P11. ACME BOOKING ENGINE

In simple words, a booking engine is an application on hotel websites and social media pages to capture and process direct online reservations.

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.

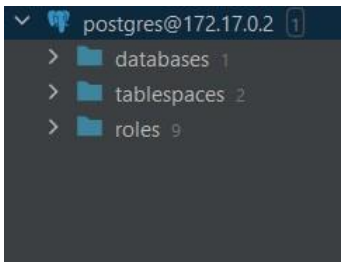
PART A

Describe the postgres **server** dockerization and the connection to a **client**. It can be psql, Pgadmin4, Beekeeper Studio or Datagrip.

First, we create a new Docker and open port 80 to then be able to access it from PgAdmin.

```
aperellop@aperellop-ubuntu:~$ sudo docker run --name postgresSGBD -e POSTGRES_PASSWORD=mysecretpassword
-d postgres
9ebd9c8bdc1d1f38ef9aa2332a72cf26538f88a4e282207ef0484f6b39390106
aperellop@aperellop-ubuntu:~$ sudo docker pull dpage/pgadmin4
Using default tag: latest
latest: Pulling from dpage/pgadmin4
Digest: sha256:38617bc122e547dcfe3adaba52143f583343928b3700ada6feb9dcf6d13e0ca6
Status: Image is up to date for dpage/pgadmin4:latest
docker.io/dpage/pgadmin4:latest
aperellop@aperellop-ubuntu:~$ sudo docker run -p 80:80 \
>     -e 'PGADMIN_DEFAULT_EMAIL=user@domain.com' \
>     -e 'PGADMIN_DEFAULT_PASSWORD=SuperSecret' \
>     -d dpage/pgadmin4
698f83bd194b9cb6213542a726b33b38a22211086820d0d81a849a311724945a
```

We create a server in PgAdmin and link it to the DataGrip.

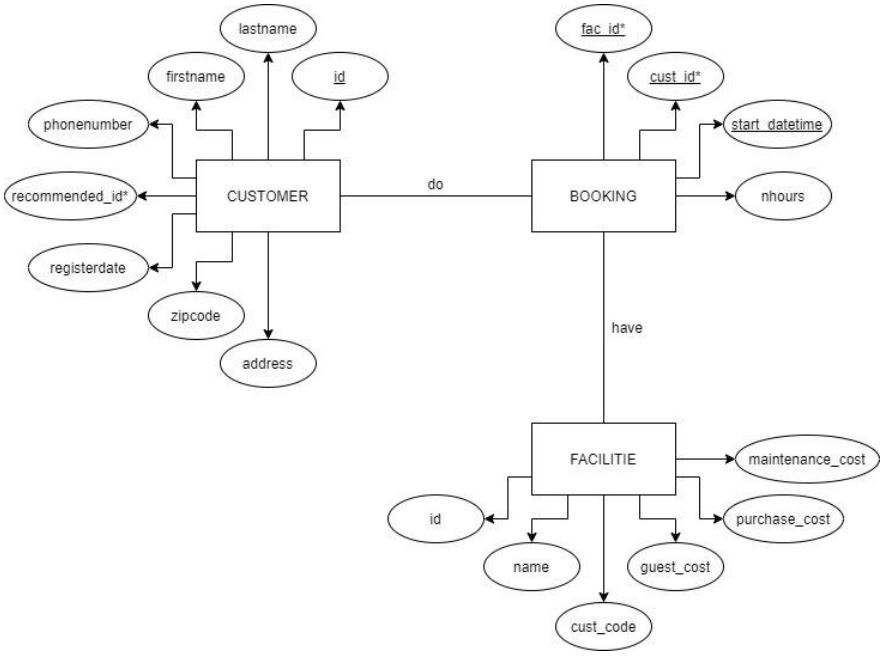


PART B

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

A booking is the reservation code that user have and contains all the information of this. The locator is the fac_id. We don't have any 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 for a new user with name 'customer01' ('customer01' must be the owner of 'bookings').

```
create user customer01;
create database bookings with owner customer01;
```

3.- Implement the relational model exposed above using PostgreSQL inside the database 'bookings'. It's up to you to choose 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

