SQL Test No. 5 13.06.2024

For each of the following tasks, write the appropriate SQL commands. We expect a solution in the form of a file containing the **contents** of the SQL commands, not the answer. Syntactically incorrect queries will not be checked — verify your solution, e.g., using `\i file.sql` ! You can send the file multiple times, only the latest version will be checked. Remember, before performing a database modification operation, you can issue the command `BEGIN;`, and then `ROLLBACK;` (if you want to roll back all changes) or `COMMIT;` (if you want to save them).

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

For this test you can earn a maximum of 14 points.

Task 1 (4 points) List offer IDs and the associated skills that are the most important for a given offer according to the `value` field. Collect the relevant skills in the form of a string containing their names separated by the `' & '` separator. You can skip offers that are not associated with any skill in `skill`. Sort the result by offer ID in ascending order.

The result should include tuples (possibly with a different permutation of names in the second component)

```
9  | Kubernetes & Azure & Docker
10 | SQL
11 | Python & SQL
```

PostgreSQL documentation: 9.21 Aggregate Functions

```
SELECT offer_id, STRING_AGG(name, ' & ')
FROM skill JOIN (
     SELECT offer_id, MAX(value) AS value FROM skill GROUP BY offer_id
  ) AS foo USING (offer_id, value)
GROUP BY offer_id;

-- or if we consider joins for filtering as ugly

SELECT offer_id, STRING_AGG(name, ' & ')
FROM skill
WHERE (offer_id, value) IN (
    SELECT offer_id, MAX(value) FROM skill GROUP BY offer_id
  )
GROUP BY offer_id;
```

Task 2 (3 points) List all rows (*details*) from the `employment_details` table, supplementing each with the following information: the average minimum salary calculated from details with the same currency, and the length of the interval (in days) in which details with the same employment type were published. Sort the results by type (`type`), then by currency, and finally – by offer ID.

Offers with the possibility of employment on a mandate contract (`mandate_contract`) *have been published over the shortest period, while offers on permanent contract* (`permanent`) *– over the longest period, although in each case the period is two weeks and a few hours. The average minimum salary for offers in pounds is 3041 (and almost 21 pence). The length of the interval in days can be calculated by subtracting the publication dates of the last and the first offers, see:* 9.9. Date/Time Functions and Operators - Table 9.32. Date/Time Operators

```
SELECT employment_details.*,
 AVG(salary_from) OVER (PARTITION BY currency),
 MAX(published_at) OVER (PARTITION BY type)
    - MIN(published_at) OVER (PARTITION BY type)
 FROM employment_details JOIN offer ON id = offer_id
 ORDER BY type, currency, offer_id;
```

Task 3 (3 points) Add a `changed` column of boolean type with a default value of `false` to the `company` table. Correct the company names by removing all spaces and tabs from their beginnings and ends. Whenever you correct a company name in any row, set the value of `changed` to `true`.

PostgreSQL documentation: 9.4 String Functions and Operators.

A string literal containing escape characters should be preceded by the character `E`*, i.e., a string containing a single tab character is* `E'\t'`.

```
ALTER TABLE company ADD COLUMN changed boolean DEFAULT false;

UPDATE company
SET name = TRIM(BOTH E' \t' FROM name), changed = true
WHERE name != TRIM(BOTH E' \t' FROM name);
```

After executing the command, the following query should return 1072.

```
SELECT count(*) FROM company WHERE NOT changed;
```

Task 4 (4 points)

For *each* company, provide the number of remote (`remote`) and office (`office`) job offers (according to `workplace_type`), considering only offers published on days of the week other than Monday, and only those companies for which the number of stationary offers is no greater than the number of remote offers. In the result, provide the company name with all spaces and tabs removed (also inside the string) – use the `TRANSLATE` function.

Sort the results alphabetically by company name, then by the second column of the result in descending order, and finally – by the third column of the result in descending order.

PostgreSQL documentation:

- 4.2.7. Aggregate Expressions,
- 9.4 String Functions and Operators.,
- 9.9. Date/Time Functions and Operators.

```
SELECT translate(c.name, E' \t', ''),
    COUNT (*) FILTER (WHERE o.workplace_type = 'remote'),
    COUNT (*) FILTER (WHERE o.workplace_type = 'office')
FROM company c LEFT JOIN
     offer o ON (c.id = o.company_id AND EXTRACT (DOW FROM published_at) != 1)
GROUP BY c.id
HAVING
  COUNT (*) FILTER (WHERE o.workplace_type = 'remote') >=
  COUNT (*) FILTER (WHERE o.workplace_type = 'office')
```

```
ORDER BY 1, 2 DESC, 3 DESC
-- LIMIT 3 OFFSET 9 -- list tuples 10, 11, 12
;
```

Part of the expected result:

```
  AbplanalpSp.zo.o.                |     0 |     0
  Acaisoft                         |     5 |     1
  Accenture                        |    42 |     3
```