Database 4125

Project Phase 3

Due: May 1, 2017

Michael Kelly & William Sobol

Queries

1. SELECT person\_id, p.p\_name

FROM has\_job NATURAL JOIN job j NATURAL JOIN Person p

WHERE comp\_id = ? ;

(test = 1)

SELECT person.p\_name, person\_id, pay\_rate

FROM Person NATURAL JOIN Job NATURAL JOIN has\_job

WHERE comp\_id = 5 AND pay\_type LIKE 'salary'

ORDER BY pay\_rate DESC;

(No error, no results. Can't figure out why this one doesn't work. Also need to use LIKE for string comparison 'salary')

1. SELECT comp\_id, c.c\_name, SUM(hourly\_pay\_rate, salary\_pay\_rate)

FROM (SELECT comp\_id, c.c\_name,

(SELECT SUM(pay\_rate \* 1920) as hourly\_pay\_rate

FROM job NATURAL JOIN company

WHERE pay\_type = ‘hourly’) ,

SELECT SUM(pay\_rate) as salary\_pay\_rate

FROM job NATURAL JOIN company

WHERE pay\_type=’salary’)

FROM job NATURAL JOIN company c)

GROUP BY comp\_id, c.c\_name ;

(Error: Invalid number of arugments in the first row, assuming it's the SUM function)

1. SELECT person\_id, p\_name, job\_id

FROM person NATURAL JOIN has\_job NATURAL JOIN job

WHERE person\_id=? (test = 3)

1. SELECT person\_id, p\_name, k\_code, title

FROM person NATURAL JOIN has\_skill NATURAL JOIN knowledge\_skill

WHERE person\_id=? ; (test = 4)

Queries: #6- 10 & #15 : {William Sobol}; revised by {Michael Kelly}

6.List the skill gap of a worker between his/her job(s) and his/her skills.

SELECT k\_code

FROM (SELECT k\_code

FROM has\_job NATURAL JOIN required

WHERE person\_id=?)

MINUS

(SELECT k\_code

FROM has\_skill

WHERE person\_id=?) ;

(test = 15)

7. List the required knowledge/skills of a job/ a job category in a readable format. (two queries)

1. SELECT k\_code, title

FROM knowledge\_skill NATURAL JOIN required

WHERE job\_id = ?;

(test = 7)

1. SELECT k\_code, title

FROM job\_category NATURAL JOIN core NATURAL JOIN knowledge\_skill

WHERE cate\_code = ?;

(test = 8, should work but not showing up, *core* shows that job\_cat 8 has 2 corresponding k\_codes)

8. List a person’s missing knowledge/skills for a specific job in a readable format.

SELECT k\_code, *title*

FROM (SELECT k\_code

FROM required

WHERE job\_id = ?)

MINUS

(SELECT k\_code

FROM has\_skill

WHERE person\_id = ?);

(test = 7,4 -- title does not have a foreign key relationship in required and has\_skill according to the schema so it can't show title)

9. List the courses (course id and title) that each alone teaches all the missing knowledge/skills for a person to pursue a specific job.

WITH skill\_gap AS(

SELECT k\_code, title

FROM ( SELECT k\_code, title

FROM required

WHERE job\_id = ?)

MINUS

(SELECT k\_code, title

FROM has\_skill

WHERE person\_id = ?) )

SELECT course\_id, C.title

FROM course C

WHERE NOT EXISTS ( ( SELECT k\_code

FROM skill\_gap)

MINUS

SELECT k\_code

FROM course\_skill as CS

WHERE C.course\_id = CS.course\_id) );

(Error, keep getting a parenthesis error no matter where I put one. Also get a flashback query error at the FROM course\_skill as CS clause)

10.WITH skill\_gap AS (

SELECT k\_code, ***title***

FROM ( SELECT k\_code, ***title***

FROM required

WHERE job\_id = ?)

