CHAPTER 3

Introdu tion to SQL

Pra ti e Exer ises

3.1 *Write the following queries in SQL, using the university s hema. (We suggest you a tually run these queries on a database, using the sample data that we provide on the web site of the book, db-book. om. Instru tions for setting up a database, and loading sample data, are provided on the above web site.)*

a. *Find the titles of ourses in the Comp. S i. department that have 3 redits.*

b. *Find the IDs of all students who were taught by an instru tor named Einstein; make sure there are no dupli ates in the result.*

. *Find the highest salary of any instru tor.*

d. *Find all instru tors earning the highest salary (there may be more than one with the same salary).*

e. *Find the enrollment of ea h se tion that was o ered in Fall 2017.*

f. *Find the maximum enrollment, a ross all se tions, in Fall 2017.*

g. *Find the se tions that had the maximum enrollment in Fall 2017.*

Answer:

a. *Find the titles of ourses in the Comp. S i. department that have 3 redits.*

```
sele t   title
from     ourse
where    dept_name = 'Comp. S i.' and  redits = 3
```

b. *Find the IDs of all students who were taught by an instru tor named Einstein; make sure there are no dupli ates in the result.*
This query an be answered in several di erent ways. One way is as follows.

```
sele t    distin t takes.ID
from      takes, instru tor, tea hes
where     takes. ourse_id = tea hes. ourse_id and
          takes.se _id = tea hes.se _id and
          takes.semester = tea hes.semester and
          takes.year = tea hes.year and
          tea hes.id = instru tor.id and
          instru tor.name = 'Einstein'
```

. *Find the highest salary of any instru tor.*

```
sele t max(salary)
from instru tor
```

d. *Find all instru tors earning the highest salary (there may be more than one with the same salary).*

```
sele t    ID, name
from      instru tor
where     salary = (sele t max(salary) from instru tor)
```

e. *Find the enrollment of ea h se tion that was o ered in Fall 2017.*

```
sele t    ourse_id, se _id,
          (sele t ount(ID)
          from   takes
          where  takes.year = se tion.year
                 and takes.semester = se tion.semester
                 and takes. ourse_id = se tion. ourse_id
                 and takes.se _id = se tion.se _id)
          as enrollment
from      se tion
where semester = 'Fall'
and       year = 2017
```

Note that if the result of the subquery is empty, the aggregate fun tion ount returns a value of 0.

One way of writing the query might appear to be:

```
sele t    takes. ourse_id, takes.se _id,  ount(ID)
from      se tion, takes
where     takes. ourse _id = se tion. ourse _id
          and takes.se id = se tion.se id_
          and takes.semester = se tion.semester
          and takes.year = se tion.year
          and takes.semester = 'Fall'
          and takes.year = 2017
group by takes. ourse_id, takes.se _id
```

But note that if a se tion does not have any students taking it, it would not appear in the result. One way of ensuring su h a se tion appears with a ount of 0 is to use the outer join operation, overed in Chapter 4.

f. Find the maximum enrollment, a ross all se tions, in Fall 2017.
One way of writing this query is as follows:

```
sele t  max(enrollment)
from    (sele t   ount(ID) as enrollment
         from     se tion, takes
         where    takes.year = se tion.year
                  and takes.semester = se tion.semester
                  and takes. ourse _id = se tion. ourse _id
                  and takes.se _id = se tion.se _id
                  and takes.semester = 'Fall'
                  and takes.year = 2017
                  group by takes. ourse_id, takes.se _id)
```

As an alternative to using a nested subquery in the from lause, it is possible to use a with lause, as illustrated in the answer to the next part of this question.

A subtle issue in the above query is that if no se tion had any enrollment, the answer would be empty, not 0. We an use the alternative using a subquery, from the previous part of this question, to ensure the ount is 0 in this ase.

g. Find the se tions that had the maximum enrollment in Fall 2017.
The following answer uses a with lause, simplifying the query.

```
with  se enrollment as (
      sele t     takes. ourse_id, takes.se id, ount(ID) as enrollment
      from       se tion, takes
      where      takes.year = se tion.year
                 and takes.semester = se tion.semester
                 and takes. ourse id = se tion. ourse id
                 and takes.se id = se tion.se id
                 and takes.semester = 'Fall'
                 and takes.year = 2017
      group by takes. ourse_id, takes.se id)
sele t    ourse_id, se id
from      se enrollment
where     enrollment = (sele t max(enrollment) from se _enrollment)
```

It is also possible to write the query without the with lause, but the subquery to nd enrollment would get repeated twi e in the query. While not in orre t to add distin t in the ount, it is not ne essary in light of the primary key onstraint on takes.

