Read the following relational model:

* CUSTOMERS (id, lastname, firstname, address, zipcode, phonenumber, recommend\_id\*, registerdate)
* BOOKINGS (fac\_id\*, cust\_id\*, start\_datetime, nhours)
* FACILITIES (id, name, cust\_cost, guest\_cost, purchase\_cost, maintenance\_cost)

Meaning of the fields:

CUSTOMERS:

* id: Customer’s id.
* lastname: Customer’s last name.
* firstname, Customer’s first name.
* address: Customer’s address.
* zipcode: Customer’s zip code.
* phonenumber: Customer’s contact telephone number.
* recommended\_id: Customer who recommended the service (if any).
* registerdate: Date when the customer joined the service.

BOOKINGS:

* fac\_id: Facility id of the booking.
* cust\_id: Customer who made the booking.
* start\_datetime: Start date/time of the booking.
* nhours: Number of hours that the facility were booked.

FACILITIES:

* id: Id of the facility.
* name: Name of the facility.
* cust\_cost: Daily cost for customers. Data type must be money.
* guest\_cost: Daily cost for guests. Data type must be money.
* purchase\_cost: Purchase cost (to the enterprise) of the facility. Data type must be money.
* maintenance\_cost: Monthly maintenance cost of the facility. Data type must be money.

0.- What’s a booking? What’s a locator in a booking? Do we have a locator in this relational model?

1.- Draw the entity relationship model of the relational model exposed above.

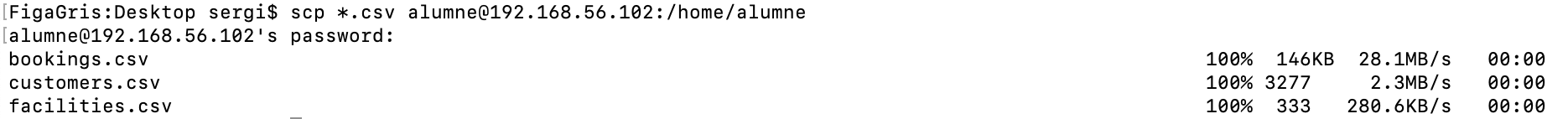
2.- Create a new database (name ‘bookings’) inside your PostgreSQL server.

3.- Implement the relational model exposed above using PostgreSQL (inside the database ‘bookings’). It’s up to you choosing the appropriate data types, integrity constraint, etc. Before beginning, check the difference between varchar and text data types here: <https://stackoverflow.com/questions/4848964/postgresql-difference-between-text-and-varchar-character-varying>

4.- Import the data of the following files inside the tables of the database:

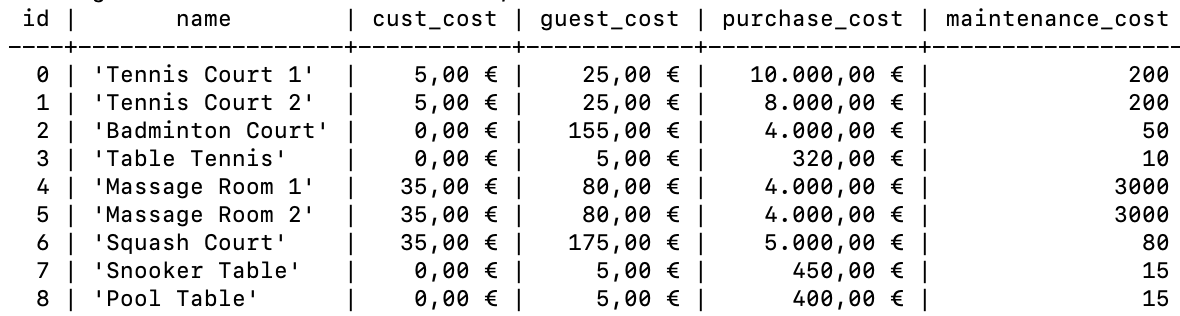
* [bookings.csv](https://drive.google.com/file/d/12aAZybql3_fQU0i6J9YPhCmIuk_d-oJB/view?usp=sharing)
* [customers.csv](https://drive.google.com/file/d/15zJOJab3g7dGknUhch9QmHbsW-LPnxHz/view?usp=sharing)
* [facilities.csv](https://drive.google.com/file/d/1fgujWGYaB3nGIWGrELdgLjhUJOWaQ4vU/view?usp=sharing)

Clue: How to copy from Linux to your virtual machine.



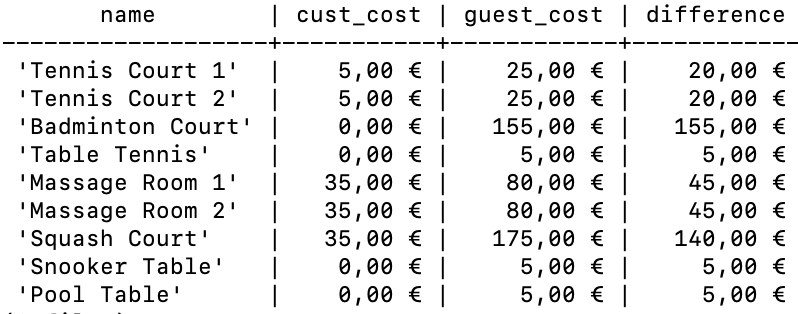
Before importing search on the Internet how to deal with nulls and quotes...