MINUS

( SELECT k\_code, ***title***

FROM has\_skill

WHERE person\_id = ? ) ) ,

gap\_courses AS (

SELECT course\_id

FROM course C

WHERE NOT EXISTS (

(SELECT k\_code

FROM skill\_gap)

MINUS

(SELECT k\_code

FROM course\_skill CS

WHERE C.course\_id = CS.course\_id ) ) )

SELECT course\_id, sec\_id, semester, year

FROM gap\_courses NATURAL JOIN section

WHERE completion\_date = (SELECT MIN(completion\_date)

FROM gap\_courses NATURAL JOIN section);

(no error, no results same *title* problem. I think the problem here is getting a MIN on a date type it may not be compatible?)

11. Find the cheapest course to make up one’s skill gap by showing the course to take and the cost (of the section price).

WITH skill\_gap AS (

SELECT k\_code, ***title***

FROM ( SELECT k\_code, ***title***

FROM required

WHERE job\_id = ?)

MINUS

( SELECT k\_code, ***title***

FROM has\_skill

WHERE person\_id = ?) ) ,

gap\_courses AS (

SELECT course\_id

FROM course C

WHERE NOT EXISTS (

(SELECT k\_code

FROM skill\_gap)

MINUS

(SELECT k\_code

FROM course\_skill CS

WHERE C.course\_id = CS.course\_id)))

SELECT course\_id, sec\_no, completion\_date

FROM gap\_courses NATURAL JOIN section

WHERE completion\_date = (SELECT MIN(completion\_date)

FROM gap\_courses NATURAL JOIN section);

(no error, no results. Basically the same query as above)

13.

SELECT cate\_code

FROM job\_category C

WHERE NOT EXISTS(

(SELECT k\_code

FROM job\_category NATURAL JOIN required

WHERE cate\_code = C.cate\_code)

MINUS

(SELECT k\_code

FROM has\_skill

WHERE person\_id = ?));

(no results)

14.

WITH jobs\_qualified\_for AS(

SELECT job\_id, pay\_rate

FROM job J

WHERE NOT EXISTS(

(SELECT k\_code

FROM required

WHERE job\_id = J.job\_id)

MINUS

(SELECT k\_code

FROM has\_skill

WHERE person\_id = ?)))

SELECT job\_id, pay\_rate

FROM jobs\_qualified\_for

WHERE pay\_rate = (SELECT MAX(pay\_rate)

FROM jobs\_qualified\_for);

(One job comes up for every person\_id, I looked it up and that job requires no skill and is also the highest paying job of them all, so technically it works.)

15. List all the names along with the emails of the persons who are qualified for a job.

SELECT person\_id, email

FROM person P

WHERE NOT EXISTS(

(SELECT k\_code

FROM required

WHERE job\_id = ?)

MINUS

(SELECT k\_code

FROM has\_skill

WHERE has\_skill.person\_id = P.person\_id)) ;

(test = 3)

16. When a company cannot find any qualified person for a job, a secondary solution is to find a person who is almost qualified to the job. Make a “missing-one” list that lists people who miss only one skill for a specified job.

SELECT person\_id

FROM person P

WHERE ( SELECT COUNT(\*)

FROM (( SELECT k\_code

FROM required

WHERE job\_id = ? )

MINUS

(SELECT k\_code

FROM has\_skill

WHERE person\_id = P.person\_id))) = 1 ;

(test = 3 / 26)

Queries #17 - #18 : {Michael Kelly} revised by {Michael Kelly}

17.

WITH skills\_required AS (

SELECT k\_code

FROM required

WHERE job\_id = ? ) ,

skill\_gap AS (

SELECT person\_id, k\_code

FROM ( ( SELECT person\_id, k\_code FROM person, required)

MINUS

(SELECT person\_id, k\_code FROM has\_skill) ) ),

persons\_missing\_one AS (

SELECT person\_id

FROM skill\_gap

GROUP BY person\_id

HAVING COUNT(k\_code) = 1)

SELECT k\_code, COUNT(k\_code)

FROM persons\_missing\_one NATURAL JOIN skill\_gap

GROUP BY k\_code

ORDER BY COUNT(k\_code) ASC ;

(no error but no results)

18. WITH required\_skills AS(

SELECT k\_code

FROM required

WHERE job\_id= ?),

skills\_gap AS(

SELECT person\_id, k\_code

FROM ((SELECT person\_id, k\_code FROM person, required\_skills)

MINUS

(SELECT person\_id, k\_code FROM has\_skill))),

skills\_missing (person\_id, amt) AS(

SELECT person\_id, COUNT(k\_code)

FROM skills\_gap

GROUP BY person\_id)

SELECT person\_id, amt

FROM skills\_missing

WHERE amt = (SELECT MIN(amt) FROM skills\_missing);

(test = 1 / 7)

19.

