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

Figure 4.12. Insurance database.

insert into accident
values (4007, '2001-09-01', 'Berkeley')

insert into participated
select o.driver-id, c.license, 4007, 3000
from person p, owns o, car c
where p.name = 'Jones' and p.driver-id = o.driver-id and
  o.license = c.license and c.model = 'Toyota'

d. Delete the Mazda belonging to “John Smith”.
   Since model is not a key of the car relation, we can either assume that only
   one of John Smith’s cars is a Mazda, or delete all of John Smith’s Mazdas
   (the query is the same). Again assume name is a key for person.

delete car
where model = 'Mazda' and license in
(select license
from person p, owns o
where p.name = 'John Smith' and p.driver-id = o.driver-id)

Note: The owns, accident and participated records associated with the Mazda
still exist.

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 report-number = “AR2197” and driver-id in
(select driver-id
from owns
where license = “AABB2000”)

4.2 Consider the employee database of Figure 4.13, where the primary keys are un-
derlined. Give an expression in SQL for each of the following queries.

a. Find the names of all employees who work for First Bank Corporation.
b. Find the names and cities of residence of all employees who work for 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.
d. Find all employees in the database who live in the same cities as the companies for which they work.
e. Find all employees in the database who live in the same cities and on the same streets as do their managers.
f. Find all employees in the database who do not work for First Bank Corporation.
g. Find all employees in the database who earn more than each employee of Small Bank Corporation.
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.
i. Find all employees who earn more than the average salary of all employees of their company.
j. Find the company that has the most employees.
k. Find the company that has the smallest payroll.
l. Find those companies whose employees earn a higher salary, on average, than the average salary at First Bank Corporation.

Answer:
a. Find the names of all employees who work for First Bank Corporation.

\[ \text{select } \text{employee-name} \]
\[ \text{from } \text{works} \]
\[ \text{where company-name = 'First Bank Corporation'} \]

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

\[ \text{select } e.\text{employee-name}, \text{city} \]
\[ \text{from } \text{employee e, works w} \]
\[ \text{where w.company-name = 'First Bank Corporation'} \text{ and } \]
\[ w.\text{employee-name} = e.\text{employee-name} \]

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

If people may work for several companies, the following solution will only list those who earn more than $10,000 per annum from “First Bank Corporation” alone.

\[ \text{select *} \]
\[ \text{from } \text{employee} \]
\[ \text{where employee-name in} \]
\[ (\text{select employee-name} \]
\[ \text{from } \text{works} \]
\[ \text{where company-name = 'First Bank Corporation'} \text{ and } \text{salary} > 10000) \]

As in the solution to the previous query, we can use a join to solve this one also.

d. Find all employees in the database who live in the same cities as the companies for which they work.
e. Find all employees in the database who live in the same cities and on the same streets as do their managers.

```
select P.employee-name
from employee P, employee R, manages M
where P.employee-name = M.employee-name and
    M.manager-name = R.employee-name and
    P.street = R.street and P.city = R.city
```

f. Find all employees in the database who do not work for First Bank Corporation.
The following solution assumes that all people work for exactly one company.

```
select employee-name
from works
where company-name ≠ 'First Bank Corporation'
```

If one allows people to appear in the database (e.g. in employee) but not appear in works, or if people may have jobs with more than one company, the solution is slightly more complicated.

```
select employee-name
from employee
where employee-name not in
    (select employee-name
     from works
     where company-name = 'First Bank Corporation')
```

g. Find all employees in the database who earn more than every employee of Small Bank Corporation.
The following solution assumes that all people work for at most one company.

```
select employee-name
from works
where salary > all
    (select salary
     from works
     where company-name = 'Small Bank Corporation')
```

If people may work for several companies and we wish to consider the total earnings of each person, the problem is more complex. It can be solved by using a nested subquery, but we illustrate below how to solve it using the with clause.
with emp-total-salary as
  (select employee-name, sum(salary) as total-salary
   from works
   group by employee-name
  )
select employee-name
from emp-total-salary
where total-salary > all
  (select total-salary
   from emp-total-salary, works
   where works.company-name = 'Small Bank Corporation' and
     emp-total-salary.employee-name = works.employee-name
  )

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.

The simplest solution uses the contains comparison which was included in the original System R Sequal language but is not present in the subsequent SQL versions.

select T.company-name
from company T
where (select R.city
   from company R
   where R.company-name = T.company-name)
  contains
  (select S.city
   from company S
   where S.company-name = 'Small Bank Corporation')

Below is a solution using standard SQL.

select S.company-name
from company S
where not exists ((select city
   from company
   where company-name = 'Small Bank Corporation')
  except
  (select city
   from company T
   where S.company-name = T.company-name))

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

The following solution assumes that all people work for at most one company.
employee (employee-name, street, city)
works (employee-name, company-name, salary)
company (company-name, city)
manages (employee-name, manager-name)

Figure 4.13. Employee database.

```
select employee-name
from works T
where salary > (select avg (salary)
    from works S
    where T.company-name = S.company-name)
```

j. Find the company that has the most employees.
```
select company-name
from works
group by company-name
having count (distinct employee-name) >= all
    (select count (distinct employee-name)
        from works
        group by company-name)
```

k. Find the company that has the smallest payroll.
```
select company-name
from works
group by company-name
having sum (salary) <= all (select sum (salary)
    from works
    group by company-name)
```

l. Find those companies whose employees earn a higher salary, on average, than the average salary at First Bank Corporation.
```
select company-name
from works
group by company-name
having avg (salary) > (select avg (salary)
    from works
    where company-name = 'First Bank Corporation')
```

4.3 Consider the relational database of Figure 4.13. Give an expression in SQL for each of the following queries.

a. Modify the database so that Jones now lives in Newtown.

b. Give all employees of First Bank Corporation a 10 percent raise.

c. Give all managers of First Bank Corporation a 10 percent raise.

d. Give all managers of First Bank Corporation a 10 percent raise unless the salary becomes greater than $100,000; in such cases, give only a 3 percent raise.
e. Delete all tuples in the `works` relation for employees of Small Bank Corporation.