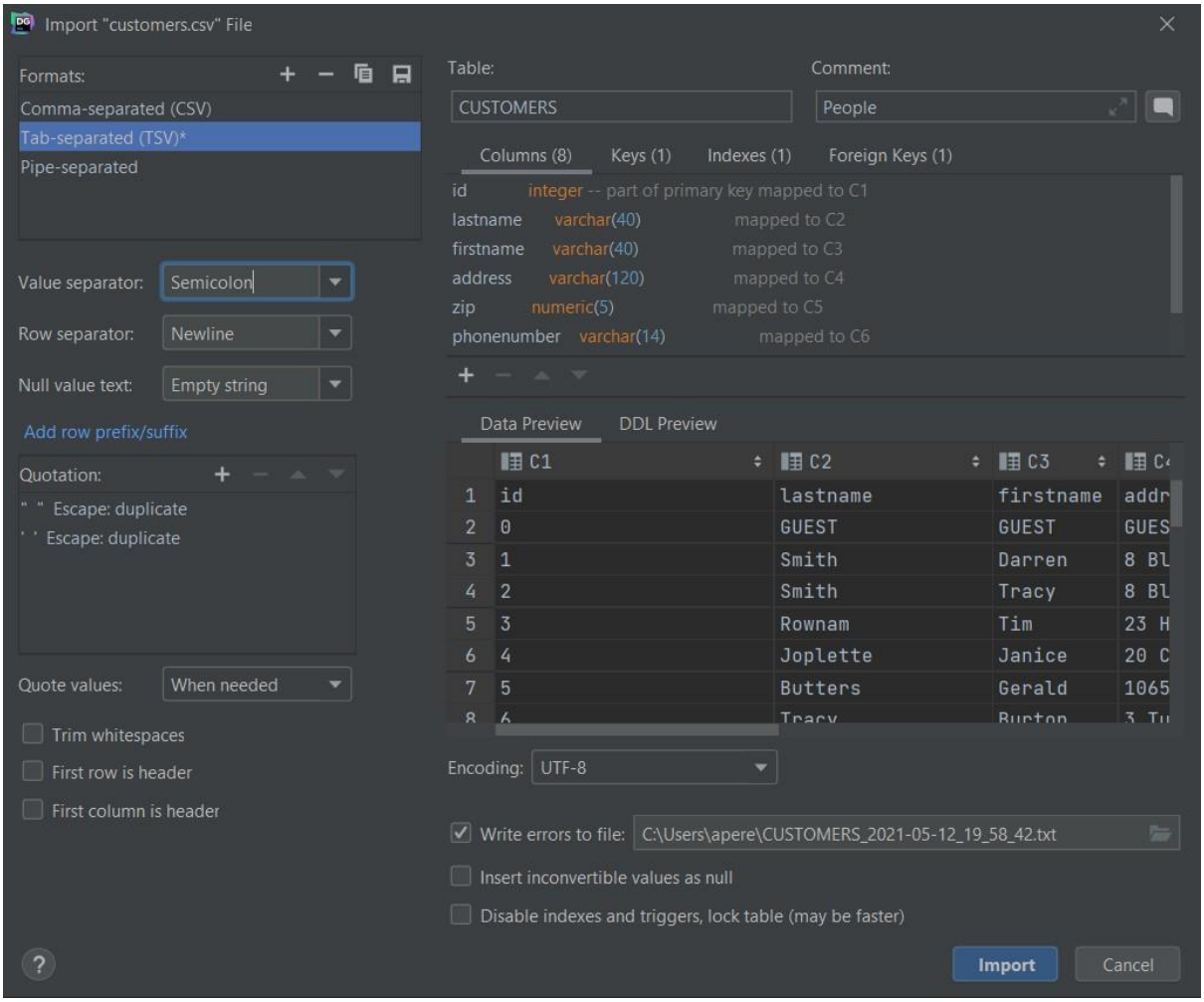
```
create schema bookings;
comment on schema bookings is 'Actual Bookings Schema';
Alter schema bookings owner to customer01;
```

```sql
create table "bookings"."FACILITIES"
(
    id               integer     not null
        constraint "FACILITIES_pkey"
            primary key,
    name             varchar(50) not null,
    cust_cost        money       not null,
    guest_cost       money       not null,
    purchase_cost    money       not null,
    maintenance_cost money       not null
);

comment on table  "bookings"."FACILITIES" is 'What you can reserve';

alter table  "bookings"."FACILITIES"
    owner to customer01;

create table  "bookings"."CUSTOMERS"
(
    id             integer      not null
        constraint "CUSTOMERS_pkey"
            primary key,
    lastname       varchar(40)  not null,
    firstname      varchar(40)  not null,
    address        varchar(120) not null,
    zip            numeric(5)   not null,
    phonenumber    varchar(14)  not null,
    recommended_id integer
        constraint "CUSTOMERS_recommended_id_fkey"
```

```sql
                references  "bookings"."CUSTOMERS"
                on update cascade,
        registerdate   timestamp    not null
);

comment on table  "bookings"."CUSTOMERS" is 'People';

alter table  "bookings"."CUSTOMERS"
    owner to customer01;

create table  "bookings"."BOOKINGS"
(
    fac_id         integer   not null
        constraint "BOOKINGS_fac_id_fkey"
            references  "bookings"."FACILITIES"
            on update cascade,
    cust_id        integer   not null
        constraint "BOOKINGS_cust_id_fkey"
            references  "bookings"."CUSTOMERS"
            on update cascade,
    start_datetime timestamp not null,
    nhours         integer   not null,
    constraint bookings_pk
        primary key (fac_id, cust_id, start_datetime)
);

comment on table  "bookings"."BOOKINGS" is 'Reservations';

alter table  "bookings"."BOOKINGS"
```

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

- bookings.csv
- customers.csv
- facilities.csv



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

```
FigaGris:Desktop sergi$ scp *.csv alumne@192.168.56.102:/home/alumne
alumne@192.168.56.102's password:
bookings.csv                                          100%  146KB  28.1MB/s   00:00
customers.csv                                         100% 3277     2.3MB/s   00:00
facilities.csv                                        100%  333   280.6KB/s   00:00
```

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

5.- Show data of the facilities available.

```
SELECT *
FROM bookings."FACILITIES";
```

id	name	cust_cost	guest_cost	purchase_cost	maintenance_cost
0	Tennis Court 1	$5.00	$25.00	$10,000.00	$200.00
1	Tennis Court 2	$5.00	$25.00	$8,000.00	$200.00
2	Badminton Court	$0.00	$155.00	$4,000.00	$50.00
3	Table Tennis	$0.00	$5.00	$320.00	$10.00
4	Massage Room 1	$35.00	$80.00	$4,000.00	$3,000.00
5	Massage Room 2	$35.00	$80.00	$4,000.00	$3,000.00
6	Squash Court	$35.00	$175.00	$5,000.00	$80.00
7	Snooker Table	$0.00	$5.00	$450.00	$15.00
8	Pool Table	$0.00	$5.00	$400.00	$15.00

