NYU Tandon School of Engineering March 23, 2020

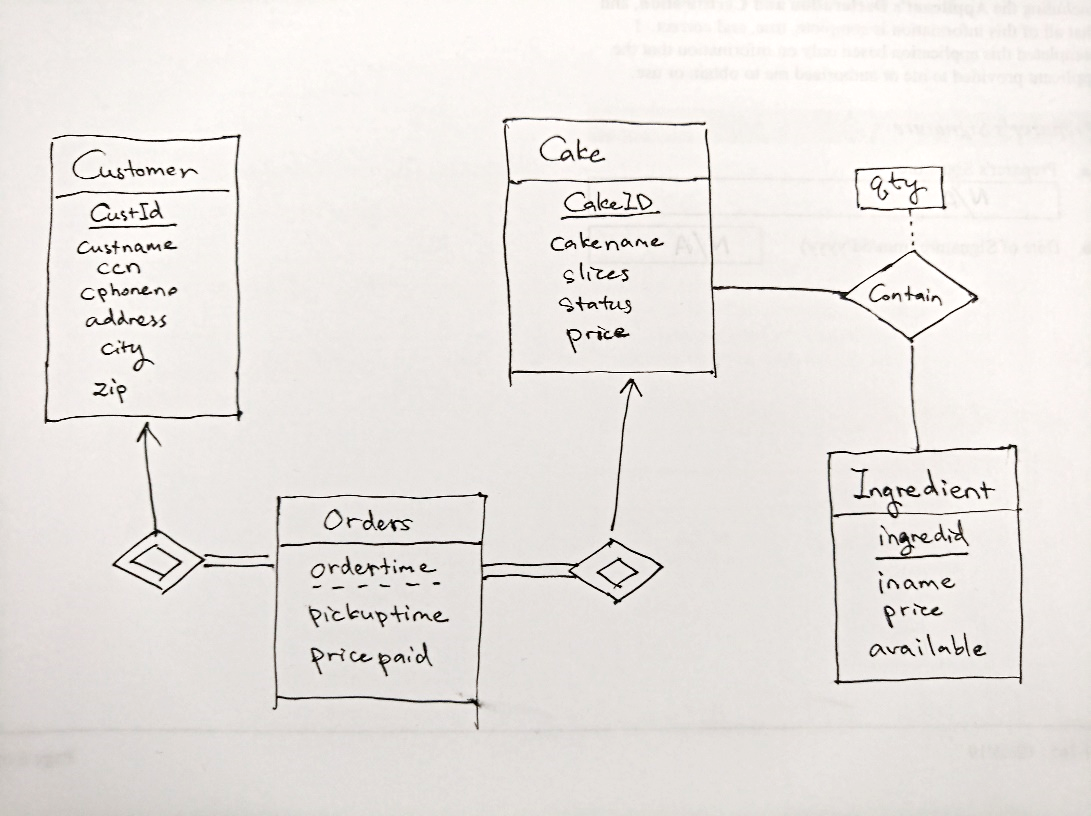
Computer Science and Engineering

CS6083, Spring 2020

**Problem Set #2 Sample Solution**

**Problem 1**

**(a)**



Weak entities: Orders

**(c)**

**I**

-- Using left join to output cakes with zero orders

select cakeid, count(custid) as num

from cake left join (select \* from orders where year(ordertime) = 2014) order\_in\_2014 using(cakeid)

group by cakeid;

A screenshot of a cell phone

Description automatically generated

II

with slice\_price as(

select cakeid, price/slices as unit\_price

from cake

where status = "available")

select cakeid, unit\_price

from slice\_price

where unit\_price = (select max(unit\_price) from slice\_price);

A screenshot of a cell phone

Description automatically generated

III

select cakeid, count(custid) as num

from orders

where year(ordertime) = 2014

group by cakeid

having count(custid) <10;

