REQUÊTES MONOTABULAIRES

liste des clients dont l'age est < 20

```
select prenom
from client
where age < 20;
```

liste des clients dont l'age est compris entre 21 et 41
`distinct` **: suppression des doublons et tri**

```
select distinct prenom
from client
where age > 20 and age < 42;
```

liste des clients dont l'age est compris entre 21 et 41
`as` **: renommage de la colonne prenom**

```
select all prenom as personne, age * 3
from client
where age between 21 and 41;
```

$\{ t[nom] \mid t \in CLIENT \wedge t[age] > 20 \}$

```
select t.nom
from client as t
where t.age > 20
```

$\{ t[nom] \mid t \in CLIENT \wedge \exists\ c \in CMD \wedge c[nprod] = 13 \wedge c[nclt] = t[nclt] \}$

```
select t.nom
from client t, cmd c
where c.nprod = 13 and t.nclt = c.nclt
```

Attention aux mots clé : il les encadrer de double quotes

```
select "date"
from "join"
where not 'F';
```

liste des clients dont l'age est compris entre 20 et 60
`order by` **: tri des prenom en ordre croissant**

```
select prenom, nom
from client
where age between 20 and 60
order by prenom desc, nom desc;
```

liste des clients dont l'age est compris entre 20 et 60 et dont le prenom contient un 'a'

```
select prenom, nom
from client
where age between 20 and 60 and like '%a%';
```

liste des clients dont l'age est compris entre 20 et 60 et dont le prenom contient un 'a' en deuxième position.
`upper` **et** `lower` **permettent de modifier la casse des chaines de caracteres.**

```
select upper(prenom), lower(nom)
from client
where age between 20 and 60 and like '_a%';
```

liste des premières lettres des prénoms des clients, et affichage de leur nom en minuscule
`substr` **: prélever un morceau de chaine de caractères.**

```
select substr(prenom, 1, 1), lower(nom)
from client;
```

`reverse` **: inverser toutes les lettres d'une chaine**
`trim : trailing` **: supprimer en début de chaine**
`leading` **: supprimer en fin de chaine**
`both` **: supprimer en début et fin de chaine**

```
select reverse(prenom), reverse(trim(trailing ' ' from nom))
from client;
```

```
select reverse(leading 'e' from lower(prenom)), reverse(trim
(trailing ' ' from nom))
from client;
```

Affichage du prénom et de la taille de la chaine de chaque client

```
select prenom, lenght(prenom)
from client;
```

Affichage du prénom et du nom séparés d'un espace de chaque client, ceci dans une colonne 'personne'

```
select prenom || ' ' || nom as personne
from client;
```

Liste de tous les clients sont le nom ressemble à 'Jean'

```
select *
from client
where soundex(prenom) = soundex('jon');
```

Affichage du code, en référence à l'algorithme de soundex, correspondant à chaque prénom (l'algorithme par défaut traite une phénétique anglaise)

```
select prenom, soundex(prenom)
from client;
```

Affichage de la date, l'heure, les deux simultanément, et l'année

```
select current_date;
select current time;
select current_timestamp;
select extract (year);
```

Compter les clients dont l'age est > 20

```
select count(*) from client
where age > 20;
```

Affichage en cinq colonne :
- nombre de clients dont le nom est distinct
- l'age minimum
- le dernier prénom en ordre croissant alphabétique
- la moyenne des ages
- la somme des ages

```
select count(distinct nom), min(age), max(prenom), avg(age), sum
(age)
from client;
```

Affichage du nombre de clients, la moyenne des ages, l'age minimum et l'age maximum, avec une création de groupes
`group by` **: de groupe**

```
select prenom, count(*), avg(age), min(age), max(age)
from client
group by prenom;
```

Affichage du nombre de clients, la moyenne des ages, l'age minimum et l'age maximum, avec une création de groupes, de tous les clients dont l'age est au moins égal à 20 ans ou non précisé, et sur le groupement, ne retenir que ceux dont l'age est supérieur à 30 ans

```
select prenom, count(*), avg(age), min(age), max(age)
from client
where age >= 20 or age is null
group by prenom
having (age > 30)
order by 3 desc
```

Liste des clients dont l'age est 20, 33 ou 42 ans

```
select * from client
where age in (20, 33, 42);
```

