2.14 List two reasons why null values might be introduced into the database.

Ans: Nulls may be introduced into the database because the actual value is either unknown or does not exist.

For example, an employee whose address has changed and whose new address is not yet known should be retained with a null address. If employee tuples have a composite attribute dependent, and a particular employee has no dependents, then that tuple's dependents attribute should be given a null.

2.15 Discuss the relative merits of procedural and nonprocedural languages.

Ans: Nonprocedural languages greatly simplify the specification of queries (at least, the types of queries they are designed to handle). That free the user from having to worry about how the query is to be evaluated; not only does this reduce programming effort, but in fact in most situations the query optimizer can do a much better task of choosing the best way to evaluate a query than a programmer working by trial and error.

On the other hand, procedural languages are far more powerful in terms of what computations they can perform. Some tasks can either not be done using nonprocedural languages, or are very hard to express using nonprocedural languages, or execute very inefficiently if specified in a nonprocedural manner.

3.11 Write the following queries in SQL, using the university schema.

a. Find the names of all students who have taken at least one Comp. Sci. course; make sure there are no duplicate names in the result.

Ans:

```
select  distinct name
from    student natural join takes natural join course
where   course.dep = 'Comp.Sci';
```

b. Find the IDs and names of all students who have not taken any course offering before Spring 2009.

Ans:

```
select  id, name
from    student natural join takes
except
select  id, name
from    student natural join takes
where   year<2009;
```

c. For each department, find the maximum salary of instructors in that department. You may assume that every department has at least one instructor.

Ans:
```
select   dept_name, max(salary)
from     instructor
group by dept_ name;
```

d. Find the lowest, across all departments, of the per-department maximum salary computed by the preceding query.

Ans:
```
select   min(maximum_salary)
from     (select  dept_name, max(salary) as maximum_salary
from     instructor
group by dept_name;
```

3.12 Write the following queries in SQL, using the university schema.

a. Create a new course "CS-001", titled "Weekly Seminar", with 0 credits.

Ans: insert into *Course* (course_id , title,dept_name , credits)

values ('CS-001' , 'Weekly Seminar ', 'Comp. Sci' , 0);

b. Create a section of this course in Autumn 2009, with *sec id* of 1.

Ans: insert into *Section* (course_id , sec_id , semester,year)

values ('CS-001','1','Fall',2009);

c. Enroll every student in the Comp. Sci. department in the above section.

Ans: Insert into takes(id,course_id,sec_id,semester,year)

Select id,'CS-001','1','Fall',2009

From *student*

Where *dpet_name*='Comp. Sci'

d. Delete enrollments in the above section where the student's name is Chavez.

Ans: Delete from takes

Where *course_id* ='CS-001'

and *sec_id* ='1'

and *semester* ='Fall'

and *year* ='2009'

and *id* in (select id from student where name='Chavez')

e. Delete the course CS-001. What will happen if you run this delete statement without first deleting offerings (sections) of this course.

Ans: delete from *course*

where *course_id* = 'CS-001'

f. Delete all *takes* tuples corresponding to any section of any course with the word "database" as a part of the title; ignore case when matching the word with the title.

Ans: delete from takes

where *course_id* in (select *course_id*

from *course*

where lower(title) like '%database%')

3.14 Consider the insurance database of Figure 3.18, where the primary keys are underlined. Construct the following SQL queries for this relational database.

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

Ans:

```
select count (distinct *)
from accident
where exists
(select *
from participated, person
where participated.driver-id = person.driver-id
and person.name = 'John Smith'
and accident.report-number = participated.report-number)
```

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

Ans:

```
update participated
set damage-amount = 3000
where report-number = "AR2197" and driver-id in
(select driver-id
from owns
where license = "AABB2000")
```

3.15 Consider the bank database of Figure 3.19, where the primary keys are underlined. Construct the following SQL queries for this relational database.

a. Find all customers who have an account at *all* the branches located in "Brooklyn".

Ans:

```
with branch count as
(select count (*) branch
where branch city = 'Brooklyn')
select customer name
from customer c where branch count =
(select count (distinct branch name)
from (customer natural join depositor natural join account
natural join branch) as d
where d.customer name = c.customer name)
```

b. Find out the total sum of all loan amounts in the bank.

Ans:

```
select sum(amount) from loan
```

c. Find the names of all branches that have assets greater than those of at least one branch located in "Brooklyn".

Ans:

```
select branch name
from branch
where assets > some
(select assets from branch
where branch city = 'Brooklyn')
```

3.17 Consider the relational database of Figure 3.20. Give an expression in SQL for each of the following queries.

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

Ans: update works

set salary = salary * 1.1

where company name = 'First Bank Corporation'

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

Ans: update works

set salary = salary * 1.1

where employee name in

(select manager name from manages) and

company name = 'First Bank Corporation

c. Delete all tuples in the *works* relation for employees of "Small Bank Corporation".

Ans: delete from works

where company name = 'Small Bank Corporation'

3.21 Consider the library database of Figure 3.21 Write the following queries in SQL.

a. Print the names of members who have borrowed any book published by "McGraw-Hill".

Ans: select name

from member m, book b, borrowed l

where m.memb no = l.memb no

and l.isbn = b.isbn and b.publisher = 'McGrawHill'

b. Print the names of members who have borrowed all books published by “McGraw-Hill”.

Ans: select distinct m.name

from member m

where not exists ((select isbn from book

where publisher = ’McGrawHill’) except

(select isbn from borrowed l

where l.memb no = m.memb no))

c. For each publisher, print the names of members who have borrowed more than five books of that publisher.

Ans: select publisher, name

from (select publisher, name, count (isbn)

from member m, book b, borrowed l

where m.memb no = l.memb no

and l.isbn = b.isbn group by publisher, name) as

membpub(publisher, name, count books)

where count books > 5

d. Print the average number of books borrowed per member. Take into account that if a member does not borrow any books, then that member does not appear in the *borrowed* relation at all.

Ans: with memcount as

(select count (*) from member)

select count (*)/memcount from borrowed

3.22 Rewrite the where clause

where unique (select *title* from *course*)

without using the unique construct.

***member*(*memb no*, *name*, *age*)**

***book*(*isbn*, *title*, *authors*, *publisher*)**

***borrowed*(*memb no*, *isbn*, *date*)**

Ans: where (

(select count(title)

from course) =

(select count (distinct title)

from course))

3.24 Consider the query:

with *dept total* (*dept name*, *value*) as

(select *dept name*, sum(*salary*)

from *instructor*

group by *dept name*),

***dept total avg*(*value*) as**

(select avg(*value*)

from *dept total*)

select *dept name*

from *dept total*, *dept total avg*

where *dept total.value* >= *dept total avg.value*;

Rewrite this query without using the with construct.

Ans: select distinct dept name d

from instructor i

where

(select sum(salary)

```
from instructor
where department = d) >=
(select avg(s)
from
        (select sum(salary) as s
        from instructor group by department))
```