**SQL test 2** 9.05.2023

For each of the tasks below, write the appropriate SQL commands. We expect the solutions to be in the form of a file containing the **content** of SQL commands, not the output obtained from running them.

**Any syntactically incorrect queries will not be checked** - check your solution e.g. using `\i file.sql` You can send the file multiple times, only the latest version will be checked.

Load the `offers.sql` file into your database. Send your solutions through the form at `https://dbserv.stud.ii/`. Do it as often as possible! All data on your computer is erased after a restart. In case of any problems, be sure to contact the course instructor before restarting your computer.

The first line of the solution should have the following format: *`-- group-name-surname`*, where `group` are the initials of the instructor leading your group (`jmi`/`mabi`/`plg`/`pwi`), e.g. `pwi-Jan-Kowalski`. The required format for the entire solution file is as follows:

```
-- group-name-surname
-- Task 1
<query>

-- Task 2
<query>
...
```

**Task 1** (3 points) For each of the required skills (`skill.name`), let's count in how many different job positions (`offer.title`) and in how many different offers it appears, and how important the given skill is for each of the offers (`skill.value`).

Write a query that returns the name of the skill, the number of different positions and the number of different offers in which it appears, as well as the minimum, maximum and rounded to the nearest natural number average value of the `value` field.

Sort the result in descending order by the number of positions and, in the second order, by the number of offers.

*The reference query returns 1456 tuples. The first four rows of the reference solution (censored):*

```
name                          |  job_title | offers | min | max | avg
------------------------------+------------+--------+-----+-----+-----
(???)                         |        271 |   1289 |   1 |   5 |   3
Python                        |        224 |   1365 |   1 |   5 |   3
Java                          |        221 |   1601 |   1 |   5 |   4
English                       |        219 |   1235 |   3 |   5 |   4
```

**Rozwiązanie**

```
SELECT
    s.name,
    count(distinct o.title) AS stanowiska,
    count(o.id) AS oferty,
    min(value),
    max(value),
    round(avg(value)) AS avg
FROM offer o JOIN
     skill s ON (o.id=s.offer_id)
GROUP BY s.name
ORDER BY 2 DESC, 3 DESC;
```

**Task 2** (2 points (+1 bonus))

Write a query that returns the offered positions (`offer.title`) and the total number of different requirements (`skill.name`) for a given position among all employers. Only include positions with more than 20 different requirements. Sort the result in descending order by the number of requirements, and then alphabetically by position.

You can earn a bonus point for adding a column that contains an array with 4 sample skills from the offers for the mentioned position.

*The sample query returns 13 tuples. The first three rows of the sample solution:*

```
       title       | count |                          example_skill
-------------------+-------+-------------------------------------------------
DevOps Engineer    |    57 | {"Amazon Web Services",Ansible,API,ArgoCD}
Java Developer     |    41 | {Agile,"Apache Kafka",AWS,Cloud}
Data Engineer      |    39 | {AI/ML,Airflow,"Apache Spark",AWS}
```

**Rozwiązanie**

```
SELECT
    o.title,
    count(distinct s.name),
    (array_agg(distinct s.name))[1:4] AS example_skill
FROM
    offer o JOIN
    skill s ON (s.offer_id=o.id)
GROUP BY
    title
HAVING
    count(distinct s.name)>20 ORDER BY 2 DESC, 1;
```

**Task 3** (3 points)

In the unlikely event of failing this course, you may want to know which companies you can apply to without any knowledge of databases or SQL.

Check it out and write a query that returns the names of companies that offer a position for which none of the `skill.name` contains the word `'SQL'` or `'database'`. Ignore case sensitivity. Sort the results alphabetically.

*The sample query returns 1019 rows.*

**Rozwiązanie**

```
SELECT DISTINCT c.name
FROM company c JOIN
     offer o ON c.id=o.company_id
WHERE
   o.id NOT IN (
         SELECT offer_id
         FROM skill
         WHERE
            name ILIKE '%sql%' OR
            name ILIKE '%database%'
         )
ORDER BY 1;
```

**Task 4** (3 points)

Write a query that returns the names of companies that do not require knowledge of databases or SQL for **any** position offered by the company (use the criterion from the previous task, i.e. `skill.name` does not contain the word `'SQL'` or `'database'`, ignoring case). Sort the results alphabetically.

*The sample query returns 799 tuples.*

**Rozwiązanie**

```
SELECT DISTINCT c.name
FROM company c
WHERE c.id NOT IN
(SELECT cp.id
FROM company cp JOIN
     offer o ON cp.id=o.company_id JOIN
     skill s ON o.id=s.offer_id
         WHERE
            s.name ILIKE '%sql%' OR
            s.name ILIKE '%database%'
```

```
        )
ORDER BY 1;
```

**Task 5** (3 points) For the 10 cities with the highest number of job offers (according to `company_branch`), write a query that returns the number of job offers where one of the requirements (`skill.name`) concerns the `'Snowflake'` system.

*The sample query returns* ***of course*** *exactly 10 tuples.*

*Note that the database contains a redundancy:* `offer` *can be joined with* `company_branch` *using the* `company_id` *attribute and the* `company` *table, or directly using* `company_branch_id`*. In the current state of the database, it does not matter which option you choose, but it should be noted that such a database schema can easily lead to data inconsistencies.*

### Solution 1

```
WITH cities AS (
       SELECT city
       FROM company_branch cb JOIN
            offer o ON cb.id=o.company_branch_id
       GROUP BY city
       ORDER BY count(o.id) DESC
       LIMIT 10),
     snowflakeSkill AS (
       SELECT name,
              offer_id
       FROM skill
       WHERE name = 'Snowflake')
SELECT cb.city,
       count(s.name)
FROM cities c JOIN
     company_branch cb on c.city=cb.city JOIN
     offer o ON o.company_branch_id = cb.id LEFT JOIN
     snowflakeSkill s ON (o.id=s.offer_id)
GROUP BY cb.city
ORDER BY 2 DESC, 1;
```

Note that the following (very) simple solution is not much short of correct. The only problem is how to sort by `COUNT(DISTINCT offer_id)`?

### Solution 2

```
SELECT city, COUNT(DISTINCT offer_id)
FROM company_branch
```

```
    LEFT JOIN offer ON company_branch_id = company_branch.id
    LEFT JOIN skill ON (offer.id = offer_id AND skill.name ILIKE '%Snowflake%')
GROUP BY city
ORDER BY COUNT(offer_id) DESC
LIMIT 10;
```