Homework Assignment 4, by 22000546 Yeeun Lee

1.
(a) rank, dense_rank
(b) range, range current row
(c) trigger
(d) table function
(e) Transaction

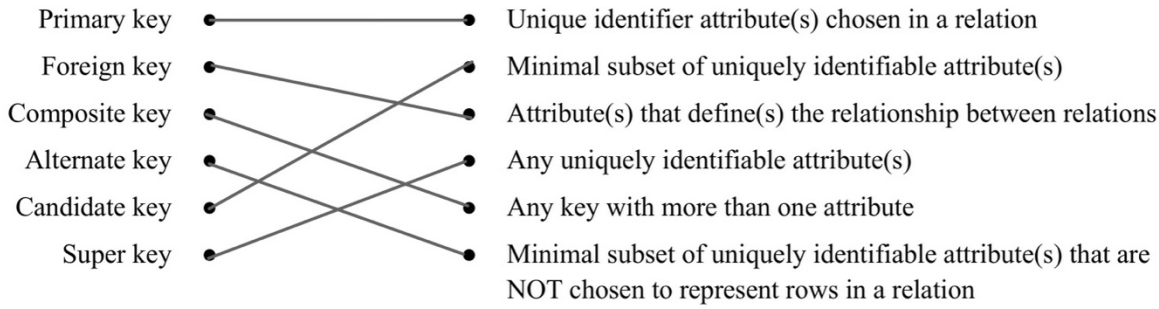
2.
(a)
The with clause applies to one particular query, but the view can be used in multiple queries once defined.

(b)
First, its FROM clause should have only one database relation. Second, SELECT clause contains only attribute names of relation, and does not have any expressions, aggregates, or DISTINCT specification. Thirdly, any attribute not listed in the SELECT clause can be set to null. Lastly, the query does not have a GROUP BY or HAVING clause.

(c)
The ACID properties of a database consist of 4 properties: atomic, consistent, isolated, durable.
Atomicity means that either all operations of the transaction are reflected properly in the database, or none are. Consistency means that execution of a transaction in isolation preserves the consistency of the database. Only valid data is saved. Isolation means that even though multiple transactions may execute concurrently, the system guarantees that transactions do not affect each other. Durability means that after a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

(d)
The computation of each student's rank takes time linear in the size of the relation, leading to an overall time quadratic in the size of the relation. So, the second query is more computationally efficient than the first query. The rank function allows you to use a relatively small amount of time to obtain results as the relationship size increases.

3.



4.

(a)

The result of the natural join of instructor table and teaches table has attributes *(ID, name, dept_name, salary, course_id, section_id, semester, year)*. In this case, the course table also has *course_id* and *dept_name*, which have different meanings. But the two values of *dept_name* should be the same because they are natural joins. Therefore, it is wrong because records with different *dept_name* values are erased.

(b)

select distinct student.ID from student left outer join takes on student.ID = takes.ID where course_id is null;

(c)

select * from student join advisor on ID = s_ID where i_ID is null;

(d)

create view tot_credits(year, num_credits) as

select year, sum(credits) from takes natural join course group by year;

year	num_credits
2001	4530
2004	7085
2005	8805
2009	8984
2008	10686
2010	10728
2007	12194
2003	12953
2002	13438
2006	13873

(e)

The FROM clause does not have only one database relation. It is not a simple view.

5.

