1.

List a company’s workers by names.

```sql
SELECT per_id,
       per_name
FROM   person
       NATURAL JOIN works
       NATURAL JOIN job
WHERE  comp_id = ’8’
       AND end_date IS NULL;
```

2

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

```sql
SELECT per_name,
       pay_rate
FROM   person
       NATURAL JOIN works
       NATURAL JOIN job
WHERE  comp_id = ’8’
       AND pay_type = ’salary’
ORDER  BY pay_rate DESC;
```

3

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

```sql
SELECT comp_id,
       SUM(CASE
             WHEN pay_type = ’salary’ THEN pay_rate
             ELSE pay_rate * 1920
           END) AS total_labor_cost
FROM   job
       NATURAL JOIN works
GROUP  BY comp_id
ORDER  BY total_labor_cost DESC;
```

4

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

```
SELECT job_code,
       start_date,
       end_date
FROM   works
       NATURAL JOIN job
WHERE  per_id = 1
ORDER  BY start_date DESC;
```

5

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

```
SELECT ks_title,
       ks_level,
       ks_description
FROM   has_skill
       NATURAL JOIN knowledge_skill
WHERE  per_id = 1;
```

6

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

```
(SELECT ks_code
 FROM   required_skill
        NATURAL JOIN works
 WHERE  per_id = 1)
MINUS
(SELECT ks_code
 FROM   has_skill
 WHERE  per_id = 1);
```

7

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

```

-- a job
SELECT ks_code,
       ks_title,
       ks_level,
       ks_description
FROM   required_skill
       NATURAL JOIN knowledge_skill
WHERE  job_code = 1; ;
-- a job category
SELECT ks_code,
       ks_title,
       ks_level,
       ks_description
FROM   skill_set
       NATURAL JOIN knowledge_skill
WHERE  cate_code = 1;
```

8

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

```
(SELECT ks_code,
        ks_title
 FROM   required_skill
        NATURAL JOIN knowledge_skill
 WHERE  job_code = 2)
MINUS
(SELECT ks_code,
        ks_title
 FROM   has_skill
        NATURAL JOIN knowledge_skill
 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.

```

```

```
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))
SELECT c_code,
       c_title
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));
```

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.

```
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)
     AS (SELECT c_code
         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))),
     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);
```

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 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

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.

```sql

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));
```

14

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

```sql
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);
```

15

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

```
SELECT per_name,
       email
FROM   person p
WHERE  NOT EXISTS ((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));
```

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)));
```

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.

```
WITH skills_needed(ks_code)
     AS (SELECT ks_code
         FROM   required_skill
         WHERE  job_code = '1'),
     missing_skills(per_id, ms_count)
     AS (SELECT per_id,
                Count(ks_code)
         FROM   person p,
                skills_needed
         WHERE  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)
SELECT ks_code,
       Count(per_id) AS total_ms_count
FROM   missing_skills ms,
       skills_needed
WHERE  ks_code IN ((SELECT ks_code
                    FROM   skills_needed)
                   MINUS
                   (SELECT ks_code
                    FROM   has_skill
                    WHERE  per_id = ms.per_id))
       AND ms_count = 1
GROUP  BY ks_code
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”.

```
WITH skills_needed(ks_code)
     AS (SELECT ks_code
         FROM   required_skill
         WHERE  job_code = 1),
     missing_skills(per_id, ms_count)
     AS (SELECT per_id,
                Count(ks_code)
         FROM   person p,
                skills_needed sn
         WHERE  sn.ks_code IN ((SELECT ks_code
                                FROM   required_skill)
                               MINUS
                               (SELECT ks_code
                                FROM   has_skill
                                WHERE  per_id = p.per_id))
         GROUP  BY per_id),
     min_missing_ks(min_ms_count)
     AS (SELECT Min(ms_count)
         FROM   missing_skills)
SELECT per_id,
       ms_count
FROM   missing_skills
       JOIN min_missing_ks
         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.

```

WITH skills_needed(ks_code)
     AS (SELECT ks_code
         FROM   required_skill
         WHERE  job_code = 1),
     missing_skills(per_id, ms_count)
     AS (SELECT per_id,
                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)
SELECT per_id,
       ms_count
FROM   missing_skills
WHERE  ms_count <= 3 --k
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.

```
WITH skills_needed(ks_code)
     AS (SELECT ks_code
         FROM   required_skill
         WHERE  job_code = '1'),
     missing_skills(per_id, ms_count)
     AS (SELECT per_id,
                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;
```

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;
```

22

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

```
WITH unemployed(per_id)
     AS ((SELECT per_id
          FROM   person)
         MINUS
         (SELECT per_id
          FROM   works
          WHERE  end_date >= current_date))
SELECT per_id
FROM   unemployed
       NATURAL JOIN works
```

```
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.

```
WITH company_size(comp_id, employee_count)
     AS (SELECT comp_id,
                Count(*)
         FROM   job
                NATURAL JOIN works
         GROUP  BY comp_id)
SELECT comp_id employee_COUNT
FROM   company_size
WHERE  employee_count = (SELECT Max (employee_count)
                         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.

```
WITH sector_size(primary_sector, employee_count)
     AS (SELECT primary_sector,
                Count(*)
         FROM   job
                NATURAL JOIN works
                NATURAL JOIN company
         GROUP  BY primary_sector)
SELECT primary_sector,
       employee_count
FROM   sector_size
WHERE  employee_count = (SELECT Max (employee_count)
                         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.

```
--WITH
--pay_rate_from_work AS (
--SELECT per_id, works.job_code, start_date, end_date, case
--                                                    when pay_type = ’salary’
--                                                    then pay_rate
--                                                    else pay_rate*1920 end
--
```

– 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.