Parley Pacheco Martins 1484000 - Assignment 2

1 - c)

$$\begin{aligned} sbc\_employees &\leftarrow \sigma_{company\_name="Small\ Bank\ Corporation"}(works) \\ \Pi_{person\_name}(\sigma_{works.salary>sbc\_employees.salary}(works \bowtie sbc\_emplyees)) \end{aligned} \tag{1}$$

2 - a)

$$\Pi_{person\_name}(\sigma_{company\_name="First\ Bank\ Corporation"}) \tag{2}$$

c)

$$\begin{aligned} jobs &\leftarrow employee \bowtie works \\ fbc &\leftarrow \sigma_{company\_name="First\ Bank\ Corporation",salary>10{,}000}(jobs) \\ &\Pi_{person\_name,street,city}(fbc) \end{aligned} \tag{3}$$

3 -

$$\Pi_{customer\_name,customer\_city}(borrower \bowtie customer) \tag{4}$$

a) Jackson does not appear in the results because he is not in the customer relation (as seen in Figure 2.4). When we include the attribute *city* in our projection, we remove Jackson from our results.

b) I would make the attribute *customer_name* in the borrower relation a foreign key, forcing any borrower to be a bank customer.

c)

$$\Pi_{customer\_name,customer\_city}(borrower \rtimes customer) \tag{5}$$

4 a)

$$\Pi_{account\_number}(G_{\mathbf{count}(account\_number)>1}(depositor)) \tag{6}$$

5 a)

$$\begin{aligned} \{t \mid \exists\ s \in works(t[person\_name] = s[person\_name] \wedge \\ s[company\_name] = "First\ Bank\ Corporation"\} \end{aligned} \tag{7}$$

c)

$$\begin{aligned} \{t \mid \exists\ s \in works(t[person\_name] = s[person\_name] \wedge \\ s[company\_name] = "First\ Bank\ Corporation" \wedge \\ s[salary] > 10,000) \wedge \\ \exists\ u \in employee(u[person\_name] = s[person\_name] \wedge \\ t[street] = u[street] \wedge t[city] = u[city])\} \end{aligned} \tag{8}$$

6 -

a) SELECT s.stuName, s.program, r.courseId

FROM student AS s, register AS r
WHERE s.studentId = r.studentid and courseId like "CSC*"
ORDER BY s.stuName, r.courseId DESC;

b) SELECT DISTINCT s.studentId, s.stuName
FROM student AS s, studentArea AS a, register AS r, offering AS o
WHERE a.depth = "major" and a.area="CSC" and s.studentId = a.studentId
and r.studentId = s.studentId and r.courseId = o.courseId and r.sectId = o.sectId
and ((r.grade < 2 and o.term = "Fall") or (r.approval = No and o.term="Winter"));

c) SELECT s.studentId, s.stuName, avg(r.grade) AS gradeAvg
FROM student AS s, register AS r, offering AS o
WHERE s.studentId=r.studentId AND r.courseId=o.courseId AND r.sectId=o.sectId
AND o.term="Fall"
GROUP BY s.studentId, s.stuName
HAVING avg(r.grade) > 2.5
ORDER BY s.stuName;

d) SELECT p.profName
FROM prof AS p, teach AS t, register AS r
WHERE p.profId = t.profId and t.courseId = r.courseId and t.sectId = r.sectId
and r.approval = Yes
GROUP BY p.profName
HAVING count(r.studentId) > 3;

e)SELECT s.studentId, s.stuName
FROM student AS s
WHERE 2 < all(select grade from register AS r, offering AS o
where s.studentId = r.studentId
and r.courseId = o.courseId and r.sectId = o.sectId and o.term = "Fall")
GROUP BY s.studentId, s.stuName
ORDER BY s.stuName;

f) SELECT DISTINCT p.profName
FROM prof AS p
WHERE (((p.profName) Not In (select p.profName
FROM prof AS p, teach AS t, offering AS o, location AS l
WHERE p.profId = t.profId and t.sectid = o.sectId and t.courseId = o.courseId
and o.term="Winter" and l.courseId = o.courseId and l.sectId = o.sectId and
l.time like "*F*")))
ORDER BY p.profName;

g)(I know this isn't the most efficient way, but I couldn't use that temporary
table that a regular sql implementation has)
SELECT s.studentId, s.stuName, r.grade
FROM student AS s, register as r

WHERE r.courseId = "PHY102" and s.studentId = r.studentId and r.grade =(select min(grade) from register where courseId = "PHY102");

h) SELECT o.courseId, o.sectId, count(r.studentId) AS qty_students
FROM register r right join offering o on r.courseId = o.courseId and r.sectId = o.sectId
GROUP BY o.courseId, o.sectId
ORDER BY o.courseId asc, o.sectId asc;

i)
j)