc_status='yes'
                union
                select (p.price-(p.price*c.disc_rate/100)) as price,s.sname
                from purchase p natural join store s natural join project pr natural join contract c
                where p.pid='82347' and p.psid='481787' and pr.disc_status='yes') as t
        group by t.sname;

select sname,min_price from min_prices
where min_price in
        (select min(min_price)
        from min_prices);
```

(iii) For each customer, output the names of their projects and the total amount spent for each project.

```
select t.cid,t.pname,sum(t.amount) as total_amount
from ( select pr.cid,pr.pname,((p.price*p.quantity)-((p.price*p.quantity)*c.disc_rate/100)) as amount
        from purchase  p natural join project pr natural join contract  c
        where pr.disc_status='yes'
        union
        select pr.cid,pr.pname,(p.price*p.quantity) as amount
        from purchase p natural join project pr
        where pr.disc_status='no') as t
group by t.cid,t.pname;
```

(iv) For store “Garden Paradise”, output the ID of the designer who gets the highest discount rate in this store.

```
Select c.did
from  contract c
where c.disc_rate in
          (select max(c1.disc_rate)
          from contract c1 natural join store s
          where sname='Garden Paradise');
```

(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.

mysql> create table customer (cid int, fname varchar(10),mname varchar(10), lname varchar(10), city varchar(10), state varchar(10), zipcode varchar(10), credit_card varchar(10) not null , phone_no varchar(10), email varchar(20), primary key(cid));

Query OK, 0 rows affected (0.02 sec)

mysql> select * from customer;

```
+-----+--------+----------+-----------+-----------+-------+---------+-------------+------------+---------------------+
| cid | fname  | mname    | lname     | city      | state | zipcode | credit_card | phone_no   | email               |
+-----+--------+----------+-----------+-----------+-------+---------+-------------+------------+---------------------+
|   1 | glancy | cajitan  | rodrigues | brooklyn  | ny    | 11209   | 1234567890  | 3476795364 | glancyrod@gmail.com |
|   2 | yash   | shashank | gandhi    | brooklyn  | ny    | 11220   | 2344567890  | 3476794353 | yashgandhi@gmail.com |
|   3 | stalen | peter    | rumao     | charlotte | nc    | 11320   | 9874567890  | 3476796787 | stalenpr@gmail.com  |
+-----+--------+----------+-----------+-----------+-------+---------+-------------+------------+---------------------+
```

3 rows in set (0.01 sec)

mysql> create table designer (did int not null, fname varchar(10), mname varchar(10), lname varchar(10
    ->), phone_no varchar(10), primary key (did));

Query OK, 0 rows affected (0.02 sec)

mysql> select * from designer;

```
+-----+--------+---------+----------+------------+
| did | fname  | mname   | lname    | phone_no   |
+-----+--------+---------+----------+------------+
|  10 | gloria | cajitan | rodrigue | 7276820232 |
|  20 | celina | bernard | rodrigue | 9850603903 |
|  30 | ricky  | james   | rodrigue | 7768976793 |
|  40 | selvin | charles | tuscano  | 7768976794 |
+-----+--------+---------+----------+------------+
```

4 rows in set (0.00 sec)

mysql> create table project(pid int not null, pname varchar(10), start_date datetime, disc_status varchar(10),cid int, did int, primary key(pid), foreign key (cid) references customer(cid), foreign key (did) references designer(did));

Query OK, 0 rows affected (0.02 sec)

mysql> select * from project;

```
+-------+------------+---------------------+-------------+------+------+
| pid   | pname      | start_date          | disc_status | cid  | did  |
+-------+------------+---------------------+-------------+------+------+
| 82347 | frontyard  | 2016-03-04 10:00:00 | yes         |    1 |   10 |
| 82348 | backyard   | 2016-03-02 11:30:00 | no          |    1 |   10 |
| 82349 | backyard   | 2016-03-03 12:30:00 | no          |    2 |   20 |
```

```
| 82350 | home_decor | 2016-05-03 12:30:00 | no          |    2 |   30 |
| 82351 | garden     | 2016-05-06 15:30:00 | yes         |    3 |   40 |
| 82352 | garden     | 2016-05-06 15:30:00 | no          |    3 |   30 |
+-------+------------+---------------------+-------------+------+------+
6 rows in set (0.00 sec)
```

mysql> create table store(sid int not null, sname varchar(15),sph_no varchar(10), primary key(sid));
Query OK, 0 rows affected (0.03 sec)

mysql> select * from store;

```
+-----+-----------------+------------+
| sid | sname           | sph_no     |
+-----+-----------------+------------+
| 100 | Garden Paradise | 7768867481 |
| 101 | Home Decore     | 7768867492 |
| 102 | Wallmart        | 7766786492 |
+-----+-----------------+------------+
3 rows in set (0.00 sec)
```

mysql> create table contract (sid int not null, did int not null, disc_rate int, primary key(sid,did), foreign key (sid) references store(sid), foreign key (did) references designer (did));
Query OK, 0 rows affected (0.03 sec)

mysql> select * from contract;

```
+-----+-----+-----------+
| sid | did | disc_rate |
+-----+-----+-----------+
| 100 |  10 |        10 |
| 100 |  20 |         8 |
| 100 |  30 |         6 |
| 100 |  40 |        15 |
| 101 |  10 |         5 |
| 101 |  30 |         9 |
| 102 |  40 |        10 |
+-----+-----+-----------+
7 rows in set (0.00 sec)
```

mysql> create table prod_serv (psid int not null, psname varchar(20), primary key (psid));
Query OK, 0 rows affected (0.02 sec)

mysql> select * from prod_serv;