A screenshot of a cell phone

Description automatically generated

IV

with cake\_cost\_price as (

select cakeid, sum(qty\*price) as cost\_price

from ingredient natural join contain

group by cakeid)

select cakename

from cake natural join cake\_cost\_price

where price < cost\_price \* 2

A screenshot of a cell phone

Description automatically generated

V

with cake\_with\_peanuts as (

select cakeid

from cake natural join contain join ingredient on (contain.ingredid = ingredient.ingredid)

where iname = "peanuts")

select distinct custid, custname

from customer natural join orders natural join cake\_with\_peanuts

where city = "Chicago";

A screenshot of a cell phone

Description automatically generated

VI

with cake\_pair as (

select c1.cakeid as cakeid1, c2.cakeid as cakeid2

from contain c1 join contain c2 on (c1.ingredid = c2.ingredid)

where c1.cakeid < c2.cakeid

group by c1.cakeid, c2.cakeid

having count(\*) >= 3)

select c1.cakename, c2.cakename

from cake\_pair join cake c1 on (cake\_pair.cakeid1 = c1.cakeid)

join cake c2 on (cake\_pair.cakeid2 = c2.cakeid)

A screenshot of a cell phone

Description automatically generated

VII

with cake\_in\_may as (

select cakeid

from orders

where month(ordertime) = 5),

ingredient\_usage as (

select ingredid, sum(qty) as num

from contain natural join cake\_in\_may

group by ingredid)

-- Using left join to output the ingredient

select ingredient.ingredid, ifnull(num,0)

from ingredient left join ingredient\_usage on (ingredient.ingredid = ingredient\_usage.ingredid)

with

cake\_in\_may as (

select cakeid, count(\*) as num

from orders

where month(ordertime) = 5

group by cakeid),

ingredient\_usage as (

select ingredid, sum(qty \* num) as weight

from contain natural join cake\_in\_may

group by ingredid)

-- Using left join to output the ingredient

select ingredient.ingredid, ifnull(weight,0)

from ingredient left join ingredient\_usage on (ingredient.ingredid = ingredient\_usage.ingredid);

A screenshot of a social media post

Description automatically generated

VIII

with

available\_cake as (

select cakeid

from cake

where status = "available"),

qualify\_customer as (

select custid

from orders natural join available\_cake

group by custid

having count(\*) >= (select count(\*) from available\_cake))

select custname

from customer natural join qualify\_customer

A screenshot of a social media post

Description automatically generated

**(d)**

(iv)

(v)

(vi)

(vii)

(viii)

**(d)**

**I**

update cake,

(select cakeid, cost\_price, price

from cake natural join (select cakeid, sum(qty\*price) as cost\_price

from ingredient natural join contain

group by cakeid) as cake\_cost\_price

where price < cost\_price \* 2) as items

set cake.price = items.cost\_price \* 2

where cake.cakeid = items.cakeid

**II**

insert into orders

select cust.custid, c.cakeid, now(), "2020-10-10 00:00:00", 0

from (

select custid

from orders

where ordertime >= "2015-01-01 00:00:00" and ordertime <= "2015-12-31 23:59:59"

group by custid

having sum(pricepaid) >=100) as cust,

(select cakeid from cake where cakename = "Apple Pie") as c;

**III**

update cake,

(select distinct cakeid from orders where ordertime > date\_sub(now(), interval 6 month)) as items

set cake.status = "discontinued"

where cake.cakeid = items.cakeid

**(e)**

I

DELIMITER $$

create trigger free\_apple\_pie after insert on orders

for each row

begin

declare total int;

set @apple\_pie\_cakeid = (select cakeid from cake where cakename="Apple Pie");

set @total = (select sum(pricepaid)

where custid = new.custid and ordertime >= "2015-01-01 00:00:00" and ordertime <= "2015-12-31 23:59:59"

group by custid);

if @total >= 100 then

insert into orders values(new.custid, @apple\_pie\_cakeid, now(), "2020-10-10 00:00:00", 0);

end if;

end

$$

II

Triggers are responses to query events, not time. We need the external job scheduler to do that job.

