**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 (...).