

1.

List a company's workers by names.

```
1 SELECT per_id,  
2         per_name  
3 FROM    person  
4         NATURAL JOIN works  
5         NATURAL JOIN job  
6 WHERE   comp_id = '8'  
7         AND end_date IS NULL;
```

2

List a company's staff by salary in descending order.

```
1 SELECT per_name,  
2         pay_rate  
3 FROM    person  
4         NATURAL JOIN works  
5         NATURAL JOIN job  
6 WHERE   comp_id = '8'  
7         AND pay_type = 'salary'  
8 ORDER  BY pay_rate DESC;
```

3

List companies' labor cost (total salaries and wage rates by 1920 hours) in descending order.

```
1 SELECT comp_id,  
2         SUM(CASE  
3             WHEN pay_type = 'salary' THEN pay_rate  
4             ELSE pay_rate * 1920  
5         END) AS total_labor_cost  
6 FROM    job  
7         NATURAL JOIN works  
8 GROUP  BY comp_id  
9 ORDER  BY total_labor_cost DESC;
```

4

Find all the jobs a person is currently holding and worked in the past.

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

5

List a person's knowledge/skills in a readable format.

```
1 SELECT ks_title,  
2       ks_level,  
3       ks_description  
4 FROM   has_skill  
5       NATURAL JOIN knowledge_skill  
6 WHERE   per_id = 1;
```

6

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

```
1 (SELECT ks_code  
2 FROM   required_skill  
3       NATURAL JOIN works  
4 WHERE   per_id = 1)  
5 MINUS  
6 (SELECT ks_code  
7 FROM   has_skill  
8 WHERE   per_id = 1);
```

7

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

```
1
2 -- a job
3 SELECT ks_code,
4         ks_title,
5         ks_level,
6         ks_description
7 FROM    required_skill
8         NATURAL JOIN knowledge_skill
9 WHERE    job_code = 1; ;
10 -- a job category
11 SELECT ks_code,
12         ks_title,
13         ks_level,
14         ks_description
15 FROM    skill_set
16         NATURAL JOIN knowledge_skill
17 WHERE    cate_code = 1;
```

8

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

```
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);
```

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.

```
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));

```

10

Suppose the skill gap of a worker and the requirement of a desired job can be covered by one course. Find the “quickest” solution for this worker. Show the course, section information and the completion date.

```

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

```

```

19      AS (SELECT DISTINCT c_code,
20                complete_date
21      FROM    SECTION
22      NATURAL JOIN fulfilling_courses
23      WHERE   complete_date >= Trunc(SYSDATE))
24 SELECT c_code,
25        complete_date
26 FROM    fulfilling_section
27 WHERE   complete_date = (SELECT Min(complete_date)
28                        FROM    fulfilling_section);

```

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).

```

1  WITH missing_ks(ks)
2      AS ((SELECT ks_code
3            FROM    required_skill
4            WHERE   job_code = 1)
5      MINUS
6      (SELECT ks_code
7      FROM    has_skill
8      WHERE   per_id = 1)),
9      fulfilling_courses(c_code, c_title, retail_price)
10     AS (SELECT c_code,
11              c_title,
12              retail_price
13      FROM    course c
14      WHERE   NOT EXISTS ((SELECT *
15                          FROM    missing_ks)
16      MINUS
17      (SELECT ks_code
18      FROM    teaches_skill ts
19      WHERE   ts.c_code = c.c_code)))
20 SELECT c_code,
21        c_title,
22        retail_price
23 FROM    fulfilling_courses
24 WHERE   retail_price = (SELECT Min(retail_price)
25                        FROM    fulfilling_courses
26                        NATURAL JOIN SECTION);

```

12

If query #9 returns nothing, then find the course sets that their combination covers all the missing knowledge/ skills for a person to pursue a specific job. The considered course sets will not include more than three courses. If multiple course sets are found, list the course sets (with their course IDs) in the order of the ascending order of the course sets' total costs.

13

List all the job categories that a person is qualified for.

```
1
2 SELECT cate_code,
3        cate_title
4 FROM   job_category jc
5 WHERE  NOT EXISTS ((SELECT ks_code
6                     FROM   skill_set ss
7                     WHERE  jc.cate_code = ss.cate_code)
8             MINUS
9             (SELECT ks_code
10              FROM   has_skill
11              WHERE  per_id = 2));
```

14

Find the job with the highest pay rate for a person according to his/her skill qualification

```
1 WITH qualified_jobs
2   AS (SELECT j.job_code
3        FROM   job j
4        WHERE  NOT EXISTS ((SELECT ks_code
5                             FROM   required_skill rs
6                             WHERE  j.job_code = rs.job_code)
7             MINUS
8             (SELECT ks_code
9              FROM   has_skill
10              WHERE  per_id = 1))),
11 q_jobs_desc
```

```
12      AS (SELECT *
13           FROM    job
14                NATURAL JOIN qualified_jobs)
15 SELECT job_code,
16        pay_rate,
17        pay_type
18 FROM    q_jobs_desc
19 WHERE   pay_rate = (SELECT Max(CASE
20                        WHEN pay_type = 'salary' THEN pay_rate
21                        ELSE pay_rate * 1920
22                        END)
23                FROM    q_jobs_desc);
```