**Problem 2**

**(a)**

CREATE VIEW branch\_movie\_available\_num AS

    SELECT bid, mid, title, **COUNT**(\*) AS numcopies

    FROM branch NATURAL JOIN copy NATURAL JOIN movie

    WHERE copyid NOT IN (

        SELECT copyid

        FROM rental

        WHERE returndate IS NULL

    )

    GROUP BY bid, mid, title;

(i)

SELECT bid

FROM branch\_movie\_available\_num JOIN

    (SELECT **MAX**(numcopies) AS maxnumcopies

    FROM branch\_movie\_available\_num

    WHERE title = 'Godfather') AS get\_max\_alias

WHERE title = 'Godfather' AND numcopies = maxnumcopies;

Sample output:



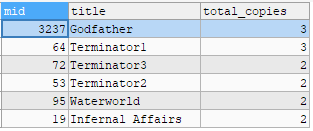
(ii)

SELECT mid, title, **SUM**(numcopies) AS total\_copies

FROM branch\_movie\_available\_num

GROUP BY mid, title;

Sample output:



**(b)**

CREATE VIEW my\_movie\_view AS

    SELECT mid, title, **year**

    FROM movie;

(i)

INSERT INTO my\_movie\_view (mid, title, **year**)

VALUES (6672, 'Minions', 2015);

(ii)

If you have added “ON DELETE CASCADE” when creating table, you can directly perform query:

DELETE FROM my\_movie\_view

WHERE **year** = 1966;

Otherwise, we have to perform a deletion query before that on the table Copy with the foreign key:

DELETE FROM copy

WHERE mid IN (

    SELECT mid

    FROM my\_movie\_view

    WHERE **year** = 1966

);

DELETE FROM my\_movie\_view

WHERE **year** = 1966;

(iii)

We cannot achieve this as the attribute “genre” is not included in the columns of the view.

(iv)