5.- Show data of the facilities available.



select \* from facilities;

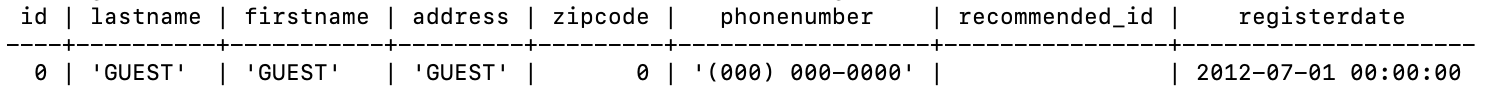
6.- Show names of facilities and cost to customers and to guest. Show the difference between costs.



select name, cust\_cost, guest\_cost, (guest\_cost - cust\_cost) as difference

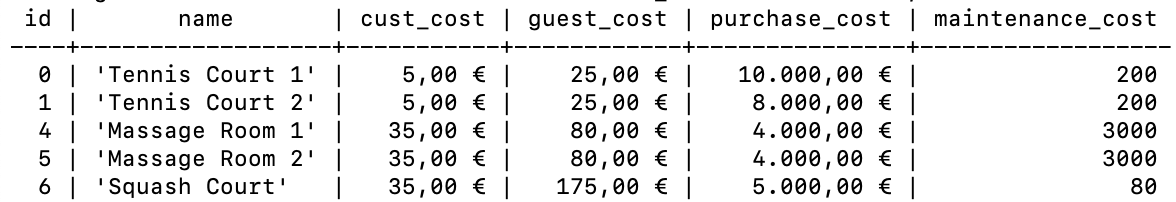
from facilities;

7.- Show if there is a user (=row inside customers) for guests. Search what’s ‘ilike’ and use it.



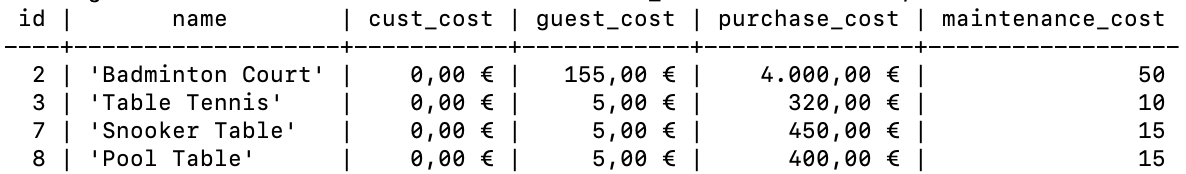
select \* from customers where first\_name ilike 'guest';

8.- Show the facilities that have a cost to customers. Before doing the exercise, read this [link](http://www.postgresqltutorial.com/postgresql-cast/).



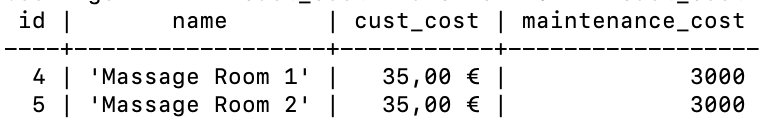
select \* from facilities where cust\_cost!=0::money;

9.- Show the free facilities to customers.



select \* from facilities where cust\_cost=0::money;

10.- Show the facilities that have a cost to customers, and that the cost per day is less than 1/50 of the monthly maintenance cost. Show the fields id, name, cust\_cost and maintenance\_cost.

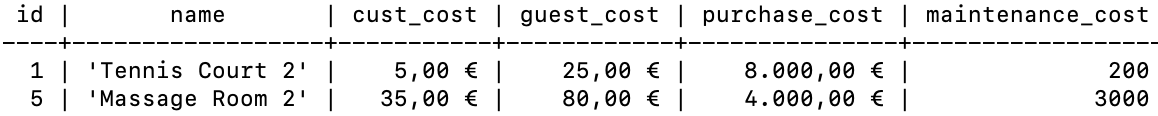


select \* from facilities

where cust\_cost!=0::money

and cust\_cost < maintenance\_cost/50;

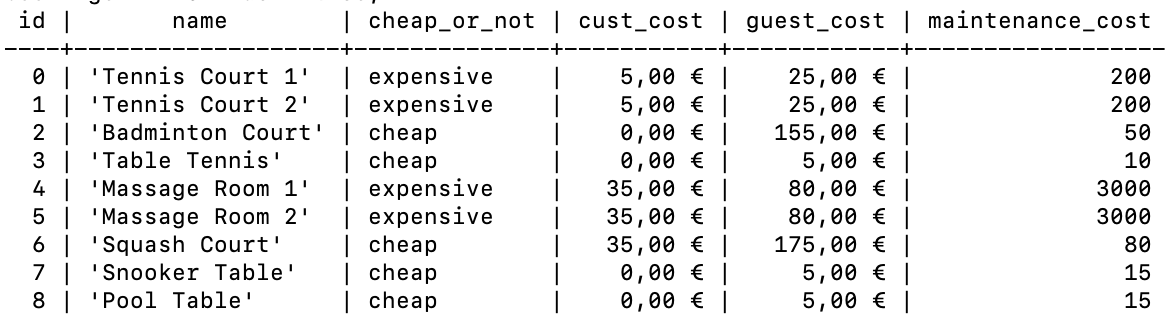
11.- Show facilities with ID 1 and 5. Do it without using the OR operator.