WITH required\_skills AS(

SELECT k\_code

FROM core

WHERE cate\_code = ? ) ,

Skills\_gap AS(

SELECT person\_id, k\_code

FROM ((SELECT person\_id, k\_code FROM person, required\_skills)

MINUS

(SELECT person\_id, k\_code FROM has\_skill) ) )

SELECT person\_id, COUNT(\*)

FROM skills\_gap

GROUP BY person\_id

HAVING COUNT(\*) <= ? ;

(test = 1,1 / 2,2)

20.

WITH required\_skills AS(

SELECT k\_code

FROM core

WHERE cate\_code = 1 ) ,

skills\_gap AS(

SELECT person\_id, k\_code

FROM ((SELECT person\_id, k\_code FROM person, required\_skills)

MINUS

(SELECT person\_id, k\_code FROM has\_skill))),

persons\_missing\_skills\_number AS(

SELECT person\_id

FROM skills\_gap

GROUP BY person\_id

HAVING COUNT(\*) <= 1)

SELECT k\_code, COUNT(person\_id) as personCount

FROM persons\_missing\_skills\_number NATURAL JOIN skills\_gap

GROUP BY k\_code

ORDER BY personCount DESC;

(test = 1,1 / 2,2)

21.

WITH all\_profiles as (

SELECT has\_job.person\_id, cate\_code

FROM has\_job JOIN job ON has\_job.person\_id = job.person\_id )

SELECT person\_id, name

FROM person NATURAL JOIN all\_profiles

WHERE cate\_code = ‘1001100’;

\* SELECT DISTINCT person\_id

FROM has\_job NATURAL JOIN job NATURAL JOIN job\_category

WHERE cate\_code = ?;\*

(test = 2 | I think this works better, I was having trouble with the ON clause)

22.

WITH past\_jobs as (

SELECT person\_id, job\_id

FROM person NATURAL JOIN has\_job

MINUS

SELECT person\_id, job\_id

FROM has\_job

WHERE end\_date = null),

past\_job\_profiles as (

SELECT past\_jobs.person\_id , job.cate\_code

FROM past\_jobs join job on job.job\_id = past\_jobs.job\_id)

SELECT person\_id

FROM past\_job\_profiles

WHERE cate\_code = ?;

\*\*SELECT person\_id

FROM has\_job

WHERE job\_id = 2 AND end\_date IS NOT NULL;

(test = 2, I think this is better?)

23.

{By number of employees}

WITH count\_employees (comp\_id, employee\_total) as (

SELECT comp\_id, COUNT (DISTINCT(person\_id))

FROM job NATURAL JOIN company

GROUP BY comp\_id)

SELECT comp\_id

FROM count\_employees

WHERE employee\_total = (SELECT MAX(employee\_total)

FROM count\_employees);

…second query on next page

{By salary}

WITH labor\_cost as (

SELECT comp\_id, SUM(tot\_pay) labor\_cost

FROM ( ( SELECT comp\_id, SUM(pay\_rate \*1920) tot\_pay

FROM job

WHERE pay\_type = ‘hourly’ AND person\_id IS NOT NULL

GROUP BY comp\_id) )

GROUP BY comp\_id

ORDER BY labor\_cost DESC)

SELECT comp\_id

FROM labor\_cost

WHERE labor\_cost.labor\_cost = (SELECT MAX(labor\_cost)

FROM labor\_cost);

24.

{by employee count}

WITH count\_employees (primary\_sector, employee\_amount) as

( SELECT primary\_sectory, COUNT(DISTINCT(person\_id) )

FROM job NATURAL JOIN company

GROUP BY primary\_sector)

SELECT primary\_sector

FROM count\_employees

WHERE employee\_amount = (SELECT MAX (employee\_amount)

FROM count\_employees);.