3.2 *Suppose you are given a relation grade points(grade, points) that provides a onversion from letter grades in the takes relation to numeri s ores; for example, an A grade ould be spe i ed to orrespond to 4 points, an A- to 3.7 points, a B+ to 3.3 points, a B to 3 points, and so on. The grade points earned by a student for a ourse o ering (se tion) is de ned as the number of redits for the ourse multiplied by the numeri points for the grade that the student re eived.*

Given the pre eding relation, and our university s hema, write ea h of the following queries in SQL. You may assume for simpli ity that no takes tuple has the null value for grade.

a. *Find the total grade points earned by the student with ID 12345 , a ross all ourses taken by the student.*

b. *Find the grade point average (GPA) for the above student, that is, the total grade points divided by the total redits for the asso iated ourses.*

. *Find the ID and the grade-point average of ea h student.*

d. *Now re onsider your answers to the earlier parts of this exer ise under the assumption that some grades might be null. Explain whether your solutions still work and, if not, provide versions that handle nulls properly.*

Answer:

a. *Find the total grade-points earned by the student with ID 12345 , a ross all ourses taken by the student.*

```
sele t sum( redits * points)
from takes, ourse, grade_points
where takes.grade = grade_points.grade
      and takes. ourse_id = ourse. ourse_id
      and ID = 12345
```

In the above query, a student who has not taken any ourse would not have any tuples, whereas we would expe t to get 0 as the answer. One way of xing this problem is to use the outer join operation, whi h we study later in Chapter 4. Another way to ensure that we get 0 as the answer is via the following query:

```
(sele t  sum( redits * points)
from        takes,   ourse,  grade  points
where takes.grade = grade points.grade
      and takes. ourse_id = ourse. ourse_id
      and ID= 12345 )
union
(sele t  0
from     student
where    ID = 12345 and
      not exists ( sele t * from takes where ID = 12345 ))
```

b. *Find the grade point average (GPA) for the above student, that is, the total grade-points divided by the total redits for the asso iated ourses.*

```
sele t sum( redits * points)/sum( redits) as GPA
from takes, ourse, grade points
where     takes.grade = grade_points.grade
      and takes. ourse_id = ourse. ourse_id
      and ID= 12345
```

As before, a student who has not taken any ourse would not appear in the above result; we an ensure that su h a student appears in the result by using the modi ed query from the previous part of this question. However, an additional issue in this ase is that the sum of redits would also be 0, resulting in a divide-by-zero ondition. In fa t, the only meaningful way of de ning the GPA in this ase is to de ne it as null. We an ensure that su h a student appears in the result with a null GPA by adding the following union lause to the above query.

```
union
(sele t null as GPA
from   student
where ID = 12345 and
      not exists ( sele t * from takes where ID = 12345 ))
```

. *Find the* ID *and the grade-point average of ea h student.*

```
sele t    ID, sum( redits * points)/sum( redits) as GPA
from      takes, ourse, grade_points
where     takes.grade = grade_points.grade
          and takes. ourse_id = ourse. ourse_id
group by ID
```

Again, to handle students who have not taken any ourse, we would have to add the following union lause:

```
union
(sele t ID, null as GPA
from   student
where  not exists ( sele t * from takes where takes.ID = student.ID))
```

d. Now re onsider your answers to the earlier parts of this exer ise under the assumption that some grades might be null. Explain whether your solutions still work and, if not, provide versions that handle nulls properly. The queries listed above all in lude a test of equality on grade between grade_points and takes. Thus, for any takes tuple with a null grade, that student's ourse would be eliminated from the rest of the omputation of the result. As a result, the redits of su h ourses would be eliminated also, and thus the queries would return the orre t answer even if some grades are null.

3.3 Write the following inserts, deletes, or updates in SQL, using the university s hema.

a. In rease the salary of ea h instru tor in the Comp. S i. department by 10%.

b. Delete all ourses that have never been o ered (i.e., do not o ur in the se tion relation).

. *Insert every student whose tot_red attribute is greater than 100 as an instru tor in the same department, with a salary of $10,000.*

Answer:

a. In rease the salary of ea h instru tor in the Comp. S i. department by 10%.

```
update instru tor
set    salary = salary * 1.10
where dept_name = Comp. S i.
```

b. Delete all ourses that have never been o ered (that is, do not o ur in the se tion relation).

person (driver_id, name, address)
ar (li ense_plate, model, year)
a ident (report_number, year, lo ation)
owns (driver_id, li ense_plate)
parti ipated (report_number, li ense_plate, driver_id, damage_amount)

Figure 3.17 Insuran e database

```
delete from ourse
where   ourse_id not in
        (sele t ourse_id from se tion)
```

. Insert every student whose tot_red attribute is greater than 100 as an instru tor in the same department, with a salary of $10,000.

```
insert into instru tor
sele t ID, name, dept_name, 10000
from   student
where  tot_red > 100
```

3.4 Consider the insuran e database of Figure 3.17, where the primary keys are underlined. Constru t the following SQL queries for this relational database.

a. Find the total number of people who owned ars that were involved in a idents in 2017.

b. Delete all year-2010 ars belonging to the person whose ID is 12345 .