(9 rows)

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

```
SELECT name, cust_cost, guest_cost, (guest_cost - cust_cost)
                AS difference
FROM bookings."FACILITIES"
```

name	cust_cost	guest_cost	difference
Tennis Court 1	$5.00	$25.00	$20.00
Tennis Court 2	$5.00	$25.00	$20.00
Badminton Court	$0.00	$155.00	$155.00
Table Tennis	$0.00	$5.00	$5.00
Massage Room 1	$35.00	$80.00	$45.00
Massage Room 2	$35.00	$80.00	$45.00
Squash Court	$35.00	$175.00	$140.00
Snooker Table	$0.00	$5.00	$5.00
Pool Table	$0.00	$5.00	$5.00

(9 rows)

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

```
SELECT *
FROM bookings."CUSTOMERS"
WHERE firstname ILIKE 'guest'
  AND lastname ILIKE 'guest'
  AND address ILIKE 'guest'
```

id	lastname	firstname	address	zipcode	phonenumber	recommended_id	registerdate
0	GUEST	GUEST	GUEST	0	(000) 000-0000		2012-07-01 00:00:00

(1 row)

8.- Show the facilities that have a cost to customers. Before doing the exercise, read this link.

```
SELECT *
FROM bookings."FACILITIES"
WHERE CAST (cust_cost AS numeric) > 0;
```

```
 id |       name       | cust_cost | guest_cost | purchase_cost | maintenance_cost
----+------------------+-----------+------------+---------------+------------------
  0 | Tennis Court 1 |     $5.00 |     $25.00 |    $10,000.00 |          $200.00
  1 | Tennis Court 2 |     $5.00 |     $25.00 |     $8,000.00 |          $200.00
  4 | Massage Room 1 |    $35.00 |     $80.00 |     $4,000.00 |        $3,000.00
  5 | Massage Room 2 |    $35.00 |     $80.00 |     $4,000.00 |        $3,000.00
  6 | Squash Court   |    $35.00 |    $175.00 |     $5,000.00 |           $80.00
(5 rows)
```

9.- Show the free facilities to customers.

```
SELECT *
FROM bookings."FACILITIES"
WHERE CAST (cust_cost AS numeric) = 0;
```

```
 id |       name       | cust_cost | guest_cost | purchase_cost | maintenance_cost
----+------------------+-----------+------------+---------------+------------------
  2 | Badminton Court |     $0.00 |    $155.00 |     $4,000.00 |           $50.00
  3 | Table Tennis    |     $0.00 |      $5.00 |       $320.00 |           $10.00
  7 | Snooker Table   |     $0.00 |      $5.00 |       $450.00 |           $15.00
  8 | Pool Table      |     $0.00 |      $5.00 |       $400.00 |           $15.00
(4 rows)
```

10.- Show the facilities that have a cost to customers, and that the cost to customers is less than 1/50 of the monthly maintenance cost. Show the fields id, name, cust_cost and maintenance_cost.

```
SELECT *
FROM bookings."FACILITIES"
WHERE CAST (cust_cost AS numeric) > 0
  AND CAST (cust_cost AS numeric) < (CAST (maintenance_cost AS numeric) /
50 );
```

```
 id |       name       | cust_cost | maintenance_cost
----+------------------+-----------+------------------
  4 | Massage Room 1 |    $35.00 |        $3,000.00
  5 | Massage Room 2 |    $35.00 |        $3,000.00
(2 rows)
```

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

```
SELECT *
FROM bookings."FACILITIES"
WHERE id = 1
UNION
SELECT *
FROM bookings."FACILITIES"
WHERE id = 5;
```

```
 id |      name      | cust_cost | guest_cost | purchase_cost | maintenance_cost
----+----------------+-----------+------------+---------------+------------------
  1 | Tennis Court 2 |     $5.00 |     $25.00 |     $8,000.00 |          $200.00
  5 | Massage Room 2 |    $35.00 |     $80.00 |     $4,000.00 |        $3,000.00
(2 rows)
```

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 name,
       CASE
            WHEN cast(maintenance_cost as numeric) > 150
                THEN
                    'expensive'
                ELSE
                    'cheap'
                END AS cheap_or_not, cust_cost, guest_cost,
maintenance_cost
                    FROM bookings."FACILITIES";
```

```
 id |      name       | cheap_or_not | cust_cost | guest_cost | maintenance_cost
----+-----------------+--------------+-----------+------------+------------------
  0 | Tennis Court 1  | expensive    |     $5.00 |     $25.00 |          $200.00
  1 | Tennis Court 2  | expensive    |     $5.00 |     $25.00 |          $200.00
  2 | Badminton Court | cheap        |     $0.00 |    $155.00 |           $50.00
  3 | Table Tennis    | cheap        |     $0.00 |      $5.00 |           $10.00
  4 | Massage Room 1  | expensive    |    $35.00 |     $80.00 |        $3,000.00
  5 | Massage Room 2  | expensive    |    $35.00 |     $80.00 |        $3,000.00
  6 | Squash Court    | cheap        |    $35.00 |    $175.00 |           $80.00
  7 | Snooker Table   | cheap        |     $0.00 |      $5.00 |           $15.00
  8 | Pool Table      | cheap        |     $0.00 |      $5.00 |           $15.00
(9 rows)
```

13.- Show customers who booked 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.lastname,c.firstname,to_char(c.registerdate, 'YYYY-MM-DD
HH:MI:SS'),
       to_char(b.start_datetime, 'YYYY-MM-DD HH:MI:SS'), f.name
FROM bookings."CUSTOMERS" AS c, bookings."BOOKINGS" AS b,
bookings."FACILITIES" AS f
WHERE date(start_datetime) > '2012-10-01'
  AND b.cust_id=c.id
  AND b.fac_id=f.id;
```

```
 id | lastname | firstname |     registerdate     |    start_datetime    |     name
----+----------+-----------+----------------------+----------------------+------------
  5 | Butters  | Gerald    | 2012-07-09 10:44:09 | 2013-01-01 15:30:00 | Pool Table
(1 row)
```

