CSCI 6333/6315
Sample Answers to HW 1

1. (100 pts) Consider the insurance database of Fig 1, where the primary keys are underlined. Construct the following SQL queries for this relational database.
    a. Find the total number of people who owned cars that were involved in accidents in 1989.

    select count(*all* drive_id)
    from (owns *inner join* participated *using* (drive_id))
        natural inner join accident
    where *year*(date) = 1989

    Note: the first join cannot be "natural".

    b. Find the number of accidents in which the cars belonging to "John Smith" were involved.

    select count (*distinct* report_number)
    from (person *natural inner join* owns) *inner join* participated
        using (car-license)
    where name = 'John Smith'

    Note: The second natural cannot be "natural".

    c. Add a new accident to the database; assume any values for required attributes.

    insert *into* accident *values* (90000,'2018-10-9 10:30:10', 'Nowhere')
    insert *into* participated *values* (101010, 'NON', 90000, 900)

    d. Delete the Mazda belonging to "John Smith".

    //find the car-license(s) owned by John Smith
    with carsByJohnSmith(car-license) as
        (select car-license
        from person *natural inner join* owns
        where name = 'John Smith')

    //delete the Mazda from cars and from owns
    delete from cars
    where car-license *in* carsByJohnSmith *and* model = 'Mazda'

    delete from owns
    where car-license *in* carsByJohnSmith

e. Update the damage amount for the car with license number "AABB2000" in the accident with report number "AR2197" to $3000.

update participated
set damage-amount = $3000
where car-license = 'AABB2000' *and* report-number = 'AR2197'

person(driver-id, name, address)
car(car-license, model, year)
accident(report-number, date, location)
owns(drive-id, car-license)
participated(driver-id, car-license, report-number, damage amount)

Figure 1. Insurance database

2. (120) Consider the employee database of Fig. 2, where the primary keys are underlined. Given an SQL query for each of the following questions.
   a. Find the names of all employees who work for First Bank Corporation.

   select distinct employee-name
   from (employee *natural join* works) *natural join* company
   where company-name = 'First Bank Corporation';

   b. Find the names and cities of residence of all employees who work for First Bank Corporation.

   select distinct employee-name, city
   from (employee *natural join* works) *natural join* company
   where company-name = 'First Bank Corporation';

   c. Find the names, street addresses, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000.

   select distinct employee-name, street-number, street, city
   from (employee *natural join* works) *natural join* company
   where company-name = 'First Bank Corporation'
   and salary >$10,000;

   d. Find all employees in the database who live in the same cities as the companies for which they work.

   select distinct employee-name, city

from (employee *natural join* works) *natural join* company

Note: the rightmost natural join enforces *company.city = employee.city*.

e. Find all employees in the database who live in the same cities and on the same streets as do their managers.

```
//find managers along their addresses
with manager(ID, street, city) as
        (select employee-id, street, city
         from employee, manages
         where employee.employee-id = manages.manager-id)

//find those employees who live in the same cities and
//on the same street as do their managers.
select distinct employee-name, street, city
from (employee natural join manages), manager
where  manages.manager-id = manager.ID and
          employee.street = manager.street and
          employee.city = manager.city
```

f. Find all employees in the database who do not work for the First Bank Corporation.

```
//find all employees
with allEmploee(ID, name) as
          (select employee-id, employee-name
          from employee),

//find all employees working for the First Bank Corporation
       allFBCEmployee(ID, name) as
          (select employee-id, employee-name
          from (employee natural join works), company
          where  works.company-id = company.company-id and
          company.company-name = 'First Bank Corporation')

//finally, find those who do not work for First Bank Corporation
allEmploee – allFBCEmployee;
```

g. Find all employees in the database who earn more than each employee of Small Bank Corporation.

```
//find max salary of Small Bank Corporation
with maxSalarySBC(maxSalary) as
          (select max(salary)
```

```
        from works natural join company
        where company-name = 'Small Bank Corporation')

//find all employees whose salary is larger than the max salary.
select distinct employee-name
from employee natural join works, maxSalarySBC
where salary > maxSalary;
```

h. Assume that the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.

```
//find cities Small Bank Corporation is located
With citySBC(city) as
        (select distinct city
        from company
        where company-name = 'Small Bank Corporation')

//find all companies located in every of those cities
select distinct C.company-name
from company as C
where   not exists citySBC
        except
        (select distinct company.city
        from company
        where company.name = C.name);
```

i. Find all employees who earn more than the average salary of all employees of their company.

```
//find average salary for each company
with avgSalaryCompany(company-id, avgSalary) as
        (select company-id, avg(salary)
        from works
        group by company-id)

//find all employees earning more than the average
select distinct employee-id, employee-name
from (employee natural join works) natural join avgSalaryCompany
where salary > avgSalary;
```

j. Find the company that has the most employees.

```
//find the total employees for each company
With employeeCount(company-id, employeeCount) as
        (select company-id, count(distinct employee-id)
```

```
        from works
        group by company-id),

//find the max employees
    maxEmployeeCount(maxEmployee) as
        (select max(employeeCount)
        srom employeeCount)

//find the companies with the max employees
select company-id, company-name
from employeeCount natural join company, maxEmployee
where employeeCount = maxEmployee;
```

k. Find the company that has the smallest payroll.

```
//find total salary for each company from works by grouping company-id
with totalPayrollCompany (company-id, totalPayroll) as
        (select company-id, sum(salary)
        from works
        group by company-id),

//find the small payroll
    minPayrollCompany (minPayroll) as
        (select min(totalPayroll)
        from totalPayrollCompany)

//find the smallest total salary
select company-id, company-name
from  totalPayrollCompany natural join company, minPayrollCompany
where totalPayroll = minPayroll;
```

l. Find those companies whose employees earn a higher salary, on average, than the average of First Bank Corporation.

```
//find the average salary for each company
with avgSalaryCompany(company-id, company-name,  avgSalary) as
        (select company-id, company-name, avg(salary)
        from works natural join company
        group by company-id),

//find the average salary of First Bank Corporation
    avgFBC(avgSalary) as
        (select avgSalary
        from avgSalaryCompany
```

where company-name = ‘First Bank Corporation’)

//find those companies.
select distinct company-id, company-name
from avgSalaryCompany, avgFBC
where avgSalaryCompany.avgSalary > avgFBC.avgSalary;

employee(employee-id, employee-name, street, city)
works(employee-id, company-id, salary)
company(company-id, company-name, city)
manages(employee-id, manager-id)

Figure 2. Employee database

Note: For each tuple of the *manages* relation, the manager-id is the manager’s employee id.