Missed Queries

The queries missed in phase_2

4

Given a person's identifier. find all the jobs this person is currently holding and worked in the past.

```
1
2  SELECT job_code,
3         start_date,
4         end_date
5  FROM   works
6         NATURAL JOIN job
7  WHERE  per_id = 1
8  ORDER  BY start_date DESC;
```

Fix - deleted category did not need.

8

Given a person's id, list a person's missing knowledge/skills for a specific job in a readable form.

```
1  (SELECT ks_code,
2          ks_title
3   FROM   required_skill
4          NATURAL JOIN knowledge_skill
5   WHERE  job_code = 2)
6  MINUS
7  (SELECT ks_code,
8          ks_title
9   FROM   has_skill
10         NATURAL JOIN knowledge_skill
11  WHERE  per_id = 1);
```

Fix - do not need job table already in required skill

9

Given a person identifier and a job code, list the courses (course id and title) that each alone teaches all the missing knowledge/skills for this person to pursue the specific job.

```
1
2  WITH missing_ks(ks)
3       AS ((SELECT ks_code
4            FROM   required_skill
5            WHERE  job_code = 1)
6           MINUS
7           (SELECT ks_code
8            FROM   has_skill
9            WHERE  per_id = 1))
10 SELECT c_code,
11        c_title
12 FROM   course c
13 WHERE  NOT EXISTS((SELECT *
14                    FROM   missing_ks)
15                   MINUS
16                   (SELECT ks_code
17                    FROM   teaches_skill ts
18                    WHERE  ts.c_code = c.c_code));
```

Fix - did not need job tables job_code in required_skill

10

```
1
2  WITH missing_ks(ks)
3       AS ((SELECT ks_code
4            FROM   required_skill
5            WHERE  job_code = 1)
6           MINUS
7           (SELECT ks_code
8            FROM   has_skill
9            WHERE  per_id = 1)),
10      fulfilling_courses(c_code)
11      AS (SELECT c_code
12          FROM   course c
13          WHERE  NOT EXISTS ((SELECT *
14                              FROM   missing_ks)
```

```
                            MINUS
                            (SELECT ks_code
                             FROM   teaches_skill ts
                             WHERE  ts.c_code = c.c_code))),
     fulfilling_section(c_code, complete_date)
     AS (SELECT DISTINCT c_code,
                         complete_date
         FROM   SECTION
                NATURAL JOIN fulfilling_courses
         WHERE  complete_date >= Trunc(SYSDATE))
SELECT c_code,
       complete_date
FROM   fulfilling_section
WHERE  complete_date = (SELECT Min(complete_date)
                        FROM   fulfilling_section);
```

Fix - did not need job and c.c_code should have been ts.c_code.

11

Suppose the skill gap of a worker and the requirement of a desired job can be covered by one course. 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 missing_ks(ks)
     AS ((SELECT ks_code
          FROM   required_skill
          WHERE  job_code = 1)
         MINUS
         (SELECT ks_code
          FROM   has_skill
          WHERE  per_id = 1)),
     fulfilling_courses(c_code, c_title, retail_price)
     AS (SELECT c_code,
                c_title,
                retail_price
         FROM   course c
         WHERE  NOT EXISTS ((SELECT *
                             FROM   missing_ks)
                            MINUS
                            (SELECT ks_code
                             FROM   teaches_skill ts
```

```
                                WHERE  ts.c_code = c.c_code)))
SELECT c_code,
       c_title,
       retail_price
FROM   fulfilling_courses
WHERE  retail_price = (SELECT MIN(retail_price)
                       FROM   fulfilling_courses
                              NATURAL JOIN SECTION);
```

12 – unable to do query.

13

Given a person's identifier, list all the job categories that a person is qualified for.

```

SELECT cate_code,
       cate_title
FROM   job_category jc
WHERE  NOT EXISTS ((SELECT ks_code
                    FROM   skill_set ss
                    WHERE  jc.cate_code = ss.cate_code)
                   MINUS
                   (SELECT ks_code
                    FROM   has_skill
                    WHERE  per_id = 2));
```

Fix- we don's why this was marked incorrect we have a set of ks_code that job_category have and we subtract it to see if a person fulfills this skill set.

14.

Given a person's identifier, find the job with the highest pay rate for this person according to his/her skill possession.

```

WITH qualified_jobs
     AS (SELECT j.job_code
         FROM   job j
         WHERE  NOT EXISTS ((SELECT ks_code
                             FROM   required_skill rs
```

```
                              WHERE  j.job_code = rs.job_code)
                            MINUS
                            (SELECT ks_code
                             FROM   has_skill
                             WHERE  per_id = 1))),
     q_jobs_desc
     AS (SELECT *
         FROM   job
                NATURAL JOIN qualified_jobs)
SELECT job_code,
       pay_rate,
       pay_type
FROM   q_jobs_desc
WHERE  pay_rate = (SELECT Max(CASE
                                WHEN pay_type = 'salary' THEN pay_rate
                                ELSE pay_rate * 1920
                              END)
                   FROM   q_jobs_desc
```

Fix - had to just redo.

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 per_id,
       per_name
FROM   person p
WHERE  1 = (SELECT Count(ks_code)
            FROM   ((SELECT ks_code
                     FROM   required_skill
                     WHERE  job_code = 1)
                    MINUS
                    (SELECT ks_code
                     FROM   has_skill hs
                     WHERE  hs.per_id = p.per_id)));
```

Fix- error in formatting.