Liste du nom de tous les clients. Si l'age n'est pas précisé, afficher '0'

```
select nom, coalesce(age, 0)
from client;
```

Liste du nom de tous les clients. Dans une seconde colonne, préciser :
- si l'age est < 20, afficher 'jeune'
- si l'age est compris entre 20 et 30, afficher 'sympa'
- sinon, afficher 'vieux'

```
select nom, case when age < 20 then 'jeune'
                 when age between 20 and 30 then 'sympa'
                 else 'vieux'
                 end
from client;
```

REQUETES MULTITABULAIRES

Liste des produits commandés par Monsieur Dupont

```
select nprod
from client, cmd
where nom = 'Dupont' and client.nclt = cmd.nclt;
```

Liste des produits et leur couleur commandés par Monsieur Dupont

```
select cmd.nprod, couleur
from client, cmd
where nom = 'Dupont' and client.nclt = cmd.nclt and cmd.nprod =
prod.nprod;
```

Liste des produits et leur couleur commandés par Monsieur Dupont, avec renommage des tables de différentes façons

```
select p.nprod, p.couleur
from client t, cmd c, prod p
where nom = 'Dupont' and client.nclt = cmd.nclt and cmd.nprod =
prod.nprod;
```

```
select p.nprod, p.couleur
from client as t, cmd as c, prod as p
where t.nom = 'Dupont' and c.nclt = c.nclt and c.nprod =
p.nprod;
```

```
select prod.nprod, couleur
from client, cmd, prod
where nom = 'Dupont' and client.nclt = cnd.nclt and cmd.nprod =
prod.nprod;
```

Liste des clients ayant commandé le même produit (sans doublons)

```
select distinct c1.nclt, c2.nclt
from cmd c1, cmd c2
where c1.nprod = c2.nprod and c1.nclt < c2.nclt
order by 2;
```

Liste des produits commandés par Monsieur Dupont, avec une jointure `join` : jointure classique

```
select nprod
from client join cmd on client.nclt = cmd.nclt
where nom - 'Dupont';

select nprod
from client join cmd on client.nclt = cmd.nclt and nom -
'Dupont';

select nprod
from client join cmd on client.nclt = cmd.nclt and nom -
'Dupont'
where client.nclt = cmd.nprod;

select nprod
from client join cmd on (client.nclt = cmd.nclt and client.nclt
= cmd.nprod)
where nom - 'Dupont';

select c.nprod
from client t join cmd c on (t.nclt = c.nclt and t.nclt =
c.nprod)
where t.nom - 'Dupont';

select t.nclt, t.nom, count(distinct c.nprod)
from client t join cmd c on (t.nclt = c.nclt)
group by t.nclt, t,nom
```

Liste des produits commandés, et nombre de produits commandés par chaque client
`right | left | full join` **: jointure externe**

```
select t.nclt, t.nom, count(nprod)
from client t left outer join cmd c on (t.nclt = c.nclt)
group by t.nclt, t,nom

select t.nclt, c.nclt, c.nprod, prod.nprod
from client t left outer join cmd c on t.nclt = c.nclt right
outer join prod on c.nprod = prod.nprod;

select t.nclt, t.nom, c.nclt, c.nprod, prod.nprod
from client t left outer join cmd c on t.nclt = c.nclt full
outer join prod on c.nprod = prod.nprod;
```

`natural join` **: jointure naturelle**
La jointure naturelle enlève les colonnes en double.

Liste des clients de plus de 20 ans ayant passé commande

```
select nom, nprod
from client natural join cmd
where age > 20;
```

Liste des produits commandés par Monsieur Dupont

```
select nclt, t.nom, c.nprod
from client t join cmd c on t.nclt=c.nclt
where t.nom = 'DUPONT';

select nclt, t.nom, c.nprod
from client t natural join cmd c
where t.nom = 'DUPONT';
```

Différentes jointures possibles ...

```
select count(*)
from client natural join prod;

select count(*)
from (select nclt as nprod from client) natural join prod;

select count(*)
from (select nclt as nprod, nom as couleur from client) natural
join prod;

select *
from client t natural left outer join cmd c natural full outer
join prod;
```