select \* from facilities

where id in (1, 5);

12.- Show facilities labelling them as 'cheap' or 'expensive' (name of the new column ‘cheap\_or\_not’) depending on if their monthly maintenance cost is more than $150. Show the fields id, name, cheap\_or\_not, cust\_cost, guest\_cost and maintenance\_cost. Clue: Use CASE.



select id, name, (CASE maintenance\_cost > 150::money

WHEN true THEN 'expensive'

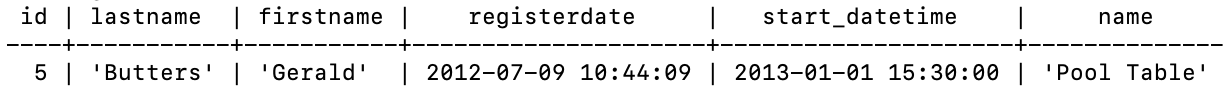
ELSE 'cheap'

END) AS cheap\_or\_not,

cust\_cost, guest\_cost, maintenance\_cost

from facilities;

13.- Show customers who booked for the first time after the start of October 2012? Return the fields customer’s id, lastname, firstname, registerdate, start\_datetime of the bookings, and facility name.



select C.id, C.last\_name, C.first\_name, C.registerdate, B.start\_datetime, F.name

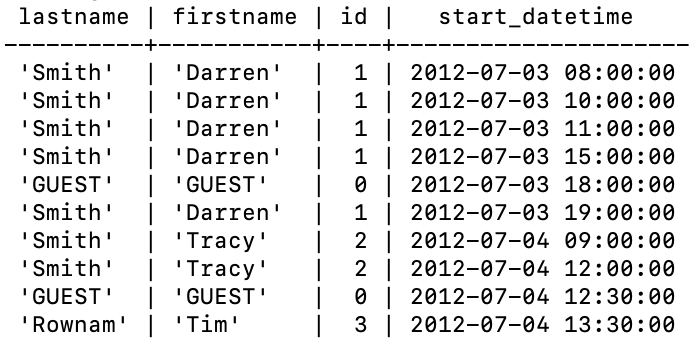
from customers C, bookings B, facilities F

where C.id =B.cust\_id

AND B.fac\_id=F.id

and B.start\_datetime > '2012-10-01';

14.- Show an ordered list of the first 10 customers who booked our facilities, order by lastname, firstname, and booking start time. It’s mandatory to use the LIMIT clause.



select C.last\_name, C.first\_name, C.id, B.start\_datetime

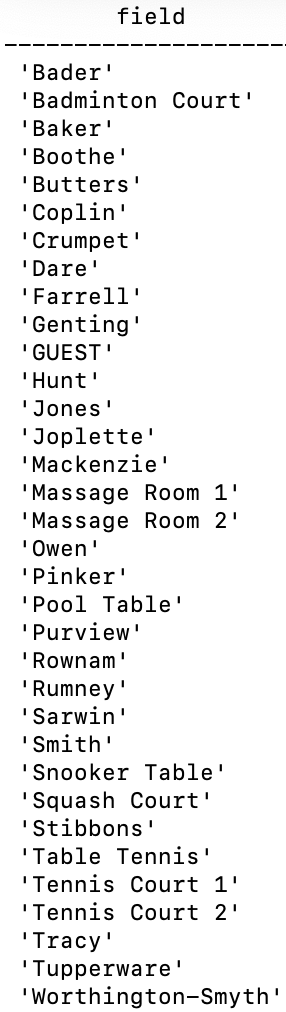
from customers C, bookings B

where C.id=B.cust\_id

order by B.start\_datetime, C.last\_name, C.first\_name

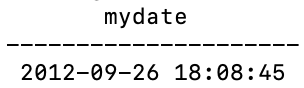
limit 10;

15.- Show a list of customer’s surnames and facility names (together). Order the results for that unique column.



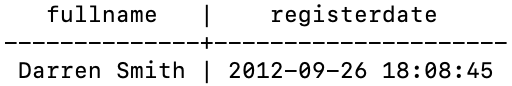
(select last\_name as field from customers) UNION (select name from facilities) order by field;

16.- Show the last date of the customer’s relation.



select max(registerdate) from customers;

17.- Show the full name (and date) of the customers who signed up in the date that you found in exercise 16. Obviously, you can not copy/paste that concrete date…