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

```
SELECT c.lastname, c.firstname, c.id, b.start_datetime
FROM bookings."BOOKINGS" as b, bookings."CUSTOMERS" as c
WHERE b.cust_id=c.id
ORDER BY b.start_datetime LIMIT 10;
```

```
 lastname | firstname | id |   start_datetime
----------+-----------+----+--------------------
 Smith    | Darren    |  1 | 2012-07-03 08:00:00
 Smith    | Darren    |  1 | 2012-07-03 10:00:00
 Smith    | Darren    |  1 | 2012-07-03 11:00:00
 Smith    | Darren    |  1 | 2012-07-03 15:00:00
 GUEST    | GUEST     |  0 | 2012-07-03 18:00:00
 Smith    | Darren    |  1 | 2012-07-03 19:00:00
 Smith    | Tracy     |  2 | 2012-07-04 09:00:00
 Smith    | Tracy     |  2 | 2012-07-04 12:00:00
 GUEST    | GUEST     |  0 | 2012-07-04 12:30:00
 Rownam   | Tim       |  3 | 2012-07-04 13:30:00
(10 rows)
```

15.- Show a list of customer's surnames and facility names (together). Order the results for that unique column (clue: union operator).

```
SELECT DISTINCT lastname AS field
FROM bookings."CUSTOMERS"
UNION
SELECT name AS field
FROM bookings."FACILITIES"
ORDER BY field;
```

```
       field
------------------
 Bader
 Badminton Court
 Baker
 Boothe
 Butters
 Coplin
 Crumpet
 Dare
 Farrell
 Genting
 GUEST
 Hunt
 Jones
 Joplette
 Mackenzie
 Massage Room 1
 Massage Room 2
 Owen
 Pinker
 Pool Table
 Purview
 Rownam
 Rumney
 Sarwin
 Smith
 Snooker Table
 Squash Court
 Stibbons
 Table Tennis
 Tennis Court 1
 Tennis Court 2
 Tracy
 Tupperware
 Worthington-Smyth
(34 rows)
```

16.- Show the last date of the customer's relation (field customers.registerdate).

```
SELECT to_char(registerdate, 'YYYY-MM-DD HH24:MI:SS') AS mydate
FROM bookings."CUSTOMERS"
    ORDER BY registerdate DESC LIMIT 1
```

```
       mydate
---------------------
 2012-09-26 18:08:45
(1 row)
```

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…
Clue: subquery.

```
SELECT concat_ws(' ', c.firstname, c.lastname), to_char(c.registerdate,
'YYYY-MM-DD HH24:MI:SS')
FROM bookings."CUSTOMERS" AS c,
        (SELECT registerdate AS mydate
        FROM bookings."CUSTOMERS"
        ORDER BY registerdate DESC LIMIT 1) AS b
WHERE c.registerdate = b.mydate;
```

```
   fullname    |     registerdate
--------------+---------------------
 Darren Smith | 2012-09-26 18:08:45
(1 row)
```

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

```
SELECT to_char(b.start_datetime, 'YYYY-MM-DD HH24:MI:SS'), c.firstname,
c.lastname
FROM bookings."CUSTOMERS" AS c, bookings."BOOKINGS" AS b
WHERE c.firstname LIKE 'E%' AND c.lastname LIKE 'Crumpet' AND
b.cust_id=c.id;
```

```
   start_datetime     | firstname | lastname
---------------------+-----------+----------
 2012-09-27 11:30:00 | Erica     | Crumpet
 2012-09-27 15:00:00 | Erica     | Crumpet
 2012-09-29 09:30:00 | Erica     | Crumpet
 2012-09-29 18:30:00 | Erica     | Crumpet
 2012-09-30 14:00:00 | Erica     | Crumpet
 2012-09-30 10:30:00 | Erica     | Crumpet
 2012-09-30 12:00:00 | Erica     | Crumpet
(7 rows)
```

19.- Show start times for bookings for tennis courts on '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 to_char(b.start_datetime, 'YYYY-MM-DD HH24:MI:SS'), f.name
FROM bookings."BOOKINGS" AS b, bookings."FACILITIES" AS f
WHERE date(b.start_datetime) = '2012-07-05'
  AND f.name LIKE 'Tennis%'
  AND b.fac_id=f.id;
```

```
   start_datetime     |       name
----------------------+----------------
 2012-07-05 17:30:00 | Tennis Court 2
(1 row)
```

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 and firstname.