`cross join` **: produit cartésien**

`union, intersect, expect` (`minus`) **: union, intersection, soustraction.**
L'union élimine les doublons par défaut. Il faut rajouter `all` **pour ne pas les éliminer.**

Liste des clients ayant moins de 20 ans ou plus de 40 ans

```
select *
from client where age < 20
union
select * from client where age > 40
order by 4 desc
```

Liste des clients ayant plus de 20 ans et moins de 40 ans

```
select *
from client where age > 20
expect
select * from client where age < 40
order by 4 desc
```

POSTGRESQL implémente la différence avec `expect` et ORACLE avec `minus`. SQL SERVER n'implémente pas la différence, on l'implémente alors avec une jointure externe.

SOUS REQUETES

MySQL Server ne supportait pas les sous requêtes, mais apparemment aujourd'hui, c'est ok.

Moyenne d'âge des clients

```
select avg(age)
from client;
```

Noms des clients ayant un âge supérieur à la moyenne d'age de tous les clients

```
select nclt, nom
from client
where age > (select avg(age)
             from client);
```

Liste des clients et ayant commandé un produit dans la même viile et dont l'age est supérieur à la moyenne d'age de tous les clients, et nombre de produits commandés

```
select l1.nclt, l1.nom, (select count(nprod)
                         from cmd
                         where cmd.nclt=l1.nclt)
from client l1
where l1.age > (select avg(l2.age)
                from client l2
                where l2.adresse - l1.adresse)
```

Même requête que la précédente, mais sans sous requête

```
select l1.nclt, l1.nom, count(distinct c.nprod)
from client l1 join client l2 on l1.adresse = l2.adresse join
cmd c on l1.nclt = c.nclt
group by l1.nclt, l1.nom
having avg(l1.age) > avg(l2.age)

select l1.nclt, l1.nom, count(distinct c.nprod)
from client l1 join client l2 on l1.adresse = l2.adresse join
cmd c on l1.nclt = c.nclt
group by l1.nclt, l1.nom, l1.age
having l1.age > avg(l2.age)
```

Liste des clients qui ont passé commande

```
select *
from client t
where exists (select *
              from cmd c
              where c.nclt=t.nclt)
```

Le client le plus jeune

```
select *
from client
where age <= all (select age from client)
```

Tous les clients sauf le plus jeune

```
select *
from client
where age <= any (select age from client)
```

Liste des clients triée par les âges et numérotée

```
select (select count(*) from client l2
        where l2.age < l1.age)
       l1.nom, l1.age
from client l1
order by 3;
```

Liste des clients triée par les âges et numérotée, en une seule requête

```
select count(*), l1.nom, l1.age
from client l1 join client l2 on l2.age < l1.age
group by l1.age, l1.nom
order by 3;

select count(l2.distinct age), l1.nom, l1.age
from client l1 join client l2 on l2.age < l1.age
group by l1.age, l1.nom
order by 3;

select count(l2.age)+1, l1.nom, l1.age
from client l1 join client l2 on l2.age < l1.age
group by l1.age, l1.nom
order by 3;
```

Liste des clients triée par les âges et numérotée, on n'affiche que les positions 2, 4 et 8

```
select count(l2.age)+1, l1.nom, l1.age
from client l1 join client l2 on l2.age < l1.age
group by l1.age, l1.nom
having count(l2.age)+1 in (2,4,8)
order by 3;
```

Liste des clients ayant commandé exactement 5 produits différents avec le total des quantités commandées

```
select nclt, sum(qte)
from cmd
group by nclt
having count(distinct nprod) = 5
```

Liste des clients ayant commandé 5 produits rouges différents avec le total des quantités commandées

```
select nclt, sum(qte)
from cmd
where couleur = 'rouge'
group by nclt
having count(distinct nprod) = 5
```

INSERTION

Insertion d'un nouveau tuple dans la table client

```
insert into client
values (12, 'Dehak', 'Reda', 29, 'Paris');

insert into client (age, prenom, nom, nclt, adresse)
values (29, 'Reda', 'Dehak', 12, 'Paris');

insert into client (nclt, nom)
values (12, 'Dehak');
```