**Answer:** The solution for part 0.a assumes that each person has only one tuple in the `employee` relation. The solutions to parts 0.c and 0.d assume that each person works for at most one company.

a. Modify the database so that Jones now lives in Newtown.

```
update employee
set city = 'Newton'
where person-name = 'Jones'
```

b. Give all employees of First Bank Corporation a 10-percent raise.

```
update works
set salary = salary * 1.1
where company-name = 'First Bank Corporation'
```

c. Give all managers of First Bank Corporation a 10-percent raise.

```
update works
set salary = salary * 1.1
where employee-name in (select manager-name
                          from manages)
    and company-name = 'First Bank Corporation'
```

d. Give all managers of First Bank Corporation a 10-percent raise unless the salary becomes greater than $100,000; in such cases, give only a 3-percent raise.

```
update works T
set T.salary = T.salary * 1.03
where T.employee-name in (select manager-name
                          from manages)
    and T.salary * 1.1 > 100000
    and T.company-name = 'First Bank Corporation'
```

```
update works T
set T.salary = T.salary * 1.1
where T.employee-name in (select manager-name
                          from manages)
    and T.salary * 1.1 <= 100000
    and T.company-name = 'First Bank Corporation'
```

SQL-92 provides a *case* operation (see Exercise 4.11), using which we give a more concise solution:-
update works T
set T.salary = T.salary *
  (case
    when (T.salary * 1.1 > 100000) then 1.03
    else 1.1
  )
where T.employee-name in (select manager-name
  from manages) and
  T.company-name = 'First Bank Corporation'
e. Delete all tuples in the works relation for employees of Small Bank Corporation.
delete works
where company-name = 'Small Bank Corporation'

4.4 Let the following relation schemas be given:

\[ R = (A, B, C) \]
\[ S = (D, E, F) \]

Let relations \( r(R) \) and \( s(S) \) be given. Give an expression in SQL that is equivalent to each of the following queries.

a. \( \Pi_A(r) \)
b. \( \sigma_{B=17}(r) \)
c. \( r \times s \)
d. \( \Pi_{A,F}(\sigma_{C=D}(r \times s)) \)

Answer:

a. \( \Pi_A(r) \)
   select distinct A
   from r

b. \( \sigma_{B=17}(r) \)
   select *
   from r
   where B = 17

c. \( r \times s \)
   select distinct *
   from r, s

d. \( \Pi_{A,F}(\sigma_{C=D}(r \times s)) \)
   select distinct A, F
   from r, s
   where C = D

4.5 Let \( R = (A, B, C) \), and let \( r_1 \) and \( r_2 \) both be relations on schema \( R \). Give an expression in SQL that is equivalent to each of the following queries.

a. \( r_1 \cup r_2 \)
b. \( r_1 \cap r_2 \)
c. \( r_1 - r_2 \)
d. \( \Pi_{AB}(r_1) \times \Pi_{BC}(r_2) \)

Answer:

a. \( r_1 \cup r_2 \)
\[
\text{(select * from } r1 \text{ union (select * from } r2 \text{))}
\]
b. \( r_1 \cap r_2 \)
We can write this using the intersect operation, which is the preferred approach, but for variety we present an solution using a nested subquery.
\[
\text{select * from } r1 \text{ where (A, B, C) in (select * from } r2 \text{))}
\]
c. \( r_1 - r_2 \)
\[
\text{select * from } r1 \text{ where (A, B, C) not in (select * from } r2 \text{))}
\]
This can also be solved using the except clause.
d. \( \Pi_{AB}(r_1) \times \Pi_{BC}(r_2) \)
\[
\text{select r1.A, r2.B, r3.C from } r1, r2 \text{ where r1.B = r2.B}
\]

4.6 Let \( R = (A, B) \) and \( S = (A, C) \), and let \( r(R) \) and \( s(S) \) be relations. Write an expression in SQL for each of the queries below:

a. \( \{ <a> \mid \exists b \ ( <a, b> \in r \land b = 17 >) \} \)

b. \( \{ <a, b, c> \mid <a, b> \in r \land <a, c> \in s \} \)

c. \( \{ <a> \mid \exists c \ ( <a, c> \in s \land \exists b_1, b_2 \ ( <a, b_1> \in r \land <c, b_2> \in r \land b_1 > b_2 ) ) \} \)

Answer:

a. \( \{ <a> \mid \exists b \ ( <a, b> \in r \land b = 17 >) \} \)
\[
\text{select distinct A from } r \text{ where B = 17}
\]
b. \( \{ <a, b, c> \mid <a, b> \in r \land <a, c> \in s \} \)