15

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

```
1 SELECT per_name,
2        email
3 FROM    person p
4 WHERE   NOT EXISTS ((SELECT ks_code
5                        FROM    required_skill
6                        WHERE   job_code = 1)
7                MINUS
8                (SELECT ks_code
9                  FROM    has_skill hs
10                 WHERE   hs.per_id = p.per_id));
```

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.

```
1
2 SELECT per_id,
3        per_name
4 FROM    person p
5 WHERE   1 = (SELECT Count(ks_code)
6              FROM    ((SELECT ks_code
7                        FROM    required_skill
```

```
8          WHERE job_code = 1)
9      MINUS
10     (SELECT ks_code
11      FROM has_skill hs
12      WHERE hs.per_id = p.per_id));
```

17

List the skillID and the number of people in the missing-one list for a given job code in the ascending order of the people counts.

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


18

Suppose there is a new job that has nobody qualified. List the persons who miss the least number of skills and report the “least number”.

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

19

For a specified job category 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;

```

20

Given a job category 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 skillID and the number of people who need it in the descending order of the people counts.

```

1 WITH skills_needed(ks_code)
2     AS (SELECT ks_code
3         FROM    required_skill
4         WHERE   job_code = '1'),
5     missing_skills(per_id, ms_count)
6     AS (SELECT per_id,
7             Count(ks_code)
8     FROM    person p,
9            (SELECT ks_code
10             FROM    skills_needed) sn
11     WHERE   sn.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     missing_people(per_id, ms_count)
19     AS (SELECT per_id,
20            ms_count
21         FROM missing_skills
22         WHERE ms_count <= 3)
23 SELECT ks_code,
24        Count(per_id) AS mp_count
25 FROM   missing_people p,
26        skills_needed
27 WHERE  skills_needed.ks_code IN (SELECT ks_code
28                                FROM   skills_needed
29                                MINUS
30                                SELECT ks_code
31                                FROM   has_skill
32                                WHERE  per_id = P.per_id)
33 GROUP BY ks_code
34 ORDER BY mp_count DESC;
```

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.

```
1 SELECT per_id
2 FROM works NATURAL JOIN job NATURAL JOIN job_category
3 where cate_code = 1;
```

22

Find all the unemployed people who once held a job of the given job identifier.

```
1 WITH unemployed(per_id)
2     AS ((SELECT per_id
3          FROM   person)
4         MINUS
5         (SELECT per_id
6          FROM   works
7          WHERE  end_date >= current_date))
8 SELECT per_id
9 FROM   unemployed
10 NATURAL JOIN works
```

```
11 WHERE job_code = 8;
```

23

Find out the biggest employer in terms of number of employees or the total amount of salaries and wages paid to employees.

```
1 WITH company_size(comp_id, employee_count)
2   AS (SELECT comp_id,
3             Count(*)
4        FROM   job
5             NATURAL JOIN works
6        GROUP BY comp_id)
7 SELECT comp_id employee_COUNT
8 FROM   company_size
9 WHERE  employee_count = (SELECT Max (employee_count)
10                        FROM   company_size);
```

24

Find out the job distribution among business sectors; find out the biggest sector in terms of number of employees or the total amount of salaries and wages paid to employees.

```
1 WITH sector_size(primary_sector, employee_count)
2   AS (SELECT primary_sector,
3             Count(*)
4        FROM   job
5             NATURAL JOIN works
6             NATURAL JOIN company
7        GROUP BY primary_sector)
8 SELECT primary_sector,
9        employee_count
10 FROM   sector_size
11 WHERE  employee_count = (SELECT Max (employee_count)
12                        FROM   sector_size);
```

- 25. Find out the ratio between the people whose earnings increase and those – whose earning decrease; find the average rate of earning improvement for the – workers in a specific business sector. – this does not work – did not do.

```
1  --WITH
2  --pay_rate_from_work AS (
3  --SELECT per_id, works.job_code, start_date, end_date, case
4  --                                when pay_type = '
        salary'
5  --                                then pay_rate
6  --                                else pay_rate*1920
        end
7  --
```

- 26. Find the leaf-node job categories that have the most openings due to – lack of qualified workers. If there are many opening jobs of a job category – but at the same time there are many qualified jobless people. Then training – cannot help fill up this type of job. What we want to find is such a job – category that has the largest difference between vacancies (the unfilled – jobs of this category) and the number of jobless people who are – qualified for the jobs of this category.
- 27. Find the courses that can help most jobless people find a job by – training them toward the jobs of this category that have the most openings – due to lack of qualified workers.
- 28. List all the courses, directly or indirectly required, that a person has – to take in order to be qualified for a job of the given category, according – to his/her skills possessed and courses taken. (required for graduate – students only)
- 4. Find all the jobs a person is currently holding and – worked in the past.