04a_grouping

Grouping and Aggregates

Contacts DB

```
SET search_path TO contacts;
```

1. Count the number of rows in the `call` table

```
SELECT COUNT(*)
FROM call;
```

2. Count the number of calls for each phone number in the `call` table, and rename the `count` column to something more appropriate.

```
SELECT phone, COUNT(*) AS n_calls
FROM call
GROUP BY phone;
```

3. Count the number of calls for each phone number in the `call` table, and order the results by the largest number of calls first.

```
SELECT phone, COUNT(*) AS n_calls
FROM call
GROUP BY phone
ORDER BY 2 DESC;
```

4. Count the number of calls for each phone number in the `call` table, and keep only the phone numbers with more than 1 call.
   - use `HAVING` with a condition
   - `HAVING` is similar to `WHERE`, but it is executed after the `GROUP BY`, while the `WHERE` is executed after the `FROM`, but before the `GROUP BY`
   - even tough `SELECT` is written first, it is actually executed after `HAVING`, but before `ORDER BY` (if present)

```
SELECT phone, COUNT(*) AS n_calls
FROM call
GROUP BY phone
HAVING COUNT(*) > 1;
```

5. We can group not only the rows of one table, but also any result table from the `FROM` part of the query
    - this query computes the number of calls for each contact (not for each phone number in `call`, but each `contact_id` in `contact`)
    - note that contacts without any calls are listed with a count of 0 because we are using a left outer join, so contacts not matching anything in `call`, in other words, contacts without any associated calls, will be kept in the results

```
SELECT contact.contact_id, COUNT(call_id) AS n_calls
FROM contact
         LEFT OUTER JOIN call
                         ON contact.contact_id = call.contact_id
GROUP BY contact.contact_id
ORDER BY n_calls DESC;
```