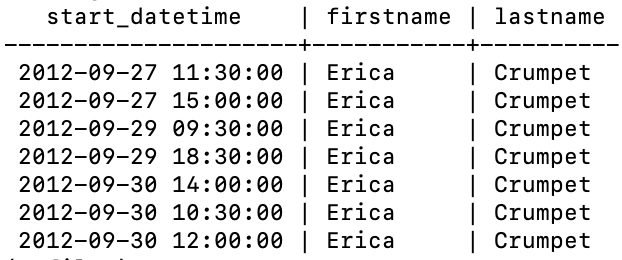


select concat(first\_name, ' ', last\_name) as full\_name, registerdate

from customers

where registerdate >= all (select registerdate from customers);

18.- Show start times for bookings by customers named 'E. Crumpet'.



select B.start\_datetime, C.first\_name, C.last\_name

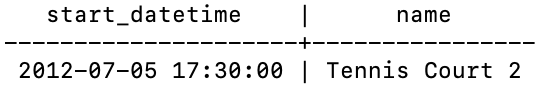
from bookings B, customers C

where B.cust\_id = C.id

and C.first\_name like 'E%'

and C.last\_name = 'Crumpet';

19.- Show start times for bookings for tennis courts, for the date '5/7/2012'? Return a list of start time and facility name pairings, ordered by the time. It’s mandatory to use in the WHERE clause the function date\_trunc.



select B.start\_datetime, F.name

from facilities F, bookings B

where F.id=B.fac\_id

and F.name ilike 'tennis%'

and date\_trunc('day', start\_datetime)='2012-07-05'::date

order by B.start\_datetime;

20.- Show all customers who have recommended another customer. Ensure that there are no duplicates in the list, and that results are ordered by (lastname, firstname).



select first\_name, last\_name

from customers

where id in (select recommended\_id from customers)

order by last\_name, first\_name;

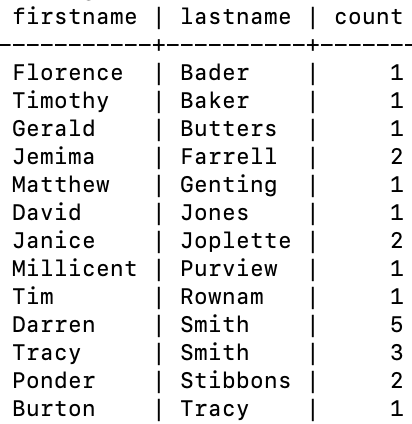
select distinct C.first\_name, C.last\_name

from customers C, customers R

where C.id = R.recommended\_id

order by last\_name, first\_name;

21. Do the same query than before but showing how many customers where recommended by them.



select C.first\_name, C.last\_name, count(R.recommended\_id)

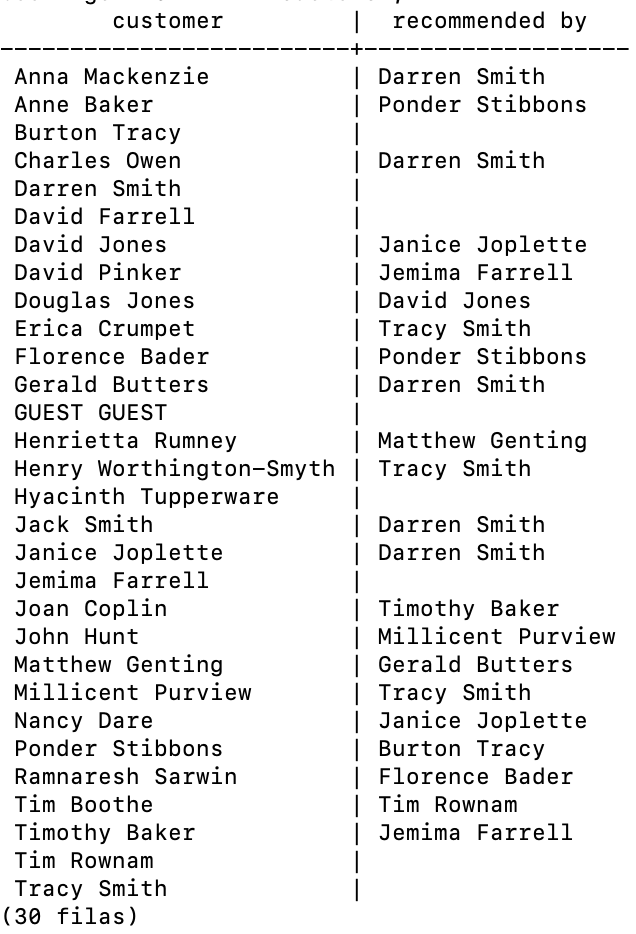
from customers C, customers R

where C.id = R.recommended\_id

group by C.first\_name, C.last\_name

order by last\_name, first\_name;

22.- Show customers’ full name, including the individual who recommended them (if any). Ensure that results are ordered by full name.



