**3.5**

a.
```
SELECT id, (CASE
          WHEN score < 40 THEN 'F'
          WHEN score < 60 THEN 'C'
          WHEN score < 80 THEN 'B'
          ELSE 'A'
  END) as grade FROM marks;
```

b.
```
SELECT grade, COUNT(id) FROM (
          SELECT id, (CASE
                  WHEN score < 40 THEN 'F'
                  WHEN score < 60 THEN 'C'
                  WHEN score < 80 THEN 'B'
                  ELSE 'A'
          END) as grade FROM marks) as grades
  GROUP BY grade;
```

**3.11**

a. SQL query:

```
select  name
from    student natural join takes natural join course
where   course.dept = 'Comp. Sci.'
```

b. SQL query:

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

Since the **except** operator eliminates duplicates, there is no need to use a **select distinct** clause, although doing so would not affect correctness of the query.

c. SQL query:

```
select  dept, max(salary)
from    instructor
group by dept
```

d. SQL query:

```
select  min(maxsalary)
from    (select dept, max(salary) as maxsalary
         from instructor
         group by dept)
```

**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.

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

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

**Answer:**

a. Give all employees of First Bank Corporation a 10-percent raise. (the solution assumes that each person works for at most one company.)

```
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.

```
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.

```
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".

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

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

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

**Answer:**

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

```
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. (We assume that all books above refers to all books in the *book* relation.)

```
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.

```
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
```

The above query could alternatively be written using the **having** clause.

d. Print the average number of books borrowed per member.

```
with memcount as
    (select count(*)
    from member)
select count(*)/memcount
from borrowed
```

Note that the above query ensures that members who have not borrowed any books are also counted. If we instead used **count**(**distinct** *memb_no*) from *borrowed*, we would not account for such members.