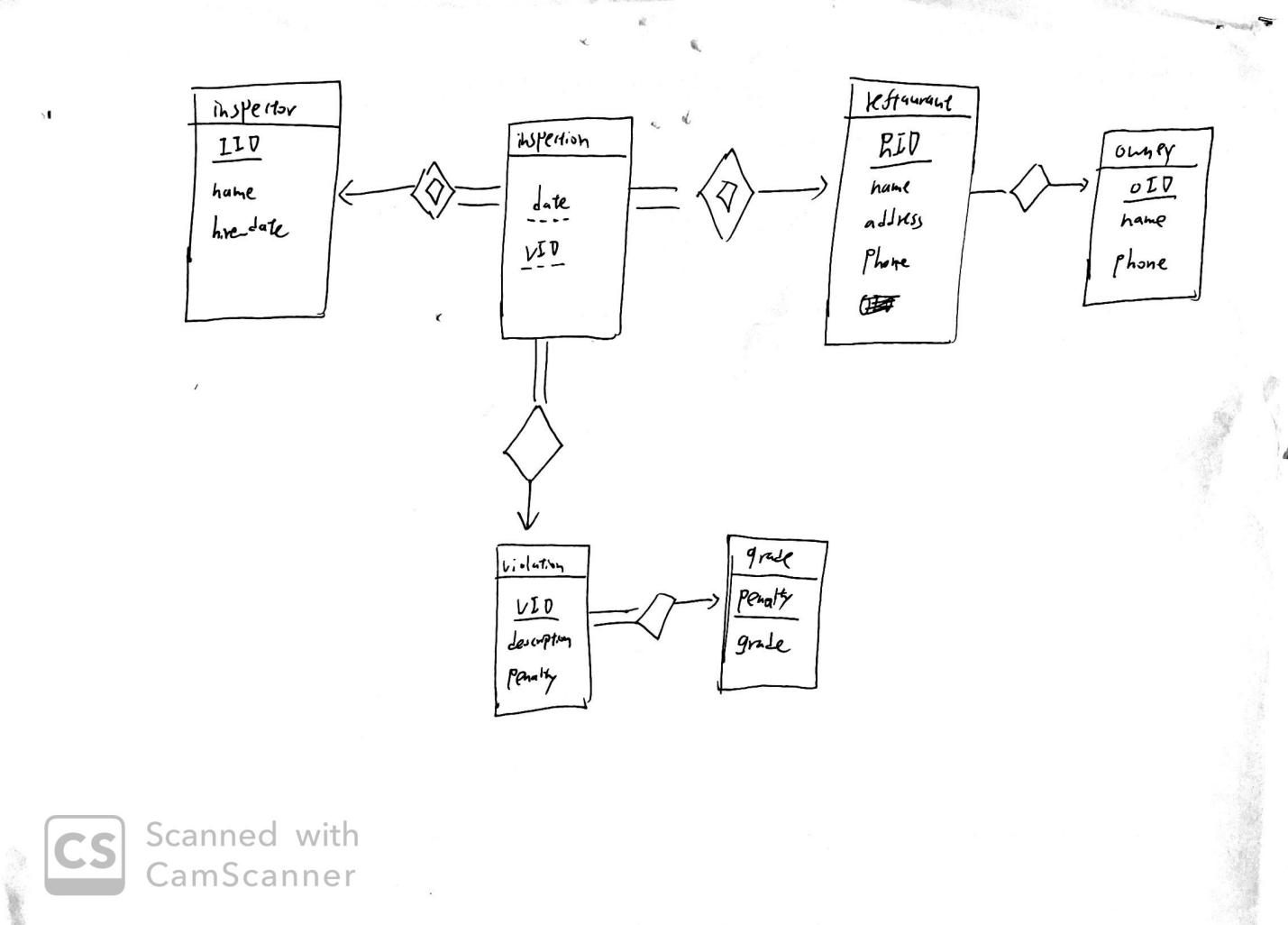
SELECT **year**, **COUNT**(\*) AS movie\_count

FROM my\_movie\_view

GROUP BY **year**;

**Problem 3**

**(a)**

****

**(b)**

Inspector(IID, name, date\_hire)

Inspection(IID, RID, Date, VID)

Violation(VID, description, penalty)

Grade(penalty, grade)

Restaurant(RID, name, address, phone, OID)

Owner(OID, name, phone)

Inspection IID refer to Inspector IID

Inspection RID refer to Restaurant RID

Inspection VID refer to Violation VID

Violation penalty refer to Grade penalty

Restaurant OID refer to Owner OID

**(c)**

1: select i.IID, average(V.penalty) from Inspector as i, Violation as v, Inspection as in where i.IID= in.IID and in.VID=v.VID and year(in.Date)=2014 group by i.IID

2: select r.RID, r.address, g.grade from Restaurant as r, Inspection as in, Violation as v, Grade as g where r.RID= in.RID and in.VID=v.VID and v.penalty=g.penalty group by r.RID order by v.Date descending limit 1

3: with recent\_inspection(name, address, penalty) as

select r.name, r.address, v.penalty from Restaurant as r, Inspection as in, Violation as v where r.RID= in.RID and in.VID=v.VID group by r.RID order by v.Date descending limit 1

select ri.name, ri.address from recent\_inspection as ri where ri.penalty= (select max(penalty) from recent\_inspection)

4:

with recent\_inspection(RID, OID, oname) as

select r.RID, o.OID, o.name from Owner as o, Restaurant as r, Inspection as in, Violation as v where o.OID= r.OID and r.RID= in.RID and in.VID=v.VID and v.description like ‘rat’ group by r.RID order by v.Date descending limit 1

select ri.oname from recent\_inspection as ri group by ri.OID having count(\*)>=2