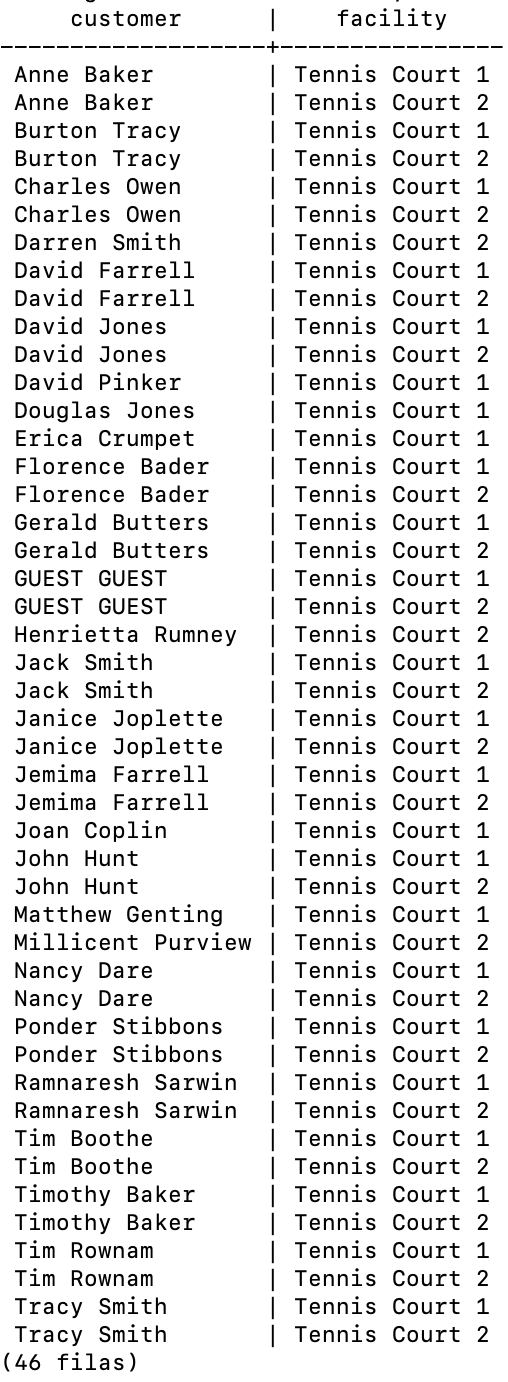
select distinct concat(C.first\_name, ' ', C.last\_name) as customer, concat(R.first\_name, ' ', R.last\_name) as "recommended by"

from customers C

left outer join customers R on C.recommended\_id = R.id

order by customer;

23.- Show full names of the customers who have used a tennis court. Don’t repeat data.



select distinct concat(C.first\_name, ' ', C.last\_name) as customer, F.name

from customers C, bookings B, facilities F

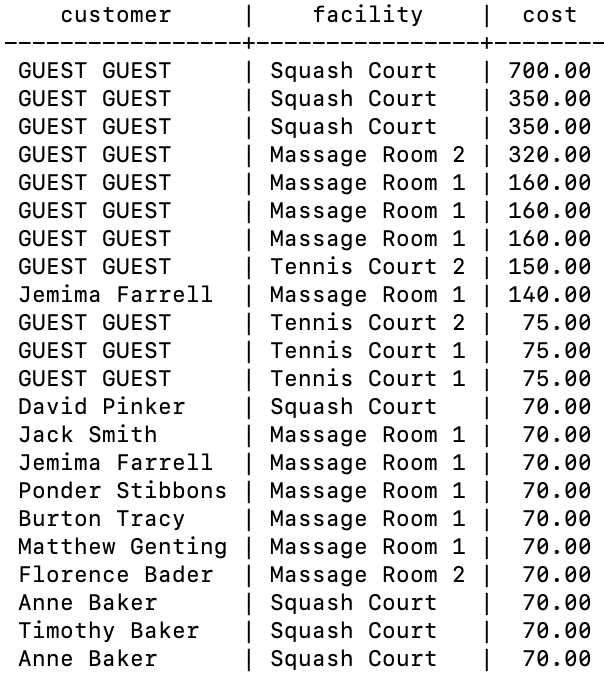
where C.id = B.cust\_id

and B.fac\_id=F.id

and F.name ilike 'tennis%'

order by customer;

24.- Show bookings on the day of 2012-09-14 which will cost the customer (or guest) more than $30. Remember that guests have different costs to customers (the listed costs are per hour 'slot'), and the guest user is always ID 0. Include in your output the name of the facility, the name of the customer formatted as a single column, and the cost. Order by descending cost, and do not use any subqueries (clue: you must use case).



select concat(C.first\_name, ' ', C.last\_name) as customer, F.name as facility,

CASE C.id='0'

WHEN true THEN (F.guest\_cost\*B.nhours)

ELSE (F.cust\_cost\*B.nhours)

END AS cost

from customers C, bookings B, facilities F

where C.id = B.cust\_id

and B.fac\_id=F.id

and date\_trunc('day', B.start\_datetime)='2012-09-14'::date

and (CASE C.id='0'

WHEN true THEN (F.guest\_cost\*B.nhours>30::money)

ELSE (F.cust\_cost\*B.nhours>30::money) END)

order by cost desc;

25.- Show a list of all customers, including the individuals who recommended them (if any), using a subquery. Ensure that there are no duplicates in the list, and that each firstname + lastname pairing is formatted as a column and ordered.