```
SELECT c.firstname, c.lastname
FROM bookings."CUSTOMERS" AS c
WHERE c.id IN (SELECT recommended_id
                 FROM bookings."CUSTOMERS")
                 ORDER BY lastname;
```

```
 firstname | lastname
-----------+----------
 Florence  | Bader
 Timothy   | Baker
 Gerald    | Butters
 Jemima    | Farrell
 Matthew   | Genting
 David     | Jones
 Janice    | Joplette
 Millicent | Purview
 Tim       | Rownam
 Darren    | Smith
 Tracy     | Smith
 Ponder    | Stibbons
 Burton    | Tracy
(13 rows)
```

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

```
SELECT c.firstname, c.lastname, COUNT(c.firstname)
FROM bookings."CUSTOMERS" AS c, (SELECT recommended_id AS rid
                                 FROM bookings."CUSTOMERS") AS x
                                 WHERE c.id = x.rid
                                 GROUP BY c.firstname, c.lastname
                                 ORDER BY c.lastname;
```

```
 firstname | lastname | count
-----------+----------+-------
 Florence  | Bader    |     1
 Timothy   | Baker    |     1
 Gerald    | Butters  |     1
 Jemima    | Farrell  |     2
 Matthew   | Genting  |     1
 David     | Jones    |     1
 Janice    | Joplette |     2
 Millicent | Purview  |     1
 Tim       | Rownam   |     1
 Darren    | Smith    |     5
 Tracy     | Smith    |     3
 Ponder    | Stibbons |     2
 Burton    | Tracy    |     1
(13 rows)
```

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

```
SELECT c.id, CONCAT_WS(' ', c.firstname, c.lastname) AS customer,
concat_ws(' ', l.firstname, l.lastname) as recommend_by
FROM bookings."CUSTOMERS" AS c,
     (SELECT DISTINCT c.id AS rid, c.firstname, c.lastname
    FROM bookings."CUSTOMERS" AS c, (select recommended_id AS rid from
bookings."CUSTOMERS") AS x
    WHERE c.id = x.rid) AS l
WHERE c.recommended_id=l.rid
UNION
SELECT c.id, CONCAT_WS(' ', c.firstname, c.lastname) AS customer, '' AS
recommend_by
FROM bookings."CUSTOMERS" AS c
WHERE recommended_id IS NULL
ORDER BY customer;
```

id	customer	recommended by
21	Anna Mackenzie	Darren Smith
12	Anne Baker	Ponder Stibbons
6	Burton Tracy	
10	Charles Owen	Darren Smith
37	Darren Smith	
1	Darren Smith	
28	David Farrell	
11	David Jones	Janice Joplette
17	David Pinker	Jemima Farrell
26	Douglas Jones	David Jones
36	Erica Crumpet	Tracy Smith
15	Florence Bader	Ponder Stibbons
5	Gerald Butters	Darren Smith
0	GUEST GUEST	
27	Henrietta Rumney	Matthew Genting
29	Henry Worthington-Smyth	Tracy Smith
33	Hyacinth Tupperware	
14	Jack Smith	Darren Smith
4	Janice Joplette	Darren Smith
13	Jemima Farrell	
22	Joan Coplin	Timothy Baker
35	John Hunt	Millicent Purview
20	Matthew Genting	Gerald Butters
30	Millicent Purview	Tracy Smith
7	Nancy Dare	Janice Joplette
9	Ponder Stibbons	Burton Tracy
24	Ramnaresh Sarwin	Florence Bader
8	Tim Boothe	Tim Rownam
16	Timothy Baker	Jemima Farrell
3	Tim Rownam	
2	Tracy Smith	

(31 rows)

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

```
SELECT DISTINCT CONCAT_WS(' ', c.firstname, c.lastname) AS customer, f.name
AS facility
FROM bookings."CUSTOMERS" AS c, bookings."FACILITIES" AS f,
bookings."BOOKINGS" AS b
WHERE f.name LIKE 'Tennis%'
  AND f.id=b.fac_id
  AND b.cust_id=c.id;
```

customer	facility
Anne Baker	Tennis Court 1
Anne Baker	Tennis Court 2
Burton Tracy	Tennis Court 1
Burton Tracy	Tennis Court 2
Charles Owen	Tennis Court 1
Charles Owen	Tennis Court 2
Darren Smith	Tennis Court 2
David Farrell	Tennis Court 1
David Farrell	Tennis Court 2
David Jones	Tennis Court 1
David Jones	Tennis Court 2
David Pinker	Tennis Court 1
Douglas Jones	Tennis Court 1
Erica Crumpet	Tennis Court 1
Florence Bader	Tennis Court 1
Florence Bader	Tennis Court 2
Gerald Butters	Tennis Court 1
Gerald Butters	Tennis Court 2
GUEST GUEST	Tennis Court 1
GUEST GUEST	Tennis Court 2
Henrietta Rumney	Tennis Court 2
Jack Smith	Tennis Court 1
Jack Smith	Tennis Court 2
Janice Joplette	Tennis Court 1
Janice Joplette	Tennis Court 2
Jemima Farrell	Tennis Court 1
Jemima Farrell	Tennis Court 2
Joan Coplin	Tennis Court 1
John Hunt	Tennis Court 1
John Hunt	Tennis Court 2
Matthew Genting	Tennis Court 1
Millicent Purview	Tennis Court 2
Nancy Dare	Tennis Court 1
Nancy Dare	Tennis Court 2
Ponder Stibbons	Tennis Court 1
Ponder Stibbons	Tennis Court 2
Ramnaresh Sarwin	Tennis Court 1
Ramnaresh Sarwin	Tennis Court 2
Tim Boothe	Tennis Court 1
Tim Boothe	Tennis Court 2
Timothy Baker	Tennis Court 1
Timothy Baker	Tennis Court 2
Tim Rownam	Tennis Court 1
Tim Rownam	Tennis Court 2
Tracy Smith	Tennis Court 1
Tracy Smith	Tennis Court 2

(46 rows)

24.- Show bookings on the day of 2012-09-14 which cost to the customer (or guest) more than $30. Remember that guests have different costs than customers (the listed costs are per hour 'slot'), and the guest user always has 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_WS(' ', c.firstname, c.lastname) AS customer, f.name AS
facility,
       CASE
        WHEN c.id=0
            THEN
                trunc((cast(f.guest_cost as numeric)*b.nhours),2)
            ELSE
                trunc((cast(f.cust_cost as numeric)*b.nhours),2)
            END AS cost

FROM bookings."BOOKINGS" AS b, bookings."FACILITIES" AS f,
bookings."CUSTOMERS" AS c
WHERE date(b.start_datetime)='2012-09-14'
  AND
      CASE
        WHEN c.id=0
            THEN
                (cast(f.guest_cost as numeric)*b.nhours) > 30
            ELSE
                (cast(f.cust_cost as numeric)*b.nhours) > 30
        END
  AND b.fac_id=f.id
  AND c.id=b.cust_id
ORDER BY cost DESC
```