```
+--------+------------+
| psid   | psname     |
+--------+------------+
| 481787 | hammer     |
| 488690 | plants     |
| 876758 | carpet     |
| 876760 | plantation |
+--------+------------+
4 rows in set (0.00 sec)
```

```
mysql> create table supply
   -> (psid int not null, sid int not null, primary key(psid,sid));
Query OK, 0 rows affected (0.02 sec)
```

mysql> select * from supply;

```
+--------+-----+
| psid   | sid |
+--------+-----+
| 481787 | 100 |
| 481787 | 101 |
| 481787 | 102 |
| 488690 | 100 |
| 488690 | 101 |
| 876758 | 102 |
| 876760 | 101 |
| 876760 | 102 |
+--------+-----+
8 rows in set (0.00 sec)
```

```
mysql> create table purchase (psid int not null, pid int not null, did int not null, sid int not null,
order_time datetime not null, price int, quantity int, foreign key (psid) references prod_serv(psid), foreign
key(pid) references project(pid), foreign key (sid) references store(sid), foreign key (did) references
designer(did), primary key(psid,pid,did,sid,order_time));
Query OK, 0 rows affected (0.03 sec)
```

mysql> select * from purchase;

```
+--------+-------+-----+-----+---------------------+-------+----------+
| psid   | pid   | did | sid | order_time          | price | quantity |
+--------+-------+-----+-----+---------------------+-------+----------+
| 481787 | 82347 |  10 | 100 | 2016-03-05 08:00:00 |    10 |        2 |
| 481787 | 82347 |  10 | 101 | 2016-03-06 08:00:00 |    11 |        3 |
| 481787 | 82350 |  30 | 100 | 2016-05-08 11:00:00 |    30 |        1 |
| 488690 | 82348 |  10 | 100 | 2016-03-03 09:00:00 |    12 |        3 |
| 488690 | 82352 |  30 | 101 | 2016-05-10 12:30:00 |     9 |        6 |
| 876758 | 82349 |  20 | 100 | 2016-03-07 10:00:00 |    15 |        2 |
| 876760 | 82349 |  20 | 100 | 2016-05-07 10:00:00 |    25 |        4 |
| 876760 | 82351 |  40 | 102 | 2016-05-09 12:00:00 |    20 |        2 |
+--------+-------+-----+-----+---------------------+-------+----------+
8 rows in set (0.00 sec)
```

(i) For each designer, output their ID and the total amount of commission they earned from orders during 2016.

```
mysql> select pr.did,sum((p.price*p.quantity)*c.disc_rate/100) as total_commission
        from purchase p natural join contract c natural join project pr
        where pr.disc_status='no' and year(p.order_time)='2016'
        group by pr.did ;
```

```
+------+------------------+
| did  | total_commission |
+------+------------------+
|   10 |           3.6000 |
|   20 |          10.4000 |
|   30 |           6.6600 |
+------+------------------+
```

3 rows in set (0.00 sec)

(ii) For product ID 481787 and project 82347, output the lowest price and the name of the store offering that price. Take into account any discounts and whether they are passed on to the customer for this project!

```
mysql> create view min_prices as
   ->   select t.sname, min(t.price) as min_price
        from  ( select p.price as price ,s.sname
                from purchase p natural join store s natural join project pr
                where p.pid='82347' and p.psid='481787' and pr.disc_status='yes'
                union
                select (p.price-(p.price*c.disc_rate/100)) as price,s.sname
                from purchase p natural join store s natural join project pr natural join contract c
                where p.pid='82347' and p.psid='481787' and pr.disc_status='yes') as t
        group by t.sname;
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> select sname,min_price
        from min_prices
        where min_price in (select min(min_price)
                            from min_prices);
+-----------------+-----------+
| sname           | min_price |
+-----------------+-----------+
| Garden Paradise |    9.0000 |
+-----------------+-----------+
```

1 row in set (0.01 sec)

(iii) For each customer, output the names of their projects and the total amount spent for each project.

```
mysql> select t.cid,t.pname,sum(t.amount) as total_amount
        from
                ( select pr.cid,pr.pname,((p.price*p.quantity)-((p.price*p.quantity)*c.disc_rate/100)) as
        amount
                from purchase p natural join project pr natural join contract c
                where pr.disc_status='yes'
                union
                select pr.cid,pr.pname,(p.price*p.quantity) as amount
                from purchase p natural join project pr
                where pr.disc_status='no') as t
```

```
        group by t.cid,t.pname;
+------+------------+--------------+
| cid  | pname      | total_amount |
+------+------------+--------------+
|    1 | backyard   |      36.0000 |
|    1 | frontyard  |      49.3500 |
|    2 | backyard   |     130.0000 |
|    2 | home_decor |      30.0000 |
|    3 | garden     |      90.0000 |
+------+------------+--------------+
5 rows in set (0.00 sec)
```

(iv) For store "Garden Paradise", output the ID of the designer who gets the highest discount rate in this store.

```
mysql> Select c.did
        from  contract c
        where c.disc_rate in
                (select max(c1.disc_rate)
                from contract c1 natural join store s
                where sname='Garden Paradise');
+-----+
| did |
+-----+
|  40 |
+-----+
1 row in set (0.00 sec)
```