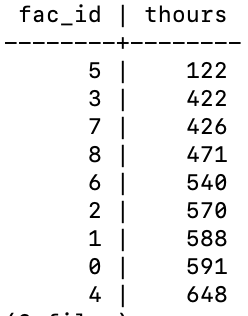
select concat(C.first\_name, ' ', C.last\_name) as customer, concat(R.first\_name, ' ', R.last\_name) as "recommended by"

from customers C

left outer join customers R on C.recommended\_id = R.id

order by customer;

26.- Produce a list of the total number of hours booked per facility in the month of September 2012.



select fac\_id, sum(nhours) as thours

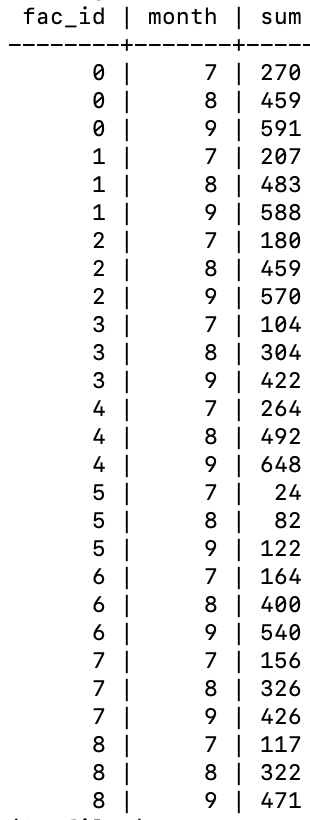
from bookings

where date\_trunc('month', start\_datetime)='2012-09-01'::date

group by fac\_id

order by thours;

27.- Produce a list of the total number of hours booked per facility per month in the year of 2012. Order by facility id and month.



select fac\_id, extract(month from start\_datetime) as month, sum(nhours) as thours

from bookings

where date\_trunc('year', start\_datetime)='2012-01-01'::date

group by fac\_id, month

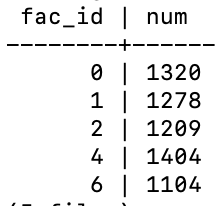
order by fac\_id;

28.- Find the total number of customers who have made at least one booking.



select count(distinct cust\_id) from bookings;

29.- SHOWf facilities with more than 1000 HOURS booked. Sort the results using facility id.



select fac\_id, sum(nhours) as num

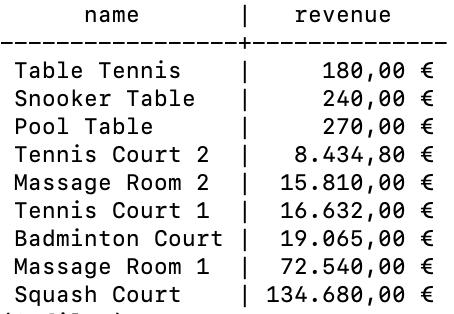
from bookings

group by fac\_id

having sum(nhours)>1000

order by fac\_id;

30.- Show facilities along with their total revenue. Remember that there's a different cost for guests and customers! Sort results by revenue.



select F.name,

sum(CASE B.cust\_id=0

WHEN true THEN B.nhours\*F.guest\_cost

ELSE B.nhours\*F.cust\_cost END) as revenue

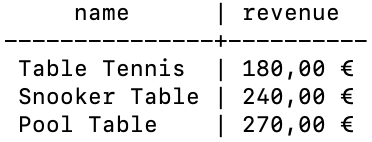
from facilities F, bookings B

where F.id=B.fac\_id

group by F.name

order by revenue;

31.- Produce a list of facilities with a total revenue less than 1000. Remember that there's a different cost for guests and customers! Sort results by revenue.



select F.name,

sum(CASE B.cust\_id=0

WHEN true THEN B.nhours\*F.guest\_cost

ELSE B.nhours\*F.cust\_cost END) as revenue

from facilities F, bookings B

where F.id=B.fac\_id

group by F.name

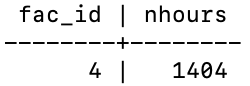
having sum(CASE B.cust\_id=0

WHEN true THEN B.nhours\*F.guest\_cost

ELSE B.nhours\*F.cust\_cost END)<1000::money

order by revenue;

32.- Output the facility id that has the highest number of hours booked.



select fac\_id, sum(nhours) as num

from bookings

group by fac\_id

having sum(nhours)>=all(select sum(nhours) from bookings group by fac\_id);

33.- Check the following query:

SELECT fac\_id, date\_part('month', start\_datetime), SUM(nhours)