customer	facility	cost
GUEST GUEST	Squash Court	700.00
GUEST GUEST	Squash Court	350.00
GUEST GUEST	Squash Court	350.00
GUEST GUEST	Massage Room 2	320.00
GUEST GUEST	Massage Room 1	160.00
GUEST GUEST	Massage Room 1	160.00
GUEST GUEST	Massage Room 1	160.00
GUEST GUEST	Tennis Court 2	150.00
Jemima Farrell	Massage Room 1	140.00
GUEST GUEST	Tennis Court 2	75.00
GUEST GUEST	Tennis Court 1	75.00
GUEST GUEST	Tennis Court 1	75.00
David Pinker	Squash Court	70.00
Jack Smith	Massage Room 1	70.00
Jemima Farrell	Massage Room 1	70.00
Ponder Stibbons	Massage Room 1	70.00
Burton Tracy	Massage Room 1	70.00
Matthew Genting	Massage Room 1	70.00
Florence Bader	Massage Room 2	70.00
Anne Baker	Squash Court	70.00
Timothy Baker	Squash Court	70.00
Anne Baker	Squash Court	70.00

(22 rows)

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 c.id, CONCAT_WS(' ', c.firstname, c.lastname) AS customer,
CONCAT_WS(' ',l.rid, l.firstname, l.lastname) AS recommend_by
FROM bookings."CUSTOMERS" AS c,
     (SELECT DISTINCT c.id AS rid, c.firstname, c.lastname
        FROM bookings."CUSTOMERS" AS c, (SELECT recommended_id AS rid
                                         FROM bookings."CUSTOMERS") AS x
        WHERE c.id = x.rid) AS l
WHERE c.recommended_id=l.rid
UNION
SELECT c.id, CONCAT_WS(' ', c.firstname, c.lastname) AS customer, '' AS
recommend by
FROM bookings."CUSTOMERS" AS c
WHERE recommended_id IS NULL
ORDER BY customer;
```

id	customer	recommended
21	Anna Mackenzie	1 Darren Smith
12	Anne Baker	9 Ponder Stibbons
6	Burton Tracy	
10	Charles Owen	1 Darren Smith
37	Darren Smith	
1	Darren Smith	
28	David Farrell	
11	David Jones	4 Janice Joplette
17	David Pinker	13 Jemima Farrell
26	Douglas Jones	11 David Jones
36	Erica Crumpet	2 Tracy Smith
15	Florence Bader	9 Ponder Stibbons
5	Gerald Butters	1 Darren Smith
0	GUEST GUEST	
27	Henrietta Rumney	20 Matthew Genting
29	Henry Worthington-Smyth	2 Tracy Smith
33	Hyacinth Tupperware	
14	Jack Smith	1 Darren Smith
4	Janice Joplette	1 Darren Smith
13	Jemima Farrell	
22	Joan Coplin	16 Timothy Baker
35	John Hunt	30 Millicent Purview
20	Matthew Genting	5 Gerald Butters
30	Millicent Purview	2 Tracy Smith
7	Nancy Dare	4 Janice Joplette
9	Ponder Stibbons	6 Burton Tracy
24	Ramnaresh Sarwin	15 Florence Bader
8	Tim Boothe	3 Tim Rownam
16	Timothy Baker	13 Jemima Farrell
3	Tim Rownam	
2	Tracy Smith	

(31 rows)

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."BOOKINGS"
WHERE date(start_datetime) > '2012-08-31'
  AND date(start_datetime) < '2012-10-1'
GROUP BY fac_id
ORDER BY thours;
```

```
 fac_id | thours
--------+--------
      5 |    122
      3 |    422
      7 |    426
      8 |    471
      6 |    540
      2 |    570
      1 |    588
      0 |    591
      4 |    648
(9 rows)
```

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."BOOKINGS"
WHERE extract(YEAR FROM start_datetime)='2012'
GROUP BY month, fac_id
ORDER BY fac_id, month;
```

```
 fac_id | month | sum
--------+-------+-----
      0 |     7 | 270
      0 |     8 | 459
      0 |     9 | 591
      1 |     7 | 207
      1 |     8 | 483
      1 |     9 | 588
      2 |     7 | 180
      2 |     8 | 459
      2 |     9 | 570
      3 |     7 | 104
      3 |     8 | 304
      3 |     9 | 422
      4 |     7 | 264
      4 |     8 | 492
      4 |     9 | 648
      5 |     7 |  24
      5 |     8 |  82
      5 |     9 | 122
      6 |     7 | 164
      6 |     8 | 400
      6 |     9 | 540
      7 |     7 | 156
      7 |     8 | 326
      7 |     9 | 426
      8 |     7 | 117
      8 |     8 | 322
      8 |     9 | 471
(27 rows)
```

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