Answer:

a. Find the total number of people who owned ars that were involved in a idents in 2017.
Note: This is not the same as the total number of a idents in 2017. We must ount people with several a idents only on e. Furthermore, note that the question asks for owners, and it might be that the owner of the ar was not the driver a tually involved in the a ident.

```
sele t   ount (distin t person.driver_id)
from     a ident, parti ipated, person, owns
where    a ident.report_number = parti ipated.report_number
         and owns.driver_id = person.driver_id
         and owns.li ense_plate = parti ipated.li ense_plate
         and year = 2017
```

b. Delete all year-2010 cars belonging to the person whose ID is 12345.

```
delete car
where year = 2010 and license_plate in
    (select license_plate
     from owns o
     where o.driver_id = 12345 )
```

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

3.5 Suppose that we have a relation marks(ID, score) and we wish to assign grades to students based on the score as follows: grade F if score < 40, grade C if 40 ≤ score < 60, grade B if 60 ≤ score < 80, and grade A if 80 ≤ score. Write SQL queries to do the following:

a. Display the grade for each student, based on the marks relation.

b. Find the number of students with each grade.

Answer:

a. Display the grade for each student, based on the marks relation.

```
select ID,
        case
            when score < 40 then 'F'
            when score < 60 then 'C'
            when score < 80 then 'B'
            else 'A'
        end
from marks
```

b. Find the number of students with each grade.

```
with    grades as
(
sele t  ID,
         ase
              when s ore < 40 then 'F'
              when s ore < 60 then 'C'
              when s ore < 80 then 'B'
              else 'A'
         end as grade
from    marks
)
sele t  grade,  ount(ID)
from    grades
group by grade
```

As an alternative, the with lause an be removed, and instead the de nition of grades an be made a subquery of the main query.

3.6 The SQL like operator is ase sensitive (in most systems), but the lower() function on strings an be used to perform ase-insensitive mat hing. To show how, write a query that nds departments whose names ontain the string s i as a substring, regardless of the ase.

Answer:

```
sele t dept name
from    department
where  lower(dept name) like '%s i%'
```

3.7 Consider the SQL query

```
sele t p.a1
from p, r1, r2
where p.a1 = r1.a1 or p.a1 = r2.a1
```

Under what onditions does the pre eding query sele t values of p.a1 that are either in r1 or in r2? Examine arefully the ases where either r1 or r2 may be empty.

Answer:

The query sele ts those values of p.a1 that are equal to some value of r1.a1 or r2.a1 if and only if both r1 and r2 are non-empty. If one or both of r1 and r2 are empty, the Cartesian produ t of p, r1 and r2 is empty, hen e the result of the query is empty. If p itself is empty, the result is empty.

3.8 Consider the bank database of Figure 3.18, where the primary keys are underlined. Constru t the following SQL queries for this relational database.

```
bran h(bran h_name, bran h_ity, assets)
 ustomer (ID, ustomer_name, ustomer_street, ustomer_ity)
loan (loan_number, bran h_name, amount)
borrower (ID, loan_number)
a ount (a ount number, bran h name, balan e )
depositor (ID, a ount number)
```

Figure 3.18 Banking database.

a. *Find the* ID *of ea h ustomer of the bank who has an a ount but not a loan.*

b. *Find the* ID *of ea h ustomer who lives on the same street and in the same ity as ustomer 12345 .*

. *Find the name of ea h bran h that has at least one ustomer who has an a ount in the bank and who lives in Harrison .*

Answer:

a. *Find the* ID *of ea h ustomer of the bank who has an a ount but not a loan.*

```
(sele t ID
from    depositor)
ex ept
(sele t ID
from    borrower)
```

b. *Find the* ID *of ea h ustomer who lives on the same street and in the same ity as ustomer 12345 .*

```
sele t F.ID
from   ustomer as F, ustomer as S
where F. ustomer street = S. ustomer street
        and F. ustomer_ity = S. ustomer ity
        and S. ustomer id = 12345
```

. *Find the name of ea h bran h that has at least one ustomer who has an a ount in the bank and who lives in Harrison .*

```
sele t  distin t bran h _name
from    a ount, depositor, ustomer
where   ustomer.id = depositor.id
        and depositor.a ount _number = a  ount.a  ount _number
        and ustomer  _ity = 'Harrison'
```