FROM bookings

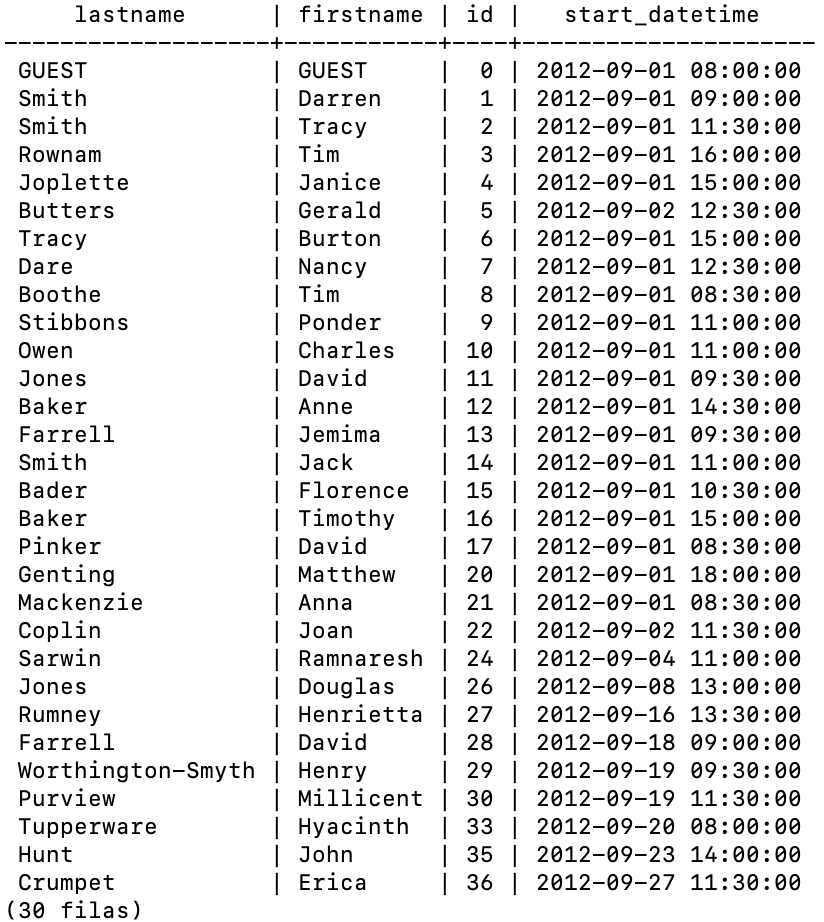
WHERE date\_part('year', start\_datetime) = 2012

GROUP BY ROLLUP(fac\_id, date\_part('month', start\_datetime))

ORDER BY fac\_id, date\_part('month', start\_datetime);

Explain the clause GROUP BY ROLLUP and the meaning of the query.

34.- Show a list of each customer name, id, and their first booking after September 1st 2012. Order by customer id.



select C.last\_name, C.first\_name, C.id, min(B.start\_datetime)

from customers C, bookings B

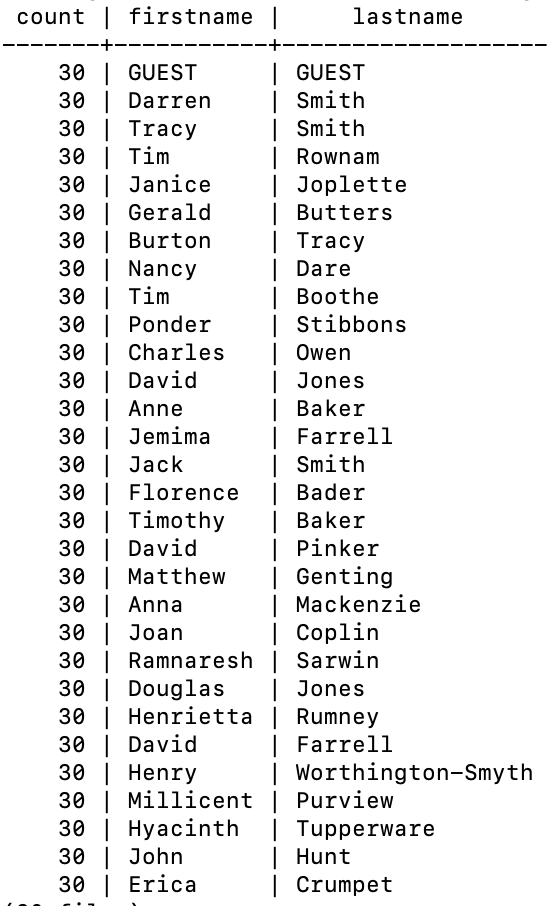
where C.id = B.cust\_id

and B.start\_datetime > '2012-09-01'

group by C.last\_name, C.first\_name, C.id

order by C.id;

35.- (Window function) Show a list of customer names, with each row containing the total customer count. Order by register date.