```
SELECT count(*) AS num FROM (SELECT DISTINCT cust_id
                             FROM bookings."BOOKINGS") AS a;
```

```
 num
-----
  30
(1 row)
```

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

```
SELECT *
FROM (SELECT fac_id, sum(nhours) as num
    FROM bookings."BOOKINGS"
    GROUP BY fac_id) AS a
WHERE a.num > 1000
ORDER BY a.fac_id;
```

```
 fac_id | num
--------+------
      0 | 1320
      1 | 1278
      2 | 1209
      4 | 1404
      6 | 1104
(5 rows)
```

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

```
SELECT a.name, cast(sum(cost) as money) AS revenue
FROM (SELECT name, cust_id,
        CASE
           WHEN cust_id=0
             THEN
               (cast(f.guest_cost as numeric)*b.nhours)
             ELSE
               (cast(f.cust_cost as numeric)*b.nhours)
             END AS cost
       FROM bookings."BOOKINGS" AS b, bookings."FACILITIES" AS f
       WHERE b.fac_id=f.id) AS a
GROUP BY name
ORDER BY revenue;
```

```
      name        |   revenue
------------------+-------------
 Table Tennis     |     $180.00
 Snooker Table    |     $240.00
 Pool Table       |     $270.00
 Tennis Court 1   |  $13,860.00
 Tennis Court 2   |  $14,310.00
 Massage Room 2   |  $15,810.00
 Badminton Court  |  $19,065.00
 Massage Room 1   |  $72,540.00
 Squash Court     | $134,680.00
(9 rows)
```

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 *
FROM (SELECT a.name, cast(sum(cost) as money) AS revenue
        FROM (SELECT name, cust_id,
            CASE
                WHEN cust_id=0 THEN
              (cast(f.guest_cost as numeric)*b.nhours)
                ELSE
              (cast(f.cust_cost as numeric)*b.nhours)
                END AS cost
      FROM bookings."BOOKINGS" AS b, bookings."FACILITIES" AS f
        WHERW b.fac_id=f.id) AS a
GROUP BY name
ORDER BY revenue) AS x
WHERE cast(x.revenue as numeric) < 1000;
```

```
     name       | revenue
---------------+---------
 Table Tennis   | $180.00
 Snooker Table  | $240.00
 Pool Table     | $270.00
(3 rows)
```

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

```
SELECT fac_id, sum(nhours) AS nhours
FROM bookings."BOOKINGS"
GROUP BY fac_id
ORDER BY nhours DESC LIMIT 1;
```

```
 fac_id | nhours
--------+--------
      4 |   1404
(1 row)
```

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. (You must look for information on the Internet).

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

```
SELECT c.lastname, c.firstname, b.cust_id AS id, b.start_datetime
FROM (SELECT a.cust_id, min(a.start_datetime) AS start_datetime
    FROM(SELECT *
        FROM bookings."BOOKINGS"
        WHERE date(start_datetime) >= '2012-09-01'
        ORDER BY cust_id, start_datetime) AS a
            GROUP BY a.cust_id) AS b,
    bookings."CUSTOMERS" AS c
WHERE b.cust_id=c.id;
```

lastname	firstname	id	start_datetime
GUEST	GUEST	0	2012-09-01 08:00:00
Smith	Darren	1	2012-09-01 09:00:00
Smith	Tracy	2	2012-09-01 11:30:00
Rownam	Tim	3	2012-09-01 16:00:00
Joplette	Janice	4	2012-09-01 15:00:00
Butters	Gerald	5	2012-09-02 12:30:00
Tracy	Burton	6	2012-09-01 15:00:00
Dare	Nancy	7	2012-09-01 12:30:00
Boothe	Tim	8	2012-09-01 08:30:00
Stibbons	Ponder	9	2012-09-01 11:00:00
Owen	Charles	10	2012-09-01 11:00:00
Jones	David	11	2012-09-01 09:30:00
Baker	Anne	12	2012-09-01 14:30:00
Farrell	Jemima	13	2012-09-01 09:30:00
Smith	Jack	14	2012-09-01 11:00:00
Bader	Florence	15	2012-09-01 10:30:00
Baker	Timothy	16	2012-09-01 15:00:00
Pinker	David	17	2012-09-01 08:30:00
Genting	Matthew	20	2012-09-01 18:00:00
Mackenzie	Anna	21	2012-09-01 08:30:00
Coplin	Joan	22	2012-09-02 11:30:00
Sarwin	Ramnaresh	24	2012-09-04 11:00:00
Jones	Douglas	26	2012-09-08 13:00:00
Rumney	Henrietta	27	2012-09-16 13:30:00
Farrell	David	28	2012-09-18 09:00:00
Worthington-Smyth	Henry	29	2012-09-19 09:30:00
Purview	Millicent	30	2012-09-19 11:30:00
Tupperware	Hyacinth	33	2012-09-20 08:00:00
Hunt	John	35	2012-09-23 14:00:00
Crumpet	Erica	36	2012-09-27 11:30:00

(30 filas)

