*-- 1. What are the identifiers of the customers (attribute cust\_no), for whom NO invoice has been*  
*-- issued. Give three answers: the first one using EXCEPT, the second one using NOT IN and the*  
*-- third one using NOT EXISTS.*  
  
select cust\_no  
from customers  
except  
select customer  
from invoices;  
  
select cust\_no  
from customers  
where cust\_no not in (select customer from invoices);  
  
select cust\_no  
from customers  
where not *exists*(select inv\_no from invoices where customer = customers.**cust\_no**);  
  
*-- 2. Which are the references, designations and unit prices of all the products, except the*  
*-- cheapest? Give four answers: the first one using the keyword ANY, the second one using the*  
*-- keywords EXCEPT and ALL, the third one using the keyword NOT IN and ALL, and the last one*  
*-- using NOT EXISTS and ALL.*  
  
select reference, designation, unit\_price  
from products  
where unit\_price > any (select unit\_price from products);  
  
select reference, designation, unit\_price  
from products  
except  
select reference, designation, unit\_price  
from products  
where unit\_price <= all (select unit\_price from products);  
  
select reference, designation, unit\_price  
from products  
where reference not in (select reference from products  
 where unit\_price <= all (select unit\_price from products));  
  
select reference, designation, unit\_price  
from products p1  
where not *exists* (select reference from products  
 where p1.**unit\_price** = unit\_price and  
 unit\_price <= all (select unit\_price from products  
 where unit\_price <= all (select unit\_price from products)));

*-- there must be a better query for the last option in your question. Would you mind sent me an email if there's any?*  
  
*-- 3. Which are the invoices identifiers for which both the technicians « Bonnaz » and « Mauras »*  
*-- have worked (give three answer: the first one using INTERSECT, the second one using IN and*  
*-- the third one using EXISTS)?*  
  
select invoice  
from interventions, technicians  
where tech\_no = technician and  
 name = 'Bonnaz'  
intersect  
select invoice  
from interventions, technicians  
where tech\_no = technician and  
 name = 'Mauras';  
  
select invoice  
from interventions, technicians  
where tech\_no = technician and  
 name = 'Bonnaz' and  
 invoice in (select invoice  
 from interventions, technicians  
 where tech\_no = technician and  
 name = 'Mauras');  
  
select invoice  
from interventions i1, technicians  
where tech\_no = technician and  
 name = 'Bonnaz' and  
 *exists*(select invoice  
 from interventions, technicians  
 where tech\_no = technician and  
 name = 'Mauras' and  
 invoice = i1.**invoice**);  
  
*-- 4. Give the list of ALL the references of all the products (no matter if they were replaced or*  
*-- not), along with the intervention numbers and the replaced quantity, when these products*  
*-- were actually replaced (if they were never replaced, then you should just obtain NULL).*  
  
select reference, intervention, qtity  
from products left outer join replacements r on products.reference = r.product;  
  
*-- 5. Give the total number of technicians.*  
  
select *count*(tech\_no)  
from technicians;  
  
*-- 6. Give the average unit price of a product. This new attribute will be called “Average”*  
  
select *avg*(unit\_price) as Average  
from products;  
  
*-- 7. List of all the invoices identifiers (sorted by ascending order) and the number of related*  
*-- interventions. Please rename the latter attribute as “number\_interv”.*  
  
select inv\_no, *count*(interv\_no) as number\_interv  
from invoices left outer join interventions on inv\_no = invoice  
group by inv\_no;  
  
*-- 8. References of the products which have been replaced at least twice (i.e. two times),*  
*-- together with the number of times they have been replaced.*  
  
select reference, number\_replaced  
from(select reference, *count*(intervention) number\_replaced  
 from products, replacements  
 where reference = product  
 group by reference) new  
where number\_replaced >= 2;  
  
*-- 9. For each intervention, give its identifier and the total amount for all the replaced parts in*  
*-- that intervention (take into account the unit price AND the quantity of the products*  
*-- replaced, for each intervention).*  
  
  
select interv\_no, *sum*(unit\_price) price, *sum*(qtity) qtity  
from interventions left join replacements r on interventions.interv\_no = r.intervention, products  
where reference = product  
group by interv\_no;  
  
  
*-- 10. Who is the customer (cust\_no and name) who had the highest number of*  
*-- interventions (provide a solution with a nested query)? Please also give the number of*  
*-- interventions for that customer.*  
  
select cust\_no, name, n\_interv  
from (select customer, *count*(interv\_no) n\_interv  
 from interventions, invoices  
 where invoice = inv\_no  
 group by customer) T1,  
 customers  
where T1.customer = cust\_no and  
 n\_interv >= all (select *count*(interv\_no)  
 from interventions, invoices  
 where invoice = inv\_no  
 group by customer);  
  