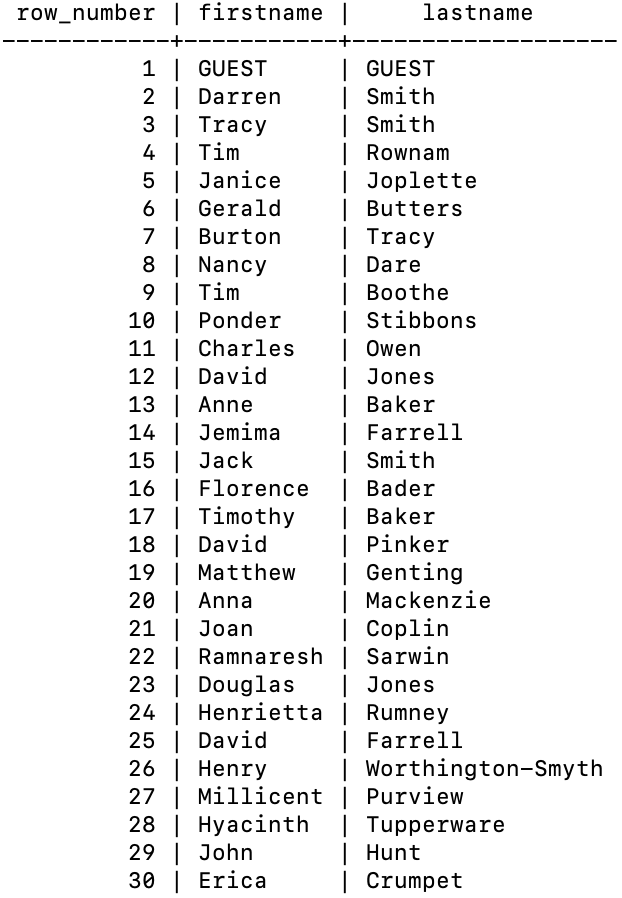


select count(\*) over (), first\_name, last\_name

from customers

order by registerdate;

36.- (Window function) Show an increasing numbered list of customers, ordered by their date of registering. Remember that customers’ ids are not guaranteed to be sequential.

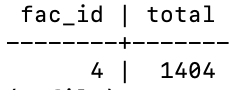


select row\_number() over(), first\_name, last\_name

from customers

order by registerdate;

37.- (Windows function) DANGER: Show the facility id that has the highest number of hours booked. All tieing results must get an output.



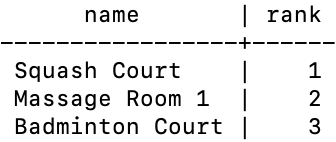
select T.fac\_id, max(T.total)

from (select fac\_id, sum(nhours) over (partition by fac\_id) as total from bookings) T

group by T.fac\_id

having max(T.total)>=all (select sum(nhours) over (partition by fac\_id) as total from bookings);

38.- (Window function) DANGER: Show a list of the top three revenue generating facilities (including ties (=”empates”)). Sort the results by rank and facility name.



select name, rank() over (order by revenue desc)

from (

select F.name,

sum(CASE B.cust\_id=0

WHEN true THEN B.nhours\*F.guest\_cost

ELSE B.nhours\*F.cust\_cost END) as revenue

from facilities F, bookings B

where F.id=B.fac\_id

group by F.name) R

limit 3;

39.- We need to add it into the facilities table new records with the following values:

* id: 9, name: 'Spa', cust\_cost: 20, guest\_cost: 30, purchase\_cost: 100000, maintenance\_cost: 800
* id: 9, name: 'Spa 2', cust\_cost: 20, guest\_cost: 30, purchase\_cost: 100000, maintenance\_cost: 800.
* id: 10, name: 'Squash Court 2', cust\_cost: 3.5, guest\_cost: 17.5, purchase\_cost: 5000, maintenance\_cost: 80.

insert into facilities (id, name, cust\_cost, guest\_cost, purchase\_cost, maintenance\_cost)

values

(9, 'Spa', 20, 30, 100000, 800),

(10, 'Spa2', 20, 30, 100000, 800),

(11, 'Squash Court 2', 3.5, 17.5, 5000, 80);

40.- Let's add, again, a new ‘Spa 3’ to the facilities table. But this time, though, we want to automatically generate the value for the next id, rather than specifying it as a constant (clue: that’s an insert with a subquery). Use the following values for everything else:

* name: 'Spa 3', cust\_cost: 20, guest\_cost: 30, purchase\_cost: 100000, maintenance\_cost: 800.

insert into facilities

select (max(id)+1), 'Spa3', 20, 30, 100000, 800 from facilities;

41.- We made a mistake when entering the data for the second tennis court. The initial purchase cost was 10000 rather than 8000: you need to alter the data to fix the error.

insert into facilities

select (max(id)+1), 'Spa3', 20, 30, 100000, 800 from facilities;

update facilities

set purchase\_cost=10000 where id =1;

42.- We want to increase the price of the tennis courts for both customers and guests. Update the costs to be 6 for customers, and 30 for guests. Use only a single sentence.

update facilities

set cust\_cost=6::money, guest\_cost=30::money

where name ilike 'tennis court%';

update facilities

set cust\_cost=6::money, guest\_cost=30::money

where id =0 or id=1;

43.- We want to alter the price of the second tennis court so that it costs 10% more than the first one. Try to do this without using constant values for the prices, so that we can reuse the statement if we want to. Use only a single sentence.

update facilities

set

(cust\_cost, guest\_cost) = (select cust\_cost\*1.1, guest\_cost\*1.1 from facilities where id=0)

where id=1;

44.- Delete all bookings.

delete from bookings;

45.- We want to remove customer 37, who has never made a booking, from our database.

delete from customers where id=37;

46.- How can we make that more general, to delete all customers who have never made a booking? Clue: Delete with subquery. To test the sentence inter customer 37 again.

insert into customers values (37, 'Smith', 'Darren', '3 Funktown Denzington Boston', '66796', '(822) 577-3541', NULL, '2012-09-26 18:08:45');

deletre from customers

where id not in (select cust\_id from bookings);