3.9 *Consider the relational database of Figure 3.19, where the primary keys are underlined. Give an expression in SQL for ea h of the following queries.*

a. *Find the ID, name, and ity of residen e of ea h employee who works for First Bank Corporation .*

b. *Find the ID, name, and ity of residen e of ea h employee who works for First Bank Corporation and earns more than $10000.*

. *Find the ID of ea h employee who does not work for First Bank Corporation .*

d. *Find the ID of ea h employee who earns more than every employee of Small Bank Corporation .*

e. *Assume that ompanies may be lo ated in several ities. Find the name of ea h ompany that is lo ated in every ity in whi h Small Bank Corporation is lo ated.*

f. *Find the name of the ompany that has the most employees (or ompanies, in the ase where there is a tie for the most).*

g. *Find the name of ea h ompany whose employees earn a higher salary, on average, than the average salary at First Bank Corporation .*

Answer:

a. *Find the ID, name, and ity of residen e of ea h employee who works for First Bank Corporation .*

---

employee (ID, person_name, street, ity)
works (ID, ompany _name, salary)
ompany (ompany_name, ity)
manages (ID, manager_id)

Figure 3.19 Employee database.

```
sele t e.ID, e.person_name, ity
from employee as e, works as w
where w. ompany_name = First Bank Corporation and
      w.ID = e.ID
```

b. Find the ID, name, and ity of residen e of ea h employee who works for First Bank Corporation and earns more than $10000.

```
sele t *
from employee
where ID in
   (sele t ID
    from works
    where ompany_name = First Bank Corporation and salary > 10000)
```

This ould be written also in the style of the answer to part a.

. Find the ID of ea h employee who does not work for First Bank Corporation .

```
sele t ID
from works
where ompany_name <> First Bank Corporation
```

If one allows people to appear in employee without appearing also in works, the solution is slightly more ompli ated. An outer join as discussed in Chapter 4 ould be used as well.

```
sele t ID
from employee
where ID not in
   (sele t ID
    from works
    where ompany_name = First Bank Corporation )
```

d. Find the ID of ea h employee who earns more than every employee of Small Bank Corporation .

```
sele t ID
from works
where salary > all
   (sele t salary
   from works
    where ompany_name = Small Bank Corporation )
```

If people may work for several ompanies and we wish to onsider the total earnings of ea h person, the problem is more omplex. But note that the

fa t that ID is the primary key for works implies that this annot be the ase.

e. Assume that ompanies may be lo ated in severalities. Find the name of ea h ompany that is lo ated in every ity in whi h Small Bank Corporation is lo ated.

```
sele t S. ompany_name
from ompany as S
where not exists ((sele t ity
                   from ompany
                   where ompany name = Small Bank Corporation )
                  ex ept
                   (sele t ity
                   from ompany as T
                   where S. ompany_name = T. ompany_name))
```

f. Find the name of the ompany that has the most employees (or ompanies, in the ase where there is a tie for the most).

```
sele t ompany name
from works
group by ompany name
having ount (distin t ID) >= all
    (sele t ount (distin t ID)
    from works
     group by ompany name)
```

g. Find the name of ea h ompany whose employees earn a higher salary, on average, than the average salary at First Bank Corporation .

```
sele t ompany name
from works
group by ompany name
having avg (salary) > (sele t avg (salary)
                       from works
                       where ompany name = First Bank Corporation )
```

3.10 Consider the relational database of Figure 3.19. Give an expression in SQL for ea h of the following:

a. Modify the database so that the employee whose ID is 12345 now lives in Newtown .

b. Give ea h manager of First Bank Corporation a 10 per ent raise unless the salary be omes greater than $100000; in su h ases, give only a 3 per ent raise.

Answer:

a. *Modify the database so that the employee whose ID is 12345 now lives in Newtown .*

```
update employee
set city = Newtown
where ID = 12345
```

b. *Give each manager of First Bank Corporation a 10 percent raise unless the salary becomes greater than $100000; in such cases, give only a 3 percent raise.*

```
update works T
set T.salary = T.salary * 1.03
where T.ID in (select manager_id
               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.ID in (select manager_id
               from manages)
      and T.salary * 1.1 <= 100000
      and T.company_name = First Bank Corporation
```

The above updates would give different results if executed in the opposite order. We give below a safer solution using the case statement.

```
update works T
set T.salary = T.salary *
    (case
        when (T.salary * 1.1 > 100000) then 1.03
        else 1.1
    end)
where T.ID in (select manager_id
               from manages) and
      T.company_name = First Bank Corporation
```