17

List each of the skill code and the number of people who misses the skill and are in the missing-one list for a given job code in the ascending order of the people counts.

```
1
2  WITH skills_needed(ks_code)
3       AS (SELECT ks_code
4           FROM   required_skill
5           WHERE  job_code = '1'),
6       missing_skills(per_id, ms_count)
7       AS (SELECT per_id,
8                  Count(ks_code)
9           FROM   person p,
10                 skills_needed
11          WHERE  ks_code IN ((SELECT ks_code
12                              FROM   skills_needed)
13                             MINUS
14                             (SELECT ks_code
15                              FROM   has_skill
16                              WHERE  per_id = p.per_id))
17          GROUP  BY per_id)
18 SELECT ks_code,
19        Count(per_id) AS total_ms_count
20 FROM   missing_skills ms,
21        skills_needed
22 WHERE  ks_code IN ((SELECT ks_code
23                     FROM   skills_needed)
24                    MINUS
25                    (SELECT ks_code
26                     FROM   has_skill
27                     WHERE  per_id = ms.per_id))
28        AND ms_count = 1
29 GROUP  BY ks_code
30 ORDER  BY total_ms_count ASC;
```

Fix

- skills_needed -> find the skills needed for a particular job
- missing_skills -> find people who are missing these skills and count of ms
- cross join each skill has 1 person that we can enumerate

18

Suppose there is a new job that has nobody qualified. List the persons who miss the least number of skills that are required for this job and report the "least number".

```
1
2  WITH skills_needed(ks_code)
3       AS (SELECT ks_code
4           FROM   required_skill
5           WHERE  job_code = 1),
6       missing_skills(per_id, ms_count)
7       AS (SELECT per_id,
8                  Count(ks_code)
9           FROM   person p,
10                 skills_needed sn
11          WHERE  sn.ks_code IN ((SELECT ks_code
12                                 FROM   required_skill)
13                                MINUS
14                                (SELECT ks_code
15                                 FROM   has_skill
16                                 WHERE  per_id = p.per_id))
17          GROUP  BY per_id),
18      min_missing_ks(min_ms_count)
19      AS (SELECT Min(ms_count)
20          FROM   missing_skills)
21 SELECT per_id,
22        ms_count
23 FROM   missing_skills
24        JOIN min_missing_ks
25          ON ms_count = min_missing_ks.min_ms_count;
```

Fix

- same as 7 except min_mssing_ks -> finds the min, for missing_skills.ms_count
- then we join them based on similar ms_count.

19

For a specified job code and a given small number k, make a "missing-k" list that lists the people's IDs and the number of missing skills for the people who miss only up to k skills in the ascending order of missing skills.

```
1
2  WITH skills_needed(ks_code)
3       AS (SELECT ks_code
4           FROM   required_skill
5           WHERE  job_code = 1),
6       missing_skills(per_id, ms_count)
7       AS (SELECT per_id,
8                  Count(ks_code)
9           FROM   person p,
10                 (SELECT ks_code
11                  FROM   skills_needed) sn
12          WHERE  sn.ks_code IN ((SELECT ks_code
13                                 FROM   skills_needed)
14                                MINUS
15                                (SELECT ks_code
16                                 FROM   has_skill
17                                 WHERE  per_id = p.per_id))
18          GROUP  BY per_id)
19 SELECT per_id,
20        ms_count
21 FROM   missing_skills
22 WHERE  ms_count <= 3 --k
23 ORDER  BY ms_count ASC;
```

Fix

- same as above but we select select only when ms count is under or equal to k

20

Given a job code and its corresponding missing-k list specified in Question 19. Find every skill that is needed by at least one person in the given missing-k list. List each skill code and the number of people who need it in the descending order of the people counts.

```
1
2  WITH skills_needed(ks_code)
3       AS (SELECT ks_code
4           FROM   required_skill
5           WHERE  job_code = '1'),
6       missing_skills(per_id, ms_count)
7       AS (SELECT per_id,
8                  Count(ks_code)
```

```
          FROM   person p,
                 (SELECT ks_code
                  FROM   skills_needed) sn
          WHERE  sn.ks_code IN ((SELECT ks_code
                                 FROM   skills_needed)
                                MINUS
                                (SELECT ks_code
                                 FROM   has_skill
                                 WHERE  per_id = p.per_id))
          GROUP  BY per_id),
     missing_people(per_id, ms_count)
     AS (SELECT per_id,
                ms_count
         FROM   missing_skills
         WHERE  ms_count <= 3)
SELECT ks_code,
       Count(per_id) AS mp_count
FROM   missing_people p,
       skills_needed
WHERE  skills_needed.ks_code IN (SELECT ks_code
                                 FROM   skills_needed
                                 MINUS
                                 SELECT ks_code
                                 FROM   has_skill
                                 WHERE  per_id = P.per_id)
GROUP  BY ks_code
ORDER  BY mp_count DESC;
```

Fix same as q17 and q19 put together.

21

In a local or national crisis, we need to find all the people who once held a job of the special job category identifier.

```

SELECT per_id
FROM   works
       NATURAL JOIN job
       NATURAL JOIN job_category
WHERE  cate_code = 1;
```

Fix - did not need to know unemployment status kinda stupid