Création d'une table n_client_parisien et ajout des tuples de la table client dont l'adresse est Paris

```
create table n_client_parisien
(
  nclt int primary key,
  nom  varchar(20),
  age  int default 20
);

insert into n_client_parisien(nclt, nom)
(
  select nclt, cast(nom as varchar(20))
  from client
  where lower(adresse) = 'paris'
);
```

SUPPRESSION

Suppression d'un tuple dans la table n_client_parisien

```
delete from n_client_parision
where nclt - 5
```

Suppression de tous les tuples de la table n_client_parisien (la table existe toujours)

```
delete from n_client_parisien;
```

Suppression de la table n_client_parisien (la table n'existe plus)

```
drop table n_client_parisien;
```

Suppression des clients ayant le même nom

```
delete from client c1
where exists(select *
             from client c2
             where c1.nclt <> c2.nclt and c1.nom - c2.nom);
```

MODIFICATION

Modification de l'age de tous les tuples : on met tout le monde à 30 ans

```
update n_clt_parisien
set age = 30;
```

Modification de l'age et du nom des tuples dont le numéro de client est 1, 4 et 8

```
update n_clt_parisien
set age = 20 and nom = 'toto'
where nclt in (1, 4, 8);
```

Modification de l'age du client numéro 3 : il a deux ans de plus

```
update n_clt_parisien
set age = age + 2
where nclt = 3;
```

VUES (tables virtuelles)

Objectifs :
- **masquer la complexité d'un schéma relationnel**
- **simplifier les requêtes SELECT complexes**
- **Préserver la confidentialité des données**
- **Contribuer à la non redondance des données**

```
create view client_parisien(nclt, nom, age, adresse)
            as (select nclt, nom, age, adresse
            from client
            where lower(adresse = 'paris'));
```

Toute modification de la table client se répercute sur la vue client_parisien

```
insert into client_parisien (nclt, nom, adresse)
values (765, 'group_a', 'paris');

insert into client_parisien (nclt, nom, age, adresse)
values (765, 'group_b', 45, 'paris');

insert into client_parisien (nclt, nom, adresse)
values (770, 'group_a', 'lyon');
```

Modification avec option (force ici a n'insérer que des clients dont l'adresse est Paris)

```
drop table client_parisien;

create view client_parisien(nclt, nom, age, adresse)
            as (select nclt, nom, age, adresse
            from client
            where lower(adresse = 'paris'))
            with check_option;

insert into client_parisien (nclt, nom, adresse)
values (765, 'group_a', 'lyon');
```

La requête suivante n'est pas valide car on tente d'insérer un client qui habite Lyon

```
insert into client_parisien (nclt, nom, adresse)
values (770, 'group_a', 'lyon');
```

CONTRÔLE D'ACCÈS

```
grant {all privileges | liste_privileges} on object
to {public | list_users}
[with grant option ];

create table group_a
(
  nclt serial primary key,
  nom varchar(20),
  age int default 20
);

insert into group_a (nom, age)
values ('reda', 29);
```

On autorise User2 à faire un `select` sur la table group_a

```
grant select on group_a
to User2;
```

On autorise User2 à faire un `select` sur la table group_a et à donner des droits à un autre utilisateur

```
grant select on group_a
to User2
with grant option;
```

Suppression de droits

```
revoke [grant option for ] {all privileges | liste_privileges }
on object from {public | list_users };

revoke grant option for select on group_a from User2;
```

TRANSACTION

```
grant select on group_a to User2;

begin;

insert into groupe_a (nom, age)
values ('group_b', 46);

rollback;

end;
```

TRIGGER

```
create trigger <name>
{before | after | instead of }
{insert | delete update [of liste_colonnes ]}
on <relation>
[order <liste_colonnes> ]
[referencing old [as ] anc_val
             | new [as ] nouv_val]
             | old table [as ] anc_tab
             | new table [as ] nouv_tab ]
[for each {row | statement }]
<code_trigger>;

create table group_a
(
  nclt      int primary key,
  nom       varchar(20),
  age       int
);

create table h_group_a
(
  action    varchar(20),
  ancnclt   int,
  nouvnclt int,
  jour      date
);

insert into group_a (nclt, nom, age)
values (1, 'toto', 13);

create trigger t_group_a
after insert on group_a
for each row
begin
  insert into h_group_s(action, nouvclt)
  values ('INSERT', :new.nclt)
end;
```