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

```
SELECT count(*) OVER (), firstname, lastname
FROM bookings."CUSTOMERS"
```

```
 count | firstname |     lastname
-------+-----------+-------------------
    31 | GUEST     | GUEST
    31 | Darren    | Smith
    31 | Tracy     | Smith
    31 | Tim       | Rownam
    31 | Janice    | Joplette
    31 | Gerald    | Butters
    31 | Burton    | Tracy
    31 | Nancy     | Dare
    31 | Tim       | Boothe
    31 | Ponder    | Stibbons
    31 | Charles   | Owen
    31 | David     | Jones
    31 | Anne      | Baker
    31 | Jemima    | Farrell
    31 | Jack      | Smith
    31 | Florence  | Bader
    31 | Timothy   | Baker
    31 | David     | Pinker
    31 | Matthew   | Genting
    31 | Anna      | Mackenzie
    31 | Joan      | Coplin
    31 | Ramnaresh | Sarwin
    31 | Douglas   | Jones
    31 | Henrietta | Rumney
    31 | David     | Farrell
    31 | Henry     | Worthington-Smyth
    31 | Millicent | Purview
    31 | Hyacinth  | Tupperware
    31 | John      | Hunt
    31 | Erica     | Crumpet
    31 | Darren    | Smith
(31 rows)
```

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 sum(0+1) OVER (ORDER BY id) AS id, firstname, lastname
FROM bookings."CUSTOMERS";
```

row_number	firstname	lastname
1	GUEST	GUEST
2	Darren	Smith
3	Tracy	Smith
4	Tim	Rownam
5	Janice	Joplette
6	Gerald	Butters
7	Burton	Tracy
8	Nancy	Dare
9	Tim	Boothe
10	Ponder	Stibbons
11	Charles	Owen
12	David	Jones
13	Anne	Baker
14	Jemima	Farrell
15	Jack	Smith
16	Florence	Bader
17	Timothy	Baker
18	David	Pinker
19	Matthew	Genting
20	Anna	Mackenzie
21	Joan	Coplin
22	Ramnaresh	Sarwin
23	Douglas	Jones
24	Henrietta	Rumney
25	David	Farrell
26	Henry	Worthington-Smyth
27	Millicent	Purview
28	Hyacinth	Tupperware
29	John	Hunt
30	Erica	Crumpet
31	Darren	Smith

(31 rows)

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

```
SELECT DISTINCT fac_id, sum(nhours) OVER (partition by fac_id) AS total
FROM bookings."BOOKINGS"
ORDER BY total DESC
LIMIT 1;
```

fac_id	total
4	1404

(1 row)

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 x.name, sum(0+1) OVER (ORDER BY x.revenue DESC)
FROM (SELECT a.name, CAST(sum(cost) AS money) AS revenue
    FROM (SELECT name, cust_id,
       CASE
          WHEN cust_id=0
              THEN
                (cast(f.guest_cost as numeric)*b.nhours)
              ELSE
                (cast(f.cust_cost as numeric)*b.nhours)
              END AS cost
      FROM bookings."BOOKINGS" AS b, bookings."FACILITIES" AS f
    WHERE b.fac_id=f.id) AS a
GROUP BY name
ORDER BY revenue DESC) AS x LIMIT 3;
```

```
       name        | rank
-----------------+------
 Squash Court     |    1
 Massage Room 1   |    2
 Badminton Court  |    3
(3 rows)
```

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: 10, name: 'Spa 2', cust_cost: 20, guest_cost: 30, purchase_cost: 100000, maintenance_cost: 800.
- id: 11, name: 'Squash Court 2', cust_cost: 3.5, guest_cost: 17.5, purchase_cost: 5000, maintenance_cost: 80.

```
INSERT INTO bookings."FACILITIES" (id, name, cust_cost, guest_cost,
purchase_cost, maintenance_cost) VALUES (9, 'Spa', '20', '30', '100000',
'8000');
INSERT INTO bookings."FACILITIES" (id, name, cust_cost, guest_cost,
purchase_cost, maintenance_cost) VALUES (10, 'Spa 2', '20', '30',
'100000', '8000');
INSERT INTO bookings."FACILITIES" (id, name, cust_cost, guest_cost,
purchase_cost, maintenance_cost) VALUES (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:

```
INSERT INTO bookings."FACILITIES" (id, name, cust_cost, guest_cost,
purchase_cost, maintenance_cost)
VALUES ((select id+1 from bookings."FACILITIES" order by id desc limit 1),
'Squash Court 2', '3.5', '17.5', '5000', '80');
```

- name: 'Spa 3', cust_cost: 20, guest_cost: 30, purchase_cost: 100000, maintenance_cost: 800.

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.

```
UPDATE bookings."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 bookings."FACILITIES"
SET cust_cost  = '$6.00',
    guest_cost = '$30.00'
WHERE id = 1 or id = 0;
```

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 bookings."FACILITIES"
SET cust_cost  = (SELECT cust_cost*1.1
                    FROM bookings."FACILITIES"
                    WHERE name='Tennis Court 2'),
    guest_cost = (SELECT guest_cost*1.1
                    FROM bookings."FACILITIES"
                    WHERE name='Tennis Court 2')
WHERE id = 1;

SELECT *
FROM bookings."FACILITIES";
```

44.- Using pgdump, dump all the database (with inserts) into a text files.

45.- Delete all bookings.

```
drop table bookings."BOOKINGS";
```

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

```
DELETE
FROM bookings."CUSTOMERS"
WHERE id = 37;

SELECT *
FROM bookings."CUSTOMERS"
WHERE id=37;
```

47.- 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.

```
DELETE
FROM bookings."CUSTOMERS"
WHERE id NOT IN (SELECT DISTINCT cust_id
                 FROM bookings."BOOKINGS");
```