(a)

```
select year, month, day, rank() over (order by shares_traded) from nyse;
```

(b)

```
select year, month, monthly, avg(monthly)
over(order by monthly rows between 2 preceding and 0 following)
from (select year, month, sum(dollar_volume) as monthly from nyse group by year, month) as A;
```

(c)

```
with A as (select student, sum(marks) as tot_marks from S group by student)
select student, ranking
from (select *, dense_rank() over (order by tot_marks desc) as ranking from A) as B
where B.ranking <= 10;
```

(d)

```
create trigger depositor_insertion
        after insert on depositor
        referencing new table as inserted
        for each statement
        begin
                insert into branch_cust
                values(select branch_name, customer_name
                        from inserted, account
                        where inserted.account_number = account.account_number);
        end;

create trigger account_insertion
        after insert on account
        referencing new table as inserted
        for each statement
        begin
                insert into branch_cust
                values(select branch_name, customer_name
                        from inserted, depositor
                        where inserted.account_number = depositor.account_number);
        end;
```

(e)

```
create trigger depositor_delete
        after delete on account
        referencing old row as orow
        for each row
        begin
                delete from depositor
                where depositor.customer_name not in
                (select customer_name from depositor
                where account_number <> orow.account_number)
        end;
```

(f)

```
create view tot_credits(year, num_credits) as
select year, sum(credits) from takes natural join course group by year;
```

year	num_credits
2001	4530
2004	7085
2005	8805
2009	8984
2008	10686
2010	10728
2007	12194
2003	12953
2002	13438
2006	13873

(g)

The FROM clause does not have only one database relation. It is not a simple view.

(h)

a.

Transaction 1	Transaction 2
1 read(A)	
2 A := A – 50	
	3 read(A)
	4 A := A + 70
5 write(A)	
	6 write(A)

b.

Suppose that the initial state of X is 100. Assuming that the operation is committed at 5 and 6, the X set at 1-2 and 3-4 are 50 and 170 respectively because only the committed values can be accessed.

This is a different result from the 120 we expected, and the lost update occurred as transaction 2 overwrites transaction 1.

c.
REPEATABLE READ does not allow other transaction to update its data.

6.
(a) null, null, null
(b) null, null, Bob
(c) null, null, null

(d) Jane, Jane, Jane
(e) Jane, Jane, Mary
(f) Jane, Mary, Amy
(g) Jane, Mary, Mary

7.
(a)
select count(*) from store;

count(*)
2

(b)
select count(distinct last_name) from actor;

count(distinct last_na...
121

(c)
select count(*) from rental natural left outer join inventory where return_date is null;

count(*)
183

(d)
select count(distinct customer_id) from rental where staff_id = 1;

count(distinct customer...
599

(e)
select count(distinct film_id) from film where rating = 'PG';

count(distinct film_id)
194

(f)

select count(*) from customer join address on customer.address_id = address.address_id where active = 1 and district = 'England';

count(*)
6

(g)

with A as (select customer_id, count(rental_id) as total_num, sum(amount) as total_money
from payment group by payment.customer_id)
select first_name, last_name, total_num, total_money from A natural join customer
where total_money = (select max(total_money) from A);

first_name	last_name	total_num	total_money
KARL	SEAL	45	221.55

(h)

select category.category_id, category.name, count(category.category_id) from inventory join film_category on inventory.film_id = film_category.film_id
join category on film_category.category_id = category.category_id
where store_id = 2 group by category.category_id
order by count(category.category_id) desc
limit 3;

category_id	name	count(category.category...
15	Sports	181
2	Animation	174
6	Documentary	164

(i)

select title from film_text where length(description) = (select max(length(description)) from film_text)
and film_id in (select distinct film_id from inventory where store_id = 2);

title
CANDIDATE PERDITION

(j)

with A as (select * from
(select film_id, count(rental_id) as rental_num from rental join inventory
on rental.inventory_id = inventory.inventory_id group by film_id) as A
where A.film_id in (select film_id from film_actor

where actor_id = (select actor_id from actor where first_name = 'FRED' and last_name = 'COSTNER')))
select title from film natural join A where rental_num = (select max(rental_num) from A);

title
BROTHERHOOD BLANKET

(k)

select name from customer_list where city = 'London';

name
MATTIE HOFFMAN
CECIL VINES

(l)

select concat(first_name, " ", last_name) as name from customer join address join city
on customer.address_id = address.address_id and address.city_id = city.city_id
where city = 'London';

name
MATTIE HOFFMAN
CECIL VINES