3.11

(a)

```
SELECT DISTINCT name
FROM student NATURAL JOIN takes NATURAL JOIN course
WHERE course.dept='Comp. Sci.'
```

(b)

```
SELECT ID, name
FROM student
EXCEPT
SELECT ID, name
FROM student NATURAL JOIN takes
WHERE year<2009
```

(c)

```
SELECT dept, MAX(salary)
FROM instructor
GROUP BY dept
```

(d)

```
SELECT dept, MAX(salary)
FROM (
        SELECT dept, MAX(salary) AS max_salary
        FROM instructor
        GROUP BY dept
)
```

3.12

(a)

```
INSERT INTO course(course_id, title, dept_name, credits)
VALUES('CS-001', 'Weekly Seminar', 'Comp. Sci.', 0)
```

(b)

```
INSERT INTO section(course_id, sec_id, semester, year)
VALUE('CS-001', 1, 'Fall', 2009)
```

(c)

```
INSERT INTO takes(ID, course_id, section_id, semester, year)
SELECT ID, 'CS-001', '1', 2009
FROM student
WHERE dept_name = 'Comp. Sci.'
```

(d)

```
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 FROM course
WHERE course_id = 'CS-001'
```

Since the course_id in section is a foreign key referenced from course, there will be no error and it will delete any tuples within the section table that has a course_id of 'CS-001'.

(f)

```
DELETE FROM takes
WHERE course_id IN (
        SELECT course_id
        FROM course
        WHERE title LIKE '%database%'
        )
```

3.14

(a)

```
SELECT count(report_number)
FROM accident NATURAL JOIN participated
WHERE driver_id IN (
        SELECT driver_id
        FROM person
        WHERE name='John Smith'
        )
```

(b)

```
UPDATE participated
SET damage_amount=3000
WHERE (license='AABB2000') AND (report_number='AR2197')
```

3.15

(a)

```
SELECT DISTINCT D.customer_name
FROM depositor AS D
WHERE NOT EXISTS (
        SELECT branch_name
        FROM branch
        WHERE branch_city='Brooklyn'
)

EXCEPT
```

```
SELECT A.branch_name
FROM depositor AS B, account as A
WHERE (B.account_number=A.account_number) AND
(D.customer_name=B.customer_name)
```

(b)

```
SELECT sum(amount)
FROM loan
```

(c)

```
SELECT branch_name
FROM branch
WHERE assets>any(
        SELECT assets
        FROM branch
        WHERE branch_city='Brooklyn'
)
```

3.16

(a)

```
SELECT employee_name
FROM works
WHERE company_name = 'First Bank Corporation'
```

(b)

```
SELECT employee_name
FROM employee NATURAL JOIN works NATURAL JOIN company
```

(c)

```
SELECT E.employee_name
FROM employee E, employee M, manages
WHERE (manages.employee_name=E.employee_name) AND
(M.employee_name=manages.manager_name) AND (M.street=E.street) AND (M.city=E.city)
```

(d)

```
SELECT employee_name
FROM works M
WHERE salary>(
        SELECT avg(salary)
        FROM works P
        WHERE M.company_name=P.company_name
)
```

(e)

```
SELECT company_name
FROM works
GROUP BY company_name
HAVING sum(salary)<=all(
        SELECT sum(salary)
        FROM works
        GROUP BY company_name
)
```

3.17

(a)

```
UPDATE works
SET salary=salary*1.1
WHERE company_name='First Bank Corporation'
```

(b)

```
UPDATE works
SET salary=salary*1.1
WHERE employee_name IN (
        SELECT manager_name
        FROM managers
) AND company_name='First Bank Corporation'
```

(c)

```
DELETE FROM works
WHERE company_name='Small Bank Corporation'
```

3.18

Two reasons that null values are introduced into the database is because the actual value is either unknown or doesn't exist.

3.19

Assume we have, value <> ALL(...), then the value will not be equal to each and every value within the list. In this regard, the function performs similarly if we have value <> NOT IN (...).