*-- 11. Total amount of the salary, for each invoice identifier. This amount is the salary of the*  
*-- technician (depending on his hourly rate and of the duration of the intervention), for each*  
*-- intervention corresponding to that invoice.*  
  
select invoice, *sum*(salary)  
from (select invoice, hour\_rate \* duration salary  
 from interventions, technicians  
 where technician = tech\_no) S  
group by invoice;  
  
  
*-- 12. Total price of each invoice. This price includes the price of the replaced places, and the*  
*-- salary of the technician (depending on his hourly rate and of the time spent), for each*  
*-- intervention depending on that invoice.*  
  
select S2.invoice, (S2.total\_salary + S3.total\_unit)  
from (select invoice, *sum*(salary) total\_salary  
 from (select invoice, hour\_rate \* duration salary  
 from interventions, technicians  
 where technician = tech\_no) S1  
 group by invoice) S2,  
 (select invoice, *sum*(unit\_price) total\_unit  
 from (select invoice, unit\_price  
 from interventions, replacements, products  
 where intervention = interv\_no and product = reference) S  
 group by invoice) S3  
where S2.invoice = S3.invoice;

1. Add the primary keys (underlined attributes) and foreign keys (add a # after the attribute name) on the relational model above.

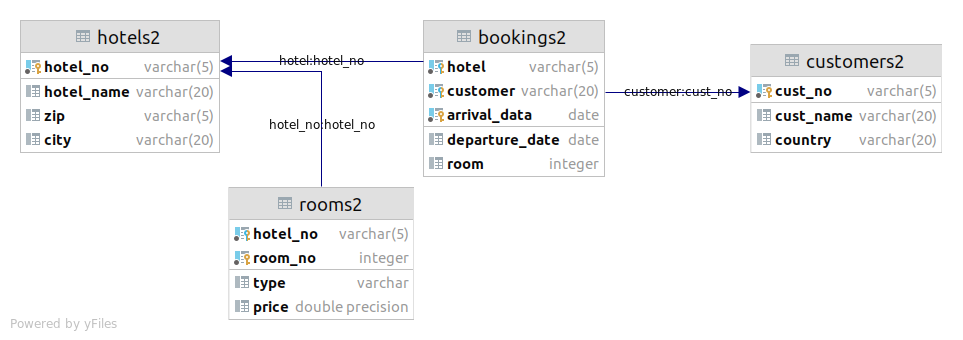
HOTELS(hotel\_no, hotel\_name, zip, city)  
ROOMS(hotel#, room\_no, type, price)  
BOOKINGS(hotel#, customer#, arrival\_data, departure\_date, room#)  
CUSTOMERS (cust\_no, cust\_name, country)

Up here I set (hotel, customer, arrival\_data) as primary key of BOOKINGS. That is, a customer can arrive at the same hotel multiple times, whether the same room or not.

Therefore, room doesn’t need (if not shouldn’t) to be set as part of primary key. However, as I create table BOOKINGS with room referencing to room\_no in ROOMS, an error raised tells that room\_no must be unique, though I think in reality it shouldn’t be compulsory: different hotels might have the same room names.

So I guess in BOOKINGS table, we can eliminate attribute room.

2. Draw a scheme of this relational database. In that scheme, every table will be represented by a rectangle and every referential integrity constraint will be represented by a link between two tables. You can add multiplicities as you would define them in an UML class diagram. No table should be completely isolated from the others.



3. Which are the hotels from Niort that hosted all the customers from Spain?

select hotel\_no  
from hotels2 h  
where city = 'Niort' and  
 not *exists* (  
 select cust\_no  
 from customers2  
 where country = 'Spain' and  
 cust\_no not in (  
 select customer  
 from bookings2  
 where hotel = h.**hotel\_no**  
)  
 );  
  
*-- select a hotel A such that it is in Niort*  
*-- query the list B of customers who've booked the hotel A*  
*-- query the list C of customers from Spain*  
*-- if there's any customer from C who is not in B*  
*-- then A is not in the result*

4. Customers who booked in all the hotels of Niort.

select cust\_no  
from customers2 c  
where not *exists* (  
 select hotel\_no  
 from hotels2  
 where city = 'Niort' and  
 hotel\_no not in (  
 select hotel  
 from bookings2  
 where customer = c.**cust\_no**  
)  
 );  
  
*-- + select the customer A:*  
*-- + query the list B of hotels that A has booked.*  
*-- + query the list C of all hotels in Niort.*  
*-- + if there's any hotel in C that is not in B*  
*-- + then A is not appear in the result*

5. Hotels that hosted all the customers from Switzerland.

select hotel\_no  
from hotels2 h  
where not *exists*(  
 select cust\_no  
 from customers2  
 where country = 'Switzerland' and  
 cust\_no not in (  
 select customer  
 from bookings2  
 where hotel = h.**hotel\_no**  
)  
 );  
  
*-- select a hotel A*  
*-- query the list B of customers who've booked hotel A*  
*-- query the list C of all the customers from Switzerland*  
*-- if there's any customers from C that is not in B*  
*-- then A is not in the result*